View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by AIS Electronic Library (AISeL)

Proceedings of the 53rd Hawaii International Conference on System Sciences | 2020

Container and VM Visualization for Rapid Incident Response

Dr. Jordan Shropshire
Information Systems Department
University of South Alabama
jshropshire@southalabama.edu

Dr. Ryan Benton
Computer Science Department
University of South Alabama
rbenton@southalabama.edu

Abstract

Most cloud security incidents are initially detected by automated monitoring tools. Because they are tuned to minimize the risk of false-negative errors, these tools cast a wide net of suspicion. Depending on the scale of the incident, the automated tools may implicate rather long lists of VMs and containers. Hence, this study proposes a new intermediate step aimed at reducing the number of VMs and containers awaiting forensic investigation.

The proposed method renders two-dimensional visualizations of container contents and virtual machine disk images. The visualizations can be used to fingerprint container / VM contents, pinpoint instances of embedded malware, and find modified code. The proof of concept is evaluated in a pilot study. The results indicate that it shows promise. Implications and future research directions are also described.

1. Introduction

Containers and virtual machines are the building blocks of cloud computer systems. They host the applications and data which collectively provide scalable, on-demand services to users on a global basis. The integrity of containers and virtual machines (VM) is paramount. If the containers and VMs providing a service are not trustworthy, then the cloud is irrelevant.

Because of their pivotal role in cloud computing, containers and VMs are frequently targeted for attack [1]. Attackers may attempt an infiltration in order to steal or corrupt data, install rootkits, or deploy malware. If successful, they can use the container as a springboard for data exfiltration, disrupting hosted services, or launching attacks against other cloud resources [2].

A large number of VMs, containers, and other resources could be implicated in a cloud security event. Automated monitoring systems cast a wide net when it comes to identifying assets which could be involved in an incident. Cloud operations team often end up placing long lists of VMs and containers in quarantine until they can be cleared [3]. The cloud hosting provider has to perform a forensic investigation on every implicated asset. This is often costly and time-consuming. They may even be forced to ask clients resolve the security issues on their own (see Figure 1).

Figure 1. Limits of public cloud forensics

Neither solution is desirable. Cloud clients may not have the ability to perform their own investigation. Furthermore, cloud client still have to pay for hosting but do not enjoy the full use of their quarantined VMs and containers.

Hence, this research proposes a new intermediate step between automated analysis and digital forensic investigation. This step would allow cloud operations teams to perform rapid analysis and adjudication of VMs and containers. This would reduce the number of assets which require forensic analysis.

The proposed new step introduces a new method for out-of-band investigation of containers and VMs. Out-of-band inspection is the process of collecting data from outside of the element being investigated. This reduces the possibility of perturbations of potential evidence. It uses a novel approach for directly accessing the container file or VM disk image and interpreting the contents.

The proposed method renders two-dimensional, colorized visualizations of the bytes contained in the

URI: https://hdl.handle.net/10125/64525
978-0-9981331-3-3
(CC BY-NC-ND 4.0)

