CYBERARK®

WHITE PAPER

SECURING PRIVILEGED ACCESS IN DEVELOPMENT ENVIRONMENTS – THREE CRITICAL USE CASES

Securing the Enterprise By Addressing Vulnerabilities In Developer Tools and Environments

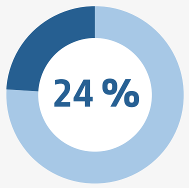
www.cyberark.com

Table of Contents

1 - Introduction: Securing Development Environments

Security and IT Must Prioritize Securing Development Environments

Digital Transformation is driving enormous change and innovation across many industries, both new and emerging. Innovations and new business models have forced decision makers across almost all industries to rethink how they do business. And today, rapidly bringing new products and services to the marketplace has become even more of a critical competitive priority than in the past.



24% flagged the greatest security risk to their organization was DevOps team not following security best practices.

CyberArk Global Advanced Threat Landscape 2019 Report

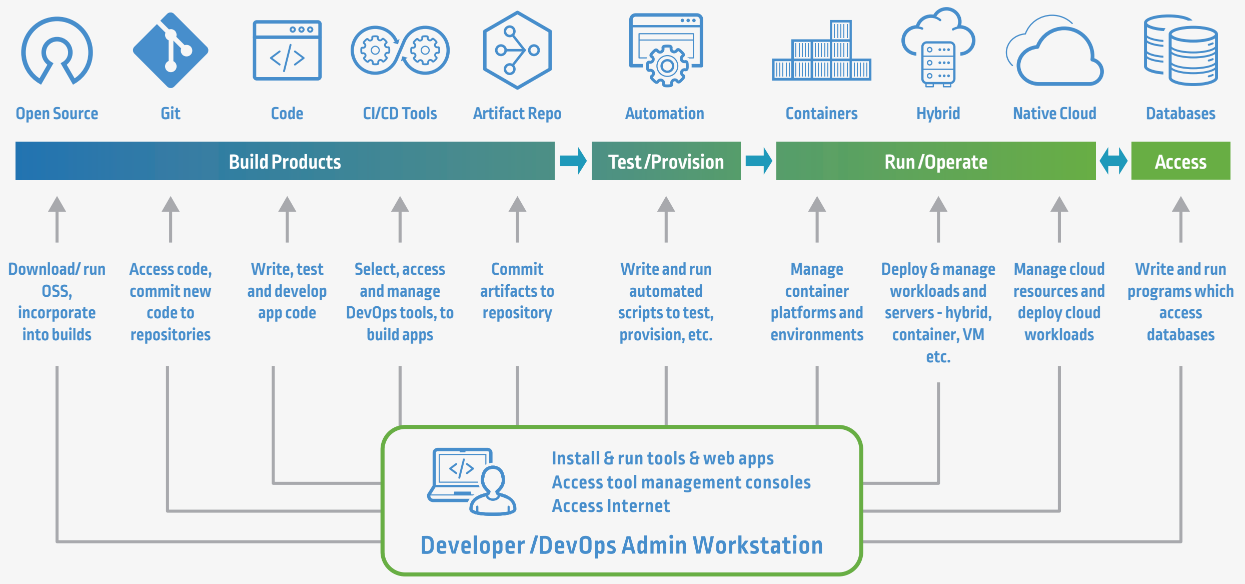
Enlightened enterprises recognize these challenges. Many are disrupting their own businesses to take advantage of digital technologies, not only defending themselves from digital innovators but leveraging digital transformation to prosper and advance their own market position to stay competitive and relevant. But, digital transformation is not just changing how organizations do business – it's also changing how they operate internally. For example, cloud infrastructure, with its ability to rapidly scale and the promise of reduced infrastructure costs, has been widely adopted. Similarly, automation tools have become widely used to streamline IT functions and improve efficiency.

Digital Transformation Is Shifting IT Security Practices And Priorities

With digital transformation requiring faster and more frequent deployment of applications and features, organizations are adopting new development strategies such as continuous integration and continuous delivery (CI/CD) and DevOps methodologies. Now as development teams become increasingly important to the business, developers are taking on increasingly important roles inside organizations. Developers are taking ownership and responsibility for certain IT security processes. For example, they are now managing access and credentials for development tools and platforms. Other traditional responsibilities are shifting away from IT security and toward development teams.

For example, IT functions such as provisioning and managing compute resources are more efficiently handled with automated tools such as Ansible. This infrastructure is often based in the cloud and the new tools and applications require coding and development expertise – very different than traditional IT admin skill sets. In too many cases, IT operations and security executives are finding that processes that once ran through their offices, or at least required their review and approval, are now happening outside their purview. As this transition continues, developers need – and are taking – increasingly powerful, expansive privileged access via applications, tools, consoles, etc. Some of these privileged roles may not be fully recognized or understood by the developers themselves, IT personnel or even the security teams. And while organizations are waking up to the new realities, attackers are exploiting these vulnerabilities.

Development Teams Use A Broad Range Of Privileged Credentials


Open Source
Git
Code
CI/CD Tools
Artifact Repo
Automation
Containers
Hybrid
Native Cloud
Databases
Build Products
Test /Provision
Run /Operate
Access
Download/ run OSS, incorporate into builds
Access code, commit new code to repositories
Write, test and develop app code
Select, access and manage DevOps tools, to build apps
Commit artifacts to repository
Write and run automated scripts to test, provision, etc.
Manage container platforms and environments
Deploy & manage workloads and servers - hybrid, container, VM etc.
Manage cloud resources and deploy cloud workloads
Write and run programs which access databases
Install & run tools & web apps
Access tool management consoles
Access Internet
Developer /DevOps Admin Workstation

But first let's recognize that development teams work with a broad range of tools and solutions as they build, test, deploy and run applications. In this example, developers use a lot of different tools and solutions – some open source, some public and shared. Many of these tools, such as Jenkins and Ansible, are effectively Tier 0 assets as they store massive amounts of credentials to allow the projects, playbooks and scripts managed by this software to access other tools, services and platforms. All these tools require very high levels of privilege.

