

Program Thursday September 21st



09.15 Opening of the day

Riet

09.30 Key note – 'ChatGPT as a Hansken co-pilot' - by Hans Henseler

Hansken.ai

10.00 - 10.15 - Break

10.15 'More like this' (project) – demo
10.45 AI Law & digital forensics

Riet

Hansken.ai

10.15 Implementing Hansken – NLA
10.45 Academy Course Overview
11.00 Introducing team & processes

Klein Canvas

Hansken.startup

10.15 Interactive session on
containerization (Hansken in containers)

Truus

Hansken.infra

12.00 - 13.00 - Lunch

cap on/cap off quiz

Riet

Demo User interface

13.00 Toucan & Aardwolf
14.00 Break
14.15 Vehicle forensics

Riet

Hansken.related

13.00 Code notebook workshop
14.30 Break
14.30 Plugin creation walkthrough

Klein Canvas

Hansken.tech

13.00 Workshop Analysing & Relating
traces (incl Neo4j, brainstorm, sketching)
14.30 Break

Truus

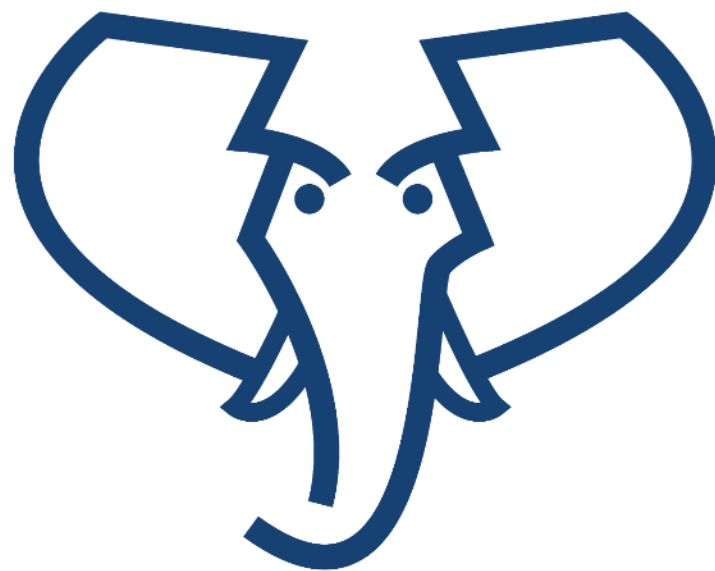
Hansken.next

15.30 - 16.00 - Closure of the Hansken.io - plenary

Riet

16.00- 17.30 - Good bye drinks

Klein Canvas



Hansken

The open digital forensic platform
investigate - innovate - share

Hansken code notebooks and extraction plugins

Hansken.IO, 21 september 2023, Amsterdam



Job

Remco

Netherlands Forensic Institute

Hans








Hansken extraction plugins

Walkthrough



Introduction

-  My nickname: remco-nfi
-  I am part of the Hansken development team
-  One of the main code contributors to the Extraction Plugin SDK
-  I am a software engineer
-  not a
 - doctor
 - case investigator
 - authority on digital forensics
 - ...



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions











Talking points for today

- **(3 min) Our fictional case...**
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions



Our fictional case

-  We are investigating a case
-  Our suspect mentions that zie was not at home and not using zir phone.
Is that True?
-  Evidence item: iPhone 15
- Imaginary & creative solution:
 -  Find out when the phone is charging, and at what rate?
 -  Find out when is the phone discharging, how fast is the phone discharging?
 -  Can we use the charging/discharging rate to prove our hypothesis?
- So...
 - How can we know this?
 - And if we know this, how can we add this knowledge to Hansken?



Our fictional case

- Background research: knowledgeC.db
 - <https://www.doubleblak.com/m/blogPosts.php?id=2>
 - <http://www.mac4n6.com/blog/2018/8/5/knowledge-is-power-using-the-knowledgecdb-database-on-macos-and-ios-to-determine-precise-user-and-application-usage>

KnowledgeC.db is a SQLite file on recent iOS versions that tracks lots of different activity on the device ranging from Battery Level and Bluetooth connections to which speaker is in use and what it is playing at any given time (...)

