

ESG SHOWCASE

Modern Development Practices Open the Door for Code Leakage

Exposing Sensitive Information Endangers Software Supply Chains

Date: December 2021 **Author:** Dave Gruber, Principal Analyst

ABSTRACT: As application deployment strategies transform to support modern cloud architectures, continuous integration/continuous delivery (CI/CD) pipelines are leveraging services and tools that insert a growing amount of sensitive information into code repositories. As development teams leverage cloud-based infrastructure in support of collaboration and speed, code leakage has become a significant concern for organizations. New security strategies are required to mitigate this growing risk.

Overview

In the spirit of accelerating the delivery of applications, modern application development practices continue to be aggressive about leveraging advances in both development and deployment tooling and services. As AppDev strategies transform in support of modern cloud architectures, continuous integration/continuous delivery (CI/CD) pipelines are leveraging services and tools that insert a growing amount of sensitive information into code repositories, especially in the context of infrastructure-as-code (IaC) and policies-as-code.

In parallel, the widespread use of cloud-based development infrastructure, including source code management (SCM) services from GitHub, GitLab, Bitbucket, and others, has helped development teams collaborate and accelerate code development. These cloud-delivered SCM services are progressively becoming collaboration and sharing platforms used to facilitate all facets of application development, including code, configuration, documentation, plans, notes, activation codes, license keys, and more. New risks are born from these widely adopted development practices, including code leakage risk, offering adversaries a new means to gain access to an array of sensitive information that can help facilitate sophisticated attacks.

Beyond the potential leakage of intellectual property and sensitive data, code leaks can lead to software supply chain breaches (like SolarWinds, Kaseya, etc.) because they can expose secrets, IaC configurations, and other information that may be used by attackers to breach the DevOps tools and infrastructure that comprise software delivery pipelines. Furthermore, source code is the blueprint of the application. When source code falls into the wrong hands, it can be reverse-engineered to create zero-day exploits.

All too often IaC templates get carried into a code base, labeled “a publicly accessible DB is unencrypted,” providing an adversary with easy access. Even exposing style and naming conventions can be problematic. For example, a non-conforming naming convention is a powerful potential indicator of tampering.

New, more holistic security strategies are required to mitigate this growing risk. These strategies must help prevent leaks, reduce their impact, and minimize organizations’ exposure when leaks do occur.

Understanding the Risks

Unintentional Leakage through SaaS-based Source Control

Modern development practices include the use of SaaS-based SCM services, leveraging providers such as GitHub, GitLab, and Bitbucket. These SCM service providers house and manage private, public, and personal repositories.

Unlike corporate email accounts or other SaaS-based enterprise applications where users are essentially provided provisional, corporate-governed access during their employment (which is revoked when they leave), developer SaaS-based SCM access is provided through personal accounts. Once logged in, developers can work on different types of code repositories, including private/corporate repositories, public/open-source repositories, or personal repositories. These personal accounts are typically open by default, supporting most developers' desire to share their personal code. This access model is radically different from other enterprise-use applications where access control is governed by corporate policies.

Offering measurable advantages that result in faster delivery times, these service-based operating environments create opportunities for developers to concurrently pull from, and contribute to, a wide variety of code repositories, spanning private, public, and personal domains. Developers, therefore, need to be extremely careful about posting code to the intended repository, as it's all too easy for a developer to unintentionally push sensitive code into a public repository, inadvertently making that code publicly available.

This fundamentally different access model adds significant risk, requiring security teams to implement additional precautionary steps to mitigate these risks.

Over-provisioned Developer Access

Least-privileged access is a central theme for end-user access to applications, leading 82% of security teams to increase spending on zero-trust initiatives over the next 12-18 months.¹ Restricting access for IT admins further helps protect organizations from costly mistakes and exposure of critical assets. Yet the application development world is often grossly overlooked, with individual developers commonly provided broad access to code that is used to power many of the most secure applications and data. This over-provisioned access opens the door for both unintentional and intentional access to infrastructure assets that should otherwise be locked down. Additionally, restricting key privileges like cloning, forking, or downloading go a long way toward reducing inadvertent code leaks.

