klocwork

A GUIDE TO SECURITY IN SOFTWARE DEVELOPMENT:

Secure Coding Standards Best Practices

Introduction

Up to 90% of software security problems are caused by coding errors, which is why secure coding standards are essential. There are several secure coding practices you can adopt. But, to write truly secure code, you need a secure coding standard.

Here, we explain what are secure coding standards and how to enforce these security standards.

Contents

What Are Secure Coding Standards?

Secure coding standards are rules and guidelines used to prevent security vulnerabilities. Used effectively, these security standards prevent, detect, and eliminate errors that could compromise software security.

Why Are Secure Coding Standards Important?

Secure coding standards are important because they help to ensure that software is safeguarded against software security vulnerabilities. What's more, secure coding is important for every development team — regardless of whether it's code for mobile devices, personal computers, servers, or embedded devices.

What Is CWE?

Common Weakness Enumeration (CWE) list identifies software security weaknesses in software and hardware. This includes C, C++, and Java. The list is compiled by feedback from the CWE Community.

Sponsored by the MITRE Corporation, the community is made up of representatives from major operating systems vendors, commercial information security tool vendors, academia, government agencies, and research institutions.

The full list is regularly updated every few months with the latest version released in August 2020. The security weakness list includes over 600 categories, which include:

- Buffer overflow
- Cross-site scripting
- Insecure random numbers

WHAT IS CWE TOP 25?

Published by MITRE, the CWE Top 25 is a compilation of the most widespread and critical weaknesses that could lead to severe software vulnerabilities. The most recent list was published in 2020 and listed vulnerabilities that allowed hackers to gain control over an affected system, steal sensitive data, and cause a denial-of-service condition.

CWE TOP 25

Here is the list of the 2020 CWE Top 25 software weaknesses:

- Improper Neutralization of Input During Web Page Generation ("Cross-site Scripting")
- Out-of-bounds Write
- Improper Input Validation
- Out-of-bounds Read
- Improper Restriction of Operations within the Bounds of a Memory Buffer
- Improper Neutralization of Special Elements used in an SQL Command ("SQL Injection")
- Exposure of Sensitive Information to an Unauthorized Actor
- Use After Free
- Cross-Site Request Forgery (CSRF)
- Improper Neutralization of Special Elements used in an OS Command ("OS Command Injection)
- Integer Overflow or Wraparound
- Improper Limitation of a Pathname to a Restricted Directory ("Path Traversal")
- NULL Pointer Dereference
- Improper Authentication
- Unrestricted Upload of File with Dangerous Type

- Incorrect Permission Assignment for Critical Resource
- Improper Control of Generation of Code ("Code Injection")
- Insufficiently Protected Credentials
- Improper Restriction of XML External Entity Reference
- Use of Hard-coded Credentials
- Deserialization of Untrusted Data
- Improper Privilege Management
- Uncontrolled Resource Consumption
- Missing Authentication for Critical Function
- Missing Authorization

What Is CERT?

CERT is a secure coding standard maintained by the Software Engineering Institute at Carnegie Mellon University. It supports commonly used programming languages such as C, C++, and Java.

The standards are developed through a broad-based community effort by members of the software development and software security communities.

The rules and recommendations target insecure coding practices and undefined behaviors that lead to security risks.

The latest rules and recommendations are available on the secure coding standard's website and are also periodically published: C and C++ in 2016 and Java in 2011.

Why Is CERT Secure Coding Important?

CERT secure coding is important because although C, C++, and Java are flexible, high-performance languages, they are still vulnerable to security risks. As these languages are extensively used in many applications, it is vital that code is reviewed at all stages of development.

The aim of the secure coding standard is to not only detect security risks with rules but also provide suggestions that can improve code quality with recommendations.

The scope of the secure coding standard is a whole program coding standard and the goal is to produce safe, reliable, and secure systems.

By using the secure coding standard, you are able to ensure that your software is secure and safeguarded from potential vulnerabilities.

CERT Risk Assessment

For each guideline included in the secure coding standard, there is a risk assessment to help determine the possible consequences of violating that specific rule or recommendation. There are three sections to the risk assessment: Severity, Likelihood, and Remediation Cost. Each section is assigned a value between 1 and 3, and based upon the results of the assessment, you are able to determine the priority of the violation.

SEVERITY —

How serious are the consequences of the rule being ignored.

