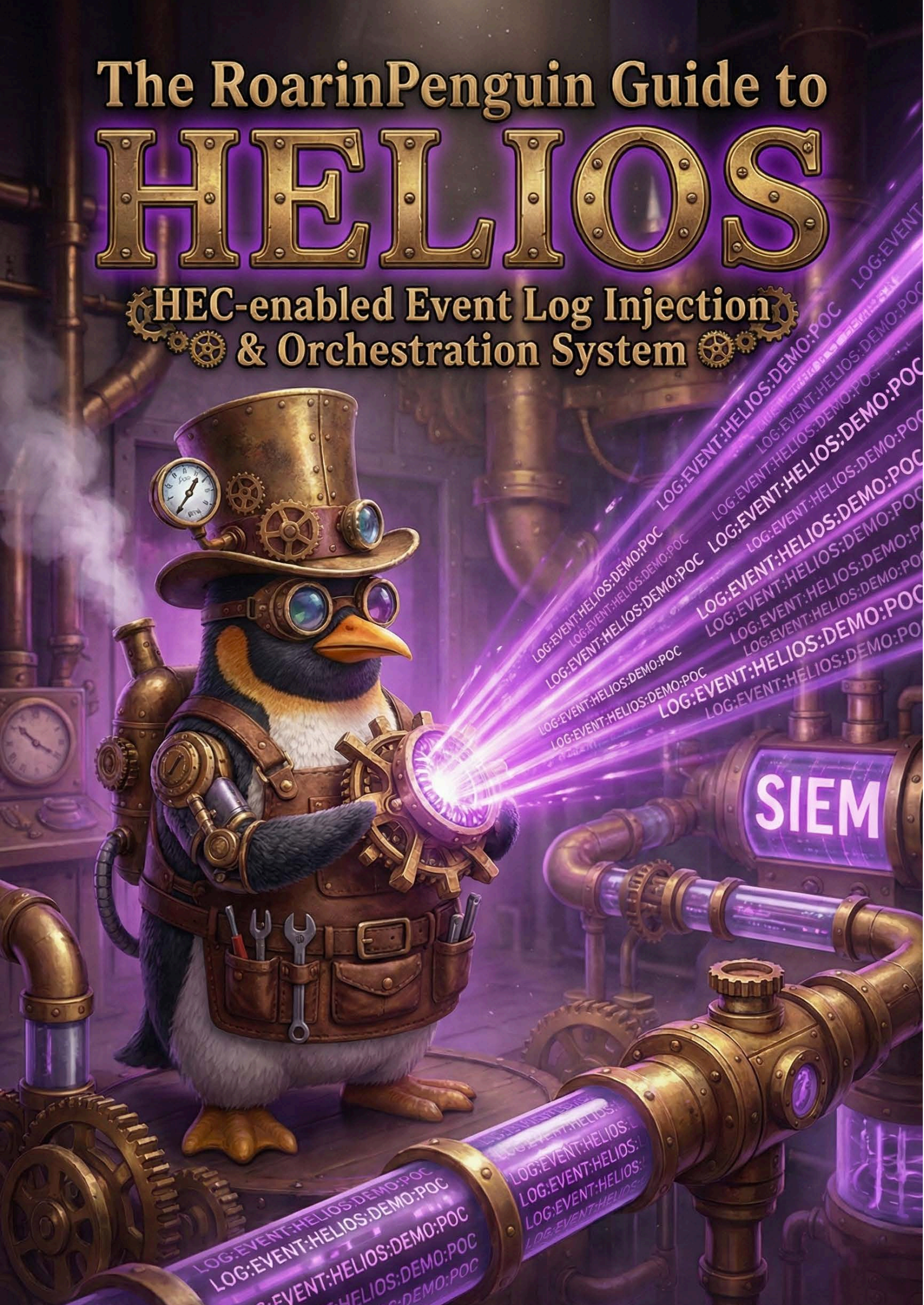
The RoarinPenguin Guide to HELIOS

HEC-enabled Event Log Injection & Orchestration System



The RoarinPenguin Guide to HELIOS

(**H**EC-enabled **E**vent **L**og **I**nject & **O**rchestration **S**ystem)

VERSION 1.2 - FEBRUARY 2026

by RoarinPenguin

INTRODUCTION

HELIOS is an OpenSource and Free (as in beer) software aimed to simplify the generation of realistic synthetic logs. Available for the galaxy on GitHub at the address https://github.com/natesmalley/jarvis_coding/ - it is a collaborative effort to enable SentinelOne people and *aficionados* to effectively deliver realistic demos and proof of concept based on plausible logs and scenarios with the SentinelOne Singularity Platform.

This guide is devoted to illustrate the usage of HELIOS and is based on use cases, to increase the effectiveness in using the tool.

Feel free to provide your feedback, ideas to enrich this, and corrections to roarinpenguin@sentinelone.com a.k.a. the RoarinPenguin.

Enjoy the ride!

NOTABENE

Although this book contains information in the context of SentinelOne, RoarinPenguin guides are a personal initiative totally coordinated, authored and supported by me… and contributed by several experts and volunteers. As such, it is not an official SentinelOne publication and cannot be supported by SentinelOne officially.

TABLE OF CONTENTS

GETTING HELIOS

Installing HELIOS is surprisingly easy, as long as you have a minimal acquaintance with Docker and containers.

The tool can be installed on your computer's main OS, or in a VM running in a virtualized system like a hypervisor.

