Code Signing Service

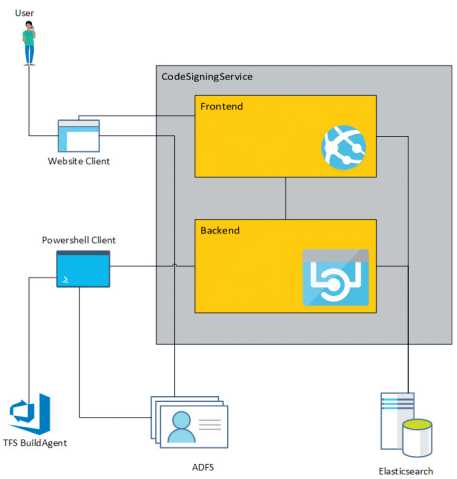
Studiengang: BSc in Informatik | Vertiefung: IT-Security
Betreuer: Gerhard Hassenstein, Dr. Annett Laube-Rosenpflanzer
Experte: Prof. Dr. Andreas Spichiger
Industriepartner: EDA, Bern

Um heute an eine gültige Signatur einer Software oder eines Scripts zu kommen, werden meist Token verwendet, welche lokal in den Computer eingesteckt werden müssen. Dieser Vorgang ist weder benutzerfreundlich noch sauber nachvollziehbar. Deshalb ist das Ziel dieses Projekts das Implementieren eines Code Signing Services, welcher das Signieren für mehrere Benutzer ermöglicht. Im Hintergrund soll jede ausgestellte Signatur nachvollziehbar in ein Log geschrieben werden.

Die Applikation besteht aus einem Web-Frontend, welches der Benutzer zum einfachen Signieren von Applikationen und Scripts verwenden kann. Die eigentliche Signatur wird anschliessend von einem Backend-Service mit dem gewünschten Zertifikat vorgenommen. Die fertig signierte Datei wird dann an den Benutzer zurückgegeben und gleichzeitig wird ein Log-Eintrag auf einem Elasticsearch-Cluster erstellt. Neben dem Frontend existiert auch ein Kommandozeilen-Tool, welches für geskriptete oder automatisierte Signaturen verwendet werden kann.

Traditionelles Code Signing

Das Code Signing Verfahren wird vor allem in Windows-Umfeld für das Signieren von Software, Libraries und auch Scripts verwendet. Mit einer gültigen Signatur auf einem Executable kann der ausführende Computer überprüfen, ob die Software auch wirklich von einer vertrauenswürdigen Quelle stammt und keine Veränderungen daran gemacht wurden.
Um eine gültige Signatur zu erstellen, muss der Signierende im Besitz eines privaten Schlüssels sein. Dieser Schlüssel muss besonders geschützt werden, da er alleine das Ausstellen von signierten Softwarepaketen ermöglicht. Dazu werden meistens Hardwaretoken verwendet, welche dazu designt wurden, ein Kopieren oder Auslesen des Schlüssels zu verunmöglichen.



Vorteile

Das Bereitstellen eines Service, welcher das Signieren der Software übernimmt, bietet einige Vorteile. So ist er für mehrere Benutzer gleichzeitig verfügbar – welches bei einem normalen Hardwaretoken schwierig umzusetzen ist, da es von Benutzer zu Benutzer händisch weitergereicht werden muss. Eine solche Implementation lässt sich auch besser in automatisch ablaufende Prozesse integrieren. Ein guter Anwendungsfall ist hier eine automatisch ablaufende Build-Pipeline, welche am Schluss die fertig kompilierte und getestete Applikation mit einer Signatur versieht – alles ohne, dass jemand von Hand ein Hardwaretoken einstecken müsste.

Marco Schmid