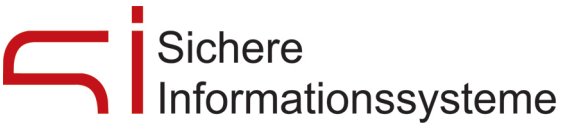


Reverse Engineering (REV3)

UE 02 – Statische Analyse – Protokoll

Jakob Mayr

WS 2023/2024

Einleitung

...

Aufgabe 1 - Statische Analyse Windows

Erstellen der Dateien

In Aufgabe 1 ist eine Anwendung, welche "infected" auf `stdout` ausgibt in `c` zu schreiben und mit Visual Studio auf 4 verschiedene Varianten zu bauen.
Die vier Varianten mit deren Eigenschaften und Unterschieden (Quelle: ChatGPT):

1. **Static Release (/MT)**:
   (a) Laufzeitbibliothek: Statisch
   (b) Debug-Informationen: Nein
   (c) Eigenschaften:
      i. Die Laufzeitbibliothek wird in die ausführbare Datei eingebettet.
      ii. Größere Dateigröße, da der Code der CRT (C Runtime Library) direkt in die Anwendung eingefügt wird.
      iii. Keine Abhängigkeit von DLLs (Dynamically Linked Libraries) zur Laufzeit.
      iv. Optimiert für Geschwindigkeit und nicht für das Debugging.
2. **Static Debug (/MTd)**:
   (a) Laufzeitbibliothek: Statisch
   (b) Debug-Informationen: Ja
   (c) Eigenschaften:
      i. Ähnlich wie /MT, aber mit zusätzlichen Debug-Informationen und weniger Optimierungen.
      ii. Erleichtert das Debugging, da Variablen leichter überwacht werden können.
      iii. Größerer Speicherbedarf und langsamere Ausführung im Vergleich zur Release-Version.
3. **Dynamic Relase (/MD)**:
   (a) Laufzeitbibliothek: Dynamisch
   (b) Debug-Informationen: Nein
   (c) Eigenschaften:
      i. Verlinkt dynamisch mit den CRT-DLLs (Z.B. msvcrXXX.dll).
      ii. Kleinere Dateigröße, da die Laufzeitbibliothek nicht eingebettet ist.
      iii. Erfordert, dass die CRT-DLLs zur Laufzeit verfügbar sind.
      iv. Optimiert für Geschwindigkeit.
4. **Dynamic Debug (/MDd)**:
   (a) Laufzeitbibliothek: Dynamisch
   (b) Debug-Informationen: Ja
   (c) Eigenschaften:
      i. Ähnlich wie /MD, aber mit zusätzlichen Debug-Informationen und weniger Optimierungen.
      ii. Erleichter das Debugging.
      iii. Erfodert, dass die Debug-Version der CRT-DLLs zur Laufzeit verfügbar ist.

Unterschiede:

1. **Statische vs. Dynamische Verlinkung**: /MT und /MTd binden die Laufzeitbibliothek statisch ein, wodurch die ausführbare Datei unabhängig von externen DLLs ist. /MD und /MDd verlinken dynamisch und erfordern, dass die entsprechenden DLLs zur Laufzeit vorhanden sind.

2. **Debug vs. Release**: Die Debug-Optionen (/MTd und /MDd) enthalten zusätzliche Debug-Informationen und sind nicht so stark optimiert wie die Release-Optionen (/MT und /MD), was das Debugging erleichtert, aber die Leistung beeinträchtigen kann.

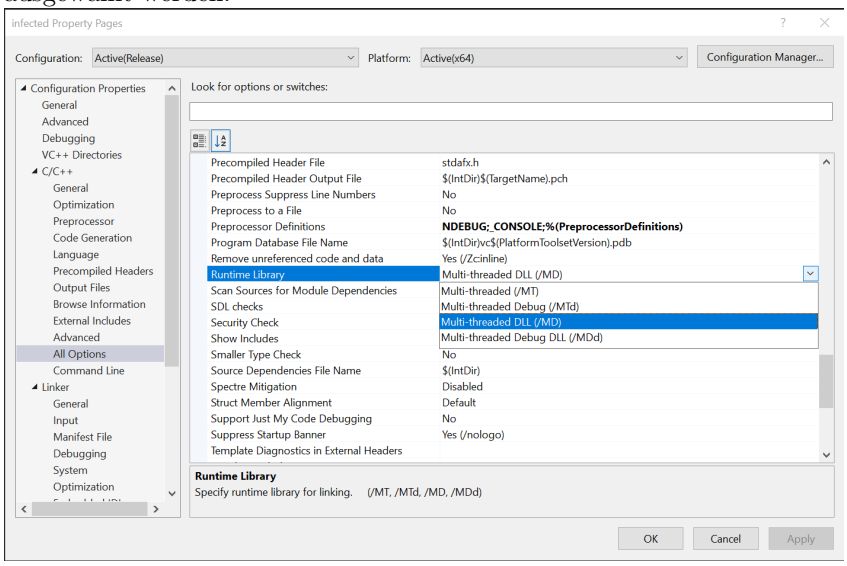
Der Programm-Code für die gewünschte Executable:

```c
#include <stdio.h>

int main() {
  printf("infected");
  return = 0;
}

