Analyzing CNN Based Behavioural Malware Detection Techniques on Cloud IaaS

Andrew McDole, M. Abdelsalam, M. Gupta, S. Mittal

Outline

1. Who am I?
2. Introduction & Motivation
3. Related Work
4. Methodology and Experimental Setup
5. Results
6. Future Work
7. Conclusion

Who am I?

- Andrew McDole
- Final Semester Masters in Computer Science
- Tennessee Technological University
- Focus in CyberSecurity

Introduction & Motivation

- According to a CISCO report [1], cloud data centers will process 94% of workloads in 2021.
- In a Sophos report [2], 70% of companies suffered a cloud breach in 2019 and 59% of those breaches are from malware or ransomware.
- People may attack data centers for monetary gain, to gather information about customers, or to make use of the data centers resources for nefarious reasons.
- Common attack types: DDoS, Botnets, Rootkits, etc.

Cloud Malware is one of the most prevalent threats!

Introduction & Motivation (cont.)

- Traditional malware detection techniques falls short in detecting new malware
  - Zero-day malware, Polymorphic malware, etc..
- Deep Learning (DL) based malware detection techniques has become more adept in detecting malware.
- Many approaches has been proposed using different DL techniques (CNN, RNN, etc.)

A proper analysis for the effectiveness of state-of-the-art DL techniques tailored specifically for online malware detection in cloud is needed.

- In this paper, we focus on **CNN** using **process performance metrics**.

Related Work

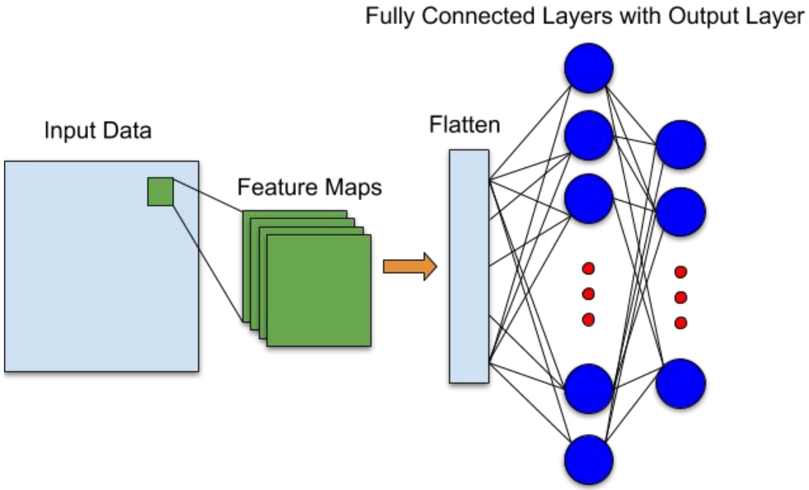
Related Work	Contribution
[3-5]	Focus on collecting API Calls
[6-8]	Focus on collecting System Calls
[9]	Focus on collecting Performance Counters
[10, 11]	Focus on collecting memory features
[3 - 11]	Limited to features that can be collected through the hypervisor
[12]	Utilizes metrics collected on the VM itself
[13]	Introduces a CNN technique to detect malware with a low profile

Our Contribution

Analyzing the effectiveness of applying **state-of-the-art CNN models** for behavioral malware detection using fine-grained light-weight **process performance metrics**.

