



Security Insider Lab II

Second Lab - Creating LSTM Model

Sayed Alisina Qaderi, Atiqullah Ahmadzai & Dusan Dordevic

15. Mai 2024

Contents

1	Introduction	1
2	Methods	2
2.0.1	Data Collection	2
2.0.2	Keyword Filtering	3
2.0.3	Language Segregation	3
2.0.4	Commit Analysis	4
2.0.5	Dataset Acquisition	5
2.0.6	Model Training	5
3	Results	7
3.0.1	Data Collection	7
3.0.2	Keyword Filtering	7
3.0.3	Language Segregation	7
3.0.4	Commit Analysis	10
3.0.5	Model Training	10
4	Discussion	11
4.0.1	Creating the LSTM models	11

List of Figures

2.1	Gathering repositories	3
2.2	GetDiffs missing os	4
2.3	GetDiffs typo issue	4
2.4	Sklearn Utils Class Weight	6
3.1	Scrapped Repositories	8
3.2	Showcases filtered repositories	8
3.3	No showcases filtered repositories	9
3.4	Repositories with Python code	9
3.5	Repositories without Python code	10
3.6	PyCommitsWithDiffs data	10

1 Introduction

This report documents the third exercise of Security Insider Lab 2, detailing our process in training our model and creating the dataset. The exercise comprised six key steps.

Data Collection : We initiated the process by scraping GitHub and gathering data from 2000 repositories.

Keyword Filtering: Next, we examined the names and readme files of the scraped repositories, filtering out those that aligned with our predefined keywords.

Language Segregation: Our third step involved segregating libraries with Python code from those without.

Commit Analysis: Subsequently, we meticulously traversed the filtered repositories, extracting all differences from their commits.

Dataset Acquisition: In the fifth step, we procured the necessary dataset provided by the central repository, which is crucial for subsequent stages.

Model Training: Finally, we proceeded to train our model based on the acquired dataset from the previous step, completing the exercise.

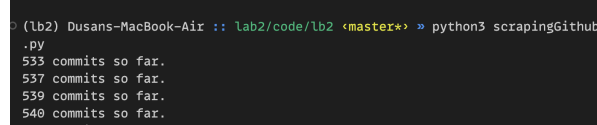
2 Methods

2.0.1 Data Collection

We initiated the data collection process by scraping GitHub repositories. Utilizing web scraping techniques, we gathered data from 2000 repositories, encompassing a diverse range of projects. The default amount of repositories was 100000, which is a considerable amount. Considering our time constraints and the need for a more efficient process, we made the decision to change the default steps to 2000, keeping you informed and involved in the process. 2.1

As we worked on **scrapingGithub.py** file, we have faced the following issues. The solutions are also mentioned in below:

- Missing **requests** library solved with *pip install requests*
- Missing **requests_oauthlib** library, solved with *pip install requests_oauthlib*
- Missing **all_commits** file which we created manually.
- Requests needed exception handling in case of losing the network.



```
(lb2) Dusan-MacBook-Air :: lab2/code/lb2 <master> » python3 scrapingGithub.py
533 commits so far.
537 commits so far.
539 commits so far.
540 commits so far.
```

Figure 2.1: Gathering repositories

- Missing Github Access Token, we generated through Github and save to **access** file inside the project folder.

After solving the above issue, they successfully ran the **scrapingGithub.py**, and its results were saved in **allcommits.json** file.

2.0.2 Keyword Filtering

Following data collection, we performed keyword filtering to refine the dataset with **filterShowcase.py** file. By examining the names and readme files of the scraped repositories, we identified and filtered out repositories that matched our predefined keywords. We have filtered the following showcases:

- `toomuchsecurity = ['offensive', 'pentest', 'vulnerab', 'security', 'hack', 'exploit', 'ctf ', ' ctf', 'capture the flag','attack']`
- `alittletoomuch = ['offensive security', 'pentest', 'exploits', 'vulnerability research', 'hacking', 'security framework', 'vulnerability database', 'simulated attack', 'security research']`

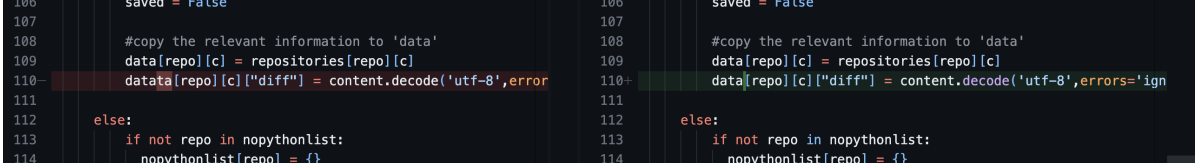
2.0.3 Language Segregation

With the help of **getDiffs.py** class file, we refined the dataset by segregating libraries with Python code from those without. This step was crucial for our subsequent analysis,



```
Code > getDiffs.py > ...
1 import requests
2 import time
3 import sys
4 import json
5 import datetime
6
7
1 import requests
2 import time
3 import sys
4 import json
5 import datetime
6 import os
7
```

