Ljavax/crypto/spec/deskeyspec;.<init>:([b)v	0.057388
landroid/app/application;.<init>:()v	0.056203
Ljava/lang/reflect/field;.gettype:()Ljava/lang/class;	0.037152
Ljava/lang/class;.forname:(Ljava/lang/string;)Ljava/lang/class;	0.027028
lorg/json/jsonarray;.tostring:()Ljava/lang/string;	0.014487
Ljavax/crypto/cipher;.getinstance:(Ljava/lang/string;)Ljavax/crypto/cipher;	0.011853
Ljava/io/bytearrayoutputstream;.flush:()v	0.008844
Ljava/io/randomaccessfile;.writelong:(j)v	0.007683
landroid/widget/toast;.maketext:(landroid/content/context;ljava/lang/charsequence;i)landroid/widget/toast;	0.005666
Ljava/lang/reflect/array;.getint:(Ljava/lang/object;i)i	0.003103

Figure 8. API-calls and attention weights for an Android malware app

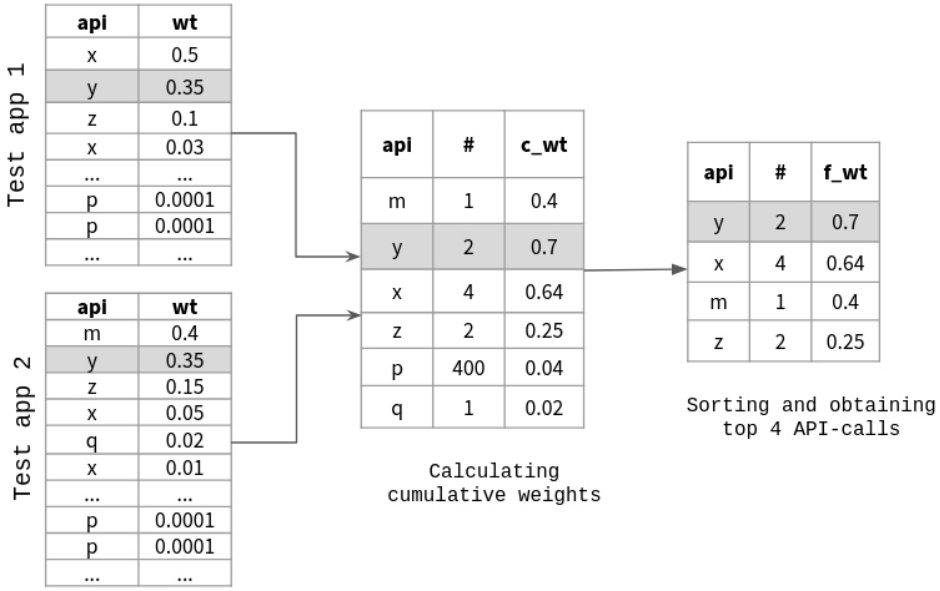

Figure 9. Top feature selection strategy: For the sake of readability, here we illustrate the selection process with a tiny set of two apps whereas the goal is to identify only top 4 features.

In the current work, we finally extracted 200 top features of malicious apps. In order to prove that these top features are valid, we plan to compare them with the manually extracted features in a future work.

Someone may wonder that instead of considering so many individual API calls as in the current work, whether it is better to merge API calls into groups such as permission related, string related, File/Network IO related, and so on. Although the features will then become more coarse-grained, it is important to explore this route in the future.

8. Conclusions

We leveraged attention-based deep learning approaches for security vetting of Android apps. API-calls extracted from the Android apps were used as the artifacts for deep-learning models. Such API-calls are quasi-sequential in nature. We experimented with two attention-based models: Bi-LSTM Attention and Self-Attention. Both models gave competitive results. Additionally, after analyzing the attention weights from these two models, we identified top 200 API-calls that reflect the maliciousness of an Android app.

Our experiments show that deep learning models can be implemented for large scale Android app security vetting. This can save human time and effort from manually handpicking the malicious features. The top features identified from the models can be further studied by the research community to potentially discover new malware signature.