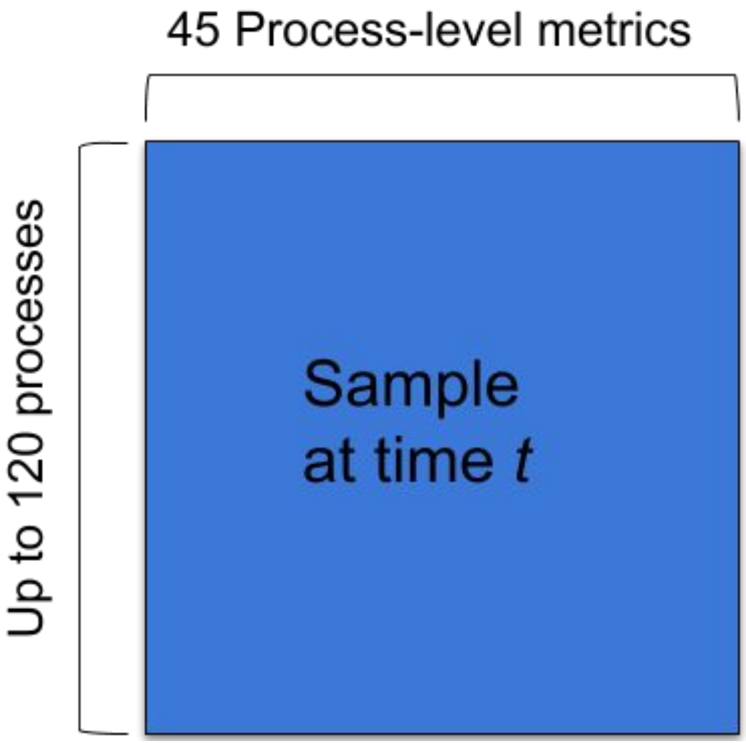
Convolutional Neural Networks

- Work on image data
- Builds spatial relationships between data



Methodology

- Up to 120 unique processes
- 45 process level metrics
- Data was organized into a matrix to represent 2 dimensional data for feeding into a CNN



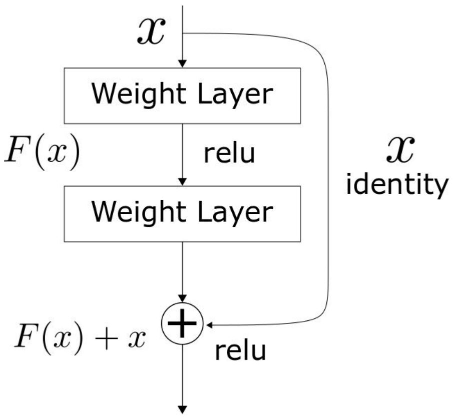
Methodology - Example Sample

Metric	Value	Metric	Value
sample_no	5672254	mem_swap	0
exp_no	23	mem_lib	0
vm_id	178	mem_text	217088
pid	1036	mem_uss	1105920
ppid	1	mem_dirty	0
sample_time	6/6/2018 19:32	mem_shared	3334144
process_creation_time	6/6/2018 19:32	mem_data	585728
status	sleeping	mem_vms	43921408
num_threads	1	mem_rss	3751936
kb_received	0	io_write_bytes	0
kb_sent	0	io_write_chars	76
num_fds	14	io_write_count	9
cpu_children_sys	0	io_read_bytes	958464
cpu_children_user	0	io_read_chars	61088
cpu_user	0.01	io_read_count	77
cpu_sys	0	ctx_switches_involuntary	43
cpu_percent	0	ctx_switches_voluntary	182
cpu_num	0	nice	0
name	dbus-daemon	ionice_ioclass	0
gid_real	111	ionice_value	0
gid_saved	111	label	0
gid_effective	111		

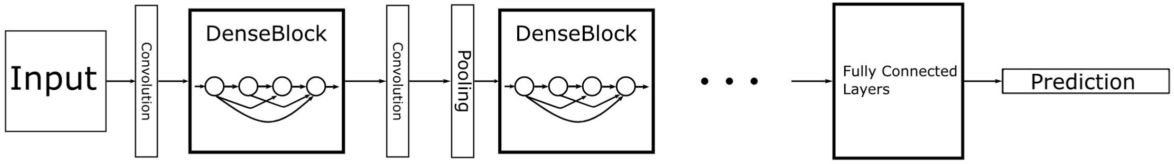
- 45 process-level features collected
- Strings were encoded using one-hot encoding

Methodology - CNN Models Used

- LeNet-5
- ResNet-50
- ResNet-101
- ResNet-152
- DenseNet-121
- DenseNet-169
- DenseNet-201



