DIATEAM

a Cy4gate company

INTEGRITY CHECKER SCRIPT

EN - 1.1

Reference: DIATEAM-integrity-checker_script-EN-1.1

DOCUMENT INFORMATIONS

Reference	DIATEAM-integrity-checker_script
Version	1.1
Nombre de pages	17

DIFFUSION

Recipient(s)	CLIENT

CHANGELOG

Version	Autor(s)	Date	Description
1.0	FBE		File creation
1.1	FBE	28/05/2026	IntegrityChecker config path modification

DIATEAM - 31 rue Yves COLLET - 29200 Brest - FRANCE | Tel : +33(0) 298 050 050

CONTENTS

DIATEAM - 31 rue Yves COLLET - 29200 Brest - FRANCE | Tel : +33(0) 298 050 050

1. CONTEXT

This script is used to analyse and permit solve the cyber range data integrity on 3 main points:

- catalog,
- services configuration,
- database data.

This is a command line script that can be directly launch from master from location:

/usr/share/hynesim/scripts/IntegrityChecker/.

DIATEAM - 31 rue Yves COLLET - 29200 Brest - FRANCE | Tel : +33(0) 298 050 050

2. SETUP

2.1 CONTENTS

Folder contents the following sub folder and files:

Folder :

- analysisResult (see content below)

Python files :

- catalogCheck.py,
- dbCheck.py,
- servicesConfigurationCheck.py,
- dbUtility.py
- catalogUtility.py,
- servicesConfigurationUtility.py,
- utility.py
- integrityCheck.py

Installation file :

- requirements.txt

2.1.1 analysisResult folder

Each time you generate an analysis, the report is stored into this folder. You will find :

- **CatalogCheck.xml :** analysis result generated by catalogCheck.py class.
- **DBCheck.xml :** analysis result generated by dbCheck.py class.
- **ServicesConfigurationCheck.xml :** analysis result generated by servicesConfigurationCheck.py class.
- **AnalyseResult_<DateOfGeneration>.xml** : The concatenation of all results from CatalogCheck.xml, DBCheck.xml and ServicesConfigurationCheck.xml files.

The AnalyseResult_<*DateOfGeneration*>.xml can be sent to Diateam support team, for analyze purpose.

2.1.2 catalogCheck.py

This file contains elements to analyze the catalog filesystem content of your Cyber Range. As it use Hapi scripts, you must configure the service and fill the **config.json** file with your Cyber Range informations to permit to this class to be functional.

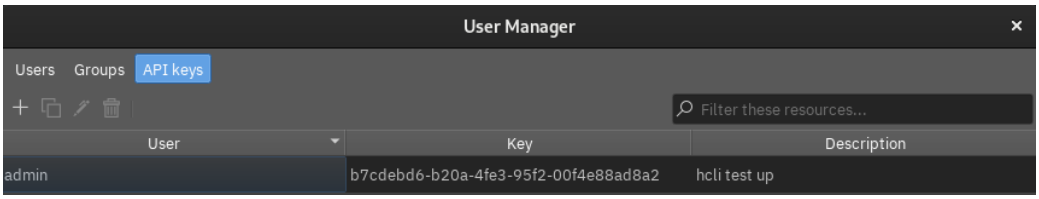
The first version use config.json file located **/usr/share/hynesim/scripts/IntegrityChecker/** folder.

For future version, this configuration file will be relocated to **/etc/integrityChecker/**.

config.json content file :

```
{
        “hapiKey": "YourApiKey”
        “endpoint": "http://api.xxx.diateam.range",
}
```

Change the **X-API-Key** value by the API key of your cyber range. You will find it by hyneview from the **User manager panel provided in the Window** menu :



The endpoint is stored into **/etc/hosts** file. Open the hosts file on Master server, and go to Diateam services section, to know the api link name.

This script can detect :

- **unused entities**
- **orphan entities**
- **entities template present into topologies definition**
- **duplicated uuids into topologies definition**

2.1.3 dbCheck.py

This file contains elements to analyze the DB content of your Cyber Range.

