**<https://www.logicworks.com/blog/2017/01/what-is-security-by-design/>**

**What is Security by Design?**

Security is “Job  Zero” for every company. That begins with taking a more proactive  approach to infrastructure security — one that doesn’t rely on the  typical protective or reactive 3rd party security tools, but instead  builds security into your infrastructure from the ground up.

As your company increases its cloud  presence, it has an opportunity to rethink who and what is responsible  for security in your environment. You also want to be able to integrate  security processes into your development pipeline and maintain  consistent security configurations even as your infrastructure  constantly changes. This has led to the rise of Security by Design.

**The Security by Design Approach**

Security by Design (SbD) is an  approach to security that allows you to formalize infrastructure design  and automate security controls so that you can build security into every part of the IT management process. In practical terms, this means that  your engineers spend time developing software that controls the security of your system in a consistent way 24×7, rather than spending time  manually building, configuring, and patching individual servers.

This approach to system design is not new, but the rise of public cloud has made SbD far simpler to execute.  Amazon Web Services has recently been actively promoting the approach and [formalizing it](https://aws.amazon.com/compliance/security-by-design/) for the cloud audience. Other vendors promote similar or related  concepts, often called Secure DevOps or Security Automation or  Security-as-Code or SecOps. The practice becomes more important as your  environment becomes more complex, and AWS actually has many native  services that, if configured and orchestrated in the right way, create a system that is more secure than a manually-configured on-premises  environment.

Does this mean that companies no  longer need security professionals, just security-trained DevOps  engineers? Not at all. When security professionals embrace this  approach, they have far greater impact than in the past. This is  actually an opportunity for security professionals to get what they have always dreamed of: introducing security earlier in the development  process. Rather than retroactively enforcing security policies — and  always being behind — they are part of the architecture planning process from Day 1, can code their desired specifications into templates, and  always know that their desired configurations are enforced. They no  longer need to be consulted on each and every infrastructure change,  they only need to be consulted when the infrastructure *templates*change in a significant way. This means less repetitive busy-work, more focus on real issues.

**Security by Design in Practice**

In practice, SbD is about coding  standardized, repeatable, automated architectures so that your security  and audit standards remain consistent across multiple environments. Your goals should be:

* **Controlled, standardized build process**: Code architecture design into a template that can build out a cloud  environment. In AWS, you do this with CloudFormation. You then code OS  configurations into a configuration management tool like Puppet.
* **Controlled, standardized update process**: Put your CloudFormation templates and Puppet manifests in a source code management tool like Git that allows you to version templates, roll  back changes, see who did what, etc.
* **Automated infrastructure and code security testing as part of CI/CD pipeline**: Integrate both infrastructure and code-level tests into code deployment process as well as the configuration management update process. At  Logicworks, we often use AWS CodeDeploy to structure the code deployment process. You can also use Docker and AWS ECS.
* **Enforced configurations in production**: Create configuration management scripts that continually run against  all your environments to enforce configurations. Usually hosted in a  central management hub, and necessitates a hub-spoke VPC design  approach.
* **Mature monitoring tools with data subject to intelligent, well-trained human assessment**: In compliant environments, your monitoring tools are usually mandated  and logs must be subject to human review; we use native AWS tools like  AWS CloudWatch, CloudTrail, and Inspector, as well as Alert Logic IDS  and Log Manager and SumoLogic  to meet most requirements. SumoLogic  helps us use machine learning to create custom alerts that notify our  24×7 Network Operations Center when unusual activity occurs, so that  those engineers can take appropriate action with more accurate real-time data.
* **Little to no direct human intervention in the environment…ever**: Once all these tools are in place, you should no longer need to  directly modify individual instances or configurations. You should  instead modify the template or script to update (or more ideally,  relaunch) the environment.

We have gone into significant  technical depth into Logicworks’ security automation practices in other  places; you can see our Sr. Solutions Architect’s talk about security  automation below, watch him talk about our general automation practices [here](https://go.logicworks.com/automation-and-security-in-a-cloud-devops-world-webinar-watch), or [read this in-depth overview of our automation practices](https://www.logicworks.com/blog/2016/12/automation-in-a-cloud-devops-world/).

Here are some other great resources about Security by Design and Secure DevOps:

* [AWS Security by Design White paper](https://d1.awsstatic.com/whitepapers/compliance/Intro_to_Security_by_Design.pdf)
* SANS Institute: [Continuous Security: Implementing the Critical Controls in a DevOps Environment](https://www.sans.org/reading-room/whitepapers/critical/continuous-security-implementing-critical-controls-devops-environment-36552)

**Compliance + Security by Design**

As you can imagine, the SbD approach  has significant positive impacts on compliance efforts. The hardest  thing to achieve in infrastructure compliance is not getting security  and logging tools set up and configured, it is maintaining those  standards over time. In the old world, systems changed infrequently with long lead-times, and GRC teams could always spend 2-3 weeks evaluating  and documenting change manually (usually in a spreadsheet). In the  cloud, when code gets pushed weekly and infrastructure is scalable, this manual compliance approach can severely limit the success of cloud  projects, slow down DevOps teams, and frustrate both business and IT.

Running applications in the cloud  requires a new approach to compliance. Ideally, we need a system that  empowers developers and engineers to work in an agile fashion while  still maintaining security and compliance standards; we need a toolchain that a) makes it easier to build out compliant environments, b)  provides guardrails to prevent engineers/developers from launching  resources outside of compliance parameters, and c) provides ongoing  documentation about the configuration of infrastructure resources. The  toolchains we have already described — templating, configuration  management, monitoring — allow us to launch new compliant environments  trivially, ensures very limited access to the environment and full  documentation on every change. Together, this means a greatly reduced  risk of undocumented configuration change, error, or lack of adequate  knowledge about where sensitive data lives, and therefore greatly  reduced risk of compliance violations.

When systems are complex, there must  be an equally powerful set of management tools and processes to enforce  and maintain configurations. Continuous compliance is only possible if  you treat your infrastructure as code. If your infrastructure can be  controlled programmatically, your security and compliance parameters are just pieces of code, capable of being changed more flexibly, versioned  in Git like any piece of software, and automated to self-correct errors. This is the future of any type of security in the cloud.

**The Future of SbD**

SbD allows customers to automate the fundamental architecture and, as AWS [says](https://d1.awsstatic.com/whitepapers/compliance/Intro_to_Security_by_Design.pdf),”render[s] non-compliance for IT controls a thing of the past.”

Recent announcements out of AWS  re:Invent 2016 are particularly exciting. AWS launched a major update to their EC2 Systems Manager tool, which is a management service that  helps you automatically collect software inventory, apply OS patches,  create system images, and configure Windows and Linux operating systems. Basically, AWS is filling the gaps in its existing SbD toolchain,  stringing together a lot of the controls described above and allowing  you to define and track system configurations, prevent drift, and  maintain software compliance. Although EC2 Systems Manager was upstaged  by several more headline-worthy releases, the service will make a  significant difference to compliance teams in the cloud.

In the future, expect AWS and other  cloud platforms to launch more comprehensive tools that make it easier  for enterprises to achieve SbD in the cloud. The tools currently exist;  but assembling these tools into a robust framework can be a challenge  for most IT teams. Expect enterprises to turn towards security-focused  partners to fill the skills gap.