Werkzeuge zur automatischen Codeanalyse

Werkzeuge zur automatischen Codeanalyse

[expand title="mehr lesen…"]

Werkzeuge zur automatischen Codeanalyse

Quelltext im Fokus

Andreas Wiegenstein

Unternehmen sind gut beraten, auf Tools zurückzugreifen, die Entwickler und Tester durch automatisierte Qualitätsanalysen entlasten. Doch worauf sollten Unternehmen achten, wenn sie einen Codescanner einsetzen wollen?

-Tract

- Code automatisch zu scannen und zu testen, gewinnt in immer mehr Unternehmen an Bedeutung, gerade bei zugekaufter, fremder Software in der eigenen Codebasis.
- Kosten und Ressourcenaufwand wachsen, je später Tests ausgeführt werden. Integration und Automatisierung sowie hohe Erkennungsraten erweisen sich als Kernfaktoren für gute Codescanner.
- Bei der Anschaffung sollten Unternehmen genau darauf

achten, was sie brauchen, und sich nicht vom Funktionsumfang blenden lassen. Eine Evaluation vor der Anschaffung spart später viel (Nach-)Arbeit.

Hacken ist eine Wachstumsbranche. Und weil fast alle Sicherheitslücken auf Programmierfehlern basieren, kommt jeder, der Software schreibt, unweigerlich früher oder später mit dem Thema Applikationssicherheit in Kontakt – je früher, desto besser. Bei der Entwicklung von Software konkurrieren allerdings Hunderte von Qualitätsanforderungen aus unterschiedlichen Bereichen um die Aufmerksamkeit und die Ressourcen der Entwickler. Und praktisch niemand kann diese alle kennen, geschweige denn richtig umsetzen, insbesondere wenn das Programmierprojekt einem gewissen Zeit- und Budgetdruck unterliegt. Gute Codeanalyse kann da sowohl finanziell als auch zeitlich einiges einsparen helfen, vom Ärger ganz zu schweigen.

Die Informationen in dieser Marktübersicht stammen zum großen Teil aus öffentlich verfügbaren Beschreibungen der Scanner im Internet, wobei sich die Suche nach genauen technischen Angaben zu den unterstützten Sprachen, Testfällen, Testverfahren und Plattformen bisweilen schwierig gestaltet. Auf den folgenden Seiten finden Sie eine Tabelle mit Kurzbeschreibungen zu 35 Tools zur Codeanalyse sowie eine ausführliche Tabelle mit vier exemplarisch ausgewählten, sehr unterschiedlichen, aber typischen Vertretern für das, was der Markt anbietet: CAST Application Intelligence Platform, Perforce Klocwork, Micro Focus Fortify und Veracode. Die Tabelle „Sprachvergleich“ zeigt die Sprachunterstützung der vier Werkzeuge im direkten Vergleich.

Sprachenwirrwarr

An erster Stelle steht dabei natürlich die Frage, ob ein einziges Tool alle vom Unternehmen verwendeten Programmiersprachen untersuchen kann. Manche Werkzeuge sind auf nur eine Sprache ausgerichtet, andere versprechen, mehrere

Sprachen untersuchen zu können. Es liegt daher nahe, dass große Unternehmen in der Regel die Tools mit den breiten Sprachpaletten bevorzugen. Sie liefern eine zentrale Plattform zur Analyse und lassen sich auch einfacher warten. Allerdings sind diese generalistischen Tools häufig nicht so gut auf die einzelnen Sprachen ausgerichtet, was sich in der Benutzbarkeit und Testqualität niederschlagen kann.

Kosten sind ein weiterer wichtiger Faktor. Dabei sind aber nicht nur Lizenzen und Support zu berücksichtigen – sofern keine Open-Source-Software zum Einsatz kommt. Auch die laufenden internen Kosten für den Betrieb, wie etwa dedizierte Hardware und das Ausrollen von Updates sowie damit verbundene Downtime, sollten in die Entscheidung einfließen.

Aber es entstehen noch weitere Kosten, die den meisten Unternehmen zunächst nicht bewusst sind. Jedes Tool zur statischen Codeanalyse (SCA) verwendet bestimmten Techniken, um Schwachstellen zu identifizieren.

Dies können einfache Mustersuchen sein, um festzustellen, ob eine bestimmte gefährliche Funktion verwendet wird. Es können aber auch komplexe Verfahren wie Datenflussanalysen sein, die zu ermitteln versuchen, ob während der Verarbeitung auch wirklich externe Eingaben an die bekannten gefährlichen Funktionen gelangen können.

