Safely and Flexibly Sandboxing Python Code in a C++ Runtime

Thema:
: Identification and Prevention of Common Programming Errors in Python running in a C++ Runtime Environment

Art:
: BA

BetreuerIn:
: Jürgen Hahn

BearbeiterIn:
: Robert Fent

ErstgutachterIn:
: Raphael Wimmer

ZweitgutachterIn:
: Christian Wolff

Status:
: in Bearbeitung

Stichworte:
: Sandboxing, Python3, C++, Security

angelegt:
: 2019-10-07

Antrittsvortrag:
: 2021-03-01

Hintergrund

Das Sandboxing von Applikationen ist eine Methode in der Software-Entwicklung, um die Kontexte, in denen Code ausgeführt werden kann, einzuschränken. Dadurch wird die Sicherheit einer Applikation vor externen Angreifern, Schadsoftware aber auch gegenüber unerwünschten Zugriffen auf Systemressourcen oder von anderen Applikationen gesteigert. Eine Art des Sandboxing ist die Verwendung von Skriptsprachen innerhalb von größeren Software-Systemen, wie Betriebssystemen oder *Game Engines*, die traditionell in C/C++ geschrieben sind. In diesem Benutzungskontext wird mithilfe der Skriptsprache das „Was?“ modelliert, das passieren soll, während das einbindende Software-System das „Wie?“ größtmöglich generisch regelt. Ein Beispiel aus dem Videospielkontext ist die Implementierung eines Projektils. Der Anwendungsentwickler „scriptet“ ein Projektil und definiert das Aussehen, die Größe, die Geschwindigkeit, den Schaden bei einem Treffer (das „Was“), usw., während die Engine die rechenaufwendigen Aspekte übernimmt, wie die Physiksimulation (z.B. Schwerkraft), Kollisionserkennung (Hit-Boxes), etc. (das „Wie“). Skriptsprachen werden für solche Anwendungsszenarien bevorzugt eingesetzt, da sie schnelle Iteration erlauben, leichter zu verwenden sind als C/C++ und auch Entwicklungen Dritter zulassen (Modding).

Zielsetzung der Arbeit

In dieser Arbeit soll systematisch untersucht werden, innerhalb welcher Szenarien es für Benutzer

möglich ist, aus einer Python-Sandbox innerhalb einer C++-Runtime auszubrechen bzw. die gesamte Anwendung zum Absturz zu bringen. Hierbei geht es primär um unbeabsichtigte „Attacken“, wie Bugs und undefiniertes Verhalten in den Scripts abzufangen, und weniger darum, die Runtime gezielt gegen böswillige Angriffe abzusichern. Anhand der evaluierten Szenarien soll im nächsten Schritt, ausgehend vom State of the Art des Sandboxing, eine Integration der Sandbox für Python3-Scripting in die C++-Runtime vorgenommen werden. In einem weiteren Schritt soll evaluiert werden, ob das Sandboxing hinreichend gegen die identifizierten Gefahren schützt.

Konkrete Aufgaben

- Aufbereitung von Literatur zum Thema (1 Woche)
- Identifizieren von Gefahren (1 Wochen)
- Integration von Sicherheitsmechanismen in die C++-Runtime (2 Wochen)
- Testen der Sicherheitsmechanismen und Bugfixing (1 Woche)
- Benutzerevaluation der Sandbox (1 Woche)
- Finalisierung der schriftlichen Ausarbeitung (2 Wochen)

Erwartete Vorkenntnisse

- gute Programmierkenntnisse in Python 3 (von Vorteil: sehr gute Kenntnisse)
- rudimentäre Programmierkenntnisse in C/C++ (von Vorteil: gute Kenntnisse)
- technisches Interesse und sorgfältige Arbeitsweise werden vorausgesetzt
- von Vorteil: Erfahrung mit Security-Aspekten und Sandboxing

Weiterführende Quellen

1. https://docs.unrealengine.com/en-US/Engine/Blueprints/index.html
2. Sewell, B. (2015). Blueprints Visual Scripting for Unreal Engine. Packt Publishing Ltd.
3. https://de.wikipedia.org/wiki/Skriptsprache
4. Prevelakis, V., & Spinellis, D. (2001, June). Sandboxing Applications. In USENIX Annual Technical Conference, FREENIX Track (pp. 119-126).
5. Ansel, J., Marchenko, P., Erlingsson, Ú., Taylor, E., Chen, B., Schuff, D. L., ... & Yee, B. (2011, June). Language-independent sandboxing of just-in-time compilation and self-modifying code. In ACM SIGPLAN Notices (Vol. 46, No. 6, pp. 355-366). ACM.