tenable.cs™
cloud security

7 HABITS OF HIGHLY EFFECTIVE DEVSECOPS TEAMS

WHITE PAPER

CONTENTS

7 HABITS OF HIGHLY EFFECTIVE DEVSECOPS TEAMS

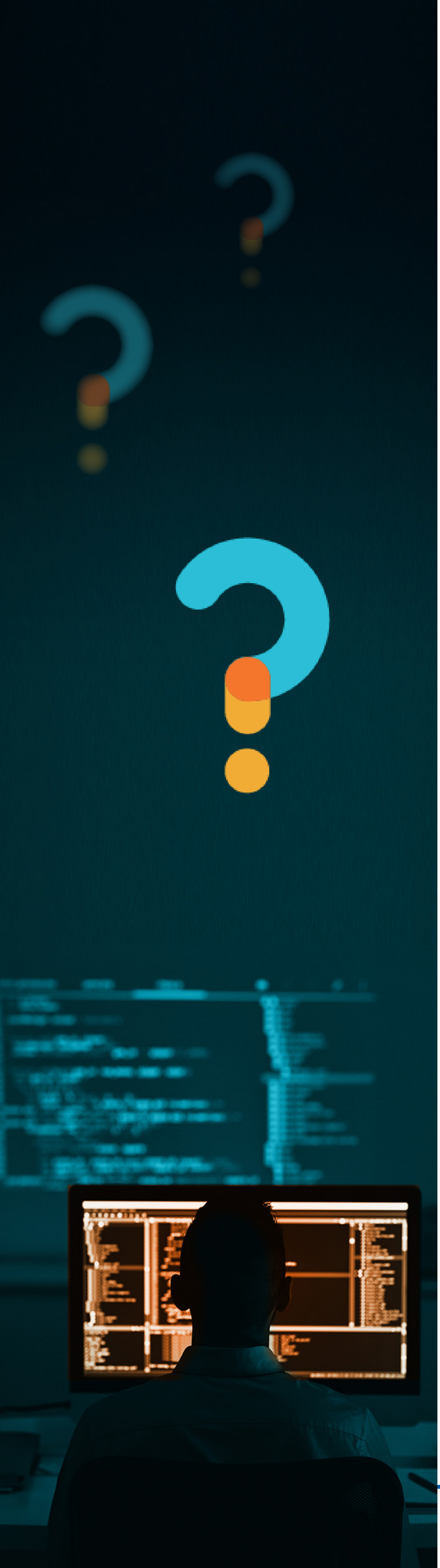
Organizations have been chasing the ideal of DecSecOps, which is commonly seen as an integrated team of development, operational, and security practitioners that can securely deliver innovation within a defined scope to market. While today's complex, dynamic cloud native projects demand this level of collaboration, most modern organizations still struggle to find a successful formula for DevSecOps.

The problem is that DevSecOps is not just a technology shift, but a cultural change as well. Tenable has helped organizations successfully implement DevSecOps by establishing seven habits that transform the way your teams collaborate and leverage their strengths.

Sadly, it's not as simple as seven small "changes" that are the magic bullet for success. In forming "habits," companies need to rethink, both organizationally and technically, how they securely build, deploy, and operate applications to be successful. The cloud has completely changed the dynamic of traditional IT with self-service access to cloud infrastructure.

This white paper describes these seven habits that address the technical, cultural, and organizational changes necessary to effectively manage security, compliance, and operational risks while accelerating the delivery of innovation to market.

DevSecOps is not just a technology shift, but a cultural change as well



THE NEED FOR DEVSECOPS

Right now, more than four million code commits are made in the cloud each and every day. That number represents a tremendous opportunity for mistakes and it also illustrates the demands that are being placed on cloud native environments and the teams that build and manage them.

Reliance on cloud services has exploded in the last 18 months, which is not a surprise considering the cloud's flexibility, speed and convenience. There are now more than 35 billion users and devices connecting to the cloud every day, and that number is expected to double in the next four years. What is surprising – and concerning – is that cloud security initiatives are not keeping pace with cloud adoption or with the increasing complexity of cloud native environments.

Development teams need to deliver at a high velocity in order to meet business objectives, but keeping up that pace can inadvertently introduce risks. Adding to the complexity, developers and DevOps teams aren't just writing code for applications; they are taking on infrastructure, deployment, and other automated processes as well.

