**Projekt PE-Parser**

Datum Begin: 27.03.2023

Datum Ende : 07.04.2023

**Index**

1. Deckblatt & Index
2. Einleitung
   1. Projektumfeld
   2. Projektziel
   3. Projektbegründung
   4. Projektschnittstellen
   5. Projektabgrenzung
3. Projektplanung
   1. Projektphasen
   2. Abweichung vom Projektantrag
   3. Ressourcenplanung
   4. Entwicklungsprozess
4. Analysephase
   1. Ist-Anal
   2. Wirtschaftlichkeitsanalyse
   3. Make or Buy-Entscheidung
   4. Projektkosten

**Einleitung**

Projektumfeld:

* Betrieb: Berufsschule
* Mitarbeiterzahl: 2
* Auftraggeber: Hr. Rath (Cracker)

Projektziel:

* Thema: Ist die Erstellung eines sowohl schlanken als auch leistungsfähigem und dennoch flexiblem Parsers für „Portable Executable“ Dateien, welcher als Produkt auf den Markt zu bringen ist.
* Ziel: Ein schlanker effizienter und erweiterbarer (PE-)Parser

Projektbegründung

* Warum ist das Projekt sinnvoll?: das Endprodukt soll eine leicht erweiterbares Programm sein, welches bsp die Entwicklung von aufbauenden Programmen vereinfacht.
* Was ist die Motivation hinter dem Projekt?: Die ursprüngliche Idee/Motivation war es einen Crypter zu bauen, da dies jedoch sehr komplex und aufwendig werden kann, haben wir beschlossen zuerst den Notwendigen „Unterbau“ zu Programmieren.

Projektschnittstellen

* Verbindung zu Externer Technik: Windows API Sets (native)
* Wer genehmigt das Projekt bzw. stellt Mittel zur Verfügung?: Der Ausbildungsbetrieb stellt alle nötigen Mittel zur Verfügung.
* Zielgruppe: Technik Begeisterte, Entwickler von Pentestprogrammen und Forensiker
* Inspektor: Hr. Rath

Projektabgrenzung

* Was ist explizit nicht Teil des Projekts (insb. bei Teilprojekten): Das Projekt beschäftigt sich ausschließlich mit dem Parsen von PE-Files, nicht jedoch mit dem Manipulieren dieser.

**Projektplanung**

Projektphasen:

1. Vorbereitung:
   1. Ideen Sammlung | Brainstorming
   2. Einholen der Genehmigung zum Projekt
   3. Zeitplanung
   4. Protokoll Rahmen erstellen
2. Ausarbeiten:
   1. Entwicklung des Parsers
   2. Entwicklung des „Graphical User Interfaces“
3. Überprüfung:
   1. Schreiben Des Protokolls
   2. Funktionstest
4. Abnahme:
   1. Hr. Rath.

Abweichungen vom Projektantrag:

• Abweichungen: Zeitmangel

Ressourcenplanung:

• Detaillierte Planung der benötigten Ressourcen (Hard-/Software, Räumlichkeiten usw.):

* Für die Entwicklung des Programms (samt GUI) werden 2 Computer mit diversen Programmen und einem Windows Betriebssystem benötigt.
* Entsprechende Räumlichkeiten

• Ggfs. sind auch personelle Ressourcen einzuplanen (z.B. unterstützende Mitarbeiter).

• Hinweis: Häufig werden hier Ressourcen vergessen, die als selbstverständlich angesehen werden (z.B. PC, Büro).

Entwicklungsprozess:

• Welcher Entwicklungsprozess wird bei der Bearbeitung des Projekts verfolgt (z.B. Wasserfall, agiler Prozess)?

**Analyse**

• Ist- Analyse:

* C++ Programm/Library (dynamisch)
  + Kann in dll und exe compiled werden
  + PE-Header können extrahiert werden
  + Signaturen (Magic values/signatures) können geprüft werden (valid PE)
  + Python ctypes compatibel
  + Designtechnik:
    - Template Funktionen (Bsp. openFile, getImageHeader)
    - Universelle Datenstruktur für Funktionsparametertyp und rückgabetyp.
    - „Concepte“ für Custom Datenstruktur
    - Compiletime Wrapper Funktionen für Offsets

• Wirtschaftlichkeit

Make or Buy?

• Anwendungsfälle

* Die Library könnte als unterbau für aufbauende Projekte dienen bsp. Crypter, Reverse-Ingeniering tool
* Das Programm (mit GUI) könnte für Forensische Zwecke verwendet werden können

• Qualitätsanforderungen

Die Library / source Library, soll in möglichst schlankem Assembly Code enden (mit Hilfe von Meta-programming).

Die Architectur soll möglichst Homogen sein.

Der Code sollte wartbar sein.

**Entwurfsphase**

• Ziehlplattform:

* C++ kompatibler Sourcecode
* Windows
* Python (ctypes) kompatible anbindung
* QT5

• Architektur Design:

* Native Windows API Sets
  + Da diese Anwendung(PE-Parser) sowieso nur für Windows konzipiert ist, sind jegliche Abstraktionsschichten unnötig.
* PyQT
  + Da Python intuitiv und einfach ist, wird die Entwicklung des GUIs auf Python QT5 ausgelagert.

• Entwurf der Benutzeroberfläche:

* Es wird QT 5 mit Python für die Erstellung des GUIs verwendet.
* •••

• Datenmodell:

* Interne Classenstruktur für homogene Parameter Rückgabetypen.

|  |
| --- |
| Pe-parser-base |
|  |

• Geschäftslogik (Programm ablauf)

INDEX:

1. Name
2. Projektziel

1 Name: Pe-Parser

2 Projektziel:

Am Ende dieses Projektes soll eine voll funktionsfähige Library\* entstehen, welche in der Lage ist ausführbare Dateien im Portable Executable Format zu validieren und zu masern.

Der Pe-Parser soll als dynamische/statische Library (.dll/.obj) und als c++ Source code (.h/.cpp) verwendbar sein.

Python Funktionsverwaltung mit GUI.

<3

3 Projektbegründung:

Der Pe-Parser soll bei „source only“ verwendung möglichst schlanken Assembly erzeugen, mit Hilfe von compiletime Metaprogramming\*\*. Der Parser soll homogen, strukturiert, erweiterbar, sein. Dies soll mit hilfe von Compiletime-wrappern\*\*, welche bereits bestehende (native) Windows API (sets) wrappen, bewerkstelligt werden.

Es soll ein Python zur darstellung und Fehlerbehandlung verwendet werden.

\*\* Compiletime-wrapper -> sind einfache/ komplexe “Hüllen“-Konstrukte, welche während der Kompilier-zeit aufgelöst werden (Meta-programming), und im späteren Assembly code nicht mehr existieren.

4 Technische Schnitstellen:

Es sollen native Windows API Sets verwendet werden.

5 Ziel Nutzer-gruppe:

Zu den Ziel-gruppen gehören Anwendungs-Entwickler.

6 Projektabgrenzung:

Dieses Projekt bietet nur die Möglichkeit zur Analyse aber nicht zum crypten.