Virtual Machine vs. Container Security

Andre Moruga, Director, Program Management, Virtuozzo

September 2016

Abstract

There is a common perception that OS/system container-based virtualization solutions are less secure than hypervisor-based solutions. Recognizing the complexity of the issue, proponents of the idea usually add "other things being equal" or a similar clause to their statements.

With this clause in mind, the issue becomes more difficult to argue. OS container virtualization does, by design, expose a unique area of potential security attacks that do not exist (or at least they differ significantly) in hypervisor-based solutions. For example, shared kernel vulnerabilities from the guest system can be exploited to gain host privileges. The truth is, however, that the "other things being equal" clause does not apply. Security is a complex collaboration of software, hardware, configuration, people, and processes. The differences between OS/system containers and hypervisors are large enough to introduce a number of factors, which when applied to real-world security threats, make the escalation risk of guest privileges negligibly insignificant when compared to other risks. All things considered, a container based infrastructure is likely to be more secure than hypervisors, because it is simpler to keep secure.

This document describes the key factors for comparing hypervisor- and container-based solutions and discusses the instruments used to assess their security.
Introduction

To compare hypervisor and container security issues, we should take a closer look at each of their unique vulnerabilities.

Hypervisor solutions provide “guest tools”: a set of drivers and applications installed inside the virtual machine to enhance virtualization performance and enable additional management features. Guest tools have a back-end in the host system/hypervisor which communicates with those tools. Bugs on both sides of that communication channel can be quite severe, opening possibilities for Denial of Service (DoS) and privilege escalation attacks. This is not just a hypothetical threat. Between 2014 and 2016, more than 20 security advisories were published for QEMU (the de-facto standard hypervisor for KVM), fixing DoS, code execution and privilege escalation issues.

Container security vulnerabilities come from the operating system kernel, rather than the hypervisor. The most common container implementations use the Linux kernel. For freshly written code, assuming similar development skills, the number of bugs and vulnerabilities discovered would be roughly proportional to the amount of code. Linux kernel is much larger than a typical hypervisor code (guest tools included), so the potential number of vulnerabilities and bugs is also larger. Linux, however, is not a new product; it’s been around for more than 25 years—much longer than most hypervisors. Some of the world’s top engineers and security professionals work on and review its code. No other hypervisor gets the same level of support. So, although larger in numbers, the quality of Linux kernel is also higher than most hypervisors.

Security, for most practical applications, is not an absolute. While the reader may initially disagree, here are some arguments. There is an easy way to dramatically improve web server security. Being disconnected from the network and power and physically protected by a banking deposit box, chances of a breach on such a server are close to zero. But that level of protection is impractical and defeats the very purpose of a web server.

A more practical approach is deploying a firewall in front of a web server, requiring the user’s authentication with government issued IDs and blocking any network connectivity before the user is authenticated.

Still, this is something that greatly compromises usability, and most website owners will find it unacceptable. So we can conclude that most of the users are not willing to trade much in terms of usability for a bit of extra security.
As an alternative approach to dramatically improving the security, a world renowned expert in security can be hired for a thorough security audit of the whole system. Implementing the recommendations of such an expert will greatly improve the security of the server in question. It's also likely that some website owners may take exactly this course of action.

However, the rest of us will also find this approach impractical due to the high costs associated with such a review.

Therefore, for the majority of users, security is a compromise between multiple factors, with cost, usability and impact of a potential breach being the most important. For practical purposes, the goal should be to achieve maximum security while balancing the other factors. It also makes sense to implement improvements starting with the "low hanging fruits", addressing the most common risks and vulnerabilities which are not prohibitively expensive to fix and have little impact on the usability of the solution.

In this document, we're focusing specifically on service provider/public cloud scenarios. However, key characteristics of that environment can also be applicable to many other organizations:

- The platform is deployed and maintained by a central entity (the service provider) and used by users
- Although the vast majority of users are benevolent, the platform owner must assume that users may have malicious intents
- When assigned the responsibility of applying security patches to their own system, users will likely have low compliance. So centrally managing the patches is a great way to improve compliance.
- Data breaches, even when not at fault of the service provider, still have significant negative impact on the business, which may include brand damage, lawsuits, SLA related refunds, and loss of customers.
Background

Surface of attacks

According to WhiteHat security, the following website vulnerabilities were most likely to be discovered in 2014:

- Insufficient transport layer protection
- Information leakage
- Cross-site scripting
- Brute force attacks
- Content spoofing (also known as content injection)
- Cross-site request forgery

The least common vulnerability in this list (the cross-site request forgery) was at a 24% probability (meaning there was a 24% chance that at least some components of a randomly chosen website were vulnerable to that attack), while chances to discover an insufficient transport layer protection in comparison were a whopping 70%.