Security teams are responsible for safeguarding assets in the cloud in order to maintain compliance and eliminate threats. Security breaches and malicious acts have resulted in more than 30 billion records being exposed in the last three years, and those are just the cases we know about. It does not seem to be a problem of failing to find the security risks; rather, it is a failure to effectively recognize and prioritize the most important risks and then collaborate with developers to deliver fixes.

Given that it only takes one bad code commit to create an opportunity for a devastating attack, the need for a concerted focus on cyber resilience is very clear. Organizations need to solve the DevSecOps puzzle, from both a cultural and technological perspective,if they are to improve security while maintaining high delivery velocity.

The seven habits address these challenges by delivering security solutions that enable developers to independently address security risks, with solid, simple fixes, within standard developer workflows. They improve collaboration within the team and catalyze the cultural changes necessary for effective DevSecOps teams that bring positive results across the entire organization.

CHALLENGES OF SECURITY IN DEVOPS

Each organization is different, but these failures often relate to difficulties assessing the risk of the myriad security findings, communicating the necessary fixes to development teams, and not bogging down the velocity of development in the process.

While cloud cyber resilience is more important than ever, most security tools are designed to be used in runtime – where risks are already exposed. It's very expensive to resolve risks this late, and any fixes applied in runtime will simply be overwritten with the next deployment. This begs the question of why teams are not addressing security risks during the development process when they are easier and cheaper to fix, and before they are exposed.

The majority of these security tools and processes rely on manual review and remediation, which not only slows development but also assumes DevOps teams have the security knowledge to fix issues as they are detected. The tens of thousands of roles that exist in some organizations makes manual processes infeasible. With just one security team member for every 100 developers in an average organization, the dependence on manual review and the concentration of security expertise in very rare individuals is not a recipe for effective security.

In fact, the sheer number of alerts created in a complex cloud native environment is overwhelming and, without any context for prioritization or remediation, most are simply ignored. Tenable research backs this up, finding that only 4% of detected alerts are actually addressed.

HABIT ONE: LEVERAGE IAC FOR REPEATABILITY AND DOCUMENTATION

As cloud native environments grow in size and complexity, it's important for organizations to have tools in place that allow them to improve the speed and consistency of deployments. At the same time, infrastructure needs to be secured throughout its lifecycle; from definition through runtime. Infrastructure as Code (IaC) delivers agility and reliability by letting you make changes to your environment in ways that can be tested, applied and audited. It also provides the important benefits of repeatability and documentation. As more environments are deployed, each with its own set of unique characteristics and standards, the relevance of these benefits when compared to manual one-off processes becomes clear very quickly. IaC delivers continuous versioning and review of your infrastructure definitions, making it much easier to understand how and why infrastructure is changing.

Implementing IaC doesn't just help DevOps teams deliver on business objectives quickly and at scale, it also allows security to be embedded much earlier in the lifecycle. The combination of these two factors will move your organization closer to achieving immutable infrastructure, a state in which infrastructure is never modified after it is deployed. Immutable infrastructure practices further improve teams ability to manage security and compliance of environments.

HABIT TWO: START SECURING EARLY, ENABLED BY IAC

According to Integration Development News, 80% of companies[1] in the United States have experienced a data breach in the last 18 months and 67% of those were attributed to cloud misconfigurations. Our existing security approaches are clearly not working, and common explanations include a lack of time to remediate problems as well as poor communication between security and development teams. Both of these problems can be addressed by involving security earlier in the process.

Threat modeling offers a powerful opportunity to proactively identify threats and establish security controls focused on those threats. Threat models can be built by analyzing IaC to determine what resources are being defined, how they are configured, and the relationships between them. This provides context that will allow you to understand how security findings may create a breach path into the environment, and the risk they represent. By starting this process with IaC, risk detection happens well before deployment.

Where threat modeling helps identify the threats to protect against, Policy as Code (PaC) provides a means to establish security controls for those threats. PaC can be integrated into your infrastructure provisioning process to detect and mitigate security and operational issues throughout the life cycle. In design, security and operational policies can be enforced before code is pushed to a source control repository. Integrating PaC into build pipelines detects violations and risks, which keeps issues from being introduced into the runtime environment. In runtime, PaC can detect violations and enforce compliance to policies, maintaining that secure posture. Integrating PaC tools into your development process lets you automate enforcement of security policy in your pipelines, which creates guardrails to help your team find and fix violations well in advance of deployment.