This script can detect :

- **invalid id -1**
- **Orphan entities**
- **Orphan users**
- **Orphan groups**
- **Orphan ACLs**

2.1.4 servicesConfigurationCheck.py

This file contains elements to analyze the service configuration content of your Cyber Range.

To work correctly, hynesim-master, hynesim-node and hynesim-glacier need some default values. These default values are present to initialize properly the hynesims service. So, this analysis will check if some key/values exists on client configuration to allow them to be written if they are not present.

If some values are less than Diateam requirements, these values will be replaced by default values.

If keys/values are not present, the system will take their proper values, that does not match Diateam recommendation. These keys/values can block the communication between client and server, for example, request to synchronize too often, …

This script will analyze:

- Hynesim glacier configuration file
- Hynesim master configuration file
- Hynesim node configuration file

2.1.5 dbUtility.py, catalogUtility.py and utility.py

These files are utility class. It contains functions most used into check classes.

2.1.6 integrityCheck.py

The entry point of program.

2.1.7 requirements.txt

File containing python dependencies to install. To perform dependencies installation launch one of the following scripts:

If you platform has Internet access, you can launch the following command :

pip3 install -r requirements.txt

Otherwise Diateam can send you offline packages and the procedure to install packages offline, before installing IntegrityChecker.

2.2 UPLOAD AND LAUNCH THE SCRIPT

If the project is not present into **/usr/share/hynesim/scripts/IntegrityChecker**, you can copy the content of **IntegrityChecker** folder on CyberRange.

Open a terminal directly into IntegrityChecker folder.

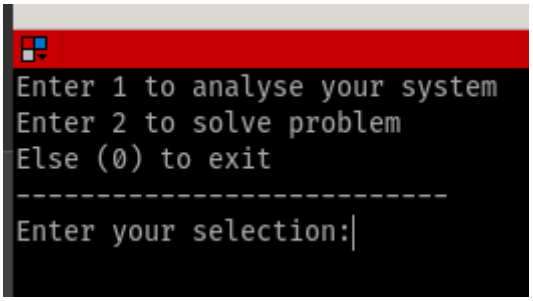
2.2.1 Use

Now, you can launch the program.

python3 ./integrityCheck.py

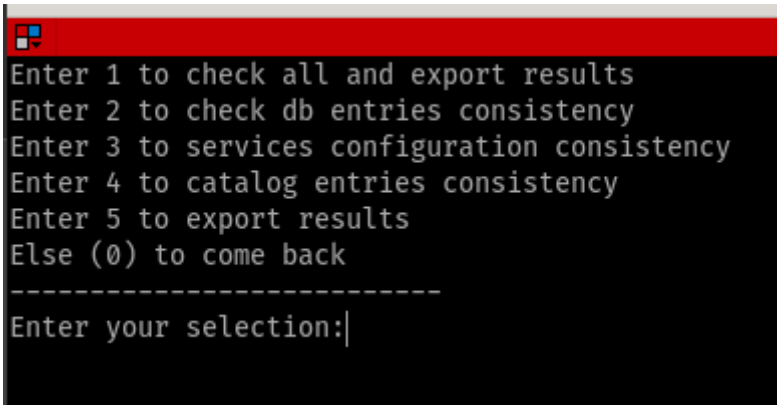
The first menu is displayed and ask you to:

- Analyse your CyberRange
- Resolve problems



2.2.2 Analyse data

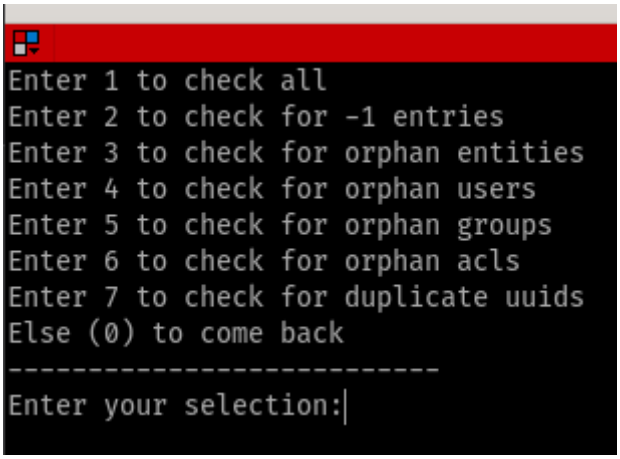
If you enter 1, you will enter into analyse menu.