For a hacker determined to breach a website's security, this would likely be a list to begin with, and there is a reason in that. Actually, multiple reasons.

First, a hacker would want to spend minimal effort to achieve their goal, with a maximum probability of achieving it - thus going over their own "low hanging fruits" list. And nothing platform specific is likely going to be on the list here, as web applications and the components they use usually offer plenty of easier-to-exploit vulnerabilities (not necessary the given website, but websites in general).

Second, our "average" hacker is likely to go through a list of known CVE-recorded vulnerabilities (CVE stands for Common Vulnerabilities and Exposures, see https://cve.mitre.org/) for the simple reason that the amount of effort and skills required to discover and exploit a new security hole are incomparably higher than running a conveniently packaged software that simply goes through most common CVE advisories one by one.

This is one of the reasons why it is extremely important to have latest security patches applied, and we'll touch more on this topic in the next chapter.

However, for the sake of argument's sake, let's assume the worst case scenario - that the website defenses are impenetrable, and the hacker is left with nothing but to attempt to break in through the platform.
This would mean the following:

1. The hacker would need to locate another application or website which shares the same physical server but runs in another container. If the provider is of a decent size (let's say more than a dozen servers) and smartly designed one network (which does not easily expose the physical server IP to the outside), the task is doomed. Same thing with the idea of legitimately buying another container-based server from the same provider - as the purchaser does not control which physical host it will be placed on, the chances of getting on the same hardware where the victim container is located are pretty slim (and a smart provider would never put trial/non-paying customers on a production server shared with paying customers).

2. Still, to consider this possibility, let's assume the hacker somehow gained access to a neighboring container. There is just one thing left before the victim's data is breached - privilege elevation to gain host access.

We rule out a targeted attack that originates from an original "insider" (legitimate neighboring tenant), as such an attack would not have a true economical motivation (since chances that both the attacker and their intended victim were originally on the same physical server are nearly non-existent). So, cases other than an attempt at vandalism (which is unlikely to be as dangerous as true data breach) can be ruled out.

Now let's see how complicated it can be to gain host privileges from a container.

Over the history of the development of Virtuozzo, there has never been a case of a vulnerability discovered which allowed the repeatable escalation of privilege to the host root. There was an issue CVE-2015-2925 which came pretty close - when exploited it could lead to file access outside the container. However, that issue only affected certain OpenVZ deployments (based on the simfs file system) and did not affect commercially supported Virtuozzo or ploop based OpenVZ containers. No recorded case of exploiting this vulnerability in production is known to the Virtuozzo team, but it's not impossible that it happened.

In sum, including this issue, the 2015 statistics looked like this:

- 1 Container file system escape, affecting certain configurations which are unsupported in the commercial product (described above)
- 18 host crashes, hangs or Denial of Service (DoS) vulnerabilities
- 3 potential privilege escalations (when no reliably working exploit was available)
- 3 memory leaks or similar problems (when a user from inside the container could access memory regions belonging to another user)
Just for reference, here is a comparison to Red Hat Enterprise Linux 6 QEMU security advisories over the same period of time, with 42 total reported issues:

- 20 of those issues did not affect Red Hat Enterprise Linux 6
- 14 were closed as wontfix since the RHEL team decided they weren’t a high enough risk of exploit

Out of 7 confirmed issues:

- 1 remote DoS attack vulnerability
- 5 local DoS attack vulnerabilities fixed
- 1 information leak (ability to gain unauthorized data access)
- 1 issue was still processing at the time of writing

As one can see, while the total number of Linux kernel security issues is larger than the total number of QEMU security issues, both are negligible when compared to a typical web application stack, and much more difficult to exploit.

Virtuozzo, as a platform vendor that delivers system containers (in addition to virtual machines), has a policy that security issues with privilege escalation vulnerabilities and a known way of exploiting them are to be patched and released within 24-48 hours of discovery (typically long before the security expert would release the exploit to the public).

If the provider does the job applying patches without delays (and Virtuozzo has all the means necessary for that, by delivering kernel security patches in a "rebootless" format so the provider doesn’t need to wait for a maintenance window to apply it), just the chance of finding a vulnerable kernel on a container-based host are very low.

