Cyral | WHITE PAPER

What Is Security As Code?

Contents

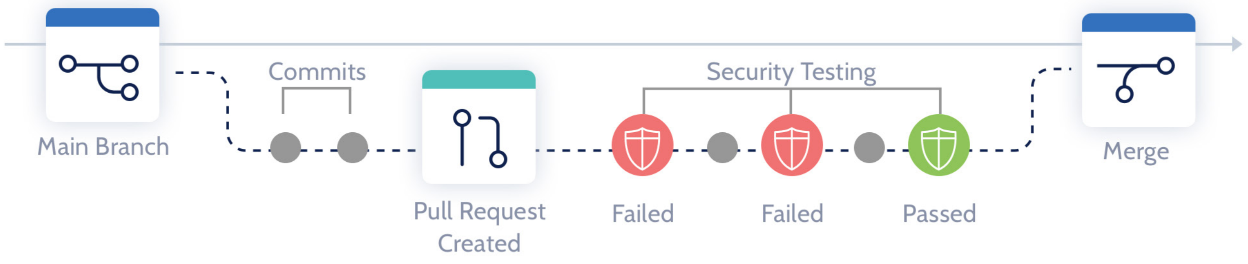
Overview

Security as Code is the methodology of codifying security and policy decisions and socializing them with other teams. Security testing and scans are implemented into your CI/CD pipeline to automatically and continuously detect vulnerabilities and security bugs. Access policy decisions are codified into source code allowing everyone across the organization to see exactly who has access to what resources. Adopting Security as Code tightly couples application development with security management, while simultaneously allowing your developers to focus on core features and functionality, and simplifying configuration and authorization management for security teams. This improves collaboration between Development and Security teams and helps nurture a culture of security across the organization.

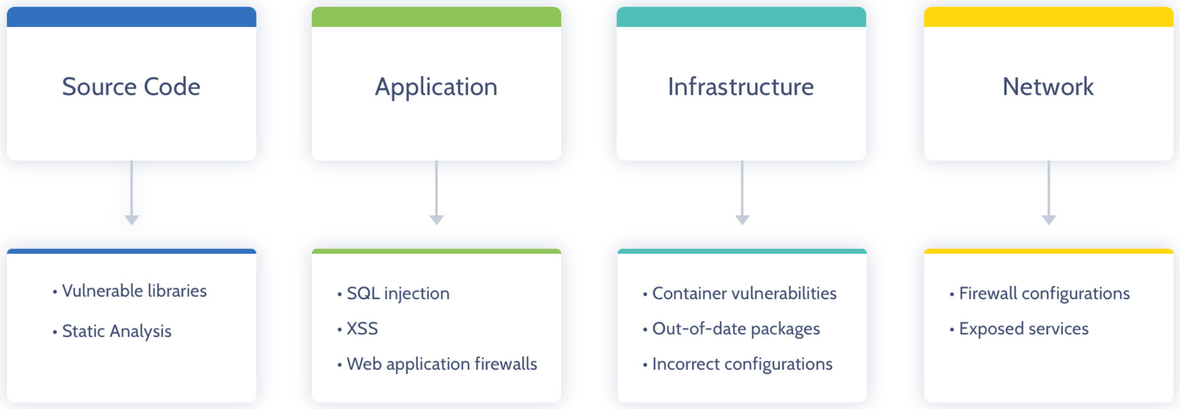
Implementing Security as Code

Security as Code generally comes in three different forms: security testing, vulnerability scanning and access policies. Each of these enable your Engineering teams to understand and fix security issues early on in development as opposed to waiting until the project is ready to ship and is blocked due to security concerns. When you take on a Security as Code mentality, you are codifying collaboration directly where your development teams are working. Security as code lifts up Development and Security teams together to allow each to focus on their core strengths.

