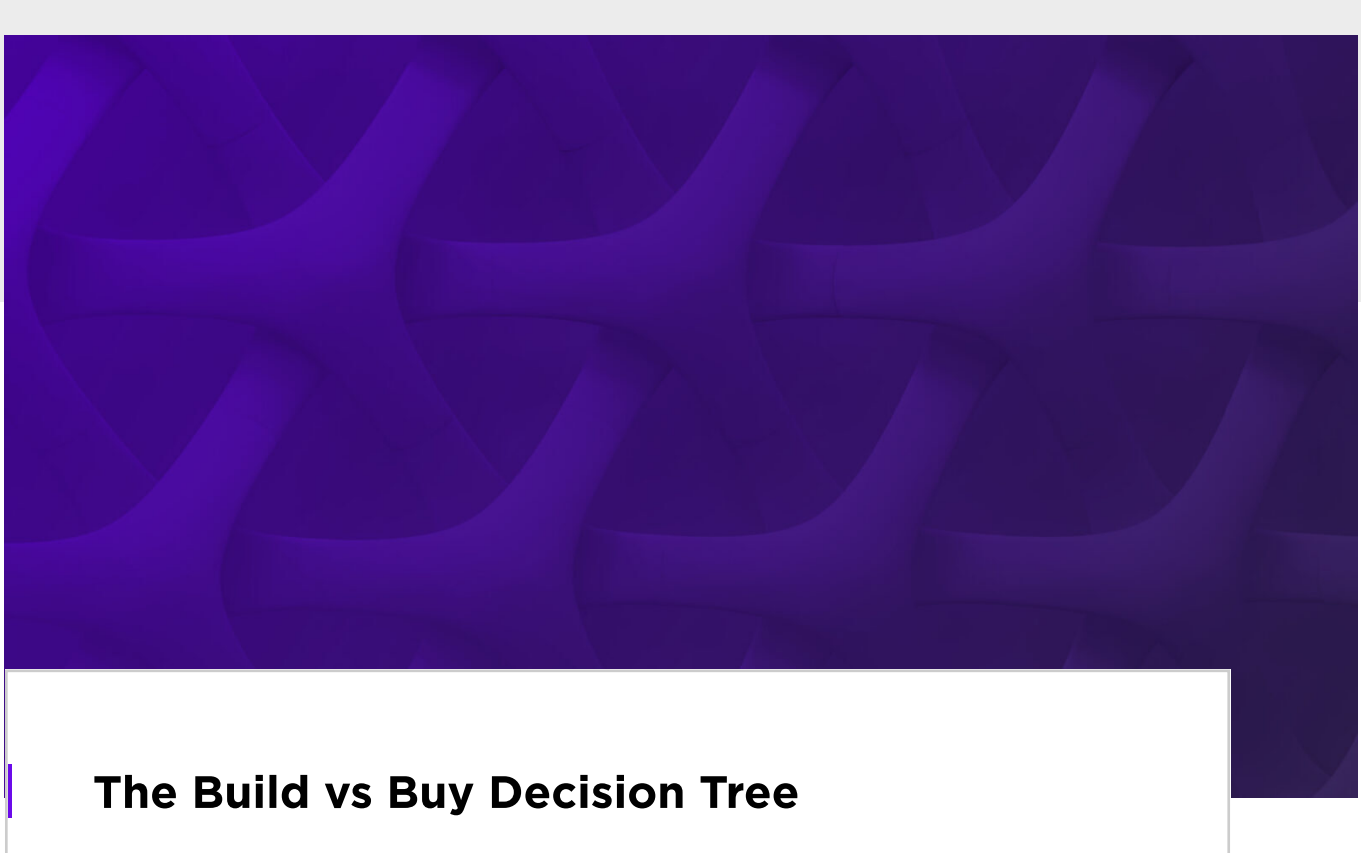
SentinelOne blog

GET A DEMO



The Build vs Buy Decision Tree

May 1, 2018
by SentinelOne

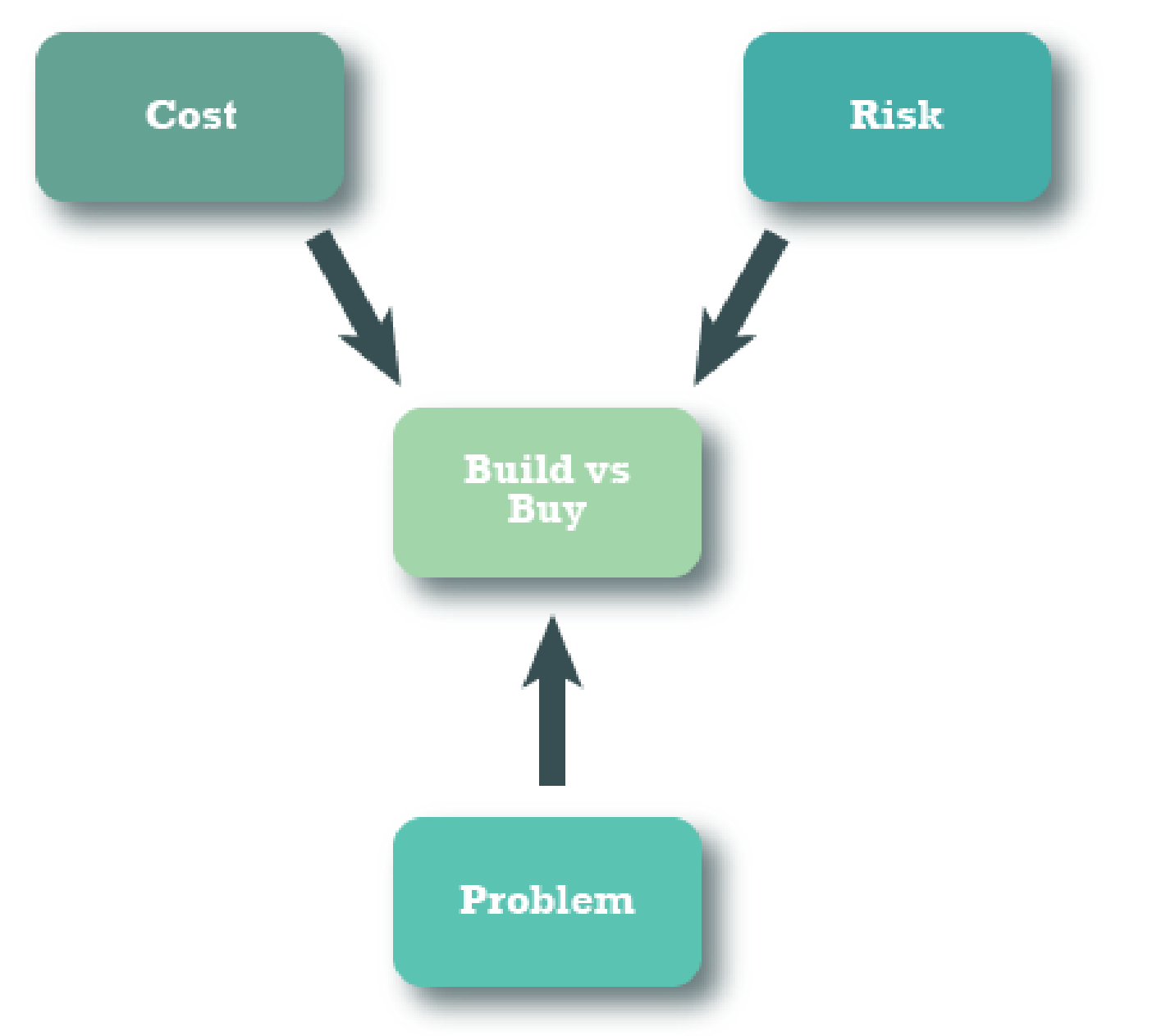
Chocolate or vanilla? Pancakes or waffles? Coke or Pepsi? We decide between similar choices every day.

Some of us have preferences, and other times it's just a feeling in the moment. A common decision in the IT world is the "build vs buy" decision. Sometimes this decision is not so cut and dry.

Can the decision to build or buy paralyze us with fear? Certainly. Do some have preferences? Definitely. However, all is not lost. There can be a logical system to decide whether to build or buy when it comes to software.

The Build vs Buy Trifecta

When the initial fear and trepidation wears off, you find out there are three different pieces that go into a build vs buy decision: cost, risk, and the problem you are trying to solve.



Some developers and managers may have preferences for various reasons. Maybe a bad experience in the past has soured them to one choice or the other. However, you need to treat each decision differently. Both choices are right some of the time. What you're asking is, "How do I use these three pieces to come up with a recommendation?"

The Bottom Line… Literally

A new project kicks off. It's been decided that your company needs a new piece of software. This is where the inevitable build vs. buy decision rears its ugly head. It is not an easy decision to make. What you decide has long-lasting consequences.

Cost should be your first concern. What is the cost of building the system? What is the cost of buying a vendor product? In order to answer these questions, you need to understand that there are different kinds of costs.

License to Bill

The first cost to consider when deciding to build vs buy is the cost to get the software up and running. For build, that is the cost of building the software. For buy, that is the cost of licensing.

Building software can be expensive if the software is large and complex, requiring a large development team to create it. However, license costs can be even more if you don't understand exactly how license costs are calculated.