The CISO Dilemma. How to Prioritize Security Without Impacting Developer Velocity

Development Environments Have Unique Requirements

In development environments, there are three broad use categories that require privileged credentials. First, the developers' workstations and laptops used in development environments to write code, build, test, provision, run, and operate applications require privileged credentials. Second, applications, scripts and other non-human identities use secrets and other credentials to communicate with other applications and tools, as well as to securely access databases and other sensitive resources. Third, there are the privileged credentials people use to access the admin consoles to manage and provision DevOps tools and platforms such as Jenkins, Ansible, OpenShift and Pivotal, and of course, the cloud consoles to access and manage the cloud resources. Each tool and cloud service has its own admin console, and each console is accessed not just by humans but also scripts, applications and other tools. All these privileged credentials must be managed.

However, development environments can be complex and unfamiliar to security teams. Today, security teams may not fully recognize the extent to which privileged credentials are being used in development environments, and how effectively these credentials are being secured, or not secured. Unfortunately, too often attackers are successfully targeting DevOps tools and platforms, such as Jenkins, Docker, and Kubernetes, to exploit unprotected credentials and gain access to the organization's protected data and other sensitive resources.

According to a 451 Survey it was:

Unclear which teams are responsible for securing CI/CD pipelines

Responsible for App security testing in CI/CD process

71%	IT Ops
61%	Operations Security Team
49%	Developers

https://www.synopsys.com/content/dam/synopsys/sig-assets/reports/devsecops-realities-opportunities-451.pdf

It is also not always clear who is responsible for securing development environments. Developers are likely focusing their energies on delivering new capabilities to customers, and only focus on security when they must. Consequently, it is essential that security understands the risks and appropriately prioritizes the need to secure their organization's development environments.

Today, as organizations continue their digital transformation journeys, as a best practice, they periodically reevaluate how the changing landscape for privileged credentials impacts their organization's attack surface so that emerging vulnerabilities can be prioritized and addressed. The CyberArk Blueprint is a prescriptive framework specifically designed for enterprises to leverage and address these changes while future proofing their environments.

In this whitepaper, we will consider the evolving privilege environment. We'll also explore use cases showing how security professionals can manage the privileged credentials used across development environments to better protect digital assets, data, and systems. Securing the privileges used in development environments is not only to protect development environments, but most importantly to prevent attackers moving laterally across the enterprise.

2 - Development Environments Are The New Risk Profile

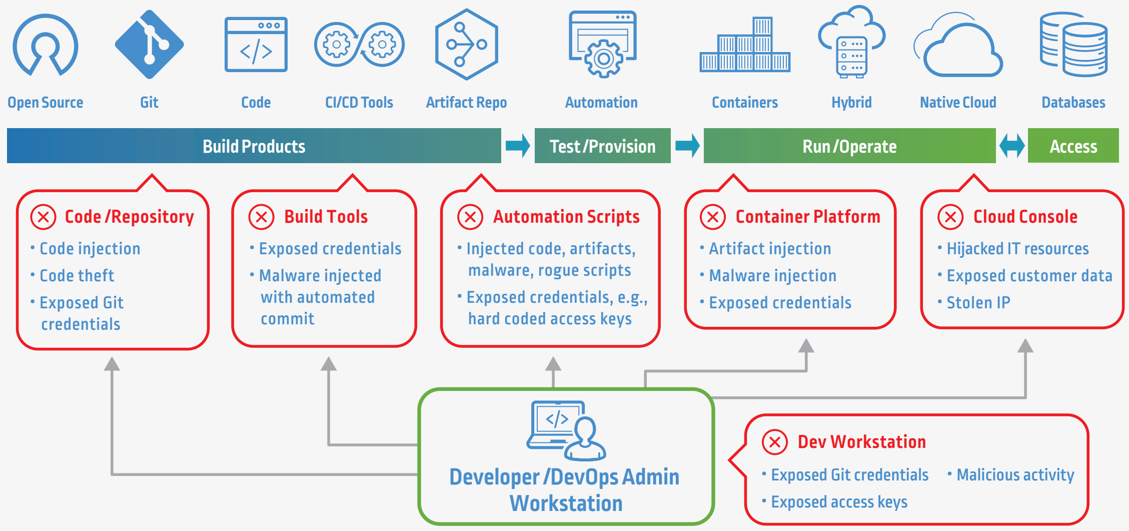
Developers Expand The Enterprise's Attack Surface

The trend toward greater independence for developers has introduced risks and cultural changes that create some unique security challenges. Specifically, the responsibility for managing risk is too often drifting from IT and security teams to development and DevOps teams.

Today's developer culture emphasizes high velocity, intensive sharing of code, ad-hoc tooling, and full-on automation. In this culture, low-security shortcuts often flourish, and traditional security processes are not always easy to integrate. For an organization to reap the rewards of short development cycles, the security team must work with their Dev and DevOps counterparts to follow cyber-security best practices, including securing privileged access.

Unfortunately, there are lots of potential areas where vulnerabilities can leverage or expose privileged credentials – each potentially creating havoc at various stages of the application development and deployment process.

What Could Possibly Go Wrong?



