Stefan Tatschner [@rumpelsepp], FOSDEM 23, 05 Feb 2022

---

gallia: An Extendable Pentesting Framework
Overview

1. Meta
2. Status Quo
3. Outlook
4. Demo
About Me

- Stefan Tatschner
- Security Researcher
- Maintainer of gallia

Github  @rumpelsepp
Matrix   @rumpelsepp:hackbrett.de
Mastodon @rumpelsepp@mastodon.social
LinkedIn in/stefan-tatschner
Gallia is an extendable pentesting framework with the focus on the automotive domain. The scope of the toolchain is conducting penetration tests from a single ECU up to whole cars. Currently, the main focus lies on the UDS interface. Acting as a generic interface, the logging functionality implements reproducible tests and enables post-processing tasks. The rendered documentation is available via Github Pages. Alternatively, the documentation is hosted on readthedocs as well. The documentation for the current stable release is available on readthedocs.

Keep in mind that this project is intended for research and development usage only! Inappropriate usage might cause irreversible damage to the device under test. We do not take any responsibility for damage caused by the usage of this tool.

https://github.com/Fraunhofer-AISEC/gallia

- **origin:** SecForCARs (secforcars.de)  
  https://youtu.be/OxkBNoBu8XQ
- **language:** Python $\geq$ 3.10 ($\text{latest - 1}$)
- **FOSS:** github:Fraunhofer-AISEC/gallia
- **PyPI:** pypi.org/projects/gallia
- **license:** Apache 2.0
- **maintainers:** @rumpelsepp, @peckto

→ modular tool for (automotive) penetration tests
penetration test

authorized simulated cyberattack on a computer system, performed to evaluate the security of the system

https://en.wikipedia.org/wiki/Penetration_test
Google search: “simpsons broken car”
Meta Challenges

- **raison d'être**: penetration tests on automotive ECUs (→ UDS)
- **postprocessing**: machine readable logs
- **reproducibility**: defined directory structure for artifacts
- **customizability**: modular software stack
- **extendability**: plugin interface and public API
- **software stack**: protocol stack needed
ONE DOES NOT SIMPLY

WRITE A UDS STACK
Status Quo
Architecture
Status Quo
Architecture

- Vehicle OBD-II Port
  - ECU 1
    - UDS Session 0x1
      - UDS Service 0x10
      - Identifier/Subfunction
    - UDS Session 0x2
      - UDS Service 0x11
      - Identifier/Subfunction
  - ECU 2
  - ECU ...
    - UDS Session ...
      - UDS Service ...

 Features

• **CLI tool**: provides ready to use scanners (nmap like);
  https://fraunhofer-aisec.github.io/gallia/uds/scan_modes.html

• **UDS stack**: including DoIP, ISO-TP, ...
  https://fraunhofer-aisec.github.io/gallia/transports.html

• **automation**: remote control power-supplies (e.g. power-cycle during scan)
  https://fraunhofer-aisec.github.io/gallia/automation.html

• **logging**: machine readable (JSON and SQL) logging format with tooling

• **virtual ECU**: for development
Status Quo

Architecture

testrun → artifacts

- rundir 1
- rundir 2
- rundir ...

- LOGFILE
- META
- PCAP
- ENV
- ...

Public information
Status Quo

Architecture

- gallia core
- plugins
- standalone
- CLI
Status Quo
Architecture

- UDS
- DoIP
- ISO-TP
- TCP
- CAN
- ECU
- scanner
- power supply

relationships:
- UDS uses DoIP, ISO-TP, TCP, CAN
- ECU contains scanner
- controls from scanner to power supply
- contains from ECU to plugins
Status Quo
Plugin Interface

```python
from argparse import Namespace
from gallia.command import Script

class HelloWorld(Script):
    """A hello world script showing gallia's plugin API."""

    COMMAND = "hello"
    SHORT_HELP = "say hello to the world"

    def main(self, args: Namespace) -> None:
        print("Hello World")

commands = [HelloWorld]
```

```
[tool.poetry.plugins."gallia_commands"]
    "hello_world_commands" = "hello_gallia.hello.commands"
```

https://fraunhofer-aisec.github.io/gallia/api.html
Random Technical Facts

- **poetry**: easy dependency management
- **asyncio**: async/await is used everywhere
- **fully typed**: passes mypy --strict
- **full config via argparse**: defaults via gallia.toml
- **entry points**: extendable using Python’s entry point API
- **transport URIs**: configure network stack on the CLI (verified by pydantic)

```bash
$ gallia foo --target doip://192.168.100.88:13400?src_addr=0x0e00&dst_addr=0x1243
```
Outlook
Outlook
MOAR!

- **power supplies**: Moar models!
- **transports**: Moar protocols (e.g. HSFZ)!
- **scanners**: Moar scanners!
- **scope**: Moar scope (plugins, scanning techniques, ...)
- **testing**: Moar breakage! Moar memes!
- **packages**: Moar distros!

Please test and contribute!
https://rumpelsepp.org/research/demos/gallia-fosdem23.mkv
Thank you!

Stefan Tatschner

<table>
<thead>
<tr>
<th>Github</th>
<th>@rumpelsepp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix</td>
<td>@rumpelsepp:hackbrettl.de</td>
</tr>
<tr>
<td>Mastodon</td>
<td>@<a href="mailto:rumpelsepp@mastodon.social">rumpelsepp@mastodon.social</a></td>
</tr>
<tr>
<td>LinkedIn</td>
<td>in/stefan-tatschner</td>
</tr>
</tbody>
</table>

https://www.aisec.fraunhofer.de/
Backup

Project Name

Hint!