Deliverable

Abstract
This report provides an overview of tools created within the TITANIUM project. This deliverable is composed by an operation manual and a user guide of each tools developed in TITANIUM project. These documents assist administrators and practitioners to carry out the Field Lab deployments and end-user evaluation.

Keyword list
Authors

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
<th>Partner</th>
<th>Contribution</th>
</tr>
</thead>
<tbody>
<tr>
<td>R. Orduna</td>
<td>Deliverable Manager</td>
<td>VICOM</td>
<td>Deliverable 6.2</td>
</tr>
<tr>
<td>B. Haslhofer</td>
<td>Contributor</td>
<td>AIT</td>
<td>Graphsense</td>
</tr>
<tr>
<td>S. Raaijmakers/ E. Cramer</td>
<td>Contributor</td>
<td>TNO</td>
<td>DW Monitor, DW Market Scraper, Self-Learning Scraper, Captcha Solver, Multimodal Vendor Profiling</td>
</tr>
<tr>
<td>E. Haspels</td>
<td>Contributor</td>
<td>COB</td>
<td>Cointel</td>
</tr>
<tr>
<td>T. Gloe / J. Hasse</td>
<td>Contributor</td>
<td>DEN</td>
<td>Image Similarity, Blockchain Investigator</td>
</tr>
</tbody>
</table>

Document Approval

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
<th>Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>N. Päivinen</td>
<td>Work Package Reviewer</td>
<td>NBI</td>
</tr>
<tr>
<td>R. King</td>
<td>TITANIUM Reviewer</td>
<td>AIT</td>
</tr>
</tbody>
</table>

Distribution

<table>
<thead>
<tr>
<th>Person</th>
<th>Role</th>
<th>Partner</th>
</tr>
</thead>
<tbody>
<tr>
<td>R. King</td>
<td>PC</td>
<td>AIT</td>
</tr>
</tbody>
</table>

Revision History

<table>
<thead>
<tr>
<th>Version</th>
<th>Status</th>
<th>Author</th>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>draft</td>
<td>R. Orduna</td>
<td>September 2018</td>
<td>Basic setup</td>
</tr>
<tr>
<td>0.2</td>
<td>draft</td>
<td>R. Orduna</td>
<td>October 2018</td>
<td>Review and partner input</td>
</tr>
<tr>
<td>0.3</td>
<td>draft</td>
<td>R. Orduna</td>
<td>October 2018</td>
<td>Review and partner input</td>
</tr>
<tr>
<td>0.9</td>
<td>Validated</td>
<td>R. Orduna</td>
<td>2018-10-30</td>
<td>Integration of all comments and appendices</td>
</tr>
<tr>
<td>1.0</td>
<td>Submitted</td>
<td>R. King</td>
<td>2018-10-31</td>
<td>Submitted version</td>
</tr>
<tr>
<td>1.1</td>
<td>Revision</td>
<td>R. Orduna</td>
<td>2019-03-25</td>
<td>Revised version</td>
</tr>
<tr>
<td>1.2</td>
<td>Revision</td>
<td>R. King</td>
<td>2019-04-11</td>
<td>Fixed broken headings, removed redundant figure</td>
</tr>
</tbody>
</table>
Executive Summary

The Horizon 2020 project TITANIUM will deploy the forensic tools and services developed in WP4 and WP5 to the associated Law Enforcement Agency stakeholders, evaluate their usage in a realistic configuration, and validate their effectiveness in reducing cost and effort in criminal investigations.

The TITANIUM partners will organise and document two rounds of Field Labs during the project. The Field Labs serve as the testing and validation mechanism in which the LEA partners will use and evaluate the tools and technologies developed in the project in near-operational conditions.

Deliverable 6.2 describes the context, requirements and conditions that are needed and set by the technical partners so that the various tools to be installed and/or used on the stakeholders’ premises. In addition, it also serves as a training manual of the application that will assist Field Lab users to familiarize themselves with application and its functionalities. This documentation is appended to this Deliverable, organized by tools.

This version of the Deliverable (2018-10-31) provides the documentation for those tools that will be deployed in the first Field Lab. An updated version of the Deliverable containing documentation for additional tools will be provided before the second Field Lab.
Field Labs Context

Design of architecture

TITANIUM partners have chosen a general policy to develop their tools, on-premises software solution, so the idea is to install all the required application directly in a customer’s in-house server and computing infrastructure. With this policy the customer become the responsible for the security availability and overall management of on-premise software.
This policy results suitable for the TITANIUM stakeholders represented by the Law Enforcement Agencies (LEAs) that must manage and analyse private and sensitive data, and they cannot risk getting out this information to third parties, even if the data or analysis are encrypted.
The final idea in the TITANIUM project is to join the tools/microservices in some cluster, like a coherent set of interoperable tools, each one accessible with an appropriate interface. To achieve this task, the interoperability among the developed tools is considered and “formats” for data that can be exchanged and manipulated in the related tools are planned.
The data used in the Field Labs will be synthetic data or on-premise data whenever possible. In case that a tool requires real data, this information will be accessible only for LEA partners and restricted for all others (investigator, etc.). Furthermore, will be impossible to share real data among LEA partners. All these analysis and results should be anonymized before sharing to ensure privacy and confidentiality.
The on-premises software policy justifies the operation manuals attached at the end of this document that explain how to install and configure all the available tools.

Risk management

Tools should be working like on-premise solution, so LEA partners can use real data because all the analysis and results remain in their own “hands”. When the tools are not ready for this kind of policy they work on cloud and so they need synthetic data, as it is described in the Deliverable 4.4.
When synthetic data are not enough by quantity or complexity and are not suitable for a specific tool the idea is to use real data that should be anonymized (or pseudonymized) properly as it is specified in the General Data Protection Regulation (GDPR). Furthermore, each LEA investigator will be aware of legal obligations when using the provided tools on real data.
The investigator also must agree to receiving email notifications and the terms when using the TITANIUM tools in the provided form (e.g., private cloud). Given feedback should be allowed to be used within Titanium project.
All tools implemented in TITANIUM are restricting and follow the applicable laws (EU, national).

Relationships between tools and use cases

To implement their tools, the partners of TITANIUM are started analysing a Generic Use Case (GUC). The generic use case is represented by criminals that offer illegal drugs on a dark market or forum (like Grams, AlphaBay, SilkRoad 2, etc.), so the LEA agents start to gather evidence from the dark market. Through the gathered information, like nicknames, virtual currency addresses, email addresses the agents want to identify seller. If a transaction is done, they want to follow the money to identify the criminal and determine the buyer. The agents also want to relate seller among the different market place and discover information/relatıon between seller and buyer (for example if they are involved in previous discussion in a forum). It is also important to extract information from confiscated devices or his virtual currencies.
The complete version of the generic use case can be found here: https://redmine.titanium-project.eu/dmsf/files/237/view
<table>
<thead>
<tr>
<th><strong>Generic Use Case Step</strong></th>
<th><strong>Which tool is related to</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>A criminal offers product for sale in a dark marketplace (e.g., Dream market, The Trade Route, Tochka Free Market)</td>
<td>DW Market Scraper, Self-learning Scraper</td>
</tr>
<tr>
<td>Gather evidence from the dark marketplace and blockchain network (discussions, sales advertisements)</td>
<td>DW Market Scraper, Self-learning Scraper, DW Monitor, Image Similarity</td>
</tr>
<tr>
<td>Track the flows of criminal’s virtual currencies (maybe different currencies, cross-ledger analysis needed): where have the funds come from, where they are going to? Where and when are the funds cashed out?</td>
<td>GraphSense, Cointel, Blockchain Investigator</td>
</tr>
<tr>
<td>How much financial profit did the criminal get from the deals (the total amount of virtual currencies earned)?</td>
<td>GraphSense, Cointel, Blockchain Investigator</td>
</tr>
<tr>
<td>Did the same seller operate in different marketplaces? How much has he been selling stuff? For how long the selling activities have been going on?</td>
<td>DW Market Scraper, Self-learning Scraper, DW Monitor</td>
</tr>
<tr>
<td>Identify the customers (maybe arrest them also for other crimes) What discussions can be found between the criminal and his customers?</td>
<td>DW Market Scraper</td>
</tr>
<tr>
<td>Find links between criminals acting in dark marketplaces (find criminal accomplices): use every possible data source (discussion forums, virtual currency transaction flows)</td>
<td>DW Market Scraper, Self-learning Scraper, Image Similarity</td>
</tr>
<tr>
<td>The Police arrests the criminal, confiscates his devices (smart phone, laptop, USB mass storage devices) and seizes his virtual currencies (Bitcoin wallets)</td>
<td>Wallet Investigator</td>
</tr>
<tr>
<td>Find evidence of criminal activities from the devices (traces left by applications, e.g., wallet software, visits to dark marketplaces with Tor browser)</td>
<td>Wallet Investigator</td>
</tr>
</tbody>
</table>

All tools are progressively updated with the identification of new use cases, less generic and more focused in real world behaviour and concrete usage, like indicated in the Deliverable 3.2. This use cases, individuated thanks to the help of the LEA agents, allowed partners to enrich their own tools with new functionalities.

To explicate how these tools could be applied in a specific use case and which operation are available, at the end of this deliverable the user guides are attached.
Field Labs Planning

What
The TITANIUM partners will organise and document two rounds of Field Labs during the project. The Field Labs serve as the testing and validation mechanism in which the LEA partners will use and evaluate the tools and technologies developed in the project in near-operational conditions.

Why
The TITANIUM partners will showcase the tools to the participants using example investigation questions based on the Generic Use Case (presented in a separate document). The showcase investigation gives the participants ideas for using the tools in their own line of work.

Where
The Field Labs will be installed in LEA partner premises in Finland, Germany, Spain and Austria.

Institution and location of the Field Labs, LEA partners:
- National Bureau of Investigation (NBI) – Vantaa, Finland
- Bundeskriminalamt (BKA) – Berlin, Germany
- Ministerio del Interior (MIR-PN) – Madrid, Spain
- Bundesministerium für Inneres (BMI) - Wien, Austria

When
The first round of Field Labs will be arranged between January and March 2019. The participants are Law Enforcement Officers from Germany, Austria, Spain and Finland; the total number of participants is expected to be at least 40. Cybercrime investigators having at least some experience on investigations dealing with virtual currencies and / or darknet marketplaces are invited to participate.

The Field Labs start with a kick-off day in January 2019 arranged simultaneously in the participating four countries. The kick-off day prepares the participants into unassisted usage of TITANIUM tools. The participants are informed about the TITANIUM project, goals of the Field Labs, the legal and ethical background as well as the practical arrangements for the Field Labs and shown how the tools are used. After the kick-off day, the participants are encouraged to test the tools by themselves and give feedback about them. The feedback is collected by KIT, NBI, BKA, BMI and MIR-PN, and analysed in D6.3: Initial Field Lab studies analysis and evaluation results, due in M24. The participants can require assistance in using the tools from the TITANIUM partners during the whole Field Lab period (January - March 2019).

The second round of Field Labs will be arranged between May and December 2019. The final results of both rounds will be summarised in D6.4: Final Field Lab analysis and evaluation results due in M36. The whole Field Labs process adheres to the ethical principles approved by the Ethics Advisory Board of TITANIUM.

How
The participants can use real case data during the Field Labs if it is legally and ethically possible (see D2.3: Legal and ethical conditions for the conduct of the Field Labs); otherwise, synthetic data provided by the TITANIUM partners will be used.
Index of Operations Manuals and User Guides

1    GraphSense [AIT]
Operation Manual..................................................................................................................8
User Guide..........................................................................................................................14

2    Image Similarity [DEN]
Operation Manual..................................................................................................................25
User Guide..........................................................................................................................31

3    Blockchain Investigator [DEN]
Operation Manual...............................................................................................................35
User Guide..........................................................................................................................41

4    Cointel [COB]
Operation Manual..................................................................................................................48
User Guide..........................................................................................................................88

5    Wallet Investigator [DEN]
Operation Manual...............................................................................................................100
User Guide..........................................................................................................................105

6    DW Monitor [TNO] .................................................................109
7    DW Market Scraper [TNO] .........................................................115
8    Self-Learning Scraper [TNO] .......................................................119
9    Captcha Solver [TNO] ..............................................................123
10   Multimodal Vendor Profiling [TNO] ...........................................128
GraphSense
Operation Manual
AIT Austrian Institute of Technology
October 2018
Table of Content

1. Functional description ................................................................. 10
2. Requirements .............................................................................. 11
   2.1. Hardware Requirement .............................................................. 11
   2.2. Software Requirement and condition ........................................ 11
   2.3. Data requirements .................................................................... 11
3. Installation Guide .......................................................................... 12
   3.1. Chain of dependences ............................................................... 12
4. Configuration or Tools ................................................................... 13
   4.1. Cluster configuration ................................................................. 13
1. Functional description

1.1 What does the tool do?

GraphSense is an open source platform for analyzing cryptocurrencies such as Bitcoin ([http://graphsense.info](http://graphsense.info)). The platform has the following features and functionality:

- **Address Clustering**: partition the set of addresses observed in a cryptocurrency ecosystem into maximal subsets (clusters) that are likely to be controlled by the same real-world entity.

- **Micro- and Macroscopic Analysis**: inspect main cryptocurrency entities (block, transaction, address) and compute summary statistics over the entire blockchain.

- **Network Perspective**: apply a network-centric perspective and traverse currency flows between addresses and clusters.

- **Horizontal Scalability**: cryptocurrency blockchains are growing and new currencies appear on the horizon. To make GraphSense future-proof, it is built on Apache Spark and Apache Cassandra for horizontal scalability.
2. Requirements

2.1. Hardware Requirement

GraphSense is built on scalable and distributed cluster technology (Figure 1).

![GraphSense Architecture](image)

Figure 1: GraphSense Architecture

2.2. Software Requirement and condition

- Apache Cassandra
- Apache Spark >= 2.2
- Scala
- Python 3
- docker (optional)

2.3. Data requirements

- Blockchain data
- Tags for cryptocurrency addresses
3. Installation Guide

3.1. Chain of dependences

Graphsense consists several of software components. They must be setup and/or executed in the following order:

- **bitcoin-client**: a Docker container encapsulating the most-recent Bitcoin-core client version ([https://github.com/graphsense/bitcoin-client](https://github.com/graphsense/bitcoin-client))
- **datafeed**: a component for ingesting raw blockchain data and currency exchange rates into Apache Cassandra ([https://github.com/graphsense/graphsense-datafeed](https://github.com/graphsense/graphsense-datafeed))
- **transformation**: a Spark pipeline for computing statistics and network representations from raw blockchain data stored in Apache Cassandra ([https://github.com/graphsense/graphsense-transformation](https://github.com/graphsense/graphsense-transformation))
- **REST-api**: an API for retrieving data from the underlying Cassandra store ([https://github.com/graphsense/graphsense-REST](https://github.com/graphsense/graphsense-REST))
- **dashboard**: a user-interface allowing search, inspection, and traversal of cryptocurrency entities

The GraphSense transformation pipeline reads raw data, which is ingested into Cassandra by the graphsense-datafeed component, and computes de-normalized views, which are again stored in Cassandra. Access to computed de-normalized views is subsequently provided by the GraphSense REST interface, which is used by the graphsense-dashboard component.
4. Configuration or Tools

For configuration of the individual components please refer to the Readme files in the GitHub repositories. The Bitcoin Core client must run as a full node (txindex=1 in bitcoin.conf).