Residual Block Diagram



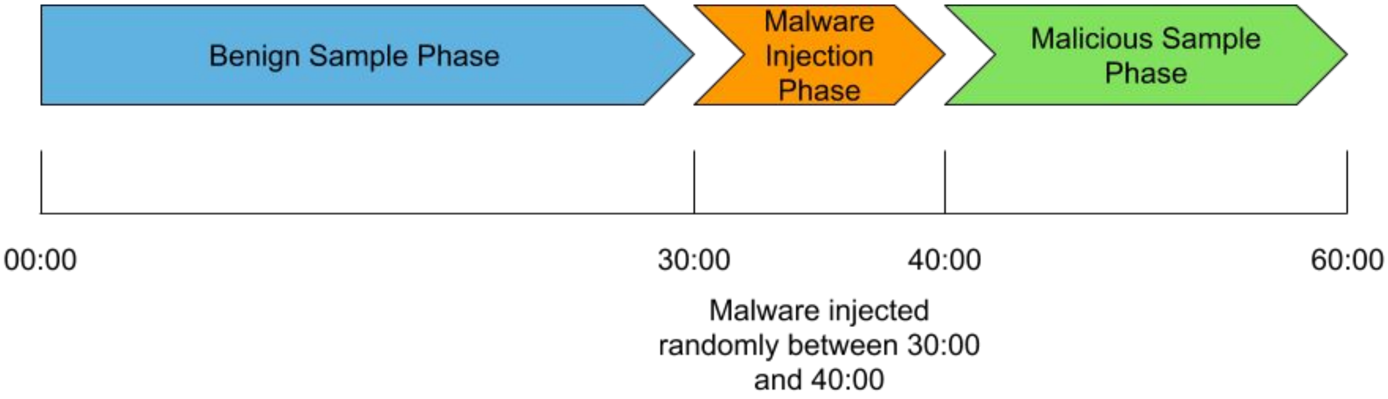
Dense Network

Experimental Setup

- Our work utilized an OpenStack testbed which allowed the malware to freely use the internet.
- This allows the malware to exhibit behavior more closely to the real world.
- Other work that involve a sandboxed environment may inhibit the malware's ability to execute, or the malware could detect the sandbox countermeasures and disable itself.

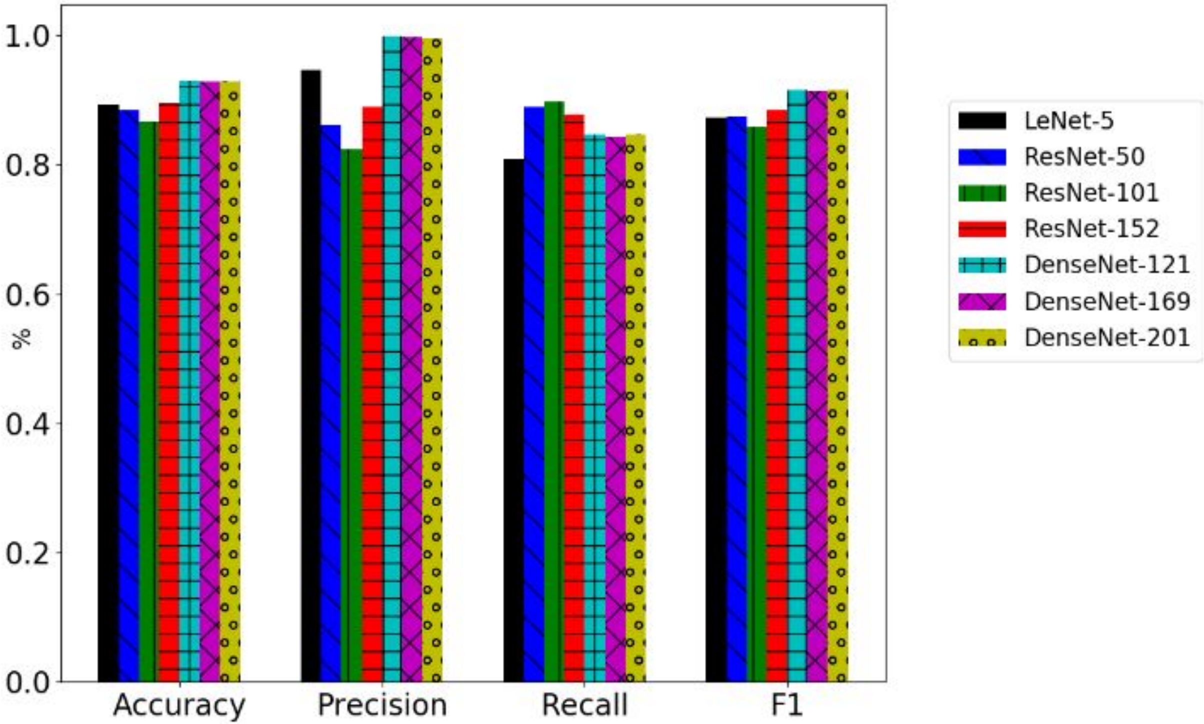
Experimental Setup

- Total runtime 60 minutes per malware
- 30 minutes of benign
- Random injection between minute 30 and 40
- Collect sample every 10 seconds on infected VM



