THE SECRET TO ACHIEVING A FASTER ATO

EBOOK SERIES

AUTHOR

MICHAEL SHIELDS

EDITOR

ANNA PATSEL-POWELL

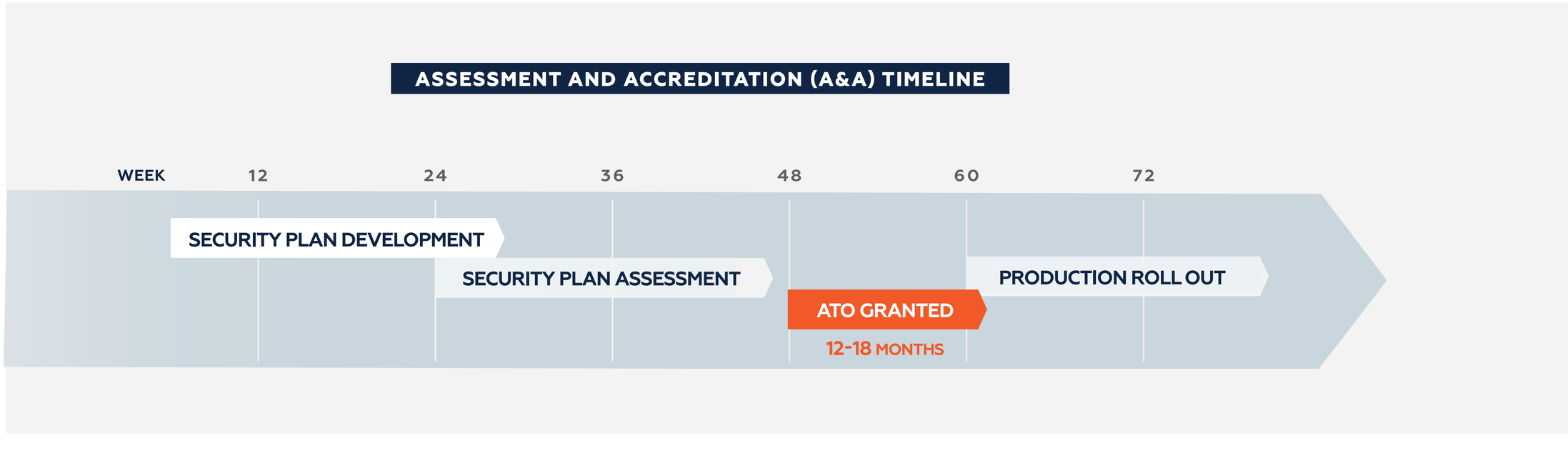
PYRAMID

SOLUTIONS THAT LAST

Imagine if your federal agency spent tens of millions of dollars and 2 years modernizing a complex, mission-critical system. Then, imagine that new, modern system sitting on the sidelines waiting for another 10 months to be introduced into the agency's environment because the Authority to Operate (ATO) certification has not been granted.

Unfortunately, this scenario happens more often than you may think, and many times agencies can't speed critical services to market fast enough. An Assessment and Accreditation (A&A) process routinely takes months or even a year before an ATO can be granted. ATOs are a requirement of the Federal Information Security Management Act (FISMA) in which Chief Information Officers (CIOs) must accept the security risks of each system in the agency's network.

One of the biggest hindrances to federal IT modernization is not actually capturing the funding or developing the technology — it's obtaining an ATO. Agencies are always looking for a more robust process to expedite the process for achieving compliance and securing an ATO without compromising any security requirements. The key to expediting the process is the adoption and use of a robust DevSecOps process.