Table 5. Common API-calls from the two models

Common API-calls
ljava/io/file;.mkdir:()z
ljava/io/file;.exists:()z
ljava/io/fileinputstream;.close:()v
ljava/lang/reflect/method;.setaccessible:(z)v
landroid/content/res/resources;.getassets:()landroid/content/res/assetmanager;
ljava/io/file;.delete:()z
ljava/io/filenotfoundexception;.printstacktrace:()v
landroid/telephony/telephonymanager;.getdeviceid:()ljava/lang/string;
landroid/os/asynctask;.onpostexecute:(ljava/lang/object;)v
ljava/io/file;.listfiles:()[ljava/io/file;
landroid/content/res/resources;.getboolean:(i)z
landroid/util/base64;.decode:([bi)[b
landroid/app/application;.onconfigurationchanged:(landroid/content/res/configuration;)v
ljava/lang/stringbuilder;.insert:(iljava/lang/string;)ljava/lang/stringbuilder;
lorg/apache/http/statusline;.getstatuscode:()i
landroid/content/sharedpreferences;.edit:()landroid/content/sharedpreferences$editor;
ljava/io/objectoutputstream;.writeobject:(ljava/lang/object;)v
landroid/app/application;.onterminate:()v
ljava/io/inputstream;.mark:(i)v
ljava/lang/process;.destroy:()v
ljava/util/zip/zipinputstream;.closeentry:()v

Table 6. Accuracy of standard ML algorithms

ML Model	App Class	Precision	Recall	F1-Score	Area Under PR Curve
Bernoulli Naive Bayes	Malicious	0.6209	0.6105	0.6156	0.6845
	Benign	0.8709	0.8757	0.8733	
K-Nearest Neighbor (K=5)	Malicious	0.9071	0.8881	0.8975	0.9471
	Benign	0.9630	0.9697	0.9663	
Support Vector Machine	Malicious	0.9133	0.8940	0.9035	0.9547
	Benign	0.9649	0.9717	0.9683	

Table 7. Deep learning models' accuracy with high quality malware

Model	**Review Type**	**Precision**	**Recall**	**F1-Score**	**Area Under PR Curve**
Bi-LSTM Attn.	Benign	0.9554	0.9772	0.9662	0.9548
	Malicious	0.9266	0.8632	0.8938	
Self Attention	Benign	0.9283	0.9586	0.9432	0.9291
	Malicious	0.9084	0.7949	0.8479	

Acknowledgement. This work has been partially supported by the U.S. National Science Foundation (NSF) under grant no. 1718214 and 1717871. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF.

References

[1] Statcounter (2021), Android OS Market Share. URL https://gs.statcounter.com/os-market-share/mobile/worldwide.

[2] Chaulagain, D., Poudel, P., Pathak, P., Roy, S., Caragea, D., Ou, X. and Liu, G. (2020) Hybrid analysis of android apps for security vetting using deep learning. In *IEEE conference on communications and network security (CNS)*.

[3] Pascanu, R., Mikolov, T. and Bengio, Y. (2012) Understanding the Exploding Gradient Problem. *arXiv e-print* : arXiv:1211.5063.

[4] Bengio, Y., Simard, P. and Frasconi, P. (1994) Learning Long-term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks* : 157–166.

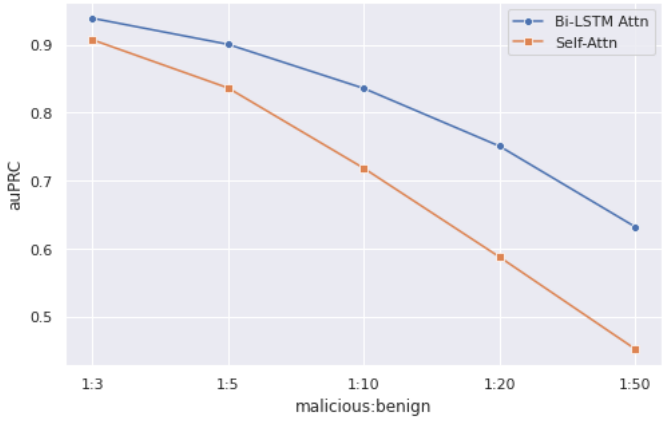

Figure 10. Experimenting with varying malware apps to benign apps ratio in the test set

[5] Gers, F.A., Schmidhuber, J. and Cummins, F. (1999) Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*: 850–855.

