Insights Gained From Constructing a Large Scale Dynamic Analysis Platform

DFRWS 2017| Austin,TX | Aug 7, 2017

Cody Miller, Dae Glendowne, Henry Cook, DeMarcus Thomas, Chris Lanclos, Patrick Pape

Mississippi State University

Outline

- Introduction
- Related Work
- System Overview
- Experiments
- Lessons Learned
- Future Work

Introduction

- Significant increases in malware reaching over 500 million in 2016 [1].
- Need for reliable, scalable and simple to use systems for analysts.
- Developed a scalable dynamic analysis platform and recorded the lessons learned

Related Work

- Effective dynamic analysis has visibility, is resistant to detection and scalable [2].

- Extracting information:
  - Most systems track API calls
  - Some follow steps between API calls
  - Some use taint analysis,
  - Some use multiple OS, bare-metal systems and hardware emulation

Related Work cont..

- Previous work compared number of samples executed per minute [3]
  - Execution time of 15 seconds
  - Barebox (2.69), VirtualBox (2.57), QEMU (3.74)

- Literature lacking an empirically selected execution time for a “large” number of samples [4]

System Overview

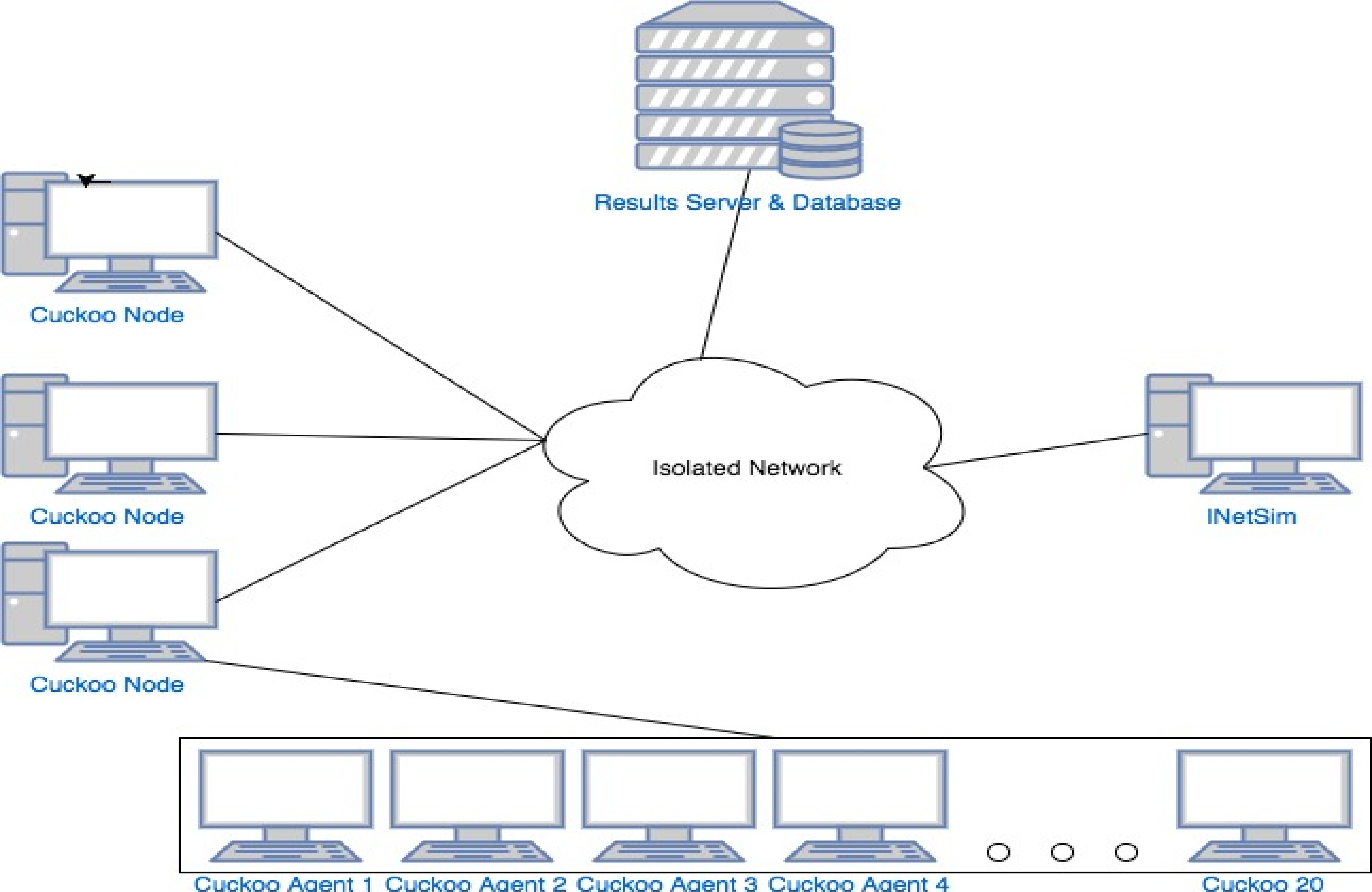
- Cuckoo Sandbox [5]
  - Collects API calls, network traffic, files dropped, memory dump, etc.
- Cuckoo Node:
  - CentOS 7 VM running Cuckoo Sandbox
  - 64 Gib of RAM and 28 virtual cores
  - Network adapter connected to an isolated network
  - The Cuckoo nodes each have 20 Cuckoo agent VMs within them.
  - QEMU 2.5.1
- Cuckoo Agents:
  - Windows 7 32-bit VMs, 512 Mib of RAM, 1 CPU Core, Adobe Reader 11 and Python 2.7