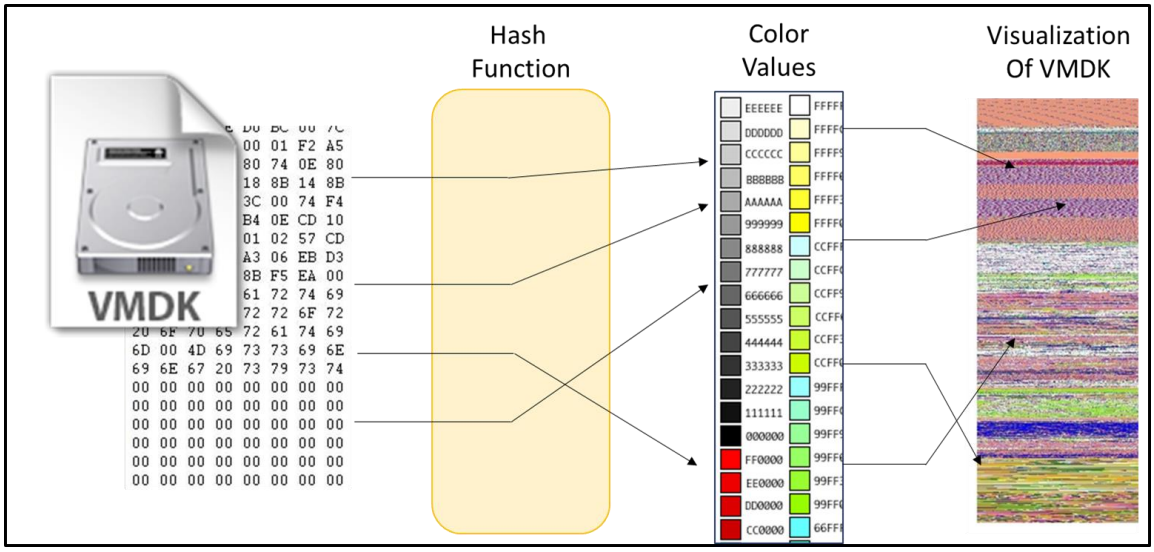

Figure 2. Visualizing container files and VM disks

VM disk or container file. Bytes are read from file and passed through a one-way privacy preserving hash and then assigned an ASCII color based on byte value. They are then transposed onto a PNG file of fixed width and variable length (Figure 2, below). The resulting visualization is intended to be interpreted by members of the operations teams. It provides insights into the contents of containers and VMs.

The proposed step is designed to provide a rapid response to cloud-based incidents. With modest classification thresholds, it could reduce the number of assets requiring forensic analysis. This would reduce operational costs and increase customer satisfaction.

A proof-of-concept test of the proposed new step is evaluated. The results indicated that it shows promise and merits additional development.

The remainder of this manuscript is organized as follows: the next section contains the background. It describes existing methods for forensically analyzing containers and virtual machines and introduces the basics of visualization. The conceptual development section follows the background. It describes the proposed method and the expected benefits. The process of testing and comparison described in the evaluation section. The results of the tests are described next. Implications and future research are then discussed. Finally, concluding comments are shared.

2. Background

This section provides background information on three topics. First, it reviews existing methods for forensically analyzing containers and virtual machines. Second, it surveys current visualization techniques. Third, it reviews related research.

2.1. Container and VM forensics

Both containers and virtual machines provide a means for software isolation, and are an essential components of any cloud based-environment. With containers, the abstraction is performed at the operating system level [4]. Multiple containers can share a single host operating system. Each container has one or more applications and their associated libraries, configuration files, and subdirectory structures [5]. One of the most common container platforms is Docker. Containers have the benefit of being lightweight. They typically provide efficiencies over and above their VM counterparts.

For their part, VMs offer more isolation, greater security, and cross-platform functionality [6]. They make use of hardware-level abstraction. Each VM includes a complete operating system, software, and host applications. Because they replicate the operating system, they are more costly in terms of performance.

Although containers and VMs differ in a number of respects, at a fundamental level they both provide a means for isolating and maintaining software that someone else may own [7]. In this sense, the methods of their forensic analysis tend to overlap. Some of the approaches to accessing and investigating a container are also used on virtual machines [6]. This is most evident in legally-motivated investigations.

Digital forensics were historically driven by the need to support judicial proceedings. A high degree of importance was placed on following process, maintaining a chain of custody, etc. [8]. However,

current analyses may also be aimed at informing inter-organizational processes and workflows. Here the emphasis is on getting enough information to support internal decision making while not violating terms of service. Hence, there are two approaches to container and VM forensics: the first is herein referred to as the legal approach while the second will be called the introspective approach because it uses interrogative techniques.

The legal approach is a formalized process which involves creating demonstrable links between points of interest without modifying the original data. The purpose is to provide evidence which conforms to the practices and standards of a respective legal system [9]. This can involve several different issues, which were partially discussed in O'Shaughnessy and Keane [10], whose discussions dealt with data collection within a cloud environment. Two key issues covered were the chain of custody and multi-jurisdictional-legislation. Chain of custody concerns with who had access to data that will be used as evidence; given a given cloud can span multiple geographical locations and be collocated with other user's data on a rack of servers, legally obtaining the data, without crossing other users' rights can be challenging. This is even more complicated when geographical location crosses jurisdictions

Once that issue is resolved and the raw files are collected and duplicates are obtained, the forensic investigator can proceed at a pace which allows for appropriate diligence and care. The investigator begins by importing a toolset which allows for brute force password cracking [11]. After access is obtained, the file and directory structure is reconstructed onto the desktop of the workstation. The files and data of interest are then gleaned.

This time-tested approach is reliable. It provides a high degree of accuracy in file classification and anomaly detection [12]. However, it is not particularly time-sensitive. This approach is generally performed post-hoc. However, it is not fast enough for real-time operations. Too much time is lost in gaining access to the container or VM. During a cyber event, the operations team needs information as quickly as possible.

The introspective approach consists of a family of techniques which uses introspection in order to gain insights into the processes being executed within [13]. Although this approach was originally developed for virtual machines, parallel techniques and tools can be used within the container space as well [4]. Introspection techniques monitor the runtime processes and applications currently running in a virtual machine or container. They give visibility into the software being executed [14]. Introspection has been used in the past to fingerprint the software running on a container or virtual machine [6].