According to a WhiteHat security report, having a vulnerable website for less than 30 days a year is considered "rarely vulnerable", with only 16% of websites meeting this mark. Compare that with the number of days that this hypothetical platform vulnerability of container isolation could be available, even when considering a provider’s potentially relaxed policy of applying the security patches.

Based the above, considering typical vulnerabilities of modern software stacks, and also that DoS’ing the server by crashing it (typically a much more common attack than privilege escalation) is, although harmful, rarely as dangerous as data leakage (which requires true privilege escalation), we can safely conclude that in the real world, added security dangers specific to the container-hosted infrastructure is nearly non-existent.
Of course, there are exceptions. There are military hackers in enemy forces who may be equipped with tools that the general public does not know about, and who are able to exploit vulnerabilities that aren't in the CVE registry.

However, if this scenario is really a concern, one should be talking to that world-renowned expert we mentioned earlier and building multi-line security defenses, isolation and intrusion detection mechanisms. This is way beyond what the typical cloud infrastructure available to individuals and businesses has to offer.

**Solution: the importance of patching**

According to Verizon's 2015 Data Breach investigative report, 99.9% of the exploited vulnerabilities had been compromised more than a year after the associated CVE was published.

That data, although hard to believe, is indirectly confirmed by the WhiteHat report: the average number of open vulnerabilities for a website ranges, depending on industry, between 2 and 11, with a median of 5.

So, when picking a random website on the Internet, there is about a 50% probability that this website will have 5 or more open (and known) vulnerabilities which readily available wannabe-hacker-kits can exploit. And 99.9% of breaches that exploit known software vulnerabilities use those "oldies but goodies" that should have been patched a year or more ago.

With that in mind, it becomes obvious that the importance of having a good policy of the timely application of security patches is hard to overestimate. What's even more important is ensuring the ongoing compliance with that policy.

In a public cloud environment, "unmanaged" services typically means that the provider's responsibility ends with the infrastructure maintenance, while the maintenance of the software stack (including security patching) is on the end user.

Needless to say, in such environments ensuring that security patches are applied in a timely manner is nearly impossible. The more that security patches of the platform are delivered "as a service" by the infrastructure owner, the better the chances that the critical components will be up to date.

And this is where container-based solutions really shine.
They use a shared kernel, and the kernel patching is owned by the infrastructure owner. Virtuozzo also delivers kernel patches in a rebootless format, allowing service providers to apply critical updates immediately, without waiting for a scheduled maintenance window. This also reduces the period of time the system is considered vulnerable.

Also, virtual machines have their unique software stacks which are also prone to security problems - that includes virtual device drivers, network protocol components, etc. In a container-based infrastructure, the equivalents of these components are centrally managed and patched on the host, and are typically hidden behind a very restricted firewall or not exposed to the internet at all - this also decreases the attack surface for individual containers when compared to their VM counterparts. A few recent examples of such vulnerabilities include CVE-2015-8767, CVE-2015-5283, CVE-2015-3212 - all which could result in denial of service, and some of which are remotely exploitable.

Applying updates to the user level software in container-based solutions is also much easier to manage centrally, and typically less disruptive to the user’s activities compared to their VM based counterparts.

Security patching, as with any routine activity, has best chance of being performed regularly when it is as easy and undisruptive as possible. Thus, between VM based and container based solutions, the latter has a much better chance of having more up-to-date components in the software stack - because more of that stack can be patched centrally and automatically even with distributed ownership of the components (container owner vs. infrastructure owner).

And, as we concluded earlier, timely patching is one of the most important components of having a secure solution.

**Summary**

Comparing Container and hypervisor virtualization solutions, both have their unique surface of attacks - guest privilege escalation through kernel vulnerabilities (for containers) and guest tools/hypervisor communication vulnerabilities (for hypervisors).

Statistically, both are rather rare and insignificant compared to other threats.

Among software and configuration related security factors, the security of operating system components and applications are the most important factors - as those vulnerabilities are typically easiest to detect and exploit, and they are the greatest in number. Keeping the system up to date with security patches can give the greatest security benefit, and systems that allow that to happen easily with routine compliance are more secure than those that are harder to patch and more difficult to verify the compliance.
Referred documents:

1. CVE security advisories https://cve.mitre.org
2. Verizon 2015 data breach investigations report

For More Information
Visit Virtuozzo: virtuozzo.com