The diagram above shows potential examples of vulnerabilities, ranging from code injection and exposed credentials, to hijacked IT resources, stolen data and code theft. Development environments are complex and there are simply too many areas where credentials can be, and are, inadvertently exposed. For example, while code repositories are an essential part of the development process, credentials can be inadvertently exposed by making code public and malicious code can be included in builds. The organization's code and IP can be stolen from repositories if credentials are compromised. Cloud access keys must be protected wherever they are stored on the endpoint, in code, in scripts and applications, etc. Additionally, once code is deployed in hybrid environments, operations and even some developers may need to connect to on-premises environments, such as legacy Unix systems, to troubleshoot. This exposes another potential threat.

DOCKER HUB BREACH EXPOSES ~190,000 USERS, APRIL 2019

- Attackers stole up to 190,000 usernames and hashed passwords
- Also exposed Bitbucket and Github access tokens for Docker autobuilds
- Tokens enable developers (or attackers) to modify code and automatically build images

Docker hub is the widely used registry for images used by Docker implementations
https://www.theinquirer.net/inquirer/news/3074793/docker-hub-breach

KUBERNETES SECURITY IS IMPROVING, BUT HAS EXPERIENCED MULTIPLE BREACHES

- June 2019: Security flaw allowed attacker to use an infected container to replace files on users' workstations. Similar to prior March 2019 flaw
- Dec 2018: Flaw allowed full admin privileges to any node running on the Kubernetes cluster
- Feb 2018: Hundreds of Kubernetes Admin consoles found without password protection, which exposed AWS credentials

https://www.sdxcentral.com/articles/news/latest-kubernetes-security-flaw-linked-to-incomplete-patch-of-past-flaw/2019/06/
https://www.theinquirer.net/inquirer/news/3074793/docker-hub-breach

3 - Use Cases And Customer Examples

While each organization's digital transformation journey and development environment is different, the following common use cases will almost certainly need to be addressed to help ensure development environments are secure.

Use Case 1: Secure Developer Workstations

While it's easy to take for granted, most developer work, especially coding tasks, takes place on a workstation, whether a Mac or Windows machine. Developers need access to privileged credentials from their laptop to access platforms and services such as Kubernetes and Git, or a Jenkins admin console. Some of these processes and systems require a high level of privilege, if only for some specific actions and for limited time periods. Yet too often these credentials are saved locally, making developers' workstations high-value targets for attackers. These credentials are vulnerable to something as simple as a phishing email, which provides the attackers an entry point to access the developer's credentials. Because of these vulnerabilities, developers' workstations are extremely important to secure. However, developers typically need to get new features and capabilities out to customers quickly. They don't want to be slowed down by the burden and overhead imposed by conventional security practices, so they can be very creative in developing workarounds. And if security restrictions become too burdensome, it can be more difficult to retain developers.

During today's fluid and concerning time, many (if not most) developers are working remotely. Security is top of mind for every organization as they work to keep developers safe while maintaining business continuity.

Following these best practices will increase the security of the organization's developers and endpoints.

SECURITY BEST PRACTICES: DEVELOPER WORKSTATIONS

Establish security best practices and processes for securing developer workstations by focusing on endpoint privilege:

1. Remove Local Admin rights from the endpoint and apply the principals of zero trust and least privilege to elevate privileges only when necessary.
2. Use MFA (Multi-factor authentication) when possible. MFA is designed to increase account security by requiring multiple forms of verification to prove the user's identity when signing into an application.
3. Control installation of apps and open source tools. Use install privileges to block unknown apps and whitelist approved tools. Remove unapproved software and flag violators.
4. Secure local credential stores, such as Chrome browser, (i.e., protect default locations where cloud access keys, such as AWS Access Keys and Git credentials, are stored on endpoints.)
5. Block malware by preventing installation of unknown apps.
6. Review restrictions on a regular basis with the goal of minimizing privilege while avoiding overly onerous restrictions which inhibit developer productivity.

Customer Example: Secure Developer Workstations

TIER 1: Top tier for power users	**Highest requirements** • On demand privilege elevation • Self-service with management approval • Only permit installs that meet policy • Flag violators	**10%** of Workstations
TIER 2: Middle tier-most users	**Some special requirements** • Edit environmental variables • Edit host files • Use Visual Studio	**50%** of Workstations
TIER 3: Lowest tier	**Basic user with few special requirements**	**40%** of Workstations

A leading car rental company wanted to accelerate bringing new applications to market while securely protecting hundreds of developer workstations. The security team recognized that the organization faced a competitive environment hiring and retaining skilled developers and was concerned that if the security approach was too restrictive, it would inhibit developer adoption, impact developer velocity, or risk losing developers. None of which were acceptable to either security or the business.

The team used CyberArk Endpoint Privilege Manager to secure the various developer workstations and deployed the solution using a tiered approach, which applied the principal of least privilege to groups of workstations. For example, as not all developers needed the highest level of access and capabilities, such as installing and running new tools, the organization managed privilege at each endpoint with three tiers of access. Each tier was based on developer roles to maximize productivity and user experience. This was effective, as only a small number of developers needed their workstations to have the highest level of access to do their jobs efficiently. The tiered approach drove a successful outcome, which achieved both developer and security objectives.

Use Case 2: Secure Application Credentials

Applications, scripts, and automation tools need credentials to securely access databases and other sensitive resources. Additionally, the typical DevOps pipeline comprises a diverse collection of development, integration, testing, and deployment tools. These tools need to access other tools and resources – from code repositories to databases and cloud environments. Hard coding credentials in applications, scripts, and other sources is a major vulnerability as the credentials cannot be easily rotated, monitored, or tracked. Hardcoded credentials, including cloud access keys, are also easy to share inadvertently and can be stolen if the code is posted to a publicly available online source code management systems such as GitHub.

