Testabdeckung und Penetration Testing

Code Coverage zeigt Angriffsvektoren auf

Die Messung der Testabdeckung ist ein unverzichtbarer Standard, wenn es um die funktionale Sicherheit von Embedded-Systemen geht. Doch haben die dafür genutzten Code-Coverage-Analyzer noch einen weiteren Nutzen, der wenig bekannt ist: Code Coverage ist sehr hilfreich, um im Rahmen des Penetration Testings Schwachstellen im Code zu erkennen und in ihrer Brisanz zu bewerten. Vor allem innerhalb des Entwicklungszyklus können die Programmierer und Teamleiter so frühzeitig Sicherheitslücken aufdecken und zeitnah schließen.

Von Dr. Sabine Poehler, Product Manager bei Verifysoft Technology GmbH

Die aktuellen Entwicklungen im Automobilbau sorgen dafür, dass die Fahrzeuge häufig als „Rechenzentrum auf Rädern“ pointiert werden. Die Formulierung ist nicht ganz falsch, im Premium-Segment arbeiten heute zum Teil über 100 verschiedene Systeme, die gemeinsam auf über 100 Millionen Code-Zeilen kommen. Und viele dieser Systeme kommunizieren untereinander und mit der Außenwelt. Besonders die Kommunikation mit externen Quellen steigt rasant an, Ansätze V2V (Vehicle to Vehicle) und V2x (Vehicle to anything) entwickeln sich schnell weiter. Doch im Gegensatz zum klassischen Rechenzentrum ist ein Auto bislang noch nicht konsequent darauf ausgelegt, digitalen Angriffen standzuhalten. Denn hier treffen sehr unterschiedliche Technologien aus der Automotive- und der IT-Welt zusammen. So sind zum Beispiel Kommunikationstechnologien wie CANbus oder Automotive Ethernet optimal auf den fahrzeuginternen Bedarf ausgerichtet, verfügen aber nicht über eingebaute Schutzmechanismen. Zudem müssen die Systeme grundsätzlich zugänglich sein, etwa für Werkstätten außerhalb des Herstellernetzes. Gleichzeitig müssen viele Systeme den Anforderungen der funktionalen Sicherheit genügen, wie sie in der generischen Norm IEC 61508 oder im Automobil-spezifischen Standard ISO 26262 verbindlich festgelegt sind. Ein wesentlicher Aspekt der Vorgaben ist die Qualitätssicherung über den gesamten Lebenszyklus der Systeme hinweg. Dabei spielt die Entwicklungsphase eine zentrale Rolle.

Innerhalb der Entwicklung sind das Testing und die statische Code-Analyse grundlegende Pfeiler der Qualitätssicherung. Zur Dokumentation und Überwachung des dynamischen Testings dient die Testabdeckung, die Code Coverage. Das Verfahren ermittelt, ob alle Bestandteile des Codes wie gefordert durchlaufen wurden. Code Coverage gehört bei der Überprüfung der funktionalen Sicherheit der Systeme zum Standard im Testing-Werkzeugkasten, um die je nach Gefahrenklasse geforderte Testabdeckung nachzuweisen. Dafür zuständig sind Code Coverage Analyzer wie etwa Testwell CTC++, der speziell für den Einsatz im Embedded-Bereich konzipiert ist. Denn Embedded-Systeme stellen hier

spezielle Anforderungen: Code Coverage Analyzer ergänzen den Code für die geforderte Testabdeckung mit Zählern, um zu erfassen, welche Code-Teile wie oft durchlaufen wurden. Durch diese Instrumentierung wird der Code erweitert. Bei kleinen Targets, wie sie aus Kostengründen im Automotive-Bereich weit verbreitet sind, ist die Hardware mit nur sehr geringen Leistungsreserven dimensioniert. Darauf muss der Code Coverage Analyzer Rücksicht nehmen.

