# Final comparison and conclusion

| **Criterion** | **CyNER** | **SecureBERT-NER** |
| --- | --- | --- |
| #Entities in MalwareTextDB | 4899 (<https://huggingface.co/datasets/naorm/malware-text-db-cyner-512>) | 6674 (<https://huggingface.co/datasets/naorm/malware-text-db-securebert-ner-512>) |
| #Entities in DNRTI | 9380 (<https://huggingface.co/datasets/naorm/dnrti-cyner-ner-512>) | 12157 (<https://huggingface.co/datasets/naorm/dnrti-securebert-ner-512>) |
| #Classes | 5 (Malware, Vulnerability, System, Organization, Indicator) | 20 (list of classes in <https://huggingface.co/CyberPeace-Institute/SecureBERT-NER>) |
| Reproducibility | We have both paper and train code | We have only the paper (data used for pre-training is unknown and data used for finetuning is not completely clear) |
| Performance | Same: 2s for 512 model (can contain 10-15 sentences in each) | |
| Advantages | - Better Detecting File paths, Registry paths, Api names, memory addresses, mail addresses (Indicator class) | - Can detect Location and Time  - Better detecting tools, MD5, protocol and more diverse set of classes |
| Precision/Recall | Info in section “Evaluation of CyNER/SecureBERT-NER on DNRTI” | |

**Conclusion:**

1. Overall SecureBERT-NER provides better results and should be first option
   1. The 20 classes are a bit too much so I recommend merging some classes (OS+TOOL, APT+IDTY) and maybe more
   2. And maybe remove some that are not needed or are bad (like VULNAME)
2. However, CyNER provides advantages in finding many indicators that SecureBERT-NER was not trained to find so mixing them together is preferred if possible (Maybe only take the Indicator class from it but maybe Vulnerability and others can also be useful..)

Testing code available in <https://github.com/naormatania/cyber-ai/tree/main/NER>

# CyNER (Model)

Trained on self-annotated dataset

**Classes:** classes-malware, indicator, system, organization, and vulnerability

1. Finetuned BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019) and XLM-RoBERTa (Conneau et al., 2019) models
2. Flair NER (Based on document-level XLM-R embeddings and FLERT. Classes: PER, LOC, ORG, MISC)
3. Cybersecurity entity extraction using Heuristics
4. Most are based on the paper "Cybersecurity Named Entity Recognition Using Multi-Modal Ensemble Learning"

self.patterns = {

'Filename': [r'[A-Za-z0-9-\_\.]+\.(txt|php|exe|dll|bat|sys|htm|html|js|jar|jpg|png|vb|scr|pif|chm|zip|rar|cab|pds|doc|docx|ppt|pptx|xls|xlsx|swf|gif)',

'^[a-zA-Z0-9](?:[a-zA-Z0-9 .\_-]\*[a-zA-Z0-9])?\.[a-zA-Z0-9\_-]+$',

"^([A-Za-z]{1}[A-Za-z\\d\_]\*\\.)+[A-Za-z][A-Za-z\\d\_]\*$"],

'Filepath': [r'[a-zA-Z]:\\([0-9a-zA-Z]+)', r'(\/[^\s\n]+)+'],

'Email': [r'[a-z][\_a-z0-9-.]+@[a-z0-9-]+[a-z]+'],

'SHA256': [r'[a-f0-9]{64}|[A-F0-9]{64}'],

'SHA1': [r'[a-f0-9]{40}|[A-F0-9]{40}'],

'Hash': ['r"([a-fA-F\d]{32})"'],

'Domain': ['"^(((([A-Za-z0-9]+){1,63}\.)|(([A-Za-z0-9]+(\-)+[A-Za-z0-9]+){1,63}\.))+){1,255}$"',

'(//|\s+|^)(\w\.|\w[A-Za-z0-9-]{0,61}\w\.){1,3}[A-Za-z]{2,6}'],

'CVE': [r'CVE—[0-9]{4}—[0-9]{4,6}'],

'URL': [r'(https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|www\.[a-zA-Z0-9][a-zA-Z0-9-]+[a-zA-Z0-9]\.[^\s]{2,}|https?:\/\/(?:www\.|(?!www))[a-zA-Z0-9]+\.[^\s]{2,}|www\.[a-zA-Z0-9]+\.[^\s]{2,})', r"https?:[a-zA-Z0-9\_.+-/#~]+ "],

'IPv4': [r'^((25[0-5]|(2[0-4]|1\d|[1-9]|)\d)(\.(?!$)|$)){4}$'],

'IPAddress': ['r"^\d{1,3}\[.]\d{1,3}\.\d{1,3}\.\d{1,3}$"'],

'Protocol': ['HTTP', 'SMS', 'HTTPS', 'AES'],

'IPv6': [r'(([0-9a-fA-F]{1,4}:){7,7}[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,7}:|([0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|([0-9a-fA-F]{1,4}:){1,5}(:[0-9a-fA-F]{1,4}){1,2}|([0-9a-fA-F]{1,4}:){1,4}(:[0-9a-fA-F]{1,4}){1,3}|([0-9a-fA-F]{1,4}:){1,3}(:[0-9a-fA-F]{1,4}){1,4}|([0-9a-fA-F]{1,4}:){1,2}(:[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:((:[0-9a-fA-F]{1,4}){1,6})|:((:[0-9a-fA-F]{1,4}){1,7}|:)|fe80:(:[0-9a-fA-F]{0,4}){0,4}%[0-9a-zA-Z]{1,}|::(ffff(:0{1,4}){0,1}:){0,1}((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])|([0-9a-fA-F]{1,4}:){1,4}:((25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9])\.){3,3}(25[0-5]|(2[0-4]|1{0,1}[0-9]){0,1}[0-9]))'],