4.1. Cluster configuration

Hardware setup
The cluster currently consists of 5 nodes – 1 master and 4 worker nodes. In total, the cluster provides a total of 48 CPUs, 1 TB memory, 40TB HDD storage, and 16TB SSD storage. All nodes are connected via a 10GBase-T Ethernet connection.

Software setup

Master and workers:
- Debian 8
- Java (Oracle) 1.8.0_151
- Apache Spark 2.2

Workers only:
- Apache Cassandra 3.11.3
Table of Contents
1  System function................................................................................................. 16
1.1  Landing page.................................................................................................. 17
1.2  Block view....................................................................................................... 18
1.3  Transaction view.............................................................................................. 19
1.4  Address view................................................................................................... 20
1.5  Cluster View.................................................................................................... 22
2  Visualization ....................................................................................................... 24
3  User profile ......................................................................................................... 24
4  Bad usage ............................................................................................................ 24
1 System function

GraphSense is a graph-centric analytics solution for digital currencies and is built on a horizontally scalable data processing and analytics platform. It allows users to explore transactions and follow the money flow, facilitates analytics by semantically enriching the transaction graph and guides analysts to anomalous data patterns. GraphSense implements one specific type of graph-centric perspective, the so-called address graph. Each vertex represents a Bitcoin address identified by a 26-35 alphanumeric character string. Each edge represents a transaction identified by its 32-bit hash and carries additional properties, such as the transaction timestamp and transaction value.

The address graph provides a useful abstraction for manually exploring and tracing flows of currency units through the Bitcoin ecosystem and identifying recurring patterns in transactions, such as frequently used target addresses. In combination with address clustering and node enrichment techniques, it can also be used for finding known, deanonymized addresses in the digital trace of digital currencies.

![Data enrichment process](image)

Figure 2: Data enrichment process

Graph construction and analytic procedures in GraphSense are implemented in Apache Spark since the volume of Bitcoin transactions already touches the physical boundaries of single-machine analytics environments. As of October 2018, the Bitcoin address graph has the following size (height 545,802):

- 423,652,330 nodes and
- 2,536,254,029 edges.

Thus, for a positive and enjoyable user experience, all data points are pre-computed and stored in a de-normalized form in Apache Cassandra.

The following provides a documentation for the GraphSense user interface (GraphSense Dashboard, available at [https://github.com/graphsense/graphsense-dashboard](https://github.com/graphsense/graphsense-dashboard); version 0.3.2), its features include:

- **search of transaction graph**: using a Google-like search interface, users can search for Bitcoin blocks and transactions as well as for addresses.
- **exploration and traversal of transaction graph**: all Bitcoin entities (blocks, transactions, addresses) are exposed as first class resources identified by a unique URI. Relationships between entities are represented as HTTP links.
- **Inspection of address clusters**: each address is assigned to a cluster, which can be further inspected.
• **exploration of address graph:** for each address, a reduced ego-net graph is displayed, which allows users to inspect and traverse the address graph.

1.1 Landing page

The landing page provides an input field, which allows the search for the following entities:

• Block: Integer number/block number
• Transaction: Hexadecimal string of a Bitcoin transaction hash
• Address: Bitcoin address, i.e. an identifier of 26–35 alphanumeric characters, beginning with the number 1 or 3

After clicking the “Go” button, the appropriate view for the specified input will be rendered.

![Figure 3: GraphSense landing page](image-url)
1.2 Block view

The block view can be accessed via the URL http://DOMAIN/block/<BLOCK_NUMBER>. On the left side of the block view, summary statistics for a shown for the specified block (Height, Timestamp and the number of transactions in the block). In the main panel a table of all transaction of the specified block is shown, which includes the following information:

- Transaction hash
- Number of inputs
- Number of outputs
- Transaction Fee
- Total transaction value (BTC, Euro or US Dollar; can be selected using the drop-down list in the top right corner)

![Figure 4: GraphSense block view](image-url)
1.3 Transaction view

The transaction view provides the following summary statistics for a single transaction in the left panel:

- Received time (i.e., the creation time of the corresponding block)
- Block number
- Total value
- Transaction Fee

In the main panel all inputs and outputs are listed for the specified transaction hash (address string and value). The transaction view can be accessed using the URL http://DOMAIN/tx/<TX_ID>.

Figure 5: GraphSense transaction view
1.4  Address view

All relevant information for a single address are rendered in the address, available through http://DOMAIN/address/<ADDRESS_STRING>. In the left panel the following statistics are shown for a specified address:

- Number of transactions (black), number of incoming transactions (green) and number of outgoing transactions (red)
- Timestamp of first transaction
- Timestamp of last transaction
- Activity period
- Total received amount
- Final balance

Additionally, basic cluster statistics are provided (purple coloured header) reusing the same fields as above and is linked to the corresponding cluster view (see next section) for the specified address.

The main panel consists of three tabs:

1. Visualization of the address graph (cf. Figure 6)
2. List of transactions (cf. Figure 7)
3. List of tags (cf. Figure 8)

Figure 6: Address view – address graph
Figure 7: Address view – list of transactions

Figure 8: Address view – list of tags
1.5 Cluster View

Each address is assigned to a cluster, which can be further inspected in the cluster view (http://DOMAIN/cluster/<CLUSTER_ID>). This view shows again the summary statistics for the cluster. The three tabs in the main panel show a visualization of the cluster graph, the list of addresses in the cluster and a table of known tags.

![Figure 9: Cluster view – cluster graph](image)

![Figure 10: Cluster view – list of addresses](image)
Figure 11: Cluster view – list of known tags
2 Visualization

In the address view, a reduced ego-net graph is displayed, which allows users to inspect and traverse the address graph in an interactive manner. The graph visualization is implemented in JavaScript using D3.js (https://d3js.org). The node size can be scaled either using the total received amount or the final balance, and the edges can be labelled with the number of transactions or the estimated currency flow (in BTC, EUR, or USD). The nodes in the address graph are distinguished by the following three categories (node colours):

- no tag available (gray),
- explicitly tagged address (darkgreen),
- implicitly tagged address through multiple-input clustering (lightgreen),

(see Figure 12).

Figure 12: ego-net graph for a specified address

A similar visualization is also used for the cluster graph.

3 User profile

In the GraphSense Dashboard user interface all users have the same access rights. The current version does not implement any separate user administration. Developers can access JSON data in restricted form by using direct calls to the provided REST API (see https://github.com/graphsense/graphsense-REST)

4 Bad usage

Known Limitations/Bugs:

- Search for tags as shown in the input filed on the landing page is not implemented
- Empty input in the search field on the landing page is not handled properly
- The new SegWit address format for bc1-addresses (BECH32) as specified in BIP0173 is supported by the GraphSense backend, but not yet accepted in the search field
- Download of the nodes list in the address/cluster view does not work
Image Similarity
Operation Manual
dence GmbH
October 2018
## Table of Content

1. Functional description ................................................................. 27
2. Requirements .................................................................................. 28
   2.1. Hardware Requirements .......................................................... 28
   2.2. Software Requirements .......................................................... 28
   2.3. Data requirements .................................................................. 28
   2.4. Legal considerations .............................................................. 28
3. Installation Guide ........................................................................... 29
   3.1. Installation of the microservice API .......................................... 29
   3.2. Installation of the user interface for development purposes .......... 29
4. Configuration .................................................................................. 30
   4.1. Microservice API ................................................................... 30
   4.2. User interface for development purposes ................................... 30
5. Error Management ........................................................................ 30
6. Backup and Update ........................................................................ 30
   6.1. User interface for development purposes ................................... 30
   6.2. Microservice API ................................................................... 30
1 Functional description
Dence image similarity is a microservice/tool to correlate images which share common attributes. For a given data set of images, the tool analyses each one and applies multiple techniques to extract information. These can be based on meta data, internal structure, visual characteristics or abstract concepts which are inherent to the image. The characteristics are stored in a scalable database and queries can be executed to match all similar images for a given image or characteristic.

The tool is developed to support crawlers and scrapers and provide additional links between sources in the clear and dark web. E.g. vendors of a darknet marketplace are incentivised to always upload up to date product images for their advertisements. Metadata or forensic characteristics hidden within those files may allow to cross correlate these images and attribute them to a single individual. Based on traces in media files, a vendor selling on two market places under different pseudonyms could be linked.

The tool consists of two components:

- API microservice
- User interface for development purposes

The current prototype implements detecting similar images by perceptual similarity. Similarity by analysis of meta data and internal structure is under development. The notion of concepts will be integrated as a follow-up. Some configuration options might not be exposed yet for the prototype.
2 Requirements

2.1 Hardware Requirements
The user interface component does not have any strong hardware requirements. Access using a common computer or mobile phone suffices.
For the API microservice, the hardware requirements strongly depend on the size of the data set to be indexed. For less than 100,000 images, a normal personal computer suffices. For indexing images in the order of millions of images, the service needs to be run on multiple computation nodes. A current, single core needs approx. 1-5 seconds to process a single image, depending on the selection and configuration of perceptual hashes.
For the test setup, the service indexed 17000 images which created ~100 MB of basic data, ~700 MB for preview thumbnails and ~50 MB for the hash databases.

2.2 Software Requirements
The user interface can be accessed through an up to date internet browser, which has the ability to disable the CORS security feature (e.g. Google Chrome).
The API microservice is a collection of Docker containers, merged by a docker-compose definition. Thus, the host requires the Docker engine (>= version 18.02.0) and docker-compose (>= version 3.6) to run the microservice API.

2.3 Data requirements
The microservice requires images as input to be added to the (internal) database. This can be a custom set of images collected by the user, or this can be generated by a scraper/crawler from an internet source. All other requirements are already embedded into the Docker containers.

2.4 Legal considerations
The image similarity search does have legal difficulties, because images do contain personal data and may be protected by copyrights. These are similar to the legal issues with scrapers and crawlers, which the image similarity search inherits when used in conjunction with these technologies. Solving this balance is in the hand of the investigator, by a) only adding images to the database which are necessary and potentially relevant to the case and b) selecting only analysis techniques which can be lawfully executed in the context of the individual case (e.g. selecting to store the serial key in clear, as hash or not to store it; selecting to store preview images or not to store them).
3 Installation Guide

3.1 Installation of the microservice API

Install Docker. Please refer to the Docker installation guide for your chosen operating system:
   > https://docs.docker.com/install/
If not included in your Docker installation, install docker-compose for your operating system following the install guide:
   > https://docs.docker.com/compose/install/
Receive the docker-compose.yaml example file for the image similarity search by contacting dence at info@dence.de. It is planned to deploy this file into the TITANIUM git.
Adjust the docker-compose file to your needs, e.g.
   > Provide a SSL certificate for the grpc web proxy
   > Adjust port settings
   > Adjust storage paths for basic data, preview images and hash databases
Start the microservice API by invoking:
   docker-compose up -d

3.2 Installation of the user interface for development purposes

   > Receive the nodejs implementation for the development UI by contacting info@dence.de. It is planned to deploy this along with the docker-compose pipeline in the future.
   > Run: npm install
   > Run: npx webpack
   > Run: node app.js
   > Open your browser on localhost:8080 to test, if the static UI files are available
4 Configuration

4.1 Microservice API
The microservice API can be configured by editing the docker-compose.yaml. The user is able to specify the storage paths on the host machine by adjusting the Docker volume mapping. This can be done for basic data (msv-postgre), image preview (msv-server) and hash databases (msv-hash-*). Parameters for subservices can be adjusted by editing the Docker ‘command’ directive and adjusting command line parameters for the internal command line subservices.

4.2 User interface for development purposes
The frontend in the browser cannot be further configured. However, you may want to change the listening port in ‘app.js’ or the API endpoint host in ‘clientapp/client.js’. After making any change, you need to rerun the installation steps and server (npx webpack; node app.js). Please remember to clear your browser cache on any change of the statically served files.

5 Error Management
Client-facing errors are shown in the browser, the javascript console of the browser or in the console of the serving node application. Error messages for the microservice API are stored in Docker logs of the running container. The main entrypoint is the msv-server container. To retrieve the log, run:
> docker logs msv-server

6 Backup and Update

6.1 User interface for development purposes
The browser application is state-less. No data is stored which needs backup. Updates can be requested by contacting dence through info@dence.de. In the future, the UI will be shipped along as a docker container and the update procedure is the same like for the microservice API.

6.2 Microservice API
The microservice stores persistent data on the host machine using Docker’s volume mapping features. The paths are specified in the docker-compose.yaml file. Please execute regular backups of these folders as well as backups before every program update. You may update the pipeline to the newest release by invoking:
> docker-compose down
> docker-compose pull
> docker-compose up -d
Table of Content
1. User profile ........................................................................................................ 33
2. System function .................................................................................................... 33
2.1. User interface for development purposes ....................................................... 33
2.2. Microservice API .............................................................................................. 34
1 User profile

The dence image similarity toolset does not implement user profiles. Access restrictions are implemented and enforced by externally restricting access to individual deployments of this tool. Privacy options can be configured individually for each deployment.

2 System function

2.1 User interface for development purposes

The user interface implements a mechanism to query an existing deployment of the microservice API. In the current prototype, the user specifies an internet URL for an image to analyse. The service responds with a list of images, which are considered to be similar. The result is presented as list which can be sorted by similarity score relating to the queried image.

Figure 13: Screenshot of user interface for development purposes. Users can query an internet URL and will receive a list of similar images. A click on ‘Similarity’ table header will sort the table by similarity score.
2.2 Microservice API

The microservice API is an endpoint for advanced users or for the integration with crawlers/scrapers. The API is implemented with grpc/protobuf. Developers should consult grpc documentation to generate a API client binding for their favourite programming language (https://grpc.io/docs/). It exposes three simple API calls: query, add and info.

- **Query**: Provide an image binary or an image url to analyse. Return unique IDs for all similar images and their similarity score. The queried image will not be stored persistently and removed as soon as the request completed.

- **Add**: Add an image binary or an image url to be analysed and stored into the internal database.

- **Info**: Query basic information about a stored image. It provides image dimensions, file size, preview image and known URLs for a unique image ID.

For technical details, please have a look at the grpc service definition within the source code.
# Operation Manual

## Table of Content

1. Functional description ........................................................................................................ 37
2. Requirements ...................................................................................................................... 37
   2.1. Hardware Requirements ............................................................................................... 37
   2.2. Software Requirements ............................................................................................... 37
   2.3. Data requirements ....................................................................................................... 37
   2.4. Legal considerations .................................................................................................. 37
3. Installation Guide ............................................................................................................... 38
   3.1. Installation of dence blockchain investigator .............................................................. 38
   3.2. Installation of the full Bitcoin blockchain ..................................................................... 38
   3.3. Installation of currency exchange rates ....................................................................... 39
4. Configuration ...................................................................................................................... 40
5. Error Management ............................................................................................................ 40
6. Backup and Update ........................................................................................................... 40
1 Functional description
dence blockchain investigator is a desktop UI application to track and analyse Bitcoin transactions in an offline, isolated environment. It focuses on a deep technical analysis of transactions, enabling investigators to derive characteristics for models, folding and clustering of data. Investigators have full control which data is processed and added into the tool.