Value	Meaning	Examples of Vulnerability
1	Low	DoS Attack
2	Medium	Data Integrity Violation
3	High	Run Arbitrary Code

LIKELIHOOD —

How likely is the coding flaw introduced by violating the rule can lead to an exploitable vulnerability?

Value	Meaning
1	Unlikely
2	Probable
3	Likely

REMEDIATION COST —

How expensive will it be to comply with the rule.

Value	Meaning	Detection	Correction
1	High	Manual	Manual
2	Medium	Automatic	Automatic
3	Low	Automatic	Automatic

Each of these three values— Severity, Likelihood, and Remediation Cost — are then multiplied together to determine priority, which is a measure of the risk, and therefore the level of the vulnerability. This can be used to prioritize the repair of violations.

PRIORITIES AND LEVELS

Level	Priorities	Possible Interpretation
L1	12, 18, 27	**Severity:** High Severity **Likelihood:** Likely **Remediation Cost:** Inexpensive to Repair
L2	6, 8, 9	**Severity:** Medium Severity **Likelihood:** Probable **Remediation Cost:** Medium Cost to Repair
L3	1, 2, 3, 4	**Severity:** Low Severity **Likelihood:** Unlikely **Remediation Cost:** Expensive to Repair

What Is CVE?

Common Vulnerabilities and Exposures (CVE) is a list of publicly known cybersecurity vulnerabilities and exposures. Each item on the list is based upon a finding of a specific vulnerability or exposure found in a specific software product, rather than a general class or kind of vulnerability or exposure.

CVE has been designed to make it easier to link information from vulnerability databases, and allow comparison of security tools and services. The list is made up of a collection of CVE Identifiers assigned to each vulnerability and exposure.

What Is a CVE Identifier?

CVE Identifiers are unique identifiers for assigned to publicly known cybersecurity vulnerabilities. The Identifiers are used as a standard method for identifying vulnerabilities and for cross-linking with other repositories.

Each Identifier includes the following:

- An Identifier number
- Indication of "entry" or "candidate" status
- Brief description of the security vulnerability or exposure
- Any pertinent references

What's Included on the CVE List?

The CVE list catalogs several types of software vulnerabilities, including:

- Denial of Service (DoS)
- Code Execution
- Buffer Overflow
- Memory Corruption
- SQL Injection
- Cross-Site Scripting (XSS)
- Directory Traversal
- HTTP Response Splitting

It's important to be able to identify each and every vulnerability that may be present in your code, and static analyzers — like Klocwork — are the most effective tools to identify and fix software security vulnerabilities.

How to Fix CVEs

To fix common vulnerabilities and exposures, follow these steps:

- Establish software design requirements, which includes defining and enforcing secure coding principles. This helps to inform how to effectively write, test, inspect, analyze, and demonstrate your code.
- Use a coding standard — such as OWASP, CWE, and CERT — to help prevent, detect, and eliminate vulnerabilities.
- Implement security checks into your CI/CD pipeline to identify software security vulnerabilities early. In addition, this helps to enforce good coding practices.
- Test your code as early and often as possible to ensure that vulnerabilities are found and eliminated.

What Is NVD and CVSS?

The National Vulnerability Database (NVD) is the U.S. government repository of standards-based vulnerability management data.

This database is synchronized with the CVE list and provides additional content, including how to fix vulnerabilities, severity scores, and impact ratings. In order to calculate severity scores, the Common Vulnerability Scoring System (CVSS) must be used.

What Is CVSS?

The Common Vulnerability Scoring System (CVSS) is an open industry standard for assessing the severity of software vulnerabilities.

For each vulnerability, CVSS assigns a severity score from 0.0 (the lowest amount of risk) to 10.0 (the highest amount of risk), which enables you to more effectively prioritize remediation of vulnerabilities.

CVSS V3.0 RATINGS

Severity	Base Score Range
None	0.0
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

To calculate the CVSS base score, you need to input CVSS metrics into the NVD CVSS Calculator. For detailed guidance on how to use the calculator, be sure to refer to the CVSS standards guide.

What Are CVSS Metrics?

The common vulnerability scoring system is made up of three groups of metrics: base, temporal, and environmental.

BASE METRICS

Base metrics are divided into two groups: exploitability and impact.

Exploitability Metrics

Exploitability metrics refer to the characteristics of the piece of software or product that make it vulnerable.

Exploitability metrics refer to the characteristics of the piece of software or product that make it vulnerable.