}

## Pros

1. Code to train model ourselves (and also dataset they used)
   1. There is also model available (needs to be validated)
2. Worked well on SecureBERT-NER specific example (but doesn’t mean much of course)

## Cons

1. Trained only on Threat Intelligence reports regarding Android (They annotated 60 reports themselves using BRAT for train)
2. Achieves ~75% F1-score
3. ~3.84s on CPU (~3s for base model) (on V100 GPU: ~1.17s)
4. **Used 512 model that seemed to be finetuned on 128 sequences only (supported both by paper & code)**

# SecureBERT-NER (Model)

<https://huggingface.co/CyberPeace-Institute/SecureBERT-NER>

This model has been finetuned with SecureBERT (<https://arxiv.org/abs/2204.02685>) on the **APTNER** dataset (in paper it claimed to use **MalwareTextDB**)

They pre-trained the model with large cyber security corpus

They developed a customized tokenizer as well as a method to alter pre-trained weights

## Pros

1. Model available (needs to be validated)
2. Pre-trained weights are available (see in <https://github.com/ehsanaghaei/SecureBERT>)
3. Much more classes (10-15)
4. Achieved ~86% F1-score (on **MalwareTextDB**)

## Cons

1. No code to finetune or to pre-train it ourselves and no data for pre-training (just the weights)
2. The paper said they finetuned with MalwareTextDB but in huggingface they said they finertuned on APTNER
3. ~2.78s for model (on V100 GPU: ~0.8s)
4. On CyNER specific example:
   1. Wasn’t able to detect OS android
   2. Wasn’t able to detect LOC UK

# Multi-features based Semantic Augmentation Networks for Named Entity Recognition in Threat Intelligence (Model)

Trained on **MalwareTextDB** and **DNRTI**

Complex architecture involving BERT, BiLSTM, unique features and more

Code: <https://github.com/liupeip-cs/ner4cti?tab=readme-ov-file>

## Pros

1. Claim to achieve very good results (86% F1-score)

## Cons

1. Complex architecture to use
2. Probably slower because of the use of BiLSTM

# (Irrelevant) Open-CyKG: An Open Cyber Threat Intelligence Knowledge Graph

Bi-GRU layer which outputs a tensor that is later passed to a TDD layer. Finally, a CRF prediction layer labels each word in our dataset by generating likelihood distributions over every available tag.

<https://github.com/IS5882/Open-CyKG/blob/main/Open_Cy_KG_NER.ipynb>

# (Irrelevant) Study of Word Embeddings for Enhanced Cyber Security Named Entity Recognition

Trained on **Auto-labelled-corpus**

<https://www.sciencedirect.com/science/article/pii/S1877050923000273>

BiLSTM, CRF on embeddings or position-wise FFN

## Pros

1. Get 88-97% F1-score (on different dataset)
2. Can also use context-independent embeddings and thus become faster than BERT-basd

## Cons

1. **No code** or model available

# MalwareTextDB (Dataset)

<https://github.com/statnlp-research/statnlp-datasets/blob/master/dataset/MalwareTextDB-1.0.zip>

# CTI-reports-dataset

<https://github.com/nlpai-lab/CTI-reports-dataset>

# APTNER (Dataset)

Dataset for NER in Advanced Persistent Threats

<https://ieeexplore.ieee.org/document/9776031>

# Auto-labelled-corpus (Dataset)

<https://arxiv.org/abs/1308.4941>

<https://github.com/stucco/auto-labeled-corpus>

# DNRTI (Dataset)

Dataset for NER in Threat Intelligence

<https://ieeexplore.ieee.org/document/9343158>

<https://github.com/SCreaMxp/DNRTI-A-Large-scale-Dataset-for-Named-Entity-Recognition-in-Threat-Intelligence>

# Analysis on MalwareTextDB

MalwareTextDB contains about 1MB of text.

1. CyNER (No flair, No heuristics)
   1. <https://huggingface.co/datasets/naorm/malware-text-db-cyner>
   2. 5742 found in **total**
   3. Indicator contains URL, ip, paths, api name, emails, ports
      1. Problems: sometimes part of api name returns (Find instead of FindFile) because of the Upper case probably. Long regex path is cut to two, missed user name and password once
      2. 1598 found
   4. Organization
      1. 1571 found
   5. Malware
      1. Problems: Detecting McAfee, CrowdStrike as Malware, UpperCase is making it difficult, some entities (websites ?) are identified as Malwares
      2. 1332 found
   6. System
      1. Problems: Windows Sleep detected, apis/CVE detected, 7-zip
      2. 1090 Found
   7. Vulnurability
      1. Find: CVEs, vulnerabilities (indentified as such in text)
      2. Problems: Doesn’t catch the full exploit (MS and not MS2048)
      3. 151 Found
   8. **Location is missing**
2. CyNER (With huristics)
   1. From 5742 to 5976
   2. Indicator is split to more granular entities like Domains, Filename, Filepath. SHA
   3. Problems:
      1. Domain contains ips, ntuser.dat, asp.net, trojan.bat
      2. Paths not always good
      3. In general doesn’t add much
3. ScureBERT-NER
   1. <https://huggingface.co/datasets/naorm/malware-text-db-securebert-ner>
   2. 6079 found **in total**
   3. Tool - 1334
      1. Relatively good
   4. IDTY (authentication identity) - good, microsoft shows (ok) but also Defense, NGO, non-profit, military, citizen - 881
   5. LOC - 884 - ok
   6. MAL - 563
   7. TIME - 424
   8. FILE - 385 found good files
   9. ACT - 385 Found some things like DDos but a mess overall
   10. APT - 262 found ISIS, Cyber arabs, APT3 but also “actors”,”threat” so pretty messy
   11. SECTEAM - good overall
   12. OS - 201
   13. PROT - 138 - good with few small bugs
   14. VULNAME - bad
   15. DOM - 91 - found good domains
   16. ENCR - 91 - good
   17. IP - 44 - good
   18. URL - 44 - good - sometimes part of url
   19. MD5 - 42 - good

# Analysis on DNRTI

DNRTI contains about 1MB of text.