Falsch ist nicht gleich falsch

Bei solchen Untersuchungen kann es aber in der statischen Analyse zu Fällen kommen, in denen nicht eindeutig feststellbar ist, ob tatsächlich ein Risiko vorliegt. In so einem Fall werden die Ergebnisse – je nach Strategie des Herstellers – entweder verworfen, trotzdem als Fehler gezeigt oder als potenzielle Fehler eingestuft.

Alle drei Wege verursachen Aufwand und Kosten beim Kunden. Wenn etwa bekannt ist, dass das Tool bei bestimmten Tests

falsch negative Ergebnisse liefert, dann muss das Unternehmen ein ergänzendes Testverfahren aufbauen, um diesen Mangel zu kompensieren, zumindest wenn diese Tests für das Unternehmen hohe Priorität haben.

Kommt es zu falsch positiven Ergebnissen, dann entstehen Kosten durch die unnötige Analyse des als fehlerhaft gemeldeten Codes, Diskussionen mit entnervten Entwicklern oder durch übereifrige Korrekturen, die eigentlich nicht nötig sind.

Kommt es zu Ergebnissen, die nur als potenzielle Fehler eingestuft sind, so muss das Unternehmen eine Strategie entwickeln, bis zu welchem Grad solche Ergebnisse zu ignorieren sind und welche manuell nachuntersucht werden müssen. Diese Managemententscheidung kann einen großen Unterschied machen und bringt erhebliche Konsequenzen mit sich: Je nachdem, wie umfangreich der eigene Code ist und wie viele Testfälle das eingesetzte Tool verwendet, laufen durch solche zusätzlichen Analysen unter Umständen enorme Kosten auf.

Das gilt insbesondere wenn ein Unternehmen große Mengen an Code untersucht, der aus einer Zeit ohne prüfbare Sicherheitsvorgaben stammt oder durch den Erwerb von Firmen oder Produkten zur Codebasis hinzukam, wo die Entwickler schlimmstenfalls keine oder abweichende Standards bei der Softwareerstellung befolgten.

Testfälle auflisten

Unternehmen sollten von der Analyse möglicher Tools eine Liste unverzichtbarer Testfälle erstellen. Diese Liste muss alle Programmierfehler enthalten, die dem Unternehmen großen Schaden zufügen würden. Für alle möglichen Programmierfehler aus der Liste sollten Programme mit verschiedenen Ausprägungen der Fehler erstellt werden, wobei vorher festgelegt wird, welche Ausprägungen ein Tool als Fehler einstufen müsste und

welche nicht. Auf dieser Basis lässt sich abschätzen, ob und wo es zu relevanten falsch positiven oder falsch negativen Ergebnissen kommen kann.

Außer der Ergebnisqualität gibt es aber noch andere Anforderungen an die technischen Fähigkeiten des gewählten Werkzeugs. Manche Unternehmen versprechen sich von Tools insbesondere einen umfassenden Satz an Know-how in Form von standardmäßig enthaltenen Testfällen. Niemand sollte jedoch blind das Tool mit den meisten ausgelieferten Testfällen kaufen. Stattdessen braucht es ein Tool, das genau die Fehler aufspüren kann, die im eigenen Unternehmen und in den untersuchten Sprachen den größten Schaden anrichten können.

In jedem Fall sollten Unternehmen eine qualifizierte Auswahl treffen, welche Tests im produktiven Betrieb den größten Sicherheitsnutzen mit möglichst geringem Aufwand bringen. Und es kann durchaus sein, dass im Laufe der Zeit eigene Tests hinzukommen müssen, weil bestimmte Anforderungen zu speziell sind, als dass sie ein allgemeines Tool abdecken würde, oder weil das Unternehmen schnell auf neue Anforderungen reagieren muss. Es ist vorteilhaft, die Standardeinstufungen der Fehler im Tool anpassen zu können, falls im Unternehmen abweichende Richtlinien gelten.

Zu den technischen Fähigkeiten gehören aber auch Angaben, wie schnell eine bestimmte Menge Code untersucht werden kann. Das ist wichtig, wenn beispielsweise Scans, bei denen es um große Mengen an Code geht, nachts oder innerhalb bestimmter Zeitfenster stattfinden müssen. Oder vielleicht wenn geplant ist, die Ergebnisse stündlich an ein SIEM weiterzugeben. Aber auch wenn Entwickler ihre Arbeit einer Zwischenprüfung unterziehen möchten, sollte der Einsatz des Tools nicht zu einer erweiterten Kaffeepause führen.

