

REVERSE ENGINEERING MIT GHIDRA

Modul IT-Forensik

Marius Eggert
21.04.2021

<EINSCHUB>

Projektideen & Gruppen

EINSCHUB

Projektideen

- Themenidee Autoschlüssel

- Analyse der Keyless go Kommunikation
- Aktuelle Gruppengröße: 1
- Software Defined Radio kann beschafft werden
(Lieferzeit ist aber noch nicht bekannt)



- Themenidee Alexa

„Alexa, wer ist der Mörder? – Wie man mit Alexa einen Mord aufklären kann“

- Untersuchung der gespeicherten Daten in einem Alexa System
(Realisiert mit einem Raspberry Pi)
- Aktuelle Gruppengröße: 3
- Raspberry Pis sollten an der HS-RM vorrätig sein



<https://tutorials-raspberrypi.de/raspberry-pi-amazon-alexa-deutsch-installieren/>

- Gruppe: Firmware-Virtualisierung
(Virtualisierung unbekannter Firmware [inkl. Schwachstellenanalyse])
 - Noch kein Target?
 - Aktuelle Gruppengröße: 2



Virtualisierung von IoT Firmware (<https://www.youtube.com/watch?v=ALn0hUxNsZl>)



Angr Framework ³



Qiling Framework ⁴

¹ <https://www.zerodayinitiative.com/blog/2020/5/27/mindshare-how-to-just-emulate-it-with-qemu>

² <https://www.zerodayinitiative.com/blog/2020/2/6/mindshare-dealing-with-encrypted-router-firmware>

³ <https://github.com/angr/angr>

⁴ <https://github.com/qilingframework/qiling>

</EINSCHUB>

Projektideen & Gruppen

GHIDRA

Einführung

- Software Reverse Engineering Framework
- Von der NSA entwickelt
 - 2017 auf WikiLeaks „enthüllt“
 - 2019 Veröffentlicht
- Kostenlos & Open-Source
- Basiert auf Java ⇒ Plattformunabhängig



GHIDRA

Was ist Software Reverse Engineering überhaupt?

- Rückgewinnung des Quellcodes einer Binärdatei
 - Disassembler: Binary → Assembly (ARM, x86, ...)
 - Decompiler: Binary → C/C++
- Vielerlei Anwendungsgebiete
 - Codeoptimierung
 - Malware-Analyse
 - Schwachstellen-Analyse
 - Software-Modifikation



Hochschule RheinMain

```
1 int main(void)
2 {
3     char *pcVar1;
4     int i;
5     int j;
6     char buffer [100];
7     int counter;
8
9     HAL_Init();
10    SystemClock_Config();
11    MX_GPIO_Init();
12    MX_USART3_UART_Init();
13    pcVar1 = (char *)MX_USART3_UART_Init();
14    do {
15        counter = 0;
16        receiveUART4Trigger((uint)pcVar1);
17        HAL_GPIO_WritePin((GPIO_TypeDef *)0x40020c00,0x80,0);
18        i = 0;
19        while (i < 1000) {
20            j = 0;
21            while (j < 1000) {
22                counter = counter + 1;
23                j = j + 1;
24            }
25            i = i + 1;
26        }
27        HAL_GPIO_WritePin((GPIO_TypeDef *)0x40020c00,0x80,0);
28        if (counter == 1000000) {
29            sprintf(buffer,"0: %d\n",1000000);
30        }
31        else {
32            sprintf(buffer,"1: %d\n",counter);
33        }
34        pcVar1 = buffer;
35        sendUART4(buffer);
36    } while( true );
37 }
38
39
40
```

GHIDRA

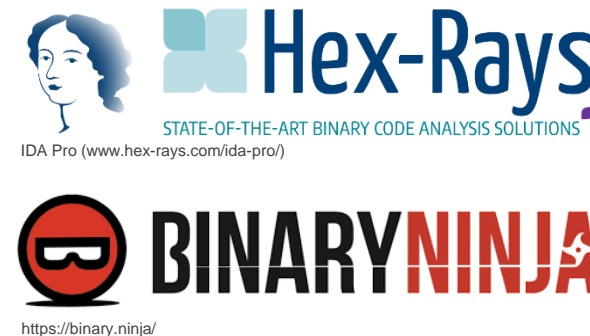
Gegenüberstellung zur Konkurrenz

- Decompiler-Zentriert
- Open-Source
 - Community getriebene Entwicklung
 - Sehr viele Plug-ins
- Mäßige Anzahl an Architekturen
(Stetig mehr durch die Community)
- Kostenlos



- Disassembler-Zentriert
- Closed-Source*
- Sehr viele Architekturen
- Kostenpflichtig*
 - Teilweise Zusatzkosten pro Architektur
 - Teilweise Zusatzkosten für Decompiler

*Ausnahme: Radare



GHIDRA

Decompiler-Zentriert

Wir müssen nicht das Assembly verstehen, sondern dem Decompiler beim verstehen helfen

- Anpassung der Typen und Parametern
- Beheben von Problemen
- Hinzufügen von fehlenden Teilen

```
2 void main(void)
3
4 {
5     long in_RSI;
6     int in_EDI;
7
8     if (in_EDI == 2) {
9         set_globals();
10        check_solution(*(undefined8 *) (in_RSI + 8));
11    }
12    else {
13        puts("Need more arguments!");
14    }
15    return;
16 }
```



```
2 int main(int argc, char **argv)
3
4 {
5     if (argc == 2) {
6         set_globals();
7         check_solution(argv[1]);
8     }
9     else {
10        puts("Need more arguments!");
11    }
12    return (int)(uint)(argc != 2);
13 }
14
```

