Assessing software integrity of virtual appliances through software whitelists: Is it any good?

Jun Ho Huh, Mirko Montanari, Derek Dagit, Rakesh B. Bobba, Dong Wook Kim, and Roy H Campbell
University of Illinois at Urbana-Champaign, Illinois, USA
{jhhuh, mmontan2, dagit, rbobba, kim628, rhc}@illinois.edu

Yoonjoo Choi
Dartmouth College, Hanover, New Hampshire, USA
yoonjoo@cs.dartmouth.edu

Abstract

Virtual appliances (VAs) are ready-to-use virtual machine images that are configured for specific purposes in Infrastructure-as-a-Service (IaaS) clouds. This paper evaluates the integrity of software packages installed on real-world VAs through the use of a software whitelist-based framework. Analysis of 151 Amazon VAs using this framework shows that there is significant variance in the software integrity across VAs and that about 9% of real-world VAs have significant numbers of software packages that contain unknown files, making them potentially untrusted. Virus scanners flagged just half of the VAs in that 9% as malicious, though, demonstrating that virus scanning alone is not sufficient to help users select a trustable VA.

1 Introduction

Many users enjoy the flexibility that comes with Infrastructure-as-a-Service (IaaS) cloud model, where users can build and configure virtual computing infrastructure by renting computing platform resources (e.g., Amazon EC2) in the form of virtual machines. Virtual machine (VM) disk images that are preconfigured with the necessary software to support specific workflows of functions are known as *virtual appliances* (VAs), and services providing repositories of VAs are referred to as *appliance stores*. The benefits of this publisher-consumer model are clear: consumers enjoy the convenience of downloading a VA that suits their needs and launching a service quickly; and publishers can get paid for providing the VA. This marketplace, however, lacks adequate mechanisms that allow users to a priori assess whether a specific VA is correctly equipped with the software packages that it claims to contain. An irresponsible publisher could publish an incomplete VA that is missing integral software packages that it claims to have. Some software packages might be partially installed (i.e., missing critical files) or have files that are modified (i.e., different from the version provided by the software vendor). VAs with such partially installed or modified packages will not meet the expectations of VA consumers, who would expect the VAs to be configured with all the packages claimed by publishers, and that those packages are unmodified (unless otherwise noted).

This paper presents an empirical study on the integrity of software packages in real-world VAs, assessed using a whitelist-based framework, showing that there is significant variance in the software integrity of software packages across VAs. Our findings show that about 9% of the evaluated real-world VAs have significant software integrity issues and would probably not meet consumers' expectations. Previous research has identified some of the quality assurance and security issues with the publisher and consumer model as well (*e.g.* [3, 2]) but the solutions focused on blacklisting approaches such as scanning for viruses and filtering to removing unwanted files.

2 Analysis of real-world Virtual Appliances

Our empirical analysis sheds light on the integrity of software packages in real-world VAs through a software whitelisting-based technique. By looking at the variances in the results, we gauge how useful a priori software integrity verification results would be for consumers in selecting well installed VAs. We analyzed 151 randomly picked Amazon VAs as described below.

Methodology We sampled the content of the Amazon market by randomly selecting publicly available Amazon Machine Images (AMIs). Our random sample is composed of 151 images from an estimated pool of 2,300 valid `rpm`-based AMIs available in the Amazon US-east zone. To verify the representativeness of the samples, we randomly shuffled the 151 VAs and divided them into three subgroups; then demonstrated that those three groups share similar characteristics with respect to the properties of interest (number of files, number of unverified files, and number of software packages).

Using a combination of shell and python scripts, we constructed a whitelist-based tool for rating integrity of software packages in VAs. We take advantage of the fact that VAs are often generated from trusted base images made available from the repository provider (e.g., Amazon Linux AMIs [1]) to speed up the analysis. Given a trusted base image and a derived VA, the tool first generates the checksums by hashing all of the files in both images. It compares the hash values between the two images and creates a list of *added* files, *modified* files, and *deleted* files. The tool then checks the added, modified and deleted files against the whitelist that we created and marks the files that are unverified or missing. Then, by figuring out which software package each unverified/missing file belongs to, the tool computes the "integrity score" of the installed software using the rules described below:

- integrity score ③—the software has *no* unverified or missing files, with the exception of configuration files;
- integrity score ②—the software has no unverified or missing critical files, but may have unverified or missing *non critical* files;
- integrity score ①—the software has unverified or missing *critical* files.

