Tamarin Python Wrapper

Luca Mandrelli

June 26, 2025

What will we talk about ?

1. Why? / Context
2. How does the config Works ?
3. High-Level Architecture

Why? / Context

The problem

- A lot of different use cases
- A lot of different but specific solution (ut-tamarin, ANSSI wrapper...)
- Specific solutions needed a development environment

Solution: Tamarin Wrapper

- Easy to install (currently via pip, future : Docker container)
- Polyvalent : you describe a complete input, you can customize it widely
- Complete error handling

What Tamarin Wrapper Provides

Key Features:

- Batch Execution: Multiple models across different Tamarin versions
- JSON Configuration: Simple, declarative execution recipes
- Progress Tracking: Real-time updates, error handling
- Output Processing: Structured JSON results with detailed summaries

How does the config Works ?

Configuration Structure

Three main sections:

```
{
  "config": { /* Global system settings */ },
  "tamarin_versions": { /* Tamarin executables linked to an alias */ },
  "tasks": { /* Description of the wrapper work */ }
}
```

- Declarative: Describe what you want, not how
- Hierarchical: Global → Task → Lemma settings
- Flexible: Override settings at any level

Global Configuration

```
{
  "config": {
    "global_max_cores": 8,
    "global_max_memory": 16,
    "default_timeout": 7200,
    "output_directory": "./results"
  }
}
```

System-wide resource limits : the wrapper will also warn the user in case of misconfiguration (ex 32GB set while system has 16GB)

Tamarin Versions

```
{
  "tamarin_versions": {
    "stable": {
      "path": "tamarin-binaries/1.10.0/bin/tamarin-prover"
    },
    "dev": {
      "path": "tamarin-binaries/1.11.0/bin/tamarin-prover",
    }
  }
}
```

Named aliases for different Tamarin executables, the path can also be a symbolic link

Task Configuration

```
{
  "tasks": {
    "wpa2_analysis": {
      "theory_file": "protocols/wpa2.spthy",
      "tamarin_versions": ["stable", "dev"],
      "output_file_prefix": "wpa2_results",
      "tamarin_options": ["-v"],
      "preprocess_flag": ["badKey"],
      "ressources": {
        "max_cores": 4,
        "max_memory": 8,
        "timeout": 3600
      }
    }
  }
}
```

This task will prove all the lemmas on the “stable” and “dev” versions of the tamarin-prover :

```
tamarin-binaries/tamarin-prover-1.10/1.10.0/bin/tamarin-prover +RTS -N4 -RTS protocols/wpa2.spthy
--prove -v -D="badKey"
--output=results_26-06-25_11-38-24/models/wpa2_analysis_stable.spthy
```