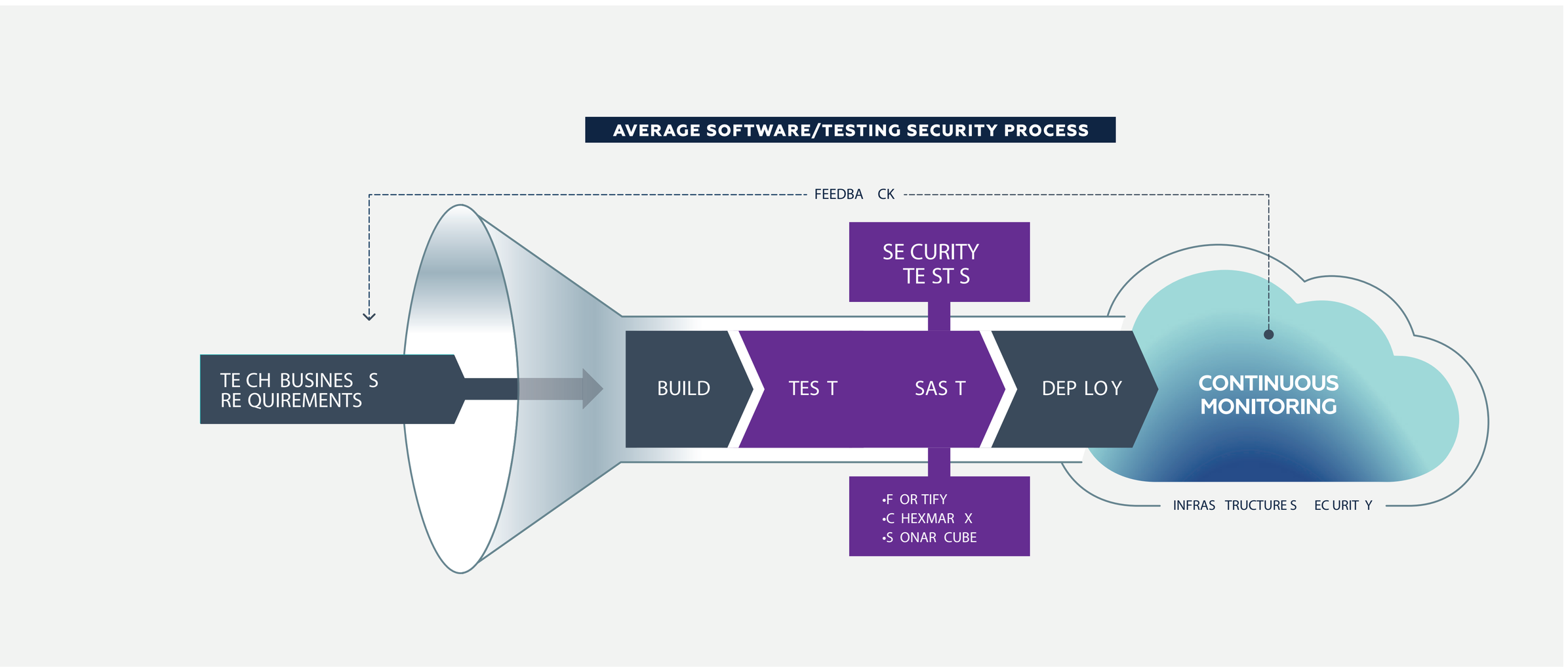


What Is a Typical DevSecOps?

Because the terms DevOps (a portmanteau of software development and information technology operations) and DevSecOps (which is DevOps plus security engineering) are terms that arose organically in industry and weren't coined by a centralized authority, no readily agreed-upon definition exists for what these terms mean.

DevOps got its start in the technology industry as companies, striving to meet consumer demand, needed to release new features and fixes continuously and reliably at a high velocity. The inclusion of "Sec" in the terminology signifies the importance of security and why risk mitigation needs to be addressed from inception and throughout the software development life cycle (SDLC). DevSecOps professes the need to instrument security controls in every phase, including architecture, application code, production environments, and beyond.

The traditional or waterfall engineering pipeline siloed developers and security teams. Engineers held onto a project and handed assets over to security only when the development portion of the project was complete. Gradually, these processes were scaled up and automated to meet the increased need for speed and eventually became DevSecOps. Now, most software development companies just use automated vulnerability scanning tools like Fortify to scan their source code once. Then, they throw their application and security reports over the fence to the security team. And that is how most companies define their DevSecOps.



What Should Your DevSecOps Be?