2 Requirements

2.1 Hardware Requirements
- Microsoft Windows 64-bit version 7 or later
- A current processor with at least two cores
- At least 8 GB of RAM
- At least 1024 x 768 display resolution
- For demo blockchain: 500 MB of disk space
- For full Bitcoin blockchain: 1 TB SSD drive
- Internet access to download initially download 300 GB for the full Bitcoin blockchain and about 2 GB per day for updates

2.2 Software Requirements
- Microsoft Windows 64-bit operating system version 7 or later
- (Program versions for MacOS or Linux operating systems might be available on request)
- A current internet browser to access update resources

2.3 Data requirements
For a local deployment, all necessary data packages are available through the customer portal (https://update.dence.de). For a full installation, the following packages need to be downloaded and installed:
- dence blockchain investigator installer package
- Initial full Bitcoin database
- Patches from the last full database version
- Currency exchange rate data files

2.4 Legal considerations
dence blockchain investigator enables the investigator to fully control, which data is used in an investigation. In a blank installation, the program is able to load synthetic blockchains and show transaction details for these. However, dence provides all data necessary to conduct fully-featured transaction tracking, clustering and service attribution in the Bitcoin main blockchain. The investigator downloads and activates the Bitcoin main blockchain. He is able to import any annotation tags or export them using CSV-files. Annotations from dence come with the Bitcoin blockchain, but can be disabled or removed depending on the investigators needs. In a nutshell, the investigator has the ability to control the data which is processed and include or exclude additional information for addresses. This allows the software to be used in many different cases with many different legal requirements in different jurisdictions.
3 Installation Guide

3.1 Installation of dence blockchain investigator

- Log into your account at https://update.dence.de
- If you have not selected your current database version yet, please enter your current database version or use an arbitrary number (e.g. “1340”) if you have not yet downloaded your database.
- From the sidebar on the right, please download the installer package in the most current version, labelled “dence blockchain investigator”.
- Execute the installer package and follow the installation steps in the wizard.
- After installation, start the program by clicking the link on the desktop.
- You will be asked for a license file. If you have not received one yet, please contact dence at info@dence.de or +49 351 850 74 817. Select the license file and press ok.
- You can now use the blockchain investigator with a limited blockchain covering the very early 150,000 Bitcoin blocks. If you want to install the full Bitcoin blockchain, please proceed to the next section 0.

3.2 Installation of the full Bitcoin blockchain

- The installer will only provide a limited blockchain for demonstration purposes. The full blockchain needs to be downloaded separately and needs to be imported into the tool. For the huge download of approx. 300 GB, the following describes how to install a download manager which can sequentially download all files of the initial database.
- Please install the Google Chrome browser
- Start your Chrome browser and install the extension “Chrono Download Manager”
  - https://chrome.google.com/webstore/detail/mciogijehkdemklbdcbfkefimifhecn
- After installing the extension, the main window of the extension is shown. Click on the configuration icon (within the extension main window). In the “Network” section, please limit “Downloads per server” to two. Allowing too many connections will not improve your download speed and you may experience connection freezes.
- Using your Chrome browser, log into your account at https://update.dence.de
- Please switch to the section “Database” (at the very top of the page)
- Click on the blue link “txt” in the descriptive text just above the link table. This will download a text file with all links which needs to be downloaded.
- Open the file and copy all links to the clipboard (Ctrl+C).
- Open the Chrono Download Manager extension and click on “+” icon to start a new download.
- Paste the whole link list into the small “URL” input field. Press Start. The links should appear in the list and the download manager will download 2 files at a time until all files have been downloaded.
- Open your download directory and check that all files (except for the last one) do have the exact same size. If that is not the case, please download the faulting files again.
- Open the dence blockchain investigator. Click on tab “Update”
- Within the “Update” view, click the directory icon and select your download location (holding the downloaded database files)
- If the “Update” button will not be enabled automatically, click the blue link “patch not shown?” to show details and inform dence at info@dence.de or call at +49 351 850 74 817.
- Click “Update”. The import process will take at least 45-60 minutes. When the import was successful, the downloaded files can be deleted. If at any time, you want to revert to a previous database version, you may need to keep the downloaded files or download them again at the time.
- Enter the update portal https://update.dence.de
- Click on “Change Database Version” and enter the version number of your database
• Check for database updates and download them to your download folder. Within “Updates” tab in the investigator, the updates will be shown automatically when the download finished. Click the “Update” button.
• You have successfully installed and updated to the latest Bitcoin database.

3.3 Installation of currency exchange rates

• Log into your account at https://update.dence.de
• If you have not selected your current database version yet, please enter your current database version.
• In the sidebar on the right, section “Currency files”, please select and download your exchange rate file (e.g. BTC to Euro).
• Start dence blockchain investigator
• Go to Preferences menu, select your currency symbol and select the downloaded currency file.
• Any bitcoin amounts from search results will be converted to your selected currency based on the selected exchange rates within the currency file.
4 Configuration
The most important configurations can be adjusted through the user interface. Start the dence blockchain investigator. Go to the Preferences menu. There you can change the following options:

▪ Database path (select the folder containing “blk*.dat” files)
▪ Configure fiat currency and provide exchange rates through the “Exchange Rate File”
▪ Configure the language of the user interface (currently German and English are supported. Please enquire any other languages you might need at info@dence.de)
▪ Configure the export and import format for the CSV module. The defaults work well with Microsoft Excel.
▪ When configuration “Consider orphaned blocks” is activated, the search will include data from orphaned/invalid blocks into the search result.

Advanced users can invoke the dence blockchain investigator with command line options. Invoke with “—help” to see the available commands. Use “—developer” to start in a mode where more, internal configuration options are accessible in the preferences pane.

5 Error Management
User-level errors will be shown as a dialog pup-up or as information within the tool (e.g. no address found in database). If anything, unexpected happens, the debug log will receive explanatory information relevant for fixing problems and bugs. A yellow icon on the bottom right of the user interface will appear and a click on it will show the debug log.

6 Backup and Update
Please conduct regular exports of your annotation data. Switch to the tab “Annotations” and click “Export”. You may also want to backup your “conf.dat” file in the database folder to keep your personal history of searches. All other data can be downloaded again from the customer portal (https://update.dence.de). Program- and database updates can be downloaded from the customer portal as well.
User Guide

Table of Content
1. User profile ................................................................................................................. 43
2. System function .............................................................................................................. 43
  2.1. Overview .................................................................................................................. 43
  2.2. Graph Tab ............................................................................................................... 43
  2.3. History Tab .............................................................................................................. 44
  2.4. Annotation Tab ....................................................................................................... 44
  2.5. Entity Tab ................................................................................................................ 45
  2.6. Block Viewer Tab .................................................................................................... 45
  2.7. Update Tab .............................................................................................................. 46
3. Further Documentation and Examples ........................................................................ 47
1 User profile

Dence blockchain investigator is a desktop application. The application uses user profile separation of the operating system. However, some data like the blockchain database is shared amongst users, because it is unfeasible to replicate >300 GB of data for each user.

2 System function

2.1 Overview

The user interface is separated in 6 tabs for different tasks (see Figure 34 no. 5). Most important functionality is always accessible through the icon bar (Figure 34 no. 1). The search bar is available in most tabs and users can issue searches for various Bitcoin data. Only in graph view, the search capabilities can be extended with the button at no. 2. The most important functionality for each tab will be briefly described in the following sections.

Figure 34: Overview of general user interface

2.2 Graph Tab

The graph tab is the most important tool of dence blockchain investigator. It features a transaction list (Figure 34 no. 3) and a transaction details list at the same area. The list can be switched with the tabs “Tx details” and “Tx table”. The overview section next to it summarizes interesting statistics about the result. It can be hidden with a click on “x” if it’s not needed. Area no. 4 is a graphical representation of transactions. Figure 4 shows the graph view with the extended search functionality expanded and a
search result for address 1GU18zL. The rectangles are transactions and show inputs on the left, and outputs on the right of the rectangle. If an in- or output does not have a line attached, they can be clicked on to show the next or previous transaction in the graph/search result. The filter can be applied to reduce the entries shown in the table to match certain criteria. By default, it matches any entered text. If a prefix of “in:” or “out:” is entered as a filter, the following text will match the input/output addresses of a transaction (which are not shown in the table, but can be filtered for).

Figure 45: Graph view with example data from Fake Cerber ransomware payments.

2.3 History Tab

The history tab just prints a list of all previous search terms (e.g. transaction hashes or addresses). The search bar automatically gives completion suggestions based on the history list. By clicking on a blue link, a new search for the term will be started.

2.4 Annotation Tab

Users can control, edit, import and export own annotations in this tab. Figure 16 shows a screenshot of some annotations. Annotations can be created in the Graph tab – selecting individual inputs/outputs of transactions. In the annotation tab, every functionality is given to maintain the users own annotation repository. Import and export uses CSV format and can be configured in the applications settings dialog
(menu “Tools” -> “Preferences” on Windows and “blockchain investigator” -> “Preferences” on MacOS).

Figure 16: Annotation Tab

2.5 Entity Tab
The entity tab allows to query information of connected addresses. When a single address or an entity number is given, the view will show all addresses, which can be connected to this address by the multiple input/cospent heuristic. If there are more than 10000 result addresses, the view might truncate the result, show an appropriate warning and give the total amount of connected addresses.

2.6 Block Viewer Tab
The block viewer shows the lowest technical structure of a block and the corresponding transactions in a browse able tree representation. It is mostly to inspect raw data for educational purposes. With this view, the raw evidence of the blockchain can be inspected. Figure 57 shows an example. Additional data and information is shown for script data in inputs and outputs.
2.7 Update Tab

The Update tab is for managing the database source and keeping it up to date. The subtab “Update Database” shows information about the currently installed database version and the last contained block date within the database. If the user wants to update the database, a link is shown to directly access the best migration path to the most recent version, considering the currently installed database version. The download folder of the browser needs to be specified. Any finished download will be picked up by the investigator and will be shown in the list of available updates. A check will make sure only patches which are applicable for the currently installed database version are shown and all patch files are undamaged. Running a single patch takes between 15-50 minutes depending on the performance of the SSD drive. If a user wants to change to a different blockchain database, or if the path to the database has changed (e.g. due to a new drive name of the SSD drive), the directory can be changed in the “Change Database” sub-tab.
Figure 18: Manage the database in the Update Tab. Note the different subtabs “Update Database” und “Change Database”.

3 Further Documentation and Examples

Customers can access a second user handbook through the customer portal at https://update.dence.de. This handbook contains a more detailed guide on how to conduct Bitcoin analysis using the dence blockchain investigator. An in-depth example analysing the ransomware “FakeCerber” is also available on request at info@dence.de. These materials are only in parts available in English yet.
# Table of Content

1. Functional description .......................................................................................................................... 50
2. Requirements ......................................................................................................................................... 51
   2.1. Hardware Requirement ..................................................................................................................... 51
   2.2. Software Requirement and conditions .............................................................................................. 51
3. Installation Guide ..................................................................................................................................... 52
   3.1. Set up a new slave instance of the database ..................................................................................... 52
   3.2. Set up a new Cointel Web application server ................................................................................... 59
4. Configuration of the services .................................................................................................................... 69
   4.1 Slave Database configuration ................................................................................................................. 69
   4.2 Web application configuration ............................................................................................................... 73
5. Error Management .................................................................................................................................... 80
   5.1 Slave server instance ............................................................................................................................. 80
   5.2 Webapplication ....................................................................................................................................... 83
6. Backup and Update ................................................................................................................................... 84
   6.1 Simulation Backup ................................................................................................................................. 84
   6.2 Backups .................................................................................................................................................. 84
   6.3 Project Update ........................................................................................................................................ 86
1. Functional description

1.1 What does the tool do?

Cointel is a crypto currency analytics tool aimed at assisting Law Enforcing Agents in their investigations. Its main functions are pattern recognition in virtual currency transactions and correlation to observed activity on the dark or clear web.

1.2 Describe the maturity of the tool.

Although the tool is mature in its current functions it still needs features with regards to logging and monitoring to be a mature tool in the sense of being capable of supporting LEA’s in a sufficient way.

The tool will be available in an intermediate stage during the field labs.
2. Requirements

2.1 Hardware Requirement

Cointel’s database architecture is based on a Master / Slave replication model that is run on servers running *Postgresql 10* databases.

There is one database called the *Master* on which updates are made and one or more *Slave* databases that automatically receive all changes that were made on this Master. The Slaves are configured as read-only so they remain *exact* copies of the Master, including all tables, users, and permissions.

Each Slave instance can be queried by one or more webservers which can also be hosted on premise. This setup ensures that the customer owned data is kept private and on premise, including all queries that are run on the database.

**Server running a slave database instance**
A (virtual) machine with:
- 4 Intel Xeon cores,
- 16 GB Ram memory,
- 2 TB SSD disk space. (NOTE 5 TB is recommended because the database will grow at a super linear pace.)

**Server running the web application instance**
A (virtual) machine with:
- 2 cores,
- 4 GB Ram memory,
- 100 GB SSD disk space.

2.2. Software Requirement and conditions

Cointel’s servers are running Ubuntu 18.04 with automatic security updates. The servers are updated periodically and with new major software releases they are upgraded to a new OS. The customer’s Slave machine could be running any other distribution as long as Postgresql10 replicating database can be setup. But we advise to use the same software as it’s a proven solution and simplifies support.

**Server running a slave database instance**

Ubuntu 18 LTS
PostgreSQL 10

```bash
~$ docker --version
Docker version 18.06.1-ce, build e68fc7a
~$ docker-compose version
docker-compose version 1.21.2, build a133471
docker-py version: 3.3.0
CPython version: 3.6.5
OpenSSL version: OpenSSL 1.0.1t 3 May 2016
```
3. Installation Guide

3.1. Set up a new slave instance of the database

Cointel's web application can be pointed to any slave database instance. We will assume you will want to setup your own and let the web application (3.2) use this one.

This section describes setting up and connecting a new Slave database to the Master and the preparations that need to be done on Master to accept the Slave. After reading this document you will be able to set up your own Slave instance to an actively synchronizing copy of Cointel's master database, assuming the necessary preparations where made and the required account information have been exchanged.

We will describe in order:

- making preparations on Master;
- preparing a Slave host;
- setting up the Slave;
- synchronizing the Slave to the Master.

The first part will be managed by the Cointel team, but it is nonetheless included to this document for completeness and reference.

What you need from the Cointel team:

- replication slot name
- replicating user name
- replicating user password

In this document we'll be using the following variables to demonstrate their respective uses:

- "DemoOrgSlotName" as the name for the replication slot that is tailored for the new Slave instance.
- "DemoRepUser" as the user name that is granted access
- "DemoRepPass" as the corresponding password

What the Cointel team needs from you:

- The IP address that belongs to the machine that will be running the new Slave instance. The Master database will only allow incoming connections from a specific IP address.

## Preparations on the Master database

**What Cointel will need from you:**

- An ip address from where the Slave will connect to the Master

*This part is taken care of by the Cointel team.*

Login to the server, then postgresql. In the psql interface check the current replication slots:
Add a new slot with the following command, then make sure the slot exists and quit.

```sql
SELECT * FROM pg_replication_slots;

SELECT * FROM pg_create_physical_replication_slot('DemoOrgSlotName');
SELECT * FROM pg_replication_slots;
\q
```

(If you want to delete a replication slot, there is also the reverse. But don’t do this now of course.)

```sql
SELECT * FROM pg_drop_replication_slot('DemoOrgSlotName');
SELECT * FROM pg_replication_slots;
\q
```

Back in the bash shell, add the IP of the new Slave to the pg_hba.conf file on the Master db:

```bash
> sudo nano /etc/postgresql/10/main/pg_hba.conf
```

Add this line, save and quit:

```
Hostssl replication rep [IPv4 of slave machine]/32 scram-sha-256
```

Now, the new configuration needs to be reloaded. If we reload the configuration from the psql terminal we don’t need to restart postgres, so:

```bash
> sudo -u postgres psql
SELECT pg_reload_conf();
\q
```

### Summary

We have:
- added a slot for replication;
- added the Slave’s ip address to the firewall;
- reloaded the configuration.