1 Idevnews | IDC Study Finds Cloud Data Breaches Impact 80% of CISOs,

HABIT THREE: SCAN IAC TO CATCH POLICY VIOLATIONS BEFORE DEPLOY

In addition to the operational benefits of IaC, it also enables you to detect misconfigurations and policy violations early in development and deliver real-time feedback to developers. By continuously scanning IaC as developers commit code in their repositories, you can find and fix violations without bogging down the development process.

Here again we see the push and pull between Dev and Sec teams. In the case of build pipelines, it's common for teams to leverage IaC's automation capabilities to provision and run pipeline resources. However, doing so creates risks to the integrity of the delivery process. If an attacker compromises the pipeline or the IaC that shapes the pipeline, the pipeline itself will automate the process of delivering the change to the production environment. This, in turn, gives threat actors the opportunity to deliver malware, access data, or totally compromise the environment.

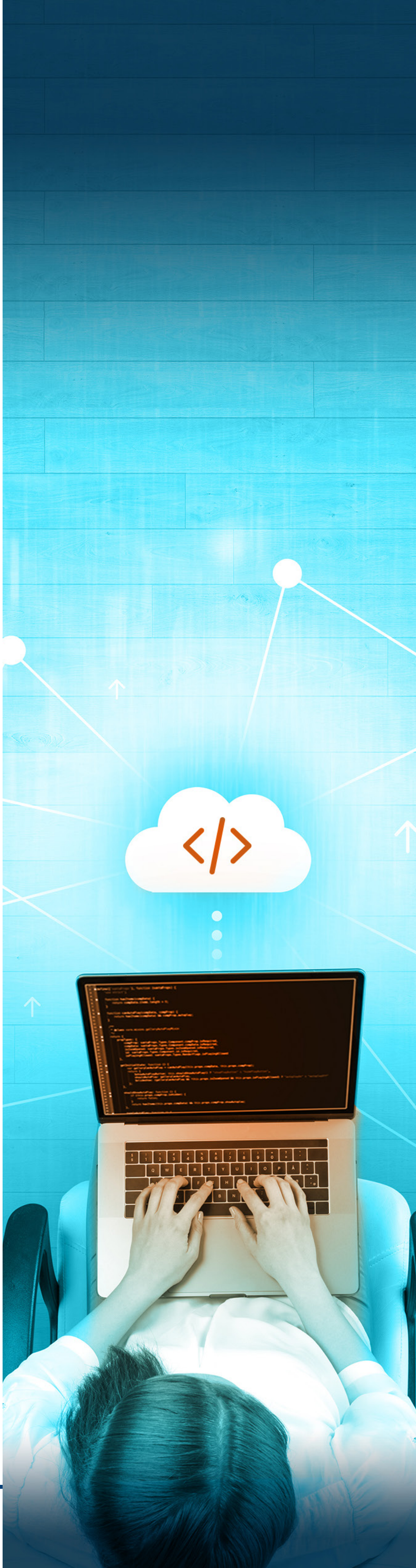
Identifying policy violations before deployment is the best way to safeguard against situations like this, and that means embedding security during development and enforcing it throughout the infrastructure's life cycle.

This paradigm is called immutable security and it is achieved by following three principles:

1. Securing IaC by mitigating risks before cloud infrastructure is provisioned.
2. Securing cloud infrastructure in runtime by detecting new resources and configuration changes that can introduce risk.
3. Eliminating risk posture drift by reconciling runtime changes against the baseline established through IaC.

To maintain immutable security you must ensure your IaC is the single source of truth. This requires continuous monitoring in runtime to identify configuration changes and assess them for risks. In the case of compliant changes, the IaC must be remediated to reflect that change and to establish a new baseline so that the change will not be overwritten during deployment. If the change introduces risk, the cloud infrastructure needs to be redeployed using the secure baseline defined through IaC, which will overwrite the risky change.

Immutable infrastructure and DevSecOps work together and provide no way for teams to accidentally introduce new risks into the runtime. If changes are required, they must be applied to the IaC, which is propagated through the pipelines and associated security controls to deploy the change.



HABIT FOUR: SCAN APP/CONTAINER CODE TO CATCH VULNERABILITIES BEFORE DEPLOY

The speed and simplicity that cloud containers offer makes them a popular solution, but the challenge is that the images, applications, and infrastructure increase the attack surface and present very real opportunities for security breaches. Just as you need to scan your IaC, it is also important to scan the code that will run inside the infrastructure.