There are some vendors that charge you based on how large your processors are. Some data analytic software charges you based on how much data you put into the system. If you are not careful, you may find yourself racking up huge license fees in order to accomplish what you feel are basic tasks.

Try to find vendors that have reasonable license requirements, such as a simple per user license or per server license. Some programs support setting up a pool of licenses on a server that can be "checked out" when users need them and then given back for others to take after use.

When you build, you have no license fees. You only need to pay for the man-hours and infrastructure needed to build the software. This is sometimes simpler but also notoriously difficult to estimate accurately. When projects take longer than expected to finish, you're stuck spending more and more money before the application is even ready for use. Also, don't forget to factor in the cost of maintaining the application over time.

Not Plug and Play

When you buy a product, especially when you're a large enterprise, don't fall into the trap of thinking that you can simply install the product and be done. You'll need a team of your own that will implement the installation of the product and integrate it with your other systems. You'll need a team that continues to install patches and manage users and features.

Also, be wary of consultants and partners of the vendor that help you to implement the product. They can be a source of expertise and help. They can also screw things up if they're not competent, costing you even more money to fix the problems they introduce. Not saying you shouldn't hire implementation partners, but let the buyer beware. Try to do your homework on them before you sign a contract.

With all this in mind, you might get the idea that you should simply choose the cheapest option. That is not really the case. There are still two factors to consider.

Risky Business

Why is this a risk? One word: bugs.

The second key piece of the build vs buy decision is risk. Risk is the likelihood and potential impact of something going wrong. Either choice has different risks, and it's up to you to know which ones matter the most.

A large risk when you build a piece of software is whether or not you actually deliver. We've all been a part of software projects that deliver late or not at all, even though large amounts of money were invested in them. This risk can be higher or lower based on what kind of software you're trying to build. Building software in a domain that you're not familiar with can lead to trouble and high expenses.

Risk rears its head quite a bit when you go for the buy option. The main risk is that you have a piece of software in your environment that you don't fully control. You also don't have access to the source code. Why is this a risk? One word: bugs.

When you find a bug in a piece of software you built, you can simply create a ticket for the development team and fix it. when you find one in a product you bought, you likely have to submit a ticket to the vendor support site. They try to recreate the bug in their own environment and then get back to you. If they agree it's a bug, then they slate it for their next release. You'll get that in a few months at the earliest. If that bug is a security vulnerability, then you might be stuck running vulnerable software in production that you can't patch right away.

Any time you introduce a piece of software you don't fully control into your environment, you're introducing risk. There are also many software as a service (SaaS) offerings where your data will be sitting in someone else's data center. Make sure they have all of the protections that you require for you to feel comfortable with giving them your data.

You don't want to spend too much money. You want to reduce the risk to your business if something goes wrong. Up next, the number one question to ask.

What Problem Do *You* Solve?

When you created your business, you did it to solve a problem. Your customers or clients pay you to solve that problem for them. This is an important factor in the build vs buy decision.

What problem are you trying to solve by building or buying software? Is the specific problem you are trying to solve related in any way to your core value proposition? If not, then buying software usually is the best way to go.

Don't go through the trouble of building from scratch something highly specialized or doesn't really add to your revenue in any way. For example, how many companies do you know that built their own custom email client? It seems silly. Email is a specialized service that doesn't really directly relate to how you make money (unless your business is email services). Therefore, you simply buy software to do it for you.

If you are a financial company, you wouldn't buy software that holds your client funds and transactions. You'll likely want to keep that in-house. Your HR system, on the other hand? Use a SaaS product so that you can concentrate on building software that makes your clients money. Buy the stuff around the edges that every company needs. You build what differentiates you.

And the Winner Is…

The build vs buy decision is often an art more than a science. However, you can apply some logic to the equation by looking at three key areas and deciding where you stand on them. When making this decision, consider the following:

- Understand the true costs of building and buying software, including license models, implementation, and maintenance costs.
- Understand the risks inherent in purchasing software and placing your data inside. You won't be able to fix bugs as easily.
- Put your development skills to use in building what truly differentiates your business. Buy software for the commodity functions that every business needs.

No matter what you choose in the end, how you implement it will decide how successful it is. Choose wisely, execute with excellence, and never lose sight of your mission.

This post was written by Justin Boyer. Justin has been a software developer for eight years, doing development across all layers and using multiple languages and frameworks. He's recently made the switch to security, becoming CSSLP and Security+ certified. He blogs about application security practices and principles at Green Machine Security.

Like this article? Follow us on LinkedIn, Twitter, YouTube or Facebook to see the content we post.

Read more about Cyber Security

- Scalyr Platform: Kubernetes Monitoring, Performance, and Usability