This completes the preparations. We will continue to the Slave instance setup.

**Step 1:** Get a fresh Ubuntu 18.04 LTS

- set keyboard, locale and location
- guided partitioning using whole disk and setup LVM;
- auto security updates;
- core += ssh-server;
- yes to grub (default setting).

Make it accessible over a terminal and supply yourself with a user account that has sudo permissions.

### Install PostgreSQL 10 & clean it out

```
> sudo apt-get update
> sudo apt-get install openssl postgresql
```

Postgresql will setup a default database and run it. But we want to do something non-default to it. So stop the cluster and verify that it indeed has stopped.

```
> sudo systemctl status postgresql@10-main.service
> q
> sudo systemctl stop postgresql@10-main.service
> sudo systemctl status postgresql@10-main.service
> q
> pg_lsclusters
```

Now, remove the data directory of the new (stopped) cluster, since we are going to overwrite it with the data from the Master:

```
> sudo rm -rf /var/lib/postgresql/10/main
```

Make a new directory for the data location and make postgres the sole owner. If you stick to the default location, this will be the same location as we just cleared out.

```
> cd /var/lib/postgresql/10/
> sudo mkdir main
> sudo chown postgres:postgres main
> sudo chmod 0700 main
```

Note: the service won’t start if the permissions aren’t as strict as 0700.

### Automate Postgres' connection to Master

Postgres will need to be able to maintain a connection to the Master by it self, so it needs an entry in the file `.pgpass` defining where to connect to and with what credentials.
At this point you need your equivalent of the following variables.
- "DemoRepUser" as the username that is granted access;
- "DemoRepPass" as the corresponding password.

*These must be provided by the Cointel team.*

```
> sudo -i
> touch /var/lib/postgresql/.pgpass
> echo "cci-master.coblue.eu:5432:replication:DemoRepUser:DemoRepPass"
> /var/lib/postgresql/.pgpass
> chown postgres:postgres /var/lib/postgresql/.pgpass
> chmod 0600 /var/lib/postgresql/.pgpass
> exit
```

### Mini Screen-tutorial

Now we are ready for replicating the Master’s data to our new Slave. This is a lengthy process and to make sure it doesn’t get interrupted by a timeout on your presumed SSH connection, we use the tool screen to mimic a local terminal. The following example will show you how to use it. For more information visit [this documentation](https://help.ubuntu.com/community/Screen)

```
> screen
[hit enter]
> echo test
[now leave the screen by pressing Control+A+D]
[it should say: detached from .....]
> screen -list
[it shows a list of active screens]
> screen -R [the screen id]
[you should see you "test" echo]
```

### Make a Master Base backup

The replication process only focuses on recent changes. But we're starting with a whole new and empty Slave instance. So we'll first create a base backup of the Master and start the replication only after that finishes.

*Note that you can run the base backup without having changed the configuration yet.*

Start a new screen session as described above. In it, execute the following command and the database backup from Master will start.
- Change the slot name to the slotname that you created earlier.
- The `-r 8M` option is used to limit the bandwidth to 8 MB per sec in this case.

Use bandwidth throttling if you want to do other stuff while downloading, but also if you've got a really fast (e.g. 1 Gbit) connection or if you're replicating from a Master residing on the same
(virtual) network. We have seen situations where the (spinning) disk couldn't keep up with the incoming data.

At this point you need your equivalent of the following variable.
- **"DemoOrgSlotName"** as the name for the replication slot that is tailored for the new Slave instance.

*These must be provided by the Cointel team.*

The command below will use the 'pgpass' file we just made and access master to make a full copy of the Master's main and place that in your main.

*Note that you'll need to supply the correct path if you're not using the default.*

```
[In the screen session]
> sudo -u postgres pg_basebackup -h cci-master.coblue.eu -U rep -v -P -D /var/lib/postgresql/10/main -R -X stream -w -S "DemoOrgSlotName" -r 8M
```

It will say with 'initiating base backup, waiting for checkpoint to complete' and this may take a while. Give it some time; after a few minutes it will start to run.

You can leave the screen by pressing CTRL+A+D

Monitor the progress by occasionally listing the space taken in by the data partition with `df -h`. The process will take a couple of hours depending on your connection speed. When finished, the files will be located in `/var/lib/postgresql/10/main`.

When the backup finishes PostgreSQL will attempt to start the database and automatically start synchronizing. This can only happen if the configuration is correct, so, while the backup is running, let's have a look at postgresql.conf

*Note: if the service didn't start automatically, starting it manually with 'systemctl' will tell you why not.*

Please read 4.1 for postgres and server configuration

### Summary

By now you have:
- managed the file system;
- automated access to the Master;
- made a base backup of the Master;
- tweaked the configuration so your instance becomes a replicating Slave (postgres.conf);
- and limited the access to both the machine (firewall) and the database (pg_hba.conf);
- optionally setup your own certificate and keys.

That means you're ready to start the new cluster. If everything was set up right at the first time, there's a good chance the cluster is already up and running; so test, then start.

```
> sudo systemctl status postgresql@10-main.service
> sudo systemctl start postgresql@10-main.service
```
This should firstly start the recovery process, because the Master has progressed after the base backup was completed. When the recovery completes it will automatically switch to the replication mode.

From now on, you can access the new Slave instance from the webfrontend. Setting your own instance of the webfrontend is covered in another document.

This concludes the Setup and this document. If you have any additional questions or improvements you can always contact the Cointel team on support@cointel.eu

### Trouble shooting

#### Postgresql service won't start

Normally running the start and status command provide a very decent insight in the problem if for some reason the service won’t start.

```
> sudo systemctl start postgresql@10-main.service
> sudo systemctl status postgresql@10-main.service
```

This is almost always related to either:
- not having all paths set correctly in postgresql if you have non-default paths;
- those paths not having the correct permissions;
- those paths not having enough remaining diskpace.

In all cases, start by evaluating the error message and checking the log for hints.

```
> tail /var/log/postgresql/postgresql-10-main.log
```

In many cases the message will indicate what’s wrong. The fastest way to a solution is a quick search for the problem online. Our setup is reasonably close to a default replicating configuration so in many cases someone posted a cause and solution for your problem already.

If you do get stuck you can always contact the Cointel team at support@cointel.eu. Please include the information provided by systemctl and log if you do.

#### Repairing a Slave that cannot automatically recover

Most out-of-sync problems can be automatically recovered by postgresql because Master and Slave are in contact with each other about their WAL buffers and the Master will stop when the Slave is at risk of missing WAL segments.

In those cases we need to check the Master and Slave’s buffers, restart the Master and auto-recovery will kick in.

If for some reason your Slave will get in a situation where there is a gap in WAL segments between the Master’s oldest and the Slave’s youngest, auto recovery cannot fix the problem.
In that case is recommended that you run a *pg_basebackup*, just like in the documented Slave instance set-up. This will copy all data from the Master before starting the replication process. So no matter the size of the gap in WAL segments; it is always possible to recover from a out-of-sync Slave instance. This just more time-consuming than the auto recovery.

## Known issues

### Slave database is not being updated anymore.

There is the possibility of the WAL partition to be written faster than the database can process the information; it is unlikely to happen, but possible. It seems having a separate (relatively small) partition for the WAL is a drawback. But actually this will protect the data partition from running out of disk space if this process goes unnoticed.

Still this can lead to a problem where *the Slave database isn't updated anymore*

The WAL disk space is used to buffer changes to the Master which haven’t been applied to the Slave. In normal scenarios, these changes will comprise of several MB’s, in which case the Slave will be running in sync with Master in a small amount of time.

If the Master is undergoing a big modification, or if there is simply a very big influx (which can happen if the Master has gone for a long period without updates or clustering), there is a chance that the Slave cannot keep up. In this case its WAL buffer will fill up completely.

Master and Slave are in contact with each other and Master knows when the Slave is in trouble. It will keep more WAL segments so Slave can pull them before they are removed. Eventually, this will also lead to the Master’s buffers running out. When that happens, the master will shut itself down; the slave will continue emptying its buffers, but there will be no new segments coming in from Master. So for the Slave instance everything will look normal, but when you look at the latest block / transaction in the UI, it will seem as if the Slave is not being updated anymore - which is true, but that’s because Master went down.

Whenever the Master goes offline, for example because of the reason mentioned above, the Cointel team will be alerted and they will discover that Master's WAL buffer is full.

*At this point the Cointel team may contact you and request to know the oldest and newest WAL segment on the Slave*

To restart the Master database, PostgreSQL will require some remaining buffer space, so the Cointel team may have to manually scrub some old WAL Segments. Before they do, they’ll make sure that no Master segments are deleted that aren’t already copied to the Slaves by requesting the oldest segments in the Slave instances. In case the Cointel team does not have access to that instance you’ll need to perform this command and report on the latest segment in the WAL dir.

```
> sudo ls -lahr /var/lib/postgresql_wal/10
```

```
> ...
> -rw------- 1 postgres postgres 16M aug 19 13:58
00000001000016D00000000A4
> -rw------- 1 postgres postgres 16M aug 19 13:20
00000001000016D00000000A3
```
In this case the Cointel team would like to know that the Slave’s oldest segment is *00000001000016D000000A3* because that means that all older segments (if any) can safely be deleted from Master.

It is also worthwhile to know the *newest segment*

```bash
> sudo ls -lah /var/lib/postgresql_wal/10

> ... 
> -rw------- 1 postgres postgres 16M aug 20 04:24 00000001000016D2000000CC 
> -rw------- 1 postgres postgres 16M aug 20 04:41 00000001000016D2000000CD 
```

Here *00000001000016D2000000CD* is the newest, which tells us that no segment newer than this one may be deleted or there will be a gap between Master’s and Slave’s log.

With this knowledge the Cointel team can safely remove older segments and thus free up buffer space and thus restart the Master again.

If the Slave’s are still running, they will automatically get in sync with the Master again.

#### Summarizing

If your UI shows blocks and transactions from several hours ago as being the latest, this indicates that your instance of Slave is not being updated anymore. Do the following:

1. Check that the service is running with `sudo systemctl status postgresql@10-main.service` and optionally restart it.

2. Check that there is enough remaining diskspace with `df -h`

If all seems right, your Slave instance probably *is not at fault* and this problem is the result of the Master having stopped. You can expect the Cointel team to resolve the problem; as long as your service keeps running, it will catch up eventually.

The Cointel team can request you for the name of your Slave’s oldest and newest WAL segment, which can be found with `sudo ls -l /var/lib/postgresql_wal/10`.

3.2. Set up a new Cointel Web application server

### Summary

The Cointel webserver has some software requirements which are Docker and Docker compose and some recommended tools like a firewall and Letsencrypt certificates. If you have those
already, the setup can be really quick and easy. This summary will cover the minimum of required steps.

1. Download or acquire the package, referenced by 'Cointel_webserver.tar.gz'
2. Extract the package, cd into it and open docker-compose.yml
3. Make sure that the service "python_uwsgi"'s environment variables are correct (they will have been set to sane defaults)
4. Build the network and the containers.
5. Run service csdb alone and let it finish the initialization and upgrades
6. Run the composition
7. visit <the site>/sysadmin, supply the admin's email address and follow the instructions from the email.

```bash
> tar -xzf Cointel_webserver.tar.gz
> cd Cointel_webserver
> cat docker-compose.yml
[ optionally edit ]
> docker network create mytunnel
> docker-compose build
> docker-compose up csdb
[ wait till it says "Done upgrading", can take a minute]
[ then CTRL + C]
> docker-compose up -d
```

That's it, you're up and running.

Please read the remainder of this document for:
- understanding the services in the infrastructure;
- how to configure them is you'd rather have it a bit differently;
- how to work with ssl certificates;
- how to update your server;
- how to solve problems should they occur.

### Preparations

We shall first go over the software packages that are required by the Cointel webserver.
It is assumed you have a working (virtual) machine running Ubuntu 18.04 with shell access (e.g. over SSH) and a non-root account that has sudo privileges.

The following three sub-chapters will install docker and docker-compose. If your system is capable of running docker-compose, you can skip ahead to the 'Letsencrypt & Certbot' sub-chapter.

### Docker

First, we have to install docker. Again, this assumes you'll be using Ubuntu 18.04. If you are using something else, please consult [digital ocean's installation manual](https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04) and see if your preferred OS has its own version available.

*If you have docker installed already, skip this step.*

```bash
> sudo apt-get update
> sudo apt install apt-transport-https ca-certificates curl software-properties-common
> curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
> sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable"
> sudo apt-get update
> apt-cache policy docker-ce
> sudo apt install docker-ce
> sudo systemctl status docker
> <q>
```

By now you should see an active, running service.

### Docker group

Now we want docker to be usable by the user without having to use sudo all the time. This is achieved by making your user account (or another) member to the docker group. *Skip this step is you do prefer the user to enter a password before docker commands can be issued.*

```bash
> sudo usermod -aG docker ${USER}
> su - ${USER}
> id -nG
> <CTRL+D>
```
The output shows your user and the group memberships; it should include docker. If not, disconnect and log in again; the permissions will be reloaded.

### Docker compose

Now we're gonna install docker-compose. *You can skip this step if you have already installed docker-compose*

We like to use an up-to-date version of docker compose. Which we will get from the docker release page instead of Ubuntu's repository. This tutorial references 1.21, but [here](https://github.com/docker/compose/releases) you can make sure whether you should use another version number. If there is a newer one, feel free to overwrite the version when copying the commands below.

```bash
> sudo chmod +x /usr/local/bin/docker-compose
> docker-compose --version
```

That last one shows the version; verify that the version is at least 1.21.

### Letsencrypt & Certbot

We assume you'll want to use ssl certificates to encrypt your client's connections to the web server with. We will also assume these are or will be installed on the host machine. One of the services of the docker compose can be configured to use them when connecting to the web server.

If you do not yet have certificates or if this web server is setup to run internally we recommend you use letsencrypt certificates and manage them with Certbot.

[This tutorial](https://www.digitalocean.com/community/tutorials/how-to-secure-nginx-with-let-s-encrypt-on-ubuntu-18-04) will tell you how to set it up on Ubuntu 18.04 (or another distribution). To summarize:

Install certbot

```bash
> sudo add-apt-repository ppa:certbot/certbot
> sudo apt install python-certbot-nginx
```
Setup a very basic configuration, so certbot will have something to work with. In the end we'll use the nginx in the webserver, but right now we need this 'place holder' so certbot can issue new certificates.

Please replace *example.com* with your own host name, where the vps can be reached.

```
> sudo touch /etc/nginx/sites-available/example.com
> sudo nano /etc/nginx/sites-available/example.com
```

[ add the following ]

```server {
    listen 80;
    listen [::]:80;

    root /var/www/example.com/html;
    index index.html index.htm index.nginx-debian.html;

    server_name example.com www.example.com;

    location / {
        try_files $uri $uri/ =404;
    }
}
```

[ save and exit ]

```
> sudo nginx -t
> sudo systemctl restart nginx
```

If nginx is reloaded we can run Certbot (with your own hostname) as follows:

```
> sudo certbot --nginx -d example.com -d www.example.com
```

Run through the menu, you don't need to choose for 'redirect'. We only need the certificates at this step. Please do enter a valid email address: it will be used to warn you if the certificate renewal will fail.