Erfolgreiche Angriffe sind nur eine Frage der Zeit

Code Coverage Analyzer eignen sich allerdings nicht nur zur Messung und zur Dokumentation der Testabdeckung, also welche Teile des Codes beim Testing durchlaufen wurden. Sie können umgekehrt auch ermitteln, welche Teile eben nicht durchlaufen wurden. Diesen Ansatz kann man sich im Rahmen des Penetration-Testings zu Nutze machen. Beim Penetration-Testing versucht der Prüfer, in das zu testende System unautorisiert einzudringen. Dazu bedient er sich der gleichen Methoden, die auch ein echter Angreifer nutzen würde. Durch die rasant zunehmende Konnektivität der Fahrzeugsysteme sind auch diese heute gefährdet. Die Frage ist nicht, ob ein echter Angreifer erfolgreiche Attacken gegen Kraftfahrzeuge durchführt. Dass dieses möglich ist, wurde in der Vergangenheit bereits mehrfach demonstriert. Die Frage ist, wann das passieren wird. Auch der Embedded-Bereich und besonders die Automobilindustrie können also auf Dauer nicht auf Penetrationstests verzichten.

In der Regel werden Penetrationstests an externe Dienstleister ausgelagert. Auch das Bundesamt für Sicherheit in der Informationstechnik (BSI) empfiehlt dieses Vorgehen, um die Unabhängigkeit der Prüfer vom zu testenden System zu gewährleisten. Auch sind die Prüfer mit Absicht in vielen Fällen nicht genau mit dem Prüfobjekt vertraut. Bei so genannten Blackbox-Tests haben die Prüfer möglichst wenig Informationen über das System. Dadurch sollen sie eine ähnliche Herangehensweise haben wie ein externer Eindringling. Der Nachteil des Blackbox-Ansatzes ist, dass die Prüfer unter Umständen Ergebnisse nicht korrekt interpretieren können. Dem gegenüber steht der Whitebox-Test, bei dem die Prüfer möglichst viel über das Prüfobjekt wissen.

Externe Berichte interpretieren

Was aus Sicherheitsgründen sinnvoll und nachvollziehbar ist, stellt die Entwicklungsteams jedoch vor Herausforderungen: Sie erhalten von den externen Prüfern Reports mit allen gefundenen – oder vermeintlichen – Schwachstellen der Software. Die Entwickler müssen anhand dieser Reports ihre Software verbessern und sich unter Umständen gegenüber der Unternehmensleitung für die aufgelisteten Sicherheitslücken rechtfertigen. Die Berichte der Penetrationstests sind zwar in der Regel aussagekräftig, haben aber zwei Schwächen: Zum einen zeigen sie nur auf, wo Schwachstellen zu

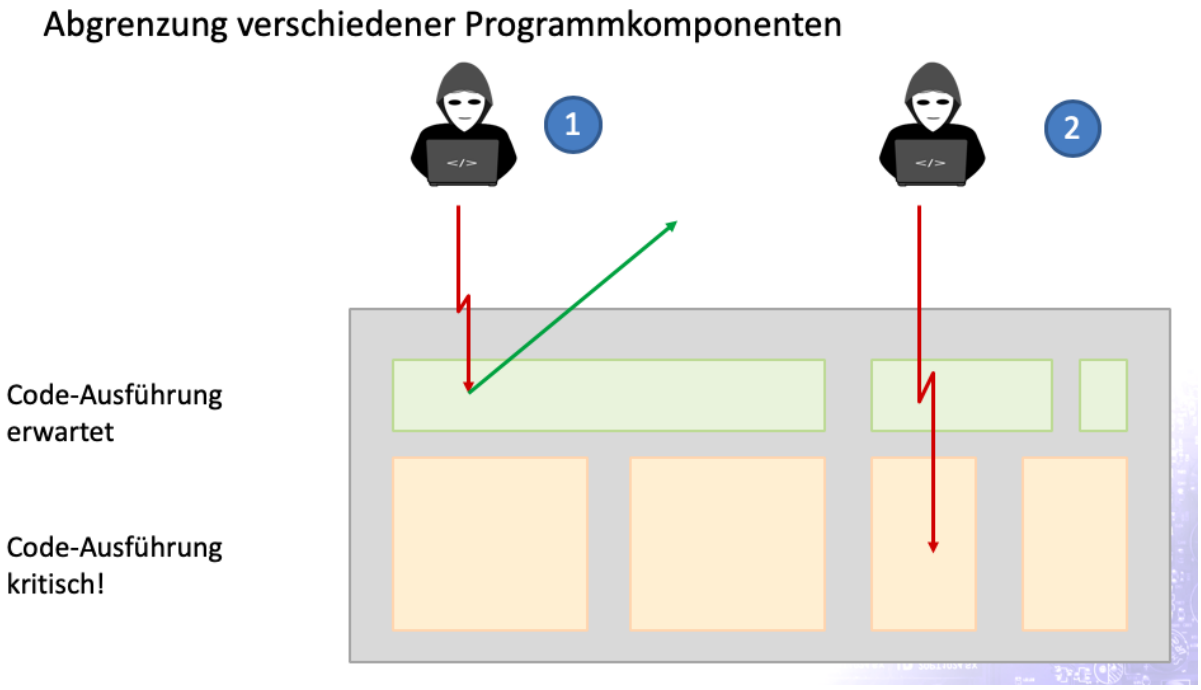
finden sind. Ob andere Bereiche des Systems wirklich sicher sind, geht aus den Berichten nicht unbedingt hervor. Zum anderen können die Testberichte auch durch den Blackbox-Ansatz Missinterpretationen und Fehleinschätzungen der Prüfer enthalten. In diesem Fall ist es für die Entwickler wichtig nachzuweisen, dass eine bestimmte Schwachstelle trotz Finding im Report nicht vorliegt.