System Overview cont..

- INetSim
  - Software suite to simulate internet services
  - Agent VMs connected to same network as INetSim
- Results Server
  - CentOS 7 VM used to collect Cuckoo samples from the Cuckoo nodes
  - Improves performance over using Cuckoo built-in API
- Database
  - CouchDB database used as central location of malware processing pipeline

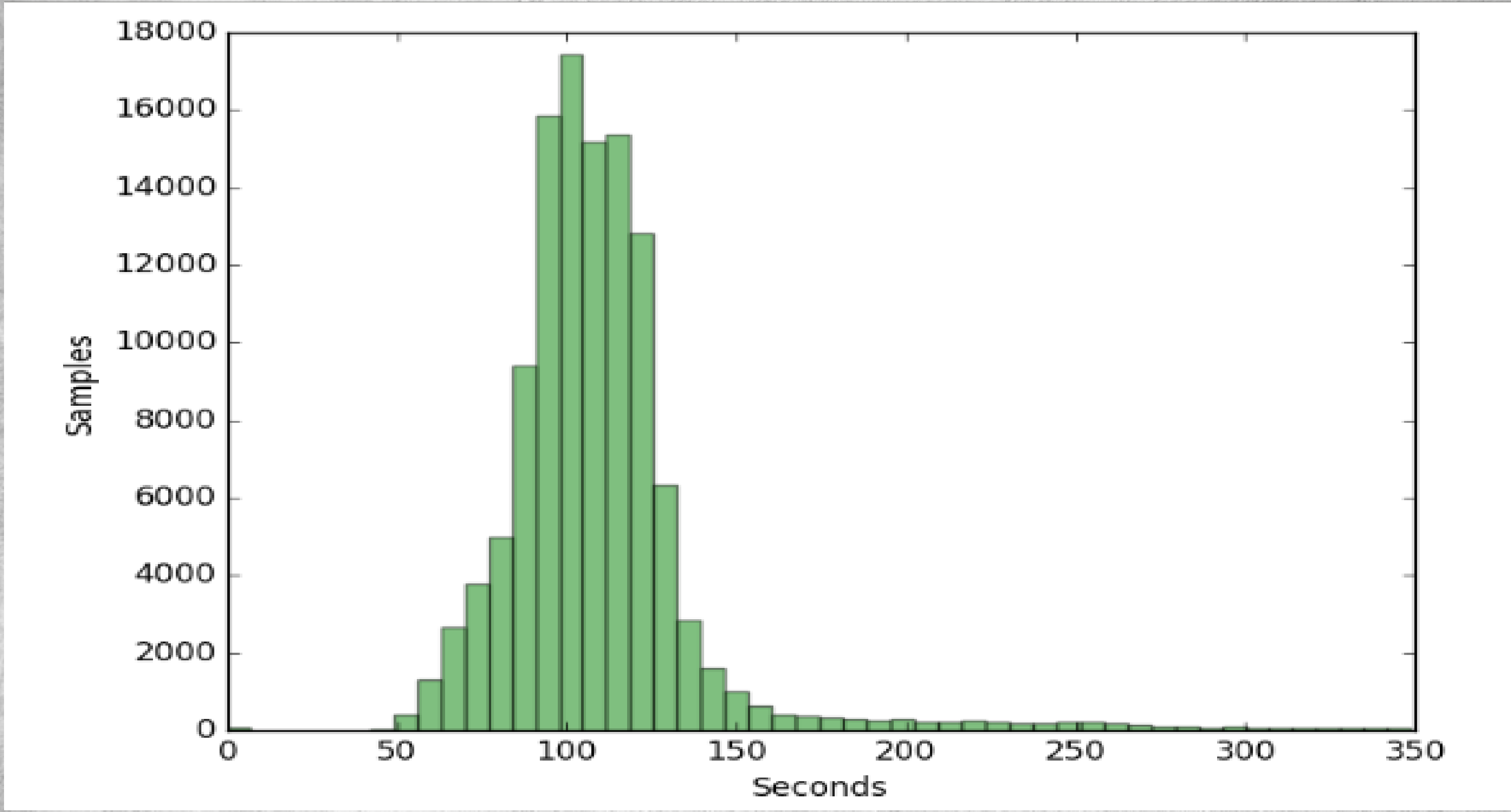
System Overview cont…

- Extended distribution script
  - Runs on result server
  - Uses the existing Cuckoo API and mounted storage to submit binaries and compress results for long term storage
  - Updates database with the status and details of samples
  - Connect to Cuckoo nodes on different subnets