This guide does not make any distinction on the OS platform used, as long as Docker service is installed and configured. On some systems, `docker compose` command is not installed as part of the main Docker service and requires an additional installation. For example, in a standard Ubuntu OS installation it might require the installation of the additional command `docker-compose` using the command:

```
sudo apt-get install docker-compose
```

The first operation before installing HELIOS is to transfer the software on your target system.

Depending on your experience with github, this can be accomplished in two ways:

- using git command to *clone* the repository on your system. The advantage in doing this is that it's much easier to keep the software updated when corrections are made or new versions are released.
  To proceed with this approach you need to have git software installed on your target system, then use the command:
  ```
  git clone https://github.com/natesmalley/jarvis_coding.git
  ```

- downloading the software directly from the web interface of the repository. Click on the link https://github.com/natesmalley/jarvis_coding and from the <> Code dropdown menu select **Download ZIP**.

Once transferred and/or decompressed the code, launch a terminal accessing the decompressed directory (e.g. `cd jarvis_coding`).

Then, proceed with a very important action that needs to be taken only once at first time setup: copy the provided file '`.env copy`' into `.env`. This will create an environment file for hard coding some variables, should you need to.

Thus, you can proceed to build and start the HELIOS services with the command:

```
docker compose up -d --build
```

The building process will take a bit more the first time, while it should be very quick after the first execution. For reference, on my system it took about 100 seconds to start.

NOTABENE: because HELIOS uses port tcp 9002, in case this is already in use on your system you might get an error on the docker compose command like the one reported here below:

```
 ⠸ Container jarvis-frontend          St...                              0.6s
Error response from daemon: failed to set up container networking: driver failed programming external connectivity on endpoint jarvis-frontend (1883c3764b41dfc5b1225e0bcb9e3771c080a84a6c88711ff53eec0b10118120): Bind for 0.0.0.0:9001 failed: port is already allocated
```

If this is the case, you can simply edit the file docker-compose.yaml, identify the lines:

```
    ports:
```

```
      - "9002:8000"
```

and change the port 9002 into an available port on your system.

Then, issue the command again and this time it will take approx 10 seconds to be ready, finishing with the following output:

```
✔ jarvis_coding-api          Built                    0.0s
✔ jarvis_coding-frontend     Built                    0.0s
✔ Container jarvis-api       Running                  0.0s
✔ Container jarvis-frontend  Started                  0.2s
```

Congratulations, you have successfully installed HELIOS!

Starting and stopping HELIOS

If you reboot your machine, HELIOS containers will restart if your Docker environment is configured to start as a service on boot.

If it is not the case, you need to manually launch the docker system and HELIOS will automatically start again.

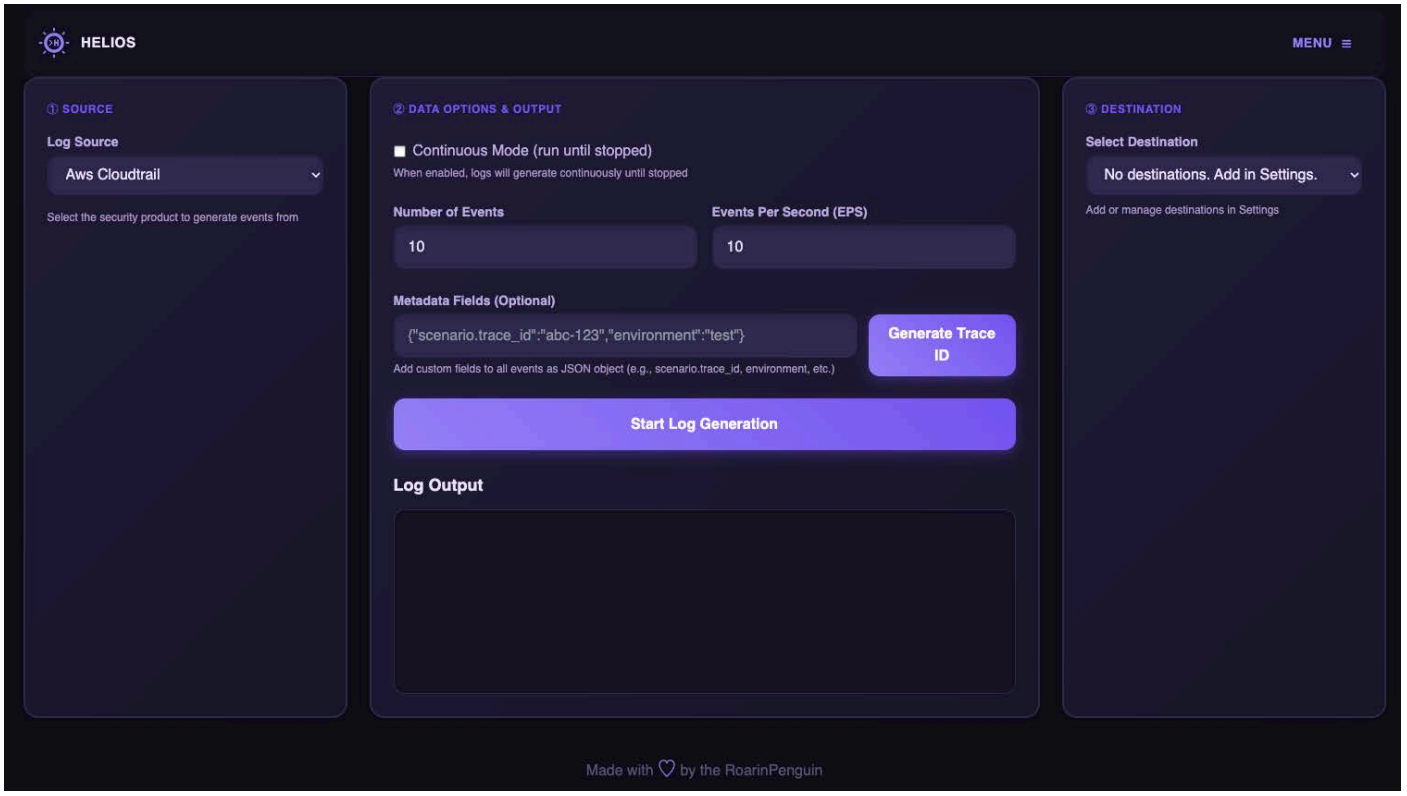
Should you need to stop HELIOS on your system, from the `jarvis_coding` directory use the command: `docker compose down`