Für die Entwicklerteams ist es also extrem hilfreich, wenn sie die gefundenen Angriffsflächen selbst an ihrem Code nachvollziehen können. Da in der Regel die vorhandene Werkzeugkiste einen Code Coverage Analyzer enthält, kann dieser dazu genutzt werden, selbst ein Penetration-Testing als Whitebox-Test durchzuführen. Dabei wird folgendermaßen vorgegangen: Vom zu testenden Programm wird eine instrumentierte Version erstellt, also eine Version, die durch den Code Coverage Analyzer beim Kompilieren um Zähler angereichert wurde. Mit dieser instrumentierten Version wird der Penetrationstest durchgeführt. Die beiden Schritte „Messung der Codeabdeckung“ und „Durchführung eines Penetrationstests“ sind dabei unabhängig voneinander. Werden sie von verschiedenen Teams oder sogar Unternehmen durchgeführt, wird die Arbeit der Penetrationstest-Teams nicht beeinflusst.

Welche Bereiche kann der Angriff erreichen?

Dabei sollten sich die Entwickler und Tester bereits im Vorfeld Gedanken machen, welche Code-Teile beim Penetrationstest auf jeden Fall ausgeführt werden müssen und welche Bereiche der Software so kritisch sind, dass der Test hier keinesfalls erfolgreich sein darf. Die folgende Analysephase aufgrund des Coverage-Berichts beginnt also mit folgenden Beobachtungen:

- Ein Code-Bestandteil wurde während des Penetrationstests erwartungsgemäß durchlaufen: Das gilt üblicherweise für weite Teile des Systems, und stellt eine Art Grundrauschen dar, das in der folgenden Analyse ausgeblendet werden muss.
- Eine Komponente wurde während des Tests wider Erwarten durchlaufen: Hier liegt ein erster Hinweis auf eine Sicherheitslücke vor.
- Eine Komponente wurde durch den Penetrationstest erwartungsgemäß nicht erreicht.
- Eine Komponente wurde durch den Penetrationstest wider Erwarten nicht erreicht: Das kann ein Hinweis darauf sein, dass der Penetrationstest nicht vollständig genug war.