```

In den Einstellungen des Visual Studio Projekts kann die gewünschte Variante für die "Runtime Library" ausgewählt werden:



Die Files nach erstellen, hier ist direkt ersichtlich, dass beide statischen Varianten größer sind (Length), da sich der Code der Libraries im File befindet:

```
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases> dir


    Directory: C:\Users\admin_sin\Desktop\rev3-s2210239021\releases


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        24.10.2023     16:15          10752 infected-md.exe
-a----        24.10.2023     16:16          14336 infected-MDd.exe
-a----        24.10.2023     16:10         139264 infected-mt.exe
-a----        24.10.2023     16:14         376832 infected-mtd.exe


PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases>
```

Analyse der Dateien

Um die erzeugten Executables zu analysieren sollen zumindest die Tools `dumpbin` und `strings` verwendet werden.

dumpbin

`dumpbin` ist ein Command-Line Tool, zur Verfügung gestellt von `Microsoft Visual Studio` um PE-Files (Portable Executable) zu analysieren. Es können beispielsweise die flags **/DEPENDENTS**, **/EXPORTS**, **/HEADERS** oder **/ALL** verwendet werden, um alle vom Tool lieferbaren Informationen zu erhalten.

Folgende Aufrufe zeigen die Ausgabe ohne Parameter:

```
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases> dumpbin.exe .\infected-mt.exe
Microsoft (R) COFF/PE Dumper Version 14.36.32535.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\infected-mt.exe

File Type: EXECUTABLE IMAGE

  Summary

        2000 .data
        2000 .pdata
        B000 .rdata
        1000 .reloc
        1000 .rsrc
       15000 .text
        1000 _RDATA
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases>
```

(a) dumpbin mt-file

```
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases> dumpbin.exe .\infected-mtd.exe
Microsoft (R) COFF/PE Dumper Version 14.36.32535.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\infected-mtd.exe

File Type: EXECUTABLE IMAGE

  Summary

        3000 .data
        4000 .pdata
       15000 .rdata
        1000 .reloc
        1000 .rsrc
       43000 .text
        1000 _RDATA
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases>
```

(b) dumpbin mtd-file

```
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases> dumpbin.exe .\infected-md.exe
Microsoft (R) COFF/PE Dumper Version 14.36.32535.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\infected-md.exe

File Type: EXECUTABLE IMAGE

  Summary

        1000 .data
        1000 .pdata
        1000 .rdata
        1000 .reloc
        1000 .rsrc
        1000 .text
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases>
```

(c) dumpbin md-file

```
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases> dumpbin.exe .\infected-MDd.exe
Microsoft (R) COFF/PE Dumper Version 14.36.32535.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\infected-MDd.exe

File Type: EXECUTABLE IMAGE

  Summary

        1000 .data
        1000 .pdata
        1000 .rdata
        1000 .reloc
        1000 .rsrc
        2000 .text
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases>
```

