OpenRelik

A containerized incident response processing pipeline

Johan Berggren
Thomas Chopitea

Johan Berggren

"Cyber Whisperer"

OpenRelik core developer

DFIR for 25+ years, hair is mostly grey at this point

@jbn.the4711.net

Google

Thomas Chopitea

"Cyber brat wrangler"

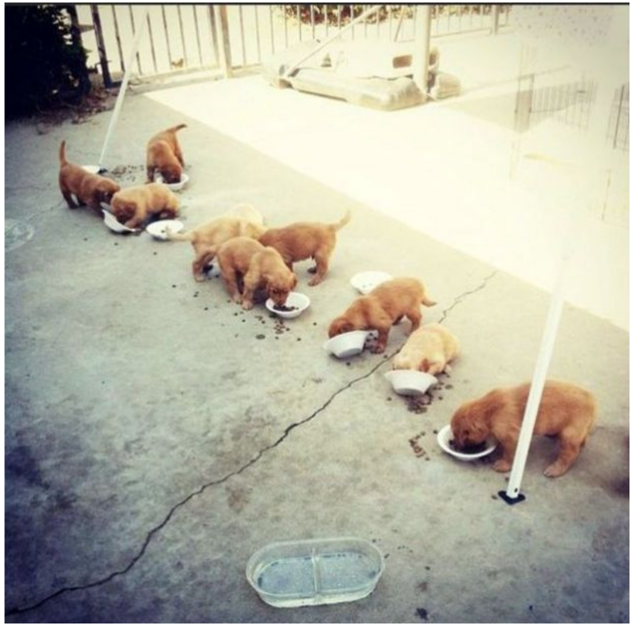
OpenRelik core sounding board

DFIR for 10+ years, hair still mostly dark

@tomchop.me

Forensic expertise: **expectations**

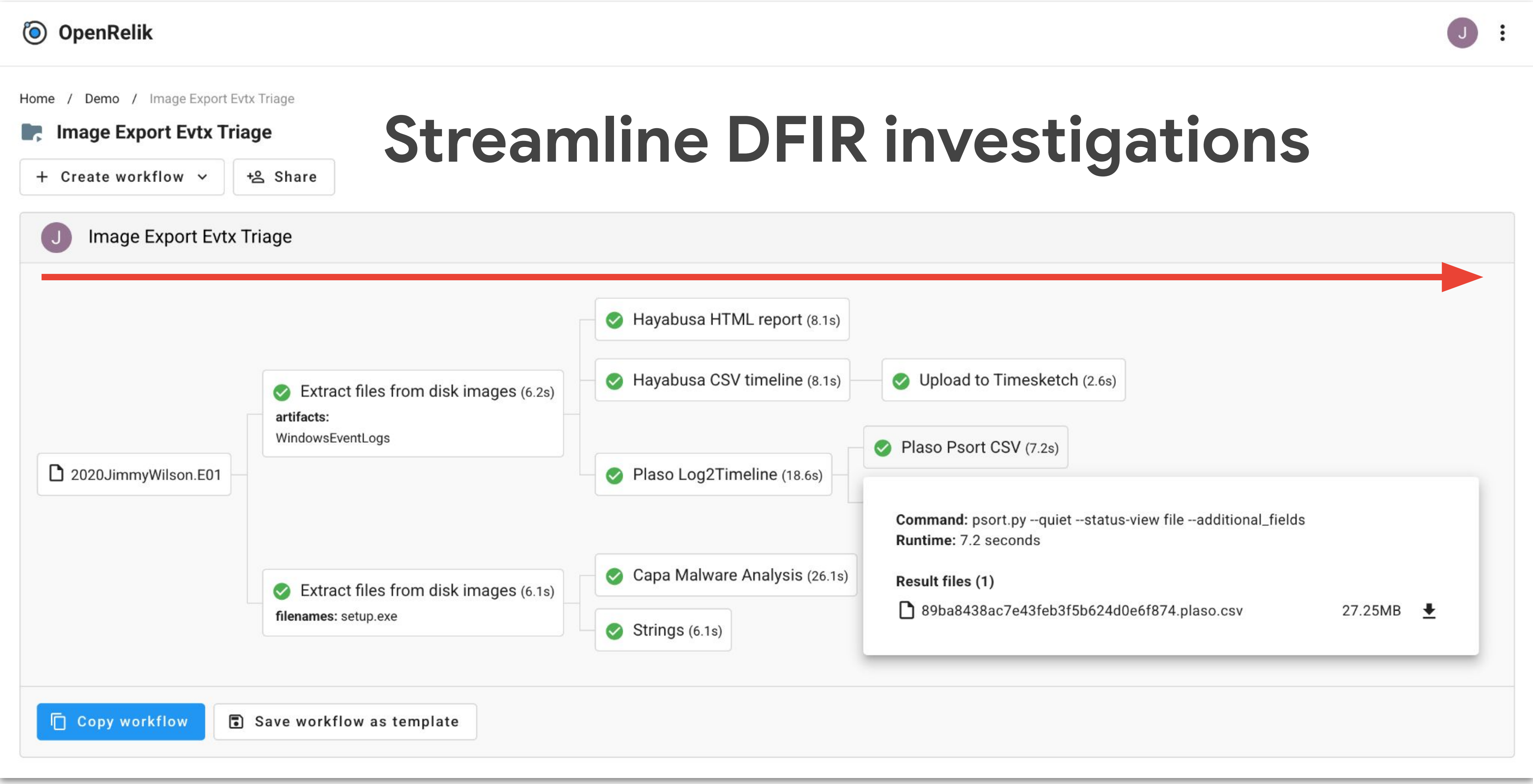
Forensic expertise: **reality**



What is OpenRelik?

Live Demo

Streamline DFIR investigations



OpenRelik

Home / Demo / Image Export Evtx Triage

Image Export Evtx Triage

Image Export Evtx Triage

2020JimmyWilson.E01

Extract files from disk images (6.2s)
artifacts: WindowsEventLogs

Extract files from disk images (6.1s)
filenames: setup.exe

Hayabusa HTML report (8.1s)

Hayabusa CSV timeline (8.1s)

Upload to Timesketch (2.6s)

Plaso Log2Timeline (18.6s)

Plaso Psort CSV (7.2s)

Capa Malware Analysis (26.1s)

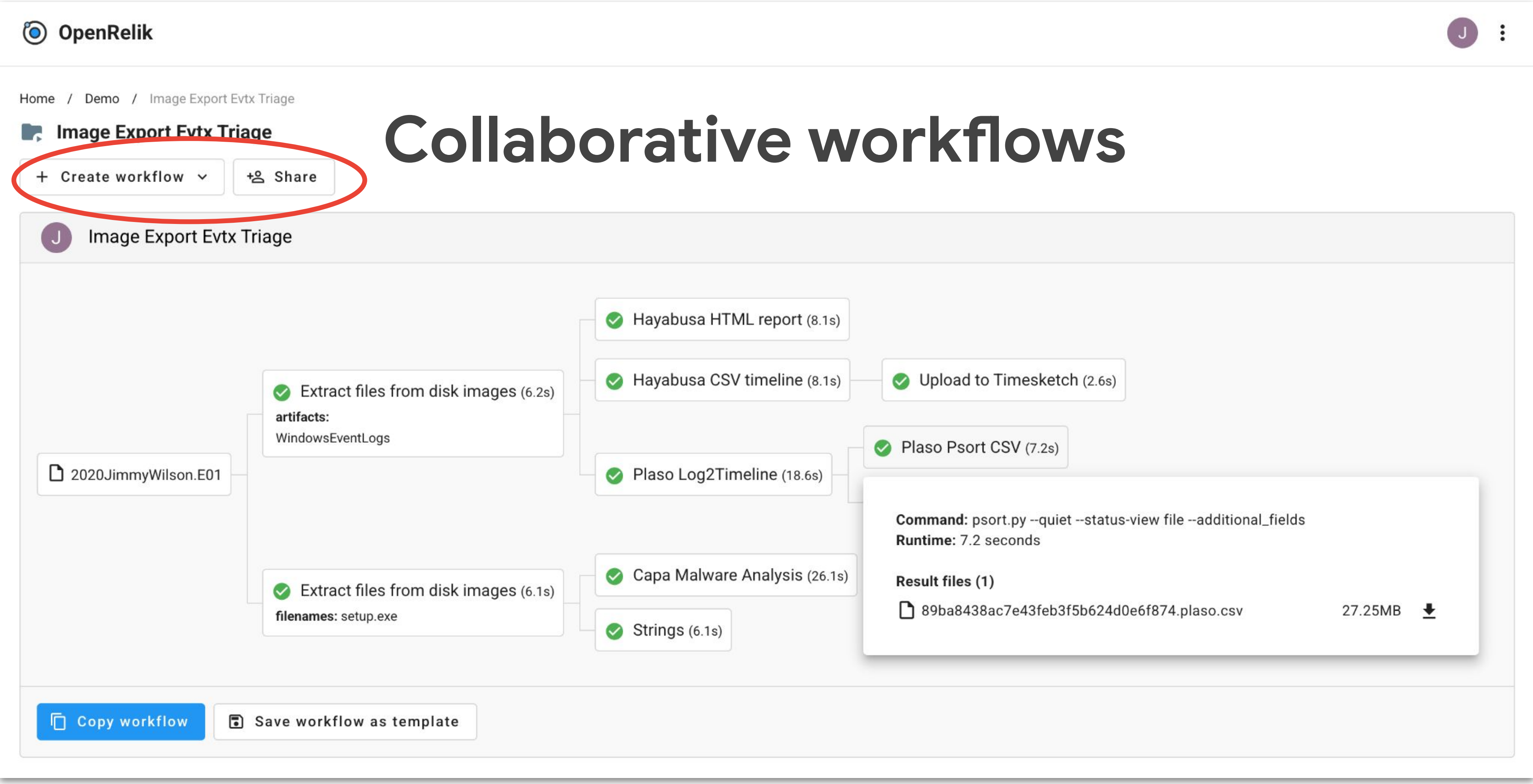
Strings (6.1s)