1. CyNER (No flair, No heuristics)
   1. **Part of words problem**
      1. **Problem with CyNER for 512 blocks: positions are incorrect but the txt is (problem with library)**
         1. **“airport .” is changed to “airport.”**
      2. **Another problem with CyNER for 512 blocks: It is more likely to break words into parts but not to add much value**
   2. <https://huggingface.co/datasets/naorm/dnrti-cyner>
   3. 13804 found
   4. Malware - 6013
      1. Poison I instad of Poison-Ivy
      2. Sometimes: B and Hack instead of B-HACK
      3. Overall good
   5. Organization - 3244
      1. Hong Kong, China not detect. Not all locations
   6. System - 1805
      1. Caught “Malaysia Airlines Flight MH370”
      2. BUBBLEWRAP identified as system instead of Malware ?
      3. B-Hack is both here and in Malware
   7. Vulnerability - 1408
      1. Caught Hack and Flash here
      2. Instead of CVE-2014-6332 it showed “CVE-2014” and “332” separately
   8. Indicator - 1334
      1. Uninformative “-” or “I-” (Maybe ignore cases of such small number of letters)
2. ScureBERT-NER
   1. **Prior words problem**
   2. <https://huggingface.co/datasets/naorm/dnrti-securebert-ner>
   3. 4587 found
   4. MAL
      1. 784 found
      2. Problems: detected “as”, “the”, “call”, “from” ,”to” (before real malware that is also detected
   5. APT (Threat participants)
      1. 700 found (very similar to MAL in intent)
      2. Found mainly B-HACK
      3. Also same problem
   6. LOC
      1. 459 found
      2. Also spot to/at and missed China at least once
   7. TIME
      1. 428 found
      2. Detect May but not the full May 2016
   8. TOOL
      1. 428 found (email, chrome, powershell, phishing)
      2. Same prior text repeats so I won’t continue to write about it
   9. SECTEAM
      1. 367 found
   10. ACT (Attack action) - similar to Tool, find spear phishing better
       1. 367
   11. FILE
       1. Really messed up. Find vulnerabilities and entities in addition to some files
   12. IDTY (Authentication identity) is confused with tools and locations and ACT. Not good
   13. VULID
       1. Find CVE relatively good
   14. VULNAME
       1. Find some nice catches but also lots of other things like exploit, vulnerability, zero-day
   15. OS
       1. Find 60. Relatively well

Problems with sentence tokenization:

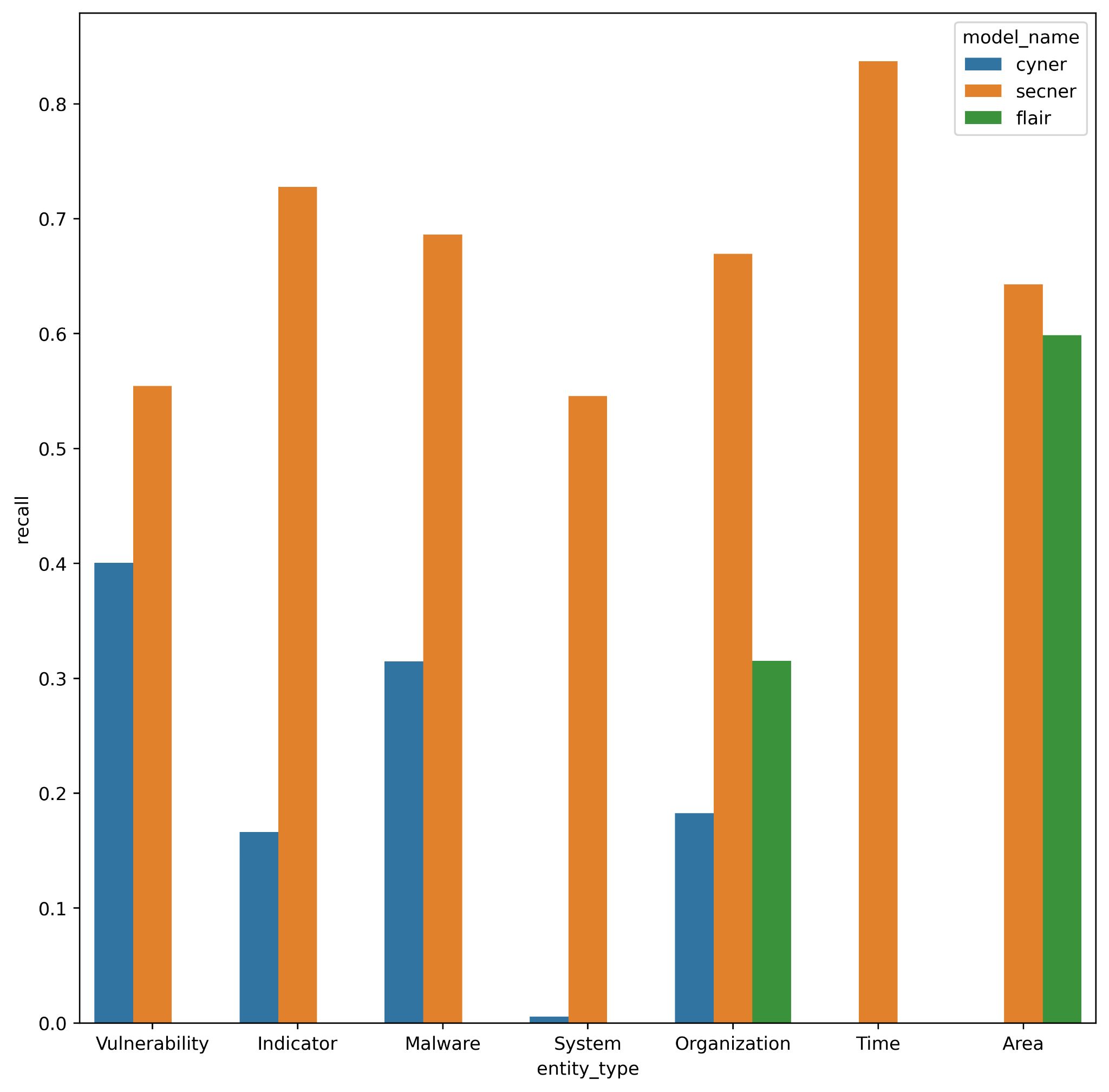
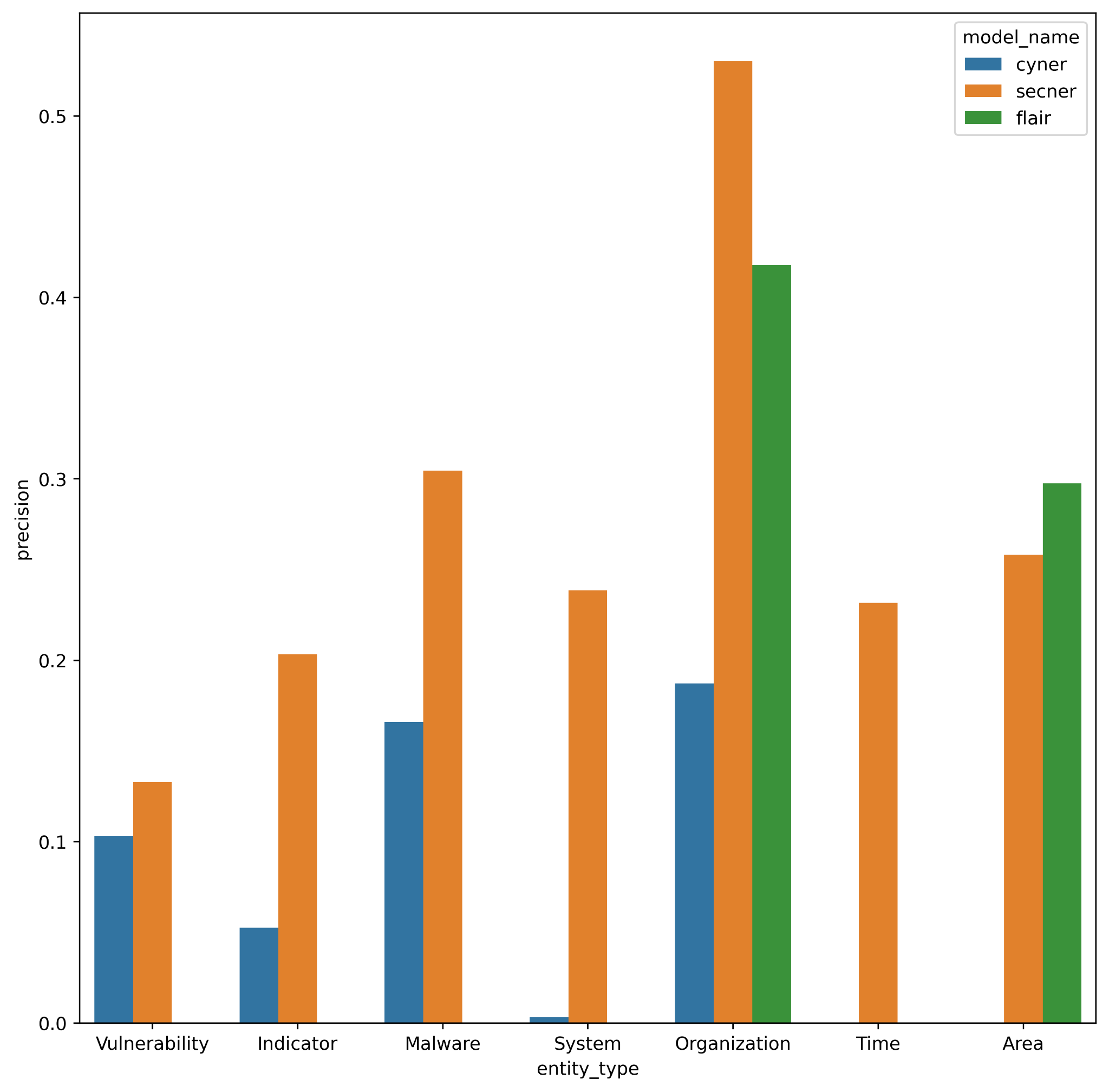