Haptik und Zielgruppe

Das führt direkt zur Handhabung des Tools. Auf welche

Zielgruppen ist es ausgerichtet? Testteams benötigen Berichte mit Kritikalitätseinstufungen und gut nachvollziehbaren Fehlerbeschreibungen, aus denen sie gegebenenfalls weiterführende manuelle Tests ableiten können. Manager brauchen eine zentrale Sicht aller Risiken und der Fortschritte, die durch den Einsatz des Tools erzielt wurden. Auditoren benötigen exportierbare Berichte, die revisionssicher verfasst sind, also nicht manipuliert werden können. Entwickler hingegen haben den größten Nutzen, wenn das Tool direkt in ihre IDE integriert ist, also während des Programmierens bereits Feedback zu Fehlern geben kann, entweder auf Knopfdruck oder – besser – fortlaufend, ähnlich einer interaktiven Rechtschreibprüfung bei Textverarbeitungssoftware.

Mit einem Scanner, der direkt in der IDE aktiv ist, erzielen Firmen den besten Wirkungsgrad. Je früher eine Sicherheitslücke im Code erkannt wird, desto kostengünstiger kann sie behoben werden. Fehler, die direkt beim Schreiben des Codes auffallen, verursachen keinen organisatorischen Aufwand, keine Nachtests, keine ungeplanten Sonderschichten oder sonstigen Änderungen am aktuellen Projekt. Sie erzielen außerdem unmittelbar einen Lerneffekt beim Entwickler. Dafür müssen dann in der IDE natürlich Hinweise abrufbar sein, was genau das Problem ist und wie es vermieden werden kann. Die Qualität der Beschreibungen und Verbesserungsvorschläge ist für alle Beteiligten elementar.

Rekursiv? Nicht immer

Eine weitere interessante Erkenntnis aus der Praxis ist, dass nicht alle Scanner die von ihnen vorgeschlagenen Lösungen auch erkennen, nachdem der Entwickler den Code korrigiert hat und diesen dann erneut scannt. Auch das manuelle Unterdrücken falsch positiver Ergebnisse oder die Behandlung von Ausnahmen funktionieren nicht immer wie erwartet – sofern diese Features überhaupt vorhanden sind. Bei einer Evaluierung sollte man

daher auf jeden Fall prüfen, was passiert, wenn man Code ändert, der mit der unterdrückten Schwachstelle nicht in Zusammenhang steht, aber in der gleichen Funktion vorkommt, in der die Ausnahme angelegt wurde. Als ergänzender Test empfiehlt sich, in einer solchen Funktion eine zweite Schwachstelle gleichen Typs anzulegen, um zu ermitteln, ob eventuell sogar mehrere Schwachstellen gleichen Typs in derselben Funktion durch den Mechanismus verdeckt werden. Es gibt tatsächlich Tools mit solchen Designfehlern.

Auch aus operativen Gesichtspunkten gibt es Anforderungen an SCA-Tools. Soll das Werkzeug nur im eigenen Unternehmensnetz (on Premises) laufen oder ist auch der Einsatz eines Cloud-Produkts denkbar? Dies hängt sicher davon ab, wie viel geistiges Eigentum im Quellcode steckt und was der Datenschutzbeauftragte dazu sagt, wie weit man einem (ausländischen) Anbieter diesen Code und insbesondere auch die darin befindlichen Sicherheitsfehler anvertrauen möchte. Ja, Hacken ist eben die angesprochene Wachstumsbranche. Ebenso wichtig ist auch die Frage, ob das Tool selbst bei einer lokalen Installation regelmäßig Daten mit dem Hersteller austauscht. Solche „Features“ muss der Kunde klären, um später böse Überraschungen zu vermeiden. Nicht zuletzt sollten sich Unternehmen auch informieren, wie viele Sicherheitspatches für das Tool bisher veröffentlicht wurden. Niemand sollte der Illusion verfallen, dass Sicherheitstools automatisch sicher sind. Da kann es lehrreich sein, mal beim Hersteller nachzufragen, wie der Prozess für das eigene Schwachstellenmanagement aussieht.

Neben den rein funktionalen Auswahlkriterien für SCA-Tools spielen noch andere Aspekte beim produktiven Einsatz von Scannern eine Rolle. Bei den meisten Unternehmen, die dem Autor bekannt sind, wurde nach dem Kauf eines Scanners zunächst eine Menge alter Code gescannt, der noch nie konsequent und flächendeckend auf Schwachstellen untersucht worden war. Dabei wurden möglichst viele Testfälle des Tools

aktiviert, um den Istzustand und die Leistungsfähigkeit des Tools zu ermitteln. Dieses Vorgehen ist verständlich, kann jedoch Komplikationen mit sich bringen.