When the process finishes, you'll have a nice set of new certificates in `/etc/letsencrypt/live`. Remember the path, we will use it in the docker configuration down the road.

Reload nginx and check that the certificates are working by visiting your http://localhost and https://localhost in a browser.
Now that we have the certificates, we can stop the host's nginx service. Actually we need to, because else the web server's nginx cannot connect to port 80 and 443.

```
> sudo systemctl stop nginx
```

You should also make sure the service does not run when the system reboots.

#### Firewall

Nginx will need to be able to accept incoming connections, so if you're using a firewall, please tell it to allow incoming connections to nginx. In the example below we show how this is done in the default Ubuntu 18 firewall *Ufw*.

```
> sudo ufw status
```

Output

```
Status: active
```

```
To Action From
--- ----- ----- ----
OpenSSH ALLOW Anywhere
Nginx HTTP ALLOW Anywhere
OpenSSH (v6) ALLOW Anywhere (v6)
Nginx HTTP (v6) ALLOW Anywhere (v6)
```

What you need is the Nginx lines to contain 'Full' in stead of HTTP, so do:

```
> sudo ufw allow 'Nginx Full'
> sudo ufw delete allow 'Nginx HTTP'
> sudo ufw status
```

#### Installation

You have obtained a latest version of the software, named `Cointel_webserver_x.x.x`, move it to a preferred location, (for example /srv/cointel) and extract.

```
> tar -xzf Cointel_webserver_x.x.x.tar.gz
> cd Cointel_webserver_x.x.x
```
Open the `docker-compose.yml` file; it is designed to hold the information for a customer-specific installation. Perhaps the Cointel team has provided a yml file with tailored variables for you already, in that case you can overwrite it with that one.

Now open the `*.yml` file and edit these settings (leave all others at their current value)

```yaml
```

csdb:
  volumes:
    - ./csdb_postgresql:/var/lib/postgresql/data  # This mounts the customer specific database's data location to this folder 'csdb_postgresql' and makes the data persistent on your disk. If the folder does not exist, it is created. Without this line the data will exist *only* inside the container and will be lost when it is deleted (useful for testing).

  # Choose a user, strong password and database name. These will be set on the first run of csdb, so you can come up with your own credentials. This whole service can be commented out if you want to run your own customer specific database on some other location.
  environment:
    - POSTGRES_USER=web
    - POSTGRES_PASSWORD=MyLocalVerySecurePassword
    - POSTGRES_DB=cointel

backup:
  # This service will run a daily backup of your CSDB database: specify where it should keep them, how many backups you want to keep and copy the csbd database credentials.
  volumes:
    - ./csdb_backups:/var/backups/postgresql  # Specify a location to keep the csdb backups
  environment:
    - CSDB_HOST=csdb  # the ip, host or service name where the customer specific database is hosted
    - CSDB_USER=web  # the user of the csdb. If the (above) service is used, set to the value of POSTGRES_USER
    - CSDB_PASSWORD=MyLocalVerySecurePassword  # the password. If the (above) service is used, set to the value of POSTGRES_PASSWORD
    - CSDB_PORT=5432  # the port number of the csdb. If the (above) service is used, this must correspond to the mapped csdb:ports (first number)
    - DAY_OF_WEEK_TO_KEEP=5 # Which day to take the weekly backup from (1-7 = Monday-Sunday)
- DAYS_TO_KEEP=14       # Number of days to keep daily backups
- WEEKS_TO_KEEP=52      # How many weeks to keep weekly backups

depends_on:
  - "csdb"

python_uwsgi:
  environment:
    - SLAVE_DB_HOST=       # the ip or hostname of the replicating slave
    - SLAVE_DB_USER=       # the user that was created for querying the database, e.g. web
    - SLAVE_DB_PASSWORD=   # the corresponding password
    - SLAVE_DB_PORT=5432   # use the port of the replicating slave, should be 5432
    - SLAVE_DB_NAME=cci    # should be cci
    - CSDB_HOST=csdb       # the ip, host or service name where the customer specific database is hosted
    - CSDB_USER=web        # the user of the csdb. If the (above) service is used, set to the value of POSTGRES_USER
    - CSDB_PASSWORD=MyLocalVerySecurePassword  # the password. If the (above) service is used, set to the value of POSTGRES_PASSWORD
    - CSDB_PORT=5432       # the port number of the csdb. If the (above) service is used, this must correspond to the mapped csdb:ports (first number)
    - CSDB_NAME=cointel    # the name of the csdb-database. If the (above) service is used, set to the value of POSTGRES_DB
    - EMAIL_SENDER=        # the sender of the mail, set is to the same as EMAIL_USERNAME
    - EMAIL_SMTP=          # the smtp server to be used
    - EMAIL_USERNAME=      # the email login
    - EMAIL_PASSWORD=      # the email account password
    - INTERNAL=true        # set this to 'true' if you do not want the landing page and 'subscribe to our mailing list' feature enabled.

nginx_cointel_http:
  ports:
    - "80:80"

nginx_cointel_https:
  ports:
    - "443:443"
    - "80:80"                 # the server will be running internally on port 80 (and 443). Change the first port number if you want it to be available on something else e.g. "10080:80" (or 10443:443)
Volumes (below) assume that the host machine contains lets encrypt certificates that are managed by certbot and kept in /etc/letsencrypt. If they are stored somewhere else, change the first path and map it to /etc/letsencrypt:

- "/etc/letsencrypt:/etc/letsencrypt"

Environment:

- "SERVER_SSL=example.eu" # hostname under which the certificates are stored.
- "SERVER_DOMAINS=example.eu www.example.eu" # the domains nginx should accept.

Directly under the nginx_cointel_https service you'll find a commented 'nginx_cointel_http' (without the 's') service. If you did not yet got around to make new certificates or if this is not possible e.g. because you don't have a working host name (yet), you can pick the http variant over the https variant so you can continue with the installation process. It will be easy to switch to the https once you do obtain the certificates.

If you want to run with http comment uncomment the http and comment the https service; there should be only one or docker will have difficulties connecting the ports.

For now we assume the certificates are up to date and stored in the default letsencrypt folder: /etc/letsencrypt. If you're not using letsencrypt and have them stored elsewhere please consult the configuration chapter.

### Build the containers

Now setup the network used by docker compose:

> docker network create mytunnel

And follow through with:

> docker-compose build --no-cache

To initiate the customer specific database, we do an initial run of the csdb service. The container will detect the absence of a data location and run the installation process. Because this one-time procedure can conflict with the webserver trying to connect, we run it separately, let it finish and then stop. On the next run the data location will be detected and the service can accept the incoming connection from the webserver.

> docker-compose up csdb

Let this run for a minute or so. Postgres is initializing and upgrading with intermediate pauses; those are intentional. Let it run until it says something similar to:

```
INFO [alembic.runtime.migration] Running upgrade e6ce7d6d229a, Initial flask security commit
INFO [alembic.runtime.migration] Running upgrade e6ce7d6d229a -> cd8c01faee5e, Organization and case management
```
done upgrading

then stop the container with CTRL+C.

Now we can run the whole composition with:

```
> docker-compose up -d
```

With all the right credentials in place, you should be able to visit the IP address of the `webfrontend` machine and see the landing page or a login form.

## Initiating the first user

A special API endpoint is available as long as there are no entries in the customer specific database. Visit the application on the `/sysadmin` endpoint. You will see a form asking for an email address.

Fill in an address you would like to use as the system administrator account. A unique link to set the password and instructions will be sent. When the password is set and the account is activated, the sysadmin can login and add manager accounts.

Likewise, an email with instructions will be sent to those account and when they are activated, managers can add new user accounts in a similar fashion.

*NOTE* The purpose of the admin account is to make one or more Manager accounts for corresponding management units and each manager can subsequently make her own user accounts for the people working in her team. The admin has no further roles; he doesn't get to see much of the content. It is advised that you also make a management account for your own, so you can explore the tool by yourself.

*NOTE* As the sysadmin (or manager) user you can make new user accounts through the data api. Please consult with the documentation service by opening the browser and visit localhost/apidocs

**The application is now fully functional.**
4. Configuration of the services

4.1 Slave Database configuration

### PostgreSQL Configuration

While the base backup is running, we can have a look at the configuration.

If you are going to use other data directories that the default ones, make sure set it up in this step. You can change the data directory easily in this configuration file, but not the location where the write ahead log operates. To do so you could create a symlink from the default location to where you actually want the WAL to be written to.

The WAL will grow to about 10GB so it is not crucial that this is put on a big disk.

In our own configuration we choose to apply the default directories, but made sure these where mounted on other partitions.

```bash
> sudo nano /etc/postgresql/10/main/postgresql.conf
```

As root open the configuration file and replace the respective settings with the following:

```bash
# FILE LOCATIONS
data_directory = '/var/lib/postgresql/10/main'

# CONNECTIONS
listen_addresses = '*'
port = 5432
max_connections = 100
unix_socket_directories = '/var/run/postgresql'

# AUTHENTICATION
ssl = on

# RESOURCE USAGE (except WAL)
# This configuration is applied to a machine that utilizes 16GB RAM.
# Change the values accordingly if you have more.
shared_buffers = 4GB
work_mem = 256MB
maintenance_work_mem = 512MB
dynamic_shared_memory_type = posix

# WRITE AHEAD LOG
# 'hot_standby' is replaced by 'replica' in PostgreSQL 10
wal_level = replica
max_wal_size = 40GB
min_wal_size = 400MB
checkpoint_completion_target = 0.9
archive_mode = on
archive_command = 'cd .'

# REPLICAATION
# These settings are normally intended for the Master server (the Slave will not be replicating to other machines).
# However, we want to be able to use the Slave server as replacement for the Master server in case of emergency. That's why we also enable this WAL.
```
max_wal_senders = 10
wal_keep_segments = 320 # About 5GB of log
max_replication_slots = 10

# This means that it's a Slave server
hot_standby = on

# QUERY TUNING

# Because of SSD usage, random_page_cost may be low
# set effective cache to 75% of available RAM
random_page_cost = 1.5
effective_cache_size = 12GB

# ERROR REPORTING AND LOGGING
log_min_duration_statement = 0
log_checkpoints = on
log_connections = on
log_disconnections = on
log_duration = on
log_line_prefix = '%t [%p]: [%l-1] user=%u,db=%d,app=%a,client=%h '
log_lock_waits = on
log_temp_files = 0

# AUTOVACUUM PARAMETERS
log_autovacuum_min_duration = 0

# CLIENT CONNECTION DEFAULTS
datestyle = 'iso, mdy'
lc_messages = 'C'
lc_monetary = 'C'
lc_numeric = 'C'
lc_time = 'C'
default_text_search_config = 'pg_catalog.english'

**Note:** for a more complete understanding of the WAL configuration you can always refer to PostgreSQL's own very extensive documentation. There is a lot about [server configuration](https://www.postgresql.org/docs/10/static/runtime-config.html), including [general](https://www.postgresql.org/docs/10/static/runtime-config-wal.html) and [detailed](https://www.postgresql.org/docs/10/static/wal-configuration.html) Write Ahead Log settings and [Replication](https://www.postgresql.org/docs/10/static/runtime-config-replication.html) including some [best practices](https://wiki.postgresql.org/wiki/Installation_and_Administration_Best_practices#WAL_Directory) regarding the WAL location.

### Limiting access

In general you would like for Slave to be able to make contact with Master, you will need SSH access and the frontend should be able to query PostgreSQL. In this example we're using the default UFW firewall.

By default, ufw is disabled:

> sudo ufw status
should show `inactive`.

Allow the necessary applications with:

```bash
> sudo ufw default allow outgoing
> sudo ufw default deny incoming
> sudo ufw allow ssh
> sudo ufw allow 5432
```

and enable the firewall with:

```bash
> sudo ufw enable
```

Verify that only ports `5432` and `22` are allowed with:

```bash
> sudo ufw status
```

and it should display:

```
Status: active
To                      Action      From
--                      ------      -----
22/tcp                   ALLOW       Anywhere
5432                     ALLOW       Anywhere
22/tcp (v6)              ALLOW       Anywhere (v6)
5432 (v6)                ALLOW       Anywhere (v6)
```

A second way of limiting database access is by editing postgres' access configuration. As default that's located here:
- `/etc/postgresql/10/main/pg_hba.conf`

Modify the `pg_hba.conf` and manage access to and from the Slave database. The webfrontend (which is probably running on some other machine) will be making queries to your Slave instance as user `web`, so in general that is the only user that needs access from another machine. So, for example, your pg_hba.conf could look like this:

```plaintext
# TYPE  DATABASE    USER      ADDRESS              METHOD
local  all         postgres  -                     peer
local  all         all       -                     peer
host   all         all       127.0.0.1/32         scram-
    sha-256
host   all         all       ::1/128              scram-
    sha-256
```
Please note that the replication process also includes postgres' users, so there is no use for making additional users in this table; they cannot be added to postgres because it is entirely read-only. If you want to access the database this could be done locally or over a ssh-tunnel from another machine of course, but that is outside the scope of this tutorial.

### Generate your own keys

The connection from the web frontend to the Slave instance is encrypted using the Slave's certificate.

The base backup that was performed in the previous steps essentially copies the whole /var/lib/postgresql folder form Master. Therefore, the server key and certificate are also included. You'll want to make your own key and certificate:

```bash
> sudo -i
> cd /var/lib/postgresql/10/main/
> ls -l
> rm server.key
> rm server.crt

Generate a new key and certificate (still as root)

```bash
> openssl req -new -x509 -days 365 -nodes -text -out server.crt -keyout server.key -subj "/CN=your.own.host.eu"
> ls -l
> chmod 600 server.key
> chmod 600 server.crt
> chown postgres:postgres server.key
> chown postgres:postgres server.crt
> exit
```

The default postgresql configuration will point at the 'snake-oil' key and certificate, which is installed per default so communication can still be encrypted even if no one would bother to install their own certificate (here having something is better than having nothing).

You will need to tell postgres where your keys are:

```bash
> sudo nano /etc/postgresql/10/main/postgresql.conf
[and add/overwrite]
# CONNECTIONS AND AUTHENTICATION
...
ssl_cert_file = '/var/lib/postgresql/10/main/server.crt'
ssl_key_file = '/var/lib/postgresql/10/main/server.key'
...
Or which ever is suitable for your files' locations.

## 4.2 Web application configuration

### Cointel’s infrastructure: the plumbing.

The frontend consist of four different services, each with its own functionality, orchestrated by docker compose. These are:

- The data api documentation: *documentation*
- Customer specific database: *csdb*
- A backup service for the customer specific database: *backup*
- The web server: *python_uwsgi*
- A caching service: *redis*
- A router: *nginx_cointel_http* or *nginx_cointel_https*

You will find them in the file `docker-compose.yml`

This chapter will go into more detail about those services and what about them can or needs to be configured.

#### ### Service: python_uwsgi

This service runs the actual web server, which consists of two layers: If you request a specific page in the application, the server first responds with an empty page generated by the template engine. Then the required contents are retrieved by the template calling the data api.

So the webserver has a template engine and a data api layer. The advantage of this setup is to be able to handle automated requests; you don't really need the frontend to make a request for a specific piece of data.

So, the webserver is represented by the service *python_uwsgi*. The installation procedure already went over the environment variables briefly. We shall provide additional information on the ones you can change.