BINÄRDATEIEN

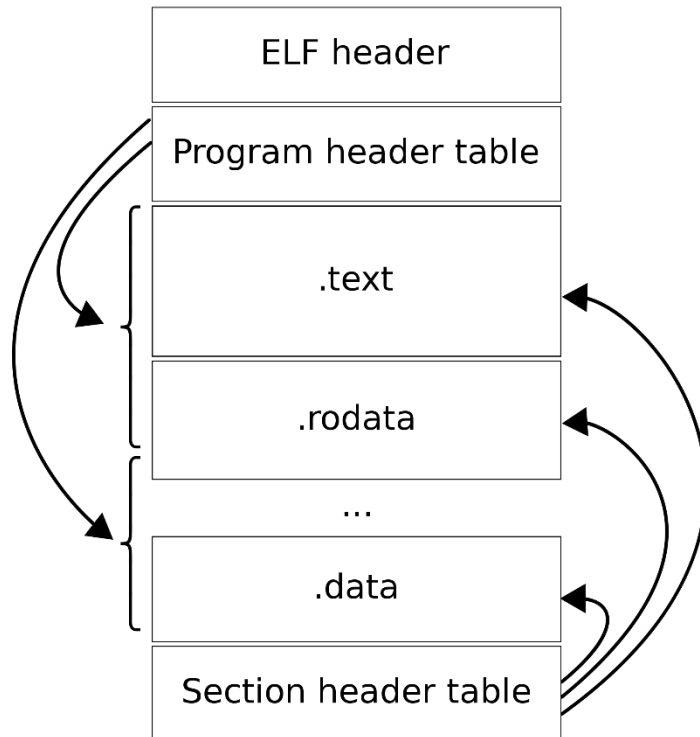
Crash Course

Marius Eggert
21.04.2021

ASSEMBLY

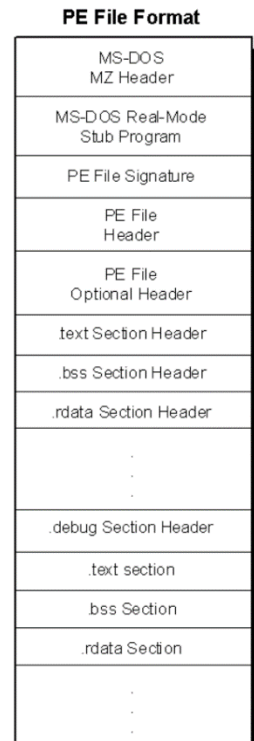
Häufigste Binärformate

- Executable and Linkable Format (ELF)
 - Linux



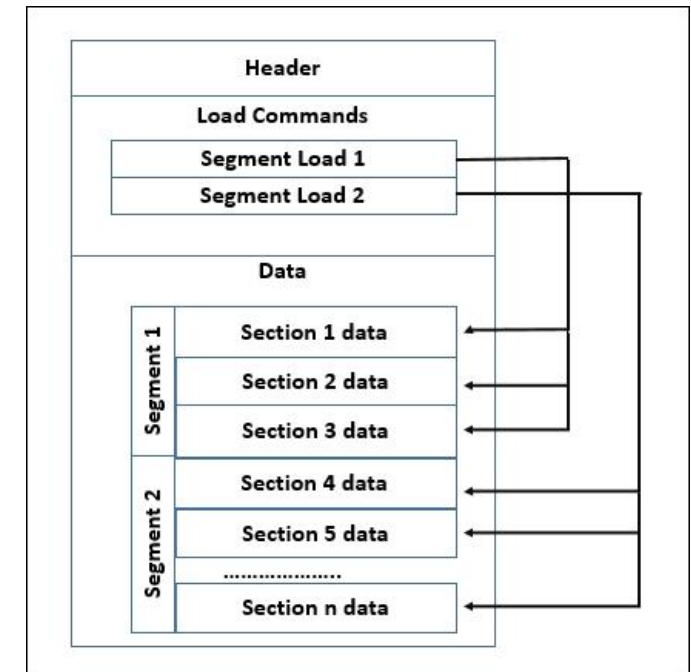
https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

- Portable Executable (PE) Format
 - Windows



<https://blog.kowalczyk.info/articles/pefileformat.html>

- Mach Object File Format (Mach-O)
 - macOS/iOS



<https://www.oreilly.com/library/view/mobile-application-penetration/9781785883378/ch02s14.html>

ASSEMBLY

Häufigste Binärformate



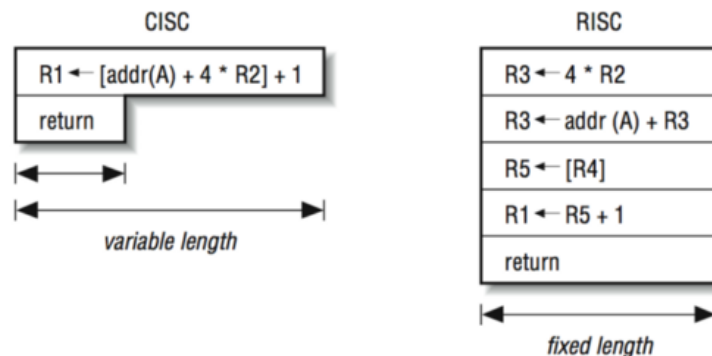
- Raw Binary
 - System on Chips & Mikrocontroller

[illegible][illegible]

ASSEMBLY

Architekturen & Syntax

- Häufigste Assembly Sprachen
 - (PCs nutzen x86/x64)
 - Smartphone/Tablet/Raspberry Pi/... (IoT) nutzen ARM/ARM64/Thumb
 - x86 \triangleq Complex Instruction Set Computer (CISC)
 - ARM \triangleq Reduced Instruction Set Computer (RISC)
 - Grundbefehle sehen sehr ähnlich aus



<https://ichi.pro/de/was-bedeutet-risc-und-cisc-im-jahr-2020-135571763210718>

x86	ARM
pop eax	pop {r0}
mov eax, ebx	mov r0, r1
add eax, ebx	add r0, r0, r1
add eax, 0x10	add r0, #16
mov eax, [ebx]	ldr r0, [r1]
mov [eax+0x10], ebx	str r1, [r0, #16]
call eax	blx r0
jmp eax	bx r0
call function	bl function (return address in lr)
ret	pop {pc} / bx lr
int 0x80	svc 0x80 / svc 0x0

Parikh, Vivek & Mateti, Prabhaker. (2017). ASLR and ROP Attack Mitigations for ARM-Based Android Devices. 350-363. 10.1007/978-981-10-6898-0_29.