To start it again, from the same directory use the command `docker compose up -d`

USING HELIOS

To access to HELIOS UI, point your browser to the URL
`http://<IP address of the machine where HELIOS is installed>:9002`
(or the port you configured in case 9002 was in use).

The beautifully purple UI will appear as shown below.



The first thing to do is to configure a destination for your generated logs.

One useful feature is to Generate (or manually set) a Trace ID, because it will ease later retrieving the set of logs in your AI SIEM Account by searching that field (see later in this document for more information).

To proceed, click on 

and then on **Settings**.

HELIOS supports two types of destinations: Syslog and HTTP Event Collector (HEC)

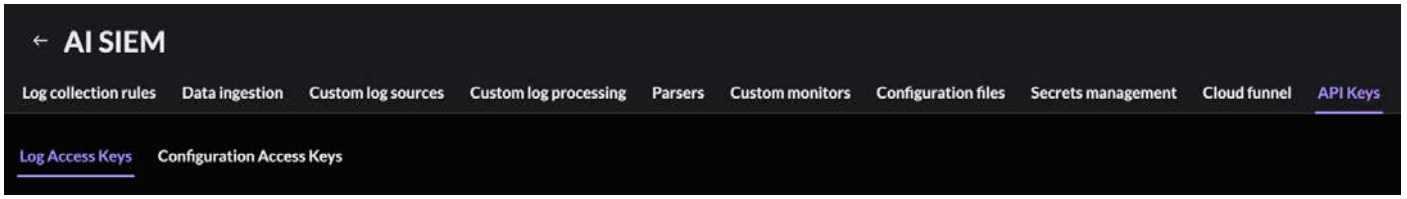
The choice between the two log streaming techniques depends on what you have as the receiving destination for the logs.

For example, if you are aiming to stream logs to AI SIEM please select HEC as the destination.

In the UI illustrated below, you can set a Name to identify your setup in the main UI drop down menu for destinations. Check that HEC URL matches your console's region (for example, if you are using a European console the URL will change to https://ingest.eu1.sentinelone.net).

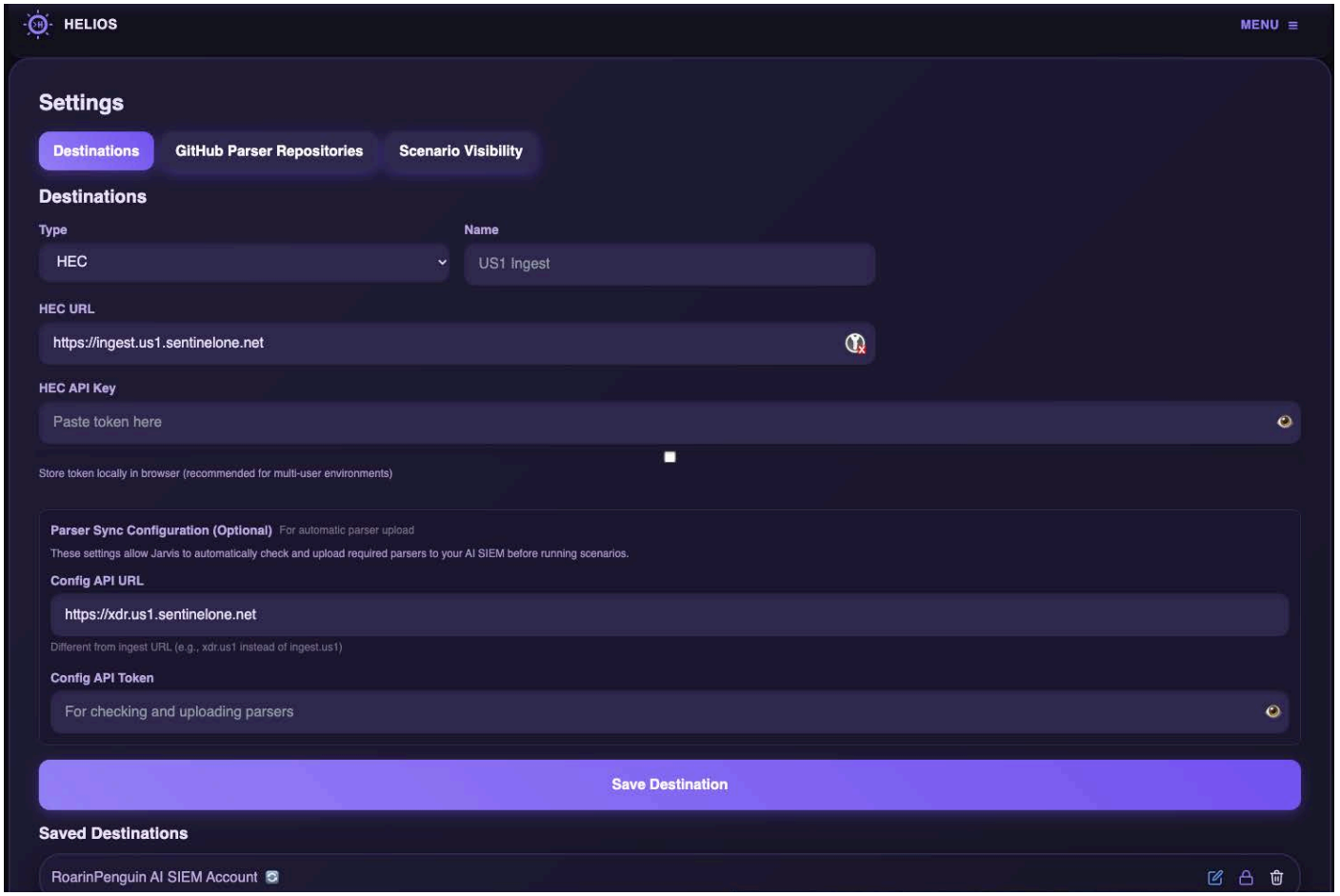
In your SentinelOne console, you can generate two types of API Keys: **Log** (read or write), used for reading or writing logs into the datalake; and **Config** (read or write), used to read or write information such as dashboards, parsers, etc.