  - Ability to add additional Cuckoo nodes


Results Server & Database
Cuckoo Node
Cuckoo Node
Isolated Network
INetSim
Cuckoo Node
Cuckoo Agent 1
Cuckoo Agent 2
Cuckoo Agent 3
Cuckoo Agent 4
Cuckoo 20

Experiments: Distribution Time

- Goal
  - Determine time overhead of distribution script on processed samples
- Not focused on the time taken to execute each binary on a Cuckoo agent
- Time delta from completion time to placement on long-term storage
- # of Samples: 118,055



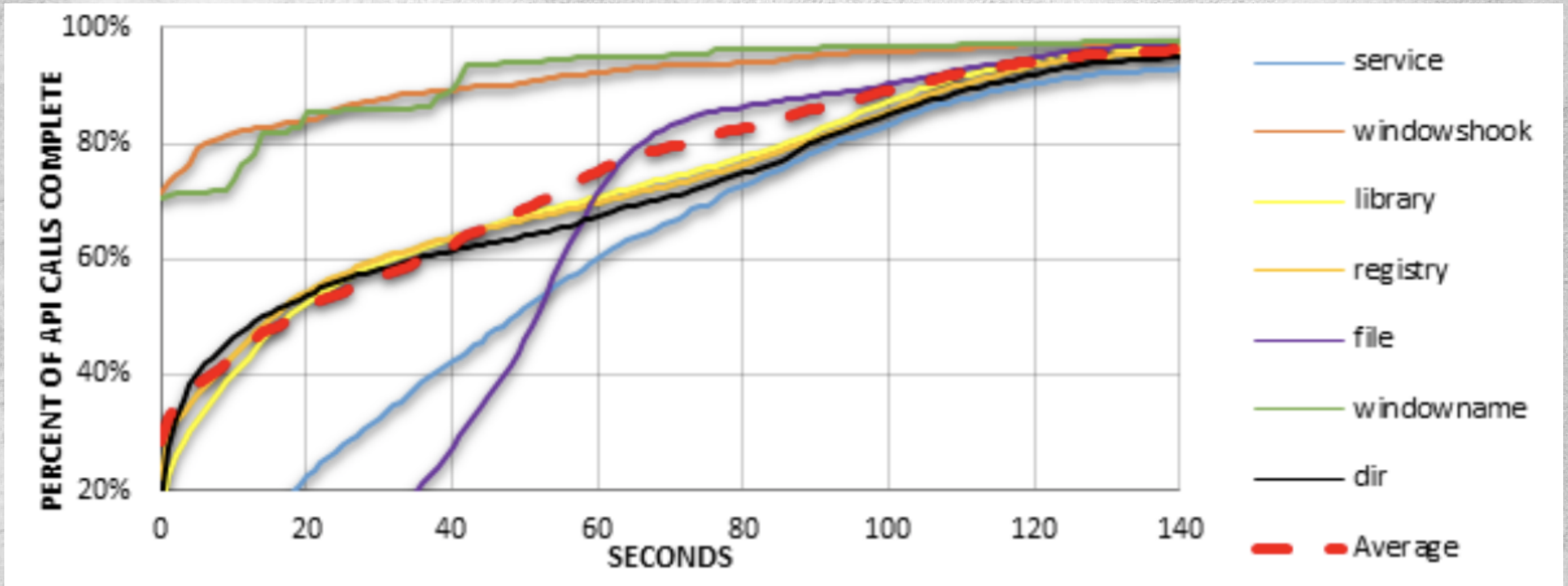
- Most samples take between 50 and 150 seconds
- Average duration of 114 seconds
- With processing capability of 60 samples concurrently, the distribution script adds ~1.9 seconds to the processing time