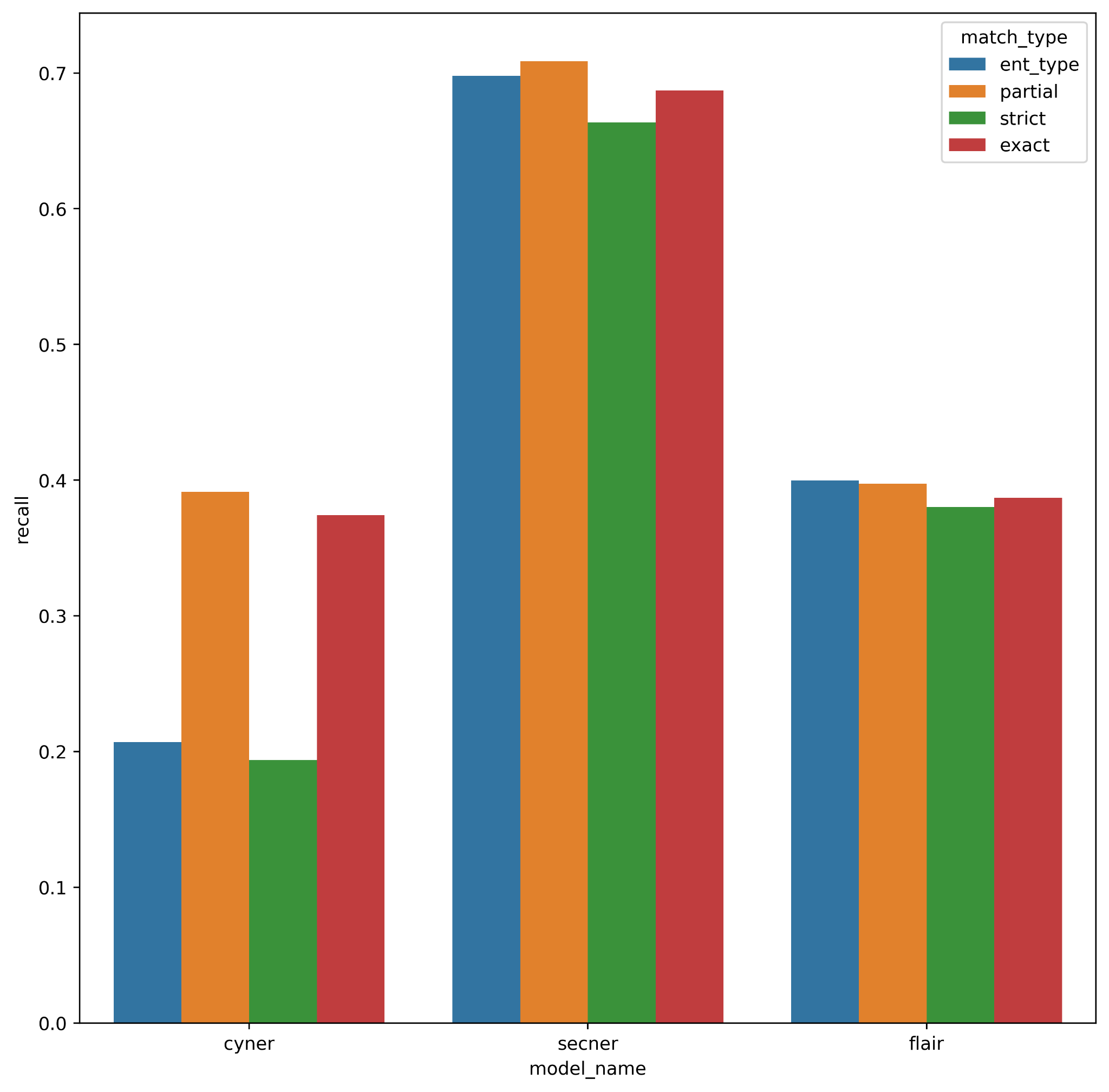
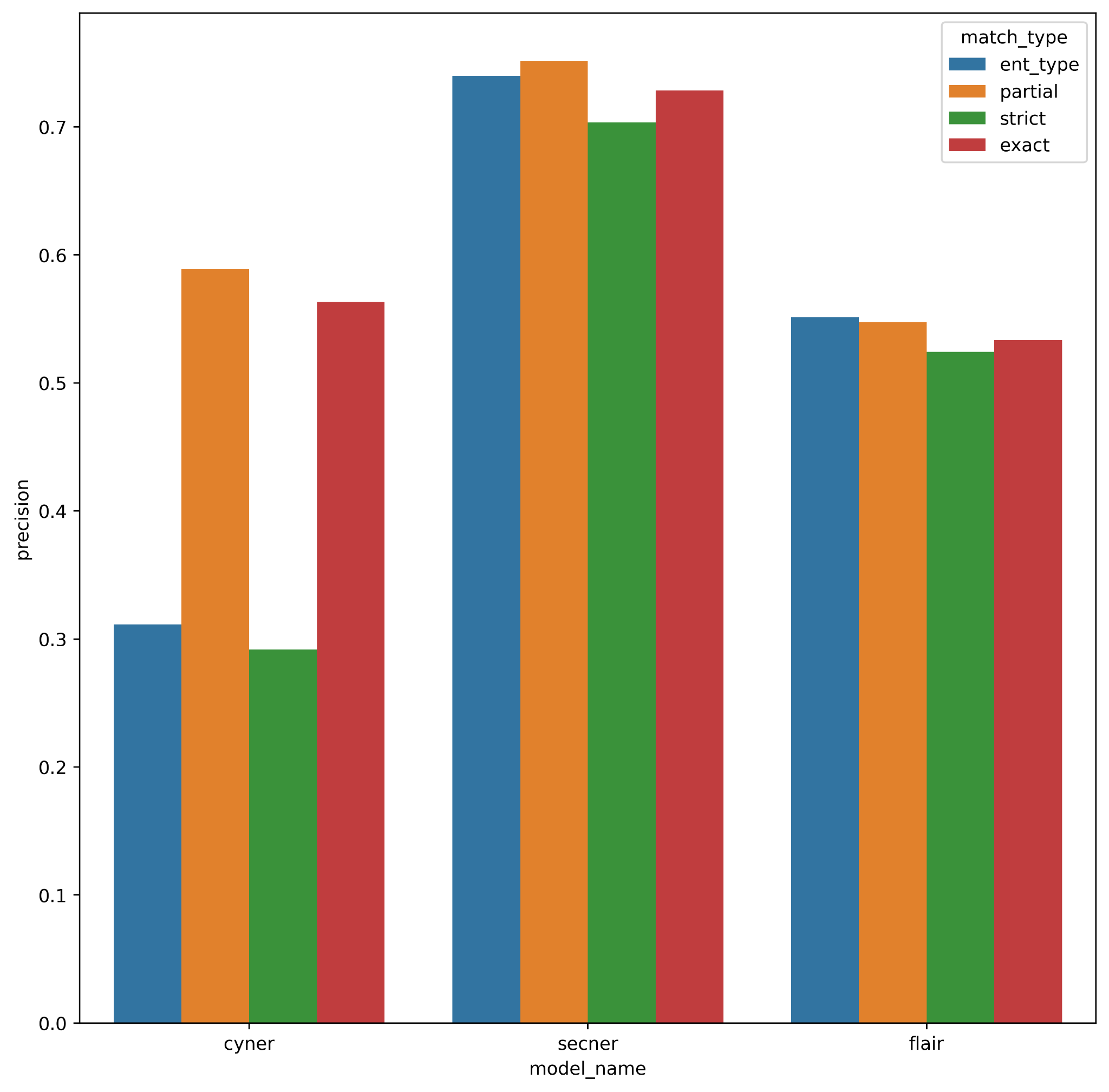
* Inc.