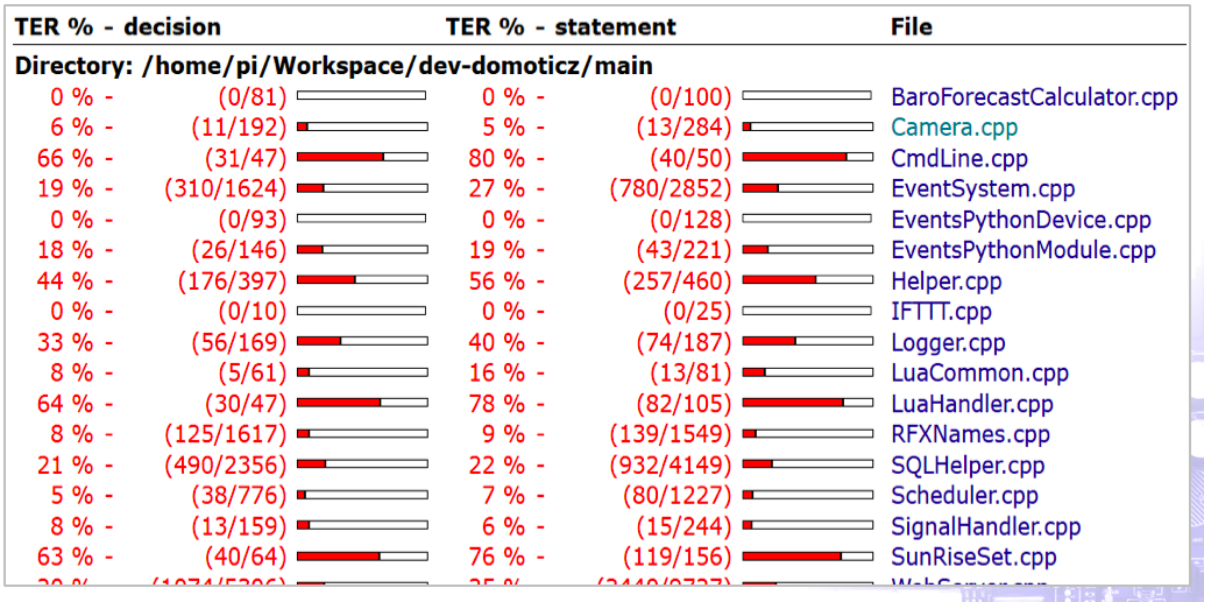
Abbilidung 1: Vor dem Penetrationstest muss eruiert werden, welche Code-Teile im Test durchlaufen werden sollten und welche nicht.

Ein fundamentaler Punkt bei dieser Einteilung ist die vorab zu formulierende Erwartung, was in den verschiedenen Komponenten des Systems passieren wird. Ein Validator für Benutzereingaben zum Beispiel wird erwartungsgemäß während eines Penetrationstests durchlaufen. Funktionen, die nur für die Fernwartung eines Geräts vorgesehen sind, sollten dagegen auf keinen Fall aufgerufen werden, wenn der Penetrationstest im Kontext der Standardnutzung durchgeführt wird. In der folgenden Detailanalyse stehen insbesondere die unerwarteten oder nicht nachvollziehbaren Resultate des Penetrationstests im Vordergrund: Für diese tatsächlichen oder vermeintlichen Sicherheitslücken kann Schritt für Schritt nachvollzogen werden, wie weit der Angreifer ins System eindringen konnte. Die Schwachstelle im System wird so eingekreist. Und damit entsprechend der davon ausgehenden Risiken bewertet und priorisiert werden. Denn nicht alle Schwachstellen, die durch Penetrationstests zutage gefördert werden, erfordern eine sofortige Reaktion der Entwickler. Das Bundesamt für Sicherheit in der Informationstechnik (BSI) teilt gefundene Schwachstellen anhand des Schadenspotenzials in vier Kategorien ein:

- **Hohes Schadenspotenzial:** Sofortige Reaktion erforderlich - Fremdsteuerung des Prüfobjekts durch Angreifer, Verlust von sensiblen Daten und Veränderung des Prüfobjekts oder Auslesen von sensiblen Daten durch den Angreifer sind möglich.

- **Mittleres Schadenspotenzial:** Kurzfristige Reaktion erforderlich – Es gibt Schwachstellen im Prüfobjekt, die schwerwiegende Angriffe ermöglichen können.
- **Niedriges Schadenspotenzial:** Mittelfristige Reaktion erforderlich - Es gibt Schwachstellen im Prüfobjekt, die ein unbestimmtes Angriffspotenzial aufweisen.
- **Zur Information:** Langfristige Reaktion erforderlich – Das Prüfobjekt weist Verbesserungspotenzial auf.