Experiments: Machinery

- Goal
  - Determine which machinery was most efficient for Cuckoo nodes
- ESXi, vSphere, and XenServer were not used because they host the Cuckoo agents directly, removing one layer of isolation.

- QEMU vs. VMware
  - VMware crashed three times during the processing of the 20,000 samples. Required manual restarts.
  - QEMU ran the 20,000 samples 2.3 times faster than VMware and was more stable.
  - QEMU is also free and open source

Experiments: Best Execution Timeout



- 30,346 samples gathered from VirusShare.com [6] to run experiment
- By 1,132 seconds, 100% of all the groups' calls were completed.
- After 125 seconds all the enhanced groups completed at least 90% of their calls, which became the time used for the Cuckoo timeout

Experiments: Anti-VM

- Goal: Determine the virtualization architecture that best evades detection
- Malware commonly uses anti-VM techniques to determine if the malware is being run on a VM
- Used Pafish [7] tool for identifying sandboxes to test QEMU 2.5.1 and VMware 12

Experiments: Anti-VM cont.

- VM Identifiers found for both
  - CPU vendor
    - QEMU - AuthenticAMD
    - Vmware - GenuineIntel
  - VM CPU for both
    - Checking hypervisor bit in CPUID
  - Under 60 Gib disk, under 2 Gib RAM, and less than 2 CPU cores

Experiments: Anti-VM cont..

- Found fixes for basic anti-VM techniques:
  - Changes in disk/RAM/CPU sizes
- VMware also flagged additional identifiers
  - Registry keys
  - VMware MAC address
  - VMware WMI Win32 bios serial number
- QEMU had less detectable virtualization techniques

Experiment: Hardware Specification

- Goal: Determine way to estimate the amount of RAM and # of CPU cores to select for a Cuckoo node running 20 agents

- On average the agents used a fourth of the RAM they were given
- Cuckoo's processing utility used 2 Gib RAM per parallel process
- QEMU used CPU cores no greater than half the number of agents running
- Processing utility used total CPU cores no more than half the number of parallel processing configured.

- Equation used to estima Cuckoo node:

$$R = C + O_C + P \times 2 + \frac{agent_{count} \times agent_{ram}}{4} \quad (1)$$

- Equation used to estimate uckoo node needed:

$$R = C + O_R + \lceil \frac{agent_{count} + P}{2} \rceil \quad (2)$$

Experiment: Improving Execution

- Goal: Determine if samples behavior would different depending on various hardening configurations

- Added additional software and usage activity to Cuckoo agents to observe variations in activity and sample execution

- Hardening configuration :
  - Added documents
    - "My Documents" has 5 JPGs, 1 txt, 5 PDFs, and 3 data files
    - "My Music" has 3 MP3s
    - "My Pictures" has 6 JPGs and 1 GIF
    - "My Videos" has 4 MP4s
  - New programs:
    - Firefox 38.0.5, Notepad++ v7, VLC 2.2.4, 7-Zip 16.02, Adobe Flash player 10.1.4, Java 6.45

Experiment: Improving Execution cont..

- New Frameworks:
  - Microsoft Visual C++ 2005, 2008, 2010, 2012, 2013, and 2015 redistributable
  - Microsoft .NET 3.5 and 4.6.1 frameworks
- Recent Documents/Programs:
  - All the added documents were opened multiple times. Each new program was run multiple times.
  - Running programs
  - Windows explorer
  - Notepad
  - All update services for new software were disabled

Experiment: Improving Execution cont..