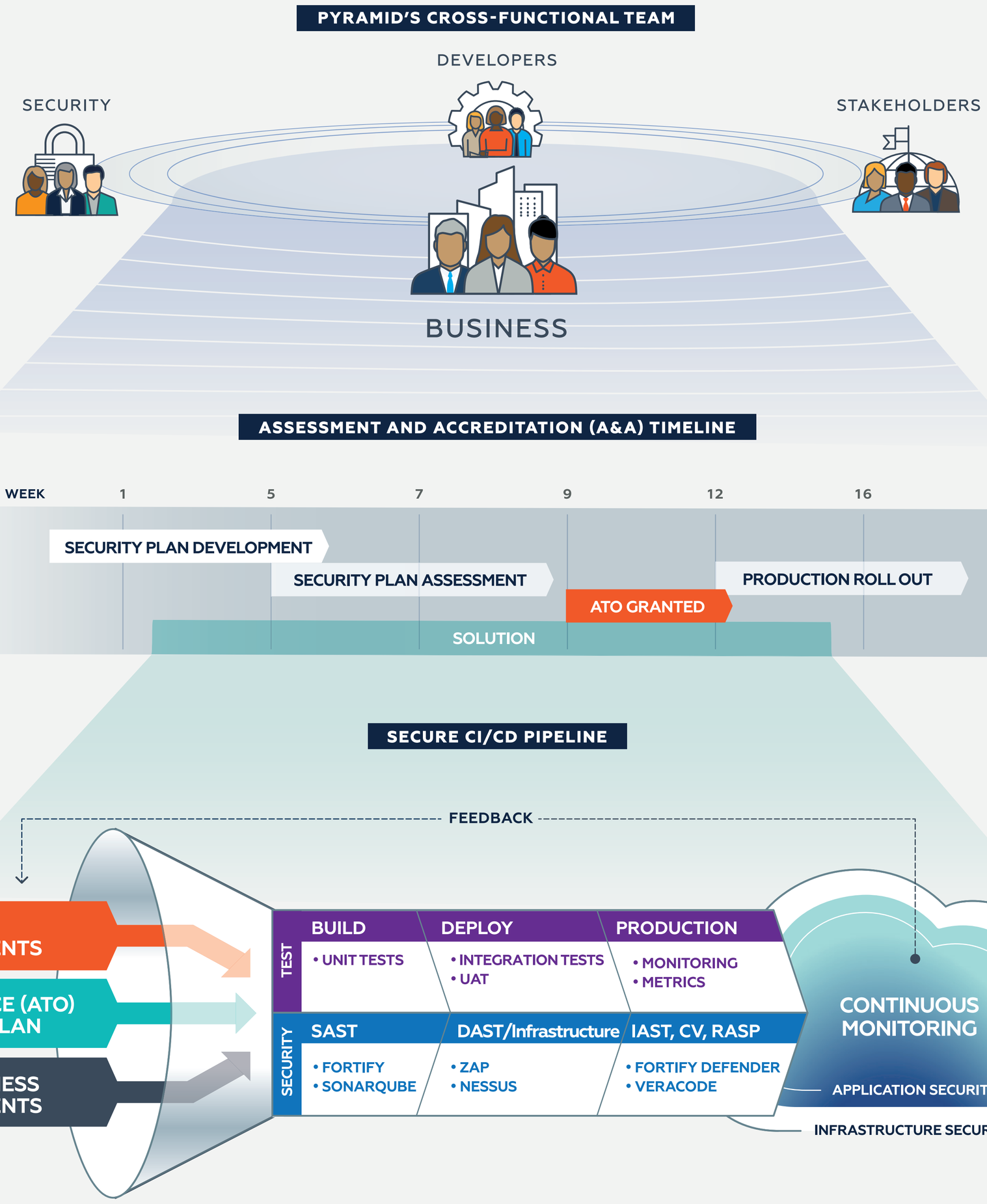
Conducting a one-source-code scan and calling it DevSecOps isn't enough. In 2017, 35,277 cybersecurity incidents were reported by federal agencies in the United States. The increasing digitization of federal agencies and their personally identifiable information (PII) data has made them a major target for acts of cybercrime and warfare. Because of these increasing threats, applications built for federal agencies should be fortified from inception, not just for a faster ATO (although it helps). Security needs to be an integral part of the enterprise architecture from the beginning of the design.

A robust DevSecOps pipeline should contain a mix of technology, processes, and people, but most pipelines nowadays focus exclusively on technology and focus less on the processes and people involved. The following sections will cover what a fortified DevSecOps looks like and how it speeds up gaining an ATO. These methods have been used for many of Pyramid's federal IT modernization projects successfully.

The Formula for a Fortified DevSecOps

Getting the all-important ATO for your software as quickly as possible requires a stringent DevSec-Ops process that aligns people, processes, and technology. Specifically, you must ensure you train your personnel, develop your security plan, test and remediate vulnerabilities during the development life cycle, and continuously monitor production for emerging security threats.

Pyramid's DevSecOps Practice



Personnel Training

Rather than shoehorning security into the code as a last-ditch effort to deliver a product that meets rules and regulations, it is important to train staff to have a security-oriented mindset from the beginning. The specifics will vary from project to project, but there are multiple aspects to consider when crafting a training program.