Pushing Sensitive Code

It's all too easy for a developer to unintentionally push sensitive code into a public repository, inadvertently making that code publicly available.

Infrastructure-as-code (IaC)

Unlike development patterns of the past where infrastructure was defined and configured by the SysOps team completely outside of an application, modern development has developers provisioning their own infrastructure, using infrastructure-as-code (IaC). IaC

configuration therefore ends up in the source code repository, exposing it to potential leakage.

Configuration errors in IAC can lead to breaches, just like vulnerabilities written into source code. And just as source code exposure can lead to zero-day exploits through reverse engineering, IAC exposure is also a blueprint that can show where resources may inadvertently be exposed. However, the dangers in IAC exposure are potentially higher in that IAC code is simpler and easier to quickly read and understand, whereas reverse engineering source code for zero-day exploits requires significant time and expertise.

¹ Source: ESG Research Report, [The State of Zero-trust Security Strategies](#), April 2021.

Embedded Secrets

While hardcoding secrets is clearly not a best practice, developers sometimes lack an alternative. Hardcoding secrets is a temporary fix, a shortcut, or simply employed due to ignorance. When secrets end up in the codebase this way, leakage becomes a greater risk. Secrets can leak in many ways, including code snippets from private repositories contributed to public repositories, unsecured log files from the build stream, Kubernetes configuration files, and more. API keys, encryption keys, tokens, and passwords can often find their way into the code as well. Attackers believing that an organization's code contains secrets can increase the odds of exposure (particularly from insider threats) because exfiltrating the code may become a key goal in mining for secrets. The net result is that when code is exposed, the presence of embedded secrets dramatically escalates the severity of the breach.

Access and Credential Hygiene Issues

Further exposure exists when developers depart to other companies while retaining privileged access to sensitive systems and data. Many organizations lack exit processes and controls to ensure developer access to systems and infrastructure is shut down upon exit. Because developers often are trusted with access to code repositories outside their specific application areas, loose ends often exist, providing past employees with ongoing access.

Malicious Insider Risk

These same mechanics could lead to intentional code leakage by a compromised or malicious insider. While most organizations want to trust their developers, many depend on a software supply chain and/or other third-party resources to contribute to their code base. This leaves plenty of opportunity for malicious insiders to expose secrets into the wild. The Kaseya attack demonstrates this threat.

When certain types of intellectual property (IP), personally identifiable information (PII), or other sensitive information is unintentionally and openly exposed to developers, insider risk can occur. The more people that have access to this information, the more people can make mistakes or be tempted to exploit their access to this information. Organizations should never assume that they are not harboring malicious insiders. Enforcing least privilege reduces this risk.

A Foothold to Attack the Software Supply Chain

As adversaries see an opportunity to gain access to embedded secrets or IaC configurations, new attack strategies have emerged that target the software supply chain. When secrets surface in code, or when IaC configurations are exposed, attackers now have a new path in, using secrets or reverse engineering an IaC template to find a vulnerability.

Malicious, broad-scale scanning of public repositories is helping adversaries to accelerate their discovery of secrets that can provide an access path into applications and infrastructure, including access to those who contribute to the software supply chain. When this happens, adversaries can gain access to source code, and embed and disguise malicious code into production systems later utilized by multiple organizations, thus providing a foothold for a plethora of malicious activities.

Developers Need Help

According to the famous quote from Spiderman, “With great power comes great responsibility.” As developers are entrusted with deeper levels of sensitive information, they need automated, assistive controls that can help protect them and their organizations from both unintentional and intentional code leakage.

Developers can certainly reduce risk by improving hygiene, avoiding hardcoding secrets, writing IaC code that is free of misconfigurations, and practicing careful discipline in the use of public, private, and personal code repositories. But leaving all this to developers is uncertain and risky.