- 10,000 samples randomly selected from VirusShare.com
  - 9,014 ran completely on the base configuration
  - 9,421 ran on the hardened configuration.
- 1,166 samples that did not have a complete run on both base and harden configuration.
  - 363 samples immediately exited with no hooked APIs called (the malware ran properly but decided to exit)
  - 474 had a Cuckoo error unrelated to the sample
  - 329 could not be determined
- 8,834 samples left for analysis

Experiment: Improving Execution cont.

- Hardening differences:
  - 54.98% of the samples exhibited an increased number of unique API calls. The average increase of these samples was 7.88.
  - 60.22% of the samples had more total API calls, 10.61% had fewer, and 29.17% had the same amount.
  - 89.28% of the samples ran for a longer duration.
  - There were no new IP addresses or domains requested. However, some samples made different network calls, though there was no substantial difference as only 2.91% of the malware did so.

Experiment: Improving Execution cont.

Name	Minimum Increase	Maximum Increase	Average Increase	Samples Increased	Samples Decreased
Read Key	-2,744	3,231	309.09	3,982	985
Read File	-41	250	4.62	3,947	716
Write Key	-30	2,516	1.24	1,615	265
Write File	-27	159	1.04	2,103	136
Mutexes	-83	124	0.73	2,537	554
Delete Key	-7	52	0.49	1,116	41
Delete File	-7	328	0.34	1,618	52
Signatures	-6	14	0.19	1,549	295

Lessons Learned: Virtual Architecture

- **Lesson:** Choose an appropriate dynamic analysis platform.

- This project's requirements:
  - System should be open sourced and freely available
  - Available for download and not web-based
  - Project under active development

- PANDA [8] vs. Cuckoo
  - PANDA was not as mature as Cuckoo at the time of consideration.
    - Lacked plugins to convert raw data, robust reporting engines

- Cuckoo-modified (Cuckoo version 1.3) [9]

Lessons Learned: Dynamic Analysis Issues

- **Lesson:** Check and truly understand your analysis.
- Checks revealed:
  - Misunderstanding of calculation done by Cuckoo
    - Duration
  - Errors in Cuckoo
    - Consistence issues

Lessons Learned: Improving Analysis Performance

- **Lesson:** Disable Unnecessary Functions

- Disabled Modules:
  - Memory
  - Dropped files
  - Static modules
  - String modules
- Separation of the processing of results from the submitting of samples

Lessons Learned: Database

- **Lesson:** Use a Database

- Provided a simple way of automating sample processing
- Pros for the use of NoSQL
  - Large volume of data
  - The ability to easily scale out the architecture
- Cons:
  - Size of samples still required for samples to be stored on a shared file system
  - Database changes will require changes for all systems

Future Work

- Automating the submission of samples for Cuckoo generation with REST API

- Expand to support multiple operating systems and versions

- Develop a system to extract information from each Cuckoo sample, which could be used to support machine learning classification and clustering.

Conclusion

- Developed a dynamic analysis platform using Cuckoo sandbox
- Optimize it by performing various experiments
- Documented lessons learned during development.

References

[1] AV-Test. Malware statistics [Online]. https://www.av-test.org/en/statistics/malware/

[2] C. Kruegel, Full system emulation: Achieving successful automated dynamic analysis of evasive malware, in: Proc. BlackHat USA Security Conference, pp. 1–7.

[3] D. Kirat, G. Vigna, C. Kruegel, BareBox: e cient malware analysis on bare-metal, in: Proceedings of the 27th Annual Computer Security Applications Conference, ACM, pp. 403– 412.

[4] T. Kasama, A study on malware analysis leveraging sandbox evasive behaviors, phdthesis, Yokohama National University.

[5] C. Guarnieri, A. Tanasi, J. Bremer, S. Mark, The cuckoo sand-box. URL https://www.cuckoosandbox.org/

[6] J.-M. Roberts. VirusShare [Online]. www.virusshare.com

[7] D. Deepen. Malicious documents leveraging new anti-vm and anti-sandbox techniques [online].

[8] moyix/panda-malrec. [Online] https://github.com/moyix/panda-malrec

[9] spender-sandbox/cuckoo-modified. [Online] https://github.com/spender-sandbox/cuckoo-modified

Questions

DeMarcus Thomas
dmt101@dasi.msstate.edu
Research Engineer
Mississippi State University