Command: psort.py --quiet --status-view file --additional_fields
Runtime: 7.2 seconds

Result files (1)

89ba8438ac7e43feb3f5b624d0e6f874.plaso.csv	27.25MB

Collaborative workflows



2020JimmyWilson.E01

Extract files from disk images (6.2s)
artifacts:
WindowsEventLogs

Extract files from disk images (6.1s)
filenames: setup.exe

Hayabusa HTML report (8.1s)

Hayabusa CSV timeline (8.1s)

Upload to Timesketch (2.6s)

Plaso Log2Timeline (18.6s)

Plaso Psort CSV (7.2s)

Capa Malware Analysis (26.1s)

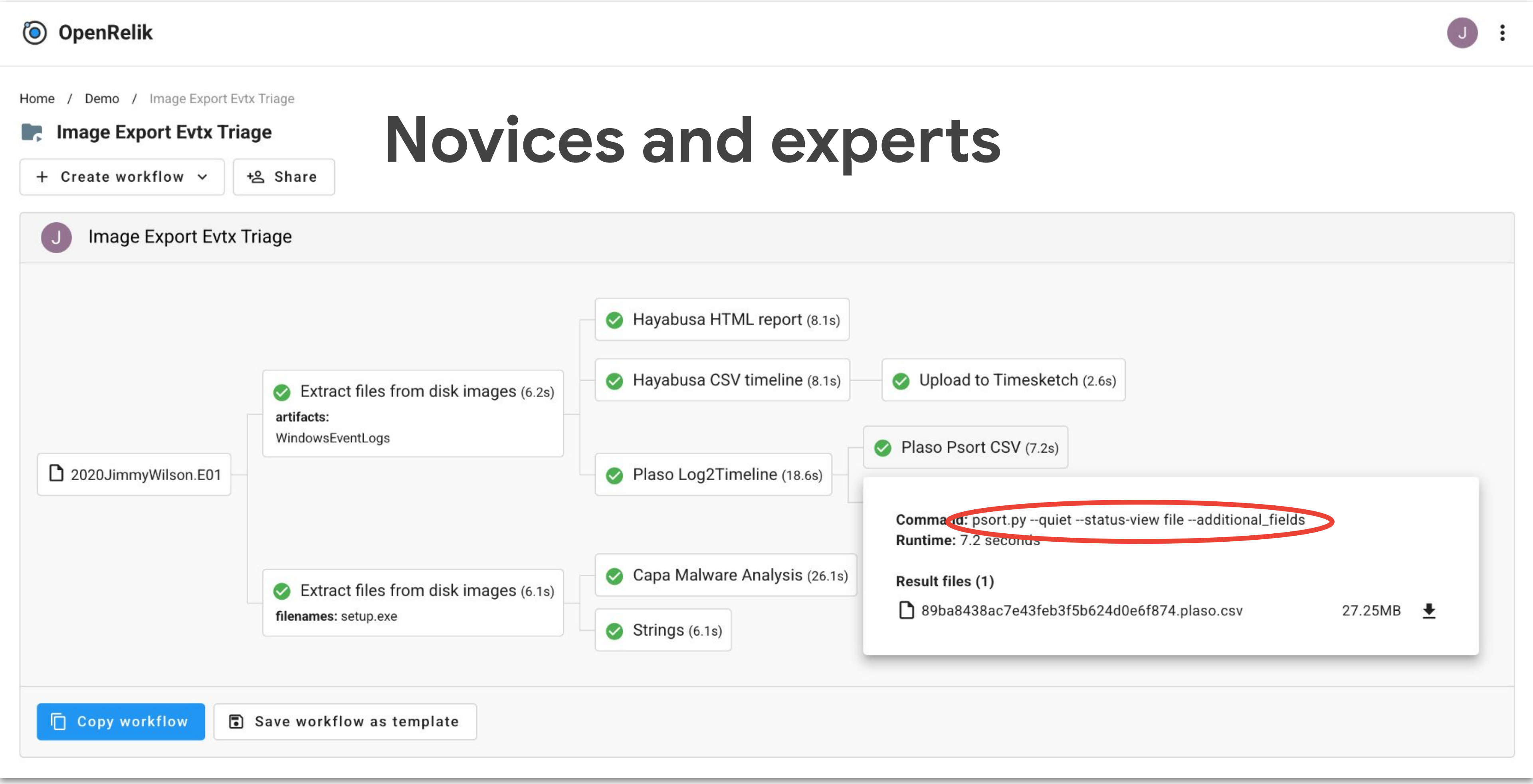
Strings (6.1s)

Command: psort.py --quiet --status-view file --additional_fields
Runtime: 7.2 seconds

Result files (1)

89ba8438ac7e43feb3f5b624d0e6f874.plaso.csv 27.25MB

Novices and experts



OpenRelik

Home / Demo / Image Export Evtx Triage

Image Export Evtx Triage

Create workflow
Share
Image Export Evtx Triage
2020JimmyWilson.E01
Extract files from disk images (6.2s)
Hayabusa HTML report (8.1s)
Hayabusa CSV timeline (8.1s)
Upload to Timesketch (2.6s)
Plaso Psort CSV (7.2s)
Plaso Log2Timeline (18.6s)
Extract files from disk images (6.1s)
Capa Malware Analysis (26.1s)
Strings (6.1s)
Copy workflow
Save workflow as template