The database is located at **/private/var/mobile/Library/CoreDuet/Knowledge/knowledgeC.db** and is made up of 12 tables.

- **For this walkthrough: we will only focus on battery events!**



Talking points for today

- (3 min) Our fictional case...
- **(3 min) Extraction plugin concepts**
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions





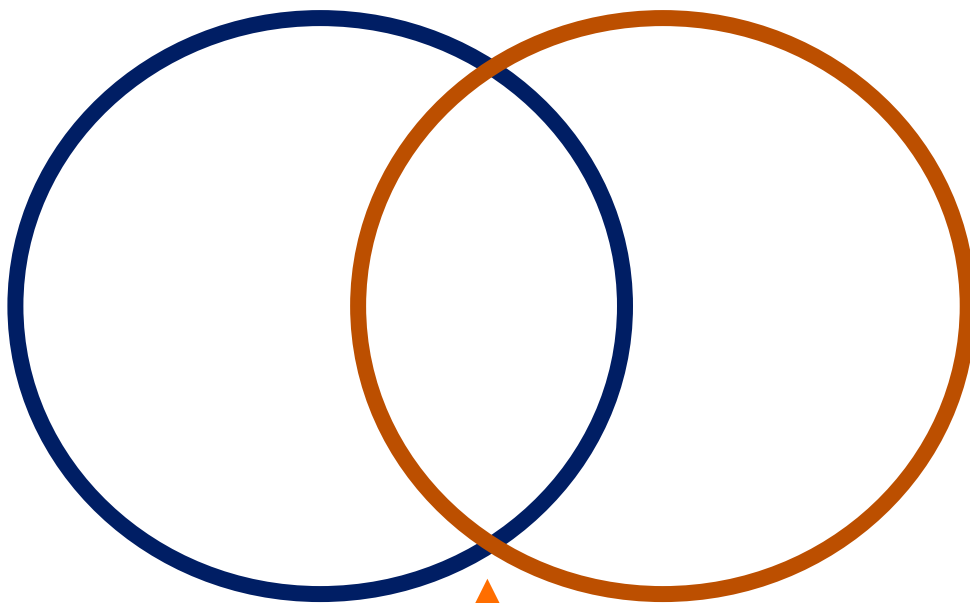
Extraction plugin concepts

Hansken.py

- Uses Hanskens REST API *
- Analysis of multiple traces *
- Visualizations *
- Can add data to Hansken*
- Manually repeatable *

Extraction Plugins

- * Runs inside Hansken *during* an extraction
- * Great for processing a lot of traces
- * Great for adding (a lot) of extracted data or child traces
- * Automatically repeatable
- * Full chain of evidence preserved
- * Run in total isolation (no network & root)



Extraction Plugins
can run as Hansken.py scripts

- * Follows the syntax of Hansken.py



Extraction plugin concepts

- **Extraction plugin Software Development Kit (SDK):**
- [Java API and tooling](#), to be able to write an extraction plugin with the [Java](#) programming language
- [Python API and tooling](#), to be able to write an extraction plugin with [Python](#) programming language
- [Test framework](#), to be able to test extraction plugins before they are used production
- [Documentation](#)
- [Examples](#) (Github)
- Community: [Extraction plugins made by the community on Gitlab](#)



