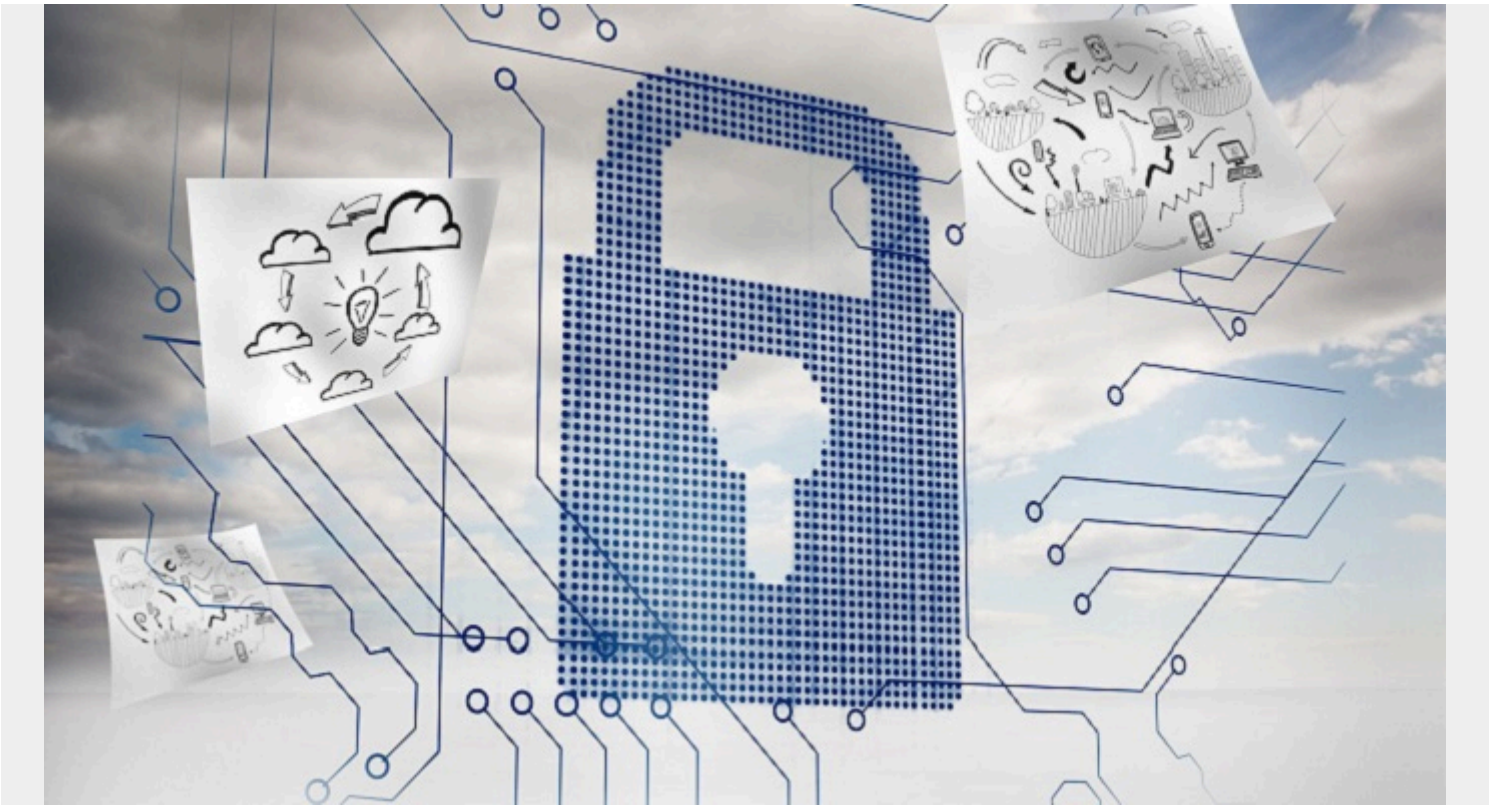
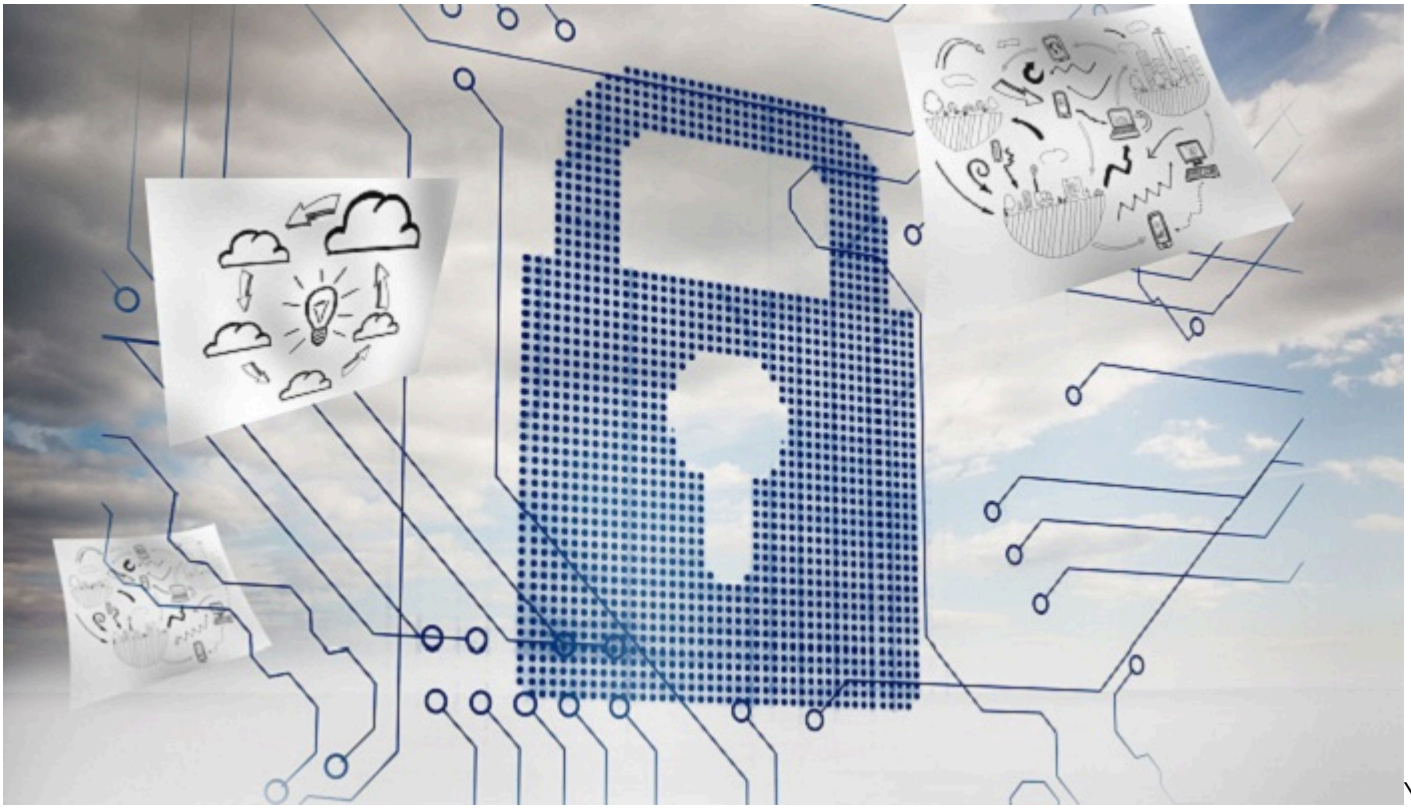
FIVE BEST PRACTICES FOR BUILDING SECURITY AS CODE INTO A CONTINUOUS DELIVERY PIPELINE