artifacts: WindowsEventLogs

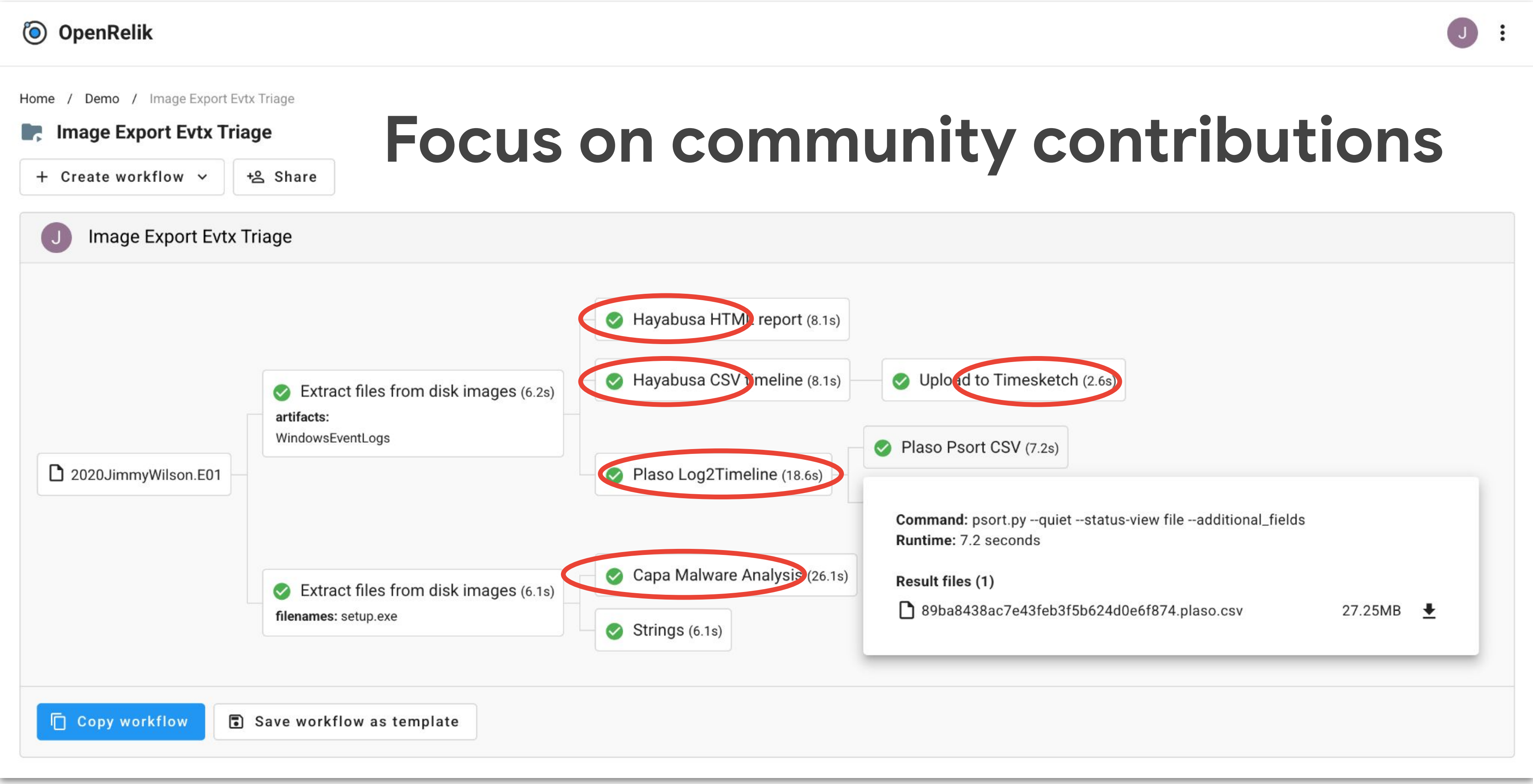
filenames: setup.exe

Command: psort.py --quiet --status-view file --additional_fields
Runtime: 7.2 seconds

Result files (1)

89ba8438ac7e43feb3f5b624d0e6f874.plaso.csv 27.25MB

Focus on community contributions



OpenRelik
Home / Demo / Image Export Evtx Triage
Image Export Evtx Triage
Create workflow
Share
Image Export Evtx Triage
2020JimmyWilson.E01
Extract files from disk images (6.2s)
artifacts: WindowsEventLogs
Extract files from disk images (6.1s)
filenames: setup.exe
Hayabusa HTML report (8.1s)
Hayabusa CSV timeline (8.1s)
Upload to Timesketch (2.6s)
Plaso Log2Timeline (18.6s)
Plaso Psort CSV (7.2s)
Capa Malware Analysis (26.1s)
Strings (6.1s)
Copy workflow
Save workflow as template

Command: psort.py --quiet --status-view file --additional_fields
Runtime: 7.2 seconds

Result files (1)

89ba8438ac7e43feb3f5b624d0e6f874.plaso.csv 27.25MB

Turbinia

Blast from the past

- Workers tightly integrated with the codebase
- Wrappers upon wrappers for task scheduling
- "just a processing pipeline": Originally no UI, hard to debug runs, hard to obtain intermediary data
- No easy access to artifacts or files besides what the workers produced

Turbinia

Lessons learned →

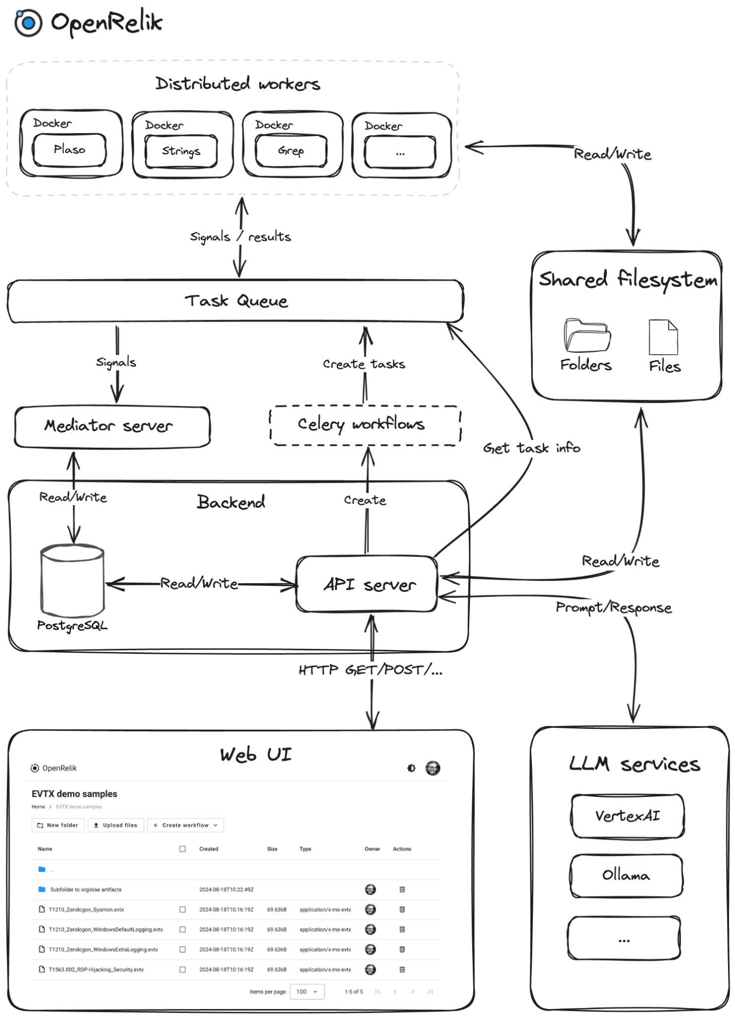
- **Worker isolation is a must** - easier dependency management and maintenance
- Use **native Celery functions** instead of reinventing the wheel
- Need for a central repository of **all files** in your investigations
- More collaboration features - clone workflows that worked, edit the ones that didn't
- Make it **extendable** and **resilient** - extensions don't depend on the core system and vice-versa

Tell me what you really, really want

- **Easy access** to files, artifacts, forensics tools that work
- **Dynamic workflows:** compose reusable workflows from the UI
- **Collaborative**: Share & remix workflows across users or cases
- **Deep dives:** Support for detailed analysis, manual and automatic
- **Resilience**: Distributed architecture with self-contained workers
- **Extensible:** Easy to add new tools to the mix, internal or external

Full-stack familiarity

- FastAPI (API backend)
- PostgreSQL (Storing metadata)
- Celery (Schedule workers)
- VueJS (Web UI)
- Docker/Podman (every component is a container)
- Shared file system (eg. Filestore, NFS)
- k8s ready (Scalable deployment)