For this study, we treat all executables, source files, web pages, and image files as "critical," and compressed or other data files as "non critical." Installed software that has an integrity score of ③ is considered fully verified and integrity-protected, falling under the "clean or high-integrity installation" category. Software that has a score of ② has only non critical missing or unverified files; hence, it is considered partially verified and falls under the "partially clean or medium-integrity" category. Score of ① represents the "modified or low-integrity" category as such software may have critical files missing or unverified.

As the final step, the tool generates a signed "verification report" containing the scores.

VA Classification I: Percentage of Unverified/Missing Files Our intuition was that the number of unverified/missing files would indicate, to some degree, the integrity level of software in a VA. As the first step to study their characteristics, we looked for and found a correlation between the number of unverified/missing files and the percentage (Pearson's: 0.84). Using that correlation, we classified the VAs into the following three "Integrity Level Groups" (ILG) to help demonstrate common VA characteristics:

Table 1. Average number of software packages to which the three integrity scores have been assigned

	Avg. # unverified/missing		*Avg. # integrity scores*		
	non critical files	*critical files*	③	②	①
ILG A	23	6	426	1	0
ILG B	122	43	416	2	1
ILG C	917	954	346	20	45

- "ILG A"—44 VAs are in this group, and they have less than 0.1% of unverified/missing files;
- "ILG B"—59 VAs are part of this group, and they have 0.1-1% of unverified/missing files;
- "ILG C"—48 VAs in this group have $> 1\%$ of unverified/missing files.

Table 1 contains the number of software packages with each integrity score. We show "truncated average" values to account for the high variance in the results. The average number of unverified/missing files is significantly higher in ILG C (1871) than in ILGs A (29) and B (165). The table indicates that the VAs in ILG C have a relatively larger number of software packages with low integrity scores (② and ①), while most of the software packages installed on VAs in ILG A have an integrity score of ③. The total number of packages averages around 420 in all three groups. About 99% of the software packages in the 44 VAs in ILG A are clean, and the numbers are not too different for the 59 VAs in ILG B. Both groups have a small number of low-integrity and medium-integrity packages. There is a big jump, however, as we reach ILG C, whose VAs have, on average, 45 low-integrity packages and 20 medium-integrity packages. Those are about 11% and 4.8% of the total number of packages, respectively. We note this as our first key finding.

Finding 1: across the VAs, there is high variance in the number of unverified/missing files and the number of low-integrity and medium-integrity software packages.

Interestingly, in most VAs, both the unverified/missing critical files and non critical files are *concentrated* in a small number of packages, and there is no strong correlation between the number of unverified/missing files and the number of packages they influence. There are 12 or so VAs, however, for which the unverified/missing files do spread out across a large number of packages, and so we investigate these further in Section 2.

VA Classification II: Percentage of Low-Integrity Packages In this second part of the analysis we further investigate the outliers identified previously, which have noticeably larger number of partially clean and modified packages. We do so by forming VA clusters based on the percentages of packages given integrity scores ① and ②.

Table 2. Characteristics of the VA clusters

	# VAs	*Avg. % of integrity scores*			*Avg. # of integrity scores*		
		③	②	①	③	②	①
Cluster 1	137	99%	0.5%	0.3%	418	2	1
Cluster 2	14	43%	17%	40%	205	77	171

The k-means clustering method—an unsupervised learning algorithm—is used to identify the clusters ($k = 2$).

