BLUBRACKET

Know your Code Security Risk Profile.

A five point checklist to determine risks and protect yourself from hacks directed at code.

Code is arguably the largest attack surface today.

Git-based systems like GitHub, Gitlab and Bitbucket are fantastic for developer collaboration, speed and productivity, they just aren't instrumented for security.

Code is now a top vector into the enterprise exploited by bad actors. SolarWinds is the largest, but there are many, many hacks that originated from credentials found in code.

Companies need to understand their code risk profile. **And quickly**.

Here's a **five point checklist** to determine your code risk profile and what steps to take to secure your code.

The result will be increased peace of mind that comes from proactively addressing the largest attack surface for the enterprise.

Let's get started.

1 Map (understand) who has access to your code on what devices.

You can't secure what you can't see, but all too often a CISO or application security engineer is in the dark when it comes to detailed information on code. This is largely because Git by nature is open and permissive, given its open source roots. Companies also want developers to be collaborative and are loath to introduce friction to an activity that is so valuable.

But it's foolish to NOT understand who has access to your code for one of the enterprise's most valuable assets, especially because Git makes it so easy to clone and redistribute code.

At a minimum, companies need to map and label critical code. Your company's trading model, of course, has far greater importance than basic web code. Start by understanding where your most critical code is and then audit who has access to it. Restrict full access to only those who really need it, and make it less permissive by default.

Without visibility, companies can't track malicious insider activity or know if contractors/employees have uploaded important IP to personal servers or open source. Security begins with knowledge.

- Run an audit of your most critical code and be sure only those who absolutely need full permissions have them.
- Purge non-current users.
- Follow the principle of least privilege for your most critical code and set up access controls per repository.
- Make sure all commits are signed, so you can verify who the original developer of the code is and that the code was not tampered with.
- Perform user access audits quarterly.

WIRED

A Boeing code leak exposes security flaws deep in a 787's guts

A security researcher found an open server on Boeing's network containing code for it's 787 aircraft

With the information, the ethical hacker disclosed how to target the networking system.

The code gave him a road map to exploit and target vulnerablilities in the airplane's critical systems.

ALL from code.

2 Detect if secrets, PII or other credentials are present in code.

"A security researcher recently found 200,000 health records that were leaked on GitHub just by doing simple searches for secrets and PII. Source: Databreaches.net "No need to hack when it's leaking."

Secrets are passwords, encryption keys, API keys, and OAuth tokens that are required for one application service to authenticate against another service. With modern software, secrets are a necessity.

The problem with secrets is they are often hardcoded into the application code, and thus can be discovered by nefarious users. In 2019, nearly half of all breaches came from the misuse of credentials that were frequently left inside code. And when these secrets are in repositories housed on GitHub or other cloud-based code repositories, they can easily fall into the wrong hands. In fact, it's one of the most common tactics used by hackers today.

Do a google search and you'll find a never-ending list of companies that have been hacked, simply because credentials and access keys were left in code on GitHub. Solar Winds for instance was started by a password left in code by an intern.

- Form a best practice to discourage the hard-coding of secrets in code. Train your developers on why this is important, even if your repositories are all presumably private.
- Add a security.md file to your repositories outlining this and other security policies.
- Mandate developers use security plugins that work with their coding tools of choice to catch secrets or PII before they are committed to a repository.
- Task AppSec/DevSecOps with continuous, real-time monitoring of Git accounts to ensure secrets are not leaked. This should include both private and public repositories.
- Use a third-party service, such as Vault, to protect secrets and have your application dynamically download the secrets from Vault during runtime only as needed.

3 Determine if your code is availiable in public repositories.

Most companies find that their protected and valuable IP is indeed available on public repositories on GitHub, in Amazon S3 servers, on dark web info-sharing sites or other spots.

This is not always a case of a bad actor stealing the IP and sharing it for profit. More often it's an unwitting mistake. A contractor may fork and then upload code he worked on to showcase his skills, not realizing that critical information is found in that code.

Hackers have published detailed guides of how to target individual developer's and contractor's personal accounts to find company code. It's imperative that companies train developers on why it's important not to share company code (unless it's been approved for open source projects.) To them, it's just a work product they want to show. But to a hacker, it's a goldmine of information, potential trade secrets and IP.