OpenRelik

EVTX demo samples

Home / EVTX demo samples

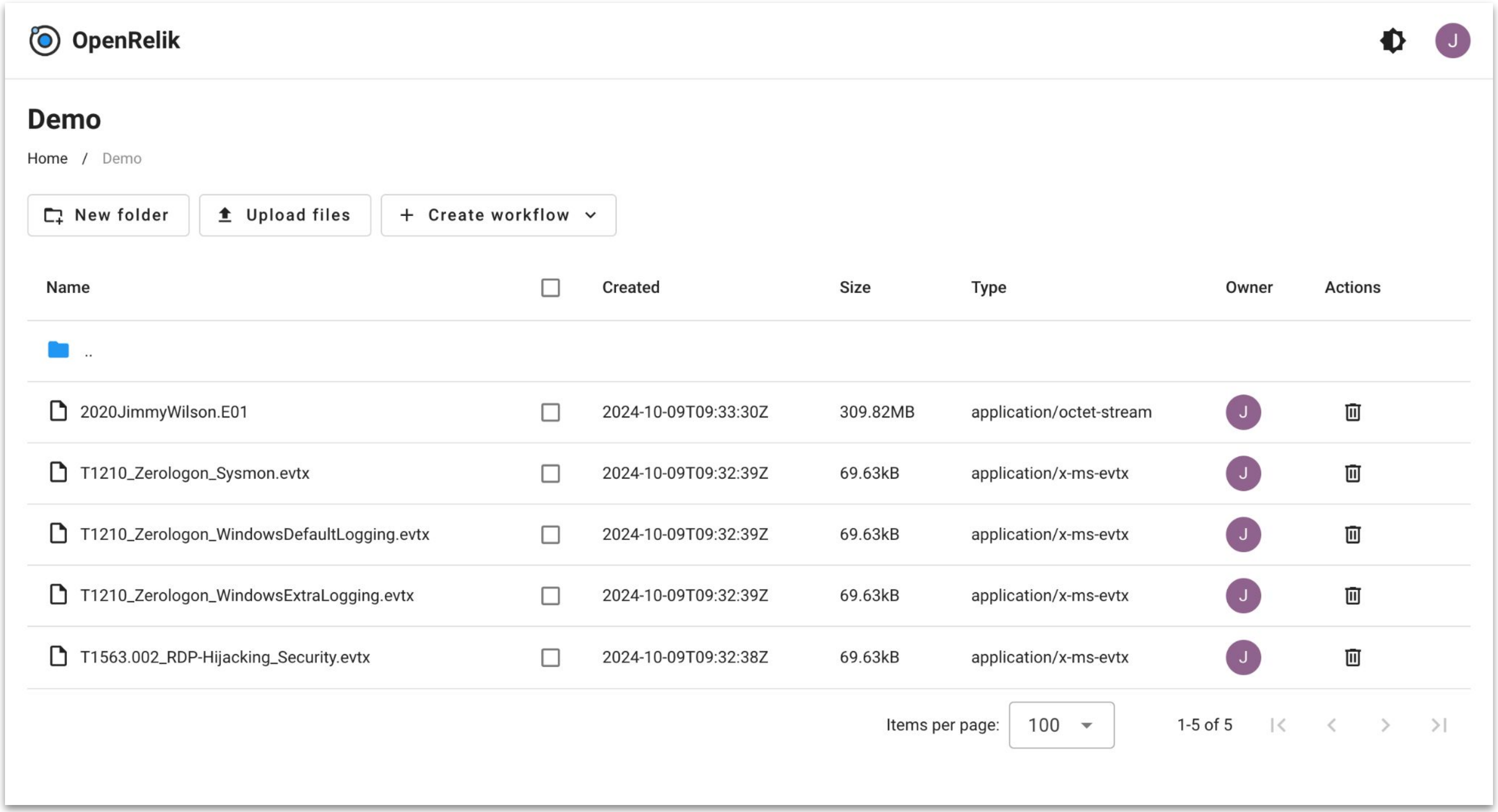
New folder | Upload files | + Create workflow

Name	☐	Created	Size	Type	Owner	Actions
..						
Subfolder to orginise artifacts		2024-08-18T10:22:49Z				
T1210_Zerologon_Sysmon.evtx	☐	2024-08-18T10:16:19Z	69.63kB	application/x-ms-evtx		
T1210_Zerologon_WindowsDefaultLogging.evtx	☐	2024-08-18T10:16:19Z	69.63kB	application/x-ms-evtx		
T1210_Zerologon_WindowsExtraLogging.evtx	☐	2024-08-18T10:16:19Z	69.63kB	application/x-ms-evtx		
T1563.002_RDP-Hijacking_Security.evtx	☐	2024-08-18T10:16:19Z	69.63kB	application/x-ms-evtx		

Items per page: 100 1-5 of 5

OpenRelik

Artifacts are organized and collaborative



OpenRelik

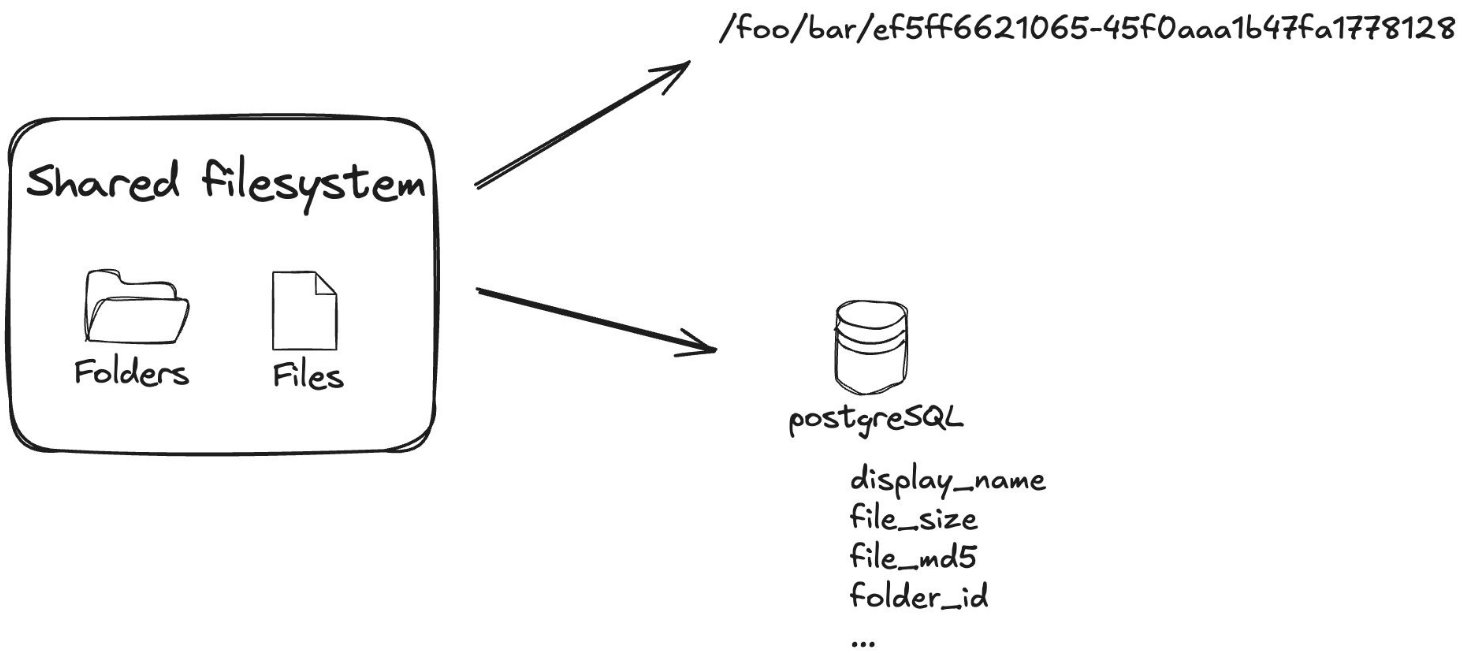
Demo

Home / Demo

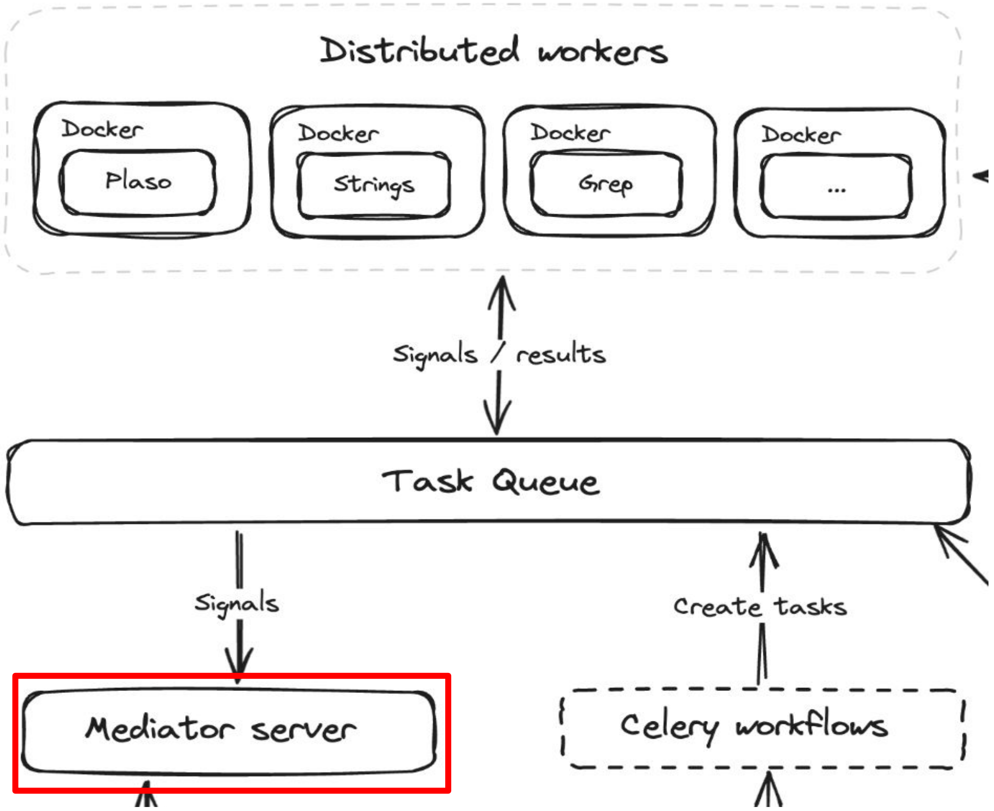
Name	Created	Size	Type	Owner	Actions
..					
2020JimmyWilson.E01	2024-10-09T09:33:30Z	309.82MB	application/octet-stream	J	
T1210_Zerologon_Sysmon.evtx	2024-10-09T09:32:39Z	69.63kB	application/x-ms-evtx	J	
T1210_Zerologon_WindowsDefaultLogging.evtx	2024-10-09T09:32:39Z	69.63kB	application/x-ms-evtx	J	
T1210_Zerologon_WindowsExtraLogging.evtx	2024-10-09T09:32:39Z	69.63kB	application/x-ms-evtx	J	
T1563.002_RDP-Hijacking_Security.evtx	2024-10-09T09:32:38Z	69.63kB	application/x-ms-evtx	J	

