# 1. Morpheus Advanced Filters: 복합 필터링 및 조건부 분석

## 프로젝트 개요

NVIDIA Morpheus 파이프라인에서 데이터 메시지를 복합적인 조건으로 필터링하고, 특정 조건에 부합하는 메시지에 대해 \*\*보안 분석 로직(예: 실패 로그인 횟수 계산)\*\*을 적용하는 방법을 실습했습니다.

## ⚙️ 주요 기술 및 역량

* **기술:** Morpheus FilterStage, SinglePortStage 커스텀 스테이지, cudf 데이터 처리.
* **역량:**
  1. **복합 필터링:** 단일 파이프라인에서 여러 개의 독립적인 FilterStage를 병렬로 배치하여, 각기 다른 조건(패킷 크기, 특정 IP 접근 등)에 맞는 메시지를 동시에 분리하는 능력을 입증했습니다.
  2. **조건부 분석 커스텀 스테이지:** Exercise-failed-logins에서처럼, 들어온 메시지의 페이로드(cuDF)를 분석하여 **실패 로그인 횟수**와 같은 보안 지표를 계산하고, 이 결과를 다운스트림으로 전달하는 **커스텀 분석 로직**을 구현했습니다.

## 🛠️ 구현 내용 (Exercise-multiple-filters.ipynb 중심)

1. **다중 필터 병렬 연결:** DeserializeStage의 출력을 두 개 이상의 FilterStage에 동시에 연결하여 메시지 복제 없이 효율적으로 데이터 스트림을 분리했습니다.
2. **다단계 전처리:** Exercise-multi-stage-pcap에서처럼, 복잡한 PCAP 데이터 전처리 과정을 **재사용 가능한 여러 커스텀 스테이지**로 분리하여 파이프라인에 순차적으로 연결했습니다.
3. **결과 확인:** 각 필터 및 분석 단계의 결과를 InMemorySinkStage로 받아, 계산된 값이 정확한지(예: failed\_count가 올바른지) 확인하여 파이프라인의 정확성을 검증했습니다.

## 📌 주요 코드 로직 (실패 로그인 분석 로직 추상화)

# 실패 로그인 분석 로직 (커스텀 스테이지 내부)  
class FailedLoginAnalysisStage(SinglePortStage):  
  
 def on\_data(self, message: ControlMessage) -> ControlMessage:  
 with message.payload().mutable\_dataframe() as df:  
 # 'failure' 상태의 레코드만 필터링하고 그룹화하여 집계 (보안 분석 로직)  
 failed\_logins = df[df['status'] == 'failure']  
 if not failed\_logins.empty:  
 # 사용자별 실패 횟수를 계산하여 임시 컬럼에 저장  
 df = df.merge(  
 failed\_logins.groupby('user').size().reset\_index(name='failed\_count'),   
 on='user',   
 how='left'  
 ).fillna({'failed\_count': 0})  
   
 # 추가적인 분석 결과 태깅 (메타데이터 활용)  
 message.set\_metadata("high\_risk\_users", df[df['failed\_count'] > THRESHOLD]['user'].unique().to\_list())  
  
 return message  
  
# 파이프라인에 적용 시:  
# pipeline.add\_edge(deserialize, FailureAnalysisStage(config))