What's Needed

Similar to the concepts of data leakage/loss prevention (DLP), development-centric DLP strategies are now needed to prevent both the intentional and unintentional exposure of sensitive information stored within modern codebases. These strategies should include:

1. Continuous monitoring for sensitive data that finds its way into the public domain, often through public repositories.
2. Continuous monitoring and auditing of developer credentials. Regular auditing for excess privileges is key to operationalizing least privilege policies and will ensure that developers only have access to the specific code bases on which they are working. Moreover, a least-privilege developer access policy that includes access reviews and deprovisioning policies will mitigate overprovisioning risks that often result in unintentional leaks and malicious insider activities.
3. Monitoring for potentially suspicious behaviors, such as downloading, forking, or cloning repositories. It is essential to pay particular attention when sensitive resources or risky time periods are involved, such as the last 30 days of a developers' employment.
4. The use of multi-factor authentication (MFA) helps ensure that developers are who they say they are, limiting risk through compromised insider accounts.
5. Deprovisioning processes must be in place to enable administrators to confidently off-ramp and restrict access to departing developers.
6. Eliminating hardcoded secrets reduces the severity of code exposure. Scanning commits and merge requests for hardcoded secrets and IaC misconfigurations therefore helps developers catch potential issues further left in the SDLC.
7. Monitoring across the entire DevOps pipeline for hardcoded secrets, exposed configurations, and other embedded sensitive information ensures a comprehensive approach, as secrets can live in many places across the SDLC.
8. Whenever possible, IaC and source code should not be stored in the same repositories. Following this best practice enables finer-grained repository-level access controls because developers typically don't need access to IaC and DevOps teams don't need access to source code.

New Security Strategies Needed

A complete code leakage solution needs to:

- Help prevent code leaks from occurring.
- Reduce damage caused by a code leak, including the likelihood it will develop into a breach.
- Minimize exposure of leaked code when a leak does happen.

Introducing Cyscale

A Software Supply Chain Security Platform for SDLC Security, Governance, and Integrity

- **Prevent theft and leakage of your most valuable asset: code.** Stop developers from inadvertently exposing proprietary code in public repositories, fingerprint code assets, and proactively identify exposure on public sites.
- **A single source of truth for DevOps pipeline governance.** Apply and enforce consistent governance and security policies across all DevOps tools and infrastructure, both in the cloud and on-premises.
- **Prevent developers from writing secrets into code.** Scan every commit or pull/merge request for hardcoded secrets and prevent them from reaching the master branch across all SCMs and programming languages.
- **Secure infrastructure-as-code (IaC) configurations and production systems.** Scan IaC for security misconfigurations and ensure compliance between defined IaC configurations and production infrastructure.
- **Maintain code and release integrity throughout the SDLC.** Validate the integrity of each handoff in the SDLC from code development and signing through deployment into production to prevent code tampering.

The Bigger Truth

Agile and cloud development models have advanced significantly, enabling faster development and delivery using SaaS-based development tools together with infrastructure-as-code strategies. Meanwhile, the software supply chain continues to expand and grow with development teams depending on the rapid assembly of trusted components created by others.

These strategies have introduced new security risks, with code leakage growing at unprecedented rates, exposing new entry points for adversaries to compromise infrastructure and steal credentials. Overlooking this important control point is simply a flawed security strategy, one that is becoming costly for an increasing number of organizations.

ESG recommends security teams embrace a risk reduction approach to code leakage that seeks to prevent breaches with proper security and governance, reduce their impact by removing secrets and misconfigurations from code, and proactively search external code sharing sites for proprietary code leaks, reducing the length of any public exposure. ESG also recommends that development and security teams explore solutions that approach software supply chain security using layered security controls from vendors like Cyscale that are focused on protecting organizations from this growing threat vector.



All trademark names are property of their respective companies. Information contained in this publication has been obtained by sources The Enterprise Strategy Group (ESG) considers to be reliable but is not warranted by ESG. This publication may contain opinions of ESG, which are subject to change. This publication is copyrighted by The Enterprise Strategy Group, Inc. Any reproduction or redistribution of this publication, in whole or in part, whether in hard-copy format, electronically, or otherwise to persons not authorized to receive it, without the express consent of The Enterprise Strategy Group, Inc., is in violation of U.S. copyright law and will be subject to an action for civil damages and, if applicable, criminal prosecution. Should you have any questions, please contact ESG Client Relations at 508.482.0188.



Enterprise Strategy Group is an IT analyst, research, validation, and strategy firm that provides market intelligence and actionable insight to the global IT community.



www.esg-global.com



contact@esg-global.com



508.482.0188