ASSEMBLY

Register

- Register x86/x64

Allgemeine Register (je 64 Bit)		
Name	(ursprüngliche) Bedeutung	
RAX	Akkumulator	
RBX	Base Register	
RCX	Counter	
RDX	Data Register	
RBP	Base-Pointer	
RSI	Source-Index	
RDI	Destination-Index	
RSP	Stack-Pointer	
R8...R15	Register 8 bis 15	
64-Bit-Media-/80-Bit-Gleitkommaregister		
Name	Bedeutung	Hinweis
FPR0 ... FPR7	FPU-Register 0 ... 7	werden gemeinsam genutzt
MMX0 ... MMX7	MMX-Register 0 ... 7	

<https://de.wikipedia.org/wiki/X64>

21.04.2021

- Register ARM/ARM64/Thumb

#	Alias	Purpose
R0	-	General purpose
R1	-	General purpose
R2	-	General purpose
R3	-	General purpose
R4	-	General purpose
R5	-	General purpose
R6	-	General purpose
R7	-	Holds Syscall Number
R8	-	General purpose
R9	-	General purpose
R10	-	General purpose
R11	FP	Frame Pointer
Special Purpose Registers		
R12	IP	Intra Procedural Call
R13	SP	Stack Pointer
R14	LR	Link Register
R15	PC	Program Counter
CPSR	-	Current Program Status Register

<https://azeria-labs.com/arm-data-types-and-registers-part-2/>

Reverse Engineering mit Ghidra - Modul IT-Forensik 7440 - Hochschule RheinMain

ASSEMBLY

Register

- Register Gegenüberstellung x86/x64 zu ARM/ARM64/Thumb

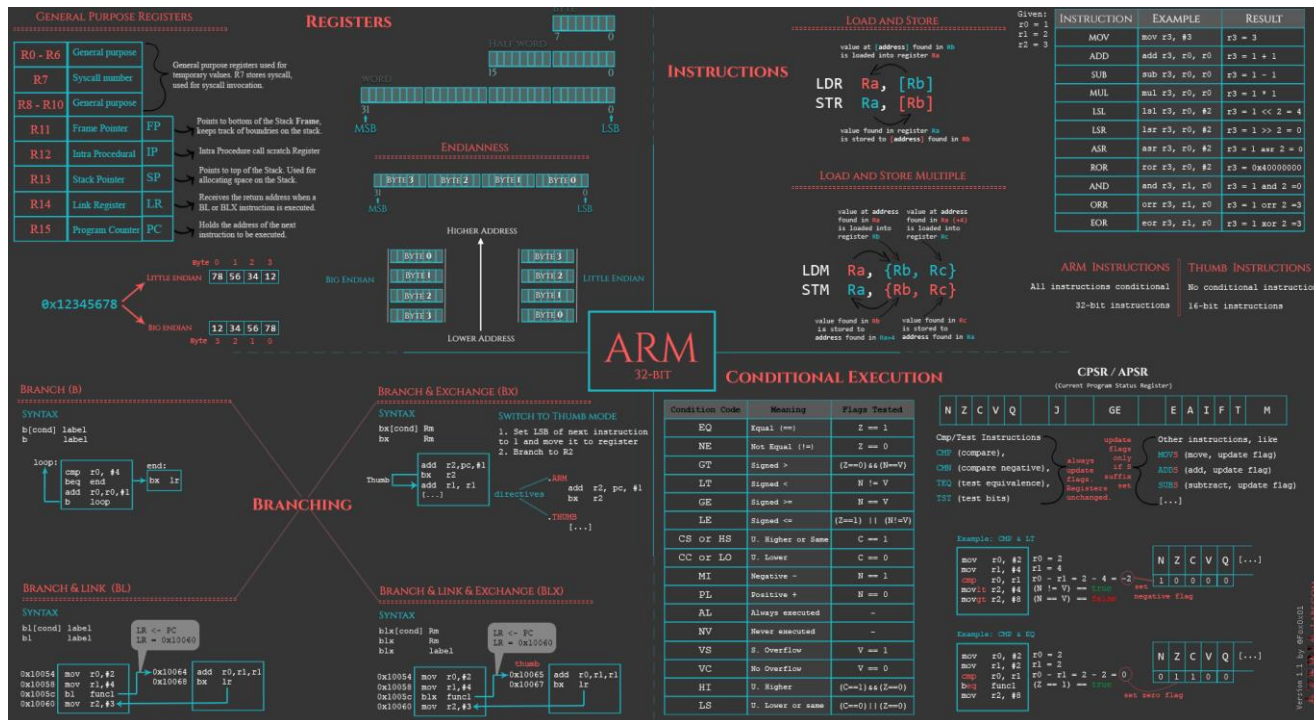
ARM	Description	x86
R0	General Purpose	EAX
R1-R5	General Purpose	EBX, ECX, EDX, ESI, EDI
R6-R10	General Purpose	-
R11 (FP)	Frame Pointer	EBP
R12	Intra Procedural Call	-
R13 (SP)	Stack Pointer	ESP
R14 (LR)	Link Register	-
R15 (PC)	<- Program Counter / Instruction Pointer ->	EIP
CPSR	Current Program State Register/Flags	EFLAGS

<https://azeria-labs.com/arm-data-types-and-registers-part-2/>

ASSEMBLY

Befehle

- Azeria Labs bietet viele gute Tutorials zu ARM an:
<https://azeria-labs.com/writing-arm-assembly-part-1/>



