Prüfmethoden für Securtity

des Kooperationsprojektes

CyberProtect

Veröffentlichung: 1.1.2019	Version: 0.1
Autoren: Anne Borcherding, Niklas Goerke, Jonas Klamroth	
Review:	

Baden-Württemberg
MINISTERIUM FÜR WIRTSCHAFT, ARBEIT UND WOHNUNGSBAU

Gefördert vom Ministerium für Wirtschaft, Arbeit und Wohnungsbau Baden-Württemberg

Aktenzeichen Zuwendungsbescheid:
3-4332.62-FZI/53

Anne Borcherding, Niklas Goerke, Jonas Klamroth

Inhaltsverzeichnis

1 Kurzfassung

Dieses Dokument bietet eine Übersicht über verschiedene Prüfmethoden für Software Sicherheit (im Sinne des englischen Wortes „Security“). Hierzu werden Kategorien und Werkzeuge vorgestellt und auf entsprechende Literatur verwiesen. Die Auswahl der Kategorien und Werkzeuge erfolgte aufgrund von wissenschaftlichen Veröffentlichungen [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11] und persönlichen Erfahrungen. Die Liste ist nicht abschließend, insbesondere gibt es in vielen Kategorien sehr viele Werkzeuge die nur für einen sehr speziellen Anwendungsfall entwickelt wurden, diese aufzulisten würden den Umfang dieses Dokuments übersteigen.

Zunächst werden die Kategorien beschrieben und die ihnen zugeordneten Werkzeuge gelistet. Anschließend folgt die Beschreibung der ausgewählten Werkzeuge / Prüfverfahren die den Kategorien zugeordnet sind. Tabelle 1 gibt eine Übersicht über die beschriebenen Werkzeuge und Verfahren, Tabelle 5 zeigt eine Klassifizierung der beschriebenen Kategorien nach verschiedenen Merkmalen.

Da sich die Kategorien nicht klar voneinander abgrenzen lassen, viele Verfahren in mehreren Kontexten angewandt werden können und viele Werkzeuge mehrere Funktionen anbieten, sind die Kategorien nicht diskjunkt und viele Verfahren sind mehreren Kategorien zugeordnet.

2 Kategorisierung der Prüfverfahren

Hier werden die verschiedenen Kategorien von Prüfverfahren aufgelistet und beschrieben. Außerdem gibt eine große Tabelle einen Überblick über die verschiedenen Tools.

2.1 Verwundbarkeitsscanner

Beschreibung

Verwundbarkeitsscanner haben die Zielsetzung bekannte Schwachstellen in dem zu testenden System zu finden. Dazu senden sie zunächst vordefinierte Eingaben an das zu testende System und werten anschließend die Antworten aus. Auf Basis der Antworten kann darauf geschlossen werden ob eine bekannte Schwachstelle wahrscheinlich vorliegt oder nicht. Diese Überprüfungen können auf vielen verschiedenen Ebenen durchgeführt werden. Im Rahmen dieser Literaturrecherche lag der Fokus auf generischen Verwundbarkeitsscannern und Verwundbarkeitsscannern für Webanwendungen. Felderer et al. [3] und Bau et al. [8] geben einen guten Überblick über die Thematik der Verwundbarkeitsscanner. Das Open Web Application Security Project (OWASP) führt eine ständig aktualisierte Liste mit Verwundbarkeitsscannern [12].

Prüfverfahren

- Nessus 4.30
- OpenVAS 4.34
- Greenbone Security Manager 4.21
- Qualys 4.36
- Nexpose 4.31
- Acunetix 4.2
- Vega 4.50
- Wapiti 4.51
- Skipfish 4.41
- ZAP 4.55
- Nikto 4.32
- Metasploit 4.29
- Nmap 4.33

2.2 Konformitätstest

Beschreibung

Bei Konformitätstests wird überprüft, ob die Schnittstellen eines Systems bestimmten Spezifikationen entsprechen. Dies können beispielsweise Spezifikationen für das Verhalten bei der Kommunikation über ein Netzwerkprotokoll oder auch Spezifikationen wie der IT Grundschutz sein.

Prüfverfahren

- Achilles 4.1

- OpenVAS 4.34
- Greenbone Security Manager 4.21
- Nessus 4.30
- Qualys 4.36
- Nexpose 4.31
- ISuTest 4.22

2.3 Formale Methoden

Beschreibung

Bei formalen Methoden geht es im Gegensatz zu den meisten anderen Verfahren darum Verifikation statt Falsifikation zu betreiben. Es wird bewiesen (deduktiv oder über (geschickte) vollständige Suche), dass ein Programm eine implizit oder explizit gegebene Spezifikation erfüllt. Diese Spezifikation kann von einfachen Eigenschaften wie *es treten keine Nullpointer-Exceptions auf* bis hin zu komplexen funktionalen Eigenschaften reichen. Der Vorteil der Garantie die eine solche Methode bietet wird durch den meist höheren Aufwand ausgeglichen.

Prüfverfahren

- KeY 4.25
- JavaPathfinder 4.24
- CBMC 4.6
- SPARK 4.43
- MALPAS 4.28
- CPAchecker 4.11
- Frame-C 4.16
- BLAST 4.4

2.4 Statische Code-Analyse

Beschreibung

Bei statischer Code-Analyse werden Fehler in gegebenem Quellcode gesucht, ohne diesen auszuführen. Hierbei können einfach Fehler wie Syntaxfehler gefunden werden, aber auch verschiedene schwerer zu findende Fehler. Unter anderem können auf Grund bspw. der vorkommenden Konstanten im Quellcode teilweise automatisiert Tests erstellt werden oder Warnung beim Auftreten verschiedener Antipatterns ausgegeben werden[13].

Prüfverfahren

- Splint 4.45
- FACT 4.15
- Checkstyle 4.7
- SpotBugs 4.46
- Covertity 4.10
- Klocwork 4.26
- Xanitizer 4.54

2.5 Dynamische Code-Analyse

Beschreibung

Bei dynamischer Code-Analyse wird der zu testende Code ggf. compiliert und dann mit speziellen Testparametern ausgeführt. Hierbei können Fehler erkannt werden, die von Laufzeitparametern abhängig sind und durch statische Code-Analyse daher nicht entdeckt werden können. Dynamische Code-Analyse kann vorhandene Fehler finden, im Regelfall aber nicht die Abwesenheit von Fehlern garantieren.

Prüfverfahren

- Frama-C 4.16
- SPARK 4.43
- Unit Tests 4.47

2.6 Code Review

Beschreibung

Als Codereview allgemein wird der Prozess bezeichnet, wenn Quellcode nachdem er geschrieben wird von einer Person (die nicht der Autor ist) auf Fehler überprüft wird (oder allgemeiner auf Qualität). Teilweise kann dieser Prozess automatisiert werden dann redet man jedoch von statischer Code-Analyse (siehe 2.4). Früher wurde Code-Analyse sehr formal betrieben was einen hohen Aufwand mit sich gebracht hat, jedoch auch nachgewiesener Maßen erhöhte Codequalität garantiert hat [14]. Heute werden Codereviews weniger formal/aufwändig betrieben. Es zeigt sich allerdings, dass dadurch trotzdem eine höhere Codequalität erreicht wird und außerdem soziale Aspekt wie Lernen von Experten oder Zusammengehörigkeitsgefühl der Entwickler auch eine wichtige Rolle spielen [15]. Entscheidend für den Erfolg von Reviews ist außerdem die Expertise des Reviewers [16].

Prüfverfahren

- GitLab 4.19
- Crucible 4.12
- Collaborator 4.8

2.7 Risk Driven

Beschreibung

Risk driven Software Testing ist eine Vorgehensweise zur Testplanung. Hierbei werden diejenigen Bereiche des zu testenden Systems identifiziert, die das größte Schadenspotential haben um die Tests auf diese Bereiche fokussieren zu können. Dies betrifft alle Phasen des Testprozesses, also die Planung, das Design, die Implementierung, die Durchführung und die Evaluation[3]. Risiko basiertes Software Testing ist daher keine Kategorie, sondern eine Kategorie übergreifende Methodik. Als solche enthält sie auch keine Tools.

2.8 Modelbasiert

Beschreibung

Modellbasiertes Security Testing verwendet Modelle um zu überprüfen, ob ein (Software-) System den gestellten Security-Anforderungen genügt. Dabei werden explizite Modelle verwendet, die das System under Test und/oder die Umgebung beschreiben. Aus diesen Modellen werden dann konkrete Testfälle für das System abgeleitet. Dabei ist eine klare Trennung zu der Kategorie des Security Testing mit formalen Methoden teilweise nicht möglich. Zu Methoden des modellbasierten Security Testing gibt es in der Literatur einige Studien [9, 3, 1].

2.9 Model checking

Beschreibung

Bei Model Checking wird das zu untersuchende System als endlicher Automat modelliert. Es kann dann gezeigt werden, dass gegebene Eigenschaften (oftmals als Temporallogik-Formeln gegeben) für diesen Automaten gelten, in dem über vollständige Suche gezeigt wird, dass vom Startzustand aus keine Zustände erreicht werden können, die diese Eigenschaft verletzen [17]. Die Suche wird in Praxis meist von SAT- oder SMT-Solvern übernommen. Für Hardwareverifikation ist diese Methode auf Grund der hohen Automatisierbarkeit und den endlichen Zustandsräumen weit verbreitet, sie wird aber auch für Softwareverifikation eingesetzt.

Prüfverfahren

- Spin 4.44
- CBMC 4.6
- JPF 4.24
- Questa Formal 4.37

2.10 Taint Analysis

Beschreibung

Taint Analyse ist eine Technik um das Verarbeiten von Eingaben in einem Programm zu verfolgen. Beispielsweise können Nutzereingaben derart verfolgt werden, dass alle Variablen die damit „infiziert“wurden identifiziert werden, um so festzustellen, welche Ausgaben davon beeinflusst werden. Dies kann unter Anderem dabei helfen zu identifizieren, ob eine Nutzereingabe Einfluss auf eine SQL-Query hat und somit eine Gefahr für eine SQL-Injection besteht.

Taint Analyse kann sowohl statisch wie auch dynamisch erfolgen. Statische Taint Analyse arbeitet auf dem Quellcode des Programms und wird dabei regelmäßig viele falsch positive Ergebnisse liefern. Dynamische Taint Analyse ist komplexer und es ist meist schwerer, Fehler damit zu entdecken [18]

Prüfverfahren

- Flowdroid 4.17
- Gotcha 4.20

2.11 Fuzzing

Beschreibung

Fuzzing beschreibt die Überprüfung eines Systems durch randomisierte Eingaben. Durch die randomisierten Eingaben werden hierbei auch Testfälle abgedeckt, die möglicherweise nicht der Spezifikation der Eingaben entsprechen. So kann überprüft werden, wie ein System auf unerwartete Eingaben reagiert. Die einzusetzenden Prüfverfahren hängen stark von dem Ziel ab das gefuzzt werden soll. Die hier vorgestellten Fuzzer legen ihren Fokus auf das Fuzzen von Netzwerkpaketen. Pfrang et al. geben einen Überblick über verschiedene Prüfverfahren in der Kategorie des Fuzzing für Netzwerkpakete und beschreiben außerdem eine Taxonomie der Verfahren [10]. Felderer et al. beschäftigen sich in ihrer Arbeit ebenfalls mit Fuzzing [3]. Das Open Web Application Security Project (OWASP) führt eine ständig aktualisierte Liste mit Prüfverfahren für das Fuzzing [19] und auch BlackArch Linux stellt eine entsprechende Liste bereit [20].

Prüfverfahren

- Boofuzz 4.5
- Peach 4.35
- ISuTest 4.22

2.12 Penetration Test

Beschreibung

Werkzeuge für Penetrations Tests (Pentests) werden zur Durchführung von Sicherheitstests von Systemen eingesetzt. Sie können einzelne Arbeitsschritte automatisieren, müssen aber für die spezifische Aufgabe ausgesucht und konfiguriert werden. Verwundbarkeitsscanner sind eng mit Pentest Werkzeugen verwandt, sind aber stärker automatisiert.

Da viele Pentest Werkzeuge für einen sehr speziellen Einsatzzweck entwickelt wurden, und daher eine sehr große Zahl existiert, behandeln wir in diesem Dokument nur die wichtigsten. Weitere sind in den folgenden Listen gesammelt:

- https://github.com/enaqx/awesome-pentest
- https://inventory.rawsec.ml/tools.html

Prüfverfahren

- Metasploit 4.29
- Nmap 4.33
- ZAP 4.55
- DoS 4.13

2.13 Fault Injection

Beschreibung

Fault Injection ist eine Technik mit der Soft- oder Hardware derart modifiziert wird, dass bisher ungetestete Funktionen ausgeführt und somit getestet werden. Bei Software können insbesondere errorhandling Zweige oft nicht getestet werden, weil die dafür „benötigten“Fehler im normalen Betrieb nicht vorkommen. Fault Injection kann entweder auf den Quellcode oder zur Laufzeit angewendet werden. Beim Modifizieren des Quellcodes wird z.B. Code eingefügt, der Variablen manipuliert (aber semantisch korrekt hält) um so ein Ausführungspfad zu erreichen, der zuvor nie erreicht wurde. Bei zur Laufzeit eingesetzter Fault Injection können z.B. gespeicherte Daten oder die Kommunikation mit anderen

Prozessen (z.B. Syscalls, Netzwerkpakete) manipuliert werden. Fault Injection kann auch auf Hardware durchgeführt werden, z.B. durch überbrücken zweier Kontakte.

Weitere Werkzeuge sind in einer losen Sammlung in diesem Repository[1] zu finden.

Prüfverfahren

- libfiu 4.27
- CORDS 4.9
- Simmy 4.40
- W-SWIFT 4.53

2.14 Robustheittests

Beschreibung

Bei Robustheitstests werden Systeme auf ihr Verhalten beim Auftreten von Fehlern und anderen Grenzsituationen untersucht. Hierfür werden fehlerhafte oder z.B. übermäßig viele Eingaben generiert und beobachtet, in wie weit das System trotzdem in der Lage ist, seine Funktionalität bereitzustellen. Robustheitstests sind eng mit Fuzzing (siehe 2.11) und Fault Injection (siehe 2.13) verwandt, beinhalten aber auch z.B. DOS Attacken.

Prüfverfahren

- CORDS 4.9
- JasperGold 4.23
- W-SWIFT 4.53
- DoS 4.13

3 Übersichtstabelle

Tabelle 1 bietet eine Übersicht über die hier untersuchten Prüfverfahren.