First, make sure engineers are trained on the fundamental security practices and controls that must be applied in the architecture, design, code, and configurations of an application. Ensure developers are trained on how to develop secure code that does not expose vulnerabilities classified by the Open Web Application Security Project (OWASP). Confirm that operations engineers are educated on how to encrypt data "in transit" and "at rest." These are just a few example topics that should be covered in security training. The team should also be trained on security tools that can be implemented and used to automate as much of the process as possible: source-code scans, runtime security protection, and security monitoring. They should also continue to keep up on modern coding practices.

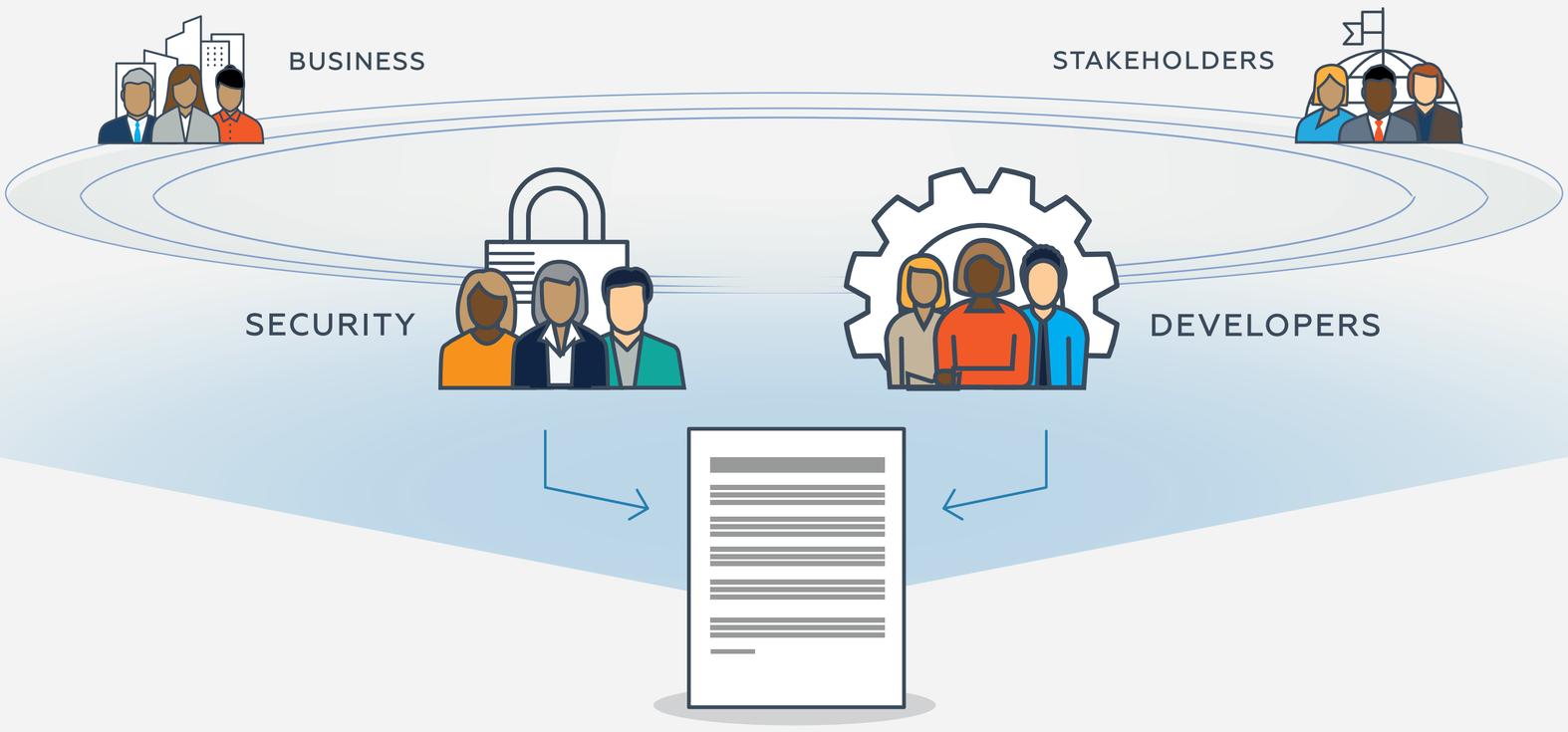
Second, everyone involved in the project should be made aware of what is needed by a specific customer. In addition to making sure that there is the necessary functionality implemented (e.g., single sign-on, multi-factor authentication), contributors should be aware of things like data handling, data needs, and other practices to ensure that things like PII or other sensitive details stay secure.

