Appendix２:

3.2.5通信確立ルートへの限定と攻撃パスの最適化

１：「資産とMITREの攻撃テクニックの組み合わせリストを作成し、それらの全組み合わせの攻撃パスを自動生成する。」

# 必要ライブラリインストール

!pip install -q pandas openpyxl networkx matplotlib

from google.colab import files

import pandas as pd

import itertools

import networkx as nx

import matplotlib.pyplot as plt

# 1. ノードファイルのアップロード

print("nodes\_assets\_techniques\_20250802.xlsxをアップロードしてください")

uploaded = files.upload()

nodes\_file = list(uploaded.keys())[0]

xls = pd.ExcelFile(nodes\_file)

# 各ステップのノードリスト(node\_id, asset\_name, technique\_id)

steps = {}

for sheet in xls.sheet\_names:

df = xls.parse(sheet)

df = df[["node\_id", "asset\_name", "technique\_id"]]

steps[sheet] = df

# 2. エッジファイルのアップロード（今回はパス列挙・可視化用途のみなら未使用でもOK）

print("edge\_assets\_techniques\_20250802.xlsxをアップロードしてください")

uploaded = files.upload()

edges\_file = list(uploaded.keys())[0]

# 3. 各stepのノードリストを列挙

step1 = steps["step1"].itertuples(index=False)

step2 = steps["step2"].itertuples(index=False)

step3 = steps["step3"].itertuples(index=False)

step4 = steps["step4"].itertuples(index=False)

# 4. 全組合せ列挙（25088通り）

combis = itertools.product(step1, step2, step3, step4)

# 5. アタックパスを構造化

attack\_paths = []

for a, b, c, d in combis:

attack\_paths.append({

"step1\_node\_id": a.node\_id,

"step1\_asset": a.asset\_name,

"step1\_tech": a.technique\_id,

"step2\_node\_id": b.node\_id,

"step2\_asset": b.asset\_name,

"step2\_tech": b.technique\_id,

"step3\_node\_id": c.node\_id,

"step3\_asset": c.asset\_name,

"step3\_tech": c.technique\_id,

"step4\_node\_id": d.node\_id,

"step4\_asset": d.asset\_name,

"step4\_tech": d.technique\_id,

})

# 6. DataFrame化してCSV保存＆ダウンロード

attack\_paths\_df = pd.DataFrame(attack\_paths)

attack\_paths\_df.to\_csv("all\_attack\_paths\_25088.csv", index=False, encoding="utf-8-sig")

files.download("all\_attack\_paths\_25088.csv")

print("all\_attack\_paths\_25088.csv をダウンロード可能")

# 7. サンプル可視化（step1→step2→step3→step4 でグラフ化、最初の数個のみ描画）

def draw\_attack\_graph(paths\_df, num\_paths=10):

G = nx.DiGraph()

for i, row in paths\_df.head(num\_paths).iterrows():

G.add\_edge(row["step1\_node\_id"], row["step2\_node\_id"])

G.add\_edge(row["step2\_node\_id"], row["step3\_node\_id"])

G.add\_edge(row["step3\_node\_id"], row["step4\_node\_id"])

plt.figure(figsize=(12, 8))

pos = nx.spring\_layout(G, seed=42)

nx.draw(G, pos, with\_labels=True, node\_size=700, font\_size=8, font\_weight='bold', arrows=True)

plt.title(f"Sample Attack Paths (First {num\_paths} Paths)")

plt.show()

draw\_attack\_graph(attack\_paths\_df, num\_paths=6)

２：「25,088の攻撃パスを通信経路を特定して最適化された攻撃パス数を抽出する。」

#必要なパッケージのインストール  
import pandas as pd

import networkx as nx

import matplotlib.pyplot as plt

from google.colab import files

# ノードファイルを手動でアップロード

uploaded\_nodes = files.upload()  # ここで1ファイルだけ選べます

# 例: nodes.xlsx もしくは nodes.csv

import pandas as pd

# 拡張子に応じて読み込み

if list(uploaded\_nodes.keys())[0].endswith('.xlsx'):

    nodes = pd.read\_excel(list(uploaded\_nodes.keys())[0])

else:

    nodes = pd.read\_csv(list(uploaded\_nodes.keys())[0])

print(nodes.head())

# エッジファイルを手動でアップロード

uploaded\_edges = files.upload()  # ここで1ファイルだけ選べます

# 例: edges.xlsx もしくは edges.csv

if list(uploaded\_edges.keys())[0].endswith('.xlsx'):

    # シートごとに読み込む場合

    sheet\_names = pd.ExcelFile(list(uploaded\_edges.keys())[0]).sheet\_names

    # 例：複数シート合体

    edge\_df\_list = []

    for sn in sheet\_names:

        edge\_df\_list.append(pd.read\_excel(list(uploaded\_edges.keys())[0], sheet\_name=sn))

    edges = pd.concat(edge\_df\_list, ignore\_index=True)

else:

    edges = pd.read\_csv(list(uploaded\_edges.keys())[0])

print(edges.head())

# edges.csv, nodes.csvをアップロード後、データの読み込み

edges\_all = edges  # 複数シートマージ想定

print(edges\_all.head())

#通常時通信ルートのみ抽出する（communication\_allowed＝１）

edges = edges\_all[edges\_all['communication\_allowed'] == 1]

#有向グラフを作成

G = nx.DiGraph()

for \_, row in edges.iterrows():

    G.add\_edge(row['source\_node\_id'], row['target\_node\_id'], step\_from=row['step\_from'], step\_to=row['step\_to'])

＃stepごとに開始・終了ノードリストを作成

step1\_nodes = set(edges[edges['step\_from']==1]['source\_node\_id'])

step4\_nodes = set(edges[edges['step\_to']==4]['target\_node\_id'])

#全step1→step4のFull4 step passのみを抽出

paths = []

for start in step1\_nodes:

for end in step4\_nodes:

# 全4ステップパスだけ（longest pathでフィルタ）

for path in nx.all\_simple\_paths(G, source=start, target=end, cutoff=3):

# pathは [step1, step2, step3, step4] の長さ4

if len(path) == 4:

# 途中ノード移動もstep系列に沿うものだけ

valid = True

for i in range(3):

e = G.edges[path[i], path[i+1]]

if e['step\_from'] != i+1 or e['step\_to'] != i+2:

valid = False

break

if valid:

paths.append(path)

＃DataFrame化して保存

import pandas as pd

result = pd.DataFrame(paths, columns=["step1", "step2", "step3", "step4"])

result.to\_csv("attack\_paths\_step4only.csv", index=False)

＃経路グラフをNetworkXで可視化（取り急ぎ1パス分のみのグラフ描画）

if len(paths) > 0:

subG = nx.DiGraph()

for i in range(3):

subG.add\_edge(paths[0][i], paths[0][i+1])

nx.draw(subG, with\_labels=True, node\_color='lightblue', arrows=True)

plt.show()

＃保存されたcsvファイルをＤＬする

from google.colab import files

files.download('attack\_paths\_step4only.csv')