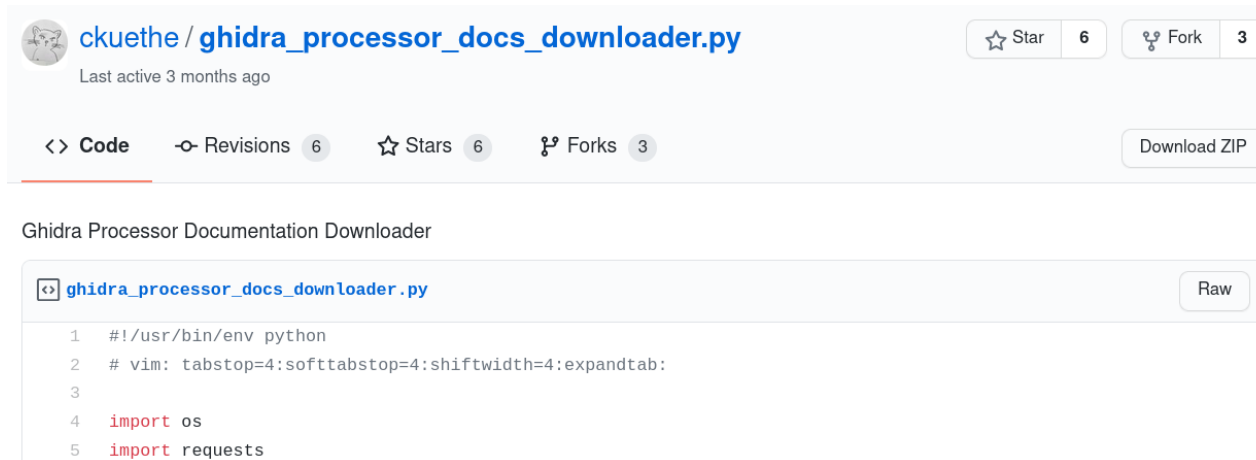
<https://azeria-labs.com/downloads/cheatsheetv1.3-1920x1080.png>

Ghidra ermöglicht zu der aktuellen Instruktion die Dokumentation aufzurufen. Hierfür müssen die unterschiedlichen PDFs in

<ghidra-installations-pfad>/Ghidra/Processors/<prozessor-name>/data/manuals/
Kopiert werden.

Es gibt ein Skript zum automatischem Download der gängigen Prozessoren:

<https://gist.github.com/ckuethe/afdc091635b32ba1077d8470776942b8>



ckuethe / **ghidra_processor_docs_downloader.py**
Last active 3 months ago

☆ Star 6 🍴 Fork 3

<> Code ↶ Revisions 6 ☆ Stars 6 🍴 Forks 3 Download ZIP

Ghidra Processor Documentation Downloader

`ghidra_processor_docs_downloader.py` Raw

```
1  #!/usr/bin/env python
2  # vim: tabstop=4:softtabstop=4:shiftwidth=4:expandtab:
3
4  import os
5  import requests
```

GHIDRA

How-To

Marius Eggert
21.04.2021

GHIDRA HOW-TO

Beispiele > Folien

- Sehr viele Ressourcen online verfügbar:
 - Cheat-Sheet: <https://ghidra-sre.org/CheatSheet.html>
 - Online Courses: <https://ghidra.re/online-courses/>
 - GhidraClass: <Ghidra-Installations-Pfad>/docs/GhidraClass
 - Sonstige Informationen:
 - <https://raw.githubusercontent.com/wiki/NationalSecurityAgency/ghidra/files/blackhat2019.pdf>
 - <https://raw.githubusercontent.com/wiki/NationalSecurityAgency/ghidra/files/recon2019.pdf>
 - ...
- Eine komplette Einführung in alle Funktionen inkl. Skripting würde mehrere Vorlesungen in Anspruch nehmen

 Wir wollen uns deshalb hier auf Beispiele konzentrieren, die das Notwendigste abbilden.
Für alles Weitere: RTFM

GHIDRA HOW-TO

Beispiele - Hinweise

- Alle Beispiele wurden auf einem x64 Linux kompiliert und sind im StudIP zu finden
- Das kompilierte Dateiformat der Beispiele ist:
ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, ...
- Die Codebeispiele und die daraus resultierenden Binärdateien für ein ARM basiertes System sind zusätzlich im Unterordner „arm“ zu finden
 - Der Source-Code wurde teilweise für die Übersetzung zu ARM angepasst
 - Es wurde lediglich die Kompilierung getestet, sodass die Präsenz der gezeigten Beispielproblematiken nicht gewährleistet ist
 - Die Dateien dienen mehr zur Übung bzw. Veranschaulichung der architekturellen Unterschiede
 - Für das Übersetzen in andere Architekturen als das eigene Host-System werden zusätzliche Compiler-Toolchains benötigt
(Beispielsweise „arm-linux-gnueabi“ für die Übersetzung von einem x64 basiertem Linux zu einem ARM basiertem Linux)

BEISPIEL I

„first_ghidra_test“

BEISPIEL I

„first_ghidra_test“

Aufgabe:

- [Programm übersetzen]
- Projekt anlegen & Datei hinzufügen
- Datei analysieren
- [Ghidra anpassen]
- Decompilat verbessern

Macht euch mit den Optionen vertraut!

Hinweise:

- gcc first_ghidra_test.c -o first_ghidra_test
- [arm-linux-gnueabi-gcc first_ghidra_test_arm.c -o first_ghidra_test_arm]

```
1  #include <string.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  void print_systeminfo() {
8      pid_t out_pid = getpid();
9      printf("PID: %d\n", out_pid);
10 }
11
12 int main(int argc, char *argv[]){
13     print_systeminfo();
14     if(argc == 2) {
15         FILE *f = fopen(argv[1], "r");
16         fseek(f, 0, SEEK_END);
17         size_t filesize = ftell(f);
18         printf("Filesize: %lu\n", filesize);
19     }
20     return 0;
21 }
```

BEISPIEL

„first_ghidra_test“

Zusammengefasst:

- Gefundene Funktionen, Imports, Labels, ... werden im Symbol Tree angezeigt
- Ghidra erkennt nicht alle Typen automatisch
⇒ Edit Function Signature
- Sehr viele hilfreiche Ansichten
 - Bytes
 - Bookmarks
 - Checksums
 - Data Type Preview
 - Function Call Graph
- Viele Anpassungsmöglichkeiten

```
Decompile: main - (first_ghidra_test_stripped)
1 |
2 | int main(int argc, char **argv)
3 |
4 | {
5 |     FILE *file;
6 |     long size;
7 |
8 |     // Methode zur Ausgabe der PID
9 |     get_sysinfo();
10 |    if (argc == 2) {
11 |        file = fopen(argv[1], "r");
12 |        fseek(file, 0, SEEK_END);
13 |        size = ftell(file);
14 |        printf("Filesize: %lu\n", size);
15 |    }
16 |    return 0;
17 | }
18 |
```

BEISPIEL II

„first_ghidra_test_stripped“

BEISPIEL

„first_ghidra_test_stripped“

Aufgabe:

- [Datei strippen]
- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern

Hinweis:

- strip first_ghidra_test -o first_ghidra_test_stripped

```
Decompile: FUN_001011a2 - (first_ghidra_test_stripped)
1
2 undefined8 FUN_001011a2(int param_1,long param_2)
3
4 {
5     FILE *__stream;
6     long lVar1;
7
8     FUN_00101179();
9     if (param_1 == 2) {
10         __stream = fopen(*(char **)(param_2 + 8),"r");
11         fseek(__stream,0,2);
12         lVar1 = ftell(__stream);
13         printf("Filesize: %lu\n",lVar1);
14     }
15     return 0;
16 }
```

BEISPIEL

„first_ghidra_test_stripped“

Zusammengefasst:

- Stripping entfernt Variablen- und Methodennamen
⇒ Selbst umbenennen (Shortcut: L)

```
Decompile: main - (first_ghidra_test_stripped)
1 |
2 | int main(int argc, char **argv)
3 |
4 | {
5 |     FILE *file;
6 |     long size;
7 |
8 |     // Methode zur Ausgabe der PID
9 |     get_sysinfo();
10 |    if (argc == 2) {
11 |        file = fopen(argv[1], "r");
12 |        fseek(file, 0, SEEK_END);
13 |        size = ftell(file);
14 |        printf("Filesize: %lu\n", size);
15 |    }
16 |    return 0;
17 | }
18 |
```

BEISPIEL III

„optimized“

BEISPIEL

„optimized“

Aufgabe:

- [Datei optimiert übersetzen]
- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern
- [Ändern der Zählervariable auf volatile]
- [Datei optimiert übersetzen]
- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern

Hinweise:

- gcc optimized.c -O3 -o optimized
- volatile long counter = 0;

```
1  #include <stdio.h>
2  #include <string.h>
3
4  int main(int argc, char *argv[]) {
5
6      int limit = 1000;
7      long counter = 0;
8
9      for(int i = 0; i < limit; i++) {
10         for(int j = 0; j < limit; j++) {
11             counter += 1;
12         }
13     }
14     printf("Counter value: %d\n", counter);
15     if(counter == limit * limit) {
16         printf("Unchanged Program Flow");
17     }
18     else {
19         printf("Changed Program Flow");
20     }
21     return 0;
22 }
```

BEISPIEL

„optimized“

Zusammengefasst:

- Optimierung verkürzt Laufzeit und/oder Größe des Binaries
- Während der Kompilierung können Schleifen und Kaskaden verändert werden
- „unsinnvolle“ Rechnungen werden ersetzt
- Keyword „volatile“ kann helfen

```
Decompile: main - (optimized)
1
2 undefined8 main(void)
3
4 {
5     printf("Counter value: %d\n",1000000);
6     printf("Unchanged Program Flow");
7     return 0;
8 }
```

BEISPIEL

„optimized“

gcc -O option flag

Set the compiler's optimization level.

option	optimization level	execution time	code size	memory usage	compile time
-O0	optimization for compilation time (default)	+	+	-	-
-O1 or -O	optimization for code size and execution time	-	-	+	+
-O2	optimization more for code size and execution time	--		+	++
-O3	optimization more for code size and execution time	---		+	+++
-Os	optimization for code size		--		++
-Ofast	O3 with fast none accurate math calculations	---		+	+++

+increase ++increase more +++increase even more -reduce --reduce more ---reduce even more

<https://www.rapidtables.com/code/linux/gcc/gcc-o.html>

BEISPIEL IV

„patching“

BEISPIEL

„patching“

Aufgabe:

- [Datei übersetzen]
- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern
- Programm „freischalten“
- Programm Exportieren
(https://github.com/schlafwandler/ghidra_SavePatch)
- Freischaltung Testen

Hinweis:

- gcc patching.c -o patching

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void main(int argc, char *argv[]) {
5      char pass[20] = "password";
6      char pass_buf[20];
7
8      gets(pass_buf);
9      printf("Entered Password: %s\n", pass_buf);
10
11     if(strcmp(pass_buf, pass) != 0){
12         printf("Wrong Password! Try Again.\n");
13         return;
14     }
15
16     printf("Unlocked!\n");
17 }
```


BEISPIEL

„patching“

Zusammengefasst:

- Dateien können je nach Bedarf geändert werden
- Ghidras Exportfunktion liefert zzt. leider keine ausführbaren Dateien ⇒ Skript benutzen