[1] https://github.com/skibum55/chaos-as-a-service

Achilles Test Plattform	GE Digital	?	kommerziell	standalone bzw. Windows	Konformitätstest 2.2
CORDS	Aishwarya Ganesan, Ramnatthan Alagappan	15.10.2018 (letzter Commit)	MIT	Linux	Fault Injection 2.13
ISuTest	Fraunhofer IOSB	30.03.2017	kommerziell	Linux	Fuzzing 2.11
Nikto	Chris Sullo	09.07.2015	GPL	Linux, Windows	Verwundbarkeitsscanner 2.1
MALPAS	Atkins	?	kommerziell	MS Windows 95/98/2000/XP/Vista/7/8	Formale Methoden 2.3
Java Pathfinder	NASA	2.7.2018	Apache 2.0 License	Linux, Windows, MacOS	Formale Methoden 2.3
Frame-C	CEA-List und INRIA	29.11.2018	GNU LGPL v2	Windows, Linux, MacOS	Formale Methoden 2.3
Rodin/Event-B	University of Southhampton	21.2.2018	open source	Windows, Linux, MacOS	Formale Methoden 2.3
libfiu	Alberto Bertogli et al.	13.12.2018	BOLA (Public Domain)	Linux, MacOS	Fault Injection 2.13
Boofuzz	Joshua Pereyda	12.03.2019	GPL-2.0	Linux, Windows, MacOS	Fuzzing 2.11
Questa Formal Verification	Mentor Graphics	?	kommerziell	?	Formale Methoden 2.3
SPARK	AdaCore	2.2019	kommerziell	Linux, Windows, MacOS	Formale Methoden 2.3
Klocwork	RogueWave	28.2.2018	kommerziell	Windows, Linux, MacOS	Statische Code-Analyse 2.4
Solidify	Averant	12.12.2017	kommerziell	Win NT, Win 2000, Sun Solaris, Linux, HP-UX	Formale Methoden 2.3
KeY	KIT, TU Darmstadt, Chalmers University	11.10.2017	GNU GPL	Linux, Windows, MacOS	Formale Methoden 2.3
Nessus	Tenable Network Security	Fast täglich neue Plugins	kommerziell	Linux, Windows, Mac	Verwundbarkeitsscanner 2.1

Crucible	Atlassian	14.2.2019	kommerziell	Webapplikation	Code Review 2.6
Uppaal	Uppsala University, Aarlborg University	20.05.2014	kommerziell	Linux, Windows, MacOS	Model checking 2.9
Metasploit	Rapid7	10.01.2019 (core)	BSD	Linux, Windows	Penetration Test 2.12
Nmap	Fyodor et al.	20.03.2018	GPLv2 ähnlich	Linux, MacOS, Windows	Verwundbarkeitsscanner 2.1
Denial of Service (DoS)	n/a	n/a	n/a	alle	Robustheittests 2.14
Wapiti	Nicolas Surribas	11.05.2018	GNU GPL v2	Linux, Windows, MacOS	Verwundbarkeitsscanner 2.1
Gotcha	Anna-Katharina Wickert	04.08.2017 (letzter Commit)	MIT	Linux	Taint Analysis 2.10
Nexpose	Rapid7	01.05.2019	kommerziell	Windows	Verwundbarkeitsscanner 2.1
Acunetix	Acunetix	26.03.2019	kommerziell	Windows, Linux	Verwundbarkeitsscanner 2.1
Valgrind	Valgrind Development Team	12.04.2019	GPL	Linux, MacOS, Android	Dynamische Code-Analyse 2.5
CBMC	Daniel Kroening, Edmung Clarke	18.12.2018	BSD License	Linux, Windows, MacOS	Formale Methoden 2.3
Skipfish	M. Zalewski, N. Heinen und S. Roschke (Google)	22.12.2012	Apache-2.0	Linux, Windows, MacOS	Verwundbarkeitsscanner 2.1
Splint	University of Virginia	12.7.2007	GNU GPL	Linux, Windows	Statische Code-Analyse 2.4
FACT	Fraunhofer FKIE	4.4.2019	GNU GPL v3	Linux	Statische Code-Analyse 2.4
JasperGold	Cadence	8.5.2019	Commercial	Linux	Formale Methoden 2.3
Checkstyle	communitybasiert	28.4.2019	GNU LGPL v2.1	Windows, Linux, MacOS	Statische Code-Analyse 2.4
Simmy	Dylan Reisenberger et al.	10.05.2019 (letzter Commit)	BSD	Windows (für C#)	Fault Injection 2.13
Wireshark	Community	21.05.2019	GPLv2	Windows, Linux, macOS	Penetration Test 2.12

Qualys Cloud Plattform	Qualys	20.02.2019	kommerziell	-	Verwundbarkeitsscanner 2.1
Collborator	SmartBear Software	26.11.2018	kommerziell	Windows, Linux, MacOS, Solaris	Code Review 2.6
BLAST	USCD	30.10.2015	Apache 2.0 License	Linux	Formale Methoden 2.3
Alloy	AlloyTools	12.04.2019	Apache-2.0	Linux, Windows, MacOS	Model checking 2.9
Enemy of the State	Adam Doupé et al.	15.01.2014	GPL-2.0	Ubuntu	Fuzzing 2.11
Covertiy	synopsys	?	kommerziell	Windows, Linux, MacOS	Statische Code-Analyse 2.4
GNU Debugger (GDB)	GNU-Projekt	11.05.2019	GPL	GNU/Linux, Windows	Penetration Test 2.12
GitLab	GitLab Inc.	22.2.2019	kommerziell/MIT License	Webapplikation	Code Review 2.6
SpotBugs	communitybasiert	1.3.2019	GNU LGPL	Windows, Linux, MacOS	Statische Code-Analyse 2.4
CPAchecker	LMU München (Dirk Beyer et al.)	Dezember 2018	Apache 2.0 License	Linux, (Mac, Windows)	Formale Methoden 2.3
FlowDroid	Steven Arzt et al.	21.01.2019	LGPL v2.1	Windows, Mac OS und Linux	Taint Analysis 2.10
W-SWFIT	Paul Jordan	02.12.2017 (letzter Commit)	MIT	Windows	Fault Injection 2.13
Peach Fuzzer	Peach Tech	10.10.2016	kommerziell / Community (MIT)	Linux, Windows, MacOS	Fuzzing 2.11
ZAP	OWASP	Wöchentliche Releases	Apache-2.0	Windows, Linux, MacOS	Penetration Test 2.12
Xanitizer	RIGS IT	2.4.2019	kommerziell	Windows, Linux	Statische Code-Analyse 2.4
Spin Model Checker	Bell Labs	17.06.2018	BSD 3-Clause	Linux	Model checking 2.9
Unit Tests	n/a	n/a	n/a	alle	Dynamische Code-Analyse 2.5

OpenVAS	Greenbone GmbH	Networks	05.04.2019	GPL-v2.0	Server auf Linux, Zugriff über Web	Verwundbarkeitsscanner 2.1
Vega	Subgraph		03.07.2011	MIT license	Linux, Windows, MacOS	Verwundbarkeitsscanner 2.1
SCMetrics	Jürgen Wüst		2.1.2018	kommerziell	Windows, Linux, MacOS	Statische Code-Analyse 2.4
Greenbone Security Manager	Greenbone GmbH	Networks	30.04.2019	kommerziell	Server auf Linux, Zugriff über Web	Verwundbarkeitsscanner 2.1

Tabelle 1: Übersichtstabelle

4 Beschreibung der Prüfverfahren

4.1 Achilles

Beschreibung

Die Achilles Test Plattform ist ein kommerzieller Verwundbarkeitsscanner der gleichzeitig die Möglichkeit zur Zertifizierung eines Geräts bietet. Es können zum einen die Achilles Zertifizierungen durchgeführt werden. Diese Zertifizierung findet immer mehr Verbreitung, mittlerweile existieren mehrere hundert Komponenten die mit Achilles zertifiziert wurden. Eine Liste zertifizierter Produkte kann hier[2] eingesehen werden. Zum anderen ist die Achilles Test Plattform für die ISASecure[3] Zertifizierung der IEC 62443 Conformance Certification zugelassen. Im Unterschied zu den meisten anderen Verwundbarkeitsscannern umfasst die Achilles Test Platform neben einer Software auch Hardware. So kann der aktuelle Zustand des zu testenden Geräts überwacht und beispielsweise ein Absturz erkannt werden.

Name	Achilles Test Plattform[4]
Autor	GE Digital
Letzter Release	?
Lizenz	kommerziell
Betriebssysteme	standalone bzw. Windows
Hauptkategorie	Konformitätstest 2.2

Kategorisierung

Die Achilles Test Plattform[5] wird von GE Digital vertrieben. Der Hersteller informiert online nicht darüber in welchen Abständen Releases veröffentlicht werden bzw. wann der letzte offizielle Release veröffentlicht wurde. Die Achilles Test Plattform wird kommerziell vertrieben. Es ist möglich, ein vollständiges System inklusive Hardware zu erwerben. Außerdem bietet GE Digital eine reine Softwarelösung an, welche ein Windows Betriebssystem benötigt. Im Rahmen von internen Tests am IOSB wurden verschiedene Geräte mit Hilfe der Achilles Test Platform untersucht, bei der großen Mehrzahl wurden mehrere Schwachstellen und die nicht vorhandene Robustheit aufgedeckt. Insbesondere für die Tests die im Rahmen der Level 1 und 2 Zertifizierung durchgeführt werden, ist die Achilles Test Platform eine geeignete Testplattform. Für weitere unterstützte Protokolle wie Profinet oder OPC US lässt sich dies nicht feststellen. Hier sind Tests teilweise nicht sinnvoll konfigurierbar und teilweise fehlerhaft. Die Tests werden über eine Benutzeroberfläche gestartet und können von außen nicht automatisiert konfiguriert oder gestartet werden. Alle Tests die Achilles durchführt sind black box Tests. Achilles kann in folgende Kategorien eingeordnet werden:

[2] https://www.ge.com/digital/applications/achilles-communications-certified-products
[3] https://www.isasecure.org/en-US/
[4] https://www.ge.com/digital/applications
[5] https://www.ge.com/digital/applications

1. Verwundbarkeitsscanner 2.1
2. Konformitätstest 2.2

Ähnliche Prüfverfahren

- Nessus 4.30
- OpenVAS 4.34
- Greenbone Security Manager 4.21
- Qualys 4.36
- Nexpose 4.31
- SAINT
- RETINA

4.2 Acunetix Vulnerability Scanner

Beschreibung

Acunetix bietet zum einen einen generischen Verwundbarkeitsscanner an und zum anderen auch einen Verwundbarkeitsscanner der speziell auf Webanwendungen zugeschnitten ist. Es ist möglich die Scans aus der Cloud oder aus einer lokalen Instanz zu konfigurieren und durchzuführen. Der Verwundbarkeitsscanner kann sowohl als black box als auch als white box Scanner eingesetzt werden. Für einen white box Test muss eine Softwarekomponente in das zu untersuchende Gerät integriert werden, welche dann interne Scans durchführen und interne Informationen nach außen führen kann.

Name	Acunetix[6]
Autor	Acunetix
Letzter Release	26.03.2019
Lizenz	kommerziell
Betriebssysteme	Windows, Linux
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Die Acunetix[7] Familie wird von der gleichnamigen Firma Acunetix vertrieben. Der letzte Release wurde am 26.03.2019 veröffentlicht (Stand 08.05.2019). Acunetix vertreibt seine verschiedenen Produkte kommerziell. Es ist möglich die Acunetix Produkte auf Linux oder auf Windows zu verwenden. Der Webserver Verwundbarkeitsscanner von Acunetix (Acunetix WVS) wurde in diversen wissenschaftlichen Studien untersucht [8, 21, 22, 23, 24, 25, 26, 27, 28]. Es ist möglich die Untersuchungen über

[6] https://www.acunetix.com/
[7] https://www.acunetix.com/

eine REST API zu automatisieren. Die Untersuchungen können sowohl black box als auch white box durchgeführt werden. Für einen white box Test muss eine Softwarekomponente in das zu untersuchende Gerät integriert werden, welche dann interne Scans durchführen und interne Informationen nach außen führen kann. Die Verwundbarkeitsscanner von Acunetix kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1

Ähnliche Prüfverfahren

- Nikto 4.32
- Vega 4.50
- Wapiti 4.51
- Skipfish 4.41
- ZAP 4.55

4.3 Alloy

Beschreibung

Mit Alloy können Objekte und Strukturen modelliert werden. Anschließend kann automatisiert überprüft werden, ob das Modell den definierten Spezifikationen entspricht. Das Prüfverfahren kann zum einen genutzt werden um die Konsistenz von Datenstrukturen zu überprüfen. Zum anderen kann es dadurch auch genutzt werden um bestimmte Sicherheitsanforderungen zu evaluieren.

Name	Alloy[8]
Autor	AlloyTools
Letzter Release	12.04.2019
Lizenz	Apache-2.0
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Model checking 2.9

Kategorisierung

Alloy[9] wird von AlloyTools entwickelt. Der letzte Release wurde am 12.04.2019 veröffentlicht (Stand 08.05.3019). Der Quellcode von Alloy ist auf github[10] unter der Apache-2.0 Lizenz verfügbar. Alloy kann auf allen gängigen Betriebssystemen verwendet werden. In der Literatur wurde Alloy in einigen

[8] http://alloytools.org/
[9] http://alloytools.org/
[10] https://github.com/AlloyTools/org.alloytools.alloy

Studien im Kontext der Sicherheit eingesetzt [29, 30, 31, 32]. Alloy arbeitet auf Modellen von Systemen und führt white box Tests durch. Die Modelle müssen manuell erstellt werden, Alloy kann jedoch durch eine API angesprochen werden. Alloy kann in folgende Kategorien eingeordnet werden:

1. Formale Methoden 2.3
2. Model checking 2.9

Ähnliche Prüfverfahren

- Spin 4.44
- Uppaal 4.48

4.4 Berkeley Lazy Abstraction Verification Tool (BLAST)

Beschreibung

Berkley Lazy Abstraction Verification Tool (BLAST) ist ein statisches Software Analysetool was unter anderem für die Verifikation von Linux Treibern eingesetzt wird. In diesem Bereich hat es auch mehrere Wettbewerbe gewonnen[33][34]. Das Tool erlaubt es temporal Safetyeigenschaften von C-Programmen zu beweisen oder Gegenbeispiele zu generieren, falls Ersteres nicht möglich ist.

Name	BLAST[11]
Autor	USCD
Letzter Release	30.10.2015
Lizenz	Apache 2.0 License
Betriebssysteme	Linux
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

BLAST wird als open source Software an der Universität von Berkeley entwickelt. Der letzte Release der Version 2.7.3 fand am 30.10.2015 statt. Seit dem ist die Entwicklung eingestellt worden. Es gibt allerdings ein Nachfolgeprojekt was laut den Autoren „alle essentiellen Features [von Blast] und vieles mehr“ abdeckt. Dieses Projekt heißt CPAchecker (siehe 4.11). BLAST ist ausschließlich für Linux verfügbar und wurde in F# entwickelt. Es kann als formale Methode (genauer Software Modelchecker) kategorisiert werden.

[11] https://forge.ispras.ru/projects/blast/

Ähnliche Prüfverfahren

- CBMC 4.6
- SPARK 4.43
- Java-Pathfinder 4.24
- KeY 4.25
- Frama-C 4.16
- CPAchecker 4.11

4.5 Boofuzz

Beschreibung

Boofuzz ist ein Fuzzer für Netwerkprotokolle und eine Weiterentwicklung des Fuzzers Sulley[12]. Er führt black box Fuzzing aus und benötigt als Grundlage eine Beschreibung des Protokolls das gefuzzt werden soll. Die Eingaben werden generation-based und mutational erzeugt. Im Gegensatz zu dem Vorgänger Sulley ist Boofuzz auch in der Lage seriell zu fuzzen sowie auf dem Ethernet- und IP-Layer.

Name	Boofuzz[13]
Autor	Joshua Pereyda
Letzter Release	12.03.2019
Lizenz	GPL-2.0
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Fuzzing 2.11

Kategorisierung

Boofuzz[14] ist eine Weiterentwicklung des Fuzzers Sulley die von Joshua Pereyda entwickelt wird. Der letzte Release wurde am 12.03.2019 veröffentlicht (Stand 08.05.2019). Die Implementierung liegt in Python vor und ist unter GPL-v2.0 lizenziert. Durch die Implementierung in Python kann Boofuzz auf allen gängigen Betriebssystemen verwendet werden. Vom Autor wird die Verwendung unter Ubuntu empfohlen. Boofuzz wurde in der Literatur bisher noch nicht umfänglich evaluiert, allerdings setzen einige in der Literatur entwickelte Systeme auf Boofuzz auf [35, 36, 37]. Boofuzz führt black box Tests aus, benötigt allerdings eine Protokollbeschreibung des zu fuzzenden Protokolls. Durch die Verwendung über die Kommandozeile können Tests mit Boofuzz automatisiert durchgeführt werden. Boofuzz kann in folgende Kategorien eingeordnet werden:

1. Fuzzing 2.11

[12] https://github.com/OpenRCE/sulley
[13] https://github.com/jtpereyda/boofuzz
[14] https://github.com/jtpereyda/boofuzz

Ähnliche Prüfverfahren

- Peach 4.35

4.6 C-Bounded Model Checker (CBMC)

Beschreibung

CBMC ist ein Tool zur Analyse von C-Programmen. Wie der Name suggeriert, ist CBMC ein Bounded Model Checker und kann daher nur Programmläufe und Speicherbereiche mit beschränkter Größe/Länge untersuchen. In diesen ist das Tool in der Lage typische Fehler Memoryleaks, Index-Out-Of-Bounds-Exceptions, Division-By-Zero-Exceptions oder Null-Pointer-Exceptions zu finden und diese mit konkreten Beispielen zu belegen[38]. Es ist außerdem möglich mit nichtdeterministischen Werten, Annahmen und Assertions auch Aussagen über abstrakte Programmläufe zu treffen. Es gibt zwei Schwesterprojekte die jeweils für Java (JBMC) bzw. Verilog oder SMV (EBMC) ähnliche Tools bereitstellen.

Name	CBMC[15]
Autor	Daniel Kroening, Edmung Clarke
Letzter Release	18.12.2018
Lizenz	BSD License
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

CBMC wird von Daniel Kroening und Edmung Clarke an der University of Oxford und der Carnegie Mellon University entwickelt. Es ist in C geschrieben. Der letzte Release ist die Version 5.11 die im Dezember 2018 veröffentlicht wurde. Es gibt Binaries für jedes gängige Betriebssystem und für einige Linux Distributionen die Möglichkeit CBMC über Packagemanager zu installieren. Das Projekt wird immer noch aktiv gepflegt und steht als open-source auf GitHub[16] zur Verfügung. CBMC kann als statisches formale Methode (genauer Software Modelchecker) kategorisiert werden.