Fallout vom Wirtschaftsprüfer

Denn falls der untersuchte Code zu einer Software gehört, die im Unternehmen Finanzdaten verarbeitet, wie etwa in SAP üblich, dann müssen die Ergebnisse solcher Scans auch dem Wirtschaftsprüfer vorgelegt werden. Nun sind Wirtschaftsprüfer aber nur selten Experten für Applikationssicherheit. Es kann also passieren, dass deren Prüfbericht verlangt, dass schlichtweg *alle* Meldungen, die dem Wirtschaftsprüfer vorliegen, behoben werden müssen. Das kann schnell zu hohen Kosten und ungeplanten, bisweilen vermeidbaren Arbeiten führen.

Nicht nur deshalb sollten Unternehmen klein anfangen, zumindest was die Analyse von Bestandscode angeht. Sie sollten langsam und in mehreren Phasen vorgehen. Vor dem Einsatz des Tools muss feststehen, welche Tests für das Unternehmen relevant sind und welche Arten von Schwachstellen es in einer ersten Phase beheben will und kann. Ebenso muss ermittelt werden, welche Tests mit einem hohen manuellen Prüfungsaufwand verbunden sind.

Projektteams sollten darauf achten, dass sie während einer Korrekturphase keine Updates des Scanners einspielen. Falls der Hersteller Veränderungen an der Testlogik vorgenommen hat, kann das auch Meldungen beeinflussen, die in der aktuellen Phase bearbeitet werden und so zu ungeplantem Mehraufwand führen.

Code verwerfen oder doch korrigieren?

Bevor Korrekturen an fehlerhaftem Code durchgeführt werden, ist immer zu prüfen, ob der überhaupt noch gebraucht wird. Code löschen oder stilllegen ist die bei Weitem sicherste und

günstigste Art, Fehler zu beheben. Manche Tools bieten auch automatisierte Codekorrekturen an. Passiert das in einer IDE, kann der Entwickler sofort erkennen, ob die Korrektur seinen Anforderungen entspricht. Passieren die Korrekturen aber im Hintergrund als Massendatenverarbeitung, kann dies sehr unschöne Effekte haben.

Viele Probleme im Umgang mit komplexen Tools werden leider erst nach längerer Nutzung ersichtlich. Beispielsweise könnte die Qualität des Supports unzureichend sein. Oder der Hersteller wird von einem größeren (ausländischen) Unternehmen aufgekauft, das ganz andere Ziele verfolgt. Unternehmen sind daher gut beraten, von Anfang an über eine Strategie nachzudenken, wie man den verwendeten Scanner irgendwann wieder ablösen könnte.

Fazit

Es gibt nicht den einen idealen Codescanner. Jedes Unternehmen sollte solche Produkte nach den Eigenschaften auswählen, die für den eigenen Einsatzzweck am besten geeignet sind. Dazu zählen vor allem die verwendeten Programmiersprachen und Entwicklungsplattformen. Zusätzlich ist es erforderlich, technische Grenzen und Schwächen der Scanner zu kennen, um diese durch ergänzende Maßnahmen möglichst weitgehend zu kompensieren. Erst nach dieser tiefen Analyse ist es möglich, zusätzliche Kosten in die Gesamtkostenbetrachtung einzukalkulieren. Wer hier die Kosten minimieren will, kommt nicht umhin, vor der Anschaffung Arbeitszeit in sinnvolle Tests zu investieren. (mfe@ix.de)

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
http://awap.sourceforge.net/	WAP	Open Source	PHP	Verwendet Datenflussanalyse mit 8 Testfällen für gängige Schwachstellen; bietet eine automatisierte Korrektur von Fehlern.
http://clang-analyzer.llvm.org/	Clang Static Analyzer	Open Source	C, C++, Objective-C	Führt eine Codeanalyse während des Build-Prozesses durch; einfache Mustererkennung für Sicherheitsfehler, sucht aber auch funktionale Fehler.

