reconurge / **flowsint** Public

Code | Issues 27 | Pull requests 5 | Actions | Projects | Security and quali

Commit b52cbbb

dextmorgn committed on Nov 17, 2025

```
feat(app): add key requirement for org_to_asn
```

main · v1.2.8 ••• v1.0.0

1 parent 658d26f commit b52cbbb

1 file changed +74 -65 lines changed

Top

Filter files…

flowsint-transforms/src/flowsint_transforms/organization

to_asn.py

1 file changed +74 -65 lines changed

Search within code

.../flowsint_transforms/organization/to_asn.py

```
@@ -1,10 +1,11 @@
- import json
- import subprocess
- from typing import List, Dict, Any, Union
+ import os
+ from typing import List, Dict, Any, Union, Optional
  from flowsint_core.core.transform_base import Transform
+ from flowsint_core.core.graph_db import Neo4jConnection
  from flowsint_types.organization import Organization
  from flowsint_types.asn import ASN
  from flowsint_core.core.logger import Logger
+ from tools.network.asnmap import AsnmapTool
```

```
 9  10
10  11  class OrgToAsnTransform(Transform):
    @@ -14,6 +15,39 @@ class OrgToAsnTransform(Transform):
14  15      InputType = List[Organization]
15  16      OutputType = List[ASN]
16  17
    18  +   def __init__(
    19  +       self,
    20  +       sketch_id: Optional[str] = None,
    21  +       scan_id: Optional[str] = None,
    22  +       neo4j_conn: Optional[Neo4jConnection] = None,
    23  +       vault=None,
    24  +       params: Optional[Dict[str, Any]] = None,
    25  +   ):
    26  +       super().__init__(
    27  +           sketch_id=sketch_id,
    28  +           scan_id=scan_id,
    29  +           neo4j_conn=neo4j_conn,
    30  +           params_schema=self.get_params_schema(),
    31  +           vault=vault,
    32  +           params=params,
    33  +       )
    34  +
    35  +   @classmethod
    36  +   def required_params(cls) -> bool:
    37  +       return True
    38  +
    39  +   @classmethod
    40  +   def get_params_schema(cls) -> List[Dict[str, Any]]:
    41  +       """Declare required parameters for this transform"""
    42  +       return [
    43  +           {
    44  +               "name": "PDCP_API_KEY",
    45  +               "type": "vaultSecret",
    46  +               "description": "The ProjectDiscovery Cloud Platform API key for
        asnmap.",
    47  +               "required": True,
    48  +           },
    49  +       ]
    50  +
```

```
17  51          @classmethod
18  52          def name(cls) -> str:
19  53              return "org_to_asn"
        @@ -42,74 +76,49 @@ def preprocess(self, data: Union[List[str], List[dict],
        InputType]) -> InputType
42  76
43  77          async def scan(self, data: InputType) -> OutputType:
44  78              """Find ASN information for organizations using asnmap."""
45      -           asns: OutputType = []
    79  +           results: OutputType = []
    80  +           asnmap = AsnmapTool()
    81  +
    82  +           # Retrieve API key from vault or environment
    83  +           api_key = self.get_secret("PDCP_API_KEY", os.getenv("PDCP_API_KEY"))
46  84
47  85              for org in data:
48      -               asn_data = self.__get_asn_from_asnmap(org.name)
49      -               if asn_data:
50      -                   asns.append(
51      -                       ASN(
52      -                           number=int(asn_data["as_number"].lstrip("AS")),
53      -                           name=asn_data["as_name"],
54      -                           country=asn_data["as_country"],
55      -                           cidrs=[],
56      -                       )
57      -                   )
58      -               else:
59      -                   Logger.info(
60      -                       self.sketch_id, {"message": f"No ASN found for org
        {org.name}."}
61      -                   )
62      -           return asns
63      -
64      -       def __get_asn_from_asnmap(self, name: str) -> Dict[str, Any]:
65      -           try:
66      -               # Properly run the shell pipeline using shell=True
67      -               command = f"echo {name} | asnmap -silent -json | jq -s"
68      -               result = subprocess.run(
69      -                   command, shell=True, capture_output=True, text=True, timeout=60
70      -               )
```

```
71        -             if not result.stdout.strip():
72        -                 return None
73    86                  try:
74        -                 # Parse the JSON array
75        -                 data_array = json.loads(result.stdout)
76        -                 if not data_array:
77        -                     return None
78        -
79        -                 combined_data = {
80        -                     "as_range": [],
81        -                     "as_name": None,
82        -                     "as_country": None,
83        -                     "as_number": None,
84        -                 }
85        -
86        -                 for data in data_array:
87        -                     if "as_range" in data:
88        -                         combined_data["as_range"].extend(data["as_range"])
89        -                     if data.get("as_name") and not combined_data["as_name"]:
90        -                         combined_data["as_name"] = data["as_name"]
91        -                     if data.get("as_country") and not
          combined_data["as_country"]:
92        -                         combined_data["as_country"] = data["as_country"]
93        -                     if data.get("as_number") and not
          combined_data["as_number"]:
94        -                         combined_data["as_number"] = data["as_number"]
95        -
96        -                 return combined_data if combined_data["as_number"] else None
97        -
98        -             except json.JSONDecodeError as e:
      87  +                 # Use asnmap tool to get ASN info, passing the API key
      88  +                 asn_data = asnmap.launch(org.name, type="org", api_key=api_key)
      89  +                 if asn_data and "as_number" in asn_data:
      90  +                     # Parse ASN number from string like "AS16276" to integer
          16276
      91  +                     asn_string = asn_data["as_number"]
      92  +                     asn_number = int(asn_string.replace("AS", "").replace("as",
          ""))
      93  +                     # Create ASN object with correct field mapping
      94  +                     asn = ASN(
```

```diff
+                        number=asn_number,
+                        name=asn_data.get("as_name", ""),
+                        country=asn_data.get("as_country", ""),
+                        description=asn_data.get("as_name", ""),
+                    )
+                    results.append(asn)
+                    Logger.info(
+                        self.sketch_id,
+                        {
+                            "message": f"[ASNMAP] Found AS{asn.number} ({asn.name}) for organization {org.name}"
+                        },
+                    )
+                else:
+                    Logger.warn(
+                        self.sketch_id,
+                        {
+                            "message": f"[ASNMAP] No ASN data or missing 'as_number' field for organization {org.name}. Data keys: {list(asn_data.keys()) if asn_data else 'None'}"
+                        },
+                    )
+            except Exception as e:
                 Logger.error(
                     self.sketch_id,
-                    {
-                        "message": f"An error occurred while parsing the JSON output from asnmap: {str(e)}"
-                    },
+                    {"message": f"Error getting ASN for organization {org.name}: {e}"},
                 )
-                return None
-
-        except Exception as e:
-            Logger.error(
-                self.sketch_id,
-                {"message": f"An error occurred while running asnmap: {str(e)}"},
-            )
```

```
112      -              return None
         119 +                  continue
         120 +
         121 +          return results
113      122
114      123          def postprocess(self, results: OutputType, original_input: InputType) ->
             OutputType:
115      124              # Create Neo4j relationships between organizations and their
             corresponding ASNs
```

Comments 0

Please sign in to comment.