Ähnliche Prüfverfahren

- Java Pathfinder 4.24
- SPARK 4.43
- BLAST 4.4

[15] http://www.cprover.org/cbmc/
[16] https://github.com/diffblue/cbmc

- KeY 4.25
- Frama-C 4.16
- CPAchecker 4.11

4.7 Checkstyle

Beschreibung

Checkstyle ist ein Tool zur Kontrolle der Einhaltung von Codingstyles in Java. Typische Checks die Checkstyle durchführt sind unter anderem: die Reihenfolge von Deklarationen, Einhaltung von Namenskonventionen, Vorhandensein von JavaDoc-Kommentaren und Codeduplikate. Diese Checks können beliebige konfiguriert (ein-/ausgeschaltet) werden. Je nach Compilereinstellungen werden Checkstylefehler als Warnung oder Fehler ausgegeben. Checkstylefehler haben oftmals keine direkten Auswirkungen auf die Funktionalität des Codes, sollen aber zur besseren Les- und Wartbarkeit beitragen. Meist wird Checkstyle als PlugIn für IDEs oder Compiler verwendet.

Name	Checkstyle[17]
Autor	communitybasiert
Letzter Release	28.4.2019
Lizenz	GNU LGPL v2.1
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

Checkstyle ist quelloffen und wird aus GitHub[18] gehosted. Die aktulle Version 8.20 wurde am 28.4.2019 veröffentlicht. Als Javatool ist Checkstyle auf allen gängigen Betriebssystemen lauffähig. Checkstyle ist als statisches Code-Analyse-Tool zu klassifizieren.

Ähnliche Prüfverfahren

- FACT 4.15
- Splint 4.45

[17] http://checkstyle.sourceforge.net/
[18] https://github.com/checkstyle/checkstyle

4.8 Collborator

Beschreibung

Collaborator ist ein Tool zur Verwaltung von Quellcode. Es kann mit verschiedenen Versionskontrollsysteme genutzt werden. Außerdem wird die Integration von unteren Anderen GitHub, GitLab, Bitbucket und VisualStudio unterstützt. In dem Tool ist es möglich Unterschiede zwischen verschiedenen Revisionen anzeigen zu lassen, Kommentare zu Code oder Commits zu schreiben und Issues sowie Pullrequests zu verwalten.

Name	Collborator[19]
Autor	SmartBear Software
Letzter Release	26.11.2018
Lizenz	kommerziell
Betriebssysteme	Windows, Linux, MacOS, Solaris
Hauptkategorie	Code Review 2.6

Kategorisierung

Collaborator wird von SmartBear Software entwickelt. Die neuste Version 11.4 wurde am 26.11.2018 veröffentlicht. Das Tool kann als Codereviewtool klassifiziert werden.

Ähnliche Prüfverfahren

- GitLab 4.19
- Crucible 4.12

4.9 CORDS

Beschreibung

CORDS [39] ist ein Tool zum Testen von verteilten Speichersystemen. Es testet, wie Systeme auf Fehler wie I/O Fehler oder Datenverlust in einer Node reagieren. Hierzu werden zunächst Fehler in einer Node eingebracht und dann beobachtet, wie und ob das Gesamtsystem den Fehler korrigiert. Da die Funktionsweise der getesteten Systeme keine Rolle spielt, handelt es sich um ein black box Tool, die Tests werden automatisiert durchgeführt. Es sind nur für eine begrenzte Zahl von Speichersystemen Plugins vorhanden (Cockroach, Zookeeper, RethinkDB).

[19] https://smartbear.com/product/collaborator/overview/

Name	CORDS[20]
Autor	Aishwarya Ganesan, Ramnatthan Alagappan
Letzter Release	15.10.2018 (letzter Commit)
Lizenz	MIT
Betriebssysteme	Linux
Hauptkategorie	Fault Injection 2.13

Kategorisierung

CORDS wurde von Wissenschaftlern der Universität Wisconsin entwickelt und im Rahmen eines Forschungsprojekts [39] genutzt. Die Weiterentwicklung verläuft nur sporadisch, es ist unklar ob das Projekt noch weiter geführt wird. CORDS wurde für den Einsatz unter Linux (Ubuntu 14.04) entwickelt, und steht unter MIT Lizenz. Es kann in folgende Kategorien eingeordnet werden:

1. Fault Injection 2.13
2. Konformitätstest 2.2
3. Robustheittests 2.14

4.10 Coverity

Beschreibung

Covertiy ist ein Tool was Code (z.B. Java, Javascript, PHP, C) schon während des Programmierens auf potentielle Sicherheitslücken kontrolliert. Dabei zeigt es die Fehler nicht nur sondern erklärt diese auch und versucht somit Entwickler für die Zukunft zu sensibilisieren. Kontrollen die beispielsweise durchgeführt werden sind Injections, unsichere direkte Objektreferenzen und Offenlegung von sensiblen Daten. Covertiy kann sowohl in bekannte IDEs wie IntelliJ, Eclipse oder Visual Studio wie auch in Build- oder CI-Server (z.B. Jenkins) integriert werden. Coverity wurde in [40] mit Klocwork und anderen ähnlichen Tools verglichen.

Name	Covertiy[21]
Autor	synopsys
Letzter Release	?
Lizenz	kommerziell
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Statische Code-Analyse 2.4

[20] http://research.cs.wisc.edu/adsl/Software/cords/
[21] https://www.synopsys.com/software-integrity/security-testing/static-analysis-sast.html

Kategorisierung

Coverity wird von Synopsys entwickelt. Es steht für alle gängigen Betriebssysteme sowie für mehrere IDEs als Plugin zur Verfügung. Covertiy kann als statisches Code-Analysetool kategorisiert werden.

Ähnliche Prüfverfahren

- FACT 4.15
- Splint 4.45
- Checkstyle 4.7

4.11 CPAchecker

Beschreibung

CPAchecker ist ein statisches Softwareanalysetool für C-Programme. Es erlaubt das Beweisen von temporalen Safetyeigenschaften oder, falls nicht möglich, die Konstruktion von Gegenbeispielen. In einem Wettbewerb für Verifikationstools schnitt es 2018 als bestes Tool in der Gesamtwertung ab[41]. CPAchecker kann als Kommandozeilen-Programm oder als Eclipse-Plugin genutzt werden.

Name	CPAchecker[22]
Autor	LMU München (Dirk Beyer et al.)
Letzter Release	Dezember 2018
Lizenz	Apache 2.0 License
Betriebssysteme	Linux, (Mac, Windows)
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

CPAchecker wird hauptsächlich an der Ludwig-Maximilians-Universität in München entwickelt. Es ist in Java geschrieben und steht unter anderem auf GitHub[23] als open-source-Projekt zur Verfügung. Prinzipiell ist CPAchecker als Java-Programm auf allen gängigen Betriebssystemen ausführbar, es werden allerdings spezielle Versionen von SMT/SAT-Solvern benötigt die nur für Linux zur Verfügung gestellt werden (können aber für andere Betriebssysteme selbst gebaut werden). Es kann als formale Methode kategorisiert werden.

[22] https://cpachecker.sosy-lab.org/
[23] https://github.com/sosy-lab/cpachecker

Ähnliche Prüfverfahren

- CBMC 4.6
- SPARK 4.43
- Java-Pathfinder 4.24
- KeY 4.25
- BLAST 4.4
- Frama-C 4.16

4.12 Crucible

Beschreibung

Crucible ist ein webbasiertes Tool zur Verwaltung von Quellcode. Es kann mit verschiedenen Versionskontrollsystemen genutzt werden (git, svn, Mercurial, CVS, Preforce). Das Tool bietet unter Anderem die Möglichkeit Kommentare zu bestimmten Commits oder Codestellen zu schreiben, Issues und Pullrequests zu verwalten und Unterschiede zwischen verschiedenen Revisionen anzuzeigen.

Name	Crucible[24]
Autor	Atlassian
Letzter Release	14.2.2019
Lizenz	kommerziell
Betriebssysteme	Webapplikation
Hauptkategorie	Code Review 2.6

Kategorisierung

Crucible wird von Atlassian entwickelt. Die neuste Version 4.7 wurde am 14.2.2019 veröffentlicht. Als webbasierte Anwendung ist Crucible auf allen gängigen Betriebssystemen lauffähig. Das Tool kann als Codereviewtool klassifiziert werden.

Ähnliche Prüfverfahren

- GitLab 4.19
- Collaborator 4.8

[24] https://www.atlassian.com/software/crucible

4.13 Denial of Service Angriff

Beschreibung

Denial of Service (DoS) ist eine Angriffstechnik bei der versucht wird die Verfügbarkeit eines Dienstes zu unterbrechen indem er gezielt überlastet wird. Hierzu werden meist eine möglichst große Zahl an Anfragen an den Dienst gestellt, sodass diese nicht schnell genug beantwortet werden können und so auch legitime Anfragen nicht mehr beantwortet werden. DoS Angriffe auf verschiedenen Ebenen durchgeführt werden, z.B. gegen einzelne Netzwerkverbindungen, gegen Server Software oder gegen Betriebssysteme (z.B. durch TCP-SYN-Flooding). Da ein einzelner Angreifer oft nicht in der Lage ist genug Anfragen zu senden und es außerdem einfacher ist einzelne Quellen durch Gegenmaßnahmen abzuschalten, werden oft auch distributed Denial of Service Angriffe durchgeführt. Bei diesen Angriffen greifen eine hohe Zahl von Angreifern koordiniert ein Ziel an.

Name	Denial of Service (DoS)
Autor	n/a
Letzter Release	n/a
Lizenz	n/a
Betriebssysteme	alle
Hauptkategorie	Robustheittests 2.14

Kategorisierung

Es gibt eine Vielzahl von Werkzeugen die für Denial of Service Angriffe genutzt werden können. Da für den Angriff nur wenig über das Ziel bekannt sein muss, handelt es sich um black box Angriffe.

DoS Angriffe können in folgende Kategorien eingeordnet werden:

1. Robustheittests 2.14
2. Penetration Test 2.12

4.14 Enemy of The State

Beschreibung

Enemy of the State ist ein modellbasierter Verwundbarkeitsscanner und Fuzzer für Webanwendungen. Dieser Verwundbarkeitsscanner entwickelt in einem ersten Schritt ein Modell des internen Status der Webanwendung. Aus diesem Modell werden dann Testfälle entwickelt, insbesondere Testfälle für das Fuzzing. Außerdem wird das Modell verwendet um den Test zu lenken und die Ergebnisse des Tests auszuwerten.

Name	Enemy of the State[25]
Autor	Adam Doupé et al.
Letzter Release	15.01.2014
Lizenz	GPL-2.0
Betriebssysteme	Ubuntu
Hauptkategorie	Fuzzing 2.11

Kategorisierung

Enemy of the State[26] wurde von Adam Doupé et al. entwickelt [42]. Die vorhandene Implementierung ist laut Beschreibung nur ein Proof of Concept. Sie ist in Java und Python geschrieben und ist unter der GPL-2.0 Lizenz verfügbar. Getestet wurde die Implementierung nur für Ubuntu. Das Prüfverfahren wurde von den Autoren selbst in einer Studie evaluiert [42]. Es führt black box Tests gegen Webanwendungen aus. Durch die Verwendung über die Kommandozeile kann Enemy of the State vollständig automatisiert ausgeführt werden. Enemy of the State kann in folgende Kategorien eingeordnet werden:

1. Model checking 2.9
2. Verwundbarkeitsscanner 2.1
3. Fuzzing 2.11

Ähnliche Prüfverfahren

- Acunetix 4.2
- Vega 4.50
- Wapiti 4.51
- Skipfish 4.41
- Nikto 4.32

4.15 FACT

Beschreibung

FACT ist eine Tool zur Analyse von Firmware. Es erlaubt große Teile der Analyse zu automatisieren, so z.B. das Entpacken der Firmware, die Zusammenstellung von verwendeten Krypto-Materialien (z.B. private Schlüssel oder Zertifikate) und ob schon bekannte Schwachstellen in Teilen der Software wiederentdeckt wurden. Es ermöglicht außerdem den Vergleich verschiedener Firmware Versionen. Das Tool wird über ein Webinterface bedient, welches auf dem lokalen Host läuft.

[25] https://github.com/adamdoupe/enemy-of-the-state
[26] https://github.com/adamdoupe/enemy-of-the-state

Name	FACT[27]
Autor	Fraunhofer FKIE
Letzter Release	4.4.2019
Lizenz	GNU GPL v3
Betriebssysteme	Linux
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

FACT ist quelloffen und wird auf GitHub[28] zum Download zur Verfügung gestellt. Es wird vom Fraunhofer FKIE entwickelt und die aktuellste Version 2.6 wurde am 4.4.2019 veröffentlicht. Es steht aktuell nur für Linux Systeme zur Verfügung. Für Ubuntu ab Version 16.04 existieren auch Installationsskripte. FACT ist als statisches Code-Analyse-Tool zu kategorisieren.

Ähnliche Prüfverfahren

- Splint 4.45

4.16 Frama-C

Beschreibung

Frama-C ist eine von zwei französischen Universitäten entwickelte Toolsuite zur Analyse und Verifikation von C-Code. Es stellt mehrere Plug-Ins zur Verfügung die unterschiedliche Funktionalität anbieten. Unter anderem ist es möglich zu beweisen, dass ein Programm eine ACSL Spezifikation erfüllt, Wertebereiche für Variablen und Ausdrücke in jedem Zustand des Programms anzuzeigen, Slicing zu betreiben und Speicherauswirkungen von einzelnen Ausdrücken oder Funktionen zu analysieren.

Name	Frame-C[29]
Autor	CEA-List und INRIA
Letzter Release	29.11.2018
Lizenz	GNU LGPL v2
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Formale Methoden 2.3

[27] https://fkie-cad.github.io/FACT_core/
[28] https://github.com/fkie-cad/FACT_core
[29] https://frama-c.com/download.html

Kategorisierung

Frama-C ist für alle gängigen Betriebssystem verfügbar und wird immer noch aktiv entwickelt (in OCaml). Das Tool wurde bereits eingesetzt um Bugs in Industrie-Software zu finden [43] und wurde mit ähnlichen Tools verglichen [44]. Je nach genutztem Plug-In kann das Tool in eine der folgenden Kategorien eingeordnet werden:

1. Statische Code-Analyse 2.4
2. Dynamische Code-Analyse 2.5
3. Formale Methoden 2.3

Ähnliche Prüfverfahren

- CBMC 4.6
- SPARK 4.43
- Java-Pathfinder 4.24
- KeY 4.25
- CPAchecker 4.11

4.17 FlowDroid

Beschreibung

Flowdroid ist ein static taint analysis Tool für Java-Programme (inklusive Android-Apps). Es wird eingesetzt, um Eingaben aus konfigurierbaren Quellen (z.B. Nutzereingaben) durch den Quellcode des Programms zu verfolgen. Dadurch können die Einflüsse der Eingaben auf die Ausführung beobachtet und potentielle Sicherheitsprobleme zu identifiziert werden. So können könnten z.B. Eingaben identifiziert werden, die nach mehrfacher Bearbeitung für ein SQL-Statement genutzt werden um potentielle SQL-Injections zu identifizieren. Dieser Prozess läuft nach Konfiguration automatisch ab, da der Quellcode analysiert wird, handelt es sich um eine white box Analyse. FlowDroid wurde in einer Studie mit Iccta, Amandroid und Droidsafe verglichen [45].

Name	FlowDroid[30]
Autor	Steven Arzt et al.
Letzter Release	21.01.2019
Lizenz	LGPL v2.1
Betriebssysteme	Windows, Mac OS und Linux
Hauptkategorie	Taint Analysis 2.10

[30] https://blogs.uni-paderborn.de/sse/tools/flowdroid/

Kategorisierung

FlowDroid wird von Wissenschaftlern der Universitäten Paderborn und Darmstadt sowie dem Fraunhofer SIT in Darmstadt entwickelt. Es ist auf Github[31] gehostet und wird aktiv weiterentwickelt. Da es in Java geschrieben ist, kann es auf allen Betriebssystemen eingesetzt werden die Java unterstützen. FlowDroid kann in folgende Kategorien eingeordnet werden:

1. Taint Analysis 2.10
2. Statische Code-Analyse 2.4

Ähnliche Prüfverfahren

- Gotcha 4.20

4.18 GNU Debugger

Beschreibung

GNU Debugger (GDB) ist ein Werkzeug um Programm zur Laufzeit zu analysieren. Er bietet die Möglichkeit Software zu debuggen indem der Ablauf des Kontrollflusses unter anderem mit Breakpoints verfolgt wird, oder sogar aktiv in den Kontrollfluss einzugreifen indem Variablen manipuliert und Funktionen aufgerufen werden. GDB bietet selbst keine grafische Oberfläche, aufgrund der hohen Verbreitung und des Funktionsumfangs nutzen aber viele andere Programme, wie z.B. Entwicklungsumgebungen GDB als Debugger und bieten somit indirekt auch eine grafische Oberfläche. GDB unterstützt sehr viele Architekturen und einige Programmiersprachen, er bietet Schnittstellen für Erweiterungen.