Results

- DenseNets performed the best overall
- ResNets performed the best in Recall



Results

- DenseNet-121 performed well while having a lower time to train than all other deep models

Model	Validation Accuracy	Epoch Reached	Elapsed Time (s)	Detection Time (ms)
LeNet-5	89.9	29	170	54
ResNet-50	90.7	67	1815	96
ResNet-101	87.0	60	2940	130
ResNet-152	88.7	99	7029	165
DenseNet-121	92.1	32	1683	164
DenseNet-169	91.9	81	5848	209
DenseNet-201	91.5	36	3060	249

Future Work

- A future similar analysis of using RNNs which learns temporal dependency can be very useful.
- In the future, we plan to develop more cloud uses cases to yield different data to train on which might require new approaches to effectively detect malware.

Conclusion

- Seven CNN models were compared in performance for malware detection in the cloud.
- LeNet-5 sacrifices accuracy for speed in terms of time to train malware and time to detect malware
- ResNet models have higher recall scores than the other models tested which make them suitable for cases where allowing a false negative is unacceptable.
- DenseNet models performed the best overall with high accuracies, but took longer to train and to detect the malware.
- One limitation of using CNNs is that it does not capture the time correlation of the data samples. This is a result of using 2D CNNs which do not have a temporal dimension.

References

[1] https://www.cisco.com/c/en/us/solutions/executive-perspectives/annual-internet-report/index.html
[2] https://secure2.sophos.com/en-us/medialibrary/pdfs/whitepaper/sophos-the-state-of-cloud-security-2020-wp.pdf
[3] Mamoun Alazab et al. Zero-day Malware Detection Based on Supervised Learning Algorithms of API Call Signatures. InProc. of the Australasian Data Mining Conference, page 171–182, AUS, 2011. Australian Computer Society, Inc.
[4] Radu S Pirscoveanu et al. Analysis of Malware Behavior: Type Classification Using Machine Learning. InProc. of IEEE International conference on cyber situational awareness, data analytics and assessment, pages 1–7, 2015.
[5] Shun Tobiyama et al. Malware Detection With Deep Neural Network Using Process Behavior. InProc. of IEEE Annual Computer Software and Applications Conference, volume 2, pages 577–582, 2016.
[6] Joel A Dawson et al. Phase space detection of virtual machine cyber events through hypervisor-level system call analysis. InProc. of IEEE International Conference on Data Intelligence and Security (ICDIS), pages 159–167, 2018.
[7] P. Luckett et al. Neural network analysis of system call timing for rootkit detection.InProc. of Cybersecurity Symposium (CYBERSEC), pages 1–6, April 2016.
[8] Gianluca Dini et al. Madam: A multi-level anomaly detector for android malware.In Igor Kotenko and Victor Skormin, editors,Computer Network Security, pages 240–253, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
[9] John Demme et al. On the feasibility of online malware detection with performance counters.ACM SIGARCH Computer Architecture News, 41(3):559–570, 2013.
[10] Khaled N Khasawneh et al. Ensemble learning for low-level hardware-supported malware detection. InProc. of International Symposium on Recent Advances in Intrusion Detection, pages 3–25. Springer, 2015.
[11] Zhixing Xu et al. Malware detection using machine learning based analysis of virtual memory access patterns. InProc. of IEEE Design, Automation & Test in Europe Conference & Exhibition, 2017, pages 169–174, 2017.
[12] Mahmoud Abdelsalam, Ram Krishnan, and Ravi Sandhu. Clustering-based iaas cloud monitoring. InProc. of IEEE International Conference on Cloud Computing(CLOUD), pages 672–679, 2017.
[13] Mahmoud Abdelsalam et al. Malware detection in cloud infrastructures usingconvolutional neural networks. InProc. of IEEE International Conference on CloudComputing (CLOUD), pages 162–169, 2018.