Security needs vary from customer to customer, and by identifying individual customers' needs ahead of time, projects can proceed at a quicker pace. Make the team aware of the required security tooling and include security tasks in the release backlog. The team can then prioritize and reliably implement the security needs as part of the development process, instead of at the end when the cost of security implementation is much higher.

Security Plan Development

Training staff to focus on security in all aspects of the SDLC is just the first step. The next step is to determine the specifics of the security needed for the application.

The DevSecOps team should lead the identification and development of a plan that ensures the client's policies and procedures will be accounted for and included during the multiple software architecture and development phases. One key source to start with is National Institute of Standards and Technology (NIST) Special Publication 800-53, which codifies the U.S. government's recommended Security and Privacy Controls for Information Systems and Organizations.



Though some agencies may call for stricter requirements, Pyramid Systems' prior experience indicates that all agencies need to meet the requirements set forth in NIST 800-53 at minimum. Furthermore, tracking of the controls specified throughout the SDLC is more manageable using an interactive security dashboard, which can reduce the overhead required to prepare the final deliverable for review when seeking ATO.

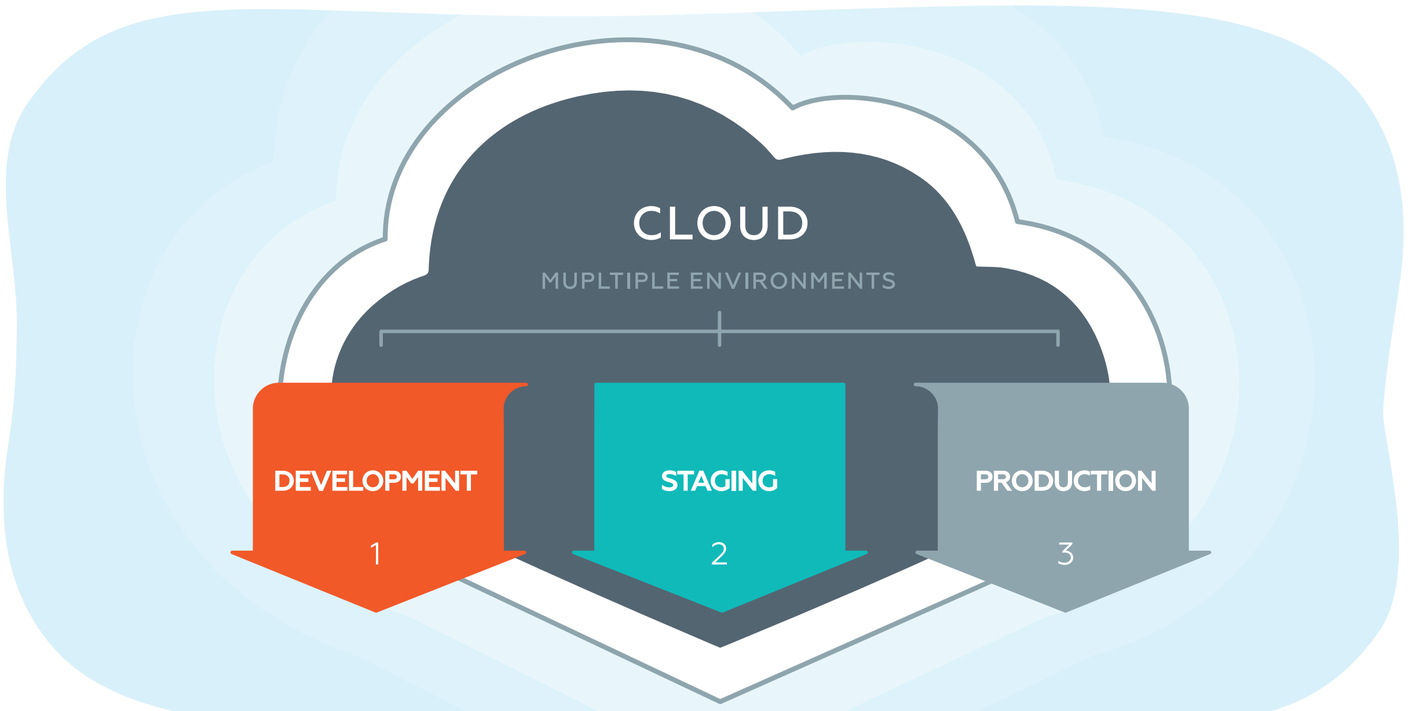
If possible, the tracking of such requirements should be automated. For example, tracking manually whether a solution meets the NIST 800-53 requirements and, more importantly, how it meets the requirements would take hundreds of Excel lines (with extensive text fields) or a 100-plus-page Word document. Not only are such files cumbersome to work with, but having a method that's easier to work with can also make it clear which requirements have been addressed and which ones haven't. Tools like Compliance Masonry (https://github.com/opencontrol/compliance-masonry) and GovReady-Q Compliance Server (https://github.com/GovReady/govready-q) enable teams to capture security control implementation statements, evidence, and artifacts in a structured machine readable format that can be easily exported to produce a System Security Plan (SSP) document.