Introspection can be achieved via a few different means. Dykstra and Sherman [15] developed FROST, which is a set of tools that operated upon OpenStack, a cloud operating system [16]. FROST permits users to retrieve an copy of virtual disks associated with that user's virtual machines; it also checked API requests and OpenStack firewall logs. One drawback is the tools are built on-top of OpenStack and integrated into the Horizon web-based user interface for OpenStack; hence the stack was directly tied to OpenStack.

Another OpenStack-based approach was proposed by Saibharah and Greethaukumari [17] who used existing tools already available within the platform. They built a framework based off of snapshots of both random-access memory and disk images, as well as working through logging systems native to OpenStack. Finally, the researchers extended their framework to incorporate network forensics. The evaluations showed that evidence could be obtained for several different types of attacks on a cloud environment.

Graziano et al. [18], unlike the previous two studies, assumed that the forensics teams did not know what hypervisor was being used. Hence, they exploited physical memory dumps of a given machine to identify (a) if a hypervisor is present and (b), if present, what type of hypervisor was being used. The concept was based upon the idea that hypervisors virtualization of memory changed how that memory is allocated.

Casalicchio and Percibali [19] focused specifically on analyzing containers. They wanted to determine if a battery of tools, that collected CPU and Disk I/O workloads, captured the same information. They determined different tools present similar but not completely equivalent results. Rather than compare tools, Watts et al. [20] examined whether Prometheus [21], an open-source introspection tool, using default metric collection, could be used to determine if a container was infected or not during an investigation. The results indicated it could be, but the authors noted an automated solution, versus manual inspection, would be desirable.

One drawback of the previous efforts was the concept the assumption that the underlying system was sound; that is, that no tampering or inconsistent information had been introduced. Thrope et al. [22] did not make this assumption, rather, they built a virtual machine profiler model and a log auditor to detect and report errors and inconsistencies within the logs; the assumption is attackers could introduce deletions and modifications to the logs. Results indicated that the system could find inconsistencies within the log, indicating that they had been modified.

Shropshire [23] approached the problem of detecting anomalous behavior within a compromised cloud system from a hardware prospective. PowerCheck was developed, which identified discrepancies by comparing the system state parameters with parameters based upon server energy consumption. Tests validated the idea of secondary system measures as legitimate integrity monitors.

Unlike the previous studies. Stelly et al. [24] focused on demonstrating the scalability of forensic analysis of containers. They developed a toolkit entitled SCARF toolkit that was shown to obtain high throughput in processing when tested upon two different clusters running containers.

2.2. Visualization techniques

The field of visualization encompasses a number of techniques for interpreting data. These techniques range in complexity from simple bar charts and line graphs to x-y plots. Even more sophisticated techniques may be used if the data structures and relationships are highly complex. Visualizations can be classified along three dimensions: the data to be visualized, the visualization technique, and the interaction technique [25, 26].

Visualizations may be based on one-dimensional data, two-dimensional data, multi-dimensional data, text or hypertext, graph or relational data, hierarchical data, or audio/visual signals [27]. In general, one dimensional data are typically represented using histograms or visualization similar to pie charts. Two dimensional data may be visualized with scatter plots and line graphs. Multi-dimensional data is often associated with icon, dense-pixel, and geometric transformations. Regardless of data type, some preprocessing is usually performed in order to identify complexities such as missing elements, trends, conversions, and skewing tendencies. Following normalization, the most appropriate visualization technique is selected.

2.3. Related Research

Several studies have used the concept of visualization for security, performance, and integrity monitoring.

Perrig and colleagues developed a method for hash visualization [28]. The visualizations were designed to be used instead of authentication tokens or strings. It was theorized that humans are better equipped to compare images than identify differences in long alphanumeric key strings.

A study by Lee et. al. [29] investigated malicious codes using visual pattern analysis. In this study, a number of malicious software packages were visualized so that pattern matching algorithms could detect repeated features. This study laid the groundwork for a number of follow-up studies in malware analysis.

A study conducted by Nataraj et. al. [30] examined the usefulness of analyzing software binaries as images, with the goal of automatically determining which binaries were malware. In that study, binaries were converted into grey-scale images. Image processing techniques were used to extract texture information, which was then feed into a classifier, which would then determine if the binary was safe or malicious. Various studies expanded upon the classification of images of software by showing texture-based classification was faster than dynamic analysis [31], creating noise-tolerate features from images [32], and finally by creating a full-fledge system based on content-based search [33]. However, all these approaches are based on file-level analysis.