Name	GNU Debugger (GDB)[32]
Autor	GNU-Projekt
Letzter Release	11.05.2019
Lizenz	GPL
Betriebssysteme	GNU/Linux, Windows
Hauptkategorie	Penetration Test 2.12

Kategorisierung

GDB wird vom GNU-Projekt entwickelt und steht daher auch unter der freien GPL. Der letzte Release, Version 8.3, ist vom 11.05.2019. GDB wird aktiv entwickelt, es ist für Unixoide Betriebssystem und Windows verfügbar. Eine Einteilung als ein black box oder white box Verfahren ist schwer, da GDB je

[31] https://github.com/secure-software-engineering/FlowDroid
[32] https://www.gnu.org/software/gdb/

nach dem wie viele Informationen zur Verfügung stehen (Quelltext, Debug Symbole) das Debuggen entsprechend vereinfachen kann. GNU Debugger kann in folgende Kategorien eingeordnet werden:

1. Penetration Test 2.12
2. Dynamische Code-Analyse 2.5
3. Code Review 2.6

4.19 GitLab

Beschreibung

GitLab ist eine Onlineplattform, die es erlaubt Git-Repositries zu hosten und mehrere dabei mehrerer zusätzliche Dienste anbietet. Unter anderem ist es in GitLab möglich den Quellcode oder Commits zu durchsuchen, Pullrequests und Issues zu erstellen und zu verfolgen und Build- oder CI-Server zu integrieren.

Name	GitLab[33]
Autor	GitLab Inc.
Letzter Release	22.2.2019
Lizenz	kommerziell/MIT License
Betriebssysteme	Webapplikation
Hauptkategorie	Code Review 2.6

Kategorisierung

GitLab ist komplett quelloffen[34], der volle Funktionsumfang ist allerdings kostenpflichtig. GitLab wird in Ruby und Go entwickelt. Die neuste Version 11.8 wurde am 22.2.2019 veröffentlicht. Als Webapplikation ist Ruby für alle gängigen Betriebssysteme verfügbar. GitLab kann unter anderem als Codereviewtool kateogrisiert werden.

Ähnliche Prüfverfahren

- Crucible 4.12
- Collaborator 4.8

[33] https://about.gitlab.com/
[34] https://gitlab.com/gitlab-org

4.20 Gotcha

Beschreibung

Go Taint **Ch**eck **A**nalyser ist ein Tool um Taint Analysen auf in Go geschriebenen Programmen durchzuführen. Es basiert auf einem zuvor veröffentlichten Paper zur Informationsflussanalyse in GO [46]. Es wird eingesetzt um die Verarbeitung von Eingaben durch den Quellcode eines Programms zu verfolgen und so alle Parameter zu identifizieren auf die eine (Nutzer-) Eingabe Einfluss hat. Daraus können dann potentielle Bedrohungen für das Programm abgeleitet werden. Da es auf dem Quellcode arbeitet, handelt es sich um ein white box verfahren.

Name	Gotcha[35]
Autor	Anna-Katharina Wickert
Letzter Release	04.08.2017 (letzter Commit)
Lizenz	MIT
Betriebssysteme	Linux
Hauptkategorie	Taint Analysis 2.10

Kategorisierung

Gotcha wurde im Rahmen der Masterarbeit der Autorin entwickelt und wird scheinbar nicht mehr aktiv weiter geführt. Es ist für den Einsatz auf Linux ausgelegt und steht unter MIT Lizenz. Gotcha kann in folgende Kategorien eingeordnet werden:

1. Taint Analysis 2.10
2. Statische Code-Analyse 2.4

Ähnliche Prüfverfahren

- FlowDroid 4.17

4.21 Greenbone Security Manager

Beschreibung

Der Greenbone Security Manager ist der kommerzielle Verwundbarkeitsscanner von Greenbone Networks GmbH. Er ist auf den professionellen Einsatz für Office IT ausgerichtet. Ebenso wie bei OpenVAS

[35] https://github.com/akwick/gotcha

hängen die durchgeführten Tests von den Testdefinitionen, den sogegannten NVTs (Network Vulnerability Tests) ab. Obwohl der Greenbone Security Manager auf Office IT ausgerichtet ist, sind in dem NVT Feed auch Testdefinitionen speziell für Automatisierungskomponenten enthalten.

Name	Greenbone Security Manager[36]
Autor	Greenbone Networks GmbH
Letzter Release	30.04.2019
Lizenz	kommerziell
Betriebssysteme	Server auf Linux, Zugriff über Web
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Der Greenbone Security Manager[37] ist der kommerzielle Verwundbarkeitsscanner von Greenbone Networks GmbH. Er verwendet unter anderem den OpenVAS Scanner, bietet aber einige zusätzliche Funktionalität an. Seit dem Jahr 2019 werden die Releases für den Greenbone Security Manager in festen Zyklen veröffentlicht, immer zum 30. April und 31. Oktober. Der Greenbone Security Manager selbst wird unter einer kommerziellen Lizenz vertieben. Greenbone bietet jedoch außerdem eine Community Version an. Das Fraunhofer IOSB hat eine bisher unveröffentlichten Vergleich zwischen verschiedenen Versionen von OpenVAS (OpenVAS9, OpenVAS10 und dem Greenbone Security Manager) durchgeführt. Zusammengefasst hat sich dabei gezeigt, dass der Greenbone Security Manager die besseren Ergebnisse erzielte. Es ist möglich, den Greenbone Security Manager voll automatisiert zu betreiben. Der Greenbone Security Manager kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Konformitätstest 2.2

Ähnliche Prüfverfahren

- Achilles 4.1
- Nessus 4.30
- OpenVAS 4.34
- Qualys 4.36
- Nexpose 4.31
- SAINT
- RETINA

[36] https://www.greenbone.net/produktvergleich/
[37] https://www.greenbone.net/produktvergleich/

4.22 ISuTest

Beschreibung

ISuTest ist ein generisches Testing-Framework für industrielle Komponenten, das am Fraunhofer IOSB entwickelt wird. Es ist in der Lage verschiedenste Geräte zu testen. Über sogenannte Monitore kann das zu testende Gerät dabei überwacht werden. So kann über beliebige Schnittstellen überprüft werden ob das Gerät noch erwartungsgemäß funktioniert. Momentan fokussiert sich ISuTest auf das Fuzzing von verschiedenen Protokollen (zum Beispiel Profinet und Ethernet). Durch seine modulare Struktur können aber beliebige andere Prüfverfahren in das System integriert werden. So wurden beispielsweise bereits Verwundbarkeitsscanner für Webanwendungen in IsuTest integriert [47].

Name	ISuTest[38]
Autor	Fraunhofer IOSB
Letzter Release	30.03.2017
Lizenz	kommerziell
Betriebssysteme	Linux
Hauptkategorie	Fuzzing 2.11

Kategorisierung

Das Security Testing Framework für industrielle Komponenten ISuTest wird am Fraunhofer IOSB entwickelt [48, 10]. Der letzte Release hat am 30.03.2017 stattgefunden. Momentan befindet sich das Framework in einer Phase der grundlegenden Neustrukturierung. Das Lizenzmodell für ISuTest ist noch nicht abschließend geklärt. ISuTest ist in Python geschrieben und für Linux Systeme optimiert. Das Framework wurde gegen verschiedene reale Automatisierungskomponenten getestet und dadurch evaluiert [47]. ISuTest führt black box Untersuchungen aus wobei die automatisierte Durchführung von Tests ein wichtiger Bestandteil ist Durch den modulare Aufbau des Framework können nahezu beliebige back box Prüfverfahren in das System integriert werden. Zum aktuellen Zeitpunkt kann ISuTest in folgende Kategorien eingeteilt werden:

1. Fuzzing 2.11
2. Verwundbarkeitsscanner 2.1
3. Konformitätstest 2.2

Ähnliche Prüfverfahren

-

[38] https://www.iosb.fraunhofer.de/servlet/is/77291/

4.23 JasperGold

Beschreibung

JasperGold ist ein Tool, das bei Design, Testen und Verifikation von Hardware eingesetzt wird. Es bietet mehrere Apps die für unterschiedliche Aufgaben in Entwurf und Testphase von Hardware eingesetzt werden können, zum Beispiel für simulationsbasiertes Testen, Verifikation durch Model-Checking oder Linter/Checker für Hardware Design Sprachen. Laut [2] hat JasperGold den größten Funktionsumfang und das beste Nutzerinterface verglichen mit den anderen Tools dieser Kategorie.

Name	JasperGold[39]
Autor	Cadence
Letzter Release	8.5.2019
Lizenz	Commercial
Betriebssysteme	Linux
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

JasperGold wird von Cadence entwickelt und kommerziell vertrieben. Das ebenfalls für Hardwareentwurf genutzte Tool Incisive, welches auch von Cadence entwickelt wurde, ist seit 2015 von JasperGold abgelöst worden. Durch die Vielzahl an Apps die in JasperGold verwendet werden können ist eine eindeutige Kategorisierung schwierig. Hauptsächlich ist das Tool jedoch als formale Methode zu sehen. Es kann allerdings in alle der folgenden Kategorien eingeordnet werden:

1. Statische Code-Analyse 2.4
2. Robustheittests 2.14
3. Formale Methoden 2.3

Ähnliche Prüfverfahren

- Questa Formal 4.37
- Solidify 4.42

[39] https://www.cadence.com/content/cadence-www/global/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform.html

4.24 Java Pathfinder (JPF)

Beschreibung

Java Pathfinder (JPF) ist ein Analysetool für Java-Programme. Es erlaubt unter anderem das Finden von typischen Fehler wie Nullpointer-Exceptions oder unbehandelten Exceptions, aber auch komplexere Fehler wie Deadlocks oder fehlgeschlagene Assertions können erkannt werden. Außerdem können mit JPF Tests generiert und Codemetriken erstellt werden. JPF kann sowohl als Kommandozeilenprogramm als auch als Plugin für Eclipse oder Netbeans genutzt werden.

Name	Java Pathfinder[40]
Autor	NASA
Letzter Release	2.7.2018
Lizenz	Apache 2.0 License
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

JPF wurde ursprünglich von der NASA entwickelt und 2005 als open source Projekt veröffentlicht[41]. Es ist selbst in Java geschrieben und somit auf allen gängigen Betriebssystemen lauffähig. Der letzte Release war im Juli 2018, es gibt allerdings keine vorkompilierten ausführbaren Dateien sondern nur den Sourcecode der selbst kompiliert werden muss (Anleitungen auf der GitHub-Seite). JPF ist in erster Linie als formale Methode anzusehen kann aber je nach Nutzung in eine der folgenden Kategorien eingeordnet werden:

1. Statische Code-Analyse 2.4
2. Dynamische Code-Analyse 2.5
3. Formale Methoden 2.3

Ähnliche Prüfverfahren

- CBMC 4.6
- SPARK 4.43
- BLAST 4.4
- KeY 4.25
- Frama-C 4.16
- CPAchecker 4.11

[40] https://github.com/javapathfinder
[41] https://github.com/javapathfinder

4.25 KeY

Beschreibung

KeY ist ein Tool mit dem deduktive Beweise über Java-Programm in einem Sequenzenkalkül geführt werden können. Das Tool erwartet JML annotierten Java-Sourcecode als Eingabe. Dieser wird dann in eine Sequenz übersetzt auf der der Beweis über das Programm sowohl (teil-)automatisch als auch manuell geführt werden kann. Mit diesem Tool wurde unter anderem eine Verletzung einer Invarianten in der Timsort Implementierung der Java Standard Library gefunden [49] und die Dual-Pivit-Quicksort in Java wurde als korrekt bewiesen [50].

Name	KeY[42]
Autor	KIT, TU Darmstadt, Chalmers University
Letzter Release	11.10.2017
Lizenz	GNU GPL
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

KeY wird in Zusammenarbeit des Karlsruher Instituts für Technologie (KIT), der TU Darmstadt und der Chalmers University of Technology entwickelt. Es ist in Java geschrieben und sowohl Binaries als auch Quellcode stehen auf der Projektwebseite[43] zur Verfügung. Als Java Anwendung ist KeY für alle gängigen Betriebssysteme verfügbar. Der letzte offizielle Release war die Version 2.6.3 im Oktober 2017. KeY ist als formale Methode zu klassifizieren.

Ähnliche Prüfverfahren

- Java Pathfinder 4.24
- SPARK 4.43
- BLAST 4.4
- CBMC 4.6
- Frama-C 4.16
- CPAchecker 4.11

[42] https://www.key-project.org/
[43] https://www.key-project.org/download/

4.26 Klocwork

Beschreibung

Klocwork ist ein Tool für die Sicherheitsanalyse von Code (z.B. C, C++, Java, C#). Dabei werden sowohl Securtity- als auch Safetyprobleme aufgedeckt. Die gefundenen Schwachstellen können nach Relevanz sortiert werden. Klocwork lässt sich sowohl in CI-Server als auch in gängige IDEs wie IntelliJ, Eclipse oder Visual Studio integrieren. Klocwork wurde in [40] mit Coverity und anderen ähnlichen Tools verglichen. Positiv herausgestellt wurde dabei dabei vor allem die Möglichkeit, die gefundenen Defekte nach Relevanz zu sortieren, während negativ aufgefallen ist, dass nicht nach Schwachstellen von verwendeten Bibliotheken gesucht wurde.

Name	Klocwork[44]
Autor	RogueWave
Letzter Release	28.2.2018
Lizenz	kommerziell
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

Klocwork wird von RogueWave entwickelt. Es steht für alle gängigen Betriebssysteme sowie für mehrere IDEs als Plugin zur Verfügung. Die neuste Version Klocwork 2018 wurde am 28.2.2018 veröffentlicht. Klocwork kann als statisches Code-Analyse-Tool kategorisiert werden.

Ähnliche Prüfverfahren

- FACT 4.15
- Splint 4.45
- Checkstyle 4.7

4.27 libfiu

Beschreibung

libfiu ist eine Bibliothek für die Programmiersprache C (es existieren auch Python Bindings), mit der man gezielt Fehler in Programmen auslösen kann um deren Verhalten zu testen. Hierfür werden zusätzliche Funktionsaufrufe in den existierenden Code eingebracht, die bei normaler Ausführung keinen Effekt

[44] https://www.roguewave.com/products-services/klocwork

haben. Wird der Code allerdings in einer Testumgebung ausgeführt, können durch die zusätzlichen Funktionsaufrufe ansonsten schwer zu testende Zustände ausgelöst werden. So kann beispielsweise simuliert werden, dass kein Speicherplatz mehr zur Verfügung steht. libfiu ist darauf ausgelegt in existierenden Code eingebaut zu werden um die Testfälle derart zu erweitern, dass eine hohe Testabdeckung auch für selten ausgeführte Teile des Codes erreicht wird. Dies betrifft insbesondere Zweige, die für das Error-Handling zuständig sind, da diese meist schwer zu testen sind. Diejenigen Fehler die den Kontrollfluss auf eben diese Pfade leiten, treten selten auf und sind auch in Testumgebungen teilweise schwierig auszulösen.

Name	libfiu[45]
Autor	Alberto Bertogli et al.
Letzter Release	13.12.2018
Lizenz	BOLA (Public Domain)
Betriebssysteme	Linux, MacOS
Hauptkategorie	Fault Injection 2.13

Kategorisierung

libfiu wird im Wesentlichen von Alberto Bertogli alleine entwickelt und steht unter einer von ihm selbst entwickelten, scheinbar aber liberalen Lizenz (BOLA). Es ist für den Einsatz auf POSIX kompatiblen Betriebssystemen ausgelegt. libfiu kann in folgende Kategorien eingeordnet werden:

1. Fault Injection 2.13
2. Penetration Test 2.12
3. Robustheittests 2.14

Ähnliche Prüfverfahren

- Simmy 4.40
- W-SWFIT 4.53

4.28 MALPAS

Beschreibung

MALPAS ist ein Verifikationstool für verschiedene Sprachen. Es ist in der Lage, Programme, die in ein Zwischensprache übersetzt wurden, auf verschiedene Eigenschaften zu überprüfen (z.B. ungenutzter Code, uninitialisierte Variablen aber auch Konformität zu einer gegebenen Spezifikation). Alle Sprachen, die manuell oder automatisch in diese Zwischensprache übersetzt werden können, werden für

[45] https://github.com/albertito/libfiu

die Verifikation unterstützt. MALPAS wurde in mehreren sicherheitskritischen Softwareprojekten, unter anderem im nuklearen Bereich und in der Avionik, eingesetzt, um Software zu verifizieren [51][52].