Remainder: hands on
guided by this slidedeck

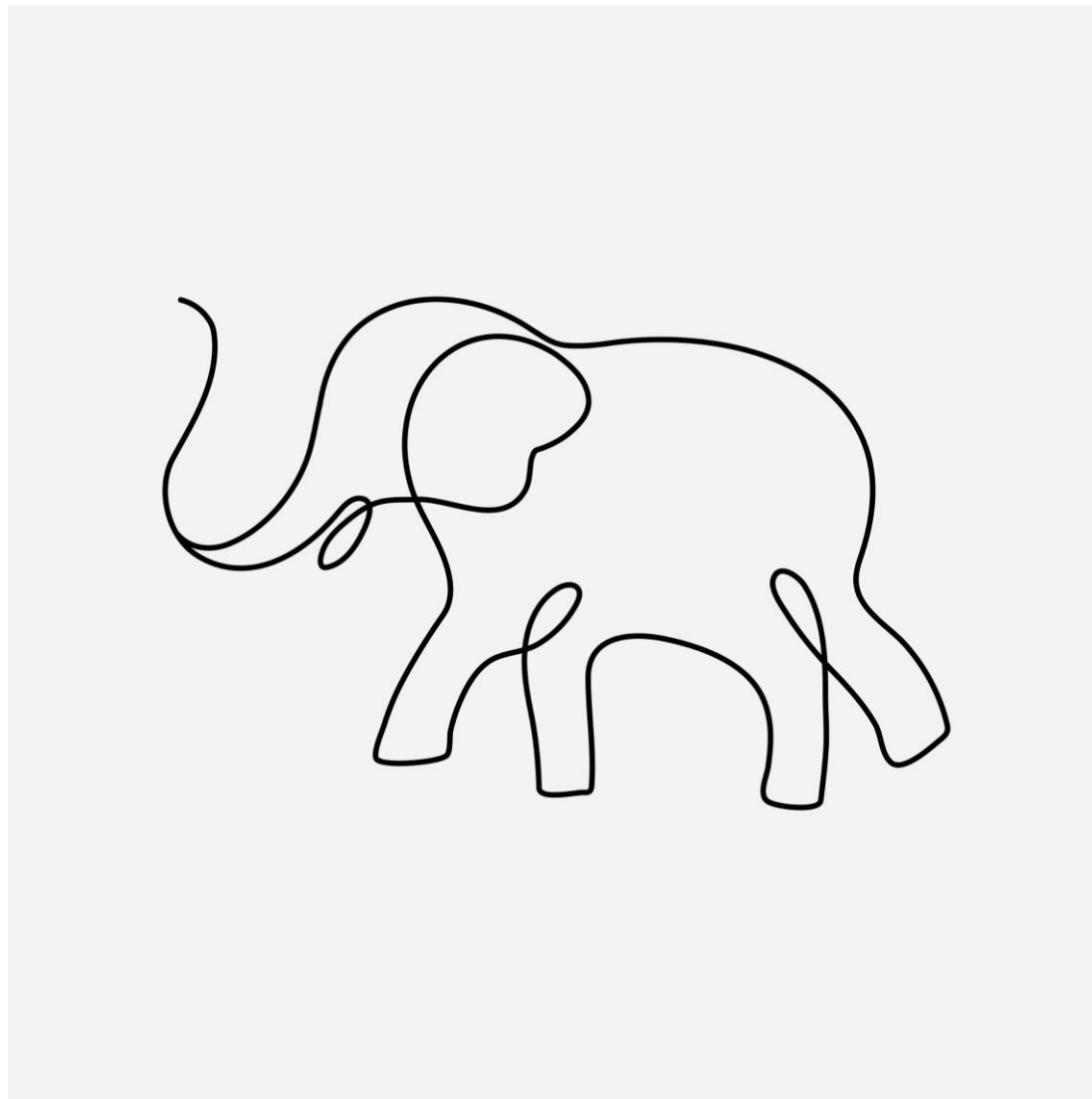


Image by rawpixel.com on Freepik



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- **(5 min) Extraction plugin template**
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions



Extraction plugin template

Steps

1. Clone an [template \(empty\) plugin from Github](#)
2. Build the template plugin
3. Publish the plugin
4. Refresh the plugin list in Hansken



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- **(5 min) Create logic: Parse data and transform to Hansken types**
- (5 min) Link logic to plugin
- (5 min) Run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions



Create logic: Parse data and transform to Hansken types

Steps

1. Determine how to store battery events in Hansken (trace model)
2. Transform database to battery events (SQLite query)
3. Add method to plugin



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- **(5 min) Link logic to plugin**
- (5 min) Run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions





Link logic to plugin

Steps

1. Plugin structure
2. Update plugin info
 1. Naming convention
 2. Matcher (common PITA)
 3. Plugin resources (common PITA2)
3. Implement process method