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
http://sparrow.fasoo.com/en/	SPARROW	proprietär	C/C++, Android Java, Java, Objective-C, JSP, HTML, C#, SQL, XML, ABAP, PHP, ASP.NET, Python, VB.NET, Swift, JavaScript, APEX, VBScript, Visualforce, XSL	Bietet Integration in die IDE, Build-Systeme und Versionskontrollsysteme; umfangreiche Testfälle, basierend auf Compliance-Standards von CWE, OWASP und anderen.
https://github.com/Microsoft/DevSkim	DevSkim	Open Source	C, Objective C, C++, C#, COBOL, Go, Java, JavaScript/TypeScript, PHP, PowerShell, Python, Ruby, Rust, SQL, Swift, Visual Basic	Liefert ein plattformübergreifendes Command-Line-Interface und IDE-Plug-ins für Visual Studio und Visual Studio Code. Die Plug-ins liefern dem Entwickler direktes Feedback zu Fehlern, inklusive Erklärungen und Korrekturvorschlägen.
https://brakemanscanner.org/	Brakeman	Open Source	Ruby	Brakeman ist ein kostenloser Schwachstellenscanner, der speziell für Ruby-on-Rails-Anwendungen entwickelt wurde.
https://discotek.ca/deepdive.xhtml	Deep Dive	Freeware	Ear, War, Jar, APK	Statisches Codeanalysetool für JVM Deployment Units. Hat einen erweiterbaren Satz an Testfällen.
https://discotek.ca/sinktank.xhtml	Sink Tank	Freeware	Java	Verwendet Datenflussanalyse und eine nicht genannte Zahl an Testfällen. Kann nur Bytecode analysieren, inkl. kompilierter JSPs.
https://dotnet-security-guard.github.io/	Roslyn Security Guard	Open Source	.NET	Analysiert .NET- und .NET-Core-Projekte im Hintergrund (IntelliSense) oder während eines Builds. Hat nur begrenzte Testfälle und verwendet einfache, intraprozedurale Analysen.
https://github.com/ajinabraham/nodejsscan	nodejsscan	Open Source	Node.js	Open-Source-Scanner für Node.js-Anwendungen
https://github.com/designsecurity/progpilot	Progpilot	Open Source	PHP	Open-Source-Scanner für PHP-Anwendungen
https://github.com/PyCQA/bandit	Bandit	Open Source	Python	Bietet ca. 70 Testfälle für Python-Anwendungen, verwendet dabei Mustersuchen. Läuft stand-alone und erzeugt Berichte.
https://github.com/securego/gosec	Gosec	Open Source	Go	Bietet ca. 30 Testfälle für Go-Anwendungen, verwendet dabei Mustersuchen.
https://github.com/wireghoul/graudit/	Graudit	Open Source	ActionScript, Android, ASP, C, COBOL, .NET, exec, fruit, Go, iOS, Java, js, Perl, PHP, Python, Nim, Ruby	Open-Source-Tool, das mit einfacher Mustererkennung arbeitet. Erfordert manuelle nachträgliche Analyse der Ergebnisse.
https://huskyci.opensource.globo.com/	HuskyCI	Open Source	Ruby, Go, Python, JavaScript, Java	Verwendet verschiedene Testssets für die Fehlersuche, wie z. B. gosec, Bandit und Brakeman.
https://pumascanpro.com/	Puma Scan	proprietär	C#, ASPX, .cshtml	Bietet umfangreiche Testfälle für C#-Anwendungen, verwendet dabei Mustersuchen und Datenflussanalyse. Arbeitet als Plug-in für Visual Studio und für .NET Frameworks.
https://rubygems.org/gems/dawnscanner	Dawnscanner	Open Source	Ruby	Verwendet Datenflussanalyse und eine nicht genannte Zahl an Testfällen. Liefert auch Lösungsvorschläge.