These keys are generated in the appropriate tab in Policy & Settings - API Keys, as shown below:



Generate a valid AI SIEM *Log Write API Key* and a valid AI SIEM *Config Write API Key* in **Policy & Settings - API Keys** at Account level, depending where you want your logs to be visible. Paste your log write token in the **HEC API Key** and your config write field in **Config API Key** and click **Save**.

Remember to double check that the Config API URL is also adapted, in case you are using an “eu1” or “ap1” variant.



The system is ready to stream to that destination!

The saved configuration are listed under *Saved Destinations* and they are editable later by clicking on the  icon on the corresponding line.

NOTABENE: If you are using the console usea1-purple.sentinelone.net the system is already set up with all the needed URLs.

HELIOS IN ACTION

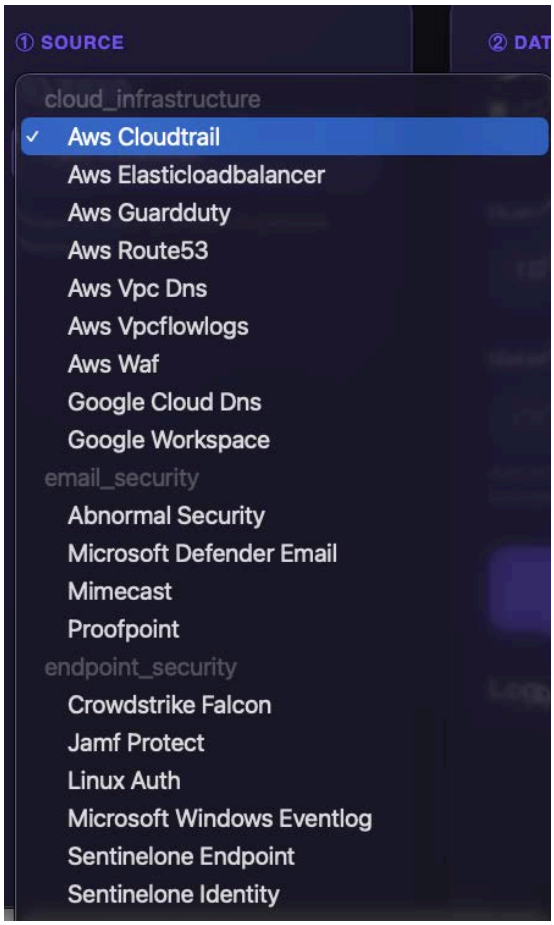
There are several ways to use HELIOS:

SOURCE	generate a single log stream for a given source
SCENARIO	leverage a set of ready made attack scenarios to produce a complex stream of events, mimicking real adversarial activity
BYOL	as in Bring Your Own Logs, with the possibility to specify the format and parser to use

Simple Log Generation

That is the default UI when you browse the site where HELIOS is running.

Ensure you have a destination configured (or do it in the Settings as instructed above), then using it is immediate: from the left hand drop down menu, select the log source you want to use.

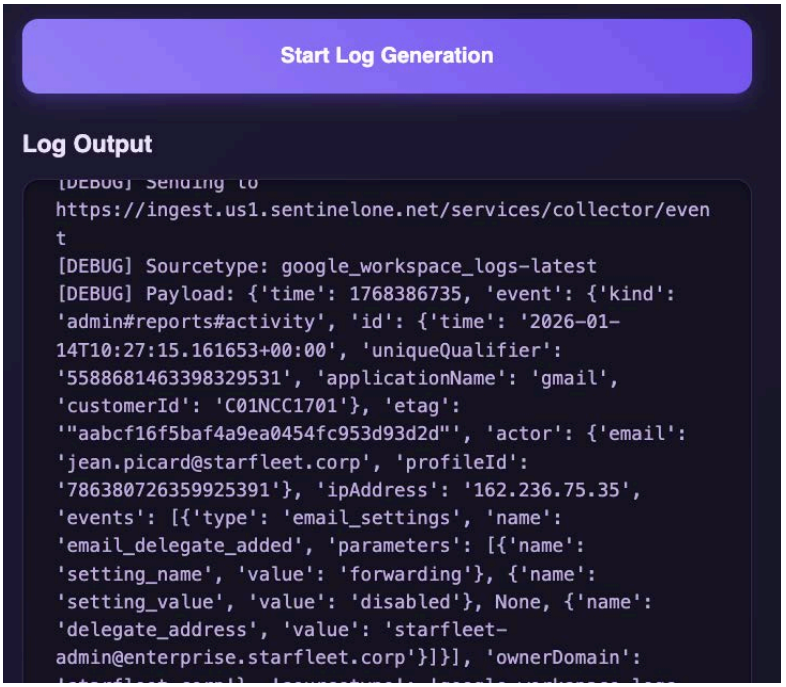


In the central part of the UI you can choose to go *Continuous Mode* or set a number of events and the rate per second.