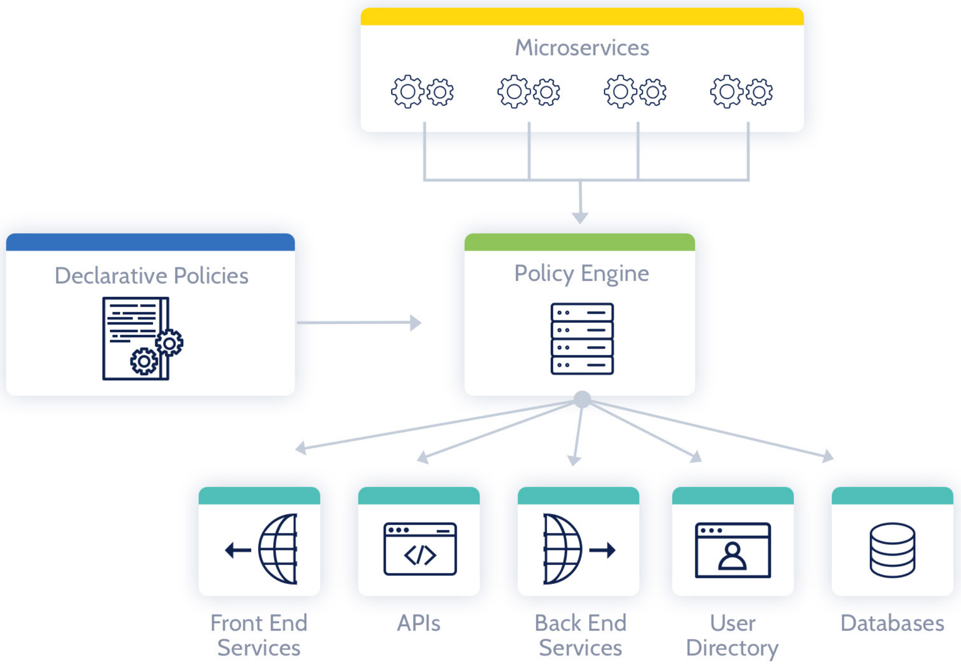
Security testing expands on best in class coding practices to add to the standard suite of tests to not only include functional and integration testing but also security focused testing. Static analysis for security vulnerabilities can be implemented on each commit or pull request. Permission boundaries can be checked to verify they cannot be crossed. APIs can be tested to ensure they're meeting authentication and authorization requirements. Security testing meets your developers where they already are, providing them immediate feedback on each and every commit.



Vulnerability scanning at every level of your architecture across your pipeline can verify that each section of your application and deployment is secured against known vulnerabilities. Source code can be scanned for vulnerable libraries. For example, applications can be scanned for susceptibility to XSS and SQL injection. Containers can be scanned for vulnerabilities in individual packages and for adherence to best in class practices. Full scanning of test, staging and production environments can be done continuously and automatically. Scan early and scan continuously to verify your expected security controls are in place and so that you can find issues sooner rather than later.



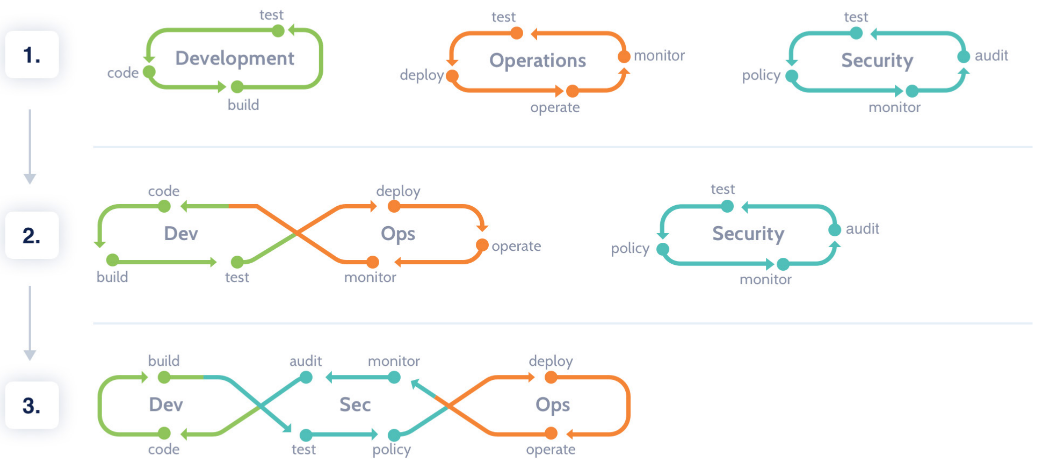
User and data access policies codify governance decisions that can then be reviewed by anyone in your organization. These policies can be standardized, reducing the toil necessary to constantly monitor and maintain one off requests. Authorization can be offloaded to external libraries allowing your Dev teams to focus on core features. Security teams now have a central repository to work directly with developers to monitor and review authorization, allowing the entire company to move faster without breaking core security and compliance requirements.