It's also best practice to run systems that can alert you on developer actions that signal code leakage. For instance set an alert when a critical code repository is forked by a new user.

The reality is your code has probably already been leaked. Invest in tools that allow you to do smart searches to find your protected code, so it can be reclaimed and checked for security credentials.

- Fingerprint your most important code so you can run queries and monitor for leakage.
- Run a check of all public repositories with your unique code fingerprint.
- Task AppSec/DevSecOps with continuous, real-time monitoring to ensure critical code is not leaked.
- Vet your developers' personal GitHub accounts. A lot of company code ends up in employees' or contractors' public repositories.

4 Fix Git misconfiguration that leave you exposed.

Last year, Mercedes Benz left 580 source code repositories open and available for anyone to access on the Web. These repositories not only contained valuable source code for vehicle components which could be used for attack, they also contained passwords and tokens that unlocked access to other Mercedes private servers.

The source of the problem was a misconfiguration of GitLab that allowed anyone to create a user account and access the code—all the code in every repository.

Misconfigurations like these are common. Companies need to set and enforce best practices for configuring Git-based systems.

For instance, make sure all repositories are marked private for company code that isn't open source. It's also crucial to make sure that .Git folders, which contain backups of Git activity, aren't inadvertently left exposed on public web servers. These files are frequently searched for and found by hackers looking for sensitive information or IP found in source code. In 2021, Nissan Motors misconfigured a Bitbucket server that left it accessible on the public web with default credentials of admin/admin.

- Check for common misconfigurations, such as leaving a .Git file on your public website.
- Audit all company repositories and ensure they are marked private. Git is public by default so it may surprise you what you find.
- Require (and audit) that all repositories are private and that all email and repo accounts have 2FA.
- Require (and audit) that all developer email accounts have 2FA.
- Audit and understand all webhooks and applications that have access to your code. Granting access to these applications can sometimes introduce security risks.

"Open .git directories are a common problem – a scan of more than 230 million web domains worldwide, in fact, uncovered 390,000 web pages that were vulnerable due to the issue."

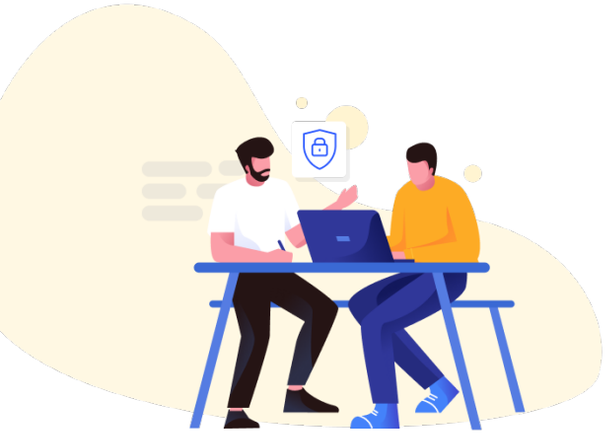
5 Monitor your Infrastructure code for misconfigurations

Infrastructure as Code (IaC) is a vital piece of DevOps. Kubernetes, AWS CloudFormation, Ansible and Terraform all make use of code to automatically provision or modify infrastructure.

Like Git, IaC has been a boon for innovation. Unfortunately most companies today are leaving their IaC files wide open to attack. A misconfigured IaC template can give hackers the ability to spin up their own infrastructure and gain access to your network or sensitive data.

Instead of focusing only on application code in repositories, also pay attention to IaC templates, files, modules and their variables and ensure they are configured for security.

- Check all IaC instances for misconfigurations and vulnerabilities that expose the files to the public Internet.
- Make IaC security checks standard in your CI/CD pipeline. Many companies do it during pull request checks.
- Check for secrets and tokens within IaC configuration files.
- Use a Vault for all secrets management within and outside IaC.
- Scan public repositories for your IaC files and check for credentials.



Blubracket can help

If you're looking for a partner on all things code security, we can help. By adding a security tool like BluBracket, companies gain the advantages of Git-based development, while mitigating the security risks and helping developers shift security left. From secrets and PII to misconfigurations and code proliferation, BluBracket reduces risk at every stage of your software development process.

You can start a free trial, use our completely free secrets scanning tool and get your real-time code risk score at blubracket.com.

BLUBRACKET