```
1
2 void main(int argc, char **argv)
3
4 {
5     int compare_result;
6     long in_FS_OFFSET;
7     char *pass;
8     undefined8 local_40;
9     undefined4 local_38;
10    char pass_buf [24];
11    long stack_canary;
12
13    // Stack Canary
14    stack_canary = *(long *) (in_FS_OFFSET + 0x28);
15    pass = (char *) 7237970109966541168;
16    local_40 = 0;
17    local_38 = 0;
18    gets(pass_buf);
19    printf("Entered Password: %s\n", pass_buf);
20    compare_result = strcmp(pass_buf, (char *)&pass);
21    if (compare_result == 0) {
22        puts("Wrong Password! Try Again.");
23    }
24    else {
25        puts("Unlocked!");
26    }
27    if (stack_canary != *(long *) (in_FS_OFFSET + 0x28)) {
28        // WARNING: Subroutine does not return
29        __stack_chk_fail();
30    }
31    return;
32 }
```

GHIDRA

Decompiler & sonstige Probleme

Marius Eggert
28.04.2021

DECOMPILER PROBLEME

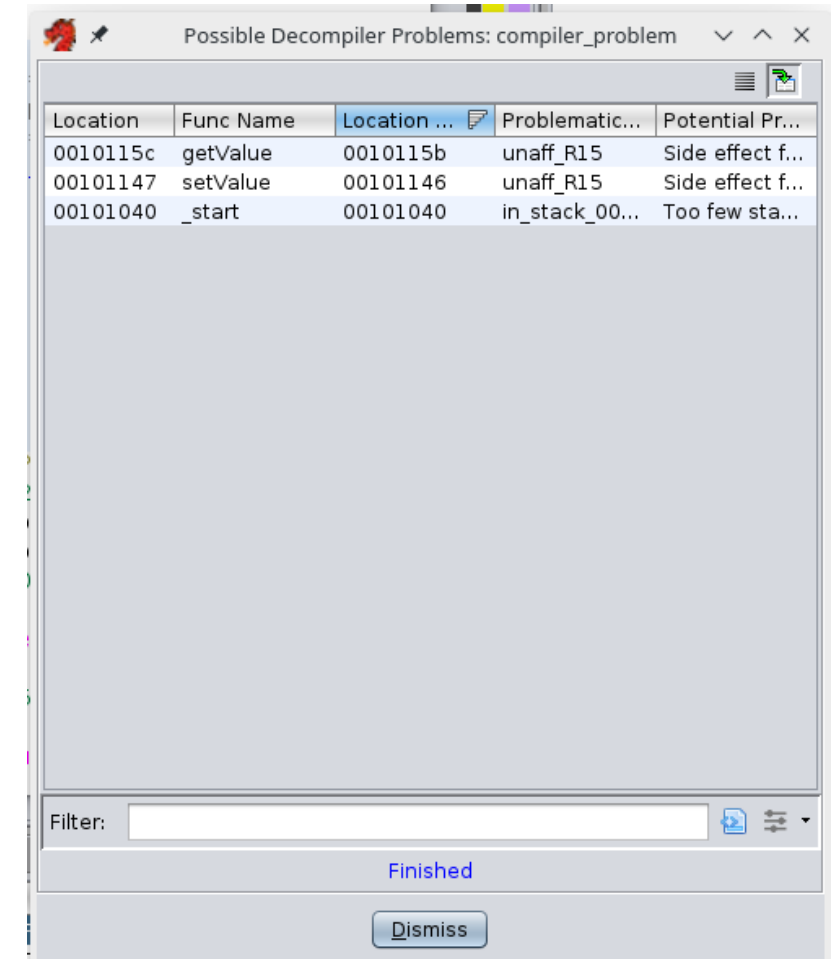
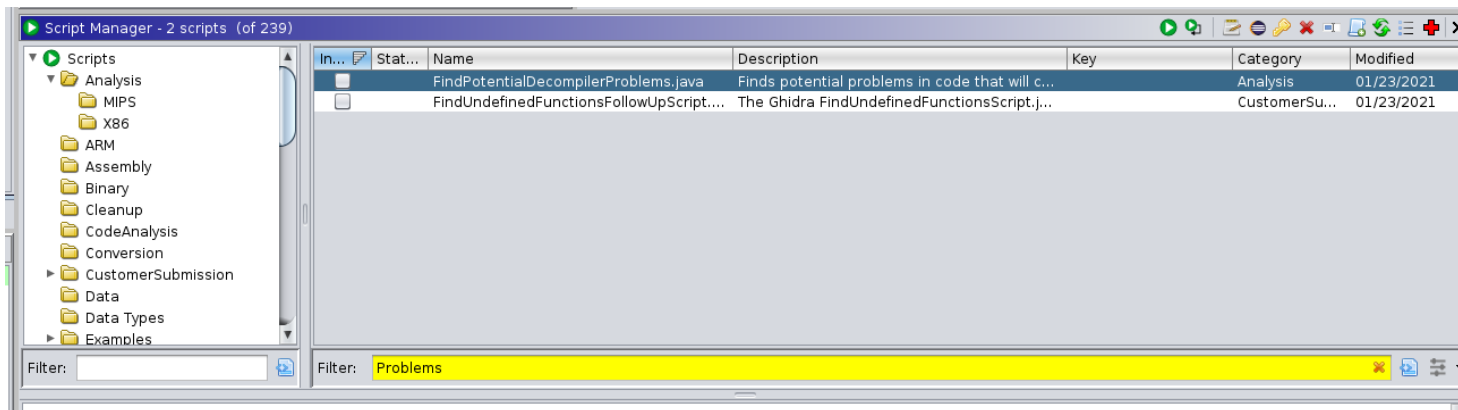
Übersicht

- Alle [Software] Reverse Engineering Tools treffen Annahmen, die nicht immer dem Original entsprechen
 - Datentypen
 - Funktionsenden
 - Calling Conventions
 - ...
- Jedes der Probleme kann je nach genutztem Compiler und dessen Parametern unterschiedlich aussehen und bedarf manueller Anpassung
- In der bei Ghidra mitgelieferten „GhidraClass“ finden sich einige Übungen hierzu, die unter <https://ghidra.re/courses/GhidraClass/Advanced/improvingDisassemblyAndDecompilation.pdf> auch nochmals erklärt sind.
(<ghidra-installations-pfad>/docs/GhidraClass)

DECOMPILER PROBLEME

Übersicht

- Einige Probleme lassen sich über das Skript „FindPotentialDecompilerProblems.java“ automatisch identifizieren



BEISPIEL V

„decompiler_problem“

BEISPIEL

„decompiler_problem“

Aufgabe:

- [Datei übersetzen]
- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern
- Skript „FindPotentialDecompilerProblems.java“ ausführen
- Methoden „reparieren“

Hinweis:

- gcc compiler_problem.c -o compiler_problem
- „Set Register Values“ vom Kontextmenü verwenden

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  long c;
5  register long *b asm("r15");
6
7  void setupRegisterPointer() {
8      b = &c;
9  }
10
11 void setValue(long x) {
12     *b = x;
13 }
14
15 long getValue() {
16     return *b;
17 }
18
19 int main(int argc, char* argv[]) {
20     setupRegisterPointer();
21     setValue(5);
22     long value = getValue();
23
24     printf("Register Value: %lu\n", value);
25     return 0;
26 }
```