# Evaluation of CyNER/SecureBERT-NER on DNRTI

Match types:

* **Strict:** exact boundary surface string match and entity type;
* **Exact:** exact boundary match over the surface string, regardless of the type;
* **Partial:** partial boundary match over the surface string, regardless of the type;
* **Type:** some overlap between the system tagged entity and the gold annotation is required;

| **Flair** | **CyNER** | **DNRTI** | **SecureBERT-NER** | **SecureBERT-NER/Precision-Strict** | **SecureBERT-NER/Recall-Strict** |
| --- | --- | --- | --- | --- | --- |
| ORG | Organisation | HackOrg | APT | 0.6494508062631456 | 0.6586868926285849 |
| ORG | Organisation | SecTeam | SECTEAM | 0.4283476898981989 | 0.8237951807228916 |
| ORG | Organisation | Idus, Org | IDTY | 0.5314456035767511 | 0.6029759891782212 |
|  | System | OffAct, Way | ACT, OS, TOOL | 0.23863452515774147 | 0.5456899741028487 |
|  | Vulnerability | Exp | VULID, VULNAME | 0.13290113452188007 | 0.5544703230653644 |
|  | Malware | Tool | MAL | 0.30466724286949004 | 0.6862426995457496 |
|  | Indicator | SamFile | FILE | 0.4626579854461892 | 0.727710843373494 |
|  | Indicator |  | DOM, ENCR, IP, URL, MD5, PROT, EMAIL, SHA1, SHA2 | 0.2033327722605622 | 0.727710843373494 |
|  |  | Time | TIME | 0.23180770497201186 | 0.8370986920332937 |
| LOC |  | Area | LOC | 0.25816164817749604 | 0.6428571428571429 |
|  |  | Purp, Features |  |  |  |



## Conclusions

1. SecureBERT-NER is able to locate between 50-80% of potential entities
2. SecureBERT-NER precision is 10-40% lower than recall
   1. Potential followup task: check how much false positives they are and if increasing the threshold to decide if entity exists could reduce amount of false positives significantly with small reduction in true positives (create Precision-Recall curve in general and also maybe per entity)
3. SecureBERT-NER is pretty good at predicting correct entity type when predicting entity (difference between strict/exact in precision/recall)
   1. Potential followup task: check the difference for every separate category to see if there are specific categories where difference is more significant
4. SecureBERT-NER outperforms CyNER in all categories
5. SecureBERT-NER also outperforms Flair in Organization category and in Area category it gets a bit higher recall but a bit lesser precision
6. It seems like there is bug with ent\_type results (but it doesn’t seem to impact anything significant)

# Performance

**SecureBERT-NER**

Model 128, 1 Sentence: avg\_time=0.5906543620427449, std\_time=0.13092765748846194, num\_entities=295

Model 128, 2 Sentences: avg\_time=0.6094897222518921, std\_time=0.11033748872752366, num\_entities=312

Model 128, 3 Sentences: avg\_time=0.6108589887619018, std\_time=0.11140611035352632, num\_entities=309

Model 128, 4 Sentences: avg\_time=0.5969663715362549, std\_time=0.1029458611519204, num\_entities=287

Model 128, 5 Sentences: avg\_time=0.6145686825116475, std\_time=0.1055091065665659, num\_entities=264

Model 128, 6 Sentences: avg\_time=0.6032228469848633, std\_time=0.09889890412883531, num\_entities=219

Model 512, 1 Sentence: avg\_time=2.1313336078325906, std\_time=0.3637195315282539, num\_entities=295

Model 512, 2 Sentences: avg\_time=2.0913076559702555, std\_time=0.3064717528174025, num\_entities=314

Model 512, 3 Sentences: avg\_time=2.1035696935653685, std\_time=0.3193642882805152, num\_entities=321

Model 512, 4 Sentences: avg\_time=2.058611660003662, std\_time=0.2922045743686901, num\_entities=313

Model 512, 5 Sentences: avg\_time=2.0472505569458006, std\_time=0.25820835112355656, num\_entities=333

Model 512, 6 Sentences: avg\_time=2.0843345642089846, std\_time=0.30569542365940255, num\_entities=319

\_\_

Model 128, 1 Sentence: avg\_time=0.01709704065322876, std\_time=0.0035500451331206815, num\_entities=1336

Model 128, 2 Sentences: avg\_time=0.016378769874572752, std\_time=0.0017302134164607179, num\_entities=1341