Additionally, many development and test tools, configuration management platforms, and service orchestration solutions have some form of native (built in) capability to "secure" the privileged credentials they use. However, not only do these capabilities have varying levels of maturity (for example, some may rotate, some may not), they are generally unable to securely share secrets with each other. This often results in "islands of security," or isolated subsystems that don't facilitate interoperability with other tools and/or aggregation of security policies, management, and audit data. Secrets scattered across, and even duplicated in, multiple tools makes tracking and managing these secrets cumbersome (at best) and impossible over time (at worst).

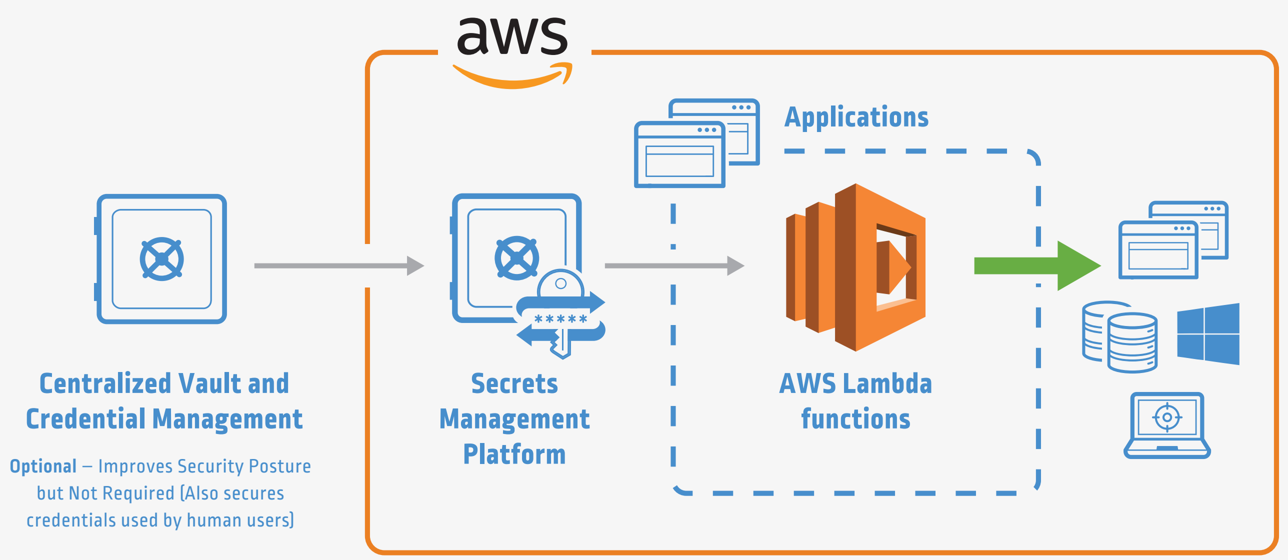
SECURITY BEST PRACTICES: APPLICATIONS, SCRIPTS AND NON-HUMAN IDENTITIES

Establish best practices for securing the credentials used by applications, scripts, and other non-human identities to securely access databases and other sensitive resources:

1. Eliminate all hard-coded credentials used to access databases, tools, and other sensitive resources. Replace with a secure approach, such as an API call to fetch the secret from a digital vault.
2. Centrally rotate, manage, store, and monitor secrets and other credentials used by applications, scripts, and other non-human identities. Ensure audit and compliance needs can be met.
3. Strongly authenticate applications, containers, and other non-human identities requesting access to secrets and credentials. Whenever possible use the native attributes of the requestor for authentication to eliminate the secret zero issue.
4. Apply principles of least privilege and role-based access controls (RBAC) so the application, container, script, or automation process only has access to the credentials it needs.

A secrets management solution solves these problems by centrally managing, rotating, and monitoring secrets and credentials in a centralized vault. Typically, when the application needs to use a credential to securely access a database or other sensitive resource, the application first authenticates to the secrets management solution, which then enables the application to fetch the privileged credential necessary to access the resource. Typically, the secrets management solution provides a robust set of capabilities to ensure the credentials are secure and can only be accessed by authorized applications. The system rotates and manages secrets according to policy and provides monitoring audit and other functions necessary to meet audit and compliance requirements.

Customer Example: Secure Application Credentials


aws
Applications
Centralized Vault and Credential Management
Optional – Improves Security Posture but Not Required (Also secures credentials used by human users)
Secrets Management Platform
AWS Lambda functions

As part of its digital transformation, this SaaS application vendor wanted to manage and secure the credentials used by AWS Lambda functions (AWS' serverless computing model) to access databases and other resources. A key deployment barrier was that developers wanted to use hard-coded credentials, which they perceived as both fast and easy. Meanwhile, the security team wanted to rotate and manage credentials to strengthen security. The solution was to select a secrets management platform that made it easy for developers to code their applications, while also making it easy for security to monitor, manage, and rotate credentials based on policy.

Now when applications running as AWS Lambda functions need to securely access a resource, such as a database, the application executes an API call. The secrets management platform first authenticates the Lambda function based on the application's native attributes and then provides the required credential to the application to access the resource. The credentials can be rotated, monitored, and managed as needed by the security team, while the developer simply includes an API call in their code instead of a hard-coded credential. The result is a win-win for both the developer and for the security team. Additionally, for an audit or compliance request, audit information is readily available to security, freeing developers from this burden.