You will find each analyse action you can do. Choose the analysis action to launch.

You can choose to analyse one element or launch all analysis actions.

When analysis are made, you can export results into an xml file to send it to Diateam support team.



Check for invalid id

This analysis will search into diamesh.db and catalog.db files, if invalid ids (with –1 value), were present. These ids may be present due to bad behavior, when persist some entities, and can cause hynesim-master service crash.

These entries must be removed from the database.

Check for orphan entities

This analysis would search into catalog.db file if each Parent to Child association declaration is valid (Domain UUID always present).

If one of the domains UUIDs is not present, then the association can be removed from entity_relationship table in catalog.db database file.

Check for orphan users

This analysis would search in user/group associations if referenced user exists and is valid.

These ghost users can be removed from the join_groups_users table in diamesh.db database file.

Check for orphan groups

This analysis would search in user/group associations if referenced group exist and is valid.

These ghost groups can be removed from database.

Check for orphan acls

When you create an entity, folder or topology, a right is automatically associated. But when you delete these entries, due to some hynesim-master service crash, some entries are not correctly deleted.

These acls affected to non existing element can be removed from the join_groups_users table in diamesh.db database file.

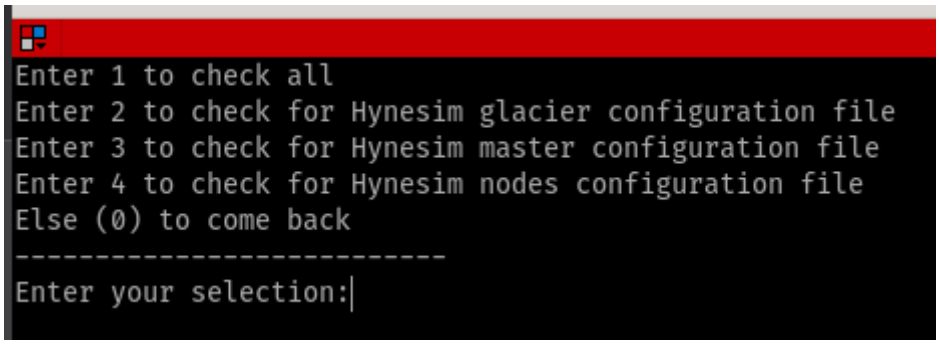
Check for duplicate uuids

The UUID associated to elements registered in database must be unique.

Sometimes, we can detect distinct entities having the same associated UUID.

This analysis can detect all of them. The judicious entry (if any) will be kept.

Service configuration analysis explanations

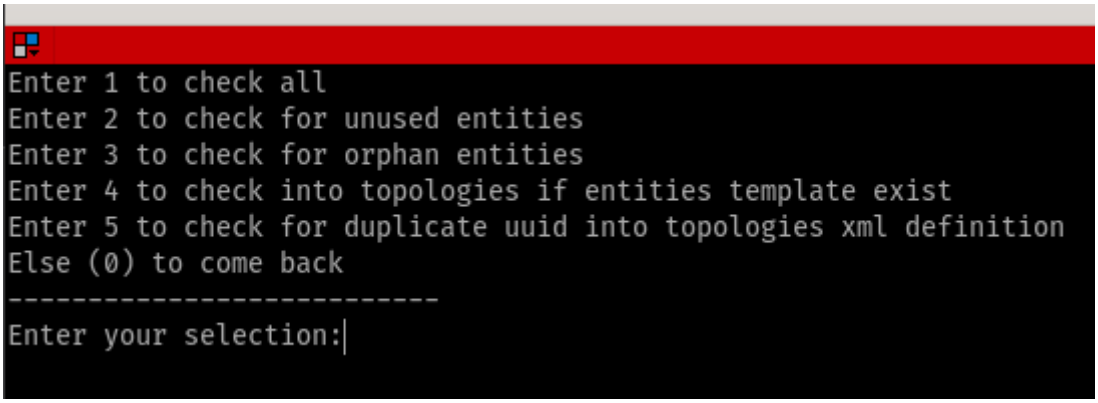


Check for Hynesim glacier and Hynesim master configuration files

These analysis will check for requested minimum key/value presence into CyberRange configuration. These values are also tested to check if the value is not less than requested value.