Items per page: 100   1-5 of 5

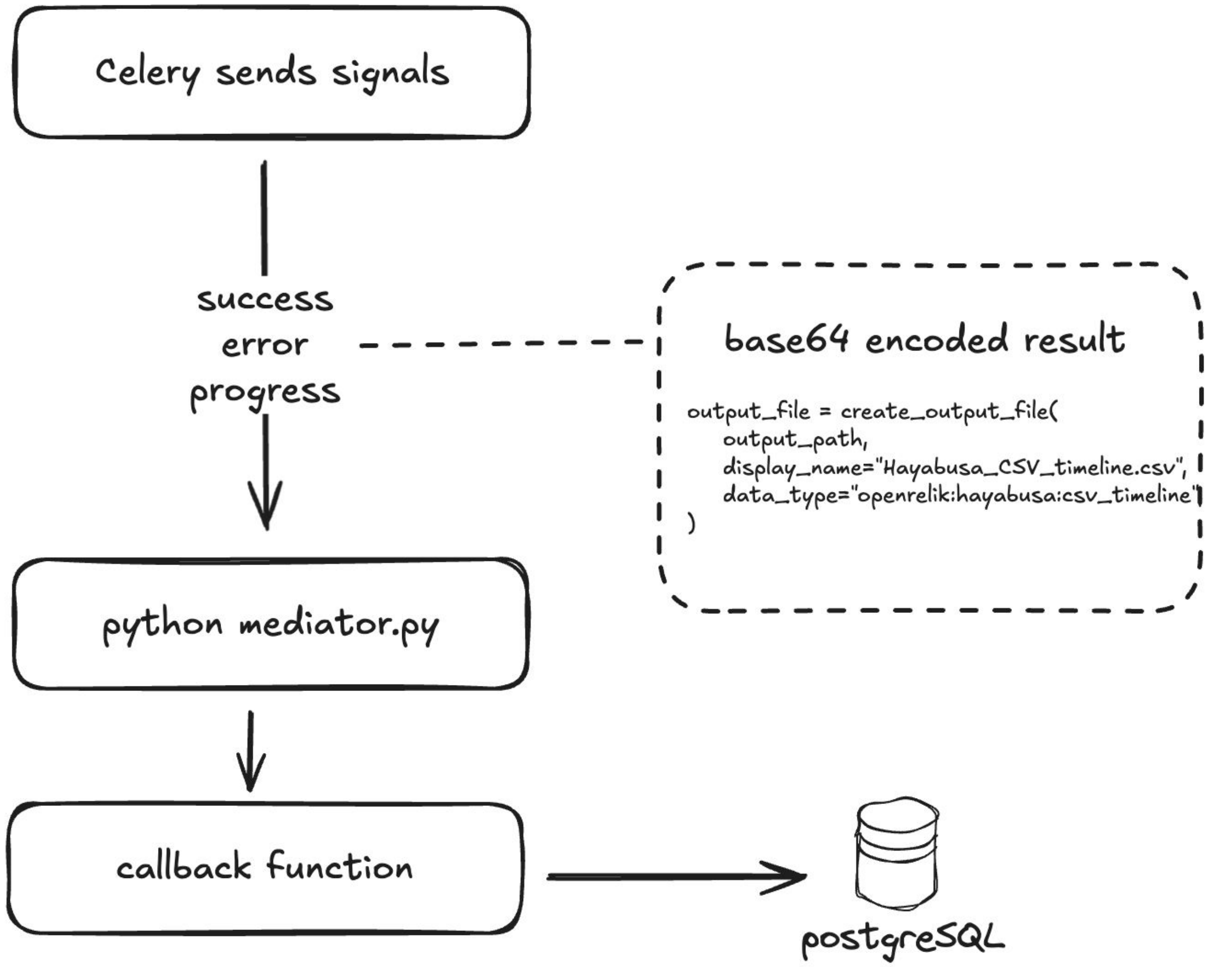
OpenRelik


/foo/bar/ef5ff6621065-45f0aaa1b47fa1778128
Shared filesystem
Folders
Files
postgreSQL
display_name
file_size
file_md5
folder_id
...

OpenRelik


Distributed workers
Docker
Plaso
Docker
Strings
Docker
Grep
Docker
...
Signals / results
Task Queue
Signals
Create tasks
Mediator server
Celery workflows

OpenRelik



```
output_file = create_output_file(
    output_path,
    display_name="Hayabusa_CSV_timeline.csv",
    data_type="openrelik:hayabusa:csv_timeline"
)
```

OpenRelik

What can we do with plain Celery?

- `chain`

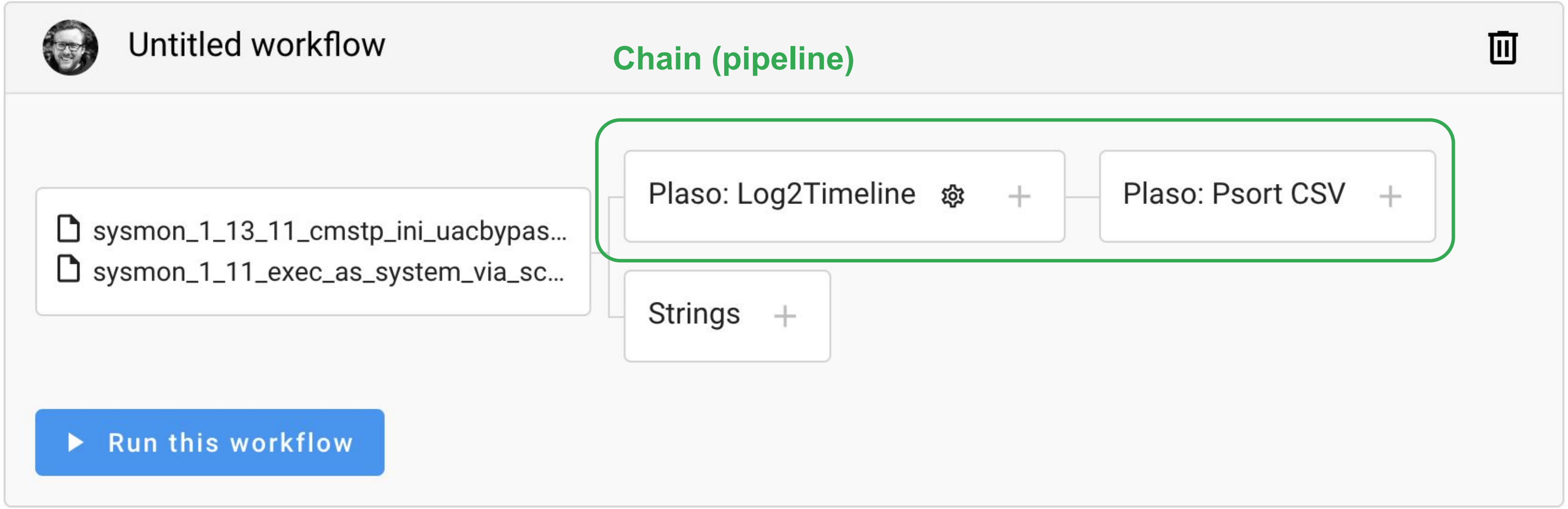
  The chain primitive lets us link together signatures so that one is called after the other, essentially forming a *chain* of callbacks (pipeline).