In this deployment example the customer used CyberArk Application Access Manager's Dynamic Access Provider for secrets management. Additionally, the customer decided to integrate the secrets management platform with CyberArk's Core Privileged Access Management Solution to centrally manage both the credentials used by human users (including developers accessing tool consoles) and the credentials and secrets used by applications to securely access sensitive resources. By centrally managing credentials using the same platform, the organization can consistently manage credentials across the enterprise to improve the organization's overall security posture. This approach is popular with organizations as it greatly improves the security posture with zero impact on developer velocity.

Use Case 3: Secure DevOps Tool Administration Consoles

Widely used tools and developer platforms – including CI/CD pipeline tools such as Jenkins and Azure Pipelines; provisioning tools like Ansible and Puppet; and container-orchestration environments like Red Hat OpenShift, Kubernetes, and VMware Tanzu provide extraordinary control over an organization's development resources.

As usage surges, these tools' administration consoles have become increasingly popular targets for attackers, who can exploit unsecured consoles to target other resources in the developer value chain. Additionally, far too many organizations use the default configurations of these tools, which in some cases do not require passwords for privileged access. Attackers use bot crawlers to systematically search out and exploit these exposed instances. While the vendors and open source communities offering today's popular developer tools have worked to address these vulnerabilities, security weaknesses remain.

SECURITY BEST PRACTICES: TOOL ADMINISTRATIVE CONSOLE

Establish security best practices and processes for tool console access by humans, scripts, etc.:

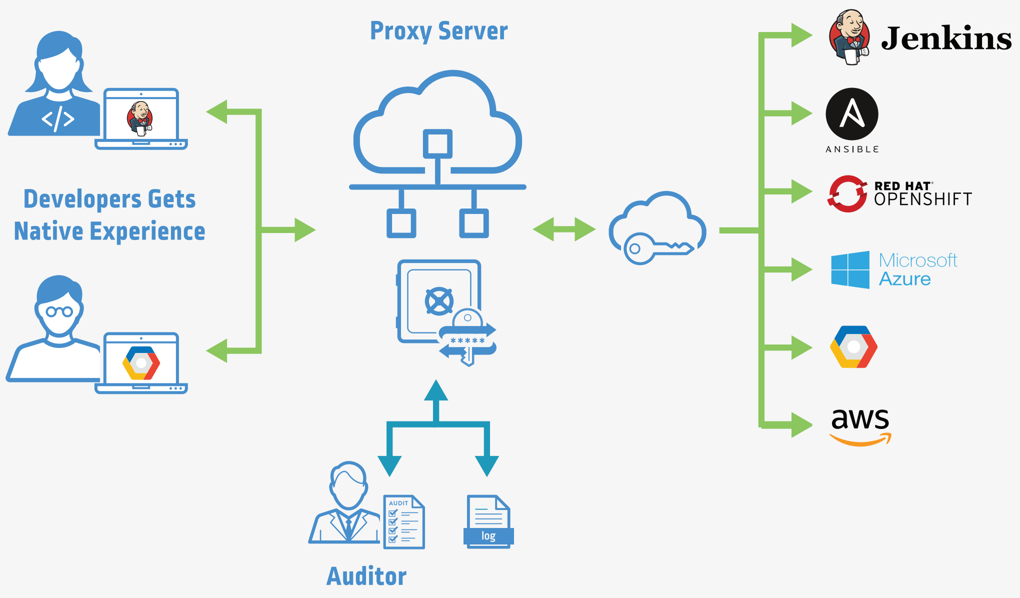
1. Apply basic password best practices (manage, rotate, etc.), and use MFA for critical operations. Ideally use MFA for all console access. Avoid the use of default configs and passwords. For example, at set up some popular tools establish a developer default user to create projects, while some CI/CD tool consoles can be accessed using http or with the default password.
2. Centrally control and manage human and other interactive access to tool management consoles and Command Line Interfaces (CLI), such as securing access to the Jenkins UI (User Interface) and UIs for other tools in the CI/CD pipeline.
3. Transparently manage and monitor sessions and make it easy for developers to adopt secure solutions by providing a native experience. Ideally limit the attack surface by only providing access to the CLI via jump servers and other monitoring tools.
4. Use MFA to prevent sensitive scripts running automatically by forcing human intervention to review or approve actions, such as pushing commits to Git.

SECURITY BEST PRACTICES: DIRECT ACCESS TO SYSTEMS

The best practice is **not** to provide direct access to production systems, but in some cases it may be unavoidable.

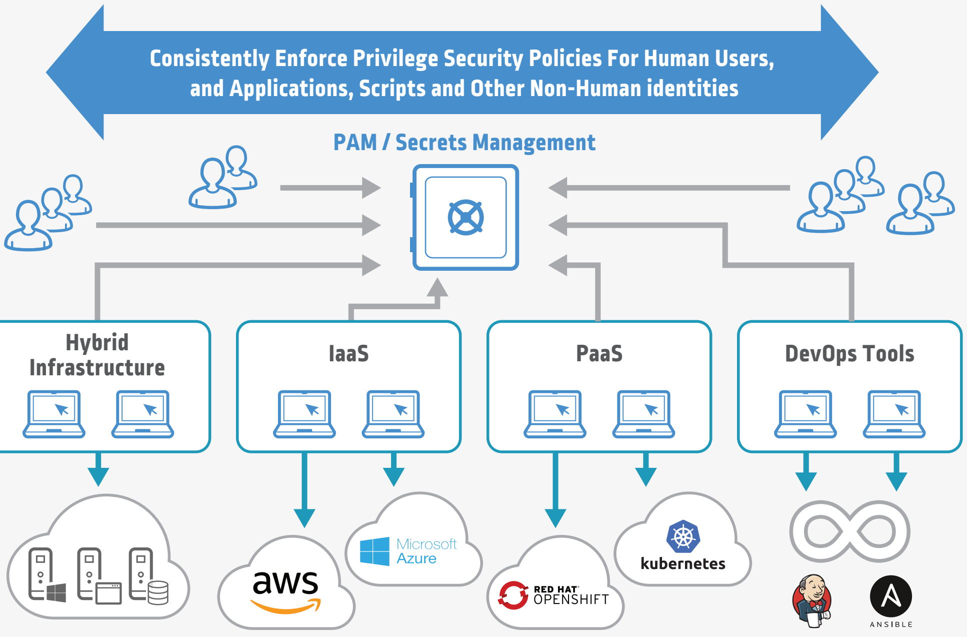
1. Segregate teams to ensure developers do not have access to productions systems.
2. Ensure any access to systems is from a jump server – enabling activity monitoring and preventing password exposure. If this is not possible and the connection is from a workstation, immediately rotate passwords.
3. Enforce least privilege principles. For example, in hybrid environments that require access to Linux and Windows servers, ensure elevation of privileges is limited by using a just-in-time approach to protect the host.