Model 128, 3 Sentences: avg\_time=0.020086204934262945, std\_time=0.00461576420094593, num\_entities=1346

Model 128, 4 Sentences: avg\_time=0.018911154747009277, std\_time=0.002429060009023274, num\_entities=1289

Model 128, 5 Sentences: avg\_time=0.022586359977722167, std\_time=0.004390608501068001, num\_entities=1238

Model 128, 6 Sentences: avg\_time=0.02439622393625225, std\_time=0.010105136631550625, num\_entities=1087

Model 128, 7 Sentences: avg\_time=0.02206373714900517, std\_time=0.0021151779691154916, num\_entities=930

Model 128, 8 Sentences: avg\_time=0.02289754867553711, std\_time=0.0027030842613920084, num\_entities=802

Model 128, 9 Sentences: avg\_time=0.03252624613898141, std\_time=0.005764841151934105, num\_entities=782

Model 128, 10 Sentences: avg\_time=0.024935176372528078, std\_time=0.002526552395800655, num\_entities=673

Model 128, 12 Sentences: avg\_time=0.02623222271601359, std\_time=0.003032062418117382, num\_entities=550

Model 128, 14 Sentences: avg\_time=0.02901285555627611, std\_time=0.0036553890397946188, num\_entities=487

Model 128, 16 Sentences: avg\_time=0.03126231829325358, std\_time=0.004389408398486043, num\_entities=434

Model 512, 1 Sentence: avg\_time=0.018037638902664185, std\_time=0.0029551244064480744, num\_entities=1335

Model 512, 2 Sentences: avg\_time=0.01734056282043457, std\_time=0.0014615596935101398, num\_entities=1352

Model 512, 3 Sentences: avg\_time=0.021392384689011258, std\_time=0.004573374352605039, num\_entities=1362

Model 512, 4 Sentences: avg\_time=0.019885924339294432, std\_time=0.0019786729393927952, num\_entities=1351

Model 512, 5 Sentences: avg\_time=0.025916088819503785, std\_time=0.0059211854332266575, num\_entities=1391

Model 512, 6 Sentences: avg\_time=0.021531671821000332, std\_time=0.002170074285343397, num\_entities=1365

Model 512, 7 Sentences: avg\_time=0.02273122914187558, std\_time=0.002731429963028724, num\_entities=1389

Model 512, 8 Sentences: avg\_time=0.028010210037231444, std\_time=0.005480988455928698, num\_entities=1388

Model 512, 9 Sentences: avg\_time=0.029241555503436496, std\_time=0.007908342536252892, num\_entities=1393

Model 512, 10 Sentences: avg\_time=0.02612596035003662, std\_time=0.003697846455085012, num\_entities=1404

Model 512, 12 Sentences: avg\_time=0.028791626294453938, std\_time=0.003591653980675231, num\_entities=1409

Model 512, 14 Sentences: avg\_time=0.02972989281018575, std\_time=0.0037421174388604594, num\_entities=1389

Model 512, 16 Sentences: avg\_time=0.04058246763925704, std\_time=0.007494713427823622, num\_entities=1389

**CyNER**

Model 128, 1 Sentence: avg\_time=0.48232346773147583, std\_time=0.09209487557597755, num\_entities=194

Model 128, 2 Sentences: avg\_time=0.48292092164357503, std\_time=0.08677430456919742, num\_entities=186

Model 128, 3 Sentences: avg\_time=0.48719712495803835, std\_time=0.09000770358505383, num\_entities=205

Model 128, 4 Sentences: avg\_time=0.4779359785715739, std\_time=0.07943797838916397, num\_entities=176

Model 128, 5 Sentences: avg\_time=0.4836884379386902, std\_time=0.08697618166571494, num\_entities=143

Model 128, 6 Sentences: avg\_time=0.4854114818572998, std\_time=0.08632611514081301, num\_entities=130

Model 512, 1 Sentence: avg\_time=1.9659549228350321, std\_time=0.2947608299694911, num\_entities=194

Model 512, 2 Sentences: avg\_time=1.943931099573771, std\_time=0.2718955234245042, num\_entities=191

Model 512, 3 Sentences: avg\_time=1.9673393058776856, std\_time=0.27825161571877927, num\_entities=223

Model 512, 4 Sentences: avg\_time=1.967146759033203, std\_time=0.2863709966176826, num\_entities=210

Model 512, 5 Sentences: avg\_time=1.950994594891866, std\_time=0.26968293406825633, num\_entities=221

Model 512, 6 Sentences: avg\_time=1.9526534175872803, std\_time=0.26904821450167277, num\_entities=220

\_\_\_

Model 128, 1 Sentence: avg\_time=0.016638317346572874, std\_time=0.031701654430750376, num\_entities=809

Model 128, 2 Sentences: avg\_time=0.01688245153427124, std\_time=0.003527863076735664, num\_entities=802

Model 128, 3 Sentences: avg\_time=0.015921781163015767, std\_time=0.0020795097319088658, num\_entities=834

Model 128, 4 Sentences: avg\_time=0.01604170036315918, std\_time=0.001900436447221994, num\_entities=751

Model 128, 5 Sentences: avg\_time=0.021511584520339966, std\_time=0.004662328881084117, num\_entities=659

Model 128, 6 Sentences: avg\_time=0.016997781342375066, std\_time=0.0017178396396699029, num\_entities=553

Model 128, 7 Sentences: avg\_time=0.016980608026464502, std\_time=0.0021242696121221215, num\_entities=479

Model 128, 8 Sentences: avg\_time=0.0167020206451416, std\_time=0.0018343889875642697, num\_entities=416

Model 128, 9 Sentences: avg\_time=0.018851388778005327, std\_time=0.004285382147937495, num\_entities=417

