Lucerne University of
Applied Sciences and Arts

HOCHSCHULE
LUZERN

Informatik

Web Application Vulnerability Detection mit Java Agents

Themenbereiche: Automated Vulnerability Detection, Anomaly Detection, Java Agent

Studierende: Tobias Sturzenegger

Betreuungsperson: Oliver Hirschi

Experte: Lorenzo Medici

Auftraggebende: Swiss Infosec AG

Keywords: IT Security, Machine Learning, OWASP Top10, Web Security, Java

1. Aufgabenstellung

Die Swiss Infosec AG betreut Kunden bei technischen Projekten im Bereich der Webentwicklung. Bei der Überprüfung der Sicherheit von Webapplikationen setzt sie manuelle und automatische Überprüfungsmethoden ein. Diese Methoden kommen bei modernen Webapplikationsframeworks teilweise an ihre Grenzen. Die Swiss Infosec AG möchte untersuchen, ob es neue Möglichkeiten gibt, Sicherheitslücken in Webapplikationen zu erkennen.

Bei einem Kundenprojekt ist die Swiss Infosec AG auf eine dieser Möglichkeiten gestossen, bei der die Leistungsfähigkeit von Java Spring Webapplikationen auf produktiven Systemen mittels einem Java Agent gemessen wurde, um Kosten einsparen zu können. Dabei kam ein kommerzielles Tool zum Einsatz, dass die Performance von einzelnen Funktionen überwacht und Warnungen bei übermässigem Gebrauch von Ressourcen ausgibt.

Dies inspirierte die Swiss Infosec AG. Sie will überprüfen, ob produktive Java Webapplikationen in Echtzeit überwacht werden können. Dabei soll erkannt werden, ob Sicherheitsschwachstellen ausgenutzt werden.

Die Arbeit soll aufzeigen, ob bekannte und unbekannte Sicherheitslücken in Java Spring MVC Webapplikationen automatisch erkannt werden können, indem eine Applikation im produktiven Betrieb überwacht wird. Als Basis sollen die OWASP Top 10 Schwachstellen dienen. Daraus ergibt sich die Fragestellung: Wie können Java Webapplikation mittels einem Java Agent auf das Ausnutzen von Sicherheitsschwachstellen überwacht werden?

2. Ergebnisse

Gemäss dem definierten Vorgehen der Bachelorarbeit wurde im Verlauf der Arbeit die Demoapplikation HSLU Computer Shop entwickelt und mit verschiedenen Sicherheitslücken nach OWASP Top 10 versehen. Als Basis für die Demoapplikation kam das Java Spring MVC Framework zum Einsatz.

Mittels dem eigens programmierten Java Agent wurden mögliche Messpunkte eruiert und auf die Tauglichkeit getestet. Der Java Agent wurde anschliessend dazu gebracht, die Messungen über eine HTTP Schnittstelle an eine weitere Softwarekomponenten zu verschicken. Dieser Metrikserver wurde entwickelt, um die Messungen zu speichern, ein Machine Learning Modell zu trainieren und mit dem trainierten Modell Anomalien in den Messwerten zu detektieren. Um die Qualität des Modells zu evaluieren wurden diverse Testdatensätze generiert. Bei den drei getesteten Sicherheitslücken konnte dabei eine sehr gute Detektionsraten erreicht werden (F1 Score von 99.49%, 99.48% und 99.48%).

Somit konnte aufgezeigt werden, dass mittels einem Java Agent relevante Messdaten über eine Java Spring Web Applikation erhoben und diese mit dem Local Outlier Factor (LOF) Machine Learning Verfahren auf Novelties untersucht werden kann. Wenn das Modell mit genügend legitimen Anfragen trainiert wird, ist es fähig, illegitime Anfragen und somit auch das Ausnutzen von Sicherheitslücken, zu erkennen.

3. Lösungskonzept

Mittels einem Java Agent sollen relevante Messpunkte zur Applikation definiert und gemessen werden. Die laufenden Messdaten werden zuerst in einer Lernphase analysiert. In dieser Phase soll die Überwachungssoftware lernen, welche Messdaten konstant bleiben. Daraus soll ein Regelwerk (Modell) gebildet werden. Dieses soll im produktiven Betrieb Abweichungen erkennen und daraus Warnungen bilden.

Das Vorgehen ist wie folgt:

- Umsetzung einer verwundbaren Demoapplikation, welche die relevanten Sicherheitslückenkategorien aus OWASP Top 10 beinhaltet. Als Framework soll Spring MVC zum Einsatz kommen.
- Analyse der Java Agent Methode und der möglichen Messpunkte.
- Umsetzung einer Software, die Sicherheitslücken aus OWASP Top 10 erkennen kann.
- Dokumentation des Entwicklungsprozesses und Ausblick, ob und wie der gewählte Ansatz weiterentwickelt werden kann.

4. Spezielle Herausforderungen

Vor allem in der Anfangsphase war die Komplexität der verschiedenen Gebiete die grösste Herausforderung. Insbesondere das Java Spring Framework legte dem Vorhaben die meisten Steine in den Weg. Der Einsatz von Reflection vom Java Spring Framework verträgt sich schlecht mit dem Java Agent. So bleibt dies auch am Ende das grösste Problem des gewählten Ansatzes. Sobald vom Java Agent eine mit Reflection zur Laufzeit generierten Klasse überwacht wird, sind Fehlermeldungen und Abstürze die Folge.

5. Ausblick

Für weitere Untersuchungen bleibt das Stabilitätsproblem durch die Verwendung von Reflection durch das Java Spring Frameworks die grösste Herausforderung.

Schwierig ist weiter, dass die Applikation im Lernmodus exakt und umfassend getestet werden muss. Wird dies nicht gemacht, werden vom Machine Learning Modell Anomalien erkannt. Dies kann man aber auch als Chance sehen, so könnte der Ansatz auch beim Testing verwendet werden. Mittels Messungen kann überprüft werden, wie gross die Testabdeckung einer Software im Vergleich zum produktiven Betrieb ist.

Alles in allem ist der Ansatz vielversprechend interne Zustände einer Applikation zu messen, um Sicherheitslücken zu erkennen.