BEISPIEL

„decompiler_problem“

Zusammengefasst:

- Es gibt diverse Probleme, die nicht automatisch vom Programm erkannt werden können
- Im Normalfall gilt die Regel:
„Sieht es komisch aus, ist irgendetwas falsch“
- Skripte können das Auffinden erleichtern

```
C: Decompile: getValue - (compiler_problem)
1
2 long getValue(void)
3
4 {
5     return c;
6 }
```

BEISPIEL VI

„custom“

BEISPIEL

„custom“

(Von den Ghidra Beispielen)

Aufgabe:

- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern
- Methoden „reparieren“

```
Decompile: FUN_00400603 - (custom)
1
2 long FUN_00400603(long param_1,long param_2)
3
4 {
5     return param_1 - param_2;
6 }
7
```

BEISPIEL VI

„no_return“

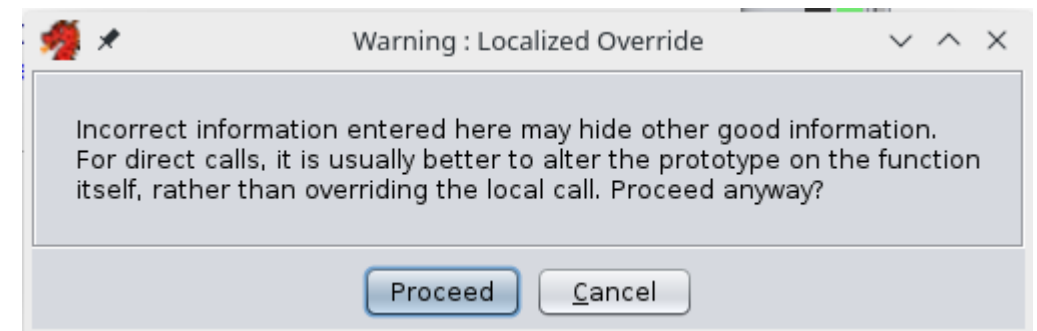
BEISPIEL

„no_return“

Aufgabe:

- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern
- Methoden „reparieren“

```
1  #include <stdlib.h>
2  #include <stdio.h>
3  #include <unistd.h>
4
5  void killMe() {
6      char command[20];
7      int pid = getpid();
8      sprintf(command, "kill %d", pid);
9      system(command);
10 }
11
12 int main(int argc, char *argv[]){
13     printf("I don't want to live anymore...\n");
14     killMe();
15 }
```



BEISPIEL VII

„structure“

BEISPIEL

„structure“

Aufgabe:

- [Datei übersetzen]
- Datei zu Projekt hinzufügen
- Datei analysieren & verbessern
- Struct erstellen und Typen anpassen

```
1  #include <string.h>
2  #include <stdlib.h>
3  #include <stdio.h>
4  #include <sys/types.h>
5  #include <unistd.h>
6
7  struct Person
8  {
9      int citNo;
10     char name[50];
11     float salary;
12 } typedef _Person_t;
13
14 void print_person(_Person_t *p) {
15     printf("name: %s\n", p->name);
16     printf("cityNo: %d\n", p->citNo);
17     printf("salary: %g\n", p->salary);
18 }
19
20 int main(int argc, char *argv[]){
21
22     _Person_t *p1 = malloc(sizeof(_Person_t));
23     char *name = "Jhon Doe";
24     strcpy(p1->name, name);
25     p1->citNo = 5;
26     p1->salary = 12345.50;
27     print_person(p1);
28     return 0;
29 }
```

BEISPIEL

„decompiler_problem“

Zusammengefasst:

- Structures sind im Binary nicht gekennzeichnet
- „komische“ Indexierungen deuten oft auf eine Klasse bzw. Struct hin
- Die automatische Generierung von Structs ist nicht fehlerfrei und muss durch Erkenntnisse des Reverse Engineers angepasst werden
- Alignment-Anforderungen der Compiler können Größen von Arrays variieren