**SLAVE_DB_HOST and SLAVE_DB_PORT**: this must name the host (or ip address) where your instance of the Slave database is running; it is assumed you that you have your own on premise.
If not you can use the one used by Cointel, but you’ll need to contact the Cointel team for access credentials.

**SLAVE_DB_USER, SLAVE_DB_PASSWORD, and SLAVE_DB_NAME** The credentials that go along with access to the slave database. As you know the Slave instance replicates from the Master instance which includes the user who can access this information. Either the Cointel team has pre-loaded this already in the docker-compose.yml, or you will have been given the credentials to connect with. *Do not change these settings*

**CSDB_HOST and CSDB_PORT**: must point at the host - or service - and port running the customer specific database. Per default this is set to the *csdb* service in this composition. So per default, docker will run a service on the same host for this database. Change it if you want to run your customer specific database at some other host. You must contact the Cointel team about how to install the required schema on that location, this is outside the scope of this document.

**CSDB_USER and CSDB_PASSWORD** These must comply to the user credentials by which you can query the customer specific database. *If you are using the csdb service these must be the same as in csdb:environment: POSTGRES_USER and POSTGRES_PASSWORD*

**EMAIL_xxxx** settings. Please configure your own smtp server so the web server can send out emails.

**INTERNAL** A setting that is used to tell the webserver that it is run as some integral part of a bigger system, in stead of a stand alone. If set to 'True' the server won't show the landing page (but instead redirect to the login page), Cointel logo or mention the Cointel name.

### Service: Redis

This service manages a small cache mapping requested objects to database results. Many components of Cointels database will never change (because of blockchain) and are therefor very convinient to store in a cache. So when a user requests the same data within a reasonable amount of time, the page will load much faster and the database isn’t burdened again.

*You don’t need to configure anything for this service.*
### Service: Documentation

The documentation container runs an interactive Swagger user interface that documents the previously mentioned data api layer. The page will show you the details of all available api endpoints and supply a way of interacting with them. This can teach you how to run an automated query.

Please explore it once everything is running.

The port 8080 is used by default and the routing service called 'nginx_cointel_http' will redirect <server>/apidocs to that service and port. By exposing this port, the documentation will also be available directly at <server>:8080. You could turn that off if that is undesirable by commenting the line and rebuilding the container.

### Service: Customer specific database

If you have setup the Slave instance you will have learned that it is an exact copy of the Master, a read-only copy. In order to be able to support multiple on premise solutions, each solution will need a unique database to store their own working environment in, including the application users.

The csdb service takes care of this. It runs a PostgreSQL10 container based on Alpine in which your organization's user accounts and working data and are stored.

If it is run for the first time it will add a user and password that are passed in from the configuration. This postgres user will also be used by the webserver when performing queries like making an new users for your organization, logging in, changing a password, etc.

#### Volumes

**-./csdb_postgresql:/var/lib/pgsql/data**

This one is really important. If you have this service specified in the webservers csdb host environment variable, this service will be responsible for storing all the private (user accounts) data.
This container is based on a PostgreSQL10 image which will store its data in the folder `'/var/lib/postgresql/data'`. So that is in the container... If you would remove the container, you would remove the database. In order for the data to be persistent, we need to map a path on the host to that data location, which is the purpose of this line.

The default location will create a folder 'conainer_csdb' in the location from where you run docker compose. If you would like a better location (e.g. /var/db/cointel/userdata or which ever you prefer), specify that path on the left side and leave the right side intact, e.g. `"/var/db/cointel/userdata:/var/lib/postgresql/data"`

You will need to create that path and apply proper permissions of course.

#### Environment

**POSTGRES_USER and POSTGRES_PASSWORD**

Set this to whatever you like, but make it strong! These credentials are installed in the customer database on the first run and are subsequently used to setup a connection for all the private database related queries. So don't change them or your users will not be able to log in to the web application anymore.

Also if the service *python_uwsgi* will use this service as its 'CSBD', then the 'SLAVE_DB_USER' and 'SLAVE_DB_PASSWORD' need to be the same as these settings, respectively.

### Service: Backup

This service will run a daily backup script which will apply PostgreSQL's tool `pg_dumpall` and output the result to a folder that was mounted to the host system. You need to specify the database credentials that can be used to run the backup; use the same as specified in the CSDB service.

For the backup to be useful you'll need to specify some local folder that can be used to mount in the container in order for the backup to be stored outside of the container. How that local folder is subsequently offloaded to an off-site backup is out of scope of this documentation.
You can also specify how many daily and weekly backups should be stored.

#### Volumes

**-/csdb_backups:/var/backups/postgresql**

The container will make periodic backups and store them in the right-hand location of this mapping. With this line you can specify *if* and *where* you would like the container to put those backups on the host machine.

Per default this will be written to <path-to-web-server>/csdb_backups, which will be made automatically. You could specify a more generic location like */var/backup/postgresql/csdb/* by changing the left-hand side of the mapping.

#### Environment

**CSDB_HOST and CSDB_PORT**: must point at the host - or service - and port running the customer specific database. Per default this is set to the *csdb* service in this composition. Change it if you have your customer specific database at some other host.

**CSDB_USER and CSDB_PASSWORD**: These must comply to the user credentials by which you can query the customer specific database. *If you are using the csdb service these must be the same as in csdb:environment: POSTGRES_USER and POSTGRES_PASSWORD*

**DAY_OF_WEEK_TO_KEEP**: Specify which day the backups script should keep as the weekly backup. Per default this is set towards the end of the week, on Saturday.

**DAYS_TO_KEEP**: Specifies the number of daily backups to keep before rotating. The default is 7, so we keep 1 week of daily backups.

**WEEKS_TO_KEEP**: Specifies how many weekly backups to keep before rotating. The default is 4, so we can go back up to a month. Feel free to increase this number.
### Service: nginx_cointel_http and nginx_cointel_https

The service that does the routing. It will be listening on ports 80 and if you configure to use https 443 and route the requests to either the web server container or the documentation container. If you want your service to be available at another port i.e. you're running another service on those default ports, map those to ports 80 and 443 respectively in the configuration.

The router will handle all static files by itself and it will redirect the rest towards the *python_uwsgi* service, which will then make an appropriate response. All request to '/apidocs' are redirected to the service *documentation* and will show the API documents page.

If you're using the https variant you will have to tell it where to find the ssl certificates for your web server. In the set-up we assumed the use of Letsencrypt and Certbot, which make issuing and renewing certificated really easy, but which also require a working host name.

#### Use your own certificates

If something went wrong, or if you don't want Certbot to manage your certificates, you can also use your own, which may be stored on some other location on the host machine. The nginx service assumes these settings and paths:

```sh
ssl_certificate /etc/letsencrypt/live/SRVSSL/fullchain.pem; # managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/SRVSSL/privkey.pem; # managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
```

Here the 'SRVSSL' placeholder is replaced by the contents of the environment variable 'SERVER_SSL'. So either make a copy of your certificates to follow similar paths, potentially from another location that is mapped by 'volumes' or change these ssl_certificate and ssl_certificate_key paths in the enclosed nginx configuration.

The paths can be changed in the file `nginx_cointel/nginx.conf-https` if you need something else entirely. The script `nginx_cointel/startupscript.sh` will replace the placeholders with the service's environment variables (the ones we're currently discussing).
This may need a bit of additional tinkering. You are free to alter these files to fit them to your needs:

- docker-compose.yml
- nginx_cointel/Dockerfile.nginx_cointel_https
- nginx_cointel/nginx.conf-https
- nginx_cointel/startupscript.sh
5. Error Management

5.1. Slave server instance

### Postgres

#### service won’t start

Normally running the start and status command provide a very decent insight in the problem if for some reason the service won’t start.

```bash
> sudo systemctl start postgresql@10-main.service
> sudo systemctl status postgresql@10-main.service
```

This is almost always related to either:
- not having all paths set correctly in postgresql if you have non-default paths;
- those paths not having the correct permissions;
- those paths not having enough remaining disk space.

In all cases, start by evaluating the error message and checking the log for hints.

```bash
> tail /var/log/postgresql/postgresql-10-main.log
```

In many cases the message will indicate what's wrong. The fastest way to a solution is a quick search for the problem online. Our setup is reasonably close to a default replicating configuration so in many cases someone posted a cause and solution for your problem already.

If you do get stuck you can always contact the Cointel team at support@cointel.eu. Please include the information provided by systemctl and log if you do.

### Repairing a Slave that cannot automatically recover

Most out-of-sync problems can be automatically recovered by postgresql because Master and Slave are in contact with each other about their WAL buffers and the Master will stop when the Slave is at risk of missing WAL segments.

In those cases we need to check the Master and Slave's buffers, restart the Master and auto-recovery will kick in.

If for some reason your Slave will get in a situation where there is a gap in WAL segments between the Master’s oldest and the Slave’s youngest, auto recovery cannot fix the problem.
In that case is recommended that you run a *pg_basebackup*, just like in the documented Slave instance set-up. This will copy all data from the Master before starting the replication process. So no matter the size of the gap in WAL segments; it is always possible to recover from a out-of-sync Slave instance. This just more time-consuming than the auto recovery.

### Known issues

#### Slave database is not being updated anymore.

There is the possibility of the WAL partition to be written faster than the database can process the information; it is unlikely to happen, but possible. It seems having a separate (relatively small) partition for the WAL is a drawback. But actually this will protect the data partition from running out of disk space if this process goes unnoticed.

Still this can lead to a problem where *the Slave database isn't updated anymore*.

The WAL disk space is used to buffer changes to the Master which haven't been applied to the Slave. In normal scenarios, these changes will comprise of several MB's, in which case the Slave will be running in sync with Master in a small amount of time.

If the Master is undergoing a big modification, or if there is simply a very big influx (which can happen if the Master has gone for a long period without updates or clustering), there is a chance that the Slave cannot keep up. In this case its WAL buffer will fill up completely.

Master and Slave are in contact with each other and Master knows when the Slave is in trouble. It will keep more WAL segments so Slave can pull them before they are removed. Eventually, this will also lead to the Master's buffers running out. When that happens, the master will shut itself down; the slave will continue emptying its buffers, but there will be no new segments coming in from Master. So for the Slave instance everything will look normal, but when you look at the latest block / transaction in the UI, it will seem as if the Slave is not being updated anymore - which is true, but that's because Master went down.

Whenever the Master goes offline, for example because of the reason mentioned above, the Cointel team will be alerted and they will discover that Master's WAL buffer is full.

*At this point the Cointel team may contact you and request to know the oldest and newest WAL segment on the Slave*

To restart the Master database, PostgreSQL will require some remaining buffer space, so the Cointel team may have to manually scrub some old WAL Segments. Before they do, they'll make sure that no Master segments are deleted that aren't already copied to the Slaves by requesting.
the oldest segments in the Slave instances. In case the Cointel team does not have access to that instance you’ll need to perform this command and report on the latest segment in the WAL dir.

> sudo ls -lahr /var/lib/postgresql_wal/10

> ...
> -rw------ 1 postgres postgres 16M aug 19 13:58 00000001000016D0000000A4
> -rw------ 1 postgres postgres 16M aug 19 13:20 00000001000016D0000000A3

In this case the Cointel team would like to know that the Slave’s oldest segment is *00000001000016D0000000A3* because that means that all older segments (if any) can safely be deleted from Master.

It is also worthwhile to know the *newest segment*

> sudo ls -lah /var/lib/postgresql_wal/10

> ...
> -rw------ 1 postgres postgres 16M aug 20 04:24 00000001000016D2000000CC
> -rw------ 1 postgres postgres 16M aug 20 04:41 00000001000016D2000000CD

Here *00000001000016D2000000CD* is the newest, which tells us that no segment newer than this one may be deleted or there will be a gap between Master’s and Slave’s log.

With this knowledge the Cointel team can safely remove older segments and thus free up buffer space and thus restart the Master again.

If the Slave’s are still running, they will automatically get in sync with the Master again.

#### Summarizing

If your UI shows blocks and transactions from several hours ago as being the latest, this indicates that your instance of Slave is not being updated anymore. Do the following:

1. Check that the service is running with `sudo systemctl status postgresql@10-main.service` and optionally restart it.

2. Check that there is enough remaining disk space with `df -h`
If all seems right, your Slave instance probably *is not at fault* and this problem is the result of the Master having stopped. You can expect the Cointel team to resolve the problem; as long as your service keeps running, it will catch up eventually.

The Cointel team can request you for the name of your Slave's oldest and newest WAL segment, which can be found with `sudo ls -l /var/lib/postgresql_wal/10`.

### 5.2. Webapplication

The webapplication can simply be restarted if ever there is a problem.

#### ## Running the webserver

`docker ps` Will tell you about active processes and what containers are running, but you'll want to be able to start, stop and update your webserver. First go where the package was deployed.

```bash
> cd <server location>/Cointel_webserver_x.x.x.
```

Here you can run a number of docker compose commands:

- `docker-compose up -d` will run the whole show. If you omit the `-d` you can read along but CTRL+C or exiting the terminal will stop the containers again (so in production, use `-d`). The command will also build the containers if the weren't already.
- `docker-compose stop` will stop the containers.
- `docker-compose rm` will remove the containers.
- `docker-compose down` will stop and remove the containers.
- `docker-compose up -d <service-name>` will run a specif container. The same can be said for 'stop', 'rm' and 'down'.
- `docker-compose build --no-cache` Will (re)build from scratch. If you changed the configuration in docker-compose the '--no-cache' can be left out. If the software is updated, please do use that flag.
6. Backup and Update

6.1. Simulation Backup

Each time you restart the network the data belonged to the previous simulation will be erased, these data is only stored in the blockchain and no in another database (this is a design choice!). So the data will be available until the integrity of the blockchain.

6.2. Backups

The slave database instance could be backed up if you wish, but actually the slave instance is already a backup from master because everything is replicated. If something happens that renders the database useless, you can simply repeat the installation process and you'll have a brand new database.

The web application server does carry unique and important data that needs to be backed up. As explained by the setup and infrastructure, the service called CSDB will run the postgresql database storing all data that is related to your organization. This is something you want to back up regularly. All data from the read-only replicating database is of less importance to back up because the Slave instance *is* in essence already a backup of the Master instance.

#### Making a backup

There are two ways of making backups:

**By yourself**

During setup you configured the docker-compose file to mount a specific host volume in which the container will retain it's database. Per default this is <path-to-web-server>. That path thus contains a 'default' PostgreSQL10 database which you can backup from the host itself. You can setup a cronjob that regularly make a copy of that location; this entirely up to you.

You could even run PostgreSQL's backup tools from the host computer, of from somewhere else, given sufficient access permissions, because the database's data_directory itself is mounted somewhere on the host. (This is configurable in docker-compose.yml)

**By the container** The container runs a cron job that will run a script that makes daily and weekly backups by applying PostgreSQL's `pg_dump` and `pg_dumpall` tools. In the docker-compose configuration file you'll find two environment variables that correspond to how many days and how many weeks those backups are kept. The backups are written to a folder that can be mapped in the configuration (please see the configuration chapter for details). Per default this is <path-to-web-server>/csdb_backups. If you want to keep more (or less) backups, change the environment variables and restart the container.
*Please note that this does not copy the backup to some external host.* If you want to keep a copy on another machine or physical location, please make sure to add your own strategy; you can hook in to the configured backup folder.

#### Restoring from a backup

If something causes you to restore the customer specific database from a backup there's multiple ways to approach this, depending on what you used to make the backup.