- `group`

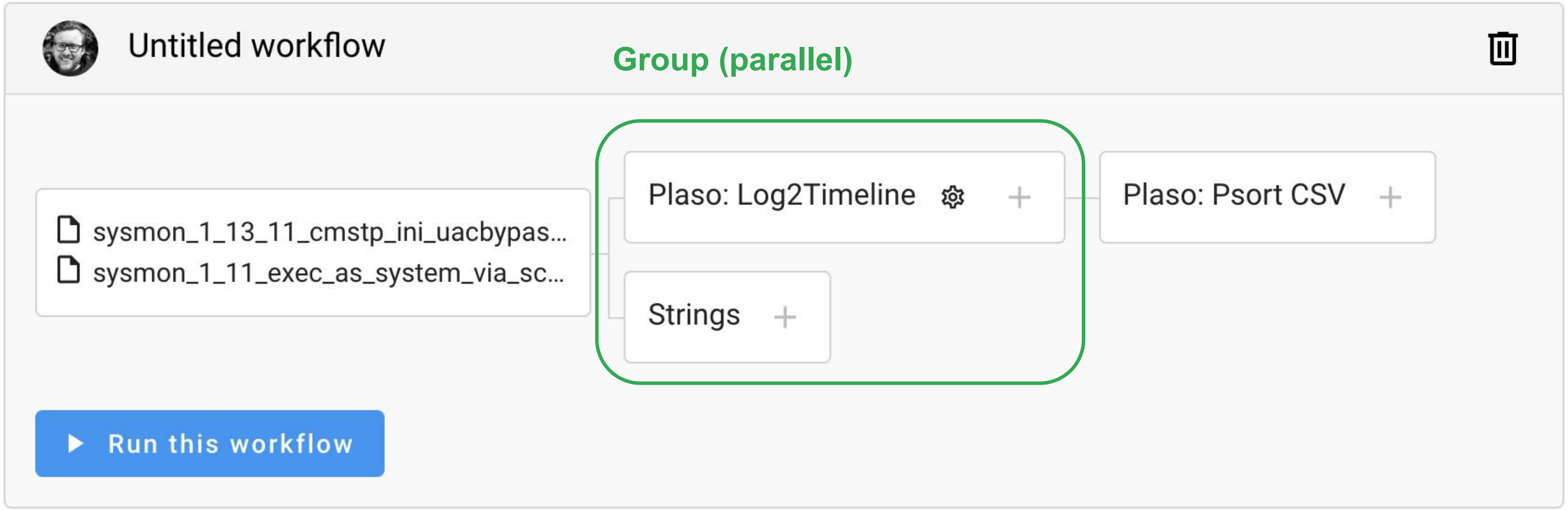
  The group primitive is a signature that takes a list of tasks that should be applied in parallel.

- `chord`

  A chord is just like a group but with a callback. A chord consists of a header group and a body, where the body is a task that should execute after all of the tasks in the header are complete.


Untitled workflow

sysmon_1_13_11_cmstp_ini_uacbypas...
sysmon_1_11_exec_as_system_via_sc...



Untitled workflow

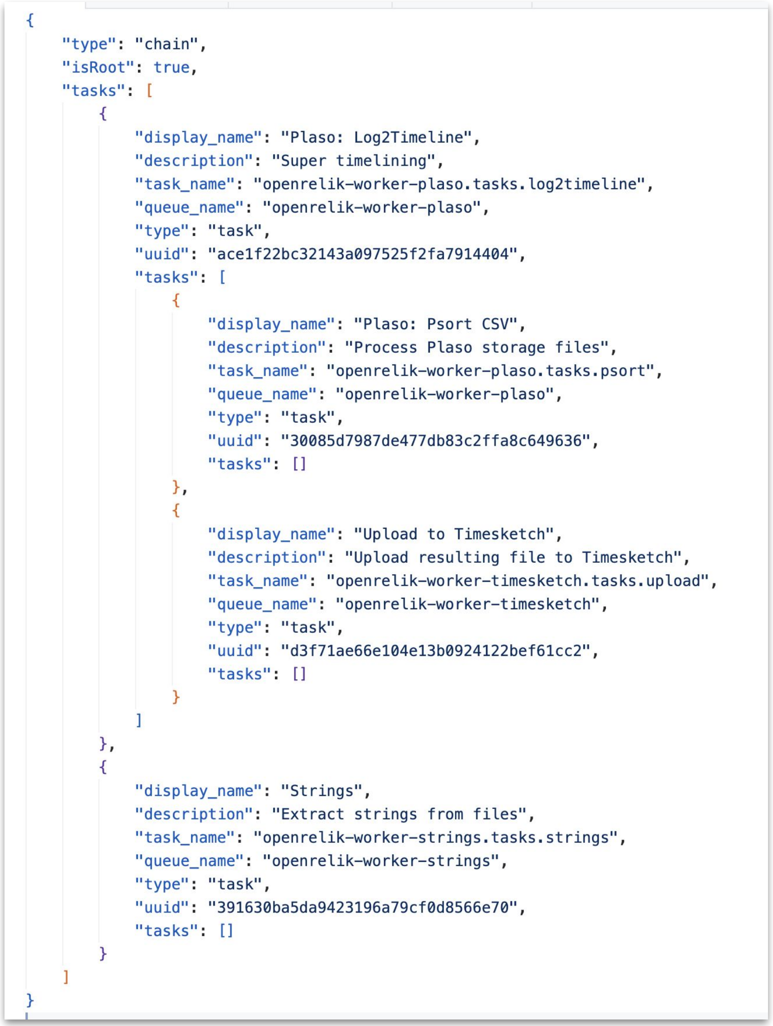
Group (parallel)

sysmon_1_13_11_cmstp_ini_uacbypas...
sysmon_1_11_exec_as_system_via_sc...

Plaso: Log2Timeline

Strings

Plaso: Psort CSV



```
{
    "type": "chain",
    "isRoot": true,
    "tasks": [
        {
            "display_name": "Plaso: Log2Timeline",
            "description": "Super timelining",
            "task_name": "openrelik-worker-plaso.tasks.log2timeline",
            "queue_name": "openrelik-worker-plaso",
            "type": "task",
            "uuid": "ace1f22bc32143a097525f2fa7914404",
            "tasks": [
                {
                    "display_name": "Plaso: Psort CSV",
                    "description": "Process Plaso storage files",
                    "task_name": "openrelik-worker-plaso.tasks.psort",
                    "queue_name": "openrelik-worker-plaso",
                    "type": "task",
                    "uuid": "30085d7987de477db83c2ffa8c649636",
                    "tasks": []
                },
                {
                    "display_name": "Upload to Timesketch",
                    "description": "Upload resulting file to Timesketch",
                    "task_name": "openrelik-worker-timesketch.tasks.upload",
                    "queue_name": "openrelik-worker-timesketch",
                    "type": "task",
                    "uuid": "d3f71ae66e104e13b0924122bef61cc2",
                    "tasks": []
                }
            ]
        },
        {
            "display_name": "Strings",
            "description": "Extract strings from files",
            "task_name": "openrelik-worker-strings.tasks.strings",
            "queue_name": "openrelik-worker-strings",
            "type": "task",
            "uuid": "391630ba5da9423196a79cf0d8566e70",
            "tasks": []
        }
    ]
}
```

```python
def create_workflow(task_data):
    if task_data["type"] == "chain":
        if len(task_data["tasks"]) > 1:
            return celery_group(
                create_workflow(task) for task in task_data["tasks"]
            )
        else:
            return celery_chain(create_workflow(task_data["tasks"][0]))
    elif task_data["type"] == "task":
        task_signature = get_task_signature(task_data)
        if task_data["tasks"]:
            if len(task_data["tasks"]) > 1:
                return celery_chain(
                    task_signature,
                    celery_group(create_workflow(t) for t in task_data["tasks"]),
                )
            else:
                return celery_chain(
                    task_signature, create_workflow(task_data["tasks"][0])
                )
        else:
            return task_signature
    else:
        raise ValueError(f"Unsupported task type: {task_data['type']}")

celery_workflow = create_workflow(workflow_spec.get("workflow"))
celery_workflow.apply_async()
```