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
https://www.securityreviewer.net/	Security Reviewer	proprietär	C#, VB.NET, VB6, ASP, ASPX, Java, JSP, JavaScript, TypeScript, eScript, Java Server Faces, APEX, Ruby, Python, R, Go, Kotlin, Groovy, Flex, ActionScript, PowerShell, Lua, HTML5, XML, XPath, JSON, C, C++, PHP, Scala, Rust, IBM Streams SPL, Objective-C, Objective-C++, Swift, COBOL, JCL, RPG, PL/I, ABAP, SAP HANA, UiPath, BPMN, BPEL, SAIL, PL/SQL, T/SQL, U-SQL, Teradata SQL, SAS-SQL, ANSI SQL, IBM DB2, IBM Informix, MySQL, FireBird, PostGreSQL, SQLite, MongoDB	Verwendet komplexe Analyseverfahren mit mehr als 1100 Prüfregeln. Breite Palette an Schnittstellen und Plug-ins für IDEs und CI/CD-Lösungen. Kann auch Binärcode untersuchen. Basiert auf mehreren innovativen und patentierten Verfahren.
https://sourceforge.net/projects/cppcheck/	Cppcheck	Open Source	C, C++	Verwendet besondere Analyseverfahren zur Erkennung von Schwachstellen in C-Code. Liefert Plug-ins für viele Entwicklungswerkzeuge.
https://sourceforge.net/projects/visualcodegrepp/	VisualCodeGrepper (VCG)	Open Source	C++, C#, VB, PHP, Java, PL/SQL, COBOL	freier statischer Codescanner für 32-Bit-Windows-Plattformen
https://spotbugs.github.io/	SpotBugs	Open Source	Java	Verwendet mehr als 400 Tests zur Fehlererkennung. Integration in Ant, Maven, Cradle und Eclipse möglich.
https://www.adacore.com/codepeer	CodePeer	proprietär	Ada	
https://www.adacore.com/sparkpro	SPARK tool set	proprietär	Spark	Verwendet verschiedene komplexe und mathematische Analyseansätze. Vollständige Integration in GNAT Studio.
https://www.blueclosure.com	BlueClosure BC Detect	proprietär	JavaScript	Untersucht Code, der mit JavaScript Frameworks wie Angular.js, jQuery, Meteor.js oder React.js verwendet wird. Kann auch verschleierten Code untersuchen.
https://www.codacy.com/	Codacy	proprietär	APEX, Bash, C, CoffeeScript, C++, C#, Crystal, CSS, Dockerfile, Elixir, Go, Groovy, Java, JavaScript, JSON, JSP, Kotlin, LESS, Markdown, PHP, PLSQL, PowerShell, Python, Ruby, SASS, Scala, Swift, TSQL, TypeScript, Velocity, Visual Basic, Visualforce, XML	Unterstützt eine breite Palette an Sprachen. Als On-Premises- und Cloud-Lösung verfügbar. Arbeitet direkt in GIT Repositories.
https://www.codescan.io/	CodeScan Cloud	proprietär	Salesforce	Scanner für die Salesforce-Plattform; als On-Premises- und Cloud-Lösung verfügbar.

Hersteller, Lizenzen und Sprachen				
Hersteller	Scanner	Lizenz	Sprache(n)	Kommentar
https://www.grammatech.com/products/codesonar	CodeSonar	proprietär	C/C++, Java, C#, Intel- und ARM-Binärdateien	CodeSonar unterstützt die Standards MISRA-C, MISRA-C++, AUTOSAR C++-14, CERT, DISA STIG, OWASP, CWE. Schwachstellen werden persistent verwaltet und über Builds hinweg verfolgt, selbst wenn sich der Code ändert. Sie können mit Anmerkungen versehen, in eine Rangfolge gebracht, zugewiesen, gesucht und verglichen werden. Unterstützung für Teamtools wie Jenkins, Visual Studio, Eclipse, Jira, GitLab und Docker.
https://www.kiuwan.com/	Kiuwan	proprietär	ABAP, ActionScript, ASP.NET, C, C#, C++, COBOL, Go, HANA SQL Script, HTML, Informix, Java, JavaScript, TypeScript, JCL, JSP, Kotlin, Natural, Objective-C, Oracle Forms, PHP, PL-SQL, PowerScript, Python, RPG4, Scala, SQL, Swift, Transact-SQL, VB.NET, Visual Basic 6, XML	Deckt gängige Sicherheitsprobleme ab. Compliance-Berichte decken folgende Standards ab: OWASP, CWE, MISRA, NIST, PCI und CERT. Unterstützt die IDEs von Eclipse, Visual Studio, IntelliJ IDEA, PhpStorm, PyCharm and WebStorm. Benötigt eine Internetverbindung.
https://www.mathworks.com/products/polyspace-bug-finder.html	Polyspace Bug Finder	proprietär	C, C++	Überprüft laut Hersteller die Einhaltung von Programmierstandards wie z. B. MISRA-C, MISRA-C++, JSF++, CERT C, CERT C++.
https://www.mathworks.com/products/polyspace-code-prover.html	Polyspace Code Prover	proprietär	C, C++	Analysiert Quellcode auf eine Reihe üblicher Fehler, verwendet dabei Mustersuchen und Datenflussanalysen. Arbeitet mit Eclipse.
https://www.parasoft.com/products	Parasoft Test	proprietär	Java, C, C++, C#	Verwendet über 2500 Regeln für die Analyse. Compliance-Berichte automatisieren die Dokumentationsanforderungen für Standards wie MISRA, AUTOSAR und CERT. Die Analyselogik verwendet Machine-Learning-Algorithmen.
https://www.reshiftsecurity.com	reshift	proprietär	Java	Deckt OWASP Top 10 ab. Optimiert für Entwickler und den SDL. Schwachstellen werden nach Risikostufen in der modernen IDE gruppiert und haben detaillierte Beschreibungen. Generiert Codevorschläge für die Behebung der Fehler inklusive GIT Request.
https://www.viva64.com/en/pvs-studio/	PVS-Studio Analyzer	proprietär	C, C++, C#, Java	Verwendet verschiedene Analyseverfahren zur Erkennung von Schwachstellen. Wertet Konstanten und Variablen aus, um Pufferüberläufe zu ermitteln. Trial Version verfügbar.
https://www.xanitizer.com/xanitizer/	Xanitizer	proprietär	Java, JavaScript/TypeScript, Angular	Laut Hersteller hat das Tool über 100 Testfälle und konnte die OWASP Benchmark Test Suite mit 100% Erfolg analysieren. Der Hersteller merkt allerdings an, dass man diese Genauigkeit nicht in echtem Code erwarten darf. Analysiert nur Bytecode.