Check for Hynesim node configuration file

Currently, no mandatory keys/value are checked.



Check for unused entities

This analysis will list between all entities registered, those who are not used into any topology. This is an information for administrator to clean FileStorage if needed.

Check for orphan entities

Do nothing for the moment

Check for entities template present into topologies definition

Check into each topology definition, if a isTemplate is set to 1. Into topologies we have implementation of entities, and no template declaration except for domains for update purpose.

Check for duplicated UUIDs into topologies definition

This analysis will detect the duplicated UUID declaration in all topologies definition. These duplicated UUID can be any of included entities and/or wires.

These duplicated UUIDs cause the impossibility to open a topology, if another one is already opened containing the same duplicated element. Physically, you have one resource share by multiple topologies. Practically, the first who open the element lock it, and then the others topologies containing the same UUID could not be opened.

In this analyse, we distinguish all elements, to be able to correct the topology.

You can launch a complete analysis, and extract the results, for Diateam support team.

The report is generated into "**analysisResult**" folder. His name begins with AnalyseResult_ + Date.xml file.

You can also find 3 other files into "**analysisResult**" folder:

- DBCheck.xml
- ServiceConfigurationCheck.xml
- CatalogCheck.xml

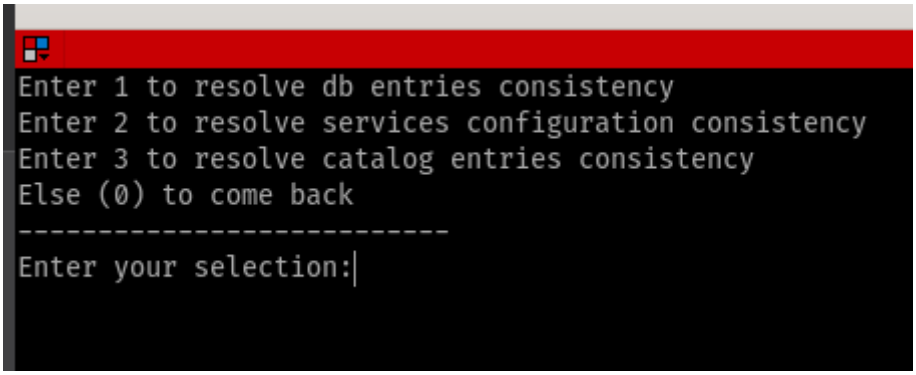
These files are automatically override, each time, you launch an analyse.

The analyse result file is not deleted.

2.2.3 Resolve data inconsistencies

For resolution part : Be sure nobody is working on CyberRange, when you want to solve any problem because the python script resolution does not communicate with hynesim. And then we cannot guaranty that the same data is not modified by correction process and by hynesim at same time.

If you enter 2, you will enter resolve menu.



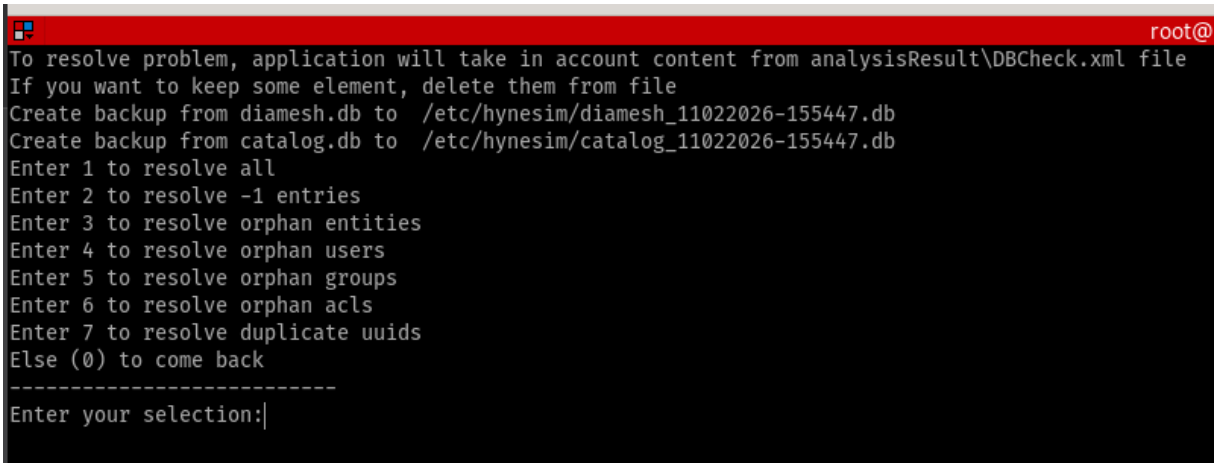
You will find each possible resolve actions. Choose the resolve action you need to launch associated process.