There is an option to define metadata fields with the syntax *{"fieldname":"value"}*.
The purpose of this option is to make the logs immediately recognizable with a search like
`<trace field name>: <trace value>`

For example, `roarin.trace_id = '1234mine'`

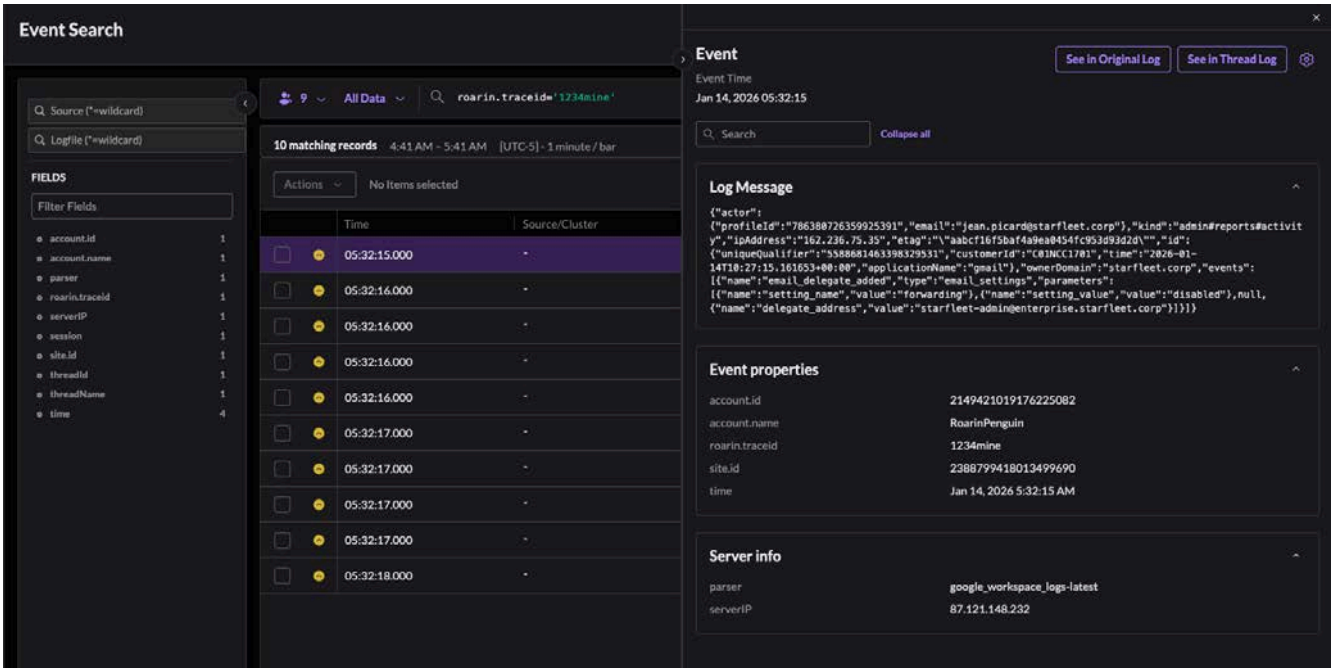
Then, simply hit **Start Log Generation**.
The synthetic logs will appear in the lower part of the UI:



At the end of a generation, it will show you the success rate:

```
Done. Delivered 10/10 successfully. Failures: 0.
INFO: Successfully sent 10 events to HEC
INFO: Log generation complete.
```

Shortly after they'll be available in your destination. Here's an example with AI SIEM:

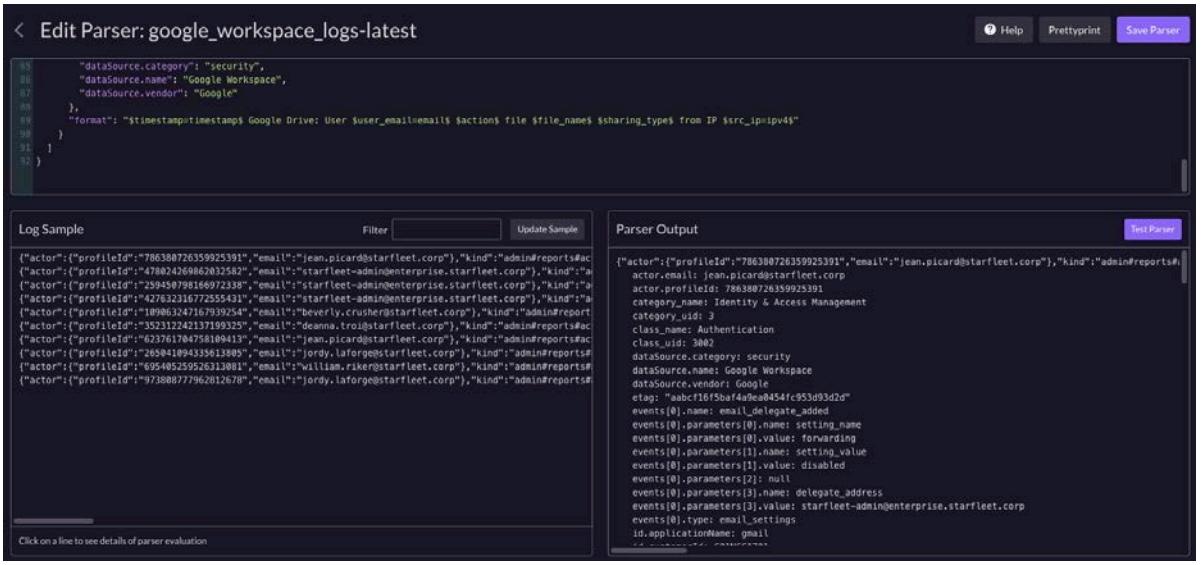


As you can see from the screenshot above, I am in the All **Data** view because I do not have the right parser installed.
But that is clearly indicated in the **Server Info** part of the screen, and that parser is likely to be available at https://github.com/Sentinel-One/ai-siem/tree/main/parsers/community/google_workspace_logs-latest

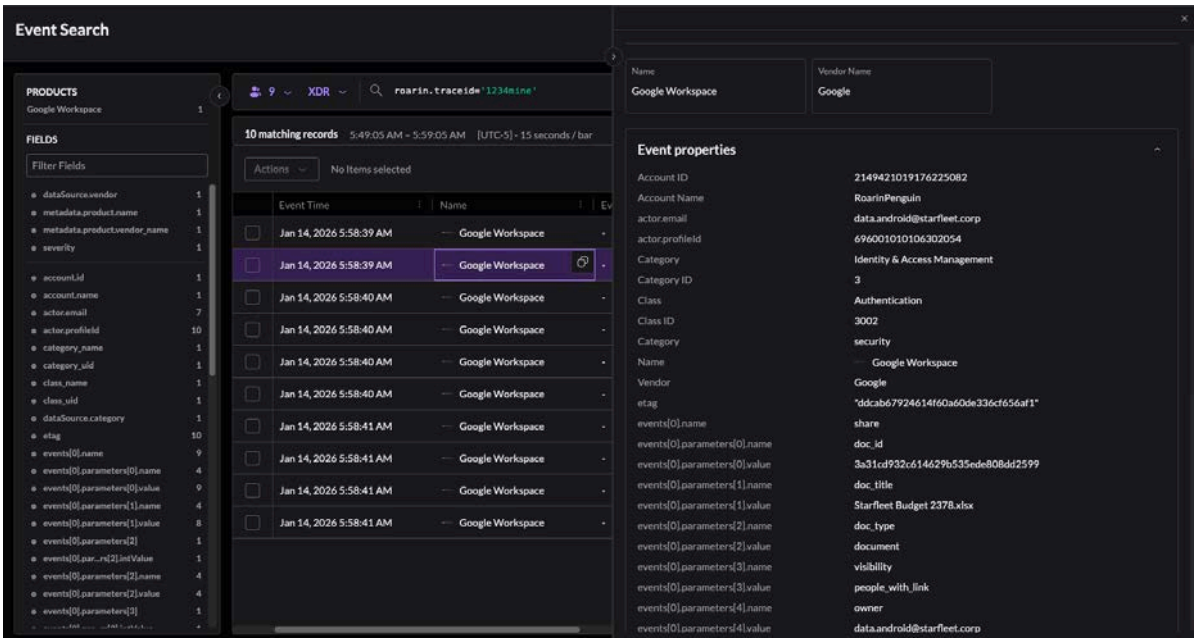
Hence, I will complete my demo setup as follows:

1) Click on **Policy & Settings** - **Parsers** in your SentinelOne Singularity Console.
2) You will see a red line showing that there are logs for the Google Workspace Parser:

3) Click on Create
4) Paste the content of the parser retrieved from the S1 Github Session and click Test Parser to see if it works:



5) Click on **Save Parser**
6) Regenerate the Log Stream to test and see the logs appearing in XDR data view as shown below.



Advanced Usage - Attack Scenarios

Scenario usage is very similar in principle to the single log source usage, meaning that it entails the same selections of an adversarial scenario type from the left side of the UI, and a destination.

At the time of writing, the list of scenarios available is the following:

LIST OF COMPLETE AND WORKING SCENARIOS			
Scenario Name	**Duration** (from gen date backwards)	**# of Events**	**Attack Phases**
Operation Digital Heist Sophisticated 14-day APT campaign against a financial services company. Simulates reconnaissance, initial access, persistence, privilege escalation, and data exfiltration.	14 days	700	• Reconnaissance & Phishing • Initial Access • Persistence & Lateral Movement • Privilege Escalation • Data Exfiltration
Finance Employee MFA Fatigue Attack Baseline (Days 1-7), MFA fatigue from Russia, OneDrive exfiltration, SOAR detections and automated response.	8 days	135	• Normal Behavior • MFA Fatigue • Initial Access • Data Exfiltration • Detection & Response
Insider Data Exfiltration via Cloud Download Insider threat scenario: anomalous large-volume M365/SharePoint downloads (180+ files), DLP classification, and removable USB media copying. Correlates Okta, M365 UAL, DLP, and EDR	8 days	280	• Baseline • Off-Hours Access • Cloud Download Spike • USB Copy • Detection
Enterprise Breach (10 min) Condensed enterprise breach scenario for quick demos.	10 minutes	120	• Initial Access • Lateral Movement • Exfiltration
Enterprise Scenario Sender (330+ events) Sends enhanced enterprise	45 minutes	330	• Initial Compromise • Credential Harvesting • Lateral Movement • Privilege Escalation