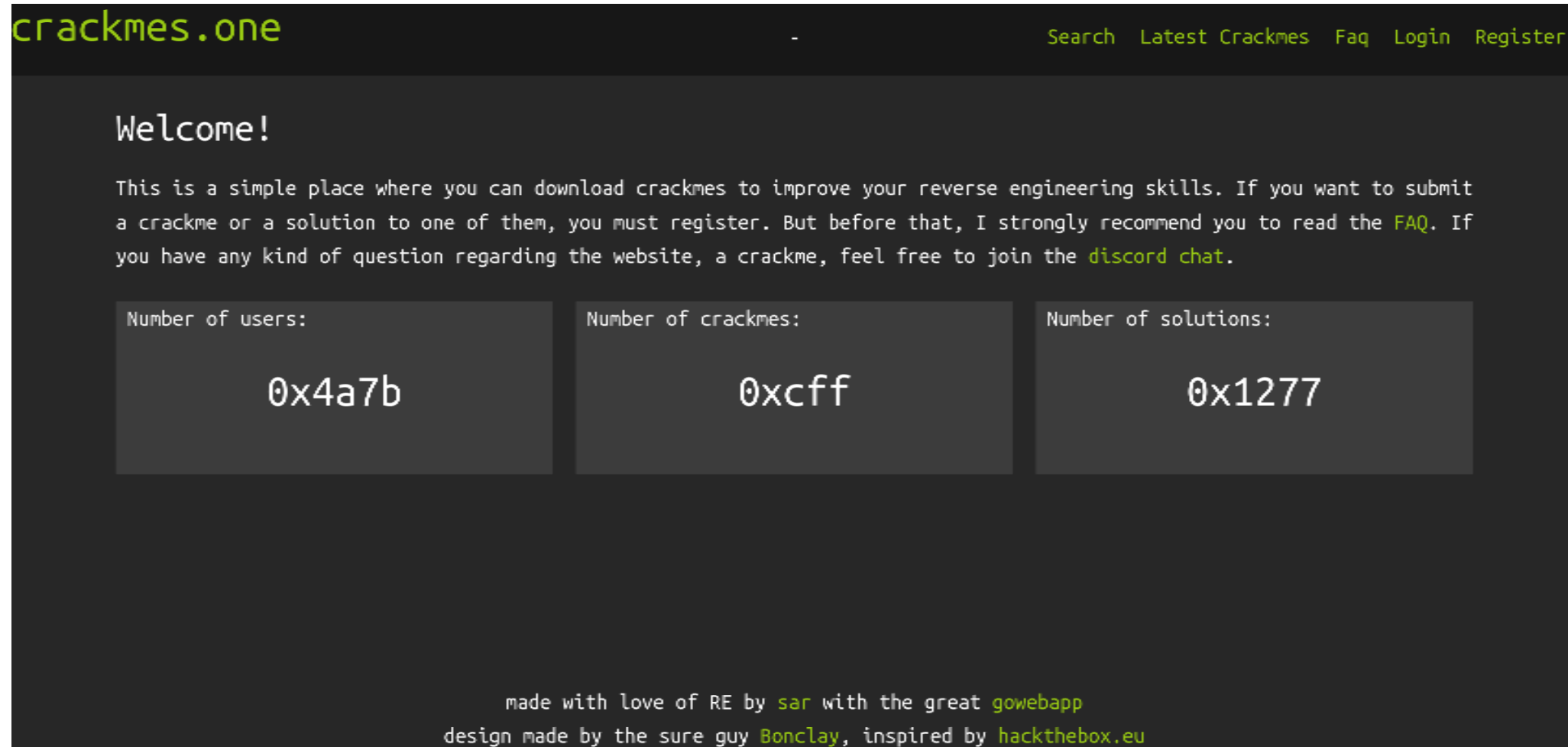
```
Decompile: main - (structure)
1
2 undefined8 main(void)
3
4 {
5     Person *p;
6
7     p = (Person *)malloc(0x3c);
8     strcpy(p->name, "Jhon Doe");
9     p->cityNo = 5;
10    p->salary = 12345.5;
11    print_person(p);
12    return 0;
13 }
```

BEISPIELE VIII & IX

„easy_reverse“ & „crackme.exe“

BEISPIELE

„easy_reverse“ & „crackme.exe“



The screenshot shows the homepage of the website crackmes.one. The site has a dark theme with green text for links and headers. At the top, the site name 'crackmes.one' is in green, followed by navigation links: Search, Latest Crackmes, Faq, Login, and Register. A 'Welcome!' message is displayed, followed by a paragraph explaining the site's purpose and providing instructions for users. Below this, three statistics are shown in a row: 'Number of users: 0x4a7b', 'Number of crackmes: 0xcff', and 'Number of solutions: 0x1277'. At the bottom, a footer credits the site's creator and mentions inspirations.

crackmes.one

Search Latest Crackmes Faq Login Register

Welcome!

This is a simple place where you can download crackmes to improve your reverse engineering skills. If you want to submit a crackme or a solution to one of them, you must register. But before that, I strongly recommend you to read the [FAQ](#). If you have any kind of question regarding the website, a crackme, feel free to join the [discord chat](#).

Number of users:	Number of crackmes:	Number of solutions:
0x4a7b	0xcff	0x1277

made with love of RE by [sar](#) with the great [gowebapp](#)
design made by the sure guy [Bonclay](#), inspired by [hackthebox.eu](#)

<https://crackmes.one/>

BEISPIEL

„easy_reverse“ & „crackme.exe“

Aufgabe:

- Have Fun!

Lösungen gerne als private Nachricht 😊

Name	Author	Language	Difficulty	Quality	Platform	Date	Solution	Comments
fl04t	X3eRo0	Assembler	3.0	4.0	Unix/linux etc.	4:48 AM 04/16/2021	0	0
101happy's #1 crackme	101happy	(Visual) Basic	2.0	2.5	Windows	11:26 AM 04/14/2021	0	1
YoRHa	empress	.NET	4.0	2.5	Windows	12:49 AM 04/13/2021	0	1
My_first_ever_crackme	ASBO_GENERAL	C/C++	1.9	3.8	Unix/linux etc.	3:04 PM 04/12/2021	3	1
insane password checker	thereddev	C/C++	3.8	3.7	Windows	2:21 PM 04/09/2021	0	2
0verney	S0lden	Assembler	2.5	3.7	Unix/linux etc.	11:41 AM 04/08/2021	2	1
SanSuu	r0B	C/C++	2.8	4.0	Windows	2:33 PM 04/05/2021	1	9
crack me idiot 2	LastByteCheatZ	.NET	2.5	2.5	Windows	2:25 PM 03/26/2021	0	1
Advanced static analysis	m4ddin	C/C++	3.0	4.0	Windows	7:28 PM 03/20/2021	1	4
Basis static	m4ddin	C/C++	1.0	3.5	Windows	7:22 PM 03/20/2021	5	12
KataVM -- Level 1	Towel	C/C++	4.0	6.0	Unix/linux etc.	6:25 AM 03/19/2021	1	2
02 CrackMe - Exploit	Exploit-	C/C++	3.0	4.0	Windows	6:25 PM 03/15/2021	0	3

<https://crackmes.one/>