The analyse of the associated point must be analysed first to permit the resolution of detected problem (if any) because the elements needed to solve the problem are extracted from "**analyseResult"** folder.

- Inputs to resolve DB problems, are extracted from **analyseResult/DBCheck.xml** file.
- Inputs to resolve service configuration problems, are extracted from **analyseResult/ServiceConfigurationCheck.xml** file.
- Inputs to resolve catalog problems, are extracted from **analyseResult/CatalogCheck.xml** file.

If you want to keep elements, you must suppress from these files the corresponding element to keep.

Db resolve explanations



Each time you open the db resolve menu, a **diamesh.db** and **catalog.db** backup is made into **/etc/hynesim** folder. You will find diamesh_date.db and catalog_date.db with the current date displayed.

Resolve invalid id

The resolution of invalid ids, will delete the registered line in corresponding table.

There is no specific problem in the resolution of this kind of problem in the database.

Resolve orphan entities

The resolution of orphan entities, will delete Invalid lines from entity_relationship table from catalog.db database file.

There is no specific problems in the resolution of this kind of problem in the database.

Resolve orphan users

The resolution of orphan users will delete invalid lines in the join_groups_users table from diamesh.db database file.

There is no specific problem in the resolution process of this kind of problem in the database.

Resolve orphan groups

The resolution of orphan groups, will delete invalid lines in the join_groups_users table from diamesh.db database file.

There is no specific problem in the resolution of this kind of problem in the database.

Resolve orphan acls

The resolution of orphan acls, will delete invalid lines in the join_groups_users table from diamesh.db database file.

There is no specific problem in the resolution of this kind of problem in the database.

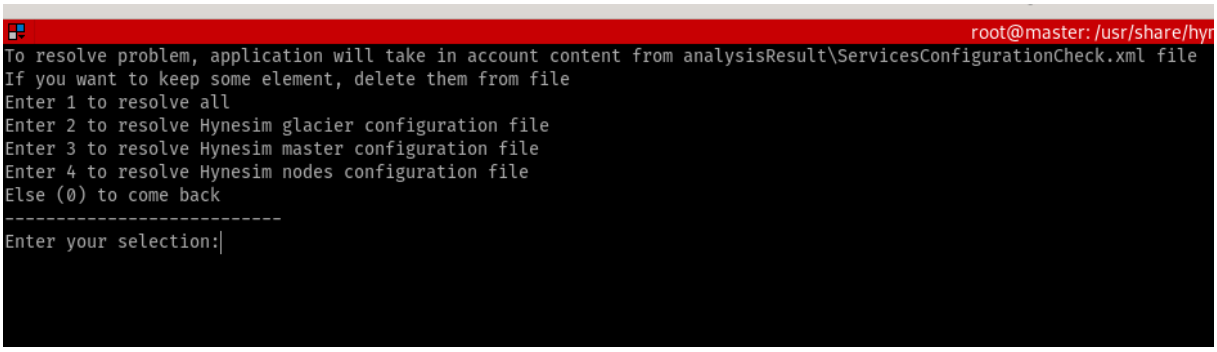
Resolve duplicate uuids

The resolve duplicated uuids, will take element to delete from **analyseResult/DBCheck.xml** file, and launch the SQL delete command on corresponding table.