Customer Example: Secure Web Access To Developer Tools



A global bank was concerned that thousands of developers spread across multiple teams and geographies were accessing web-based management consoles for dev tools and cloud platforms without being consistently secured and monitored. They wanted to secure developer access to these consoles without impacting velocity and were concerned adoption would be limited if developers were given anything but a native experience. So, the security team identified a monitoring solution, CyberArk's Privileged Session Manager for Web (part of the CyberArk Core Privileged Access Security Solution), which can provide developers with native access to CI/CD tool consoles. The team then developed a deployment plan that focused on winning over influential development teams and then leveraging their initial successes with a phased rollout.

Customer Example: Secure Tool Admin Consoles And Application Credentials


Consistently Enforce Privilege Security Policies For Human Users, and Applications, Scripts and Other Non-Human identities
PAM / Secrets Management
Hybrid Infrastructure
IaaS
PaaS
DevOps Tools
aws
Microsoft Azure
RED HAT OPENSHIFT
kubernetes
ANSIBLE

As part of their digital transformation journey, a global publishing company wanted to improve its overall security posture with a holistic solution that could consistently secure all the privileged credentials used by both human users as well as applications, scripts and other non-human identities. In addition to IT functions the organization wanted to secure the credentials used by users who required access to the administrative console for DevOps tools, including Jenkins and Ansible, as well as the cloud management consoles. However, the organization faced resource constraints and multiple priorities. The security team needed to ensure their initial investments aligned with their overall security roadmap and would be meet the enterprise's long-term needs. The organization made the decision to deploy the solution in phases, addressing the most vulnerable areas first, while moving toward its long-term goal – the ability to apply the same security policies across the enterprise for both human users and non-human identities, regardless of the compute environment and development tools being used.

4 – Deployment Example: A Holistic Approach With All Three Use Cases

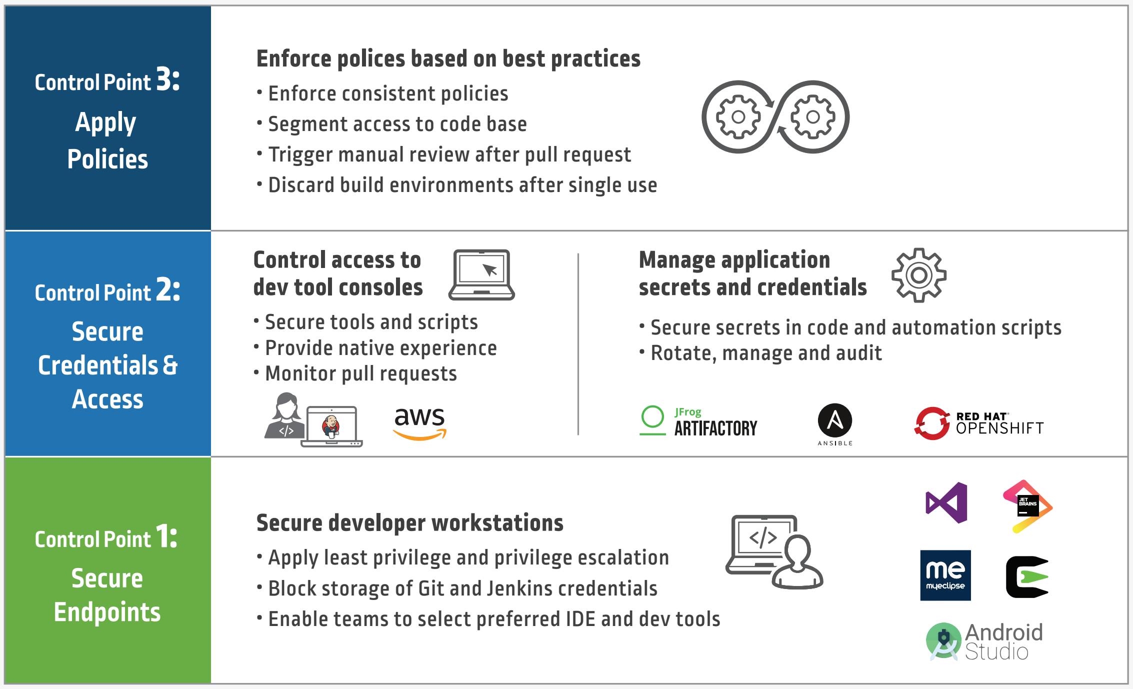
This scenario incorporates each of the three use cases explored earlier: securing the developer workstations, securing the secrets and credentials used by applications, and securing human access to the management console.