Characteristics of the VA Clusters: Table 2 summarizes the characteristics of the two VA clusters, showing that the 14 VAs in Cluster 2 have significantly high percentage of packages with scores ① and ②; the absolute numbers are also very high, averaging 171 and 77 for scores ① and ②, respectively. Only 43% of the packages are cleanly installed (high-integrity). The percentage of the unverified/missing critical files is high too, averaging 73%. In contrast, for the VAs in Cluster 1, 99% of the packages are cleanly installed, and the percentage of unverified/missing critical files averages only 22%. Considering that our samples are a representative set, here is our second key finding:

Finding 2: About 9% of the VAs have a significant portion of low-integrity and medium-integrity packages installed.

Virus scan results showed that 41 of the unverified files across the VAs are potentially malicious, infecting 7 of them 14 VAs. The VAs are from different publishers, were built to support different functions, and do not share common base images. What is most worrying about those 14 VAs is that none of them, in their name or image-description, mention anything about software customization or modification efforts. Just looking at the VA descriptions, they all appear to have only the cleanly installed, standard packages.

Whitelist Size To evaluate the scalability of the assessment approach, we analyzed the size of the whitelist that we created to verify our sample of 151 VAs and the corresponding 47 base images, showing how the size changes as the number of VAs grows. Figure 1 plots the percentage of the VAs used against the percentage of the whitelist coverage (i.e., how much of the total whitelist was created). To get the values, we randomly shuffled all the VAs and incrementally generated the whitelist, adding the per-VA whitelist entries (the hashes used to verify a specific VA) to a global whitelist while removing any duplicate entries; the size of the global whitelist was recorded after each step. We repeated this process 10 times to get the average size for each incremental step, and used these average values to calculate the percentage values. Intuitively, we see that the whitelist coverage percentage grows as the number of VAs used to construct the whitelist goes up. It does so more rapidly at first, but then slows down progressively because some software packages are installed on multiple VAs, and thus do not add new entries. After 20% of the VAs are used, about 40% of the total global whitelist is already constructed.

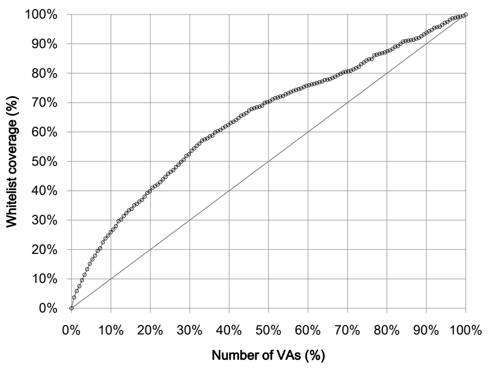

Figure 1. Percentage of the VAs used vs. percentage of the whitelist coverage

3. Conclusions

Our study shows that a significant portion (9%) of real-world VAs contain software packages that are modified. Such VAs are offered to consumers without any indication on the publishers' customization efforts and appear to have standard installations. When a consumer pays for a VA to use, she has the right to know that a VA is configured properly and meets her software integrity expectations. To that end, Findings 1 and 2 demonstrate the need for a priori software integrity assessment. Our software integrity assessment reports would help a consumer avoid VAs that are configured badly and choose ones with high-integrity packages. Further, our scalability analysis shows that the rate at which additions are made to the whitelist will decrease over time because of installation of a large number of packages on multiple VAs, allowing the whitelists to gain authority.

Acknowledgement

This material is based on research sponsored in part by the Air Force Research Laboratory and the Air Force Office of Scientific Research, under agreement number FA8750-11-2-0084. This work has also been partially supported by the Boeing Company.

References

[1] Amazon Elastic Compute Cloud. `http://aws.amazon.com/ec2/`.

[2] Bugiel, Sven and Nürnberger, Stefan and Pöppelmann, Thomas and Sadeghi, Ahmad-Reza and Schneider, Thomas. AmazonIA: when elasticity snaps back. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, pages 389–400, New York, NY, USA, 2011. ACM.

[3] J. Wei, X. Zhang, G. Ammons, V. Bala, and P. Ning. Managing security of virtual machine images in a cloud environment. In *Proceedings of the 2009 ACM Workshop on Cloud Computing Security*, CCSW '09, pages 91–96, New York, NY, USA, 2009. ACM.