Vier Hersteller im Featurevergleich				
	CAST Application Intelligence Platform	Perforce Klocwork	Micro Focus Fortify	Veracode
Scan-Engine				
Wie hoch ist die Falsch-positiv-Rate (Schätzung)?	<5%	<10%	k. A.	<5%
Wie hoch ist die Falsch-negativ-Rate (Schätzung)?	<5%	<10%	k. A.	k. A.
Wie viele Testfälle hat das Tool?	k. A.	1000	800	k. A.
Können Kunden eigene Testfälle entwickeln?	–	✓	✓	–
Funktioniert die Analyse auch mit Code, der (noch) nicht kompiliert werden kann?	✓	✓	✓	–
Können Binaries oder Objektcode ebenfalls untersucht werden?	✓	–	teilweise	✓
Kann man falsch positive Ergebnisse für künftige Scans unterdrücken? Wenn ja, ist dieser Mechanismus stabil genug, um auch Änderungen im umgebenden Code zu verkraften?	–	✓ / ✓	✓ / ✓	✓ / ✓
Erkennt das Tool die Implementierung der von ihm vorgeschlagen Methoden der Fehlerbehebung zuverlässig?	✓	✓	✓	✓
Hat das Tool Einschränkungen bezüglich des untersuchten Codes, verwendeter Libraries oder Frameworks, der Binaries oder des Objektcodes?	keine	keine	keine	keine
Kann der Kunde die Standardbewertungen der Testfälle ändern?	–	✓	✓	✓
Worauf basiert die Analyse? Kommen nur simple Pattern-Matching-Methoden oder auch komplexere Daten- und Kontrollflussanalysen zum Einsatz?	komplexe Analysen	komplexe Analysen	komplexe Analysen	komplexe Analysen
Installation und Wartung – Installation and Maintenance				
Wird das Tool on Premises installiert oder ist es eine Cloud-Lösung?	beides möglich	beides möglich	beides möglich	Cloud
Kann das Tool (auf einem Server) im Continuous-Monitoring-Betrieb laufen / Code automatisiert untersuchen? Gibt es SIEM-Support?	✓ / ✓	✓ / k. A.	✓ / ✓	✓ / ✓
Ist das Tool in die IDE integrierbar? Falls ja, bietet es automatische Korrekturen für Fehler an?	✓ / k. A.	✓ / ✓	✓ / ✓	✓ / k. A.