ADDITIONAL SECURITY BEST PRACTICES: DEVELOPMENT ENVIRONMENTS

Minimize potential exposure by taking a holistic, systematic approach to securing development environments in addition to the best practices covered in the prior use cases. To do this, analyze potential vulnerabilities across the entire pipeline and tool chain and identify best practices for the overall environment and each tool:

1. Prioritize security requirements such as preventing code theft vs. loss of cloud access and application keys, and establish security procedures which meet these priorities.
2. Enables support of multiple development environments to meet each development and project teams' unique needs but also enforce consistent policies.
3. Structure build processes and approaches to incorporate security in the initial design, such as segmenting access to the code base, triggering a manual review after a pull request, and discarding build environments after use to prevent credentials being shared from a prior build.
4. Block storage of Git, Jenkins, and other credentials on developer workstations.

Deployment Example: Secure Full Development Environment



Control Point 3: Apply Policies	**Enforce polices based on best practices** • Enforce consistent policies • Segment access to code base • Trigger manual review after pull request • Discard build environments after single use
Control Point 2: Secure Credentials & Access	**Control access to dev tool consoles** • Secure tools and scripts • Provide native experience • Monitor pull requests **Manage application secrets and credentials** • Secure secrets in code and automation scripts • Rotate, manage and audit
Control Point 1: Secure Endpoints	**Secure developer workstations** • Apply least privilege and privilege escalation • Block storage of Git and Jenkins credentials • Enable teams to select preferred IDE and dev tools

The software vendor took a holistic approach to secure sensitive development environments as the team progressed on their digital transformation journey and increased focus on developing SaaS based applications.

The developmentn team partnered with security to establish their priorities and objectives. They established that preventing the injection of malicious code and artifacts into solutions used by customers was their most critical priority. Code theft was a secondary concern. They also wanted to leverage their Jenkins CI/CD pipeline processes, as well as automation tools such as Ansible. Not surprisingly, developer adoption was a concern: development leadership wanted to largely retain existing developer workflows, while avoiding restrictions that would inhibit developer adoption.

The team analyzed potential vulnerabilities across the entire pipeline and tool chain and identified best practices for the overall environment, and by tool, to minimize potential exposure. For example, they segmented the code by development team and used MFA to force human intervention and review of pull requests (to prevent manipulation of build scripts to inject malicious code or artifacts into the code base – a top concern). The team used endpoint protection to secure the developers' workstations, privileged access management (PAM) to secure human access to the tool admin consoles, and a secrets management solution to secure application secrets and other non-human identities.

Initially the development team found some of security processes too burdensome and pushed back on security. To help overcome this challenge, security and development leadership systematically reevaluate and adjust developer restrictions to ensure security policies are met without over-burdening the developers. The result has been very successful with the Dev teams successfully adopting the processes and approach while increasing velocity.

5 – Security Should Take a Proactive Holistic Approach

Digital transformation delivers significant and unquestionable business benefits. But the development environments used to execute digital transformation strategies can also expand the attack surface and allow unprotected privileged accounts, credentials, and secrets to become damaging security vulnerabilities.

To protect development environments, organizations must understand the unique security challenges that development environments present – and take steps to address them head on.

For example, CyberArk offers the CyberArk Blueprint, which is a prescriptive framework to help guide organizations with a risk-based approach to building their privileged access management programs. The Blueprint is based on three guiding principles: preventing credential theft, stopping lateral and vertical movement, and limiting privilege escalation and abuse across technologies – including hybrid and multi-cloud environments. Infrastructure-as-a-service admins, CI/CD consoles, and dynamic applications are covered across the initial three stages of the Blueprint, ensuring privilege is secure wherever it exists across the application development and deployment process.

Developers Need to Be On Board

While the need to secure developers is clear, development environments present unique challenges to security teams. Developers typically won't fully address security processes on their own. Yet IT and security team members are less familiar with development tools and processes and may not understand how far development teams are embedded into the organization's operations. At the same time, developers can perceive security as slowing them down. Developers are creative and will find workarounds to do their jobs and deliver solutions to the business, so it's important to offer solutions to secure credentials and secrets that developers will use.

There’s the problem of overconfidence as well. Too often developers may think they understand security, can select security solutions, and have the skills to secure applications themselves. But security is complex and requires a specific, focused perspective. Developers may not fully recognize the complexity involved or the critical need to maintain secure processes. An attacker only needs to find one vulnerability in the development processes to exploit and cause damage. Clearly, the security team must ensure development environments are secure.

ENGAGING DEVELOPERS

Security team members know that they need to work more proactively with their Dev and DevOps counterparts but often don’t know where to start. Research on approaches for securing DevOps environments from CISOs and security leaders at the Global 2000 identified five key approaches to successfully engage with and help developers adopt secure practices:

1. Transform the security team into a DevOps partner.
2. Prioritize securing DevOps tools and infrastructure.
3. Establish enterprise requirements for securing credentials and secrets in DevOps and cloud environments.
4. Adapt secure processes for application testing.
5. Evaluate the results of the DevOps security program.

Refer to the full report, [The CISO View: Protecting Privileged Access In DevOps And Cloud Environments]() for additional insights.