attack scenario events to HEC using proper routing.			• Data Exfiltration
LIST OF INCOMPLETE OR NOT FUNCTIONAL SCENARIOS			
Scenario Name	**Duration** (from gen date backwards)	**# of Events**	• **Attack Phases**
Scenario HEC Sender Generic scenario sender that replays a scenario JSON to HEC.	15 minutes	150	• Replay
Integration Test (Star Trek) Integration test scenario for end-to-end validation and fun output.	3 minutes	20	• TesT
Quick Scenario (Simple) Minimal scenario for smoke testing pipeline and parsers.	2 minutes	30	• Access • Movement
Quick Scenario (Comprehensive)	5 minutes	80	• Initial Access • Reconnaissance • Movement • Exfiltration
AI SIEM Showcase Scenario Showcase scenario demonstrating multi-platform correlation across EDR, Email, Identity, Cloud, Network, WAF, and more.	30 minutes	200	• Phishing • Compromise • Movement • Privilege Escalation • Exfiltration

Each of these scenarios has a purpose and should be studied and practiced *before* running it in a demo or in a POC, to understand how to pivot from one phase to another, and impress your audience at best.

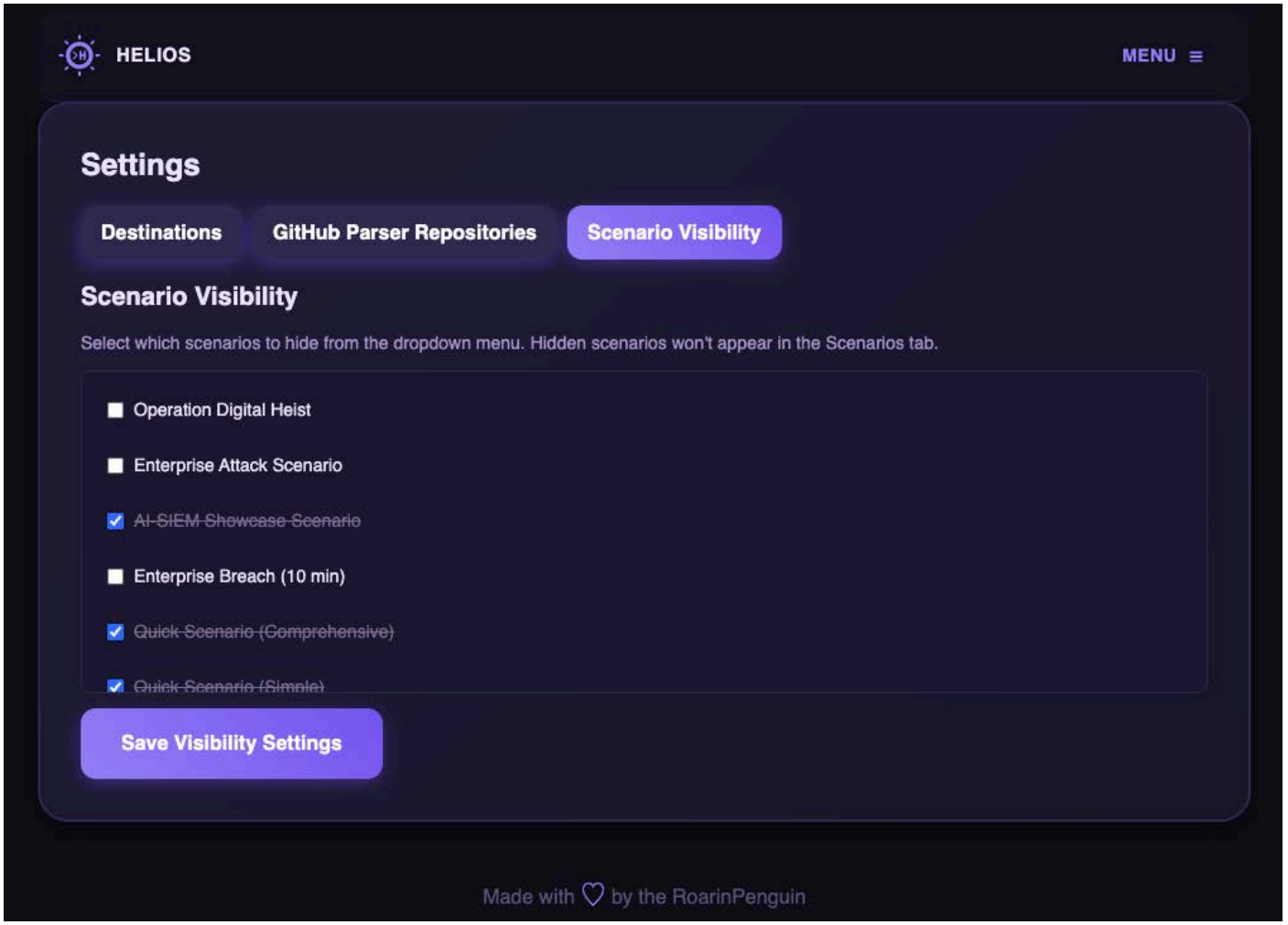
Additional Settings Options

Hiding Scenarios

A new functionality has been added to hide the scenarios you will not need or use.