Figure 2.2: GetDiffs missing os



```

106 saved = False
107
108 #copy the relevant information to 'data'
109 data[repo][c] = repositories[repo][c]
110 datata[repo][c]["diff"] = content.decode('utf-8',error
111
112 else:
113     if not repo in nopythonlist:
114         nopythonlist[repo] = {}
115
106 saved = False
107
108 #copy the relevant information to 'data'
109 data[repo][c] = repositories[repo][c]
110 data[repo][c]["diff"] = content.decode('utf-8',errors='ign
111
112 else:
113     if not repo in nopythonlist:
114         nopythonlist[repo] = {}
115
```

Figure 2.3: GetDiffs typo issue

focusing specifically on repositories containing Python code. During this process, we faced the following issues:

- missing os library which resolved by *import os* 2.2.
- **datata** typo in line 110, which resolved by changing to **data** 2.3.

2.0.4 Commit Analysis

In this step, we meticulously analyzed and downloaded the commits of the filtered repositories with the help of the **getData.py** file. By extracting all differences from their commits, we gained insights into the projects' evolution and the changes made over time. We faced with only one issue in this phase *ModuleNotFoundError: No module named 'keras.layers.convolutional'* which resolved by *pip install keras* and the rest of code worked ideally and saved the result in **PyCommitsWithDiffs.json** JSON file.

2.0.5 Dataset Acquisition

To augment our dataset, we procured additional data from the main repository. This supplementary dataset, provided by the main repository, was essential for enhancing the depth and scope of our analysis in subsequent stages.

2.0.6 Model Training

The final step involved training our model based on the augmented dataset acquired in the previous step. This step handled with the help **makemodel.py**. The Python script is a machine learning pipeline for training and evaluating an LSTM (Long Short-Term Memory) model for identifying vulnerabilities. We have trained the model over four types of vulnerabilities, XSS (Cross-Site Scripting), Path Disclosure, Remote Code Execution. The model specification is 10 for minimum count, 200 for iterations, 300 for vector size.

While working with this script we have faced the following issues due to library deprecations and changes in the Python version:

1. **Sklearn.util class__weight** issue which the input parameters changed in the latest versions. The differences shown in the figure 2.4.
2. **AttributeError: The vocab attribute was removed from KeyedVector in Gensim 4.0.0.**, the solution is changing vocab to **key_to_index**.
3. **vector = w2v_model[t] 'Word2Vec' object is not subscriptable** resolved by changing *w2v_model* to *m2v_model.wv*.
- 4.


```
# Before the changes
# class_weights = class_weight.compute_class_weight('balanced', numpy.unique(y_train), y_train)

# After the changes
classes = numpy.unique(y_train)
class_weights = class_weight.compute_class_weight(class_weight='balanced', classes=classes, y=y_train)
```

Figure 2.4: Sklearn Utils Class Weight

3 Results

3.0.1 Data Collection

All results are saved in `allcommits.json()` 3.1 file that we are required to create prior to executing the script. If the file is not created accordingly, the following error message is printed in the terminal: "The file is empty or does not exist."

3.0.2 Keyword Filtering

The result of `filterShowcase.py` is saved to the **DataFilter.json** file. It contains two JSON arrays for handling showcases and not showcases. The result is shown in figures 3.2 and 3.2.

3.0.3 Language Segregation

The result, which separates those repositories with Python code from those without Python code, is also saved in the **DataFilter.json** file. The results have been saved in two separated JSON arrays with names **no-python** and **python**, as shown in Figures 3.4 and 3.5.