This approach allows teams to develop the security documentation and assemble evidential artifacts in a collaborative and agile manner and manage them in source control tools such as GitHub. As new features and capabilities are implemented, the team is able to quickly update security implementation statements and produce an updated SSP in an automated manner that can be incrementally reviewed and used to support an ongoing ATO prior to every release into production.

The goal of creating a plan ahead of time and tracking individual components serves multiple future needs — for example, reducing any possible issues from insufficient security protocols downstream and allowing for rapid identification of vulnerable system components when new exploits are discovered. It is far less costly and time-consuming to implement as much security as possible early in the SDLC.

Programmable Security in Multiple Environments

Application development teams develop, test, and prove their applications in several non-production environments prior to promoting them to production. With the advent of Infrastructure as Code, Containers, and continuous integration/continuous development (CI/CD) practices, teams are able to provision, configure, and deploy to these environments on the fly in a predictable and consistent manner. This approach also eliminates the "cannot reproduce the production issue in development



environment" problems typically encountered by application teams. DevSecOps teams should strive to work in environments that are as close to their production environment as possible.

When application development teams develop, test, and prove their security compliance via scripting and code in lower environments prior to promoting it to production, typical risks of production deployments are reduced to an absolute minimum.

It is also recommended to automate security control requirements as prescribed by NIST 800-53. These would include but not be limited to encryption of application traffic through the network using Transport Layer Security /Secure Sockets Layer and encryption of application data "at rest" using cryptographic keys and certificates. The goal should be to create programmable security controls as much as possible in a modularized manner and ingrain them into the overall orchestration process.

Configuration management tools such as Puppet, Chef, Ansible, and Terraform can now fully automate the operational configuration, including the operational security configuration in both virtualized (e.g., VMware) and cloud platforms (e.g., AWS and Azure). Tools such as Chef InSpec go further in helping to test and validate security compliance, enabling "Compliance as Code" strategies. Teams can express a series of verification tests such as "is the web server listening only on port 443?" and execute those rules post-deployment to detect security deviations and authoritatively validate compliance.

These practices not only simplify and accelerate the A&A process necessary to achieve an ATO but also enable the team to mature to an Ongoing Authorization model proving security compliance prior to every release deployment.

Preemptive Security Testing

Automated testing is a critical DevOps practice teams adopt to build sustainable quality in their applications. These tests typically focus on functional and technical aspects of the application. DevSecOps advocates extending this testing practice to include application security. Static Application Security Testing (SAST) and Dynamic Application Security Testing (DAST) are two well-known web application security testing methods to detect vulnerabilities.

By integrating these security tests into CI/CD pipelines and adding these tests to the existing phase gates to interrupt the pipeline when failures or noncompliances are detected, developers and stakeholders can get a real-time view of the development process. Continuous detection of vulnerabilities as part of the day-to-day development process allows for faster resolution of issues and preempts such issues from leaking into higher environments.

A DevSecOps-based CI/CD pipeline interweaves security testing, instrumentation, and validation in every stage of the pipeline. A high-level synopsis is provided below.