A project conducted by Jain et. al. [34] created a visual image of Android binaries in order to study the effect of optimization and obfuscation techniques; the latter is often used to hide the fact that malicious code has been developed. The inspection was done manually, and has aided by the fact Android binaries are generally structured; hence, color coding techniques based on the structure were utilized to improve understanding. This work was expanded to include predicting what type of obfuscation was being used; accuracy of nearly 90% were achieved [35].

A number of visualizations techniques have been used for network forensics and security [36]. Directed and undirected graphics, radials, and hub-and-spoke networks can be constructed from packet flows to support easier interpretation among human analysts. Additionally, new generations of network visualization incorporate clustering and random walks.

3. Conceptual Development

Most cloud security incidents are initially detected by automated monitoring tools. Because they are generally tuned to minimize the risk of false-negative errors, these tools cast a wide net of suspicion. Depending on the scale of the incident, the automated tools may implicate rather long lists of VMs and containers. Typically, these assets have to wait in quarantine until they can be forensically investigated and cleared. This could anger clients and increase operational costs for the cloud service provider.

Hence, this study proposes a new intermediate step aimed at reducing the number of VMs and containers awaiting investigation. The proposed method uses

visualization techniques to quickly interpret the contents of VMs and containers and clear assets which are unrelated to the incident.

The proposed new method is out-of-band, meaning it is undetectable to the container or VM being investigated. The container or VM is inspected from a peering point within the hypervisor or container engine. Furthermore, there is no impact on container performance.

The proposed new method is designed to balance speed with reliability. Further, it does not rely on brute force password cracking. Additionally, it is highly interactive. An investigator can manipulate the visualization associated with the proposed method in order to make rapid inferences.

The visualization methods support investigation of the functionality of software housed in containers and virtual machines. It enables the investigator to fingerprint the contents of a container, identify anomalous software, and detect content or media which might be illegal.

The workflow is as follows: when suspicious activity is reported or detected within the cloud, monitoring software traces the activity over the network back to a subset of potential offenders. These containers or virtual machines may fall within the same subnet, broadcast domain, or reside in the same physical host or data center.

Each suspicious container or VM is traced from its host back to the location where its container file or VM disk image is permanently stored. Here the proposed visualization techniques would be used to create a PNG image file for each file or disk image. The PNG image contains a two dimensional visualization of the raw contents of the container file or VM disk.

Once the visualizations are collected, members of the operations team perform inspections in order to identify their software contents. The team then looks for anomalous modifications, rootkits, other instances of malware, and illicit content. With little training it is possible to make meaningful inferences from the visualizations. For instance, contrasting visualizations of the same container over time will yield a time-ordering of changes in its contents (See Figure 3).

Assets which are clearly not part of an ongoing cyber incident could be returned to production. This would reduce time-in-quarantine, please clients, and reduce forensic backlogs.

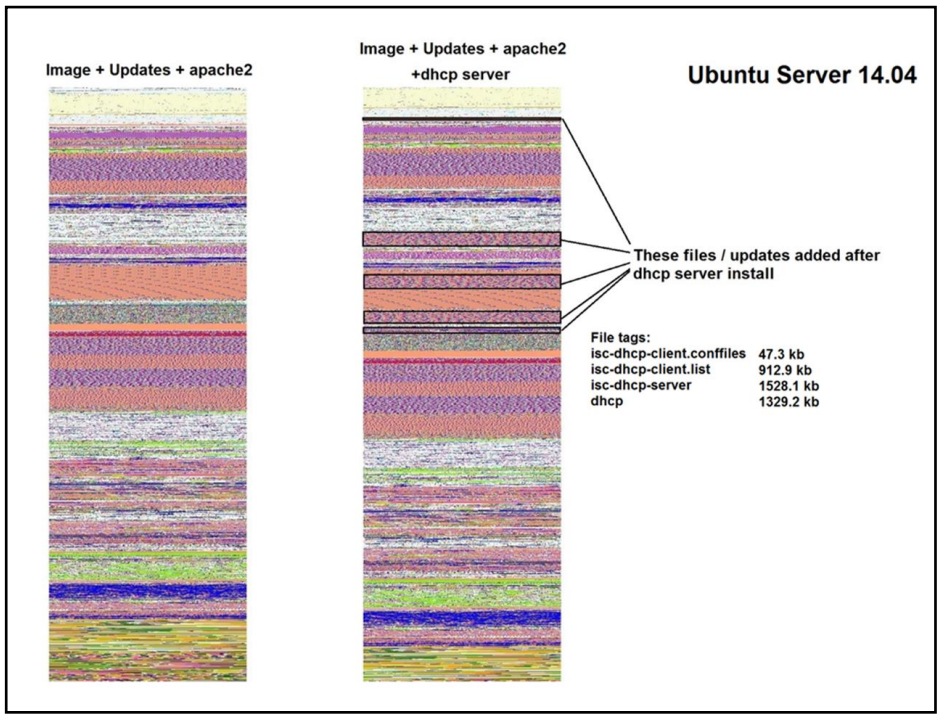

Figure 3. Detecting changes in container / VM contents

As described in the introduction, the process for creating each visualization is as follows: each byte from the container file or VM disk is sampled and run through a one-way privacy-preserving hash function. Each hashed byte is then mapped to 1 of 256 ASCII color values. Each color value is used to shade the corresponding pixel in the visualization PNG file.