Name	MALPAS[46]
Autor	Atkins
Letzter Release	?
Lizenz	kommerziell
Betriebssysteme	MS Windows 95/98/2000/XP/Vista/7/8
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

MALPAS wird von Atkins entwickelt und kommerziell vertrieben. Außer dem Tool selbst bietet Atkins auch Dienstleistung an die bei der Verifikation mit MALPAS helfen. Das Tool steht ausschließlich für Windows bis zur Version 8 hin zur Verfügung. MALPAS kann als formale Methode kategorisiert werden.

Ähnliche Prüfverfahren

- Java Pathfinder 4.24
- CBMC 4.6
- BLAST 4.4
- KeY 4.25
- Frama-C 4.16
- CPAchecker 4.11

4.29 Metasploit

Beschreibung

Metasploit ist ein umfangreiches Framework welches für die Informationsgewinnung, das gezieltes Testen auf Sicherheitslücken, das Ausnutzen von Sicherheitslücken, das Generieren von Payload und zum Kontrollieren eines übernommenen Systems eingesetzt werden kann. Es verfügt über eine umfangreiche Datenbank von Modulen, die eine Vielzahl von bekannten Sicherheitslücken abdecken. Nach manueller Auswahl der Module führt Metasploit Angriffe automatisch durch und prüft auch automatisch Systeme auf Schwachstellen. So kann im einfachsten Fall z.B. eine privilege Escalation durch den Befehl „getsystem“durchgeführt werden. Mit Metasploit können auch Payloads, wie z.B. Reverse Shells für verschiedene Programmiersprachen, in verschiedenen Codierungen und mit Schutz vor Antivirenprogrammen erstellt werden.

[46] http://malpas-global.com/

Name	Metasploit[47]
Autor	Rapid7
Letzter Release	10.01.2019 (core)
Lizenz	BSD
Betriebssysteme	Linux, Windows
Hauptkategorie	Penetration Test 2.12

Kategorisierung

Metasploit unterliegt der BSD Lizenz, allerdings vertreibt der Hersteller, Rapid7, auch eine Pro Version unter proprietärer Lizenz, welche weitere Funktionen, wie z.B. ein Webinterface bietet. Zusätzlich zu den Releases der Framework Software werden nahezu tägliche neue Module geschrieben, mit denen sich bekannte Schwachstellen automatisiert finden und ausnutzen lassen. Metasploit ist für den Einsatz unter Linux und Windows ausgelegt, es handelt sich um ein black box Tool. Metasploit kann in folgende Kategorien eingeordnet werden:

1. Penetration Test 2.12
2. Verwundbarkeitsscanner 2.1

4.30 Nessus

Beschreibung

Nessus ist ein pluginbasierter Verwundbarkeitsscanner für Schwachstellen verschiedenster Kategorien. Verschiedene ältere Plugins sind frei verfügbar, die meisten Plugins sind jedoch kostenpflichtig für die kommerzielle Nutzung. Insgesamt gibt es momentan mehr als 83000 Plugins, wobei nahezu täglich neue Plugins hinzukommen.

Name	Nessus[48]
Autor	Tenable Network Security
Letzter Release	Fast täglich neue Plugins
Lizenz	kommerziell
Betriebssysteme	Linux, Windows, Mac
Hauptkategorie	Verwundbarkeitsscanner 2.1

[47] https://www.metasploit.com/
[48] www.nessus.org

Kategorisierung

Der Verwundbarkeitsscanner Nessus[49] wird von Tenable Network Security vertrieben. Nahezu täglich werden neue Plugins für die Überprüfung auf konkrete Schwachstellen hinzugefügt. Der Verwundbarkeitsscanner wird kommerziell vertrieben und ist für gängige Betriebssysteme (Linux, Windows, Mac) verfügbar [53]. Nessus ist besonders darauf ausgelegt, Systeme automatisiert auf Schwachstellen zu überprüfen und verwendet dabei einen black box-Ansatz. Da Nessus ein weit verbreiteter Verwundbarkeitsscanner ist, wurde er bereits in verschiedenen Studien evaluiert [54, 55, 56]. Nessus kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1

Ähnliche Prüfverfahren

- OpenVAS 4.34
- Greenbone Security Manager 4.21
- Qualys 4.36
- Nexpose 4.31
- SAINT
- RETINA

4.31 Nexpose

Beschreibung

Nexpose ist ein Verwundbarkeitsscanner der außerdem Verwundbarkeitsmanagement anbietet. So können die Schwachstellen beispielsweise priorisiert werden und ihr Bearbeitungsstand dokumentiert werden. Nexpose ist als Cloud-Scanner konzipiert, die Benutzerinteraktion erfolgt also über eine Webschnittstelle. Sollen lokale IP-Adressen untersucht werden muss eine sogenannte On Premise Engine verwendet werden, welche auf Windows installiert wird. Der Fokus von Nexpose sind regelmäßige Scans ganzer Netzwerke. Ebenso kann der Verwundbarkeitsscanner aber auch gegen einzelne Geräte eingesetzt werden. Neben den Verwundbarkeitsscans kann Nexpose auch überprüfen ob ein Gerät verschiedenen Poicies entspricht. Erweitert werden kann Nexpose um verschiedene weitere Produkte von Rapid7, wie beispielsweise InsightVM (Verwundbarkeitsmanagement) oder InsightAppSec (Verwundbarkeitsscanner speziell für Webanwendungen). Soll damit eine Webanwendung untersucht werden, muss dieser zunächst ein Beweis hinzugefügt werden, dass man tatsächlich der Besitzer der Webanwendung ist (Meta-Tag).

[49] www.nessus.org

Name	Nexpose[50]
Autor	Rapid7
Letzter Release	01.05.2019
Lizenz	kommerziell
Betriebssysteme	Windows
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Nexpose[51] wird von Rapid7 kommerziell vertrieben. Der letzte Release wurde am 01.05.2019 veröffentlicht. Prinzipiell ist Nexpose als Cloud-Scanner konzipiert, darauf kann mit allen üblichen Browsern zugegriffen werden. Für lokale Scans ist eine Installation auf einem Windows System notwendig. Nexpose wurde auch in der Forschung bereits mit anderen Verwundbarkeitsscannern verglichen [8]. Außerdem wurde es bei internen Untersuchungen am Fraunhofer IOSB eingesetzt. Die Untersuchungen von Nexpose sind black box und können über eine API automatisiert durchgeführt werden. Nexpose kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Konformitätstest 2.2

Ähnliche Prüfverfahren

- Achilles 4.1
- Nessus 4.30
- Greenbone Security Manager 4.21
- Qualys 4.36
- OpenVAS 4.34
- SAINT
- RETINA

4.32 Nikto

Beschreibung

Nikto ist ein Verwundbarkeitsscanner der den Fokus auf Webserver legt. Er überprüft Webserver auf bekannte Schwachstellen indem er möglicherweise gefährliche Dateien sucht, die vorliegenen Softwareversionen untersucht und auch bestimmte Cross-Site-Scripting[52] Schwachstellen in bekannten

[50] https://www.rapid7.com/products/nexpose/
[51] https://www.rapid7.com/products/nexpose/
[52] https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

Programmen überprüft. Es ist möglich Nikto als Plugin für Nessus zu verwenden.

Name	Nikto[53]
Autor	Chris Sullo
Letzter Release	09.07.2015
Lizenz	GPL
Betriebssysteme	Linux, Windows
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Der Webserver-Verwundbarkeitsscanner Nikto[54] wird von Chris Sullo implementiert. Der letzte Release wurde am 09.07.2015 veröffentlicht, die letzte Aktivität hat jedoch am 15.04.2019 stattgefunden (Stand 03.05.2019). Nikto ist unter GPL lizenziert, der Quellcode ist auf github[55] zu finden. Der Quellcode ist in Perl geschrieben und kann unter Linux und Windows verwendet werden. Nikto wurde im Rahmen einer Studie am Fraunhofer IOSB auf die Performanz gegenüber vorsätzlich verwundbaren Webanwendungen und gegenüber realen Automatisierungskomponenten untersucht [47]. Der Verwundbarkeitsscanner ist auf black box Untersuchungen ausgerichtet und kann automatisiert betrieben werden. Nikto kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Penetration Test 2.12

Ähnliche Prüfverfahren

- Acunetix 4.2
- Vega 4.50
- Wapiti 4.51
- Skipfish 4.41
- ZAP 4.55

4.33 Nmap

Beschreibung

Nmap ist ein Portscanner, der mit umfangreichen Scripten zu einem mächtigen Werkzeug erweitert wurde. Er setzt je nach Parametrisierung TCP SYN Pakete, ganze TCP Verbindungsaufbauten, UDP

[53] https://cirt.net/Nikto2
[54] https://cirt.net/Nikto2
[55] https://github.com/sullo/nikto/

Requests oder ICMP ein. Weiterhin kann Nmap anhand verschiedener Charakteristika der Antworten raten, welches Betriebssystem auf dem gescannten Host läuft. Nmap liefert einige Scripte mit, mit denen auf bekannte Sicherheitslücken, wie z.B. Eternalblue geprüft werden kann. Da Nmap das zu scannende System nur „von außen“sieht, handelt es sich um ein black box Tool. Nmap kann bis zu einem gewissen Grad automatisiert werden, allerdings ist es empfehlenswert je nach Anforderung die einzusetzenden Techniken manuell auszuwählen und Nmap entsprechend zu parametrisieren. Nmap ist ein textbasiertes Programm wobei verschiedene grafische Oberflächen existieren, die Erstveröffentlichung erfolgte bereits 1997.

Name	Nmap[56]
Autor	Fyodor et al.
Letzter Release	20.03.2018
Lizenz	GPLv2 ähnlich
Betriebssysteme	Linux, MacOS, Windows
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Nmap wird hauptsächlich von Gordon Lyon (Pseudonym: Fyodor Vaskovich) entwickelt und steht unter einer auf der GPLv2 basierenden Lizenz, die die freie Nutzung für Endnutzer erlaubt. Zusätzlich wird eine kommerzielle Lizenz angeboten, die sich an Firmen richte die Nmap mit ihren eigenen Produkten weiter verbreiten wollen. Er ist für die gängigen Betriebssysteme verfügbar. Nmap kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Penetration Test 2.12
3. Robustheittests 2.14

4.34 OpenVAS

Beschreibung

OpenVAS ist ein Open Source Verwundbarkeitsscanner der auf Office-IT Umgebungen fokussiert ist. Die durchgeführten Tests hängen von den Testdefinitionen, den sogenannten NVTs (Network Vulnerability Tests) ab. Diese NVTs werden in einem ständig aktualisierten NVT Feed bereitgestellt. Neben dem kostenlosen OpenVAS existiert auch eine kostenpflichtige Version der Software, den Greenbone Security Manager. Dieser bietet unter anderem einen größeren Umfang an NVTs an, so beispielsweise auch Tests die speziell für Automatisierungskomponenten zugeschnitten sind.

[56] https://nmap.org/

Name	OpenVAS[57]
Autor	Greenbone Networks GmbH
Letzter Release	05.04.2019
Lizenz	GPL-v2.0
Betriebssysteme	Server auf Linux, Zugriff über Web
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Der freie Verwundbarkeitsscanner OpenVAS[58] wird von Greenbone entwickelt. Er ist ebenfalls eingebettet in die kostenpflichtige Software Greenbone Security Manager. Der letzte Realese des OpenVAS Scanners war am 05.04.2019, die letzte Aktivität hat am 02.05.2019 stattgefunden (Stand 03.05.2019). Auf github[59] ist der Quellcode des OpenVAS Scanners zu finden, der unter der Lizenz GPL-2.0 verfügbar ist. Der Quellcode ist in C geschrieben und kann so für alle gängigen Plattformen kompiliert werden. OpenVAS wurde in einer Studie am Fraunhofer IOSB auf die Performanz gegenüber vorsätzlich verwundbaren Webanwendungen und gegenüber realen Automatisierungskomponenten untersucht [47]. Außerdem wurde ein bisher unveröffentlichter Vergleich zwischen verschiedenen Versionen von OpenVAS (OpenVAS9, OpenVAS10 und dem Greenbone Security Manager) durchgeführt. Zusammengefasst hat sich dabei gezeigt, dass der Greenbone Security Manager die besseren Ergebnisse erzielte. OpenVAS kann über eine Schnittstelle voll automatisiert betrieben werden und ist auf black box Tests ausgerichtet. OpenVAS kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Konformitätstest 2.2

Ähnliche Prüfverfahren

- Achilles 4.1
- Nessus 4.30
- Greenbone Security Manager 4.21
- Qualys 4.36
- Nexpose 4.31
- SAINT
- RETINA

[57] http://www.openvas.org/
[58] http://www.openvas.org/
[59] https://github.com/greenbone/openvas-scanner

4.35 Peach

Beschreibung

Peach ist ein kommerzieller Fuzzer von dem auch eine Community Edition mit eingeschränkten Funktionalitäten existiert. Im Umfang von Peach sind bereits die Definitionen von einigen Netzwerkprotokollen enthalten, andere Protokolle können über eine XML-Datei spezifiziert und eingebunden werden. Ebenso können weitere, gerätespezifische Einstellungen über eine XML-Datei definiert werden. Peach erzeugt neue Eingaben über Generation und Mutation.

Name	Peach Fuzzer[60]
Autor	Peach Tech
Letzter Release	10.10.2016
Lizenz	kommerziell / Community (MIT)
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Fuzzing 2.11

Kategorisierung

Der kommerzielle Fuzzer Peach Fuzzer[61] wird von Peach Tech vertrieben. Der letzte Release wurde am 10.10.2016 veröffentlicht. Neben der kommerziellen Version existiert auch eine Community Version[62], welche einige Einschränkungen mit sich bringt. Für diese wurde der letzte Release am 03.04.2016 veröffentlicht. Die Community Edition steht unter der MIT Lizenz zur Verfügung. Beide Versionen können unter den gängigen Betriebssystemen Linux, Windows und MacOS verwendet werden. Peach wurde in einigen Studien in der Literatur evaluiert. [10, 57, 58] Die Tests werden black box ausgeführt und können Protokolldefinitionen im XML-Format verwenden. Peach kann in folgende Kategorien eingeordnet werden:

1. Fuzzing 2.11

Ähnliche Prüfverfahren

- Boofuzz 4.5

[60] https://www.peach.tech/products/peach-fuzzer/
[61] https://www.peach.tech/products/peach-fuzzer/
[62] https://sourceforge.net/projects/peachfuzz/

4.36 Qualys

Beschreibung

Die Qualys Cloud Plattform ist ein Verwundbarkeitsscanner der primär aus der Cloud betrieben wird. Für lokale IP Adressen oder offline Untersuchungen können aber auch lokale Instanzen eingesetzt werden. Neben den Verwundbarkeitsscans kann Qualys auch überprüfen ob ein Gerät verschiedenen Poicies entspricht. Die Untersuchungen können black box durchgeführt werden, können aber auch durch sogenannte Agents in eine white box durchgeführt werden. Neben einer kommerziellen Version ist auch eine Community Edition verfügbar.

Name	Qualys Cloud Plattform[63]
Autor	Qualys
Letzter Release	20.02.2019
Lizenz	kommerziell
Betriebssysteme	-
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Die Qualys Cloud Plattform[64] wird von Qualys kommerziell vertrieben. Der letzte Release wurde am 20.02.2019 veröffentlicht. Neben der kommerziellen Version ist auch eine Community Version verfügbar, welche jedoch nicht quelloffen zur Verfügung steht. Da die Qualys Cloud Plattform cloudbasiert agiert, kann eine Interaktion mit den üblichen Browsern erfolgen. Lokale Scans können durch lokale Scanner durchgeführt werden, welche als vollständige virtuelle Maschinen angeboten werden. Auch hier ist demnach kein spezielles Betriebssystem notwendig. Die Qualys Cloud Plattform wurde unter ihrem früheren Namen (Qualys Guard) in der Literatur untersucht [22]. Außerdem wurde es bei internen Untersuchungen am Fraunhofer IOSB eingesetzt. Andere Produkte von Qualys wurden ebenfalls in Untersuchungen einbezogen [8, 24, 28]. Die Untersuchungen werden black box durchgeführt, können aber auch durch sogenannte Agents in eine white box durchgeführt werden. Durch eine API ist es möglich die Scans automatisiert zu konfigurieren und durchzuführen.

