Hochschule Luzern
Informatik
Frühlingssemester 2024
SECPROJ

Sicherheitsprojekt

Digital Rights Management (DRM) Systeme in Videospielen

Analyse von Obfuskationstechniken in Videospielen

Autor
Fabio Schmidt, Informatik – Information & Cybersecurity

Betreuender Dozent
Tim Blazytko

Abgabetermin
01. Februar.2024

Danksagung

Vielen Dank an meinen Betreuer Dr. Tim Blazytko, der mein Projekt im Rahmen des SECPROJ-Moduls sorgfältig überwacht und fachkundig betreut hat. Sein Einblick und sein Fachwissen haben die Qualität und den Erfolg meiner Arbeit verbessert. Auch bin ich für seine konstruktive Kritik und seine Unterstützung sehr dankbar.

Dr. Blazytko hat sich immer die Zeit genommen, meine Fragen zu beantworten oder mir wertvolle Ratschläge zu geben. Sein professioneller Ansatz und sein Engagement für das Projekt hat mich inspiriert und motiviert.

Abstract

Das Projekt behandelt das Thema der Code-Obfuskation in Digital Rights Management (DRM)-Systemen. Die Arbeit ist in verschiedene Phasen gegliedert: In einem ersten Schritt geht es darum, sich mit der Thematik vertraut zu machen. Eine Analyse zu Apples Fairplay Komponente wurde gelesen und in eigenen Worten zusammengefasst, wobei vorallem auf die wichtigsten Punkte aus dem Artikel Bezug genommen wurde. In einem weiteren Schritt wurde das Programm Tigress für die Anwendung von Obfuskationstechniken wie Control Flattening, Opaque Predicates und Encode Arithmetic genutzt.

Darauffolgend wurde neues Wissen zu "auf virtuelle Maschinen basierende" (VM-based) Obfuskation angeeignet. Eine Internetrecherche wurde zu der Thematik vorgenommen und das Wichtigste zu Text verfasst. Dies diente als Vorbereitung für die Analyse der vom Dozenten bereitgestellten Binärdatei.

In einem letzten Schritt ging es um die Analyse der EasyAntiCheat Komponente. Diese ist als Teil vom Easy Anti-Cheat Programm anzusehen, das für die Erkennung von Betrüger in Onlinespielen zuständig ist. Auch war es zuletzt vorgesehen, zufällig gewählte Funktionen in ebendieser Binärdatei auszusuchen und anhand des gewonnenen Wissens beurteilen zu können, ob es sich dabei um obfuskierten oder normalen Code handelt.

Inhaltsverzeichnis

1. Ausgangslage

Digital Rights Management (DRM) Systeme übernehmen die digitale Rechteverwaltung in Multimedia-Software und Libraries. Um die Analyse durch Reverse Engineering zu erschweren, machen diese oft Gebrauch von sogenannten Code- Obfuskationstechniken, beispielsweise Control-flow Flattening (Collberg, o. J.) oder Virtualisierung (Blazytko, 2021). Waehrend Obfuskationstechniken in der Praxis haeufig zu finden sind (beispielsweise in Malware), finden sich in kommerziellen DRM-Systemen meist die stärkeren und interessanteren Varianten.

2. Ziel(e) der Arbeit

Die Arbeit soll als Mittel zur Wissensaneignung dienen. Es ist vorgesehen, dass der Autor sich mit der Thematik auseinandersetzt und schlussendlich mehrere Analyseberichte zu den ausgesuchten Proben präsentiert.

2.1 Definierte Ziele

Das Ziel der Arbeit sind die systematische Auseinandersetzung und Analyse von Obfuskationstechniken in Videospielen. Dabei liegt der Schwerpunkt auf

- der Auseinandersetzung und Wissensaneignung von Code-Obfuskationstechniken
- der Anwendung des gesammelten Wissens im Rahmen einer Analyse eines kommerziellen DRM-Systems

2.2 Projektleistungen

Aus der Arbeit heraus sollen schlussendlich mehrere Analyseberichte entstehen. Diese sollen die ausgesuchten Proben näher erklären und einen Einblick in dessen Funktionsweise und Code geben. Der Fokus während der Analyse wurde vermehrt auf die Präsentierung und Erläuterung von obfuskierten Code gelegt. Speziell interessante Codeteile respektive spezifische Funktionen wurden aus den Binärdateien näher untersucht und die dadurch entstandenen Gedankengänge niedergeschrieben. Während der Arbeit wurde der persönliche Lernforschritt stets dokumentiert.

3. Methodik

Zu Beginn soll eine intensive Literaturrecherche als erste Vorgehensweise dienen. Mehrere Artikel, die sich mit dem Thema der Code-Obfuskation auseinandersetzen, sollen gelesen und in eigenen Worten zusammengefasst werden. In einem weiteren Schritt sollen mehrere Musterexemplare untersucht werden, um sich ein grundlegendes Wissen über Code-Obfuskation-Techniken und deren Variationen anzueignen. Zum Schluss soll dann so viel wie möglich über das ausgesuchte DRM-System und die verwendete Obfuskierung eines modernen Videospiels mittels Reverse Engineering erfahren werden.