For resolution process, 2 distinct cases can be found:

- One element has a physical path on filestorage, and others not. Then we will keep the entry with non empty filestorage path. Others are included topology template duplications and will be deleted all.
- All duplicated entries are included topology template with empty filestorage path, and then can be delete all.

There is no specific problem in the resolution of this kind of problem in the database.



Resolve Hynesim glacier configuration file

The resolution of hynesim glacier service configuration, will add missing key, and/or replace bad values into /etc/hynesim/hynesim-glacier.conf configuration file.

Add missing key into hynesim-master.ini file, to ensure the good system behaviour.

Replace bad values, by minimum Diateam requested values, to ensure system behaviour.

Resolve Hynesim master configuration file

The resolution of hynesim master service configuration, will add missing key, and/or replace bad values into /etc/hynesim/hynesim-master.ini configuration file.

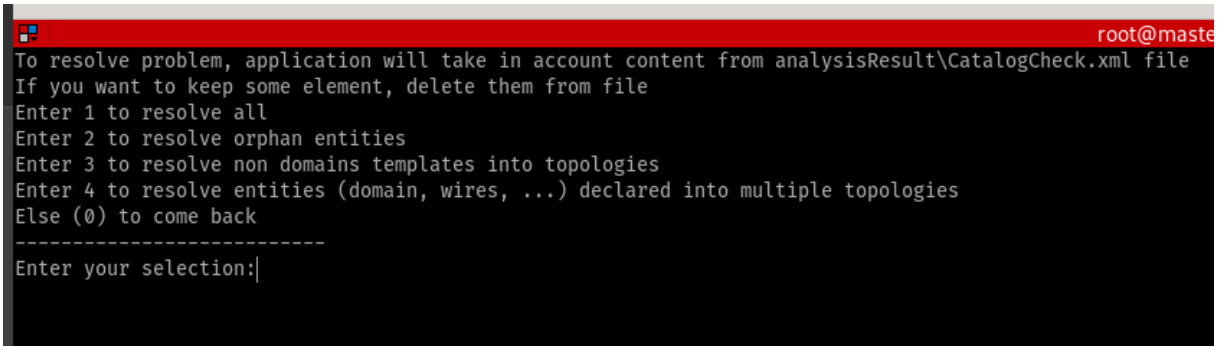
Add missing key into hynesim-master.ini file, to ensure the good system behaviour.

Replace bad values, by minimum Diateam requested values, to ensure system behaviour.

Resolve Hynesim node configuration file

Do nothing, because nothing is checked on hynesim-node service configuration.

Catalog resolve explanations



Resolve orphan entities

Do nothing for instant.

Resolve non domains entities template present into topologies definition

The resolve non domains entities template, will take element to delete from **analyseResult/CatalogCheck.xml** file.

The resolution will replace into to corresponding topology xml definition file, the isTemplate tag present for topology templates entities (or wires) having an inPlaceUUID property, to set this value to "0" in any case. Resolve duplicates uuids into topologies definition

The resolve duplicated uuids into topologies, will take element to delete from **analyseResult/CatalogCheck.xml** file.

Multiple case can be found:

- The duplicated uuid is referenced only by wires shared by multiple topologies.
- The duplicated uuid is referenced by templates entities shared by multiple topologies.
- The duplicated uuid is referenced by single catalog entity with physical filestorage shared by multiple topologies.
- The duplicated uuid is referenced by single catalog domain with physical filestorage shared by multiple topologies.
- The duplicated uuid is referenced by different kind of entities/wires and shared by multiple topologies.

For wire and template entity case, the resolver will replace the UUID associated to entity, with a new one not already used.

For non-domain entities with physical storage, we will replace entity registration by a topology template registration in the topology xml file by extracting the xml definition of entity from its proper xml file. The UUID value now stored in inPlaceUUID property is set to a new unexisting value.

For domain with physical storage, we must inform the user, to let him make the correction, into listed topologies found by analyser.

For mixed case (never seen), the informations are given and an administrator will have to resolve the problems.