- Attack Vector — Shows how a vulnerability may be exploited.
- Attack Complexity — Refers to how easy or difficult it is to exploit the discovered vulnerability.
- Authentication — Refers to the number of times that an attacker must authenticate to a target to exploit it.
- User Interaction (UI) — Refers to the requirement for a human user — other than the attacker — to participate in the successful compromise of the vulnerable component.
- Privileges Required (PR) — Refers to the level of privileges an attacker must possess before successfully exploiting the vulnerability.

Impact Metrics

Impact metrics deal with the worst case scenario, if the piece of software or product were to be attacked and the effects of a successfully exploited vulnerability.

- Confidentiality — Refers to the impact on the confidentiality of data processed by the system.
- Integrity — Refers to the impact on the integrity of the exploited system.
- Availability — Refers to the impact on the availability of the target system.

TEMPORAL METRICS

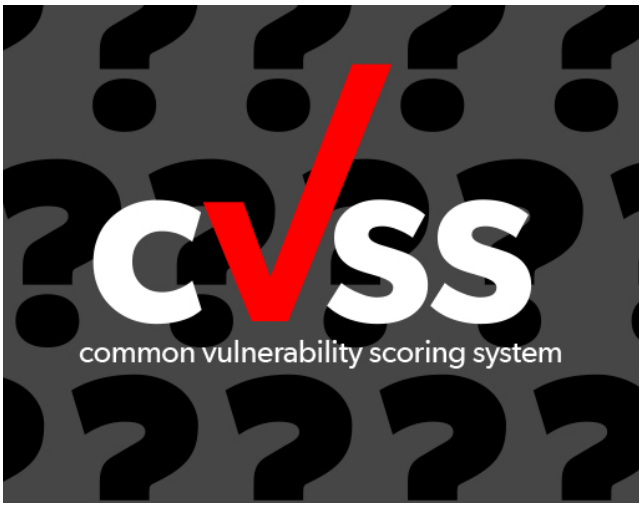
Unlike the other CVSS metrics, the value of temporal metrics change over the lifetime of the vulnerability. This is due to exploits being developed, disclosed, and automated along with mitigations and fixes being made available.

- Exploitability — Refers to the current state of exploitation techniques or automated exploitation code.
- Remediation Level — Refers to the amount of mitigations and official fixes that are available to decrease the amount of vulnerabilities.
- Report Confidence — Refers to the level of confidence in the existence of the vulnerability and the credibility of the technical details for the vulnerability.

ENVIRONMENTAL METRICS

The environmental metrics use the base metrics score and the temporal metrics score to assess the severity of a vulnerability to the piece of software or product that is currently in development.

- Collateral Damage Potential — Measures the potential loss or impact on either physical assets — such as equipment, hardware, and users — or the financial impact, if the vulnerability is exploited.
- Target Distribution — Measures the proportion of vulnerable systems.
- Impact Subscore Modifier — Measures the specific security requirements for confidentiality, integrity, and availability. This metric enables you to customize the environmental score based upon your environment.



What Are the Top 10 Most Exploited Vulnerabilities?

There are hundreds of vulnerabilities on the CVE list, but these are the top 10 most exploited.

Vulnerability ID	Vulnerability Summary	CVSS Severity
CVE-2019-19781	Citrix Application Delivery Controller Vulnerability	9.8 — Critical
CVE-2018-7600	Drupal Remote Code Execution Vulnerability	9.8 — Critical
CVE-2015-1641	Microsoft Office Memory Corruption Vulnerability	9.3 — Critical
CVE-2017-8759	Microsoft .NET Framework Remote Code Execution Vulnerability	7.8 — High
CVE-2018-4878	Adobe Flash Player Vulnerability	9.8 — Critical
CVE-2017-0143	SMB Server Vulnerability in Older Versions of Windows and Windows Server	8.1 — High
CVE-2019-0604	Remote Code Execution Vulnerability in all Modern Versions of Sharepoint	9.8 — Critical
CVE-2012-0158	Microsoft Office Vulnerability	9.3 — Critical
CVE-2017-5638	Apache Struts Vulnerability	10.0 — Critical
CVE-2017-0199	Microsoft Office Remote Code Execution	7.8 — High

What Is DISA STIG?

DISA STIG refers to an organization (DISA — Defense Information Systems Agency) that provides technical guides (STIG — Security Technical Implementation Guide).

DISA is part of the Department of Defense (DoD). It's a combat support agency that provides IT and communication support to all institutes and individuals working for the DoD. DISA oversees the IT and technological aspects of organizing, delivering, and managing defense-related information.