Wie immer bei der Qualitätssicherung in der Softwareentwicklung gilt auch hier: Je früher Probleme und Fehler erkannt werden, desto einfacher und kostengünstiger können sie auch beseitigt werden. Sich auf die Berichte der externen Penetrationstester zu verlassen und darauf zu vertrauen, dass diese alle Probleme erkennen und richtig interpretieren, wäre blauäugig. Die Entwickler sollten selbst in der Lage sein, mögliche Sicherheitslücken nachzuvollziehen, diese frühzeitig aktiv adressieren und so aus den Reports der externen Tester den optimalen Nutzen ziehen.



TER % - decision		TER % - statement		File
Directory: /home/pi/Workspace/dev-domoticz/main				
0 % -	(0/81)	0 % -	(0/100)	BaroForecastCalculator.cpp
6 % -	(11/192)	5 % -	(13/284)	Camera.cpp
66 % -	(31/47)	80 % -	(40/50)	CmdLine.cpp
19 % -	(310/1624)	27 % -	(780/2852)	EventSystem.cpp
0 % -	(0/93)	0 % -	(0/128)	EventsPythonDevice.cpp
18 % -	(26/146)	19 % -	(43/221)	EventsPythonModule.cpp
44 % -	(176/397)	56 % -	(257/460)	Helper.cpp
0 % -	(0/10)	0 % -	(0/25)	IFTTT.cpp
33 % -	(56/169)	40 % -	(74/187)	Logger.cpp
8 % -	(5/61)	16 % -	(13/81)	LuaCommon.cpp
64 % -	(30/47)	78 % -	(82/105)	LuaHandler.cpp
8 % -	(125/1617)	9 % -	(139/1549)	RFXNames.cpp
21 % -	(490/2356)	22 % -	(932/4149)	SQLHelper.cpp
5 % -	(38/776)	7 % -	(80/1227)	Scheduler.cpp
8 % -	(13/159)	6 % -	(15/244)	SignalHandler.cpp
63 % -	(40/64)	76 % -	(119/156)	SunRiseSet.cpp

Bild 2: Die Statement Coverage auf Dateiebene, hier mit Testwell CTC++, gibt eine erste Orientierung, welche Code-Teile vom Penetrationstest überhaupt erreicht wurden.

Fazit

Auch im Embedded-Bereich werden Penetrationstests über kurz oder lang zumindest bei kritischen Systemen zum Standard gehören. Die hohe Vernetzung von Devices im Feld und den Backend-Systemen mit teils sensiblen oder zumindest unternehmenswichtigen Daten erfordert ein entsprechendes Sicherheitsniveau. Der Aufwand, Penetrationstests mittels Code Coverage aussagekräftiger zu machen, ist gering und hilft unmittelbar dem Entwickler bei der Analyse des Penetrationstests. Dadurch lässt sich zum einen die Code-Qualität weiter steigern und zum anderen das Risiko einer verzögerten Markteinführung senken. Auch sind proaktive Penetrationstests gerade in einem sensiblen Marktumfeld wie der Medizintechnik ein wichtiger Baustein, um langfristige und vertrauensvolle Kundenbeziehungen zu pflegen.

Eigene Penetrationstests sind wichtig

Grundsätzlich ist es sinnvoll, dass die Entwicklungsabteilung selbst Penetrationstests vornimmt und deren Ergebnisse mit einem Code Coverage Analyzer erfasst. Denn werden diese Tests erst auf Veranlassung des Kunden durchgeführt, gerät das Entwicklungsteam leicht in eine schwache Position und kann die Tests nicht mehr aktiv mitgestalten. Zudem werden externe Penetrationstests aus Kostengründen meist erst am fertigen Produkt vorgenommen. Tauchen dann noch gravierende Probleme auf, entstehen zum Teil immense Aufwände. Im schlimmsten Fall können diese Lücken nicht beseitigt werden, der Code muss in Teilen neu geschrieben werden. Die Open-Source-Welt hält einige ausgefeilte Test-Suiten für Penetrationstests bereit, die auch von den Entwicklern eingesetzt werden können – das nötige Wissen vorausgesetzt. Die Code Coverage hilft in diesem Fall dabei, sowohl den Penetrationstest vollständig durchzuführen als auch die Schwachstellen der eigenen Software nachvollziehen zu können und dann so früh wie möglich darauf zu reagieren.