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- **(5 min) Run plugin with Hansken.py**
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions





Run plugin with Hansken.py

Steps

1. Upload test data to Hansken (single file)
2. Run plugin from the IDE
 1. Determine program arguments
 2. Define run profile
 3. Run!
3. Verify results



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Run with Hansken.py
- **(3 min) Add tests to our code**
- (5 min) Run the plugin during an extraction
- (3 Min) Wrap up and questions





Add tests to our code

Steps

1. Why?
2. Have a look at the documentation
3. Add test data to plugin
4. Generate & verify baseline results



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Demo: run with Hansken.py
- (3 min) Add tests to our code
- **(5 min) Run the plugin during an extraction**
- (3 Min) Wrap up and questions





Run the plugin during an extraction

Steps

1. Package the plugin (`tox -e package`)
2. Refresh plugin list in Hansken
3. Run an extraction
4. Inspect results
5. Inspect chain of evidence
6. BONUS: Plugin-card



Talking points for today

- (3 min) Our fictional case...
- (3 min) Extraction plugin concepts
- (5 min) Extraction plugin template
- (5 min) Create logic: Parse data and transform to Hansken types
- (5 min) Link logic to plugin
- (5 min) Demo: run with Hansken.py
- (3 min) Add tests to our code
- (5 min) Run the plugin during an extraction
- **(3 Min) Wrap up and questions**



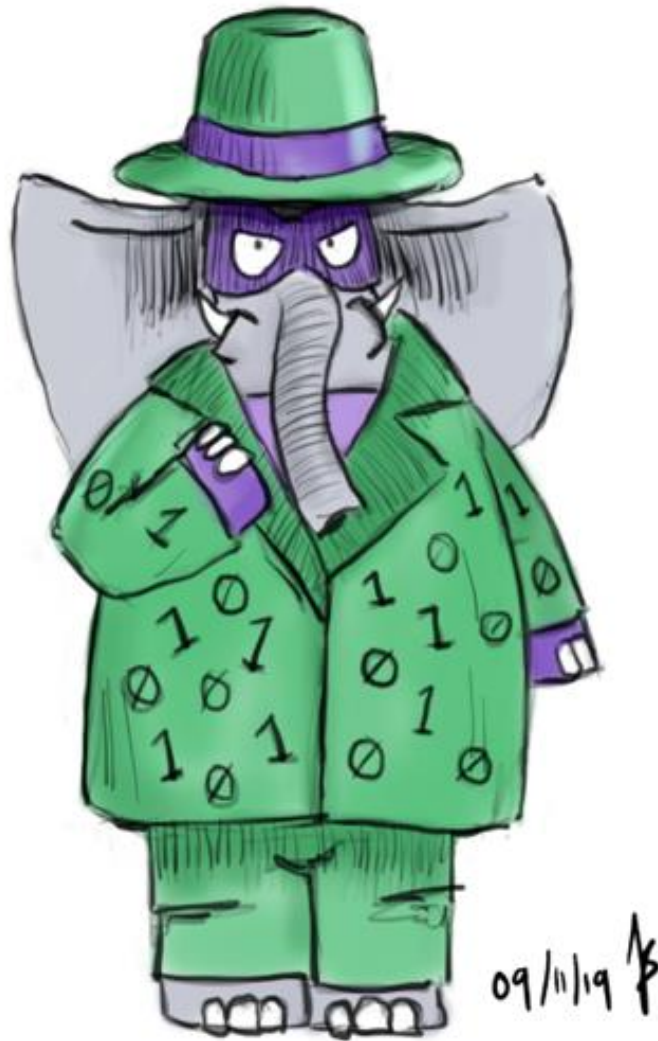
Wrap up

1. We learned:

1. What extractions plugins are, how they relate to Hansken.py use cases
2. Where to find plugins created by other community members
3. How to create, test, and run a new plugin by downloading a plugin template and add an actual implementation

2. Extraction plugins in practice

1. Starting to prove itself in action
2. Technology preview
please report bugs, and share your experience!
3. Debugging can be difficult -> “why does my plugin not work”?
4. Needs to grow!





Thank you for your attention