Ähnliche Prüfverfahren

- Achilles 4.1
- Nessus 4.30
- Greenbone Security Manager 4.21
- Nexpose 4.31
- OpenVAS 4.34

[63] https://www.qualys.com/cloud-platform/
[64] https://www.qualys.com/cloud-platform/

- SAINT
- RETINA

4.37 Questa Formal

Beschreibung

Questa Formal ist eines von mehreren Prüfverfahren, die Mentor Graphics zum Testen, Simulieren und Verifizieren von Hardwareentwürfen anbietet. Questa Formal spezialisiert sich dabei auf die Verifikation von Designs, die in register transfer language (RTL) spezifiziert wurden. Die grundlegende Technik hierbei ist Model Checking. Auf diese Art kann die Korrektheit des Entwurfs bewiesen werden (im Gegensatz zu Tests oder Simulationen). Laut [2] hat Questa Formal eine gut zu bedienende Benutzeroberfläche, kann allerdings nicht alle Features vergleichbarer Prüfverfahren bieten.

Name	Questa Formal Verification[65]
Autor	Mentor Graphics
Letzter Release	?
Lizenz	kommerziell
Betriebssysteme	?
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

Questa Formal Verification wird von Mentor Graphics entwickelt. Das Tool kann als formale Methode klassifiziert werden.

Ähnliche Prüfverfahren

- JasperGold 4.23
- Solidify 4.42

4.38 Rodin/Event-B

Beschreibung

Rodin ist ein Tool welches es erlaubt Systeme in der Sprache Event-B zu entwerfen und den Entwurf zu verifizieren. Event-B ist eine auf Mengentheorie basierte Sprache, die es erlaubt System zunächst sehr

[65] https://www.mentor.com/products/fv/questa-formal/

abstrakt und dann immer detaillierter zu spezifizieren. Mit Rodin wird dann gezeigt, dass alle Versionen (verschiedene Abstraktionslevel) konsistent sind.

Name	Rodin/Event-B[66]
Autor	University of Southhampton
Letzter Release	21.2.2018
Lizenz	open source
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

Rodin ist ein quelloffenes Tool welches von der University of Southhampton entwickelt wird. Die neuste Version 3.4 ist am 21.2.2018 erschienen. Es existieren Versionen für Linux, MacOS und Windows. Rodin ist als formale Methode zu klassifizieren.

Ähnliche Prüfverfahren

- MALPAS 4.28
- CBMC 4.6

4.39 SDMetrics

Beschreibung

SDMetrics ist eine Tool um UML-Designs auf Fehler zu überprüfen. Dafür stellt das Tool sowohl Metriken zur Verfügung um die Qualität des Designs quantitativ feststellen zu können, als auch Regeln die bestimmte Probleme im Design auf decken. Beispiele für solche Regeln sind *eine abstrakte Klasse darf keine nicht abstrakte Elternklasse haben*, *ein Variablenname ist ein Java oder C-Keyword* oder *ein Usecase beinhaltet sich selbst*. Sowohl Metriken als auch Regeln können auch manuell hinzugefügt werden. Das Programm lädt das UML-Design als XML-Metadata-Interchange-Datei (XMI-Datei). Auf diese Weise ist es kompatibel mit allen Programmen die dieses Format unterstützen.

[66] http://www.event-b.org/

Name	SCMetrics[67]
Autor	Jürgen Wüst
Letzter Release	2.1.2018
Lizenz	kommerziell
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

SDMetrics wird von Jürgen Wüst entwickelt. Der die neuste Version 2.35 wurde am 2.1.2018 veröffentlicht. Der Kern des Programms (ohne GUI und die Regeln bzw. Metriken) ist quelloffen. Die gesamte Anwendung wird direkt auf der Webseite vertrieben (kostenlose Lizenzen für akademischen Gebrauch sind möglich). SDMetrics ist in Java geschrieben und somit auf jedem javafähigen Betriebssystem lauffähig.

Ähnliche Prüfverfahren

- Checkstyle 4.7
- Splint 4.45

4.40 Simmy

Beschreibung

Simmy ist ein Fault-Injection Werkzeug für C# / .NET, welches mit dem Polly[68] Framework arbeitet. Es wird dafür genutzt um zufällige Fehler auf Ebene des Quellcodes in Projekte einzuschleusen. Hierfür gibt es die Möglichkeit, Exceptions (Faults), Latenzen oder anderes Verhalten zufallsbasiert an bestimmten Stellen in den Code einzufügen.

Name	Simmy[69]
Autor	Dylan Reisenberger et al.
Letzter Release	10.05.2019 (letzter Commit)
Lizenz	BSD
Betriebssysteme	Windows (für C#)
Hauptkategorie	Fault Injection 2.13

[67] https://www.sdmetrics.com/
[68] https://github.com/App-vNext/Polly
[69] https://github.com/Polly-Contrib/Simmy

Kategorisierung

Es handelt sich um ein Projekt von zwei Hauptautoren, welches stark an das Polly Framework angebunden ist. Ein Release ist angekündigt, die Deadline wurde allerdings schon um zwei Wochen überschritten, ob der verfügbare Quellcode bereits lauffähig ist, ist unklar. Nach manuellem Einfügen der Fehler in den Quellcode des zu testenden Programms arbeitet Simmy automatisch. Da der Quellcode verfügbar sein muss, handelt es sich um ein white box Verfahren. Simmy kann in folgende Kategorien eingeordnet werden:

1. Fault Injection 2.13

Ähnliche Prüfverfahren

- libfiu 4.27
- CORDS 4.9
- W-SWIFT 4.53

4.41 Skipfish

Beschreibung

Skipfish ist ein Verwundbarkeitsscanner für Webserver der sich zum Ziel gesetzt hat, häufig auftretende Probleme bei anderen Verwundbarkeitsscanner zu adressieren. Aus diesem Grund wird ein besonderer Augenmerk auf die Performanz und die Bedienbarkeit gelegt. Skipfish untersucht einen Webserver auf verschiedenste Schwachstellen, insbesondere auch auf Schwachstellen der Webanwendung an sich. Die gefundenen Schwachstellen werden nach Durchführung der Untersuchung nochmals automatisiert überprüft um Dopplungen zu entfernen und die Verständlichkeit der Ergebnisse zu erhöhen.

Name	Skipfish[70]
Autor	M. Zalewski, N. Heinen und S. Roschke (Google)
Letzter Release	22.12.2012
Lizenz	Apache-2.0
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Verwundbarkeitsscanner 2.1

[70] https://github.com/spinkham/skipfish

Kategorisierung

Der Verwundbarkeitsscanner für Webserver Skipfish[71] wurde von Google entwickelt und steht unter der freien Lizenz Apache-2.0 zur Verfügung. Am 22.12.2012 wurde das letzte Mal ein Update für Skipfish zur Verfügung gestellt. Die Verwendung von Skipfish ist auf allen gängigen Betriebssystemen möglich. Skipfish wurde in verschiedenen Vergleichsstudien betrachtet [24, 42, 59, 60]. Insbesondere wurde Skipfish in einer Studie am Fraunhofer IOSB auf die Performanz gegenüber vorsätzlich verwundbaren Webanwendungen und gegenüber realen Automatisierungskomponenten untersucht [47]. Da Skipfish über die Kommandozeile gesteuert wird, ist es insbesondere möglich die Steuerung der black box Tests zu automatisieren. Skipfish kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1

Ähnliche Prüfverfahren

- Acunetix 4.2
- Vega 4.50
- Wapiti 4.51
- Nikto 4.32
- ZAP 4.55

4.42 Solidify

Beschreibung

Solidify bietet funktionale Verifikation von Hardwaredesigns welche in der register transfer language (RTL) beschrieben sind. Durch die vollständige Abdeckung aller erreichbaren Zustände kann ein Design so gegen eine Spezifikation (bspw. in SVA, PSL oder OVL) verifiziert werden. Fehler die unter anderem gefunden werden sind Deadlocks, Array-Out-Of-Bounds-Fehler und unerreichbarer Code. Das Tool besitzt eine grafische Benutzeroberfläche.

Name	Solidify[72]
Autor	Averant
Letzter Release	12.12.2017
Lizenz	kommerziell
Betriebssysteme	Win NT, Win 2000, Sun Solaris, Linux, HP-UX
Hauptkategorie	Formale Methoden 2.3

[71] https://github.com/spinkham/skipfish
[72] http://www.averant.com/products-solidify.html

Kategorisierung

Solidify wird von Averant entwickelt. Die neuste Version 6.5 ist am 12.12.2017 erschienen. Es steht für mehrere gängigen Betriebssysteme zur Verfügung. Als Verifikationstool ist Solidify als formale Methode zu klassifizieren.

Ähnliche Prüfverfahren

- JasperGold 4.23
- Questa Formal 4.37

4.43 SPARK

Beschreibung

SPARK ist eine Kombination aus Programmiersprache und Verifikationstool, die es zum Ziel hat, besonders geeignet für die Entwicklung sicherheitsrelevanter Software zu sein. Die Programmiersprache baut auf Ada auf, wobei Standard Ada oder C Code wiederverwendet werden kann. Die Verifikationstools erlauben Beweise von Standardeigenschaften bis hin zu Beweisen von formalen Spezifikationen. In diesen formalen Spezifikationen können unter anderem auch Informationflow-Eigenschaften spezifiziert werden. Spark wurde für mehrere Projekt in industriellem Umfeld eingesetzt [61].

Name	SPARK[73]
Autor	AdaCore
Letzter Release	2.2019
Lizenz	kommerziell
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Formale Methoden 2.3

Kategorisierung

SPARK wird von AdaCore entwickelt und vertrieben. SPARK Version 19 wurde im Februar 2019 released. Es ist für alle gängigen Betriebssysteme verfügbar. SPARK kann sowohl als Programiersprache als auch als Toolsuite für dynamische und statische Code-Analyse sowie für Testen aber auch als formale Methode kategorisiert werden. Zusammenfassend kann SPARK also in folgende Kategorien eingeordnet werden:

1. Statische Code-Analyse 2.4
2. Dynamische Code-Analyse 2.5

[73] https://www.adacore.com/sparkpro

3. Formale Methoden 2.3

Ähnliche Prüfverfahren

- Java Pathfinder 4.24
- CBMC 4.6
- BLAST 4.4
- KeY 4.25
- Frama-C 4.16
- CPAchecker 4.11

4.44 Spin

Beschreibung

Spin ist ein Prüfverfahren um Modelle auf bestimmte Vorgaben zu überprüfen. Dabei lassen sich auch Sicherheitseigenschaften als Vorgaben spezifizieren gegen die ein Modell überprüft werden soll.

Name	Spin Model Checker[74]
Autor	Bell Labs
Letzter Release	17.06.2018
Lizenz	BSD 3-Clause
Betriebssysteme	Linux
Kategorien	Model checking 2.9

Kategorisierung

Der Spin Model Checker wurde bereits 1980 in den Bell Labs entwickelt. Seit dem wird er ständig weiterentwickelt und ist seit 2016 unter dem BSD 3-Clause Open Source Lizenz verfügbar. Der letzte Release wurde am 17.06.2018 veröffentlicht (Stand 08.05.2019). Spin kann unter Linux verwendet werden und ist in den offiziellen Paketquellen für Debian enthalten. Obwohl Spin schon lange existiert gibt es kaum Literatur, die Spin für die Überprüfung von Sicherheitseigenschaften behandelt. In einer Veröffentlichung wurde es jedoch für die Überprüfung eines Routing-Protkolls eingesetzt [62]. Die Erstellung der Modelle und Vorgaben muss manuell erfolgen, die Überprüfung kann automatisiert geschehen. Spin ist in die white box Verfahren einzuordnen. Spin kann in folgende Kategorien eingeordnet werden:

1. Formale Methoden 2.3
2. Model checking 2.9

[74] http://spinroot.com

Ähnliche Prüfverfahren

- Alloy 4.3
- Uppaal 4.48

4.45 Splint

Beschreibung

Splint ist eine inoffizielle Weiterentwicklung von Lint, dem vielleicht ersten statischen Codechecker der ursprünglich für C geschrieben wurde und von dem bis heute das Wort linten/linter geprägt wurde. Das Tool ist in der Lage, typische Programmierfehler zur Compilezeit zu entdecken und darauf hinzuweisen. Klassische Beispiele für solche Fehler sind Zuweisungen statt Vergleiche, fehlende breaks in switch-Anweisungen, die Nutzung von uninitialisierten Variablen oder implizite Typumwandlungen.

Name	Splint[75]
Autor	University of Virginia
Letzter Release	12.7.2007
Lizenz	GNU GPL
Betriebssysteme	Linux, Windows
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

Splint ist quelloffen und wird auf GitHub[76] zum Download zur Verfügung gestellt. Es wird von der University of Virginia entwickelt und die aktuellste Version 3.1.2 wurde am 12.7.2007 veröffentlicht. Es ist hauptsächlich für Linux Systeme verfügbar, ein Windowsinstaller existiert aber auch. Splint ist ein klassisches statisches Code-Analyse-Tool.

Ähnliche Prüfverfahren

- Checkstyle 4.7
- SDMetrics 4.39
- SpotBugs 4.46

[75] http://splint.org/
[76] https://github.com/ravenexp/splint

4.46 SpotBugs

Beschreibung

SpotBugs ist der inoffizielle Nachfolger von FindBugs, einem Tool zur Erkennung typischer Programmierfehler/Schwachstellen in Javacode. Das Tool kann sowohl als Kommandozeilentool, in einer eigenen GUI oder als Plugin für Eclipse, Ant, Maven oder Gradle verwendet werden. Es sucht nach typischen Antipatterns wie z.B. das fehlende Schließen eines Streams oder das Vergleichen von Strings mit == anstatt mit equals. Es existiert ein Plugin für SpotBugs (FindSecBugs), welches sich auf die Warnung vor Securityfehlern spezialisiert hat.

Name	SpotBugs[77]
Autor	communitybasiert
Letzter Release	1.3.2019
Lizenz	GNU LGPL
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

SpotBugs wird als Communityprojekt in Java entwickelt. Es steht auf GitHub[78] quelloffen zur Verfügung. Die neuste Version 3.1.12 wurde am 1.3.2019 veröffentlicht. SpotBugs ist für alle gängigen Betriebssysteme verfügbar. Es ist als statisches Code-Analyse-Tool zu kategorisieren.

Ähnliche Prüfverfahren

- FACT 4.15
- Splint 4.45
- Checkstyle 4.7

4.47 Unit Tests

Beschreibung

Als Unit Tests bezeichnet man die funktionale Prüfung einzelner Softwaremodule. Dies geschieht durch Definition von Testfällen, dann auf die einzelnen Module angewandt werden und so im Idealfall das

[77] https://spotbugs.github.io/
[78] https://github.com/spotbugs/spotbugs

komplette Verhalten des Moduls prüfen. Unit Tests führen hierzu da Modul, bzw. einzelne Funktionen daraus mit Testeingaben aus und prüfen ob die Ausgabe der Erwartung entspricht.

Name	Unit Tests
Autor	n/a
Letzter Release	n/a
Lizenz	n/a
Betriebssysteme	alle
Hauptkategorie	Dynamische Code-Analyse 2.5

Kategorisierung

Unit Tests sind kein Tool sondern eine Methodik die von vielen Tools, wie z.B. Entwicklungsumgebungen, bereitgestellt wird. Da der Quellcode bekannt ist, zählen Unit Tests zu den white box Verfahren.

Unit Tests können in folgende Kategorien eingeordnet werden:

1. Dynamische Code-Analyse 2.5

4.48 Uppaal

Beschreibung

Uppaal überprüft ebenso wie Spin ein Modell darauf ob es bestimmten Spezifikationen entspricht. Der Unterschied liegt jedoch darin, dass Uppaal auch die Zeit abbildet und in das Modell aufnehmen kann.

Name	Uppaal[79]
Autor	Uppsala University, Aarlborg University
Letzter Release	20.05.2014
Lizenz	kommerziell
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Model checking 2.9

Kategorisierung

Uppaal[80] wurde von dem Department of Information Technology der Uppsala University und dem Department of Computer Science der Aalborg University entwickelt. Für akademische Nutzung steht das

[79] http://www.uppaal.org/
[80] http://www.uppaal.org/

Programm kostenlos zur Verfügung, in anderen Fällen muss eine kommerzielle Lizenz erworben werden. Der letzte Release wurde am 25.05.2014 veröffentlicht. Das Programm ist für Linux, Windows und MacOS verfügbar. Die Erstellung der Modelle und Vorgaben muss manuell erfolgen, die Überprüfung kann automatisiert geschehen. Dies ist in die white box Verfahren einzuordnen. Upaal wird in der Literatur anhand verschiedener Szenarien evaluiert [63, 64, 65]. Uppaal kann in folgende Kategorien eingeordnet werden:

1. Formale Methoden 2.3
2. Model checking 2.9

Ähnliche Prüfverfahren

- Alloy 4.3
- Spin 4.44

4.49 Valgrind

Beschreibung

Valgrind ist eine Werkzeugsammlung zum Auffinden von Fehlern in der Speicherverwaltung und zur Laufzeitanalyse. Es nutzt eine virtuelle Maschine, in der speziell kompilierter, plattform unabhängiger Bytecode ausgeführt wird. Mittels verschiedener Werkzeuge kann Valgrind Laufzeitfehler wie z.B. Speicherüberläufe, Speicherlecks und User-After-Free finden. Je nach Werkzeug werden aber auch falschpositive Meldungen generiert.