Interpretation of the visualizations requires some degree of contextualization. In many cases it is useful

to compare container or VM visualizations against labeled segmentations of other images. This allows the inspector to identify various segments within the visualization of interest. For instance, such comparisons can be made to identify operating systems, libraries and specific applications. Once a software component is identified it can be contrasted against other visualizations of trusted instances of the same component. Any unexplained differences would be considered anomalies.

It is predicted that the proposed visualization method will result in more accurate and timely identification of container/ VM contents. It is further expected to result in more accurate and timely detection of anomalies within identified software components.

4. Evaluation

A proof-of-concept evaluation was performed to assess to the efficacy of the proposed new method. Specifically, the evaluation sought to answer two questions:

- How fast is the proposed method relative to other investigative techniques?
- How reliable is the proposed method relative to other investigative techniques?

4.1. Experimental Groups

Subjects were randomly assigned to either the test group or the control group. Test group subjects used the proposed visualization method to analyze containers and VMs during a simulated cyber event. The proposed visualization method was operationalized for this experiment as a SaaS platform (see Figure 4).

The platform was custom built for this research using a combination of python 3 Anaconda libraries for creating visualizations and JavaScript on the front end for user interaction. It has modules for comparing visualizations, identifying software components within visualizations, and detecting anomalous areas within known software.

The control group used the contemporary method to analyze containers and VMs associated with the same scenario. This group used Kali Linux for brute force password cracking, data extraction, and timeline reconstruction.

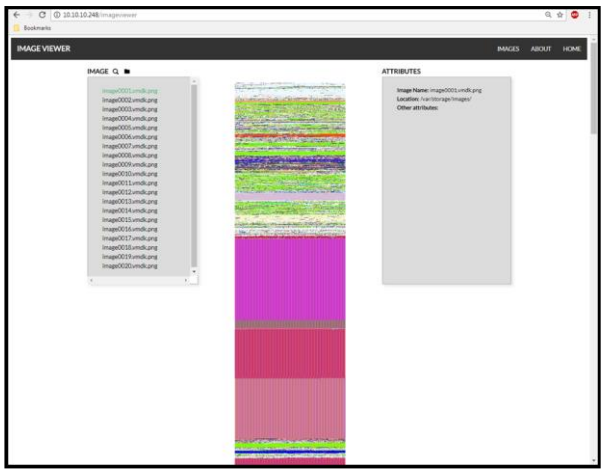

Figure 4. Forensic analysis using the SaaS platform

4.2. Participants

A total of 42 individuals assisted in evaluation of the proposed new methodology. Individuals were either graduate students who had recently completed a course on cloud computing, digital forensics, or operating systems or were recent graduates. To overcome biases, unfair experience, and any pre-existing familiarity with commercial toolsets, only individuals with no prior professional experience in digital forensics were included in the study. Individuals were evenly distributed between the control group and the test group. Each subject completed a 45 minute online training session which described how to use the forensic tool associated with their group. Subjects then completed a short online quiz to ensure their familiarity with the toolset.

4.3. Procedure

The purpose of the evaluation is to assess the relative speed and accuracy of the proposed visualization method. Each subject was asked to assess a large number of cloud-based assets during a limited period of time. As previously indicated, half of the subjects used the visualization method and half used traditional techniques.

The analysis includes fingerprinting the software in the suspicious containers / VMs, identifying anomalous software, and correctly classifying individual instances as benign or infected.

Subjects logged into a subset of a private, IaaS (Infrastructure-as-a-Service) cloud which was constructed for the purposes of this experiment. Each subset contained the analytical tool associated with the subject's assigned and replications of the same

container and VM instances. During their analysis, participants recorded their findings conclusions for each container or VM instance they analyzed within a web-based form. The form consisted for 30 sections – one for each container or VM. There was a space to record the software inventory and denote the absence or presence of anomalous code for each instance.

4.4. Means of Comparison

Some 15 Docker containers and 15 ESX-based VMs were included. The 15 containers were clones of a single MEAN (mongoDB, express, angular, node.js) stack web application. The MEAN stack was chosen because although it is widely used, it is of sufficient complexity to warrant careful forensic analysis. The latest stable version of each of the MEAN stack elements was used in the image. Of the 15 containers 5 were infected with a rootkit which consists of modified code in the node.js script and a compressed key string in the angular library (see Figure 5).