This is the fourth blog in our mini-series that illustrates how BMC was able to use agile development, cloud services, an Infrastructure as Code approach, and new deployment technology to deliver a new cloud native product.

Be sure to also read: Getting Started with Cloud Native Applications, 9 Steps for Building Pipelines for Continuous Delivery and Deployment, and Infrastructure and How "Everything as Code" changes everything



You need agility to meet the requirements for digital business, but it's also critical to ensure that security and compliance are built into your apps from the start. That involves thinking about security and compliance as code, and not as an afterthought or add-on. When security is not initially built into code, then compliance becomes much more onerous. As a result, just as a release is nearing deployment readiness it requires security vulnerability testing, penetration testing, threat modeling and assessments. These activities are done using tools, security checklists and documents. If critical or high vulnerabilities or security issues are found, the release is stopped until these are fixed, causing delays of weeks or even months.

The "Security-As-Code" principle addresses this challenge by codifying security practices as code and automating delivery as a part of a DevOps pipeline. This approach ensures that security and compliance are measured and mitigated very early– at the use case, design, development and testing stages of application delivery. Moving these processes earlier in the DevOps pipeline is known as the "Shift left" approach, which effectively allows you to manage security just like code by treating security attributes such as policies, tests and results as you would lines of code – storing them in the same repository and progressing them through the same DevOps pipeline.