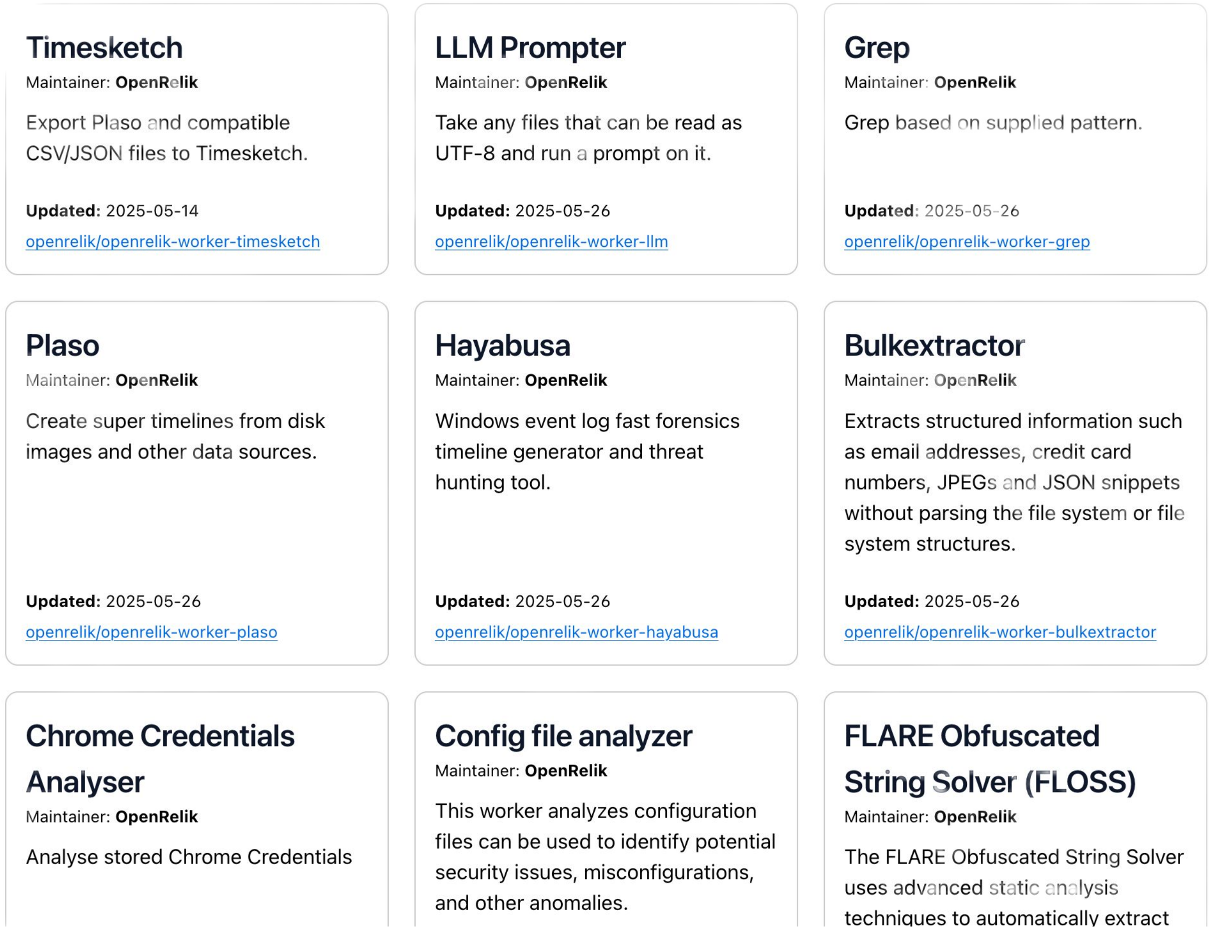
Easy to develop new workers

- No dependencies on the core system
  - Auxiliary functions, e.g. `create_output_file`

1. Fork https://github.com/openrelik/openrelik-worker-template
2. Add your cmdline invocations
3. Start your container
4. Profit!!

```python
@celery.task(bind=True, name=TASK_NAME, metadata=TASK_METADATA)
def grep_command(
    self,
    pipe_result: str = None,
    input_files: list = None,
    output_path: str = None,
    workflow_id: str = None,
    task_config: dict = None,
) -> str:
    input_files = get_input_files(pipe_result, input_files or [])
    output_files = []
    command = ["grep", "-Ei", task_config.get("regex")]

    for input_file in input_files:
        output_file = create_output_file(
            output_path, display_name=input_file.get("display_name") + ".grep"
        )
        command += [input_file.get("path")]

        with open(output_file.path, "w") as fh:
            subprocess.Popen(command, stdout=fh)

        output_files.append(output_file.to_dict())

    return create_task_result(
        output_files=output_files, workflow_id=workflow_id, command=" ".join(command)
    )
```

https://openrelik.org/marketplace/



Timesketch
Maintainer: **OpenRelik**

Export Plaso and compatible CSV/JSON files to Timesketch.

Updated: 2025-05-14

openrelik/openrelik-worker-timesketch

LLM Prompter
Maintainer: **OpenRelik**

Take any files that can be read as UTF-8 and run a prompt on it.

Updated: 2025-05-26

openrelik/openrelik-worker-llm

Grep
Maintainer: **OpenRelik**

Grep based on supplied pattern.

Updated: 2025-05-26

openrelik/openrelik-worker-grep

Plaso
Maintainer: **OpenRelik**

Create super timelines from disk images and other data sources.

Updated: 2025-05-26

openrelik/openrelik-worker-plaso

Hayabusa
Maintainer: **OpenRelik**

Windows event log fast forensics timeline generator and threat hunting tool.

Updated: 2025-05-26

openrelik/openrelik-worker-hayabusa

Bulkextractor
Maintainer: **OpenRelik**

Extracts structured information such as email addresses, credit card numbers, JPEGs and JSON snippets without parsing the file system or file system structures.

Updated: 2025-05-26

openrelik/openrelik-worker-bulkextractor

Chrome Credentials Analyser
Maintainer: **OpenRelik**

Analyse stored Chrome Credentials

Config file analyzer
Maintainer: **OpenRelik**

This worker analyzes configuration files can be used to identify potential security issues, misconfigurations, and other anomalies.

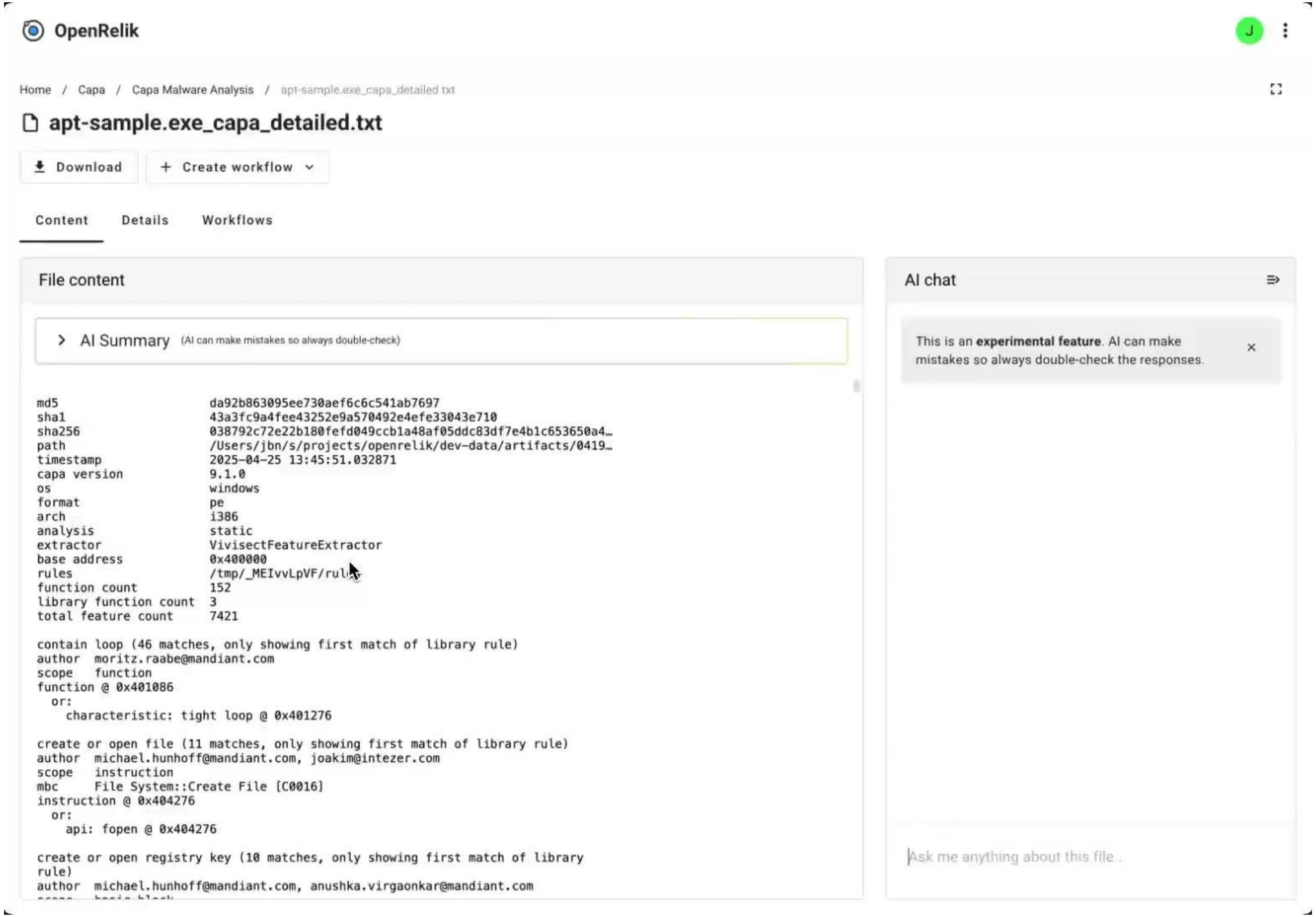
FLARE Obfuscated String Solver (FLOSS)
Maintainer: **OpenRelik**