THE CISO VIEW

Protecting Privileged Access in DevOps and Cloud Environments

RECOMMENDATIONS

Security Needs to “Shift Left” And Engage With Developers Early In The Development Process

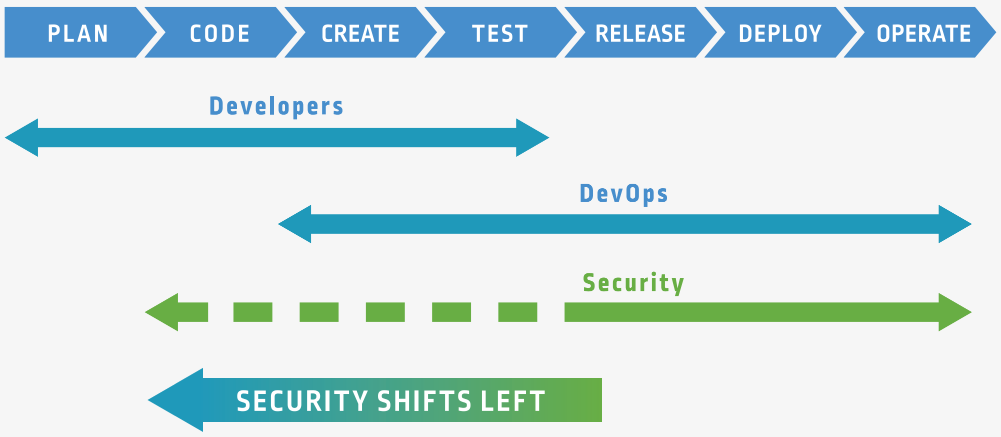
One example, as noted earlier in use case 2, secrets management solutions enable critical vulnerabilities to be eliminated by removing hard-coded secrets (needed by applications to securely access databases and resources) from code. But when should hard-coded credentials be removed from applications and where, and in what system should they be secured? Is the system securing these credentials intrinsically secure?

Different teams share responsibility across the development lifecycle (from planning, coding, creating a build, testing, releasing, and deploying into production). Developers primarily write and test code, while DevOps and Ops typically build the systems that flow the code into production. Too often security only gets involved just before the code goes live. This is a problem since developers need to code their applications to fetch these secrets early in the process.

Security teams can more effectively work with developers to secure applications by getting involved earlier in the development process, commonly referred to as “shifting left.” This avoids problems of waiting until code gets close to deployment when critical security decisions will, deliberately or inadvertently, already have been made. These decisions are not only tough to re-do, but when security jumps into the process late it has the potential to further antagonize developers who may already view security as a roadblock.

While there are, almost certainly, separate development and production environments because the build process is so automated, it is critical that security is involved from planning and coding all the way through to production.

Security Shifts Left



Development Environments Must Be Secured… Consistently

Development environments have privileged credentials used by both the human users (developers, tool admins, site reliability engineers, cloud admins, etc.), as well as the applications, automation process, and other machine and non-human identities. To secure development environments both human and non-human identities must be secured.

Security and access policies should be consistently enforced across the entire organization, regardless of the cloud and compute environments, development philosophy, or organizational complexity, to minimize the attack surface. The approach should address the organization's needs as its development environments and compute infrastructure evolve. This should be achieved with a single point of control that enables consistent management of privileged accounts, credentials, and secrets across each of the development and compute environments.

CyberArk Can Help Organizations Secure Their Development Environments

No other vendor has the broad experience or offers a comparable breadth of privileged access management solutions for securing both human users and non-human identities across the entire organization – including DevOps, hybrid, and native cloud environments. The CyberArk Privileged Access Security Solution enables organizations to secure the privileged credentials and secrets used across their entire enterprise, including the development environments that are driving the organizations digital transformation.

About CyberArk

CyberArk is the global leader in privileged access management, a critical layer of IT security to protect data, infrastructure and assets across cloud and throughout the DevOps pipeline. CyberArk delivers the industry's most complete solution to reduce risk created by privileged credentials and secrets. The company is trusted by the world's leading organizations, including more than 50 percent of the Fortune 500. To learn more about how CyberArk can help your organization secure its development environments, visit www.cyberark.com.

CYBERARK PRODUCTS AND SOLUTIONS WHICH ADDRESS THE USE CASES OUTLINED IN THIS WHITEPAPER INCLUDE:

- Endpoint Privilege Manager enforces least privilege on the endpoint (Windows and Mac) to contain attacks early in their lifecycle. It enables revocation of local administrator rights, while minimizing impact on user productivity, by seamlessly elevating privileges for authorized applications or tasks. Application control, with automatic policy creation, allows organizations to prevent malicious applications from executing, and runs unknown applications in a restricted mode. This, combined with credential theft protection, helps to prevent malware from gaining a foothold and contains attacks on the endpoint.
- Core Privileged Access Security (Core PAS) provides a comprehensive set of capabilities for securing the credentials used by human users, including credential protection and management, session isolation and monitoring, privileged analytics and threat detection, and least privilege management. It is the core of CyberArk Privileged Access Security Solution that enables every product to work independently while taking advantage of shared resources and data. This flexible approach allows an organization to start small and scale to a complex, distributed, enterprise solution over time.
  - Privileged Session Manager for Web is part of Core PAS and allows authorized users to connect directly to cloud consoles and applications without any additional security steps and without ever being exposed to credentials. This approach balances both security and adoption requirements for securely accessing those highly privileged accounts.
- CyberArk Application Access Manager's secrets management solution, Dynamic Access Provider, is designed to address the unique requirements of native-cloud and DevOps environments while simplifying how developers secure and use secrets. The solution integrates with a wide range of DevOps tools such as Ansible, Jenkins, and Puppet; PaaS/Container orchestration platforms such as Red Hat OpenShift, Tanzu, Cloud Foundry, and Kubernetes, whether running on hybrid or multi-cloud platforms. The solution integrates with CyberArk Core PAS to provide a single enterprise-wide platform for securing privileged access. The open source secrets management solution that Dynamic Access Provider is based on, Conjur Open Source, is available at www.conjur.org.