This includes STIG guidelines. These guides outline how an organization should handle and manage security software and systems.

What Is STIG Security?

STIGs (Security Technical Information Guides) are security guidelines from DISA. There are 100s of STIGs maintained and updated by DoD.

Complete STIG Security List

There's a complete STIG list that provides critical updates on the standards for DoD IA and IA-enabled devices/systems. Each STIG provides technical guidance to secure information systems/software that might otherwise be vulnerable.

The DoD regularly updates STIGs to ensure that developers are able to:

- Configure hardware and software properly.
- Implement security protocols.
- Organize training processes.

You can use the STIG list to identify potential weaknesses in your code.

But the best way to use the STIG list is by pairing it with a SAST tool. SAST tools like Klocwork help you to identify security weaknesses faster.

What Are DISA STIG Compliance Levels?

There are three DISA STIG compliance levels, called categories. The categories indicate the severity of the risk of failing to address a particular weakness.

From most to least severe, these are:

- Category I.
- Category II.
- Category III.

CATEGORY I

Category I refers to any vulnerability that will directly and immediately result in loss of confidentiality, availability, or integrity. What's more, these vulnerabilities can allow unauthorized access to classified data or facilities. This can lead to a denial of service or access.

These risks are the most severe. They may result in loss of life, damage to facilities, or a mission failure. If you don't address these risks, you won't be granted an Authorization to Operate.

The only exceptions are:

- When the system is critical.
- When a failure to use the system could lead to a failed mission.

CATEGORY II

Category II refers to any vulnerability that can result in loss of confidentiality, availability, or integrity.

Category II vulnerabilities can:

- Lead to a Category I vulnerability.
- Result in personal injury, damage to equipment or facilities.
- Degrade a mission.

CATEGORY III

Category III refers to any vulnerability that degrades measures to protect against loss of confidentiality, availability, or integrity.

Category III vulnerabilities can:

- Lead to a Category II vulnerability.
- Delay in recovering from an outage.
- Affect the accuracy of data and information.

What Is OWASP and OWASP Top 10

OWASP is the Open Web Application Security Project. It's an international nonprofit organization that educates software development teams on how to conceive, develop, acquire, operate, and maintain secure applications.

All of the organization's materials (which includes articles, methodologies, documentation, tools, and technologies) are freely available and easily accessible. These materials improve application security through people, process, and technology.

What Are The OWASP Top 10?

The OWASP Top 10 is the most well-known resource that the organization produces. Each year, a team of security experts from across the globe updates the report. This report features the 10 most critical web application and API security risks.

The current list includes:

1. Injection
2. Broken Authentication
3. Sensitive Data Exposure
4. XML External Entities (XXE)
5. Broken Access Control
6. Security Misconfiguration
7. Cross-Site Scripting (XSS)
8. Insecure Deserialization
9. Using Components With Known Vulnerabilities
10. Insufficient Logging and Monitoring

OWASP Top 10: A Closer Look

1. INJECTION

Injection flaws occur when untrusted data is sent as part of a command or query. The attack can then trick the targeted system into executing unintended commands. Or it could give the cybercriminal access to protected data.

2. BROKEN AUTHENTICATION

Authentication is often implemented incorrectly. This enables cybercriminals to take advantage of security vulnerabilities. They can even gain access to users' identities.

3. SENSITIVE DATA EXPOSURE

Sensitive data is often not properly protected. This enables cybercriminals to easily take advantage of security vulnerabilities.

4. XML EXTERNAL ENTITIES (XXE)

Older or poorly configured XML processors evaluate external entity references within XML documents. These external entities can be used by cybercriminals to gain access to sensitive information. They could even launch a denial of service attack.

5. BROKEN ACCESS CONTROL

User restrictions are often not properly enforced. This creates security vulnerabilities that cybercriminals can exploit.

6. SECURITY MISCONFIGURATION

Security misconfiguration is often a result of:

- Insecure default configurations.
- Incomplete or ad hoc configurations.
- Open Cloud storage.
- Misconfigured HTTP headers.
- Verbose error messages that contain sensitive information.

7. CROSS-SITE SCRIPTING (XSS)

Cross-site scripting flaws occur when an application includes untrusted data in a new webpage — without proper validation. It can also occur when an existing web page is updated with user-supplied data using a browser API that can create HTML or JavaScript. Cybercriminals take advantage of cross-site scripting to execute scripts in the targeted system.