[6] Bahdanau, D., Cho, K. and Bengio, Y. (2016), Neural machine translation by jointly learning to align and translate. 1409.0473.

[7] Luong, M.T., Pham, H. and Manning, C.D. (2015), Effective approaches to attention-based neural machine translation. 1508.04025.

[8] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L. *et al.* (2017), Attention is all you need. 1706.03762.

[9] Arzt, S., Rasthofer, S., Fritz, C., Bodden, E., Bartel, A., Klein, J., Le Traon, Y. *et al.* (2014) FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps. *SIGPLAN* : 259–269.

[10] Wei, F., Roy, S., Ou, X. and , R. (2018) Amandroid: A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps. *ACM Transactions on Privacy and Security* : 1–32.

[11] Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H. and Rieck, K. (2014) DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In *Symposium on Network and Distributed System Security (NDSS)*: 23–26.

[12] Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., Ranganath, V.P. *et al.* (2015) Experimental study with real-world data for android app security analysis using machine learning. In *Proceedings of the 31st Annual Computer Security Applications Conference (ACSAC)*: 81–90.

[13] Aafer, Y., Du, W. and Yin, H. (2013) DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. In *International conference on security and privacy in communication systems*: 86–103.

[14] Onwuzurike, L., Mariconti, E., Andriotis, P., De Cristofaro, E., Ross, G. and Stringhini, G. (2017) MaMaDroid: Detecting Android Malware by Building Markov Chains of Behavioral Models. *arXiv e-prints* : arXiv:1612.04433.

[15] Gong, L., Li, Z., Qian, F., Zhang, Z., Chen, Q., Qian, Z., Lin, H. *et al.* (2020) Experiences of landing machine learning onto market-scale mobile malware detection. *Proceedings of the Fifteenth European Conference on Computer Systems* .

[16] Ke, X., Li, Y., Deng, R.H. and Chen, K. (2018) DeepRefiner: Multi-layer Android Malware Detection System Applying Deep Neural Networks. In *2018 IEEE European Symposium on Security and Privacy (EuroS P)*: 473–487.

[17] Hou, S., Saas, A., Chen, L. and Ye, Y. (2016) Deep4MalDroid: A Deep Learning Framework for Android Malware Detection Based on Linux Kernel System Call Graphs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*: 104–111.

[18] Karbab, E.B., Debbabi, M., Derhab, A. and Mouheb, D. (2018) MalDozer: Automatic Framework for Android Malware Detection using Deep Learning. *Digital Investigation* : S48–S59.

[19] (2018), Genymotion Android Emulator. URL https://www.genymotion.com/.

[20] Sutskever, I., Vinyals, O. and Le, Q.V. (2014), Sequence to sequence learning with neural networks. 1409.3215.

[21] Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. and Bengio, Y. (2014), Learning phrase representations using rnn encoder-decoder for statistical machine translation. 1406.1078.

[22] Wu, B., Chen, S., Gao, C., Fan, L., Liu, Y., Wen, W. and Lyu, M.R. (2021) Why an android app is classified as malware: Toward malware classification interpretation. *ACM Trans. Softw. Eng. Methodol.* **30**(2).

[23] Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H. and Xu, B. (2016) Attention-Based Bidirectional Long Short-Term Memory Networks for Relation Classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*: 207–212.

[24] Hochreiter, S. and Schmidhuber, J. (1997) Long Short-Term Memory. *Neural Computation* : 1735–1780.

[25] Schuster, M. and Paliwal, K. (1997) Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing* : 2673–2681.

[26] Cho, K., van Merrienboer, B., Bahdanau, D. and Bengio, Y. (2014), On the properties of neural machine translation: Encoder-decoder approaches. 1409.1259.

[27] Allix, K., Bissyandé, T.F., Klein, J. and Le Traon, Y. (2016) AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories*: 468–471.

[28] Pendlebury, F., Pierazzi, F., Jordaney, R., Kinder, J. and Cavallaro, L. (2019) TESSERACT: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium*: 729–746.