POLYVERSE

POLYMORPHIC DEVSECOPS WITH POLYVERSE BUILD FARM

Although DevSecOps greatly improves security, the risk of cyberattacks remains high because many DevSecOps practitioners only address the application layer of the software stack. To be effective, security measures must cover everything: infrastructure, platform, CI/CD, service mesh and the application. The Polyverse Build Farm addresses vulnerabilities up and down your DevOps stack.

From voting machines to employee data systems, government systems have a lot that hackers want and can sell. Whether it's financial data, employee records or secure transactions, the stakes (and the black ops value) are high. Hackers tend to follow paths of least resistance, taking advantage of new risks and compliance holes, all the while adding costs to your software development and deployment efforts.

At the heart of security risk is the opportunity for hackers to use a single exploit to attack thousands or even millions of devices and servers that all run the same hypervisors, operating systems, security rules and applications. Hackers reap benefits through economies of scale, targeting the same vulnerabilities over and over with little effort, making it easy for them to quickly and economically compromise thousands of systems.

Adding Security to Every Layer of Your DevOps Stack—Not Just the Application

For some beleaguered teams, implementing DevSecOps strategies has enabled them to think more proactively about, and tackle, security issues throughout the DevOps software development cycle. Improvements include better code provenance, tighter CI/CD integration and better monitoring. But the fundamental problem—the risk of a broad, debilitating attack—remains because many DevSecOps practitioners just address the application layer of the software stack, the part customers see and interact with. To be effective, security measures must address everything: infrastructure, platform, CI/CD, service mesh and the application.

With traditional DevSecOps, you and your team may have dramatically improved the security approaches you bake into the software stack, but you may still devote chunks of time to patch and monitor systems to stay ahead, particularly at scale. And you still need visibility, management and control of your systems while retaining the capacity to build and maintain custom packages as needed to protect current and legacy systems without recreating the wheel.

Practitioners need new ways of thinking about DevSecOps, breaking the stack into distinct layers and identifying the specific vulnerabilities of each. (see figure 1)

The layers vs. the Process

Consider the layers at the base of the stack: infrastructure and platform. This is where your on-prem or cloud servers, networks, hypervisors and storage live. Even if you're using a cloud service —and as a result presume security at the base is handled—a lot of complex physical infrastructure can still be vulnerable to attacks.

The base of the stack is also where the

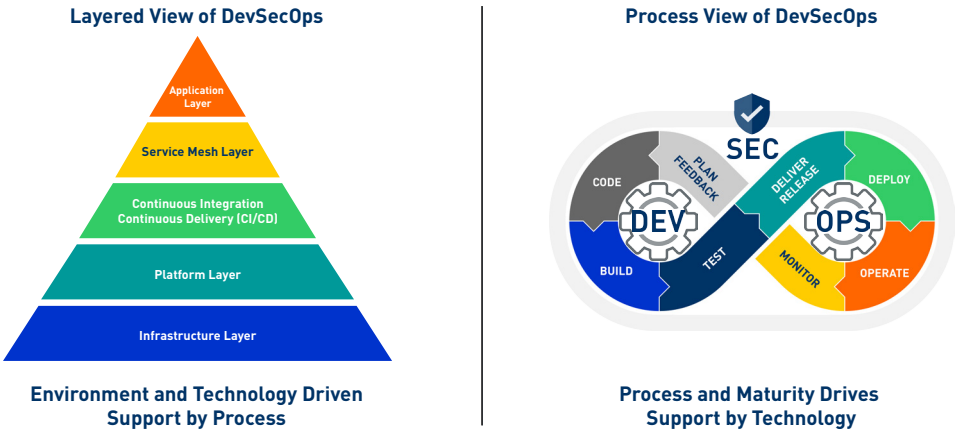

Figure 1

platform presents itself to your software environment and interfaces with your kernel and OS. All these components should be treated as entry points that bad actors can exploit.

As you move up the stack to your CI/CD workflows, perhaps integrating and pulling code from GitHub or container images from DockerHub, your principal worry is the risk of bad actors injecting malicious code into your build process. This can be hard to spot and often requires complex security countermeasures and code analysis to prevent, particularly if the malicious code is injected early and starts using system resources from the first stages of your deployments. For example, the system load, monitored over time, won't necessarily show anomalies because the bad code has been eating up resources from the start.

Further up the stack is the service-mesh layer, where your usual security integration lives. This is where firewall settings, POSIX rules and other overt system security settings are applied. By design, this layer blocks access to your systems and services at a macro level, keeping vulnerable ports closed and preventing system access to unauthorized users. However, if left as an afterthought, these services provide easy openings for malicious actors.