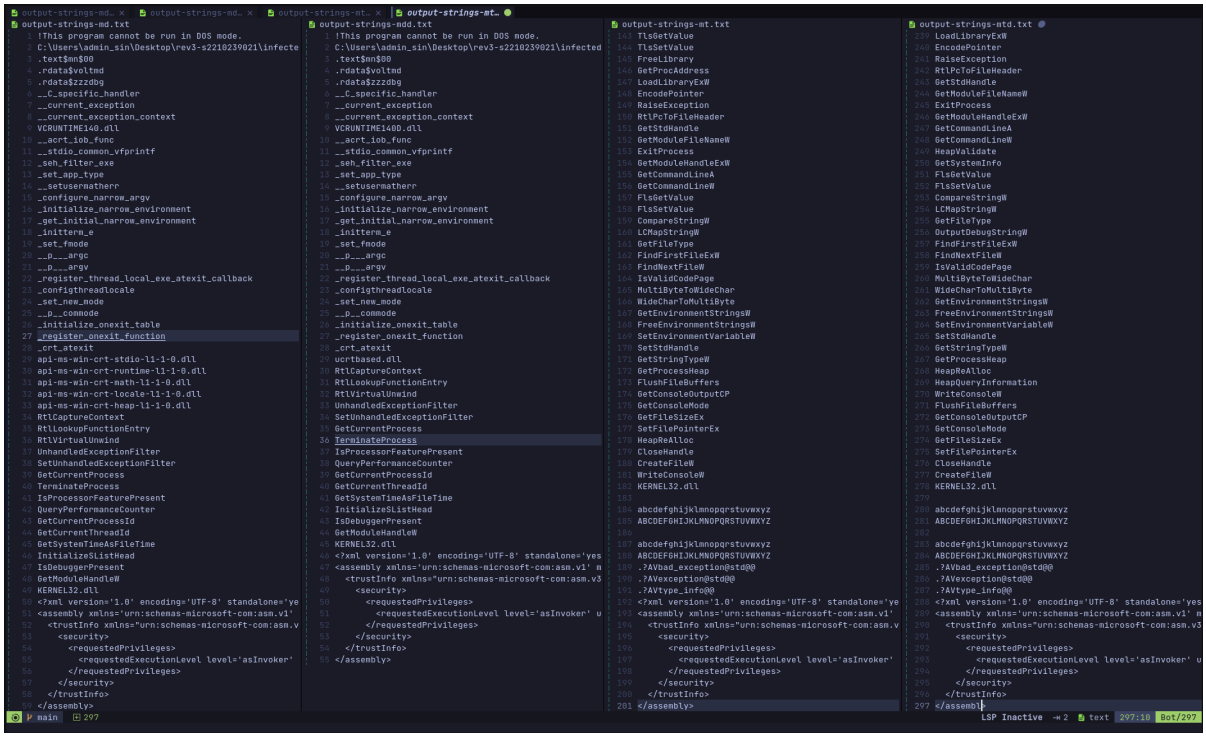
(d) dumpbin mdd-file

Figure 1: All infected.exe files analysed with dumpbin

Die Debug-Versionen (infected-MDd.exe und infected-mtd.exe) haben tendenziell größere .text Sektionen als ihre entsprechenden Release-Versionen, da sie zusätzliche Debug-Informationen enthalten.
Die statisch verlinkten Versionen (infected-mt.exe und infected-mtd.exe) haben größere .text und .rdata Sektionen im Vergleich zu den dynamisch verlinkten Versionen, da die C Runtime Library direkt in die ausführbare Datei eingebettet ist.
Die dynamisch verlinkten Versionen (infected-md.exe und infected-MDd.exe) sind im Allgemeinen kleiner, weil sie zur Laufzeit externe DLLs verwenden.

strings

Der Befehl strings wird verwendet, um alle druckbaren Zeichenfolgen in einer Binärdatei zu extrahieren und anzuzeigen. Eine Zeichenfolge in diesem Kontext ist eine Sequenz von druckbaren Zeichen, die typischerweise durch Null-Bytes (\0) oder durch nicht-druckbare Zeichen getrennt ist. Dieses Tool ist nützlich, um Text, Pfade, URLs, Versionen und andere Informationen aus ausführbaren Dateien, Bibliotheken oder beliebigen Binärdateien zu extrahieren.
Folgende zwei screenshots zeigen sowohl den Anfang als auch das Ende der 4 verscheidenen Files:



Fragen

1. Welche Imports werden verwendet?
   Durch den Aufruf von `dumpbin /IMPORTS <PE-filename>` können die Imports ermittelt werden:

```
PS C:\Users\Quickemu\repos\REV3\UE02\rev3-s2210239021\windows\releases> dumpbin /IMPORTS .\infected-mt.exe
Microsoft (R) COFF/PE Dumper Version 14.37.32825.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\infected-mt.exe

File Type: EXECUTABLE IMAGE

  Section contains the following imports:

    KERNEL32.dll
             140016000 Import Address Table
             14001FE98 Import Name Table
                     0 time date stamp
                     0 Index of first forwarder reference

                         4F5 RtlCaptureContext
                         4FD RtlLookupFunctionEntry
                         504 RtlVirtualUnwind
                         5E6 UnhandledExceptionFilter
                         5A4 SetUnhandledExceptionFilter
                         232 GetCurrentProcess
                         5C4 TerminateProcess
                         3A8 IsProcessorFeaturePresent
                         470 QueryPerformanceCounter
                         233 GetCurrentProcessId
                         237 GetCurrentThreadId
                         30A GetSystemTimeAsFileTime
                         38A InitializeSListHead
```