8. INSECURE DESERIALIZATION

Deserialization flaws often result in remote code execution. This enables cybercriminals to perform replay, injection, and privilege escalation attacks.

9. USING COMPONENTS WITH KNOWN VULNERABILITIES

Components — such as libraries or frameworks — run the same privileges as the application. A vulnerable component can be exploited by a cybercriminal. This causes serious data loss or server takeover.

10. INSUFFICIENT LOGGING AND MONITORING

Insufficient logging and monitoring can enable cybercriminals to attack systems. They can even tamper, extract, or destroy data.

What Is PA DSS?

PA DSS — the Payment Application Data Security Standard — is a global security standard. It applies to the development of payment application software. It used to be known as the Payment Application Best Practices (PABP).

Why PCI DSS Is Important

It is expected that all payment applications that handle credit card information are safe and secure for customers to use. That's why PCI DSS is important.

All major credit card companies rely on the Payment Card Industry Security Standards Council (PCI SSC). This ensures that payment applications are protected from potential software vulnerabilities.

The PCI SSC works to ensure safety and security by enforcing standards, like PCI DSS. Every organization that handles credit cards need to comply with PCI DSS. This ensures customers' data is handled in a secure manner.

PA DSS: PART OF PCI DSS

The Payment Application Data Security Standard is part of PCI DSS. Software vendors that make and sell payment applications need to follow the standard. This ensures the security of all the software components of an application that processes payment card data.

If payment applications are not compliant with the standard, it could result in major fines. It also leaves customers' personal information vulnerable to data breaches. For that reason, it is important that you understand the payment security standard and how you can meet payment application compliance requirements.

Important Requirements for PA DSS Compliance

The payment security standard requirements apply to storing, processing, and transmitting cardholder data and sensitive authentication data.

The following requirements are required for all organizations who handle credit card information:

- Do not retain full track data, card verification c ode or value (CAV2, CID, CVC2, CVV2), or PIN block data.
- Protect stored cardholder data.
- Provide secure authentication features.
- Log payment application activity.
- Develop secure payment applications.
- Protect wireless transmissions.
- Test payment applications to address vulnerabilities and maintain payment application updates.
- Facilitate secure network implementation.
- Cardholder data must never be stored on a server connected to the internet.
- Facilitate secure remote access to payment applications.
- Encrypt sensitive traffic over public networks.
- Secure all non-console administrative access.
- Maintain a PA DSS Implementation Guide for customers, resellers, and integrators.
- Assign PA DSS responsibilities for personnel. And maintain training programs for personnel, customers, resellers, and integrators.

How to Achieve PA DSS Compliance?

Here's how to achieve payment application compliance by using a SAST tool, such as Klocwork.

The payment security standard provides requirements for developing payment application software. But only Requirement 5 — Develop secure payment applications — is directly involved.

1. COMPLY WITH REQUIREMENT 5

Requirement 5 outlines the process for how to develop secure payment applications. It requires that development processes are "based on industry standards and/or best practices". Code must be reviewed to ensure that it has been developed according to secure coding guidelines.

2. APPLY CODING STANDARDS

To ensure compliance, the standard requires best-practice secure coding guidance. This includes OWASP, CWE, and CERT. The results of code reviews and any resulting code changes also need to be documented.

3. TRAIN DEVELOPERS IN SECURE CODING

So, it is important that developers are trained in secure programming practices. This will mean they can identify potential coding flaws that can lead to application vulnerabilities.

4. USE SAST TOOLS TO ENFORCE IT AUTOMATICALLY

Unfortunately, even the most experienced developers can miss subtleties in code that can cause security issues. This is where automated SAST tools become essential.

A good SAST tool — like Klocwork — enforces best practice coding knowledge. Klocwork automatically scans code — as it is created — to immediately alert developers to security issues. Plus, you can track the status of issues. A change record will be automatically created for traceability and auditing purposes.

What Is IEC 62443?

IEC 62443 is a set of security standards for the secure development of Industrial Automation and Control Systems (IACS). It provides a thorough and systematic set of cybersecurity recommendations. It's used to defend industrial networks against cybersecurity threats.

What Are IEC 62443 Security Levels (SL)?

A key part is security levels (SL). SL is used to assess the cybersecurity risks to each system. And it helps you understand how to best address those risks.

There are five Security Level values, 0–4. SL 0 is the minimum level of risk and SL 4 is the maximum. That means that SL 4 has stricter compliance requirements than SL 0.

