Effizientere IT-Sicherheitstest mit Hilfe von Usage-based Testing

Martin A. Schneider
Fraunhofer-Institut FOKUS
Berlin, Deutschland
martin.schneider@fokus.fraunhofer.de

Steffen Herbold
Institut für Informatik
Universität Göttingen
Göttingen, Deutschland
herbold@cs.uni-goettingen.de

***Abstract*—IT-Sicherheitstests untersuchen Systeme auf sicherheitsrelevante Schwachstellen, indem diese ausgeführt werden. Eine inzwischen verbreitete Technik hierfür ist das sogenannte Fuzzing, bei dem die Schnittstellen eines Systems mit ungültigen Daten stimuliert werden. Diese können zufallsbasiert, mit Beschreibungen der Eingabedatenformate, beispielsweise mit Hilfe von Grammatiken, oder zusätzlich mit Verhaltensmodellen automatisiert erzeugt werden. Da der Eingaberaum für ungültige Daten riesig oder gar unendlich groß ist, stellt sich die Herausforderungen, wie man das System effizient mit den vorhandenen Ressourcen testet. Wir möchten hier eine Idee zur Kombination von Usage-Based Testing und IT-Sicherheitstesten vorstellen, die dieses Problem abmildern kann.**

Keywords— IT-Sicherheitstests; Fuzzing; Benutzungsbasiertes Testen; Usagge-Based Testing

I. EINFÜHRUNG

IT-Sicherheitstests dienen dazu, potenzielle sicherheitsrelevante Schwachstellen in Systemen aufzudecken. Mit der zunehmenden Komplexität der Systeme und der immer stärkeren Vernetzung von bisher isolierten Systemen (zum Beispiel cyber-physische Systeme, Internet der Dinge) entstehen ganz neue Herausforderungen, deren Sicherheit zu gewährleisten. Mit den bisherigen Methoden wird der Aufwand, diese Systeme auf ihre Sicherheit zu testen, steigen, und es werden ganz neue Systeme auf ihre IT-Sicherheit getestet werden müssen, für die dies bisher nicht notwendig war. Die bisherigen Methoden und Techniken müssen daher effizienter werden, um den höheren Aufwänden bei hinreichender Qualitätssicherung, gerade bezüglich IT-Sicherheit, gerecht werden zu können.

Eine inzwischen verbreitete Methode für IT-Sicherheitstests ist das sog. Fuzzing. Dabei wird das zu testende System mit ungültigen Eingaben stimuliert, die zu sicherheitsrelevanten Problemen führen können, wenn diese vom System nicht abgewiesen, sondern verarbeitet werden. Ein klassisches Beispiel ist der sogenannte Buffer Overflow, bei dem die Länge eines Eingabedatums nicht überprüft wird, so dass sehr lange Eingaben Speicherbereiche überschreiben und so Code und Daten in ein System einschleusen können. Diese ungültigen Daten können beispielsweise zufallsbasiert, mit Hilfe von Grammatiken für gültige Eingabedaten oder auch zusätzlich mit Hilfe von Verhaltensmodellen erzeugt werden. Da der Raum der ungültigen Eingabedaten riesig bzw. sogar unendlich groß ist, müssen möglichst effizient die Daten gefunden werden, die tatsächlich in der Lage sind, Schwachstellen aufzudecken. Dazu wird versucht, sogenannte semi-valide Eingabedaten zu erzeugen, also Daten, die größtenteils gültig und nur in wenigen Punkten ungültig sind. Um auf eine Buffer Overflow-Schwachstelle zu testen, würde man gültige Daten ungültiger Länge erzeugen. Ziel ist es damit, nur einzelne Mechanismen in der Schnittstelle des zu testenden Systems zu identifizieren, die fehlerhaft arbeiten. Trotzdem ist die Menge der so erzeugbaren Testfälle so groß, dass häufig die Ressourcen nicht ausreichen, um alle diese Testfälle auszuführen. Neben dem Erzeugen von ungültigen Eingabedaten können ebenso ungültige Sequenzen von Funktionsaufrufen beim Fuzzing von Verhalten (engl. behavioral fuzzing) generiert werden, um bspw. Schwachstellen in Authentifizierungsmechanismen aufzudecken.