1. BUILD

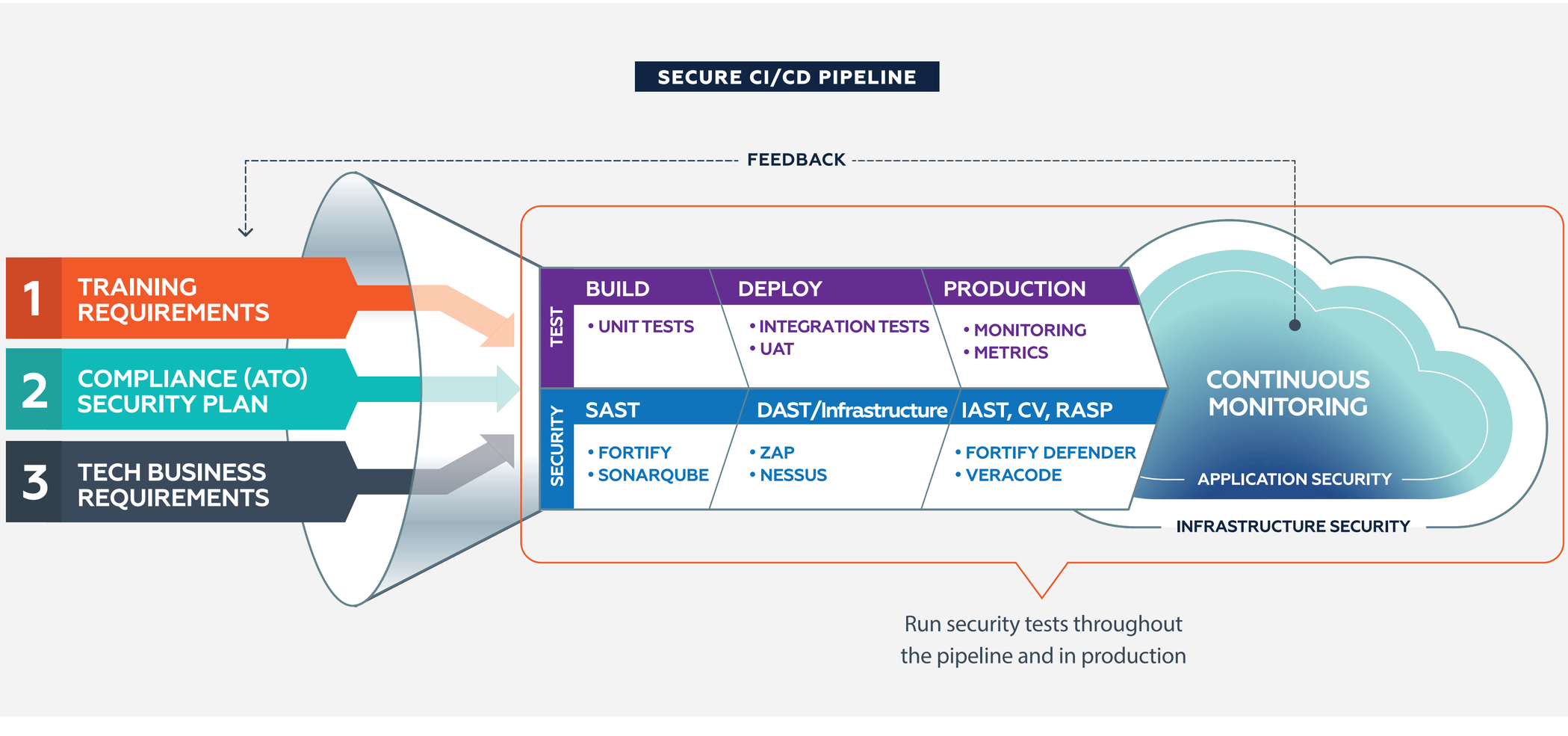
Compiles application code, packages into a deployable module, and runs series of static tests.

a. Unit Test – Tests components of the application as individual units.
b. Static Application Security Testing (SAST) – Analyzes application source code for patterns indicative of security vulnerabilities using well-known tools such as:
   - Chexmarx
   - Fortify
   - SonarQube

2. DEPLOY

Deploys application to environments and executes series of black-box tests against running application.

a. Integration Test – Tests application as a whole and verifies end-to-end functionality.
b. Dynamic Application Security Testing (DAST) – Simulates security attacks against application runtime and detects vulnerabilities using well-known tools such as:
   - Burp Suite
   - WebInspect
   - Zap
   - Veracode
   - Contrast Security
c. Interactive Application Security Testing (IAST) – Performs source code analysis when the application is running. IAST pinpoints precise code location and causes of vulnerability and recommends the correct remediation approach. Popular tools include:
d. Compliance Verification (CV) – Verifies the deployed configuration of the infrastructure, platform, and application conform to the security control requirements and implementation approach as per design. Popular "Compliance as Code" and "Policy as Code" tools include:
   - Chef InSpec
   - HashiCorp Sentinel
   - OpenSCAP