3 Results

```
Code > {} all_commits.json > ...
1 {"https://github.com/openucx/ucx": {"98b8e8c0c9722541485f7a4efde59dbc3b29eba8": {"url": "https://api.github.com/repos/openucx/ucx/commits/98b8e8c0c9722541485f7a4efde59dbc3b29eba8", "sha": "98b8e8c0c9722541485f7a4efde59dbc3b29eba8", "keyword": "buffer overflow prevent"}, "d66092921c073838ee1670e2530151acfea2ebde": {"url": "https://api.github.com/repos/openucx/ucx/commits/d66092921c073838ee1670e2530151acfea2ebde", "html_url": "https://github.com/openucx/ucx/commit/d66092921c073838ee1670e2530151acfea2ebde", "sha": "d66092921c073838ee1670e2530151acfea2ebde", "keyword": "buffer overflow prevent"}}, "https://github.com/irontec/sngrep": {"f3f8ed8ef38748e6d61044b39b0dabd7e37c6809": {"url": "https://api.github.com/repos/irontec/sngrep/commits/f3f8ed8ef38748e6d61044b39b0dabd7e37c6809", "html_url": "https://github.com/irontec/sngrep/commit/f3f8ed8ef38748e6d61044b39b0dabd7e37c6809", "message": "Fix Buffer Overflow in SIP Header Processing\n\nResolved a critical buffer overflow in handling \"Call-ID\" and \"X-Call-ID\" SIP headers. This patch adds bounds checking and ensures string null-termination, preventing potential arbitrary code execution or DoS from malformed SIP messages.", "sha": "f3f8ed8ef38748e6d61044b39b0dabd7e37c6809", "keyword": "buffer overflow prevent"}}, "https://github.com/ericgoins/vifm": {"d0d27d206d9794e82ce578e59b4d3353c569699e": {"url": "https://api.github.com/repos/ericgoins/vifm/commits/d0d27d206d9794e82ce578e59b4d3353c569699e", "html_url": "https://github.com/ericgoins/vifm/commit/d0d27d206d9794e82ce578e59b4d3353c569699e", "message": "Fix a couple of buffer overflow warnings\n\nThey prevent optimized build with -Werror.", "sha": "d0d27d206d9794e82ce578e59b4d3353c569699e", "keyword": "buffer overflow prevent"}, "5cfd9cf0fa6599b1051cdd5b87166ea036749654": {"url": "https://api.github.com/repos/ericgoins/vifm/commits/5cfd9cf0fa6599b1051cdd5b87166ea036749654", "html_url": "https://github.com/ericgoins/vifm/commit/5cfd9cf0fa6599b1051cdd5b87166ea036749654", "sha": "5cfd9cf0fa6599b1051cdd5b87166ea036749654", "keyword": "buffer overflow fix"}, "e55e42aeb8393b3085caa49a86f30f68a1a3e952": {"url": "https://api.github.com/repos/ericgoins/vifm/commits/e55e42aeb8393b3085caa49a86f30f68a1a3e952", "html_url": "https://github.com/ericgoins/vifm/commit/e55e42aeb8393b3085caa49a86f30f68a1a3e952", "sha": "e55e42aeb8393b3085caa49a86f30f68a1a3e952", "keyword": "buffer overflow prevent"}}
```

Figure 3.1: Scrapped Repositories

```
{
  "showcase": {
    "nethackathon/nethackathon-nethack": {},
    "elfmz/far2l": {},
    "TheBombSquad/SMB2WorkshopMod": {},
    "ccavxx/exploit-db": {},
    "arancour69/exploit-database": {},
    "artyang/exploitdb": {},
    "merlinepedra25/EXPLOITDB": {},
    "razortag97/offsec_exploits": {},
    "conan25216/exploitdb": {},
    "chris-0x01/https-gitlab.com-exploit-database-exploitdb": {},
    "Jaboox/offensive-security-exploit-database": {},
    "michael101096/offensive-security-exploitdb": {},
    "jameser/exploitdb-reduced": {},
    "Brocks-Collections/exploit-database": {},
    "NoorahSmith/Exploit-DB-offsec": {},
    "sawhtetkhine-soe/exploit-database": {},
    "HackYourShit/exploit-database": {},
    "ubboolean/exploitdb": {},
    "chushuai/webappsecurity": {},
    "anhilo/exploit-database": {},
    "3v1lW1th1n/exploit-exploitdb": {},
    "offensive-security/exploitdb": {},
    "EthicalSecurity-Agency/exploit-database-exploitdb": {},
    "xianlimei/exploitdb": {},
    "jhe8281/exploitdb": {},
    "merlinepedra25/EXPLOITDB": {},
    "SYNgularity1/exploits": {},
    "bsd-hacker/freebsd": {},
    "brain-hackers/buildroot": {},
    "slsa-framework/slsa-verifier": {},
    "hackagadget/tarfs": {},
    "Innovation-Web-3-0-Blockchain/Hacking-Smart-Contracts": {},
    "p0-security/iam-privilege-catalog": {},
    "jitsecurity-soss/langchain": {},
    "jitsecurity-soss/python-code-": {},
    "H4lo/awesome-IoT-security-article": {},
    "mullvad/mullvadvpn-app": {},
    "derekarends/solidity-vulnerabilities": {}
  }
}
```