Name	Valgrind[81]
Autor	Valgrind Development Team
Letzter Release	12.04.2019
Lizenz	GPL
Betriebssysteme	Linux, MacOS, Android
Hauptkategorie	Dynamische Code-Analyse 2.5

Kategorisierung

Valgrind wurde initial Julian Seward entwickelt [66] und wird nun von einer Community weiter geführt. Die aktuelle Version, 3.15.0, ist vom 12 April 2019, das Projekt wird aktiv weiterentwickelt. Linux, macOS und Android werden direkt unterstützt, es existieren aber auch experimentelle Ports für BSD Derivate. Obwohl Valgrind auf bytecode ausgeführt wird, ist es als white box Tool zu klassifizieren, da eben dieser bytecode zunächst aus dem Quelltext kompiliert werden muss.

[81] http://www.valgrind.org/

Valgrind kann in folgende Kategorien eingeordnet werden:

1. Dynamische Code-Analyse 2.5

4.50 Vega

Beschreibung

Vega ist ein Verwundbarkeitsscanner für Webserver. Zusätzlich zu den klassischen Fähigkeiten eines Verwundbarkeitsscanners bietet Vega einen Proxy an, mit dessen Hilfe Netzwerkpakete betrachtet und manuell untersucht werden können. Durch den Proxy ist es also möglich, automatisierte, hybride oder manuelle Tests durchzuführen. Bedient wird Vega über eine grafische Benutzeroberfläche.

Name	Vega[82]
Autor	Subgraph
Letzter Release	03.07.2011
Lizenz	MIT license
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Der Verwundbarkeitsscanner für Webserver Vega[83] wurde von Subgraph entwickelt. Der letzte offizielle Release fand am 03.07.2011 statt, das github Repository[84] wurde das letzte Mal am 29.06.2016 aktualisiert. Lizenziert ist Vega unter der MIT Lizenz. Der Quellcode ist in Java geschrieben. Vega kann nur bedingt automatiesiert verwendet werden, da der Code eine Verwendung über eine grafische Benutzeroberfläche vorsieht. Da das Projekt jedoch OpenSource vorliegt, kann durchaus eine Anbindung an eine Kommandozeile oder an eine andere Schnittstelle erfolgen. Am Fraunhofer IOSB wurde eine prototypische Implementierung gegen eine Python Schnittstelle zu ISuTest entwickelt. Der Verwundbarkeitsscanner Vega kann unter allen gängigen Betriebssystemen verwendet werden. Vega wurde in verschiedenen Vergleichsstudien betrachtet [67, 26]. Insbesondere wurde Vega in einer Studie am Fraunhofer IOSB auf die Performanz gegenüber vorsätzlich verwundbaren Webanwendungen und gegenüber realen Automatisierungskomponenten überprüft [47]. Die Untersuchungen von Vega werden nach einem black box Ansatz durchgeführt. Vega kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1

[82] https://subgraph.com/vega/
[83] https://subgraph.com/vega/
[84] https://github.com/subgraph/Vega

Ähnliche Prüfverfahren

- Acunetix 4.2
- Skipfish 4.41
- Wapiti 4.51
- Nikto 4.32
- ZAP 4.55

4.51 Wapiti

Beschreibung

Wapiti ist ein Verwundbarkeitsscanner für Webanwendungen. Bei den Untersuchungen wird ein Fokus darauf gelegt, zunächst alle möglichen Uniform Resource Locators (URLs) und die Eingaben darauf zu finden und diese anschließend mit zufälligen Werten zu testen. Dabei verhält sich Wapiti wie ein Fuzzer.

Name	Wapiti[85]
Autor	Nicolas Surribas
Letzter Release	11.05.2018
Lizenz	GNU GPL v2
Betriebssysteme	Linux, Windows, MacOS
Hauptkategorie	Verwundbarkeitsscanner 2.1

Kategorisierung

Wapiti[86] ist ein quelloffener Verwundbarkeitsscanner für Webanwendungen von Nicolas Surribas. Der letzte Release wurde am 11.05.2018 veröffentlicht. Der Quellcode von Wapiti ist in Python geschrieben und unter der GNU GPL v2 Lizenz verfügbar. So kann der Code auf jedem Betriebssystem ausgeführt werden das Python unterstützt, insbesondere also Linux, Windows und MacOS. Wapiti wurde in verschiedenen Vergleichsstudien betrachtet [59, 26]. Insbesondere wurde Wapiti in einer Studie am Fraunhofer IOSB auf die Performanz gegenüber vorsätzlich verwundbaren Webanwendungen und gegenüber realen Automatisierungskomponenten untersucht [47]. Da Wapiti über die Kommandozeile gesteuert wird, ist es insbesondere möglich die Steuerung der black box Tests zu automatisieren. Wapiti kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Fuzzing 2.11

[85] http://wapiti.sourceforge.net/
[86] http://wapiti.sourceforge.net/

Ähnliche Prüfverfahren

- Acunetix 4.2
- Vega 4.50
- Skipfish 4.41
- Nikto 4.32
- ZAP 4.55

4.52 Wireshark

Beschreibung

Wireshark ist ein Werkzeug um Datenpakete an verschiedenen Netzwerkinterfaces abzugreifen und für den Nutzer verständlich darzustellen. Hierfür ist es mit umfangreichen Darstellungsoptionen für alle gängigen Protokolle ausgestattet. Es bietet umfangreiche Filtermöglichkeiten um die anzuzeigenden Pakete vorzufiltern und somit auch große Mengen von Paketen handlen zu können. Wireshark kann außerdem aufgezeichnete Pakete darstellen.

Name	Wireshark[87]
Autor	Community
Letzter Release	21.05.2019
Lizenz	GPLv2
Betriebssysteme	Windows, Linux, macOS
Hauptkategorie	Penetration Test 2.12

Kategorisierung

Wireshark wurde 1998 von Gerald Combs unter dem Namen „Ethereal“ins Leben gerufen und leitet auch weiterhin die Entwicklung. Zusätzlich werden rund 600 weitere Entwickler auf der Webseite gelistet, die Beiträge geliefert haben. Wireshark wird aktiv entwickelt, der letzte Release erfolgte am 22.05.2019, Wireshark steht unter GPLv2. Obwohl es viele Abläufe automatisiert, ist Wireshark ein Werkzeug um die manuelle Auswertung von Datenpaketen zu unterstützen. Verschiedene Netzwerksniffer wurden in [68] verglichen. Wireshark kann in folgende Kategorien eingeordnet werden:

1. Penetration Test 2.12

[87] https://www.wireshark.org/

4.53 W-SWFIT

Beschreibung

W-SWFIT implementiert die in [69] vorgestellte G-SWFIT fault injection Technik für Windows. Diese Technik kann dafür eingesetzt werden um verschiedene Arten von Fehlern in Software zu emulieren.

Name	W-SWFIT[88]
Autor	Paul Jordan
Letzter Release	02.12.2017 (letzter Commit)
Lizenz	MIT
Betriebssysteme	Windows
Hauptkategorie	Fault Injection 2.13

Kategorisierung

Es ist auf den Einsatz in Windows x64 ausgelegt, sollte aber auch auf mit 32 bit Software funktionieren. Der letzte Commit ist vom 2.12.2017, es ist daher davon auszugehen, dass das Projekt nicht weiterentwickelt wird. W-SWFIT kann in folgende Kategorien eingeordnet werden:

1. Fault Injection 2.13

Ähnliche Prüfverfahren

- libfiu 4.27
- CORDS 4.9
- Simmy 4.40

4.54 Xaniziter

Beschreibung

Xaniziter ist ein Tool zur Aufdeckung von Securityschwachstellen in Webapplikationen geschrieben in Java oder Scala. Es meldet typische Anfälligkeiten wie: Schwachstellen in Bibliotheken, SQL Injection, Fehlkonfigurationen oder hardgecodete Credentials. Dabei untersucht es neben dem Quellcode auch Konfigurationsdateien. Xanitizer hat eine eigene graphische Benutzeroberfläche, es besteht aber auch die Möglichkeit, es in CI-Server einzubinden.

[88] https://github.com/paullj1/W-SWFIT

Name	Xanitizer[89]
Autor	RIGS IT
Letzter Release	2.4.2019
Lizenz	kommerziell
Betriebssysteme	Windows, Linux
Hauptkategorie	Statische Code-Analyse 2.4

Kategorisierung

Xanitizer wird von RIGS IT entwickelt. Der die neuste Version 4.2.3 wurde am 2.4.2019 veröffentlicht. Das Tool ist sowohl für Linux als auch für Windows Systeme verfügbar. Xanitizer ist als statisches Code-Analyse-Tool zu kategorisieren.

Ähnliche Prüfverfahren

- FACT 4.15
- Splint 4.45
- Checkstyle 4.7

4.55 ZAP

Beschreibung

Der OWASP Zed Attack Proxy (ZAP) ist ein nicht-transparenter Proxy. Nicht-transparent bedeutet hier, dass ZAP als Proxy im Browser des Nutzers eingestellt werden muss. ZAP bietet einem Nutzer dann die Möglichkeit HTTP und HTTPS Pakete feingranular zu betrachten und zu manipulieren. Zusätzlich zu der Proxy-Funktionalität bietet ZAP verschiedene Funktionalitäten für einen Verwundbarkeitsscan an. Durch diese Verbindung von Proxy und Verwundbarkeitsscanner bietet sich ZAP besonders für die Untersuchung von speziell entwickelten bzw. benutzerdefinierten Webservern an.

Name	ZAP[90]
Autor	OWASP[91]
Letzter Release	Wöchentliche Releases
Lizenz	Apache-2.0
Betriebssysteme	Windows, Linux, MacOS
Hauptkategorie	Penetration Test 2.12

[89] https://www.rigs-it.com/xanitizer/
[90] https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project
[91] https://www.owasp.org

Kategorisierung

Der OWASP Zed Attack Proxy wird von dem Open Web Application Security Project (OWASP) entwickelt. OWASP[92] ist eine internationale Organisation die sich für die Sicherheit von Webanwendungen einsetzt. Hierzu bietet sie unter anderem eine große Informationsbasis für Schwachstellen, Angriffe, Gegenmaßnahmen und weitere verwandte Themen an. Außerdem stellt sie verschiedene Tools für diverse Problemstellungen zur Verfügung. Von ZAP werden wöchentlich Releases veröffentlicht, der letzte große Release hat am 28.11.2017 stattgefunden (Stand 08.056.2019). Der Code von ZAP ist in Java geschrieben und steht unter der Apache-2.0 Lizenz auf github[93] zur Verfügung. Er kann auf allen gängigen Plattformen ausgeführt bzw. kompiliert werden. ZAP ist ein weit verbreiteter Proxy und Verwundbarkeitsscanner und wurde in diversen Studien untersucht [24, 60, 59, 25, 67, 27, 26]. Insbesondere wurde ZAP in einer Studie am Fraunhofer IOSB auf die Performanz gegenüber vorsätzlich verwundbaren Webanwendungen und gegenüber realen Automatisierungskomponenten untersucht [47]. Die automatischen Verwundbarkeitsscans von ZAP können durch eine API vollautomatisiert durchgeführt werden. Dabei führt ZAP black box Tests durch. ZAP kann in folgende Kategorien eingeordnet werden:

1. Verwundbarkeitsscanner 2.1
2. Penetration Test 2.12

Ähnliche Prüfverfahren

- Acunetix 4.2
- Vega 4.50
- Wapiti 4.51
- Skipfish 4.41
- Nikto 4.32

5 Taxonomie

Um die Kategorien der Tools besser einordnen zu können stellen wir in diesem Abschnitt eine Taxonomie derselben vor, die die Kategorien anhand verschiedener markanter Merkmale kategorisiert. Zu diesem Zweck ordnen wir die Kategorien anhand von sechs verschiedenen Eigenschaften:

Automatisierbarkeit Die vorgestellten Methoden unterscheiden sich stark in ihrer benötigten Nutzerinteraktion. Wir unterscheiden hier zwischen Kategorien deren Tools meist vollautomatisch funktionieren (also nahezu ohne Nutzerinteraktion auskommen), teilautomatisierten Lösungen (hier ist es beispielsweise denkbar, dass eine Spezifikation vom Nutzer geschrieben werden muss, dass Tool diese aber dann automatisch überprüft) und Kategorien in denen das Tool hauptsächlich als Unterstützung für eine manuelle Tätigkeit genutzt wird.

[92] https://www.owasp.org
[93] https://github.com/zaproxy/zaproxy

Positives/Negatives Verfahren Eine grundsätzliche Unterscheidung kann getroffen werden, je nachdem ob ein Verfahren darauf ausgelegt ist Fehler zu finden, oder die Korrektheit des zu untersuchenden Systems zeigen soll. Klassische Testverfahren sind dabei immer „nur “auf Fehlersuche aus wobei formale Methoden meist die Korrektheit eines Systems beweisen.

dynamisch/statische Verfahren In diesem Dokument bezeichnen wir Verfahren als statisch wenn sie während oder vor Compilezeit ausgeführt werden und alle anderen Verfahren als dynamisch. Statische Verfahren sind somit unabhängig von Deployment und Laufzeitumgebung während dynamische Verfahren oftmals gerade unter Berücksichtigung dieser Aspekte testen wollen.

Black box/White box Verfahren Als black box Verfahren bezeichnen wir jedes Verfahren, welches ohne den Code des zu untersuchenden Systems angewendet werden kann. Im Gegensatz dazu ist für white box Verfahren der Quellcode Voraussetzung für die Anwendung. Während black box Verfahren den Vorteil bieten weniger Voraussetzungen zu haben können white box Verfahren oftmals mehr Fehler finden.

Getestete Eigenschaften Wir unterscheiden zwischen zwei Arten von Eigenschaften auf die getestet werden kann: Standardeigenschaften, die für jedes Programm gelten können/sollten wie z.B. keine Nullpointer-Dereferenzierung, keine SQL-Injection-Möglichkeit usw. und manuell spezifizierte Eigenschaften z.B. das die Sortierfunktion auch wirklich eine sortierte Liste zurückgibt. Standardeigenschaften sind eine Teilmenge von manuell spezifizierten Eigenschaften.

Verwundbarkeiten Die Methoden können auch anhand der von ihnen überprüften Verwundbarkeiten unterschieden werden. So kann entweder überprüft werden, ob das zu untersuchende System Indizien für bestimmte bekannte Schwachstellen aufweist oder es können unbekannte Schwachstellen gesucht werden.