Genesis of Security as Code

Historically, organizations consisted of separate and siloed Development, Operations and Security teams. Dev teams followed waterfall development and more often than not, so did the deployment from one team to the next. Dev teams finished a project and marked it as code complete. At this point, Ops teams would then be responsible for actually getting it into production. Security teams were unfairly given a reputation for saying 'no' to everything and were maybe informed of this process at some point, but were often the last to know.

This operating model often pitted teams against each other with tension from one step to the next and between each of the teams. This lack of collaboration also led to long release cycles, cost overruns and ultimately delays to delivering new features and functionality that would scale and be secure. Each team had differing goals and collaboration suffered.



As businesses moved to the cloud, adopted a microservices-centric architecture, and began pushing the envelope on release frequency, this operating model started to change completely. Development and Operations teams began to work together in a DevOps model. Infrastructure as a service allowed for the popularization and widespread use of Infrastructure as Code (IaC). Resources no longer needed to be specified out months in advance, ordered and physically racked in data centers. Instead, programmatic APIs could be utilized to create brand new resources on demand. Those resources could be automatically scaled up or down.

Infrastructure could now be completely created and managed using code. IaC removed the friction and toil associated with teams manually provisioning and managing fleets of servers, databases, operating systems, containers and at this point, all infrastructure associated with software applications. Dev and Ops team are no longer separate teams, but rather working together to build and scale applications together.

Security as Code builds off the gains that these organizations have seen from IaC. Security as code similarly sees a migration to security and policy as code to remove the toil and friction associated with securing software in an IaC mindset. Security and policy as code began with standard software testing of areas like permission boundaries. These unit and functional tests were Security as Code before being labeled as such. Security as Code also arose out of the desire for automation from internal and external red teams and pentesters to automate all of the things. Known as DevSecOps or DevOpsSec, this methodology has become the way organizations can enable collaboration, agility and security, early and often across their entire infrastructure.

Security as Code Benefits

When moving to a Security as Code model, there are a number of key benefits that are realized across the organization. One of the key benefits and early drivers was fostering collaboration and enabling agility between and among Dev and Security teams. Another key benefit has been visibility for many teams across the organization. Finally, codifying both security and policy simplifies management and reduces toil across the organization.

Greater Collaboration: As Dev teams moved to agile workflows, Security teams were often left behind still operating in a waterfall methodology, being brought in at the very end. Dev teams were quickly iterating and ignoring or subverting security processes that hadn't been yet updated. Security teams that quickly recognized the benefits of agile methods started working directly with Dev teams to meet them where they were. This naturally led to collaboration when they both began to work on shared problems. No longer were they working on orthogonal problems with different motivations, they were working together, directly on the same code base, making sure tests passed before code moved to the next step.

Improved Morale: Another problem that arose in organizations was that many teams outside of Security and Compliance had very little visibility into their decisions. Dev teams hoped for an approval and were distraught with what seemed like constant no's. As security and compliance requirements become codified, there is no longer a question as to why a decision is made, it's clear from the code. For example, if you have integrated Kubernetes with Open Policy Agent (OPA), you can codify the users and groups that have direct access to each Kubernetes cluster. This allows you to set consistent policy that corresponds to service ownership instead of ad-hoc permission requests. If security is fully baked into your pipeline, there are fewer surprises and last minute blocks when it's code complete.

Increased Visibility: Security as Code helps simplify and centralize user and data access reducing toil and further providing visibility. Access and policy changes can now be tracked, and requests for changes can be self service. For example, you may be using Terraform to manage IAM resources for your cloud provider. By tracking IAM changes in source code, anyone can now see all permissions and can make a pull request directly to the Terraform repo to request changes. When you centralize your decisions to a declarative policy engine, you no longer need to make the same decision over and over again in separate systems. Long gone are the scattered policies of authorization to scattered applications.

Shorter Release Cycle: When you integrate security requirements early on in design and development, issues can easily be addressed resulting in increased velocity. Dev and Security teams are no longer trying to address minor to complex to systemic issues after a new feature or functionality is "code complete". With the advent of Security as Code libraries, application development can be decoupled from the fraught process of implementing your own custom authorization. For example, by integrating with OPA, developers can enable Role Based Access Controls (RBAC) in only the time it takes to enable the integration. Traditionally, this would have required multiple sprints from the Security, Product and Development teams to understand the requirements, what RBAC is, development time and finally full code review. Developers can focus on their core strengths and speed up application development. Additionally, as Security teams continue to adopt this approach, they will begin to adopt or develop their own libraries and tooling to further speed up releases by providing resources to ensure that applications are secure by default.