When you reach the top of the DevSecOps stack (which is the customer-facing application itself), you're finally addressing OS and code vulnerabilities, which is where most teams spend their time and resources. But if the containers you're using all deploy the same unaltered Linux OS or unpatched version of Python3, you're creating a fleet of application nodes that all share the same vulnerabilities. An unpatched SSH flaw in an Ubuntu-based container, for example, will be multiplied by every other Ubuntu container in your fleet.

Prevent Attacks Through System Diversity

Polyverse's polymorphic technology tackles all these problems and makes it easy to deploy and monitor secure software on thousands of devices by making each host unique. Allowing you to create unique base Linux OS images using flavors you know and love, this technology first compiles and scrambles your code, creating continuous system diversity. As a result, each instance isn't a vulnerable clone but, instead, a unique, randomized instance that offers no insight into any other servers or services in your fleet. Exploits that target common vulnerabilities simply won't work.

When you deploy with Polyverse Build Farm, you don't just target the application layer, though, but are able to address vulnerabilities up and down your DevOps stack. (See figure 2)

A Build Farm enables you to deploy unique OS instances on-prem or in the cloud without having kernel experts on staff and lets you compile your code in different ways to create real differences that bad guys can't attack without a lot of extra work. Bad actors are unlikely to commit the time and resources it would take to craft individual, customized attacks. Instead of exploiting your unique systems, they would go elsewhere and find vulnerabilities common to thousands of Linux servers.

Polyverse Build Farm enables you to do this at scale without changing your code, your base OS, your network configurations, your software delivery practices or anything in your stack. The ability to detect code anomalies and zero-day vulnerabilities is built in and provides you and your team with monitoring data about the entire stack, from memory usage to port targeting. A Build Farm does this without slowing down your deployments, distributing secure systems across your entire environment, regardless of whether you're using 25-year-old legacy systems or the latest containerized apps tuned for Kubernetes.

Applications compiled and delivered with Polyverse Build Farm remain easy to monitor and confidently maintain. You can, for example, build in trust by verifying your code end to end, test your configurations in real time, and deploy hundreds or thousands of instances quickly and reliably.

Build Farm also gives you the ability to:

- Confirm source libraries against code platforms such as GitHub, ensuring no extraneous code is injected into your builds
- Create and verify your own instances using base Linux distributions you already know

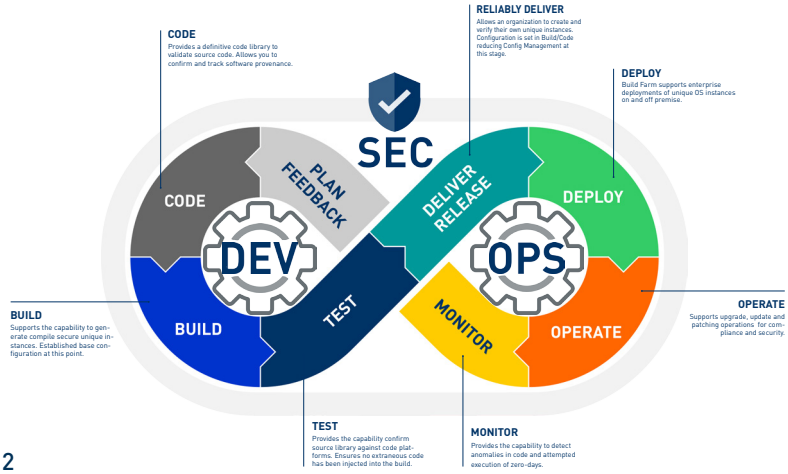

Figure 2

- Defend against the latest memory-based and fileless attacks
- Deploy unique, secure instances on- or off-prem

Robust Security Across Your DevSecOps Practice

Organizations that automate the software build process with Polyverse Build Farm often see significant cost savings. They can reduce the time needed to patch and optimize systems by 40 percent and the time it takes to configure systems by 30 percent. This means lower costs, happier teams, and better security across your entire software stack.

The features built into Polyverse Build Farm make your DevOps much more secure, hardening your security measures and making your systems far less appealing to bad actors by increasing their risk of discovery and leading them to move on to other, less secure operations.

If you're responsible for securing hundreds or thousands of systems on public networks, give the bad guys the headaches for a change. Try Polyverse Build Farm today.

Other Resources

- DevSecOps for Your Full Stack Webinar
- Polymorphing Build Farm for Government Datasheet
- Polymorphing for Linux Whitepaper

For more information, contact
sales@polyverse.com

or visit our website
https://polyverse.com

POLYVERSE