Die Untersuchungen sollen anhand von aus dem Reverse Engineering bekannten Programmen wie Ghidra, IDA oder Binary Ninja (oder ähnliche) und Techniken erfolgen. Es soll vermehrt auch programmübergreifend gearbeitet werden, sodass bei Unstimmigkeiten zwischen den Programmen, trotzdem ein nachvollziehbares Ergebnis geliefert werden kann. Auch ergibt sich so automatisch auch eine Erfahrungssammlung mit mehreren bekannten Programmen. Der Gesamtprozess, wie auch die Teilprozesse sollen stetig dokumentiert werden in Bezug auf analysierte Obfuskationstechniken, verwendete Binaries sowie Erkenntnisse ueber deren Funktionsweise.

Platformen wie Discord wurden für die Kommunikation und GitHub für das Teilen der Analysen aus der Heuristik genutzt.

3.1 Vorgehensplan

Vor dem Beginn des Projekts wurde ein Vorgehensplan ausgearbeitet, der als Grundlage für die Arbeit an das Projekt diente.

1. *Read: https://nicolo.dev/en/blog/fairplay-apple-obfuscation/*
   - *1.1. Understand the fundamentals of the article*
   - *1.2. 1.2 Summarize the article in own words.*

2. *Install and understand the Tigress Obfuscation Software*
   - *2.1. Get a simple C program and obfuscate it with tigress. e.g. https://github.com/mrphrazer/r2con2021_deobfuscation/blob/main/samples/src/fib.c*
   - *2.2. Look at the compiled binary with e.g. Ghidra*
     - *2.2.1. Focus on the following obfuscation techniques (one per one):*
       - *Control Flattening*
       - *Opaque Predicates*
       - *Encode Arithmetic*

3. *Read: https://synthesis.to/2021/10/21/vm_based_obfuscation.html*
   - *3.2 Summarize the article in own words.*
   - *3.3 Analyze the vm_base.bin binary.*

4. *Analyze the EasyAntiCheat.sys binary.*
   - *4.1. Focus on some of its functions.*
   - *4.2. Focus on the whole binary and try to "quickly" spot (with the acquired knowledge) obfuscated code.*

4. Schlussfolgerung und Erfahrung

Während des gesamten Projekts konnte neues Wissen angeeignet werden. Anhand der Analyse des Fairplay Artikels konnten erste Erkenntnisse zu gängigen Obfuskationstechniken gesammelt werden. Durch die Nutzung der Tigress Software wurde gelernt, mit ebendiesem weitverbreiteten Programm umzugehen. Auch wurden mit Tigress normale Codebeispiele zu obfuskiertem Code umgewandelt, womit man so Code Obfuskation auch direkt in Praxis untersuche und typische Charakteristika analysieren konnte.

In der Phase zur Virtualisierung wurde spezifisch die VM-based Obfuskationstechnik näher kennengelernt. Durch die Auseinandersetzung mit der dazugehörigen Binärdatei konnte neues Wissen zur Thematik gesammelt und zusätzlich auch neue Erkenntnisse in der Assemblersprache gewonnen werden.

Die EasyAntiCheat.sys Datei präsentierte ein Beispiel von kommerzieller Software. Diese diente als Mittel, um auch praxisnahe Code Obfuskationstechniken genauer anschauen zu können. Auch wurde in einem letzten Schritt versucht, Funktionen aus der Binärdatei zufällig zu wählen und zu bestimmen, ob es sich bei diesen um obfuskiertem Code handelte oder nicht. Diese Übung sollte als Abschluss dienen, um die erlernten Fähigkeiten in Praxis umzusetzen.

Im Allgemeinen konnte während der gesamten Arbeit der Umgang mit bekannten Reverse Engineering Programmen (wie IDA64, Ghidra, BinaryNinja...) erlernt und neue Kenntnisse zu Programmiersprachen wie C und Assembler gewonnen werden. Auch wurde neues Wissen über die Funktionsweise und Analyse von obfuskiertem Code angesammelt und bereits bestehendes erweitert.

5. Quellenverzeichnis

5.1 Literaturverzeichnis

- Blazytko, T. (2021, Oktober 21). Writing Disassemblers for VM-based Obfuscators. Synthesis.To. https://synthesis.to/2021/10/21/vm_based_obfuscation.html
- Collberg, C. (o. J.). Flatten. Tigress.Wtf. Abgerufen 25. November 2023, von https://tigress.wtf/flatten.html

5.2 Abkürzungsverzeichnis

- **DRM** – Digital Rights Management