Run security tests throughout the pipeline and in production

Pyramid recommends using tools to collect the output of the above tools into reports summarizing the findings, providing the following benefits:

- Increasing transparency into the development and implementation process
- Amplifying confidence in the security of the solution
- Contributing to security reports that strengthen the A&A package
- Ensuring production environments are covered by runtime protection

Responsive Security in Production

Once the application is deployed and available for customers to use in production, the DevSecOps strategy shifts from preemptive to responsive security with the aim of protecting users, applications, and data from security attacks and breaches. Organizations should implement various operational security practices and tools to detect potential threats and defend against security attacks, including:

- Web Application Firewalls (WAFs) – Akami, Amazon CloudFront
- Intrusion Detection Systems (IDS)/Intrusion Prevention Systems (IPS)
- Security Information and Event Management (SIEM) – Splunk

While the above operational security practices are necessary, they do not fully protect applications from cyberattacks such as "zero day" that seek to exploit newly discovered security vulnerabilities. Runtime Application Security Protection (RASP) is an increasingly popular mechanism by which application code is instrumented with self-protection capability, enabling them to autonomously detect, block, and mitigate security attacks in real time. Some of the market-leading RASP tools include:

- Contrast Protect
- Fortify Defender
- Veracode

Leveraging Pre-Approved Tools

Modern software engineering has moved away from a monolithic architecture to lean applications that leverage one or more third-party services. One of the ways to get to a faster ATO is to lean on cloud-based services that have been FedRAMP authorized, such as AWS and Azure. Another strategy is to take advantage of platform-as-a-service (PaaS) and low-code providers — such as the General Services Administration's Cloud.gov, Salesforce.com, and Appian Cloud — that provide a suite of secure capabilities out of the box and can speed up development time and time to market and, more importantly, minimize the time and effort to secure an ATO.

The Benefits of DevSecOps for Faster ATO

The previous sections detailed a robust DevSecOps process that adds a lot of moving parts and pieces to a traditional DevOps pipeline. What was once an automation pipeline for deployment now includes personnel training; additional meetings to plan for security needs; and additional testing, both automated and manual.

With that said, Pyramid has found that such changes do result in numerous benefits, including:

- Finding security problems early in the software development process. If there are bugs and vulnerabilities, these are identified and fixed early in the process, where it is cheapest to do so. Furthermore, identifying these issues and fixing them ahead of time means that they aren't found by security teams, which speeds up the ability to achieve ATO.

- Delivering Compliance as Code. Since you built your security plan early, you can code into your DevSecOps pipeline a lot of the compliance, therefore automating parts of the process, strengthening your code from threats, and demonstrating that your solution meets all agency security requirements.

- Improving communication. Using a security dashboard to display information gathered during the development process keeps stakeholders informed and involved, which can assist in decision-making, preparing for approvals processes. This can also clarify the amount of time the DevSecOps team needs for security-related tasks to the engineering teams.

- Increasing the amount of security data to improve future releases. Building security monitoring into the application allows teams to gather information that can be used to reinforce security tooling during future releases. Continuous hardening of software is a process that can only proceed successfully when driven by metrics gathered by in-place instrumentation.

Conclusion

The A&A process is seen as a persistent barrier for IT modernization because of the regulatory framework that teams must navigate. But fortifying your DevSecOps processes through security-oriented mindset training, early development of a system security plan, mirroring the production environment for development, building test gate phases into the pipeline, and leveraging tools that are FedRamp authorized will ensure a faster ATO.

We've studied innovative ways our government project teams have been able to overcome the ATO purgatory challenge. This paper is the compilation and development of best practices based on numerous project successes.

If you are looking to modernize and secure your IT systems using a fortified DevSecOps process that results in a faster ATO, contact us today at info@pyramidsystems.com.

About Pyramid Systems, Inc.

Pyramid Systems is an award-winning technology leader driving digital transformation across federal agencies. We solve complex problems with advanced technologies and modern methodologies. By leveraging leading-edge cloud, analytics, and low-code platforms with Agile and DevSecOps approaches, we deliver highly secure, mission-critical solutions. Pyramid has professionalized innovation to deliver consistent results, reducing time from prototype to production to scale. We partner with clients to optimize for better citizen experiences, faster user adoption, greater efficiencies, and improved mission outcomes. Our unique culture is the main driver of why we build solutions that last.

Learn more at PyramidSystems.com.

Follow us on social media!