Airlock WAF ML Integration

Studiengang: MAS | Vertiefung: MAS Information Technology

In einer internen Forschung der Ergon Informatik AG wurde aufgezeigt, dass mittels unsupervised Machine-Learning Modellen, Anomalien in Web-Sessions detektiert werden können. In dieser Masterarbeit soll eine Architektur erarbeitet werden, wie dieses Know-How in das Produkt Airlock WAF integriert werden kann. Die Funktionsfähigkeit des gewählten Ansatzes soll mit einem Prototyp belegt werden.

Frank Meier

Ausgangslage

Airlock WAF ist ein Sicherheitsprodukt der Firma Ergon Informatik AG. Dabei handelt es sich um eine Web Application Firewall, welche als Appliance vor zu schützende Web-Applikationen geschaltet wird. Die WAF kontrolliert und steuert den Zugriff auf Ressourcen der Web-Applikation (z. B. erlaubt Zugriffe nur auf gewisse Ressourcen oder nur für authentisierte Benutzer). Weiter analysiert die WAF die HTTP Requests, prüft diese mittels eines regelbasierten Systems auf mögliche Angriffe und blockiert diese gegebenenfalls oder informiert ein Umsystem.
Um ein zusätzliches Mittel zur Angriffsdetektion zu erhalten, wurde in einem internen Forschungsprojekt analysiert, ob es möglich ist, mittels eines unsupervised Machine-Learning Ansatzes, Anomalien in HTTP Requests resp. Sessions zu erkennen. Der daraus entstandene PoC (Proof of Concept) hat aufgezeigt, dass dies möglich ist.

Zielsetzung

Ziel dieser Masterarbeit ist es eine Architektur zu erarbeiten, mit der das Wissen des PoC in das Produkt Airlock WAF integriert werden kann. Technologisch muss beantwortet werden, wie die ML-Modelle des PoC, welche auf Python basieren, mit der WAF Software, welche in C++ geschrieben ist, genutzt werden können. Die Anforderungen an die Performance könnte eine Implementation in C++ erfordern.
Es soll möglich sein die Modelle auf der WAF-Appliance trainieren zu können, um danach Web-Sessions im Betrieb zu qualifizieren. Basierend auf dieser Qualifikation sollen Aktionen, wie blockieren von Requests oder informieren eines Umsystems, definiert werden können.
Ein Prototyp soll am Ende belegen, dass die erarbeiteten Ansätze funktionieren.

Vorgehen

Nicht alle im PoC erarbeiteten Modelle haben einen signifikanten Beitrag zur Qualifizierung der Web-Sessions geleistet. Daher musste zuerst definiert werden, welche Modelle sich überhaupt für eine Produktisierung eignen.
Darauf folgte eine technologische Forschung, welche Möglichkeiten es gibt die Modelle mit dem WAF Code zu kombinieren. Z.B. welche Frameworks und Libraries es gibt, die Modelle in C++ abzubilden oder wie die Modelle in Python belassen und über eine Service-Schnittstelle genutzt werden könnten.
Danach wurden Architektur-Varianten gebildet und mittels eines Scoring-Models verglichen um die präferierte Lösung zu finden.

Resultate

Es wurde erkannt, dass es verschiedene Frameworks gibt, um die Lösung in C++ zu implementieren. Jedoch sind die Community und die Verfügbarkeit von ML-Modell-Implementationen einiges kleiner als in Python. Da die Flexibilität und Erweiterbarkeit der Lösung einen sehr hohen Stellenwert haben, wurde beschlossen die Modelle in Python zu belassen und diese über eine Service-Schnittstelle anzusprechen. Der Fokus des Prototyps lag dann darauf zu eruieren wie gut eine solche Lösung performt.
Dieser konnte belegen, dass die gewählte Lösung funktioniert, jedoch sank der maximale Request-Durchsatz der WAF auf ein unvertretbares Mass ab. Dabei stellten aber nicht die ML-Modelle, sondern die Service-Schnittstelle und die Datenaufbereitung im Python Code den Flaschenhals dar.

Ausblick

Da der Grundsatz der Architektur funktioniert, jedoch die Performance ungenügend ist, müssen für eine Produktisierung noch Optimierungen vorgenommen werden. Dies kann durch «Tuning» des Python Codes oder durch Verschiebung gewisser Datenaufbereitungsschritte in den C++ Teil möglich sein. Die Modell-Implementationen sollten aber in Python bleiben.