Five essential best practices for applying security and compliance as code

How do you deliver apps quickly while ensuring that they are secure and compliant? We've identified five key best practices using "Security-As-Code" and "Compliance-As-Code" based on our experience in using a cloud native application running on Amazon Web Services (AWS).

1. ***Define and codify security policies***
   Security is defined and codified for any cloud application at the beginning of a project and is kept in a source code repository. As an example, a few AWS cloud security policies are defined below:

- Security groups/firewalls secured
- Virtual Private Cloud (VPC), Elastic Load Balancing, and other requirements are enforced and secured
- All Elastic Compute Clouds (EC2) and other resources must be in the VPC and each resource is individually firewalled
- Encrypt all AWS resources, as well as logs, and use Key Management Services

    Security policies need to be automated. By pressing a button, you should be able to evaluate security policies for any application at any stage and environment. This is a major shift that happens when you follow the principle of security as code – everything about security is codified, versioned, and automated. Security artifacts are checked in as "code" in a repository and versioned.

    Enterprises should build standardized security patterns to allow easy reuse of security across multiple organizations and applications. Using standardized security templates (also as "code") will result in out-of-the-box security, instead of having each organization or application owner define the policies and automation by team. For example, if you're building a 3-tier application, a standard cloud security pattern must be defined. Similarly, for a dev/test cloud, another cloud security pattern must be defined and standardized. Both hardened servers and hardened clouds are patterns that can yield security out of the box.

    With AWS cloud, the 5 NIST AWS Cloud Formation templates can be used to harden your networks, VPCs, and identity and access management roles. The cloud application can then be deployed on this secure cloud infrastructure where these templates form the "hardened cloud." For private cloud or on-premises servers, the server hardening can be done by products such as BMC BladeLogic Server Automation. For public cloud and application hardening, we have a new cloud service being developed to define and validate public cloud services.

2. ***Define security user stories and do a security assessment of the application and infrastructure architecture***
   Security-related user stories must be defined in the agile process, just like any other feature stories. Examples include "input validation for cross-site scripting and SQL injection," "SSL/TLS enabled for all communication," "automated application security testing," and so on. This process will ensure that security is not ignored. Security controls are identified based on the application security risk and threat models for the application and infrastructure architecture. These controls are implemented in the application code or through policies.
3. ***Test and remediate security and compliance at application code check-in***
   As frequent application changes are continuously deployed to production, the security testing and assessment is automated early in the DevOps pipeline. Security testing is automatically triggered as soon as code check-in happens for both application and infrastructure changes.

    There are many security vulnerability scanning tools available from BMC and other vendors or as open source software that can help automate and speed the detection and remediation of vulnerabilities. If critical or high vulnerabilities are found, automation tools can create "defects/tickets" for resolution that will be assigned to the owner of the component.

    For modern cloud native applications, security and compliance testing at code check-ins involve evaluating infrastructure-as-code artifacts such as AWS cloudformation templates

against policies. A new BMC cloud service can be used to enable this automation and integrated in a pipeline tool, such as Spinnaker.

4. ***Test security policies to ensure they don't violate requirements***
   Security policies need to be tested for violations of security policies, such as "no s3 buckets should be publicly readable or writable" or "do not use open security groups with 0.0.0.0/all traffic allowed." This testing needs to be automated as a part of the application delivery pipeline using security and compliance tools. For example, in our application delivery pipeline for a cloud native application, we automatically run security scans that are orchestrated from Spinnaker to ensure that every push to production goes through security scanning.For regulatory compliance policy checks — such as compliance with the Center for Internet Security and Defense Information Systems Agency requirements on servers, databases and networks — BMC BladeLogic suite of products can be effectively used to detect as well as remediate compliance.
5. ***Test and remediate security and compliance in production***
   As an application moves from development to QA to test, the automated security and compliance testing continues to happen. However, the risk of finding issues at this point is much lower because most of the security automation is done much earlier in the DevOps pipeline. Tools such as BMC BladeLogic Server Automation and BMC Threat Director can be used for production security patching after scanning production servers for mutable infrastructure. This is particularly critical because more than 80% of attacks target known vulnerabilities and 79% of vulnerabilities have patches available on the day of disclosure. For applications deployed in public clouds such as AWS, security and compliance evaluations can be automated through Spinnaker and new BMC cloud services.

   In modern cloud-native application stacks, components are never updated or patched but they are always replaced whenever changes are needed in the application or infrastructure. Even in these cases, new containers or servers are first tested for security and compliance before deploying them to production.

Shift Left

Instead of approaching security as an add-on process at the end of the release, or as periodic security scanning of production environments, start with security at the beginning of an application release. Represent it as "code" and bake it in. By following these best practices, successful product companies can ensure security while maintaining the agility and speed of innovation.