Vier Hersteller im Featurevergleich				
	CAST Application Intelligence Platform	Perforce Klocwork	Micro Focus Fortify	Veracode
Wird das Tool zentral installiert/gewartet oder auf jedem System einzeln?	lokale Installationen, zentrale Sammlung der Ergebnisse	beides möglich	beides möglich, in jedem Fall zentrale Sammlung der Ergebnisse	–
Auf welchen Betriebssystemen läuft das Tool?	Windows, Linux	Windows, Linux, SUN Solaris, IBM Power	Windows, Linux, macOS	–
Welche Versionen der Programmiersprache/Runtime werden unterstützt?	32x und 64x	k. A.	siehe Dokumentation auf Webseite	siehe Dokumentation auf Webseite
Wartung				
Wie häufig kommen neue Features/Versionen?	wöchentlich	alle 3 Monate	2 Major-Produkt-Releases und 4 Rulepack-Updates pro Jahr. Die Rulepack-Updates aktualisieren die für den Scan genutzte Schwachstellendatenbank.	mindestens einmal pro Monat
Gibt es eine (relevante) Downtime bei Updates oder Upgrades?	Downtime < 3 Minuten	Downtime kann vermieden werden	Nein, da es möglich ist, zwei verschiedene Scan-Engines parallel zu installieren. Für den SaaS-Bereich gibt es ein Upgrade ohne Downtime.	Nur in seltenen Fällen handelt es sich um eine Downtime-Release – in diesen Fällen werden die Anwender darauf aufmerksam gemacht.
Lassen sich Updates automatisiert einspielen und benötigen sie mehrere manuelle Schritte?	Updates sind manuell	Updates sind manuell.	Bei dem Cloud-Dienst sind Updates automatisch. On-Premises-Installationen benötigen manuelle Updates.	automatisch (in der Cloud)
Ergebnispräsentation				
Werden gefundene Fehler ausführlich erklärt?	✓, inklusive Lösungsvorschlägen	✓, inklusive Lösungsvorschlägen	✓, inklusive Lösungsvorschlägen	✓
Kann man die Ergebnisse einzeln exportieren / in ein externes Ticketing-System überführen? Automatisch?	✓, z. B. Jira	✓ (mittels REST-API)	✓	✓, z. B. Jira, Azure DevOps, GitLab oder GitHub (API)
Wird die Fehlerbehebung ausführlich erklärt?	✓	✓	✓ Es gibt zusätzlich noch die Möglichkeit, das Thema in E-Learning-Videos zu vertiefen.	✓ Veracode bietet zusätzlich die Möglichkeit, Fragen mit Experten via WebEx zu klären.
Verfügt das Tool über ein interaktives Dashboard?	✓	✓	✓	✓
Lassen sich gefundene Fehler revisionssicher dokumentieren?	✓	✓	✓	✓
Sicherheit/Security				
Benötigt das Tool eine Onlineverbindung zum Hersteller?	–	–	–	✓

Vier Hersteller im Featurevergleich				
	CAST Application Intelligence Platform	Perforce Klocwork	Micro Focus Fortify	Veracode
Übermittelt das Tool (regelmäßig) Daten an den Hersteller? Wenn ja, welche?	–	–	–	✓, aber nur den zu testenden Binärcode
Wie viele Sicherheitspatches wurden für das Tool bisher herausgegeben?	k. A.	<5	k. A.	k. A.
Kosten				
Lizenz (OSS oder proprietär; bei OSS, welche Lizenz?)	proprietär	proprietär	proprietär	proprietär, jährliche Subscription
Wie wird lizenziert – nach Entwicklern / nach Codevolumen / nach untersuchten Systemen / nach Firmengröße / …?	Anzahl Entwickler und LOCs	Anzahl Benutzer	on Premises: nach Anzahl von Applikationen und Installationen oder nach Anzahl der Entwickler; on Demand: Pay-per-Use-Modell bzw. nach der Anzahl der Scans	Lizenzkosten richten sich nach der Anzahl der Application Profiles.
Wie hoch sind die jährlichen Wartungskosten (in Prozent zum Kaufpreis)?	20%	20%, wenn eine Lizenz erworben wurde	on Premises: 28%; on Demand: Support und Wartungskosten sind bereits im Preis enthalten.	drei Stufen für 10, 20 und 27 Prozent des Lizenzwertes

Sprachenvergleich																		
Hersteller	ABAP	C	C++	C#	Delphi/Object Pascal	Haskell	Java	JavaScript	Kotlin	Lua	PHP	Objectiv-C	Python	Ruby	Rust	Swift	Visual Basic	Visual Basic .NET	
Veracode	–	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓	✓	–	✓	✓	✓	vollständige Liste der unterstützten Sprachen und Frameworks auf Webseite
Micro Focus	✓	✓	✓	✓	–	–	✓	✓	✓	–	✓	✓	✓	✓	–	✓	✓	✓	Plus: COBOL, ActionScript, APEX, ASP.NET, ColdFusion, Go, JSP, TypeScript, PL-SQL, T/SQL, Scala, ASP VBScript
Perforce	–	✓	✓	✓	–	–	✓	–	–	–	–	–	–	–	–	–	–	–	
CAST	✓	✓	✓	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	vollständige Liste der unterstützten Sprachen und Frameworks auf Webseite

Andreas Wiegenstein

ist Geschäftsführer beim SAP-Cybersecurity-Unternehmen SERPENTEQ.

[/expand]