If you simply made a copy of the mounted data_directory on the host (see docker-compose.yml), then you could stop the container, replace its csdb data_directory with your backup and start the container again.

If you used PostgreSQL's database backup tool to make the backup, you should use PostgreSQL's database restorations tool to put it back in place. This could be done if the tool is installed on the host machine, but also inside the docker container.

The container's backup process involves applying the standard tool `pg_dump` and `pg_dumpall` so from the container, you'll also be able to restore these dumps [as documented here](https://www.postgresql.org/docs/current/static/backup-dump.html).

You can run a shell inside the container, go to the backup location and apply PostgreSQL's method of restoring database dumps:

**Assuming the containers name is "webserver_csdb_1" (this can be looked up with `docker ps`)**

```bash
> cd <path-to-webserver>
> docker ps
> docker exec -it webserver_csdb_1 '/bin/sh'
   [ inside the container ]
> cd /var/backups/postgresql/<your-backup>
```

There use the tooling offered by PostgreSQL10, mentioned in the link. For examples

```bash
> cat filename.sql.gz | gunzip > my_readable_sql_file
> cat my_readable_sql_file
> cat my_readable_sql_file | psql dbname
```

or

```bash
> pg_restore -d dbname filename.custom
```
If you are unfamiliar with these, best try them out on a separated docker environment.

6.3. Project Update

The slave instance will be updated automatically because it will follow the Master database, so if that migrates, the slave will follow.

This also means that it is important to keep the webapplication software up to date; the slave database schema will automatically change; the code that it queries will not.

This section will describe how to perform an web application upgrade.

Once in a while you’ll need to update your old version of the Cointel webserver with a newer one. By using docker to run the show, it is really easy to keep both versions, or try out one while still running the other.

We assume the following.

- You are on version A, and updating to version B
- Version A was once extracted to /srv/cointel/Cointel_webserver_A
- Version A is currently online
- You downloaded (or otherwise acquired) the package as 'Cointel_webserver_B.tar.gz'
- You want to stop using A and start using B with little downtime
- You have some central location where the Customer database is stored, e.g. '/var/db/cointel/userdata'

If paths any of those paths are different, keep that in mind with the following instructions:

Extract B next to A

> sudo cp Cointel_webserver_B.tar.gz /srv/cointel/
> cd /srv/cointel/
> sudo tar -xzf Cointel_webserver_B.tar.gz
> sudo chown -R ${USER} Cointel_webserver_B.tar.gz
Map the customer specific database in docker-compose.yml to the same path as is set in version A. This can be found in the docker compose file, service CSDB, volume mapping 'var/lib/postgresql/data'

  > cd /srv/cointel/
  > cat Cointel_webserver_A/docker-compose.yml | grep "var/lib/postgresql/data"
  > cd /srv/cointel/Cointel_webserver_B
  > nano docker-compose.yml

If the database is contained in versions 'A' folder (e.g. '/srv/cointel/Cointel_webserver_A/csdb_postgresql'), this might be a good moment to give it a more suitable location. But best do this after version A was stopped so you don't risk someone committing data after making the copy.

Build (not run) your new containers

  > cd /srv/cointel/Cointel_webserver_B
  > docker-compose build --no-cache

This will make brand new containers that can exist next to the old ones. They shouldn't be started because that will result in conflicting ports (and the CSDB) of course.

Switch containers

  > cd /srv/cointel/Cointel_webserver_A
  > docker-compose down
  > cd /srv/cointel/Cointel_webserver_B
  > docker-compose up -d

Now is something is wrong, you can revert to version A simply by reversing those commands.
User Guide

Table of Contents
1 User profile .......................................................................................................................... 90
  1.1 Administrator .................................................................................................................. 90
  1.2 Manager ........................................................................................................................ 90
  1.3 User ................................................................................................................................ 90
2 Application Overview .............................................................................................................. 91
  2.1 Application structure ...................................................................................................... 91
  2.2 Application pages .......................................................................................................... 91
    2.2.1 Home page ............................................................................................................. 91
    2.2.2 Transactions page .................................................................................................... 92
    2.2.3 Transaction detail page .......................................................................................... 92
    2.2.4 Addresses page ........................................................................................................ 92
    2.2.5 Address details page ............................................................................................... 93
    2.2.6 Identities page ......................................................................................................... 93
    2.2.7 Identity detail page ................................................................................................... 93
    2.2.8 Blocks page ............................................................................................................. 94
    2.2.9 Blocks detail page .................................................................................................... 94
    2.2.10 Profile page ............................................................................................................ 95
3 Application functions .............................................................................................................. 96
  3.1 User creation .................................................................................................................... 96
  3.2 User log in ........................................................................................................................ 96
  3.3 Search ............................................................................................................................... 97
  3.4 Interpret results ............................................................................................................... 97
  3.5 Traversing the Bitcoin graph ........................................................................................... 99
4 Bad usage ............................................................................................................................... 99
# User profile

The application Cointel has three user profiles.

## 1.1 Administrator

The administrator is the user with the rights to manage other user accounts like managers and users. An admin can create and remove manager accounts.

## 1.2 Manager

The manager is the user who oversees a team of people that need to cooperate with each other. The manager can make (and disable) new user accounts, inviting them to Cointel. The manager can make (and delete) new cases and assign users to the case.

## 1.3 User

The user is the default account. A user can access all of the generic content (block-chain related content) and work on the cases that are relevant to him – these are managed by the manager.
2 Application Overview
Cointel is an application aimed at supporting Law Enforcing Agents in their investigations by providing insights in Bitcoin transactions and possibly the real world identity related to the address.

2.1 Application structure
The application has a home page and some subpages. The overall structure is displayed here.

2.2 Application pages
An example of every page will be shown here to get to know the application. For more details on specific functions please see chapter 3.

2.2.1 Home page
2.2.2 Transactions page

2.2.3 Transaction detail page

2.2.4 Addresses page
2.2.5 Address details page

2.2.6 Identities page

2.2.7 Identity detail page
2.2.8 Blocks page

2.2.9 Blocks detail page
2.2.10 Profile page
3 Application functions
In this chapter all functions of the Cointel application will be described.

3.1 User creation
As an administrator or manager you can create users by clicking on “Create user” under the profile tab.

3.2 User log in
When a user is created he can log in via the provided URL. Using the email address as username and a password created by the user themselves.
3.3 Search

Now that you are logged in you can start using the product. Use the search bar to look for a specific bitcoin address. An example is: 1HB5XMLmzFvJ8ALj6mfBsbfRoD4miY36v

3.4 Interpret results

If there are results for the bitcoin address you searched for you are presented with the specific address details page. This page contains an overviews of:

A. Basic information about the address (creation data, last transaction date, etc.)
B. Easy links to external search results
C. An overview of all input, output and unspent Bitcoins
D. Some statistics about the in and outputs (giving insights in the spent pattern
E. A graphical representation of the in and output
F. All known related addresses
G. Possible related identities found on the internet
H. Spent patterns in days of the year and hours of the day
I. All related transactions

The following screenshots show these features.
At the bottom of the page you will find an overview of the time of the day on which transactions took place (expressed in UTC) as well as the day of the year. The colors indicate the amount of transactions on that day.

If you want to look into specific transactions you can click on them at the bottom of the page.
3.5 Traversing the Bitcoin graph

When you want to go from one address to the next following a transaction you can use the small arrows next to the input and output addresses. A graphical representation of this path including a log of all traversed nodes will be part of a later release of the product.

4 Bad usage

In this section, the bad usages and the problems derived from this bad usage will be presented.

The main area of bad usage is misinterpretation of presented information. An example of bad usage would be:

Interpret a transaction from one address to another as being evidence of a crime. There can be multiple ways in which a transaction between addresses is unrelated to the owner of the address (mixing service for example).
Table of Content

1. Functional description .......................................................................................................... 102
2. Requirements .......................................................................................................................... 102
2.1. Hardware Requirements ..................................................................................................... 102
2.2. Software Requirements ...................................................................................................... 102
2.3. Data requirements ............................................................................................................. 102
2.4. Legal considerations .......................................................................................................... 102
3. Installation Guide .................................................................................................................. 102
4. Configuration .......................................................................................................................... 103
5. Error Management .................................................................................................................. 104
6. Backup and Update ............................................................................................................ 104
1 Functional description

Dence wallet investigator is a tool to aid efficient detection and analysis of traces left by client applications for virtual currencies. It can extract traces like address labels, public and private keys, transactions or traces of usage patterns from the evidence files. In the current version, the tool does not yet implement a graphical user interface and can only be used on the command line. Because of the early implementation stage, the documentation covers only the basic concepts. The documentation will be updated when a user interface is available.

2 Requirements

2.1 Hardware Requirements

- Intel-compatible PC

2.2 Software Requirements

- A 64-bit operating system: either Microsoft Windows 7 or later, MacOS 10.13 or later, or Linux

2.3 Data requirements

The investigator analyzes a wallet artefact, e.g. a wallet.dat file which was seized from a physical device. The input file is read and intelligence is extracted from the file. The tool supports to read different wallet file formats.

2.4 Legal considerations

The tool is used as an automation aid for forensic analysis of seized devices. It is always executed in the context of a single case/investigation. It is assumed that extracting the personal information from the evidence is based on a legal mandate. Assuming this scenario, no legal difficulties are foreseen at this point.

3 Installation Guide

- The software is distributed as a platform-specific archive, e.g., the following archive is a release for the Windows platform:
  - wallet-investigator_v0.0.3-win.zip
- The platform-specific archive is available through the dence customer portal:
  - https://customer.dence.de/en
- The software archive contains the following files:

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>wallet-investigator(.exe)</td>
<td># executable binary</td>
</tr>
<tr>
<td>LICENSE</td>
<td># licencing information</td>
</tr>
<tr>
<td>README.md</td>
<td># basic information</td>
</tr>
<tr>
<td>docs/</td>
<td></td>
</tr>
<tr>
<td>index.html</td>
<td># user documentation (installation, configuration,</td>
</tr>
<tr>
<td>dist-files/</td>
<td></td>
</tr>
<tr>
<td>examples/</td>
<td># directory with some example wallet files</td>
</tr>
</tbody>
</table>

- Additionally, the Berkeley DB extractor can be installed as a separate tool. The Berkeley DB extractor reads Berkeley DBs (used by * Core Wallet files, e.g., Bitcoin Core Wallet files) and transforms them to a JSON representation. This conversion step is necessary for further analysis with wallet investigator. Please download the files, if you want to analyse Bitcoin Core wallet files. Currently Berkeley DB ver <= 5.6 is supported (including Bitcoin, Dash, Litecoin and
ZCash wallet files). The Berkeley DB extractor is distributed as a separate software archive, which includes the source code and binaries for all 3 platforms (Windows, Linux and Mac OS).

- We recommend to extract both tools, wallet investigator and Berkeley DB extractor, to the following directory structure:

```plaintext
wallet-tools/  # the main directory to install the provided tools
    wallet-investigator  # just extract the contents of the
                           # wallet investigator archive here
    bDBExtractor      # optional, extract the contents of
                       # the Berkeley DB extractor here
```

- If all files are correctly extracted, the following structure of directories and files should be created:

```plaintext
wallet-tools/
    wallet-investigator/
        wallet-investigator
            README.md       # basic information
            LICENSE         # licensing information
            dist-files      # folder with example wallet files
    bDBExtractor/
        Darwin_x86_64/  # Mac OS binary Berkeley DB extractor
        Linux_x86_64/   # Linux binary Berkeley DB extractor
        Windows_x86_64/ # Windows binary Berkeley DB extractor
        src/            # source code
```

### 4 Configuration

Configuration settings are stored in a text file “config.yaml” in the directory “.dence/wallet-investigator”. The directory can be found in the users home directory on all platforms:

- Windows: c:\Users\[user name]\.dence\wallet-investigator
- Linux: /home/[user name]/.dence/wallet-investigator
- Mac OS (Darwin): /Users/[user name]/.dence/wallet-investigator

Configuration file and corresponding directory are automatically created, if no configuration file can be found.

The following example shows default setting for the user “analyst” on a Mac OS X platform. It is possible to change the folder of the Berkley DB extractor, the standard storage folder for wallet analysis results and dence API settings.
5 Error Management
Errors are printed to the console. Please report bugs or feature requests to blockchain@dence.de.

6 Backup and Update
You may want to backup your “config.yaml” and generated analysis reports. Program updates are made available through dence customer portal (https://update.dence.de).
# User Guide

## Table of Content

1. User profile .......................................................................................................................... 107
2. System function .................................................................................................................... 107
2.1. Overview ............................................................................................................................ 107
2.1.1 Wallet investigation ......................................................................................................... 107
2.1.2 Basic statistics ................................................................................................................ 107
2.1.3 General information / help ............................................................................................ 107
2.2. Usage example .................................................................................................................. 107
2.2.1 Basic example ................................................................................................................ 107
2.2.2 Example - Core wallet file analysis .............................................................................. 108
2.2.2 Example - HD wallet file analysis ................................................................................. 108
1 User profile
Dence wallet investigator is a desktop application. The application uses the user profile separation of the operating system.

2 System function

2.1 Overview
Wallet investigator is a command line tool. Interaction with the tool is based on a set of command line parameters and the normal execution of the tool has the following structure:

```bash
wallet-investigator [command] [parameter list] [flags]
```

The most recent documentation of commands, parameters and flags is available through the software and the help command. Commands can be classified according to their purpose: 1. wallet investigation, 2. basic statistics and 3. general program information.

2.1.1 Wallet investigation

<table>
<thead>
<tr>
<th>command</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>bDBToJSON</td>
<td>Interface to external Berkley DB extractor to convert a Berkley DB to a key value JSON representation</td>
</tr>
<tr>
<td>wallet</td>
<td>Investigate wallet file</td>
</tr>
<tr>
<td>xKey</td>
<td>Extended key utility (HD key) to determine and check first addresses</td>
</tr>
</tbody>
</table>

2.1.2 Basic statistics

<table>
<thead>
<tr>
<th>command</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>apiStatus</td>
<td>Query most recent block available through dence API</td>
</tr>
<tr>
<td>addrStats</td>
<td>Query statistics about 1 or more addresses through dence API</td>
</tr>
<tr>
<td>addrTxStats</td>
<td>Query address transaction statistics through dence API</td>
</tr>
<tr>
<td>blockBasics</td>
<td>Query basic block information</td>
</tr>
<tr>
<td>rawTx</td>
<td>Query parsed transaction data</td>
</tr>
</tbody>
</table>

2.1.3 General information / help

<table>
<thead>
<tr>
<th>command</th>
<th>info</th>
</tr>
</thead>
<tbody>
<tr>
<td>help</td>
<td>Print this help message</td>
</tr>
<tr>
<td>help [command]</td>
<td>Get help about a specific command</td>
</tr>
<tr>
<td>version</td>
<td>Show program version</td>
</tr>
</tbody>
</table>

2.2 Usage examples

Descriptions of examples in this Section use Windows-specific path separators / executable names (i.e., wallet-explorer.exe in Windows vs. ./wallet-explorer in MacOS and Linux). After adjusting the OS-specific separators and executable name, examples can be executed under Linux and MacOS.

2.2.1 Basic example

The following steps describe a basic example for the interaction with the program. You can find a web video showing the interaction with the tool in the online documentation.