The FLARE Obfuscated String Solver uses advanced static analysis techniques to automatically extract

✨ AI-enhanced investigations ✨

Demo 1

[File Summary and Chat]()


OpenRelik
J
Home / Capa / Capa Malware Analysis / apt-sample.exe_capa_detailed.txt

apt-sample.exe_capa_detailed.txt

Download
Create workflow
Content
Details
Workflows
File content
AI Summary (AI can make mistakes so always double-check)
AI chat

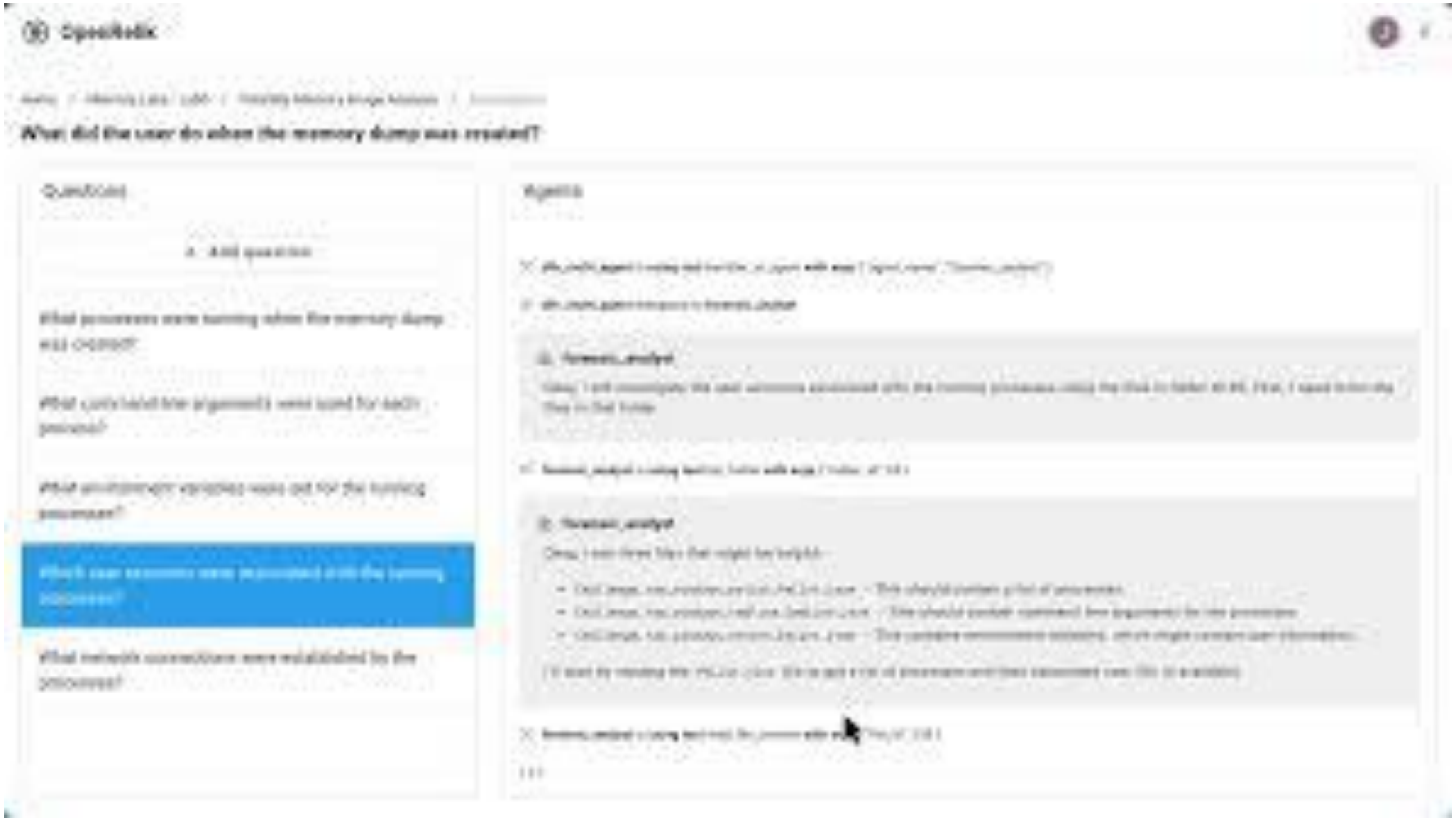
This is an **experimental feature**. AI can make mistakes so always double-check the responses.

```
md5                     da92b863095ee730aef6c6c541ab7697
sha1                    43a3fc9a4fee43252e9a570492e4efe33043e710
sha256                  038792c72e22b180fefd049ccb1a48af05ddc83df7e4b1c653650a4…
path                    /Users/jbn/s/projects/openrelik/dev-data/artifacts/0419…
timestamp               2025-04-25 13:45:51.032871
capa version            9.1.0
os                      windows
format                  pe
arch                    i386
analysis                static
extractor               VivisectFeatureExtractor
base address            0x400000
rules                   /tmp/_MEIvvLpVF/rul
function count          152
library function count  3
total feature count     7421

contain loop (46 matches, only showing first match of library rule)
author  moritz.raabe@mandiant.com
scope   function
function @ 0x401086
  or:
    characteristic: tight loop @ 0x401276

create or open file (11 matches, only showing first match of library rule)
author  michael.hunhoff@mandiant.com, joakim@intezer.com
scope   instruction
mbc     File System::Create File [C0016]
instruction @ 0x404276
  or:
    api: fopen @ 0x404276

create or open registry key (10 matches, only showing first match of library
rule)
author  michael.hunhoff@mandiant.com, anushka.virgaonkar@mandiant.com
```

✨ AI-enhanced investigations ✨

Demo 2

What DFIQ is happening?



OpenRelik

[illegible] / Memory Lab / LLM / [illegible] Memory Image Analysis / [illegible]

What did the user do when the memory dump was created?

Questions

+ Add question

What processes were running when the memory dump was created?

What command line arguments were used for each process?

What environment variables was set for the running processes?

Which user accounts were associated with the running processes?

What network connections were established by the processes?

Agents

[illegible]

[illegible] forensic_analyst

forensic_analyst

[illegible]

[illegible]

forensic_analyst

[illegible]

- [illegible]
- [illegible]
- [illegible]

[illegible]

[illegible]

...

AI-enhanced investigations

- Model agnostic
- No vendor lock-in
- No tight coupling with AI features

Key takeaways

- **Server-based forensic processing and analysis platform**
- Fully open-source stack
- De-coupled architecture allows for easy integration of custom, internal workflows
- Augment analysis using AI-powered features
- Enable everyone to contribute