Lemma-Level Configuration

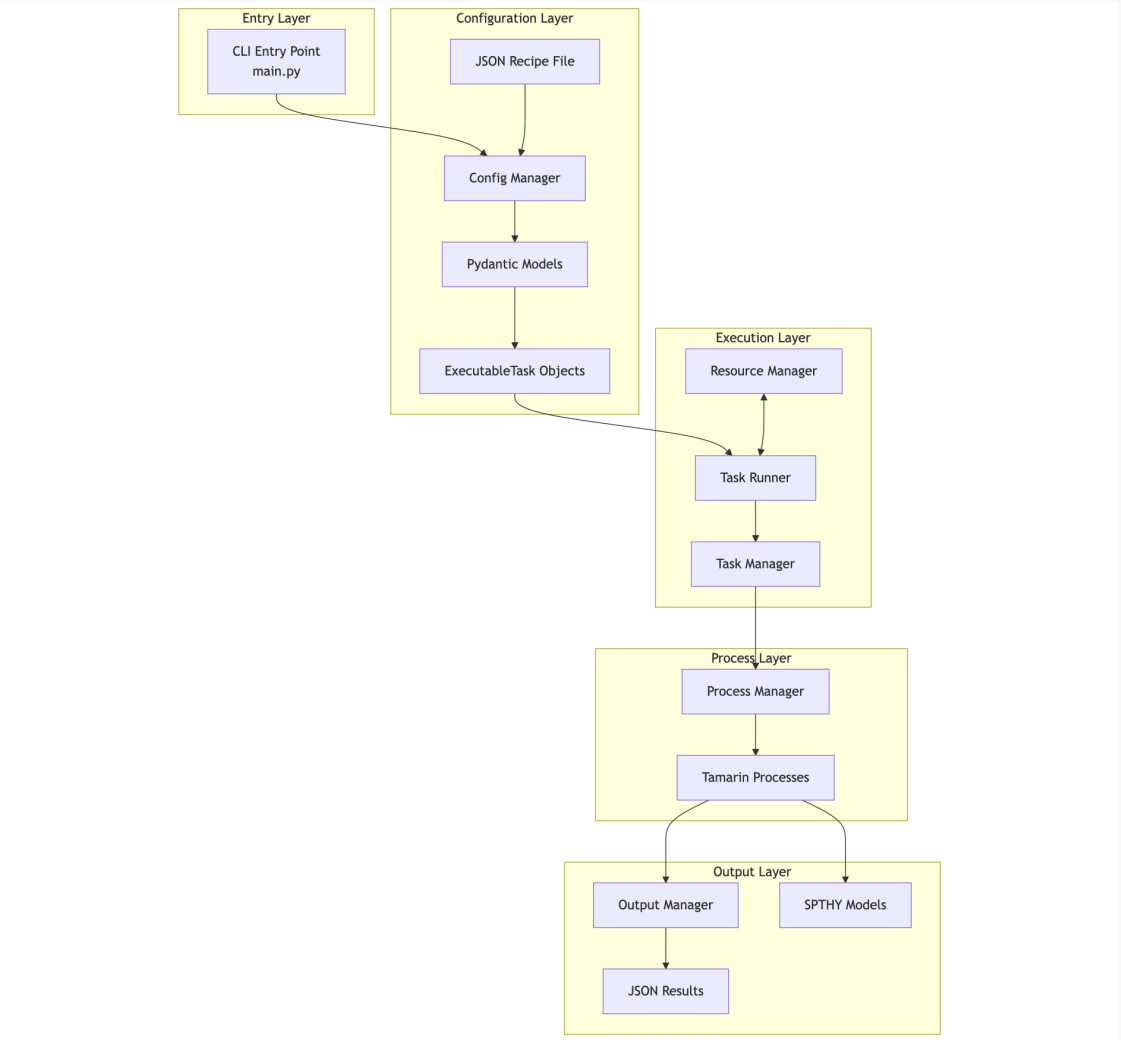
```
"tasks": {
"wpa2": {
  "theory_file": "protocols/wpa2_four_way_handshake_unpatched.spthy",
  "tamarin_versions": ["stable", "dev"],
  "output_file_prefix": "WPA2_Example",
    "preprocess_flags": ["goodKey"],
    "tamarin_options" : ["-v"],
  "ressources": {
    "max_cores": 2,
    "max_memory": 8,
      "timeout": 3600
  },
    "lemmas" : [
    {
      "name": "nonce_reuse_key_type",
      "tamarin_options":["--heuristic=S"],
      "ressources": {
        "max_cores": 1
      }
    },
    {
      "name": "authenticator_rcv_m2_must_be_preceded_by_snd_m1",
      "tamarin_versions": ["stable"],
      "preprocess_flags": ["badKey"],
      "ressources": {
        "max_memory": 16,
        "timeout": 30
      }
    }
}
```

Per-lemma overrides with configuration inheritence

Global Config
↓
Task Config (inherits & overrides)
↓
Lemma Config (inherits & overrides)

High-Level Architecture

System Architecture Overview



Key modules and their roles:

- `main.py`: CLI entry point
- `runner.py`: Coordination of the wrapper
- `config_manager.py`: JSON loading and validation
- `task_manager.py`: Individual task execution
- `process_manager.py`: Low-level process control
- `resource_manager.py`: Global resource tracking
- `output_manager.py`: Result parsing and formatting

Output Structure

```
output_directory/
├── success/
│   ├── task_lemma_version.json
│   └── ...
├── failed/
│   ├── task_lemma_version.json
│   └── ...
└── models/
    ├── task_lemma_version.spthy
    └── ...
```

Organized results with clear success/failure separation

How to find the project

- On the team gitlab : https://projects.cispa.saarland/cc/tamarin-wrapper
- On PyPI, with the last stable version you can test

You can give your ideas to add features to the wrapper with the gitlab issues: https://projects.cispa.saarland/cc/tamarin-wrapper/-/issues

GitLab Project

GitLab Issues

Questions?

PyPI Package