SECURITY LEVEL 0

No specific requirements or security protection necessary.

SECURITY LEVEL 1

Protection against casual or coincidental violation.

SECURITY LEVEL 2

Protection against intentional violation using simple means with:

- Low resources.
- Generic skills.
- Low motivation.

SECURITY LEVEL 3

Protection against intentional violation using sophisticated means with:

- Moderate resources.
- IACS specific skills.
- Moderate motivation.

SECURITY LEVEL 4

Protection against intentional attacks with sophisticated means with:

- Extended resources.
- IACS specific skills.
- High motivation.

7 Security Level Foundational Requirements

There are seven specific foundational requirements that must be met for each SL. Meeting these requirements ensures that an IACS has the right security and safety safeguards.

The foundational requirements for an SL are:

1. IDENTIFICATION AND AUTHENTICATION CONTROL

Reliably identify and authenticate all users (humans, software processes, and devices) attempting to access the ICS.

2. USE CONTROL

Enforce the assigned privileges of an authenticated user (human, software process, or device) to perform the requested action on the system or assets. Monitor the use of these privileges.

3. SYSTEM INTEGRITY

Ensure the integrity of the IACS to prevent unauthorized manipulation.

4. DATA CONFIDENTIALITY

Ensure the confidentiality of information on communication channels and in data repositories. Prevent unauthorized disclosure.

5. RESTRICTED DATA FLOW

Segment the control system via zones and conduits to limit the unnecessary flow data.

6. TIMELY RESPONSE TO EVENTS

Respond to security violations. Notify the proper authority reporting needed evidence of the violation. Take timely corrective action when incidents occur.

7. RESOURCE AVAILABILITY

Ensure the availability of the control system against the degradation or denial of essential services.

Each foundational requirement has multiple conditions that need to be met depending on the SL. The higher the SL, the more conditions that must be met for the foundational requirement.

How to Comply with IEC 62443?

IEC 62443 provides guidance on how to ensure the secure development of an IACS. But only Part 4-1 directly applies to software development. Part 4-1 of the standard outlines the framework for how to enforce security standards compliance. An important part of that framework is the use of a static code analyzer.

A static code analyzer automatically identifies vulnerabilities and defects as you code. In addition, you are able to apply a coding standard to ensure that your software is compliant and reliable.

IEC 62443 requires that a static code analyzer be used to enforce secure coding standards.

These include:

- CWE
- CERT
- OWASP
- DISA STIG

This helps to ensure that your code is secure. And it keeps your code free from software vulnerabilities, reliability, and other general coding errors.

How to Apply Secure Coding Standards

The best way to ensure secure coding is to use a static code analyzer.

Static code analyzers enforce coding rules and flag security violations. Both Helix QAC and Klocwork come with code security modules to ensure secure software.

Each one includes:

- Fully documented rule enforcement and message interpretation.
- Extensive example code.
- Fully configurable rules processing.
- Compliance reports for security audits.

How Perforce SAST Tools can Help Enforce Secure Coding Standards

Klocwork is the most accurate and trusted SAST tool for C, C++, C#, and Java. It provides software development teams with the ability to automate source code analysis as the code is being written. And, Klocwork has been designed to easily scale to projects of any size.

What's more, Klocwork's Differential Analysis enables teams to perform very fast incremental analysis on only the files that have changed while providing results equivalent to those from a full project scan. This ensures the shortest possible analysis times.

In addition, Klocwork provides software developers with the following benefits:

- Detecting code vulnerabilities, compliance issues, and rule violations earlier in development. This helps to accelerate code reviews as well as the manual testing efforts of developers.
- Enforcing industry and coding standards.
- Reporting on compliance over time and across product versions.

See for yourself how Klocwork can help you identify security vulnerabilities earlier in development. Request your free trial today.

START TRIAL

perforce.com/products/sca/free-static-code-analyzer-trial

About Perforce

Perforce powers innovation at unrivaled scale. With a portfolio of scalable DevOps solutions, we help modern enterprises overcome complex product development challenges by improving productivity, visibility, and security throughout the product lifecycle. Our portfolio includes solutions for Agile planning & ALM, API management, automated mobile & web testing, embeddable analytics, open source support, repository management, static & dynamic code analysis, version control, and more. With over 9,000 customers, Perforce is trusted by the world's leading brands, including NVIDIA, Pixar, Scania, Ubisoft, and VMware. For more information, visit www.perforce.com.