Benutzungsbasiertes Testen (engl. usage-based testing) ist ein Ansatz, bei der Tests für ein System basierend auf der tatsächlichen Benutzung des Systems hergeleitet werden. Der Grundgedanke ist hierbei, die vom Benutzer erlebte Qualität zu optimieren. Grundlage hierfür sind stochastische Benutzungsprofile, welche durch die Überwachung der Nutzung des Systems gewonnen werden. Hierzu wird zuerst die Nutzung des Systems aufgezeichnet, zum Beispiel, welche Funktionen aufgerufen werden oder wo Benutzer hin klicken. Anschließend wird aus der Aufzeichnung das Benutzungsprofil abgeleitet. Aus dem Benutzungsprofil werden dann Tests abgeleitet, welche die wahrscheinliche

Benutzung repräsentieren. Das Einsatzgebiet des benutzungsbasierten Testens ist der Regressionstest, wenn man sicherstellen will, dass die vom Benutzer erlebte Qualität nicht negativ beeinflusst wird.

II. Ein Ansatz zur Kombination von Usage-based Testing und IT-Sicherheitstesten

Wie oben dargestellt, sind auch fortschrittliche Techniken zum Aufdecken von IT-Sicherheits-schwachstellen immer noch sehr aufwändig. Um diesen Aufwand zu reduzieren, wobei gleichzeitig die Effektivität der IT-Sicherheitstests erhalten bleiben soll, schlagen wir einen Ansatz zur Kombination von Usage-based Testing mit IT-Sicherheitstests vor. Dieser Kombination liegt die Vermutung zugrunde, dass häufig verwendete Funktionalität eines Systems weniger fehlerträchtig ist als selten oder gar nicht verwendete Funktionalität. Für einen Angreifer ist die Häufigkeit der Nutzung einer bestimmten Funktionalität nicht relevant – in einer naiven Annahme sind alle Funktionalitäten potenziell gleich-wahrscheinlich für einen Angriff. Bei entsprechender Kenntnis des genutzten Systems, wie es bei gezielten Angriffen bspw. auf Industrieanlagen der Fall ist, sucht sich ein Angreifer u. U. sogar weniger genutzte Funktionalität als Angriffsziel aus, so dass Störungen infolge eines Angriffs weniger schnell entdeckt werden als bei häufiger genutzten Funktionen.

Da für das benutzungsbasierte Testen Benutzungsprofile aufzeichnet und als Grundlage für die Generierung von Regressionstests verwendet, ist die Intensität von Tests der häufig benutzten Funktonalität deutlich höher, zugleich liegen durch die verschiedenen Nutzereingaben größere Testdatenmengen vor als bei weniger häufig genutzten Funktionen. Unter der Annahme, dass diese weniger häufig genutzten und damit weniger intensiv getesteten Funktionen fehlerträchtiger, insbesondere für IT-sicherheitsrelevante Fehler sind, kann man sich die Nutzungsprofile zu Nutze machen, um diese Funktionen zu identifizieren und intensiveren Schwachstellentests zu unterziehen.

Dazu werden die Annahmen des benutzungsbasierten Testens invertiert. Anstatt Testszenarien zu erstellen, welche wahrscheinlich benutzt werden, werden solche erstellt die üblicherweise nicht von Benutzern ausgeführt werden. Hierbei werden sowohl die Nutzungshäufigkeiten von Funktionen, wie auch die verschiedenen Kombinationen von Benutzerdaten berücksichtigt. Dadurch können komplexe Szenarien generiert werden, die selten benutzte Funktionen einschließen, als auch solche für die nur wenige Nutzereingabedaten im Nutzungsprofil hinterlegt sind.

Bei entsprechender Anpassung eines Nutzungsprofils führt die Kombination bspw. mit Fuzzingtechniken dazu, dass komplexe Testszenarien auf selten genutzten Funktionalitäten aufgerufen werden. Diese Sequenzen dienen dann als Ausgangspunkt, um mit Hilfe des Nutzungsprofils die Stimuli auszuwählen, für die ungültige Eingabedaten mit Hilfe von Data Fuzzing generiert werden oder, im Fall von Behavioral Fuzzing, Modifikationen an der Sequenzen der Stimuli vorzunehmen.

III. Kontext: das Forschungsprojekt MIDAS

Im europäischen Forschungsprojekt MIDAS wird eine Cloud-basierte Plattform für automatische Tests von Webservices entwickelt. Dazu bietet die Plattform verschiedene Dienste zur modellbasierten Testfallgenerierung mit Techniken, unter anderem für benutzungsbasiertes Testen und IT-Sicherheitstesten bereit. Die Plattform bietet dabei auch Möglichkeiten, verschiedene Services zu orchestrieren.

Es ist geplant, den oben beschriebenen Ansatz auf der MIDAS-Plattform zu realisieren und mit Hilfe der Piloten zu evaluieren. Diese stammen aus den Bereich eHealth und Logistik. Das Forschungsprojekt MIDAS befindet sich im dritten Jahr und in der Evaluationsphase. Es läuft noch bis August 2015.

Acknowledgements. Diese Arbeit wird finanziert durch das EU FP7 Projekt MIDAS (no. 318786).