Why your Security Products are Inherently Insecure

You're being sold snake oil every day in the world of IT. It is about time that we just lay this out honestly. The products that you are buying are not solutions. They are methodologies. Why does this semantic difference matter? It matters because we are blindly putting tools into place under the assumption that they are a solution to a problem. The truth is that they are merely tools in the fight to solve the problem.

Conceptual – Logical – Physical

Go back to the basics of systems architecture and infrastructure design for a moment. We view things in three stages of the design process as conceptual, logical, and physical. Conceptual design is thinking at a high level on the goal such as "the application servers will be protected from intrusion". Moving to the logical physical version to expand on that concept would be something like "Layer 4-7 firewalls will be deployed at the ingress and egress point for the application servers". Getting down to the physical is something like "Product X will be deployed to provide layer 4-7 firewall protection" which is the result of designing to meet the first two requirements.

The issue that we face as an industry is two-fold. First, we often start at "Product X will be deployed" without having done the due diligence on what the actual business and technical requirements are which need to be solved. The second issue is that we buy Product X, deploy Product X, and then everyone goes for a project completion dinner and celebrates that we have finished up the deployment with the bold assumption that we are inherently secure.

Many organizations are buying products or embracing some new technologies into their environments based on a promise. Promises should always bear translated to assumptions. I'll start with one that I am seeing a lot of lately which is this:

> "Containers are more secure for applications than virtual machines"

This is both true and false at the same time. The wording is important. What the phrase should say is "containers have the ability to be architected and deployed to be more secure for applications than traditional virtual machines".

Here's why phrasing is important.

Why is your security product inherently insecure?

You can't buy a bow and arrow and suddenly you are an archer. The same goes for security. Just because you have bought a security product, it does not mean that you are secure. It's actually the polar opposite. Your environment is inherently insecure. Even if you are absolutely sure that you are deployed in the most secure manner possible, you should ALWAYS ASSUME that you have been breached.

What's the solution for this? This comes in three forms:

1. Accept that you are insecure and build processes around that assumption

2. Deploy and continuously test your security platforms
3. Engage third-party testers and products to ensure continuous objective testing

Let's dive into these three areas a little bit further.

Accept that you are insecure and build processes around that assumption

Point 1 is the key to begin with. Assume you have been breached. Now what? How are you aggregating your logs? How are you protecting the logging both locally on the application endpoints as well as in your central logging environments? If you have to assume that your ingress has been compromised, you also have to assume that your log environments have been compromised as well. You need local protection on each system plus centralized, read-only aggregation with regular snapshots of that environment to ensure its integrity too.

The build process you use will inevitably call on some external dependencies. It could be patches, software updates, or any of a wide variety of files and applications. Assume that these are inaccessible or compromised as you define your programmatic build process to use locally cached data and application dependencies as much as possible. And yes, the programmatic build process is key to ensuring consistency and security. You should include checksum and signature detection for all source files as you put them into the virtual application instances.

Deploy and continuously test your security platforms

Test-driven development is a great methodology. I have long been a user and a proponent of what is known as test-driven infrastructure and this includes the need for security as a part of the cycle. The only way that you know your detection system is working is if you test it when there is an issue. Assuming detection without truly testing the response means that you are relying on the assumption. Your CISO does not rely on assumptions, and neither do your customers.

Whichever products you embrace in your IT security portfolio, they will inevitably come with some form of baked in testing procedures and processes. Be aggressive and adamant with your vendors that this is a requirement for you. Nobody wants to be caught going back after a vulnerability to have to find out that it was detectable and preventable.

Engage third-party testers and products to ensure continuous objective testing

I hire someone to do my taxes. Yes, I can do them myself. That doesn't mean that I'm an expert and can find every advantage within the tax code to get the best results. Why would I treat security and vulnerability testing any differently than any other discipline in my business and IT organization. Using 3rd party companies will give you the ability to lean on them for expertise, and most importantly, certification and validation of your security stance in an active environment.

Having spent years in financial services environments which have stringent requirements around auditing and security, I can tell you that no matter how secure even the IT security team thought they were, a 3rd party can come in and teach some rough lessons in a couple of hours.

Turn Assumptions into Actions

Going back to the example that containers are more secure than virtual machines gives us a great one to work from. Containers typically run thinner and provide a smaller attack surface for vulnerabilities, malware, and other attacks by bad actors. No, not Lorenzo Lamas, but anyone who is attempting to breach your environment. We will usually hear them being referred to as bad actors.

The truth is that containers as a construct, are solving deployment challenges first. Security is a secondary win that implies you have the practices in place to assure that security is greater than that of a traditional virtual machine. Containers are leveraging namespaces and other methods if isolation with the underlying server host to provide some potentially powerful protection. It does not mean that by default the container version of your application is more secure. It means that at the lowest possible layers, not including poor application code, SQL injection, XSS and many of a thousand different other attack vectors are solved by deploying in a container versus a traditional virtual machine.

The long and the short of it is that security products, or any technology products for that matter, are inherently insecure unless you deploy them with all of the practices in place around them to ensure the security.

This conversation on Twitter is a nice way to show how challenging it is to convey the message:

> TOTALLY AGREE. Suggesting that containers makes "apps more secure" is not correct. Operationalizing them perhaps. https://t.co/HVrVOaJIlE
>
> — Hoff (@Beaker) April 9, 2017