SAST, DAST, and IAST are all used for application security testing, and there are pros and cons associated with each. SAST (static application security testing) can find vulnerabilities in applications as they are developed but can have a high rate of false positive or irrelevant findings. DAST (dynamic application security testing) can detect vulnerabilities in running applications but isn't able to identify their exact location in the code. IAST (interactive application security testing) delivers many of the benefits offered by both DAST and SAST, but the downside is that it requires specific language support.

There are several solid open source options available to help you maintain container integrity and security, including Semgrep (SAST), OWASP ZAP (DAST), and Clair (container scanning). Tenable is also excited about our integration with GitLab, which combines our understanding of IaC with GitLab's ability to find application vulnerabilities and automate DevOps processes. Creating a deeper understanding of the context and exploitability of such vulnerabilities provides you with the opportunity to improve how you prioritize and remediate, ensuring appropriate focus is given to the most important issues.

HABIT FIVE: CREATE AUTOMATIC PULL REQUESTS FOR VIOLATIONS

Earlier this year, ESG conducted a survey of nearly 400 IT and cybersecurity professionals and found that only 12% had not experienced cyber incidents targeting their cloud-native apps or infrastructure over the past year. Detecting the violations that lead to these incidents is obviously important, but it solves at most half of the problem. As we pointed out earlier, developers are not security experts. As a result, they need tools that will help them resolve risk without hindering development.

When a misconfiguration is identified, developers should be notified of the problem and given the code to remediate the problem through a pull request or merge request. Technologies such as Remediation as Code (RaC) can automatically generate these fixes and submit them directly to the repository where the offending code resides. Developers review the recommendation and merge the change into their code as part of their normal workflow. In doing so, the issue is addressed and resolved, the IaC is established as the secure baseline and the infrastructure is deployed, all without creating bottlenecks.

Embracing RaC allows you to do more than simply fix issues – it greatly improves your security posture. It also facilitates self-healing cloud infrastructure where misconfigurations are automatically discovered, then resolved quickly and efficiently without ever sacrificing development velocity. Because RaC is closely aligned with PaC, it works wherever PaC is being used including both development and runtime.

HABIT SIX: UTILIZE POLICY ENFORCEMENT DURING DEPLOY

One of the key problems when dealing with complex, dynamic systems is that it's difficult to predict early in development what you might encounter during deployment. Many organizations conduct security reviews in late stages out of necessity, but if problems crop up then out of band, often manual processes make it hard to reproduce and remediate them. This creates friction in the deployment process, which can manifest as impediments to the innovation pipeline.