Model 128, 10 Sentences: avg\_time=0.02521048307418823, std\_time=0.004084150583623624, num\_entities=356

Model 128, 12 Sentences: avg\_time=0.01741632961091541, std\_time=0.00244971527006539, num\_entities=287

Model 128, 14 Sentences: avg\_time=0.017255137364069622, std\_time=0.0017555753750601918, num\_entities=242

Model 128, 16 Sentences: avg\_time=0.017858149513365732, std\_time=0.002065247813397539, num\_entities=215

Model 512, 1 Sentence: avg\_time=0.01849769139289856, std\_time=0.0033556562525482564, num\_entities=811

Model 512, 2 Sentences: avg\_time=0.020178873538970948, std\_time=0.004445517929352908, num\_entities=813

Model 512, 3 Sentences: avg\_time=0.02184544994445618, std\_time=0.005388895049938937, num\_entities=864

Model 512, 4 Sentences: avg\_time=0.019722519874572755, std\_time=0.0023691553589020916, num\_entities=856

Model 512, 5 Sentences: avg\_time=0.02098231911659241, std\_time=0.0037000429054427826, num\_entities=878

Model 512, 6 Sentences: avg\_time=0.029653355033097867, std\_time=0.010011665176755278, num\_entities=872

Model 512, 7 Sentences: avg\_time=0.023986609665663926, std\_time=0.0055756814034308095, num\_entities=890

Model 512, 8 Sentences: avg\_time=0.025785512924194336, std\_time=0.006267029358370661, num\_entities=887

Model 512, 9 Sentences: avg\_time=0.03498728360448565, std\_time=0.011713483623360998, num\_entities=907

Model 512, 10 Sentences: avg\_time=0.03244863510131836, std\_time=0.009947895434015606, num\_entities=912

Model 512, 12 Sentences: avg\_time=0.03502129656927926, std\_time=0.011714845940739538, num\_entities=886

Model 512, 14 Sentences: avg\_time=0.0391899844010671, std\_time=0.01225369751871037, num\_entities=909

Model 512, 16 Sentences: avg\_time=0.055350606403653586, std\_time=0.024700700591952906, num\_entities=873

# Study Plan

1. Attention is all you need, BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), DeBERTa
2. NER head for BERT
3. How metrics of NER are computed
   1. Micro, Macro and Weighted Average of F1-Score
4. CRF
5. position-wise FFN
6. XLM, XLM-RoBERTa (Conneau et al., 2019)
7. Customized tokenizer in SecureBERT
8. TNer Paper
9. More sophisticated ideas in “Multi-features based Semantic Augmentation Networks for Named Entity Recognition in Threat Intelligence”

# Tasks

1. Finish with the important parts of literature review
2. Validate the classes of CyNER
   1. {7: 'B-Indicator',
   2. 6: 'B-Malware',
   3. 2: 'B-Organization',
   4. 1: 'B-System',
   5. 4: 'B-Vulnerability',
   6. 9: 'I-Indicator',
   7. 10: 'I-Malware',
   8. 3: 'I-Organization',
   9. 8: 'I-System',
   10. 5: 'I-Vulnerability',
   11. 0: 'O'}
3. Maximal size to analyse (prefer passage, article size) - internal pipeline for sentences ?
   1. CyNER uses <https://www.nltk.org/api/nltk.tokenize.PunktSentenceTokenizer.html> to split text into separate sentences and pass each one to the model separately
      1. Pro: can handle text of any size
      2. Con: If the model can work on larger sequence wouldn’t it make it perform better ? Also slower for larger textx
   2. SecureBERT-NER is limited to sequence size (max\_position\_embedding is 514)
   3. **Proposed solution**: separate to sentences. Concatenate them together to be below 512 token size each time and run the model on each separately (sequentially or in parallel)
4. Understand why CyNER was able to unite entities together and SecureBERT-NER not
   1. <https://github.com/asahi417/tner> was doing post-processing to unite them together. Now I use it as well
5. Take a dataset (either with ground truth or not. Of what size ?)
   1. How many entities they found ? How much of each kind ? **Validate the results**
   2. How much time it took for each
6. Figure out if CyNER was trained sentence by setence
   1. Yes. Inputs is list of sentences (see get\_dataset:decode\_file)
7. Compare two options (in terms of performance and num of results)
   1. Combine full sentences until 512 tokens reached
   2. Split at 512 tokens without keep on full sentences
8. Explore CyNER on DNRTI
9. If shown part of word as entity mark the whole word as entity
   1. If another part of word is of different entity do not do it
10. Check if 512 token return reasonable results
11. Remove also duplicate from SecureBert-NER for fair comparison + 1/128 + 10/512
12. SecureBERT-NER: if two words are adjacent and of the same type unite them
13. Check the problem of adjacent words (tner vs pipe)
14. Apply the same logic into NDRTI (Create NDRTIDataset in github)
15. Create comparison in document between CyNER and SecureBert-NER and decide between them
16. Understand difference between tner and cyner locally
    1. CyNER relies on tokenizer.decode to get the values which causes CyNER to work only on tokenizers that don’t add whitespace in the begginning. AI4Sec/cyner-xlm-roberta-base tokenizer doesn't add whitespace in the beginning of sentence but CyberPeace-Institute/SecureBERT-NER does (which is the standard <https://github.com/huggingface/tokenizers/issues/608>)
17. Optional tasks:
    1. Check if the other CyNER “models” add more new data compared to its LLM
    2. Prepare Literature presentation & questions for the team
    3. F1 evaluation of two models
    4. Add CRF layer to SecureBERT-NER and train again
    5. Create deployable code
    6. Train CyNER but with DeBERTa (V3 ?) ?