Click on **Menu**, then on **Settings**, then select **Scenario Visibility**.

You will see options to select the scenario that you don't want to see in the Scenario menu, then click on Save Visibility Settings. Please refer to the screenshot below:



GitHub Parser Repositories

HELIOS will automatically add any missing parser before sending the logs you are generating, as long as they are container either locally or in any GitHub repository configured in **Settings** - **GitHub Parser Repositories**.

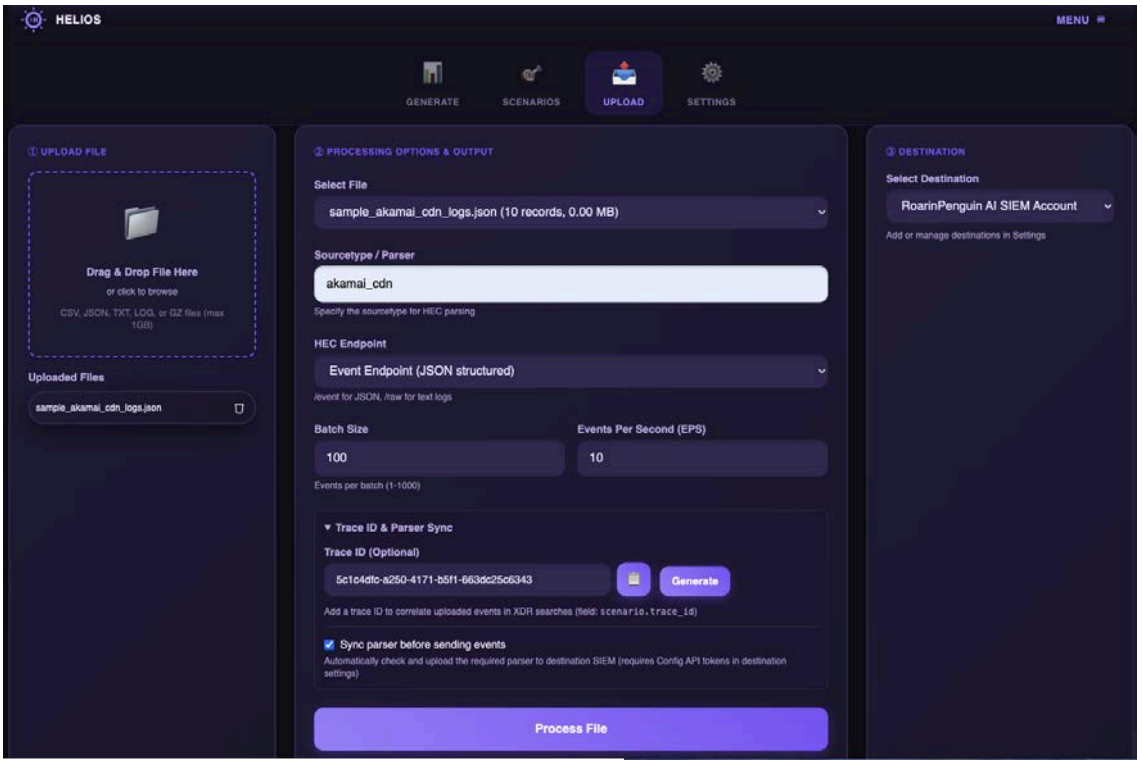
There, you can define up to three repo that HELIOS will check before streaming to the final destination.

There is a specific field to configure your **GitHub Personal Access Token (PAT)** - which will be used for two use cases:

1) The repos are public - in this situation, having your PAT will increase the API calls to GitHub - ensuring that all the configured repos are queried and the parsers retrieved.
2) One or more of the repos is private - in this situation, your PAT will be used to access the protected repo(s).

Adding your logs

The section UPLOAD allows you to load your own logs and stream them using the HELIOS capabilities to the configured destination. The UI is pretty self explanatory and in the **docs** directory you will find two log samples from an *akamai_cdn* source - one in JSON format and the other in TXT format - to allow you experimenting the two HEC endpoints: Event and Raw.
You can configure the parameters as shown in the screenshot below:



The parser will be searched and synced in case it is not present in your AI SIEM, and seconds later you'll see your telemetry in *XDR* (or *All Data*) *View* in case your parser cannot make it to your account).

TROUBLESHOOTING

In this section you will find some frequently asked questions about issues in using HELIOS, with the solution.

Q. I can't see the list of the available sources in the *Generate* UI view

A. Most likely you did not copy the file `.env copy` into `.env` in your HELIOS directory. Stop the container with `docker-compose down`, copy the file (and customize if you need it), then restart the HELIOS containers with `docker-compose up -d`.

Q. There is something strange in the logs generated but I don't want to manually scroll in the Output Log window. Can I export my logs?

A. Yes. Use the **Download Log** available in the *Scenario* UI view.

Q. Can I see the details about generated logs by my scenario? In the Output Log I see only the main steps.

A. Yes. In the Scenario UI view expand the Output Options and select the Verbose/Debug Mode flag. Then re-run your scenario and the details will appear in the Scenario Output window.

CREDITS

A book like this is not a single person production or initiative.

Even if this final result is written, coordinated, and maintained by me, below you can find the list of people who have inspired, taught, helped, corrected, and supported me in building the contents that you can enjoy in this single place.

I am immensely grateful to this list of cyber-heroes.

Mick Brown

Shane Harsch

Allan Klein

Seweryn Jodlowski

Joel Mora

Stefano Morotti

Nate Smalley