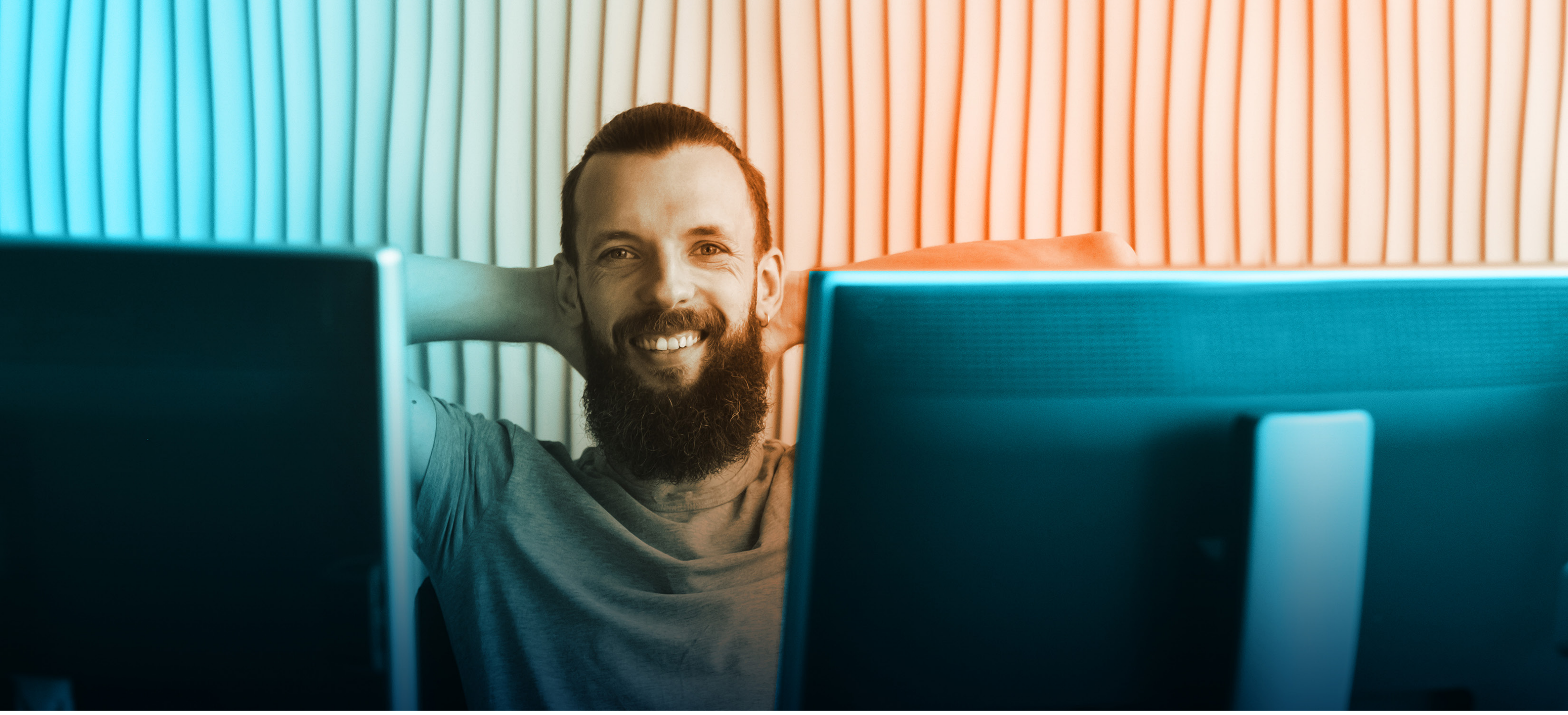
We've included habits to help ensure that vulnerabilities and violations are identified before deployment, and RaC eases the effort required to fix any problems. You can be confident that the artifacts you built are secure. Deployment processes typically use artifacts and the IaC to deploy into a runtime environment. But you can't simply assume that deployments will only use the previously secured artifacts; you should establish PaC controls to verify that artifacts and IaC are compliant during the deployment process itself.

HABIT SEVEN: UTILIZE POLICY ENFORCEMENT IN RUNTIME

Establishing a secure posture through IaC during development does not mean your work is done. Configuration changes to cloud infrastructure occur in runtime for a number of reasons and with very high frequency. In fact, we found that over 90% of organizations allow this to happen. While changes in runtime are sometimes necessary, they break immutability and cause the configuration to drift away from what was defined through IaC.

To maintain a secure posture, enforce the same policies in runtime that are enforced during the development and deployment processes. Ideally this enforcement will prevent configuration changes that are not compliant with the policies.

Continuously monitor the runtime environment for configuration drift and leverage Drift as Code capabilities to ensure any compliant changes implemented in runtime are propagated back to the IaC. Automating the propagation of safe runtime changes back to the IaC enables teams to preserve immutability and ensure that the runtime configuration always reflects the IaC. This helps to avoid friction in the deployment process because developers don't need to worry about overwriting important runtime configurations. If non-compliant changes are detected in runtime, immediately redeploy from IaC to eliminate the risky change.



CREATING HABITS FOR SUCCESSFUL TEAMS

DevSecOps teams today have the challenge of "staying in their lanes" while still effectively collaborating to achieve their company's objectives. This pursuit is further complicated by cloud native environments that are ever evolving and increasing in complexity, not to mention the growing demands being placed on those environments. Utilizing the seven habits above gives teams pragmatic, real world solutions from build through runtime, as well as a best practices framework that will allow them to innovate in the cloud with confidence. Tenable is here to support this effort with platforms and services designed to help organizations of all sizes achieve and maintain cloud cyber resilience. ■

ABOUT TENABLE

At Tenable, we recognize the value of embracing shift left security as a way for organizations to innovate in the cloud with confidence. We deliver an integrated, end-to-end security solution to help organizations better protect their cloud environments. It provides a complete picture of cyber risks across the modern attack surface, with unified visibility into code, configurations, assets and workloads. Learn more about Tenable.cs and how our platform enables DevSecOps with integrated controls for development and runtime workflows, focused on IaC.

tenable®