   (a) infected-mt.exe
      i. KERNEL32.dll

   (b) infected-mtd.exe
      i. KERNEL32.dll

   (c) infected-md.exe
      i. VCRUNTIME140.dll
      ii. api-ms-win-crt-stdio-l1-1-0.dll
      iii. api-ms-win-crt-runtime-l1-1-0.dll
      iv. api-ms-win-crt-math-l1-1-0.dll
      v. api-ms-win-crt-locale-l1-1-0.dll
      vi. api-ms-win-crt-heap-l1-1-0.dll
      vii. KERNEL32.dll

   (d) infected-MDd.exe
      i. VCRUNTIME140D.dll
      ii. ucrtbased.dll
      iii. KERNEL32.dll

2. Welche Sektionen werden verwendent?
   Durch den Aufruf von `dumpbin <PE-filename>` können die Sektionen ermittelt werden:

```
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases> dumpbin.exe .\infected-md.exe
Microsoft (R) COFF/PE Dumper Version 14.36.32535.0
Copyright (C) Microsoft Corporation.  All rights reserved.


Dump of file .\infected-md.exe

File Type: EXECUTABLE IMAGE

  Summary

        1000 .data
        1000 .pdata
        1000 .rdata
        1000 .reloc
        1000 .rsrc
        1000 .text
PS C:\Users\admin_sin\Desktop\rev3-s2210239021\releases>
```

   (a) infected-mt.exe
      i. .text
      ii. .rdata
      iii. .data
      iv. .pdata
      v. _RDATA
      vi. .rsrc
      vii. .reloc

   (b) infected-mtd.exe
      i. .text
      ii. .rdata
      iii. .data
      iv. .pdata
      v. _RDATA
      vi. .rsrc
      vii. .reloc

   (c) infected-md.exe
      i. .text
      ii. .rdata
      iii. .data
      iv. .pdata
      v. .rsrc
      vi. .reloc

   (d) infected-MDd.exe
      i. .text
      ii. .rdata
      iii. .data
      iv. .pdata
      v. .rsrc
      vi. .reloc

3. Was kannst du über die*den Author*in sagen? Ein direkter Author ist nicht erkennbar. Allerdings könnte man auf "admin_sin" schließen, da das File für die "Program Database" (infected.pbd) im Pfad den User nennt:

```
1        Debug Directories
2
3       Time Type         Size       RVA  Pointer
4       -------- ------- -------- -------- --------
5       6537D1B0 cv             66 00003464     2264    Format: RSDS, {6F72B84A-D687
        -4743-A104-E88EF40E7E96}, 4, C:\Users\admin_sin\Desktop\rev3-s2210239021\
        infected\x64\Release\infected.pdb
6       6537D1B0 feat           14 000034CC     22CC    Counts: Pre-VC++ 11.00=0, C/C
        ++=30, /GS=30, /sdl=1, guardN=29
7       6537D1B0 coffgrp       284 000034E0     22E0    4C544347 (LTCG)
8       6537D1B0 iltcg           0 00000000        0
9
10
```

Dies ist in allen Files gleich herauszulesen.

4. Was kannst du über die Umgebung, in der das Sample erzeugt wurde, sagen?
Auskünfte über die Umgebung können durch die Compiler-Version, eingebettete Ressourcen oder Abhängigkeiten und der weiteren gegeben werden.

2. Aufgabe - Statische Analyse Linux

create file

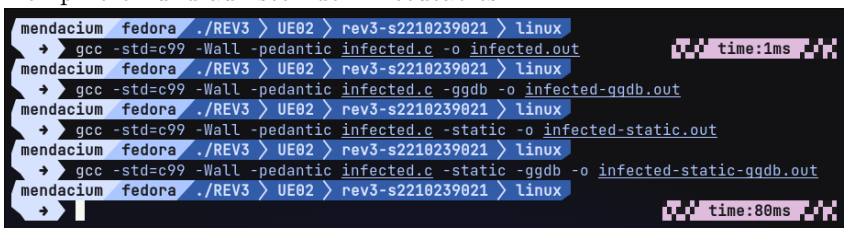
Gleicher code wie zuvor:

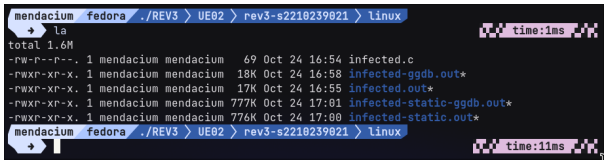
```c
#include <stdio.h>

int main() {
  printf("infected");
  return = 0;
}

```

Kompilieren und auflisten der Executables:





References

[1] *The Official Radare2 Book*, [Online; abgerufen im Oktober 2023], `https://book.rada.re/`.