Figure 3.2: Showcases filtered repositories

3 Results

```
"showcase": {--
},
"noshowcase": {
  "openucx/ucx": {},
  "ironotec/sngrep": {},
  "ericgoins/vifm": {},
  "Arena-Rosnav/rosnav-rl": {},
  "2ndBaseChris/vifmChallenge": {},
  "thesourcerer8/hddsupliclone": {},
  "selimvuz/Library-Automation-System-in-C": {},
  "golded-plus/golded-plus": {},
  "Ridderrasmus/RPVoiceChat": {},
  "blooo-io/LedgerHQ-app-plugin-THORSwap": {},
  "nrfconnect/sdk-nrf": {},
  "nikso-itu/duckdb-oml": {},
  "netdata/netdata": {},
  "qwx9/syro": {},
  "RealYukiSan/chat_app": {},
  "dtrebilco/PortalsSokol": {},
  "Chysn/VIC20-wAx2": {},
  "memoryhole/libkiss": {},
  "rcjcooke/apfc": {},
  "google-deepmind/mujoco": {},
  "saprykin/plibsys": {},
  "quectel-official/QModemHelper": {},
  "gonoso/blender": {},
  "Bforartists/Bforartists": {},
  "UPBGE/upbge": {},
  "mrohne/ngspice": {},
  "ridobe/ridobe-seiei": {},
  "imr/ngspice": {},
  "amalej/onfire-cli": {},
  "RobertONelson/linux-stable-rcn-ee": {},
  "TaranovK/linuxnext": {},
  "Aquatic-Symbiosis-Genomics-Project/curation_tool": {},
  "torvalds/linux": {},
  "alexzahnaudio/PFM10": {},
  "casept/linux-samsung-smartwatch": {},
  "xu1119/torvalds-linux": {},
```

Figure 3.3: No showcases filtered repositories

```
Code > {} DataFilter.json > {} showcase
1 |
2 > "showcase": {--
44 | },
45 > "noshowcase": {--
1487 | },
1488 > "no-python": {--
2795 | },
2796 | "python": {
2797 |   "Arena-Rosnav/rosnav-rl": {},
2798 |   "recursionpharma/gflownet": {},
2799 |   "theroyallab/tabbyAPI": {},
2800 |   "amiyuki7/ethics_game": {},
2801 |   "mr-mdd/C5CK541---EOM---Group-A": {},
2802 |   "2lambda123/https-github.com-NationalSecurityAgency-ghidra": {},
2803 |   "salukadev/guide_app_rpi": {},
2804 |   "sab-rinakl/ctf": {},
2805 |   "Freedom-of-Form-Foundation/FFFTail": {},
2806 |   "securesauce/precli": {},
2807 |   "NetBSD/pkgsrc": {},
2808 |   "DL124/aaf-external": {},
```

Figure 3.4: Repositories with Python code

3 Results

```
Code > {} DataFilter.json > {} showcase

1  {
2  >   "showcase": { -
44  },
45  >   "noshowcase": { -
1487  },
1488   "no-python": {
1489     "openucx/ucx": {},
1490     "irontec/sngrep": {},
1491     "ericgoins/vifm": {},
1492     "2ndBaseChris/vifmChallenge": {},
1493     "thesourcerer8/hddsupliclone": {},
1494     "selimvuz/Library-Automation-System-in-C": {},
1495     "golded-plus/golded-plus": {},
1496     "Ridderrasmus/RPVoiceChat": {},
1497     "blooo-io/LedgerHQ-app-plugin-THORSwap": {},
1498     "nrfconnect/sdk-nrf": {},
1499     "nikso-itu/duckdb-oml": {},
1500     "netdata/netdata": {},
1501     "qwx9/syro": {},
1502     "RealYukiSan/chat_app": {}.
```

Figure 3.5: Repositories without Python code

[illegible]

Figure 3.6: PyCommitsWithDiffs data

3.0.4 Commit Analysis

As mentioned before, the result of this step has been saved in the **PyCommitsWithDiffs.json** JSON file. The files are structured as JSON objects; each object is a repository, and commit diffs are located inside that. Figure 3.6 shows a sample of this step result.

3.0.5 Model Training

Present the results of training the model on the augmented dataset, including model performance metrics, such as accuracy, precision, recall, F1-score, and any insights derived from the trained model's predictions.

4 Discussion

4.0.1 Creating the LSTM models

Makemodel.py script is used for creating data models. It splits the data in 3 different segments (train, validate, final). On the line of code 134, the samples are randomized with **for** loop. However, there are certain issues regarding this code. Library **Keras** that is imported in **myutils.py** file is causing a build error. Keras package is now **included** in previous installation of tensorflow package. Therefore, imports needed to be adjusted accordingly. "from keras.datasets import imdb" is now "tensorflow.keras.datasets". Also, we need to install all necessary packages (gensim, sklearn, tensorflow). Next, datasets are created using word2vec models created in previous exercise. Unfortunately, new error occurs where we need to adjust `if t in word_vectors.key_to_index and t != " ":` this line of code to match the new key to index parameter. Console is clear from errors so we can proceed to make our new vulnerability model. After loading the data, script creates new training dataset `print("Creating training dataset... (" + mode + ")")`, followed by `print("Creating validation dataset...")` and finally `print("Creating finaltest dataset...")`.