Eigenschaft	Merkmalsausprägung 1	Merkmalsausprägung 2	Merkmalsausprägung 3
	voll automatisch	teilautomatisiert	manuell
Automatisierbarkeit	Verwundbarkeitsscanner Fuzzing Fault Injection Robustheittests	Statische Code-Analyse Dynamische Code-Analyse Model checking Taint Analysis Konformitätstest Formale Methoden Penetration Test	Code Review Deduktiv
	Falsifikation	Verifikation	
positives/negatives Verfahren	Verwundbarkeitsscanner Fuzzing Fault Injection Robustheittests Statische Code-Analyse Dynamische Code-Analyse Taint Analysis Code Review Penetration Test	Konformitätstest Model checking Formale Methoden Deduktiv	
	dynamisch	statisch	
dynamisch/statisch	Fuzzing Fault Injection Robustheittests Verwundbarkeitsscanner Dynamische Code-Analyse Taint Analysis Penetration Test	Model checking Formale Methoden Deduktiv Code Review Konformitätstest Statische Code-Analyse	

Eigenschaft	Merkmalsausprägung 1	Merkmalsausprägung 2	Merkmalsausprägung 3
	Ja	Nein	
Black-boxfähig	Verwundbarkeitsscanner Fuzzing Fault Injection Robustheittests Konformitätstest Penetration Test	Dynamische Code-Analyse Model checking Taint Analysis Formale Methoden Code Review Deduktiv	
	Standardeigenschaften	manuell spezifiziert	
Getestete Eigenschaften	Verwundbarkeitsscanner Fuzzing Fault Injection Robustheittests Statische Code-Analyse Taint Analysis Penetration Test	Dynamische Code-Analyse Code Review Konformitätstest Model checking Formale Methoden Deduktiv	
	bekannte Verwundbarkeiten	unbekannte Verwundbarkeiten	
Verwundbarkeiten	Verwundbarkeitsscanner Model checking Formale Methoden Deduktiv Code Review Konformitätstest Robustheittests Dynamische Code-Analyse Taint Analysis Penetration Test	Penetration Test Fuzzing Fault Injection	

Literatur

[1] TIAN-YANG, Gu ; YIN-SHENG, Shi ; YOU-YUAN, Fang: Research on software security testing. In: *World Academy of science, engineering and Technology* 69 (2010), S. 647–651

[2] ARMSTRONG, Robert C. ; PUNNOOSE, Ratish J. ; WONG, Matthew H. ; MAYO, Jackson R.: Survey of existing tools for formal verification. In: *SANDIA REPORT SAND2014-20533* (2014)

[3] FELDERER, Michael ; BÜCHLER, Matthias ; JOHNS, Martin ; BRUCKER, Achim D. ; BREU, Ruth ; PRETSCHNER, Alexander: Security testing: A survey. In: *Advances in Computers* Bd. 101. Elsevier, 2016, S. 1–51

[4] PISTOIA, Marco ; CHANDRA, Satish ; FINK, Stephen J. ; YAHAV, Eran: A survey of static analysis methods for identifying security vulnerabilities in software systems. In: *IBM Systems Journal* 46 (2007), Nr. 2, S. 265–288

[5] CHESS, Brian ; MCGRAW, Gary: Static analysis for security. In: *IEEE security & privacy* 2 (2004), Nr. 6, S. 76–79

[6] POTTER, Bruce ; MCGRAW, Gary: Software security testing. In: *IEEE Security & Privacy* 2 (2004), Nr. 5, S. 81–85

[7] BINKLEY, David: Source code analysis: A road map. In: *2007 Future of Software Engineering* IEEE Computer Society, 2007, S. 104–119

[8] BAU, Jason ; BURSZTEIN, Elie ; GUPTA, Divij ; MITCHELL, John: State of the art: Automated black-box web application vulnerability testing. In: *2010 IEEE Symposium on Security and Privacy* IEEE, 2010, S. 332–345

[9] FELDERER, Michael ; ZECH, Philipp ; BREU, Ruth ; BÜCHLER, Matthias ; PRETSCHNER, Alexander: Model-based security testing: a taxonomy and systematic classification. In: *Software Testing, Verification and Reliability* 26 (2016), Nr. 2, S. 119–148

[10] PFRANG, Steffen ; MEIER, David ; FRIEDRICH, Michael ; BEYERER, Jürgen: Advancing Protocol Fuzzing for Industrial Automation and Control Systems. In: *ICISSP*, 2018, S. 570–580

[11] MILLER, Barton P. ; FREDRIKSEN, Louis ; SO, Bryan: An empirical study of the reliability of UNIX utilities. In: *Communications of the ACM* 33 (1990), Nr. 12, S. 32–44

[12] (OWASP), Open Web Application Security P.: *Category: Vulnerability Scanning Tools*. `https://www.owasp.org/index.php/Category:Vulnerability_Scanning_Tools`

[13] LIVSHITS, V B. ; LAM, Monica S.: Finding Security Vulnerabilities in Java Applications with Static Analysis. In: *USENIX Security Symposium* Bd. 14, 2005, S. 18–18

[14] FAGAN, Michael: Design and code inspections to reduce errors in program development. In: *Software pioneers*. Springer, 2002, S. 575–607

[15] SIY, Harvey ; VOTTA, Lawrence: Does the modern code inspection have value? In: *Proceedings of the IEEE international Conference on Software Maintenance (ICSM'01)* IEEE Computer Society, 2001, S. 281

[16] MCINTOSH, Shane ; KAMEI, Yasutaka ; ADAMS, Bram ; HASSAN, Ahmed E.: An empirical study of the impact of modern code review practices on software quality. In: *Empirical Software Engineering* 21 (2016), Nr. 5, S. 2146–2189

[17] BAIER, Christel ; KATOEN, Joost-Pieter: *Principles of model checking*. MIT press, 2008

[18] CAI, Jun ; ZOU, Peng ; MA, Jinxin ; HE, Jun: SwordDTA: A dynamic taint analysis tool for software vulnerability detection. In: *Wuhan University Journal of Natural Sciences* 21 (2016), Nr. 1, S. 10–20

[19] (OWASP), Open Web Application Security P.: *Fuzzing*. `https://www.owasp.org/index.php/Fuzzing`

[20] LINUX, BlackArch: *Fuzzer*. `https://blackarch.org/fuzzer.html`

[21] DOUPÉ, Adam ; COVA, Marco ; VIGNA, Giovanni: Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In: *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* Springer, 2010, S. 111–131

[22] KHOURY, Nidal ; ZAVARSKY, Pavol ; LINDSKOG, Dale ; RUHL, Ron: An analysis of black-box web application security scanners against stored SQL injection. In: *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing* IEEE, 2011, S. 1095–1101

[23] MCALLISTER, Sean ; KIRDA, Engin ; KRUEGEL, Christopher: Leveraging user interactions for in-depth testing of web applications. In: *International Workshop on Recent Advances in Intrusion Detection* Springer, 2008, S. 191–210

[24] FERREIRA, Alexandre M. ; KLEPPE, Harald: *Effectiveness of automated application penetration testing tools*. 2011

[25] VEGA, Esteban Alejandro A. ; OROZCO, Ana Lucila S. ; VILLALBA, Luis Javier G.: Benchmarking of Pentesting Tools. In: *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering* 11 (2017), Nr. 5, S. 590–593

[26] IDRISSI, SE ; BERBICHE, N ; GUEROUATE, F ; SHIBI, M: Performance evaluation of web application security scanners for prevention and protection against vulnerabilities. In: *International Journal of Applied Engineering Research* 12 (2017), Nr. 21, S. 11068–11076

[27] ALASSMI, Shafi ; ZAVARSKY, Pavol ; LINDSKOG, Dale ; RUHL, Ron ; ALASIRI, Ahmed ; ALZAIDI, Muteb: An analysis of the Effectiveness of Black-box Web Application Scanners in Detection of Stored XSSI Vulnerabilities. In: *International Journal of Information Technology and Computer Science* 4 (2012), Nr. 1

[28] SUTO, Larry: Analyzing the accuracy and time costs of web application security scanners. In: *San Francisco, February* (2010)

[29] AKHAWE, Devdatta ; BARTH, Adam ; LAM, Peifung E. ; MITCHELL, John ; SONG, Dawn: Towards a formal foundation of web security. In: *2010 23rd IEEE Computer Security Foundations Symposium* IEEE, 2010, S. 290–304

[30] BAU, Jason: *Network and Web Security Modeling and Analysis*, Stanford University, Diss., 2014

[31] BRUNEL, Julien ; RIOUX, Laurent ; PAUL, Stéphane ; FAUCOGNEY, Anthony ; VALLÉE, Frédérique: Formal safety and security assessment of an avionic architecture with alloy. In: *arXiv preprint arXiv:1405.1113* (2014)

[32] BUGLIESI, Michele ; CALZAVARA, Stefano ; FOCARDI, Riccardo: Formal methods for web security. In: *Journal of Logical and Algebraic Methods in Programming* 87 (2017), S. 110–126

[33] BEYER, Dirk: Second competition on software verification. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* Springer, 2013, S. 594–609

[34] BEYER, Dirk: Competition on software verification. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* Springer, 2012, S. 504–524

[35] NIEDERMAIER, Matthias ; FISCHER, Florian ; BODISCO, Alexander von: PropFuzz—An IT-security fuzzing framework for proprietary ICS protocols. In: *2017 International Conference on Applied Electronics (AE)* IEEE, 2017, S. 1–4

[36] FOWLER, Daniel S. ; BRYANS, Jeremy ; SHAIKH, Siraj A. ; WOODERSON, Paul: Fuzz Testing for Automotive Cyber-Security. In: *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)* IEEE, 2018, S. 239–246

[37] MELO, Bruno da S. ; GEUS, Paulo L. ; GRÉGIO, André A: Robustness Testing of CoAP Server-side Implementations through Black-box Fuzzing Techniques. (2017)

[38] CLARKE, Edmund ; KROENING, Daniel ; LERDA, Flavio: A Tool for Checking ANSI-C Programs. In: JENSEN, Kurt (Hrsg.) ; PODELSKI, Andreas (Hrsg.): *Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2004)* Bd. 2988, Springer, 2004 (Lecture Notes in Computer Science). – ISBN 3–540–21299–X, S. 168–176

[39] GANESAN, Aishwarya ; ALAGAPPAN, Ramnatthan ; ARPACI-DUSSEAU, Andrea C. ; ARPACI-DUSSEAU, Remzi H.: Redundancy Does Not Imply Fault Tolerance: Analysis of Distributed Storage Reactions to Single Errors and Corruptions. In: *15th USENIX Conference on File and Storage*

Technologies (FAST 17). Santa Clara, CA : USENIX Association, 2017. – ISBN 978–1–931971–36–2, 149–166

[40] EMANUELSSON, Pär ; NILSSON, Ulf: A comparative study of industrial static analysis tools. In: *Electronic notes in theoretical computer science* 217 (2008), S. 5–21

[41] BEYER, Dirk: Automatic verification of C and Java programs: SV-COMP 2019. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems* Springer, 2019, S. 133–155

[42] DOUPÉ, Adam ; CAVEDON, Ludovico ; KRUEGEL, Christopher ; VIGNA, Giovanni: Enemy of the state: A state-aware black-box web vulnerability scanner. In: *Presented as part of the 21st {USENIX} Security Symposium ({USENIX} Security 12)*, 2012, S. 523–538

[43] PARIENTE, Dillon ; LEDINOT, Emmanuel: Formal verification of industrial C code using Frama-C: a case study. In: *Formal Verification of Object-Oriented Software* (2010), S. 205

[44] MANTERE, Matti ; UUSITALO, Ilkka ; RONING, Juha: Comparison of static code analysis tools. In: *2009 Third International Conference on Emerging Security Information, Systems and Technologies* IEEE, 2009, S. 15–22

[45] QIU, Lina ; WANG, Yingying ; RUBIN, Julia: Analyzing the Analyzers: FlowDroid/IccTA, AmanDroid, and DroidSafe. In: *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York, NY, USA : ACM, 2018 (ISSTA 2018). – ISBN 978–1–4503–5699–2, 176–186

[46] BODDEN, Eric ; PUN, Ka I. ; STEFFEN, Martin ; STOLZ, Volker ; WICKERT, Anna-Katharina: Information flow analysis for Go. In: *International Symposium on Leveraging Applications of Formal Methods* Springer, 2016, S. 431–445

[47] MEIER, David ; PFRANG, Steffen ; BORCHERDING, Anne ; BEYERER, Jürgen: Automated Security Testing for Web Applications on Industrial Automation and Control Systems. In: *at-Automatisierungstechnik* 67 (2019), Nr. 5

[48] PFRANG, Steffen ; MEIER, David ; KAUTZ, Valentin: Towards a modular security testing framework for industrial automation and control systems: Isutest. In: *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)* IEEE, 2017, S. 1–5

[49] DE GOUW, Stijn ; ROT, Jurriaan ; BOER, Frank S. ; BUBEL, Richard ; HÄHNLE, Reiner: OpenJDK's Java. utils. Collection. sort () is broken: The good, the bad and the worst case. In: *International Conference on Computer Aided Verification* Springer, 2015, S. 273–289

[50] BECKERT, Bernhard ; SCHIFFL, Jonas ; SCHMITT, Peter H. ; ULBRICH, Mattias: Proving jdk's dual pivot quicksort correct. In: *Working Conference on Verified Software: Theories, Tools, and Experiments* Springer, 2017, S. 35–48

[51] WARD, NJ: Code Verification with the aid of MALPAS. In: *IEE Colloquium on High Integrity Ada* IET, 1991, S. 3–1

[52] HAYMAN, KJ: An analysis of ordnance software using the MALPAS tools / ELECTRONICS RESEARCH LAB ADELAIDE (AUSTRALIA). 1989. – Forschungsbericht

[53] SECURITY, Tenable N.: *Software Requirements for Nessus*. `https://docs.tenable.com/nessus/Content/SoftwareRequirements.htm`

[54] MCMAHON, Emma ; PATTON, Mark ; SAMTANI, Sagar ; CHEN, Hsinchun: Benchmarking Vulnerability Assessment Tools for Enhanced Cyber-Physical System (CPS) Resiliency. In: *2018 IEEE International Conference on Intelligence and Security Informatics (ISI)* IEEE, 2018, S. 100–105

[55] KARABAŠEVIĆ, Darjan ; STANUJKIĆ, Dragiša ; BRZAKOVIĆ, Miodrag ; MAKSIMOVIĆ, Mlađan ; JEVTIĆ, Milena: Importance of vulnerability scanners for improving security and protection of the web servers. In: *Bizinfo (Blace)* 9 (2018), Nr. 1, S. 19–29

[56] GOURDIN, Baptiste ; SOMAN, Chinmay ; BOJINOV, Hristo ; BURSZTEIN, Elie: Toward Secure Embedded Web Interfaces. In: *USENIX Security Symposium* Bd. 14, 2011, S. 113

[57] REBERT, Alexandre ; CHA, Sang K. ; AVGERINOS, Thanassis ; FOOTE, Jonathan ; WARREN, David ; GRIECO, Gustavo ; BRUMLEY, David: Optimizing seed selection for fuzzing. In: *23rd {USENIX} Security Symposium ({USENIX} Security 14)*, 2014, S. 861–875

[58] MCNALLY, Richard ; YIU, Ken ; GROVE, Duncan ; GERHARDY, Damien: Fuzzing: the state of the art / DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION EDINBURGH (AUSTRALIA). 2012. – Forschungsbericht

[59] ESPOSITO, Damiano ; RENNHARD, Marc ; RUF, Lukas ; WAGNER, Arno: Exploiting the potential of web application vulnerability scanning. In: *ICIMP 2018, Spain, July 22-26, 2018* IARIA, 2018, S. 22–29

[60] MAKINO, Yuma ; KLYUEV, Vitaly: Evaluation of web vulnerability scanners. In: *2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)* Bd. 1 IEEE, 2015, S. 399–402

[61] CHAPMAN, Roderick ; SCHANDA, Florian: Are we there yet? 20 years of industrial theorem proving with SPARK. In: *International Conference on Interactive Theorem Proving* Springer, 2014, S. 17–26

[62] DE RENESSE, F ; AGHVAMI, AH: Formal verification of ad-hoc routing protocols using SPIN model checker. In: *Proceedings of the 12th IEEE Mediterranean Electrotechnical Conference (IEEE Cat. No. 04CH37521)* Bd. 3 IEEE, 2004, S. 1177–1182

[63] HESSEL, Anders ; LARSEN, Kim G. ; MIKUCIONIS, Marius ; NIELSEN, Brian ; PETTERSSON, Paul ; SKOU, Arne: Testing real-time systems using UPPAAL. In: *Formal methods and testing*. Springer, 2008, S. 77–117

[64] HAVELUND, Klaus ; LARSEN, Kim G. ; SKOU, Arne: Formal verification of a power controller using the real-time model checker Uppaal. In: *International AMAST Workshop on Aspects of Real-Time Systems and Concurrent and Distributed Software* Springer, 1999, S. 277–298

[65] BENGTSSON, Johan ; LARSEN, Kim ; LARSSON, Fredrik ; PETTERSSON, Paul ; YI, Wang: UPPAAL — a tool suite for automatic verification of real-time systems. In: ALUR, Rajeev (Hrsg.) ; HENZINGER, Thomas A. (Hrsg.) ; SONTAG, Eduardo D. (Hrsg.): *Hybrid Systems III*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1996, S. 232–243

[66] NETHERCOTE, Nicholas ; SEWARD, Julian: Valgrind: a framework for heavyweight dynamic binary instrumentation. In: *ACM Sigplan notices* Bd. 42 ACM, 2007, S. 89–100

[67] SUTEVA, Natasa ; ZLATKOVSKI, Dragi ; MILEVA, Aleksandra: Evaluation and testing of several free/open source web vulnerability scanners. In: *The 10th Conference for Informatics and Information Technology (CIIT 2013)* (2013)

[68] GANDHI, Dr C. ; SURI, Gaurav ; GOLYAN, Rishi P. ; SAXENA, Pupul ; SAXENA, Bhavya K.: Packet sniffer–a comparative study. In: *International Journal of Computer Networks and Communications Security* 2 (2014), Nr. 5, S. 179–187

[69] DURAES, Joao A. ; MADEIRA, Henrique S.: Emulation of software faults: A field data study and a practical approach. In: *Ieee transactions on software engineering* 32 (2006), Nr. 11, S. 849–867