The 15 VMs were clones of a single LAMP (Linux, Apache, MySQL, PHP) stack web application. The LAMP stack was selected because it provides a balance between familiarity and complexity. The Ubuntu 18.04 Linux flavor was used, along with the latest stable versions of the other elements. (The study participants all reporting having at least an introductory level of Linux proficiency.) The stack was sufficiently large enough to require a careful investigation. Of the 15 VMs 8 were infected with a rootkit which modified code within the glibc library and stored compressed malware in the MySQL database.

Individuals were scored across two key metrics: software fingerprinting accuracy and adjudication accuracy. Fingerprint accuracy is defined as the correct classification of each software component within an instance. One point was awarded for correctly identifying each software component. For instance, for a container, one point would be awarded for identifying each MEAN component (and the node.js code base) for a total of five points per container.

Similarly, seven points were available for each of the main components of a LAMP VM. Adjudication accuracy is the accuracy with which one correctly classifies a container or VM as benign or infected. One point is awarded for each correct classification while a point is deducted for making an incorrect classification. Overall, a total of 180 software fingerprinting points and 30 adjudication points were available for each candidate.

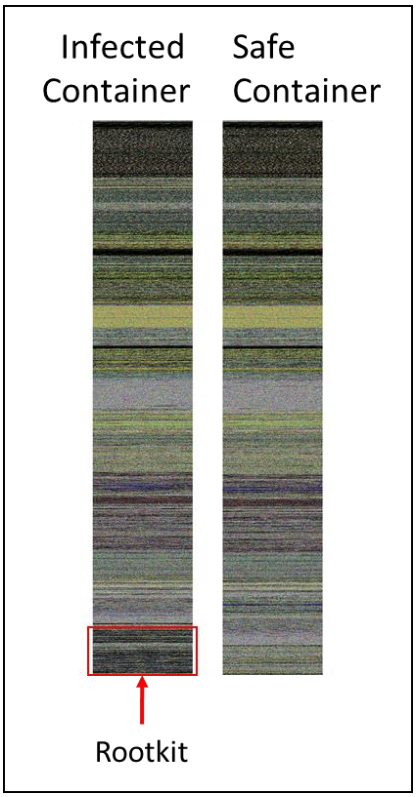

Figure 5. Comparing Container Visualizations

5. Results

Following the completion of the tests, the demographic data and test results were imputed into a spreadsheet for further analysis. The demographics indicate that the subjects skewed towards a younger age and gender skewed towards male. These data are illustrated in Table 1 (below). To compare the performance of the proposed forensic method against the standard method, a series of T-tests of significant differences were completed.

The first test compared relative performance at fingerprinting. The results of this test are shown in Table 2 (below). The results indicate that the test group earned significantly more points for fingerprinting than the control group. This is likely because once the individuals in the test group learned to visually recognize specific software components in the first few visualizations they only needed to procure visualizations of the other instances to make quick comparisons. On average, members of the test group blueprinted 12 containers and 4 VM instances (for an average of 88 points) while the control group inspected 5 containers and 1 VM (37 points on average).

It appeared that the control group did not suffer in terms of fingerprinting accuracy. Of the images they analyzed, their accuracy was either on par or above the level of the test group. However, they were limited in their ability to project their acquired insights across the domain. The traditional approach is costly in terms of the time consumed acquiring access credentials for each instance. Further, it does not provide a single snapshot of the software contents. This has to be determined manually for each instance.

A second t-test of significant differences was conducted to assess adjudication accuracy (see Table 3). This is the extent to which a container or a VM is correctly classified as containing suspicious software. Although there were significant differences, the gap was somewhat less dramatic. The test group earned an average 12 points while the control group earned 4 points on average.

Members of the test groups did not have to wait for access the containers or VMs. Hence they were able to inspect more instances in the same period of time. On a per-instance basis, it appears that the accuracy rates are relatively equivalent between groups. The test and control groups averaged a classification rate of approximately 68% and 71%, respectively. Neither approach is conclusively more accurate than the other.

To sum, the results of the tests indicate that the proposed visualization method outperforms the contemporary methods in terms of the speed and accuracy of software inventorying and adjudication.