1. Start your preferred command line shell (e.g., command prompt (standard), Git Bash)
   o Command prompt can be easily found by pressing the Windows button Win and typing cmd. Press Enter to start the command prompt.
If Git Bash is installed, type `git` instead of `cmd` and press Enter.

2. Go to the program installation folder.
   - If the software is stored in the folder `wallet-tools` on the desktop type:
     ```
     cd Desktop\wallet-tools\wallet-investigator
     ```

3. Start the program:
   - Command prompt: `wallet-investigator.exe`
   - Git Bash: `\.\wallet-investigator.exe`
   - You should see a general help page listing possible commands and parameters.

4. Query current status of the remote blockchain data:
   ```
   wallet-investigator.exe apiStatus
   ```

   - Executing any command for the first time creates a config file in the user folder `c:\Users\<your user name>\.dence\wallet-investigator\config.yaml`. More information is available in the Section configuration of the operation manual or in the online documentation.
   - By default created reports are stored in the folder `.dence\wallet-investigator`, but this can be changed either by modifying the config file or by adding the command line parameter `--projDir <path>` to specify a project specific path.

5. Query some basic address statistics:
   ```
   wallet-investigator.exe addrStats 1F1tAaz5x1HUXrCNLbtMDqw6o5GNn4xqX
   ```
   (address of Silk road seized coins)

2.2.2 Example - Core wallet file analysis

The following steps describe the analysis of a Core wallet file (e.g., generated by Bitcoin Core software). You can find a web video showing the interaction with the tool in the online documentation.

Analyze a wallet.dat file distributed with the program package:

1. Convert the file `wallet.dat` to a JSON representation using the Berkeley DB extractor:
   ```
   wallet-investigator bDBToJSON <source wallet filename> <destination wallet filename>
   ```

2. Execute `wallet-investigator.exe wallet <destination wallet filename>`

3. Open report in default directory `<user home directory>/.dence/wallet-investigator` and investigate your traces.

4. Keys can be copied in full length from the report (even when dots are shown to improve readability of the table)

2.2.3 Example - HD wallet file analysis

The following steps give examples of the analysis of a HD wallet. You can find a web video showing the interaction with the tool in the online documentation.

### Extended public key

- Query if an extended key was in use:
  ```
  wallet-investigator.exe xKey
  xpub6BfKpqjTWvH21wJGWFexLppb8sU7C6FJge2kWb9315oP4ZVqCXG29cdUtkyu7YqhhYfA5nt63nzcN2HYmqXYH0dxYo8mm1Xq1dAC7YtodwUR m/0
  ```
- Change flags to collect information about more derived keys:
  ```
  wallet-investigator.exe xKey
  xpub6BfKpqjTWvH21wJGWFexLppb8sU7C6FJge2kWb9315oP4ZVqCXG29cdUtkyu7YqhhYfA5nt63nzcN2HYmqXYH0dxYo8mm1Xq1dAC7YtodwUR m/0 --from 0 --to 100
  ```
Table of Content
1. Functional description ............................................................................................................. 111
2. Requirements ............................................................................................................................ 111
2.1. Hardware Requirement ........................................................................................................ 111
3. Installation and User Guide .................................................................................................... 111
3.1. Persistent Monitor ................................................................................................................. 111
3.2. Ephemeral Monitor .............................................................................................................. 113
4. Configuration ............................................................................................................................. 114
5. Error Management ................................................................................................................... 114
1. Functional description

The TNO dark web monitor consists of two separate tools:

a. a persistent monitor that allows for longitudinal views on crawled and scraped dark web forum data (to be extended later with marketplaces)

b. an ephemeral monitor that allows for stateless and controllable access to dark web marketplaces.

The ephemeral monitor can be used for on the spot inspection (‘following up a hunch’) in the pre-stages of a formal police investigation. The persistent monitor can, once the investigation has formally started, serve as a fall-back for gathering historical evidence based on a concrete suspicion.

The persistent dark web monitor is a full-fledged monitor for analysing dark web forums. The monitor runs as a web service, and indexes posts and entire forums. The monitor applies various forms of NLP (natural language processing) to raw forum posts. Including topic labelling (drugs, weapons, cybercrime, ...), language identification and topic trends.

The persistent dark web monitor contains currently over 30M posts, from a variety for forums, which in turn are associated with dark web marketplaces.

The persistent monitor is mature and can be demonstrated in a Field Lab in 2019. The ephemeral monitor has lower maturity and will also be subjected to testing in the first round of Field Labs in 2019. Acceptance tests (to be developed within TITANIUM) still have to be carried out.

2. Requirements

2.1. Hardware Requirement

The monitor runs on TNO managed servers and can be used from any platform or machine using current browsers.

3. Installation and User Guide

3.1. Persistent Monitor

It is necessary to obtain a login code (username + password) from TNO. The persistent monitor is accessible as a website. After logging in, forum, host and user statistics as well as forum content can be searched and browsed through. These options are available through the links on the menu to the left of the page. Cryptocurrency information is added as experimental content.
3.2. Ephemeral Monitor

The ephemeral monitor can be accessed over an SSH connection. To do so, one needs an SSH client and an X server for showing the graphical user interface. On Linux this is typically available if you run a graphical environment. For Windows, a (portable) application that provides both is available, called MobaXTerm. As separate software, Putty and XMing server may be used. For Mac OS, an X server needs to be installed, e.g. XQuartz.

To run the ephemeral monitor, you need to obtain connection information and a private key from TNO. A detailed installation guide can be obtained through TNO also, at time of writing, this guide can be found at https://nextcloud.darkwebmonitor.eu/index.php/s/VwBAhdjeVrCMNug.

The ephemeral monitor allows to start a search on the Dream Market by filling out the search form as shown in the figure below. Clicking on the Open GUI button will open a web interface that allows to browse the pages that were cached from the crawl and will show relationships of entities discovered during the crawl.
4. Configuration
The ephemeral monitor requires an access link (onion) to the market of interest. This is retrieved from popular listing pages, but in case of connection issues, a link may be configured manually.

5. Error Management
Errors in the persistent monitor are logged and handled without need of user interaction. Errors in the ephemeral monitor are shown when the login is made with an open terminal. Debug information will be printed on screen.
Table of Content
1. Functional description ......................................................... 117
2. Requirements ........................................................................ 117
   2.1. Hardware Requirement ................................................... 117
   2.2. Software Requirement and conditions ........................... 117
   2.3. Data requirements .......................................................... 117
3. Installation Guide ................................................................. 118
4. Configuration ......................................................................... 118
1. Functional description

This package contains a scraper for darkweb markets (currently working on Dream Market). It uses Tor and Scrapy to collect product listings. These are stored in ElasticSearch, which allows analysis in Kibana or loading the listings in a custom front-end. The GUI for the scraper itself will display the 50 latest scraped market listings. Images in product listings are stored on the named Docker volume "stored_images". The file /scraper/src/market/market/spiders/listings_spider.py contains the spider classes. The DreamMarketSpider provides an example for Dream Market, which can serve as a basis to build more scrapers. Keep in mind that the login flow can differ per market.

The package is included in the Dark Web Monitor and is not available as a stand-alone software tool with Graphical User Interface.

The tool is mature and can be demonstrated in a Field Lab in 2019. Acceptance tests (to be developed within TITANIUM) still have to be carried out.

2. Requirements

2.1. Hardware Requirement

A machine with a modern processor and at least 8GB of RAM, but more is recommended. Disk usage totally depends on how much data is going to be scraped, but as images are downloaded, this grows quickly. Because of the parallel downloading of files, a fast internet connection is needed for good performance.

For server deployments with 16GB RAM or more, the Java heap size for Elasticsearch can be increased by enabling ES_JAVA_OPTS in docker-compose.yml. Documentation about this setting can be found on: https://www.elastic.co/guide/en/elasticsearch/reference/2.3/setup-configuration.html

2.2. Software Requirement and conditions

All the scripts used for creating the dockerized bitcoin testbed are developed in Java and Python. The tool runs on the latest docker versions (no version dependencies)

2.3. Data requirements

When a crawl is running, listings, vendor profiles and ratings are stored in ElasticSearch. The index names are "products", "vendors" and "ratings". If a listing or vendor profile contains images, they will be stored in a Docker volume with the name "stored_images".
3. Installation Guide

- Install GIT

- Install Docker CE
  [Docker CE installation guide](https://docs.docker.com/engine/installation/)

- Install Docker Compose (Linux Only)
  [Docker Compose installation guide](https://docs.docker.com/compose/install/)

To use Docker without sudo

```bash
# Add your username to the Docker group
sudo usermod -aG docker $USER

# Logout and login again for this to take effect
Logout
```

4. Configuration

Having Docker and Docker Compose installed, deployment should be as simple as:

```bash
# Go to the market-scraper directory
cd market-scraper

# Build the Docker images
docker-compose build

# Start the Docker containers in daemon mode
docker-compose up -d

# To check the status of the containers enter
docker-compose ps
```
# Operation Manual

## Table of Content

1. Functional description ........................................................................................................... 121
2. Installation Guide .................................................................................................................. 121
1. **Functional description**

The package is included in the Dark Web Monitor and is not available as a stand-alone software tool with Graphical User Interface.

- This scraper learns to extract information from raw web content (HTML), using machine learning and text mining techniques. The latest version supports scraping of *usernames, titles, posts text and post dates* from forums.

- Its model was trained with a dataset of 3493 forum pages from 26 different forums (both darkweb and clear web), and 8 different forum frameworks, some of them in various versions:
  - FluxBB
  - MiniBB
  - phpBB (standard and PHP-Nuke port)
  - vBulletin (versions 3.8.4, 3.8.8, 4.2.3, 4.1.12)
  - XenForo
  - SMF (versions 1.1.8, 1.1.19, 1.1.21, 2.0.2, 2.0.11, 2.0.14, 2.0.15)
  - MyBB
  - One custom forum framework

It contains a simple annotation tool to annotate training data manually.

The tool is mature and can be demonstrated in a Field Lab in 2019. Acceptance tests (to be developed within TITANIUM) still have to be carried out.

2. **Installation Guide**

A `docker run` command is sufficient for running the software, once the image has been pulled to a local machine.
REMARK

This package is not available as a stand-alone software tool with its Graphical User Interface, for this the operation manual could seem poor. However, in the context of TITANIUM and the field labs this tool/functionality is integrated in the “Dark Web Monitor”
Captcha Solver
Operation Manual
TNO
October 2018
# Table of Content

1. Functional description ........................................................................................................ 125
2. Requirements ..................................................................................................................... 126
   2.1. Hardware Requirement ............................................................................................... 126
   2.2. Software Requirement and conditions ....................................................................... 126
   2.3. Data requirements ....................................................................................................... 126
3. Installation Guide ................................................................................................................ 126
   3.1. Chain of dependences ............................................................................................... 126
4. User Guide .......................................................................................................................... 127
5. Backup and Update ............................................................................................................. 127
1. Functional description

Captcha are automated tests often used by public websites to tell human users and agents (computers) apart. If the user does not pass the captcha test the website can choose to deny the user its services. The following figure shows an example of a captcha where the user is asked to identify and fill in the letters within the rectangular box.

To help an agent automatically crawl through such website we aim to design a system that can automatically solve captchas. Since the whole purpose of captcha is to design tests that are difficult for an agent to pass, this is not an easy problem. One of the option to circumvent the challenges presented by this problem is to keep a human in the loop when solving captchas. The idea is that the efforts from the human should be minimal (and keep on decreasing as agent learn over a course of time) in solving/assisting an agent in solving the captchas, however, the agent must not be completely helpless if it faces a tough challenge.

Currently, the tool works by first removing unnecessary text in the captcha image, e.g., by removing any color in the image that is not blue. Then the tool identifies the rectangle in the image by the unique pixel pattern a rectangle poses. Finally, we the tool applies a publicly available optical character reader (OCR) to extract text from the image. This text is then further process to best identify the four letters that are the right solution to the captcha.

The package is included in the Dark Web Monitor and is not available as a stand-alone software tool with Graphical User Interface.

The package is mature enough to be demonstrated and used at hackathons and field labs.
2. Requirements

2.1. Hardware Requirement
The tool on its own requires minimal hardware. Ideally, the tool is to be integrated with a automatic scraper and crawler framework. Addition of this tool to this framework, in our opinion, will require no extra hardware. (So a simple modern machine will do.)

2.2. Software Requirement and conditions
This tool is fully dockerized. Thus, docker is required for this tool. The tool has some dependencies all of which are noted in the docker file. They are listed below:

- git
- g++
- autoconf
- automake
- libtool
- pkg-config
- libpng-dev
- libjpeg62-turbo-dev
- libtiff5-dev
- zlib1g-dev
- libleptonica-dev

The tool also depends on library tesseract that can be installed via github:

RUN git clone https://github.com/tesseract-ocr/tesseract.git && \
    cd tesseract && \
    ./autogen.sh && \
    ./configure && \
    make install && \
    ldconfig &&

2.3. Data requirements
Tesseract requires the dataset publicly available at:
git clone tessdata (https://github.com/tesseract-ocr/tessdata)

3. Installation Guide

3.1. Chain of dependences
If docker is installed, then the dockerfile will take care of the installation. Else a linux system with python and installation of above mentioned dependencies is also straightforward. (apt-get ... )
4. User Guide

Currently, the captcha solver can be used with help of a python script that takes in an captcha image as a .jpg. The output of the script is the predicted letters. Example

```
python apply_ocr_dreammarket_captcha.py dreammarket1.jpg
```

The output of this is a string that can be either written to a file or can be directly integrated with a python-based crawler that can take the string and use it to solve the captcha. The captcha solver is used in the ephemeral monitor.

5. Backup and Update

The code for the tool along with all the dependencies and instruction is available at: TNO’s own repository management tool, under the repository CAPTCHAs. All the updates are checked into this repository.

REMARK

This package is not available as a stand-alone software tool with its Graphical User Interface, for this the operation manual could seem poor. However, in the context of TITANIUM and the field labs this tool/functionality is integrated in the “Dark Web Monitor”
Table of Content
1. Functional description ................................................................. 130
2. Installation Guide ........................................................................ 131
1. Functional description

- Multimodal vendor profiling consists of a number of tools to describe a vendor based on their collected data, i.e., the content scraped from Dark web fora and marketplaces (e.g., text and images).

- Text analysis is comprised of TNO’s authorship attribution. This tool classifies text as pertaining to a certain vendor. Features for classification are constructed using compression techniques and TF-IDF. Furthermore, the tool gives insights into which parts of the text contribute to the classifiers choice.

![Figure 1: Example output of text classified as belonging to vendor x. The coloured sections indicate which parts contribute positive (highlighted in green) or negative (marked red) to the decision.](image)

- Image analysis is a fusion of both dence’s byte level image analysis tools, that perform similarity measurements between individual images, and TNO’s concept detection module that compares images on a semantic level. Concept detection is able to describe what is in an image or localizes specific objects in order to, for example count them. It uses the latest trained deep learning models for images classification plus tools to easily add new concepts.

- The aggregated results of the analysis tools help in discovering vendor pseudonyms or build up a general user profile.

The toolset is not yet advanced enough to be demonstrated. A number of functional modules are still being developed.
2. Installation Guide

A `docker-compose up -d` command is sufficient for running the software, once the images have been pulled to a local machine.

**REMARK**

This package is not available as a stand-alone software tool with its Graphical User Interface. Currently, it is in a develop phase, for this the operation manual could seem poor. However, in the context of TITANIUM and the field labs this tool/functionality will be integrated in the coming months in the “Dark Web Scraper”