Better Security: When looked at holistically, each test, scan or policy that you can integrate, early, often and continuously, will find problems sooner so they can be addressed before others find them. Undertake this approach for all sorts of add-on benefits, but ultimately we're all in this together to better secure the data we all care about.

Security as Code with Cyral

The principles of Security as Code and API-first have been at the core of design and development at Cyral. We have embraced cloud-first, everything as code and API-first design to meet our customers where they are.

Our commitment to Security as Code starts first with building a security product that is developer friendly. We have designed our product to naturally fit into existing development workflows. Our application can be easily deployed as part of your testing, staging and production environments to enhance tracing and security at each step of the way. No matter your setup, we have developed options to fit your deployment requirements. We have focused on IaC options from Terraform to Helm and more to support your existing workflow.

To truly be able to expand to all existing workflows, we have built our product based on API-first principles. We recognize that Cyral is only one part of your existing toolset and so we have built out dozens integrations across the stack from notifications to logging to issue tracking and more. Cyral's focus will continue to be on data layer security and advanced data tracing across any number of data repositories available.

One of the key components of security as code is to integrate security directly into your CI/CD pipeline, bringing security testing directly and automatically as your application moves from code commit to production. For each step of the pipeline, Cyral will enable advanced tracing and consistent authentication and authorization. Cyral completely supports this model and recommends integrating Cyral in every environment to fully take advantage of our advanced security and tracing capabilities. Cyral's application comes with out of the box templates to support your IaC workflows and install our sidecar in your infrastructure, the way you deploy the rest of your infrastructure.

Cyral can be integrated into your CI/CD pipeline and can be deployed in dev, staging and production along with your application code, ensuring that all data layer activity from every application is automatically observed, controlled and protected. By starting with Cyral in your dev environment, users can now also measure and validate that data layer performance and control do not regress with each new release. Cyral's advanced tracing provides full visibility into what your users and your applications are doing, allowing you to triage and find issues quicker.

Cyral also utilizes Security as Code for data and access policy decisions. We have integrated with Open Policy Agent (OPA), the standard for "policy-based control for cloud native environments", as the basis for our policy engine. OPA allows our users to write declarative policy for granular access to data repositories and the data that is contained within them. Cyral users write their declarations for user and data access in YAML. In the backend, we then use this as a data input to our prewritten Rego queries to verify adherence to policy. By implementing it this way, Cyral remains performative and allows our customers to write configs in a markup language they're likely already using. YAML also encourages all levels of internal stakeholders to be able to review, edit and comment on policy based code. Writing policy as code with YAML means that you don't need to be an engineer to contribute.

Cyral Integrations

Cyral is fully committed to supporting Security as Code with our customers, and helping them improve their agility and reduce risk. Our client can have a policy repo in Github, so when they push a new version of policy to their repo, a Github Action is called to automatically update policy in their Cyral deployment. Any runtime changes in the application, as it gets promoted in the continuous delivery pipeline (for example, Spinnaker) from testing to canary to production, get tracked through metrics and traces generated by Cyral and routed to the team's monitoring and logging platforms, such as Datadog and Jaeger or an ELK stack. If any alerts are generated, they can then be sent to the messaging and issue tracking systems, such as Slack, Jira and Pagerduty. By integrating Cyral into your full pipeline, any risks and vulnerabilities are caught as soon as possible and all applications promoted to production by the Development team come with built in access control policies, which can be reviewed by the Security teams if necessary. Together, we have implemented a full CI/CD security and policy as code pipeline in production.

About Cyral

Cyral delivers enterprise data security and governance across all data services such as S3, Snowflake, Kafka, MongoDB, Oracle and more. The cloud-native service is built on a stateless interception technology that monitors all data endpoint activity in real-time and enables unified visibility, identity federation and granular access controls. Cyral automates workflows and enables collaboration between DevOps and Security teams to automate assurance and prevent data leakage. Cyral is venture-backed by Redpoint, A.Capital, Costanoa and SVCI. Follow the company on Twitter at @CyralInc.

Want to learn more about how Cyral can help your organization? Sign up for a tech talk with one of our engineers!

cyral.com/tech-talk