Age	18-24	25-29	30-39	40-49	50-59	60+
	19	18	4	1	0	0
Gender	Male	Female	Other			
	27	15	0			
Ethnicity	White	Black	Hispanic	Asian	Am. Indian	Other
	28	3	0	11	0	0

Table 1. Demographics

	Levene's Test		t-test for Equality of Means						
								95% Confidence Interval	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Diff.	Std. Err. Diff.	Lower	Upper
Equal variance assumed	9.32	.000	4.41	40	.000	51	6	49.13	52.87
Equal variance not assumed			5.01	38.14	.000	51	6	49.98	53.54

Table 2. t-Test of Significant Differences at Fingerprinting

	Levene's Test		t-test for Equality of Means						
								95% Confidence Interval	
	F	Sig.	t	df	Sig. (2-tailed)	Mean Diff.	Std. Err. Diff.	Lower	Upper
Equal variance assumed	1.332	.005	2.98	40	.005	8	2.01	7.38	8.62
Equal variance not assumed			3.01	38.72	.005	8	2.01	7.42	8.68

Table 3. t-Test of Significant Differences at Adjudication

6. Implications and future research

The proof-of-concept test described in the previous sections yields several implications. It appears that subjects using the visualization method could adjudicate more VMs and containers than subjects using traditional methods in the same time period without a significant increase in errors. During a massive cloud security incident it would be beneficial to use the proposed method in order to reduce backlogs of assets awaiting forensic analysis.

Future research should focus on exploring the relationship between the granularity of the visualization, analytical speed, and classification accuracy. It is expected that down-sampled images allow for faster analysis although they increase the likelihood that subtle details will be missed. Further, future research should focus on automating the process of software blueprinting. Machine learning methods such as near-neighbor could be useful for classifying installed.

7. Conclusions

It is concluded that the proposed method of rapid incident response could of significant value when time is of short supply and/or a large quantity of containers or VMs must be evaluated. An additional analytical step between automated incident detection and forensic investigation could save considerable time and effort if it reduces investigation backlogs.

The proposed method provides an out-of-band approach to investigating the contents of hosted instances. It uses a new visualization technique to display data which might be otherwise difficult to understand. In this case, the data represents raw bytes taken from cloud storage. This is a novel viewpoint which users could not ordinarily access or interpret.

The proof-of-concept tests suggest that the proposed new step merits additional testing and development. Using the visualization tools, Individuals were able to successfully detect malware approximately 70% of the time. With more research and development this could rise even higher. Future combinations of visualizations with more advanced, intelligent forensics will likely provide even better results for cloud computer systems.

8. Conclusions

This work is supported in part by the National Science Foundation award IIP-1740434 and in part by the Industry Advisory Board of the Center for Advanced Research in Forensic Science.

9. References

[1] Kandukuri, B., R. Paturi, and A. Rakshit, Cloud Security Issues, in 2009 IEEE International Conference on Services Computing. 2009: Bangalore, IN.

[2] Jansen, W., Cloud Hooks: Security and Privacy Issues in Cloud Computing, in 44th Hawaii International Conference on System Sciences. 2011: Kauai, HI.

[3] Aikat, J., et al., Rethinking Security in the Era of Cloud Computing. IEEE Security & Privacy, 2017. 15(3): p. 60-69.

[4] Watts, T., et al., Insight from a Docker Container Introspection, in Proceedings of the 52nd Hawaii International Conference on System Sciences. 2019: Maui, HI.

[5] Xavier, M., et al., Performance Evaluation of Container-Based Virtualization for High Performance Computing Environments, in 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. 2013: Belfast, UK.

[6] Rosenblum, M. and T. Garfinkel, Virtual Machine Monitors: Current Technology and Future Trends. Computer, 2005. 38(5): p. 39-47.

[7] Felter, W., et al., An Updated Performance Comparison of Virtual Machines and Linux Containers, in 2015 IEEE International Symposium on Performance Analysis of Systems and Software 2015: Philadelphia, PA.

[8] Olsson, J. and M. Boldt, Computer Forensic Timeline Visualization Tool. Digital Investigation, 2009. 6(1): p. 78-87.

[9] Osborne, G., H. Thinyane, and J. Slay, Visualizing Information in Digital Forensics, in 8th International Conference on Digital Forensics. 2012: Pretoria SA.

[10] O'Shaughnessy, S. and A. Keane, Impact of cloud computing on digital forensic investigations, in IFIP International Conference on Digital Forensics, pp 291-303, Jan. 2013.

[11] Palomo, E., et al., Visualisation Of Network Forensics Traffic Data with a Self-organising Map for Qualitative Features, in Proceedings of International Joint Conference on Neural Networks. 2011: San Jose, CA. p. 1740-1247.

[12] Vlastos, E. and A. Patel, An Open Source Forensic Tool to Visualize Digital Evidence. Computer Standards & Interfaces, 2007. 30(6): p. 614-625.

[13] Garfinkel, T. and M. Rosenblum, A Virtual Machine Introspection Based Architecture for Intrusion Detection, in Proceedings of the Network and Distributed System Security Symposium. 2003: San Diego, CA.

[14] Choo, K., C. Esposito, and A. Castiglione, Evidence and Forensics in the Cloud: Challenges and Future Research Directions. IEEE Cloud Computing, 2017. 4(3): p. 14-19.

[15] Dykstra, J. and A.T. Sherman, Design and Implementation of FROST: Digital Forensic Tools for the OpenStack Cloud Computing Platform. Digital Investigation, 2013. 10, pp. S87-S95.

[16] What is OpenStack?, accessed Sep 21, 2019 https://www.openstack.org/software/.

[17] Saibharath, S. and G. Geethakumari, Design and Implementation of a forensic framework for Cloud in OpenStack cloud platform, in International Conference on Advances in Computing, Communications and Informatics, pp 645-650, Sep 2014.

[18] Graziano, M., A. Lanzi, and D. Balzarotti, Hypervisor Memory Forensics, in International Workshop on Recent Advances in Intrusion Detection, pp. 21-40, Oct. 2013.

[19] Casalicchio, E. and V. Perciballi, Measuring Docker Performance: What a Mess!!!, in Proceedings of the ACM/SPEC on International Conference on Performance Engineering Companion, pp. 11-16, Apr 2017.

[20] Watts, T., R. G. Benton, W. B. Glisson, and J. Shropshire, Insight from a Docker Container Introspection, in Hawaii International Conference on System Sciences, pp. 7194-7203, Jan 2019.

[21] Prometheus - Monitoring system & time series database, accessed 21 Sep 2019, https://prometheus.io/.

[22] Thorpe, S., I. Ray, T. Grandison, A. Barbir, and R. France, Hypervisor event logs as a source of consistent virtual machine evidence for forensic cloud investigations, in IFIP Annual Conference on Data and Applications Security and Privacy. Pp. 97-112, July 2013.

[23] Shropshire, J, Securing Cloud Infrastructure: Unobtrusive Techniques for Detecting Hypervisor Compromise, in International Conference on Cloud Security and Management, pp. 86-99 , 2015.

[24] Stelly, C. and V. Roussev, SCARF: A container-based approach to cloud-scale digital forensic processing. Digital Investigation, 2017. 22, p. S39-S47.

[25] Hansen, C. and C. Johnson, The Visualization Handbook, ed. E. Butterworth-Heinemann. 2005, Burlington, MA.

[26] Munzner, T., Visualization Analysis and Design. 2014, New York, NY: CRC Press.

[27] Marschner, S. and P. Shirley, Fundamentals of Computer Graphics. 4th ed. 2-15, Boca Raton: Taylor & Francis.

[28] Perrig, A. and D. Song, Hash visualization: A new technique to improve real-world security, in Workshop on Cryptographic Techniques and E-Commerce. 1999: Hong Kong.

[29] Lee, D., et al., A Study on Malicious Codes Pattern Analysis Using Visualization, in 2011 International Conference on Information Science and Applications. 2011: Jeju Island, South Korea.

[20] Nataraj, L., S. Karthikeyan, G. Jacob, and B.S. Manjunath, Malware Images: Visualization and Automatic Classification in International Symposium on Visualization for Cyber Security (VizSec), 7 pages, Jul. 2011.

[31] Nataraj, L., V. Yegneswaran, P. Porras, and J. Zhand, A comparative assessment of malware classification using binary texture analysis and dynamic analysis in ACM Workshop on Security and Artificial Intelligence, pp 21 – 30, Oct 2011.

[32] Kirat, D., L. Nataraj, G. Vigna, and B. S. Manjunath, SigMal: A Static Signal Processing Based Malware Triage in Annual Computer Security Applications Conference, pp 89-98, Dec 2013.

[33] Nataraj, L., D. Kirat, B. S. Manjunath and G. Vigna, SARVAM: Search And RetrieVAl of Malware in Annual Computer Security Applications Conference (ACSAC) Workshop on Next Generation Malware Attacks and Defense (NGMAD), 9 pages, Dec. 2013.

[34] Jain, A., H. Gonzalez, and N. Stakhanova, Enriching reverse engineering through visual exploration of Android binaries, in Program Protection and Reverse Engineering Workshop, 9 pages, Dec. 2015.

[35] Kaur, R., Y. Ning, H. Gonzalez, and N. Stakhanova, Unmasking Android Obfuscation Tools Using Spatial Analysis, in Annual Conference on Privacy, Security and trust, 10 pages, Nov. 2018.

[36] Shiravi, H., A. Shiravi, and A. Ghorbani, A Survey of Visualization Systems for Network Security. IEEE Transactions on Visualization and Computer Graphics, 2011. 18(8): p. 1313 - 1329.