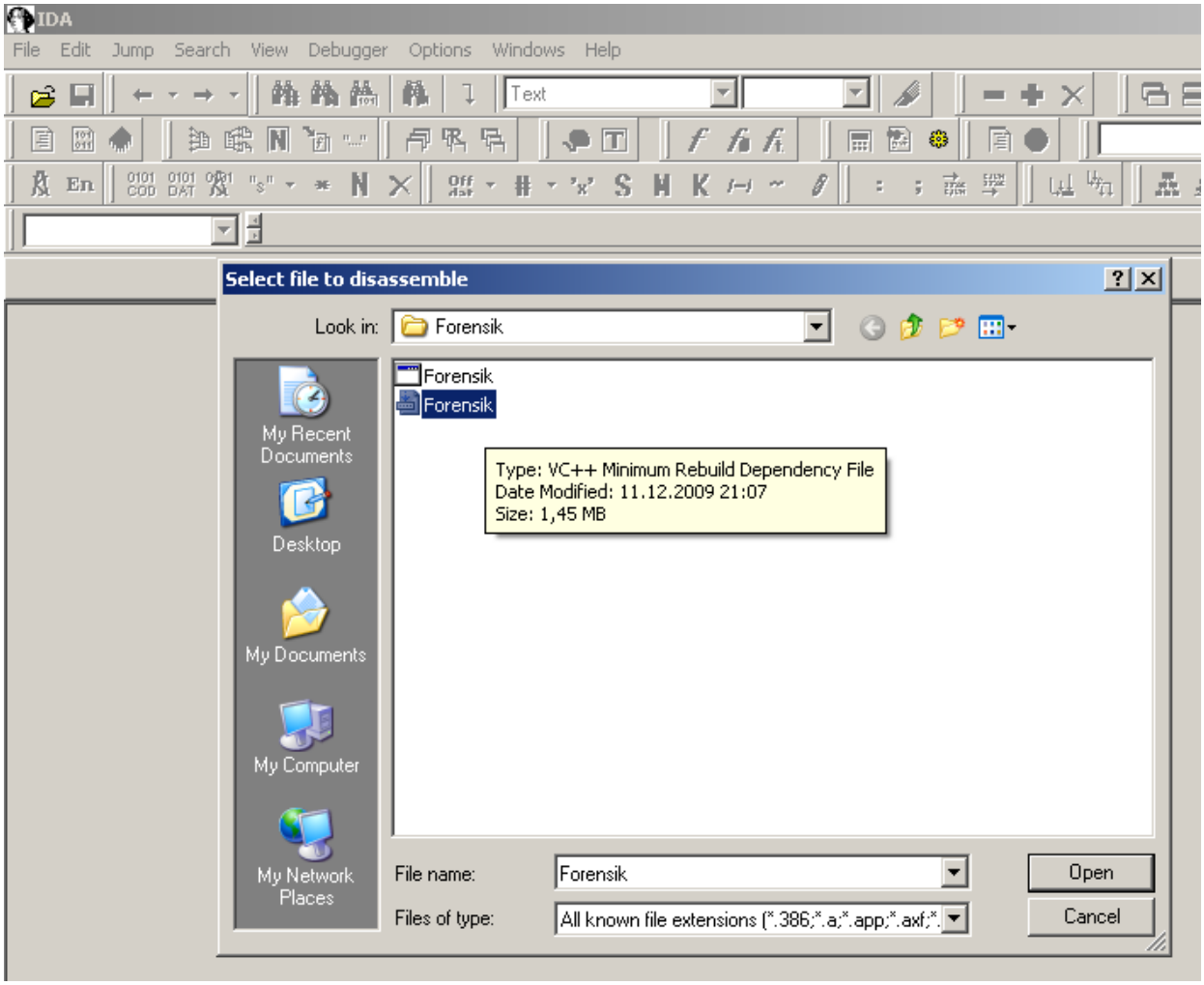
IT-Forensik @ FH-Aachen
Reverse Engineering

Diese kleine Challenge soll dir einen Einblick in die Welt des Reverse Engineering geben. Versuch anhand eines einfachen Dechiffrier Programms im Disassembler das Entschlüsslungsverfahren und das Passwort zu erarbeiten. Natürlich steht es dir auch frei das Passwort durch Probieren zu erraten.

Viel Erfolg!

1. Starte die Datei Forensik.exe
2. Kannst du das Passwort erraten ?
   (Tipp: Es ist eine Zahl!)

3. Öffne zuerst den Disassembler IDA Pro
4. Lade die bereits erstellte Projekt Datei „Forensik.idb“

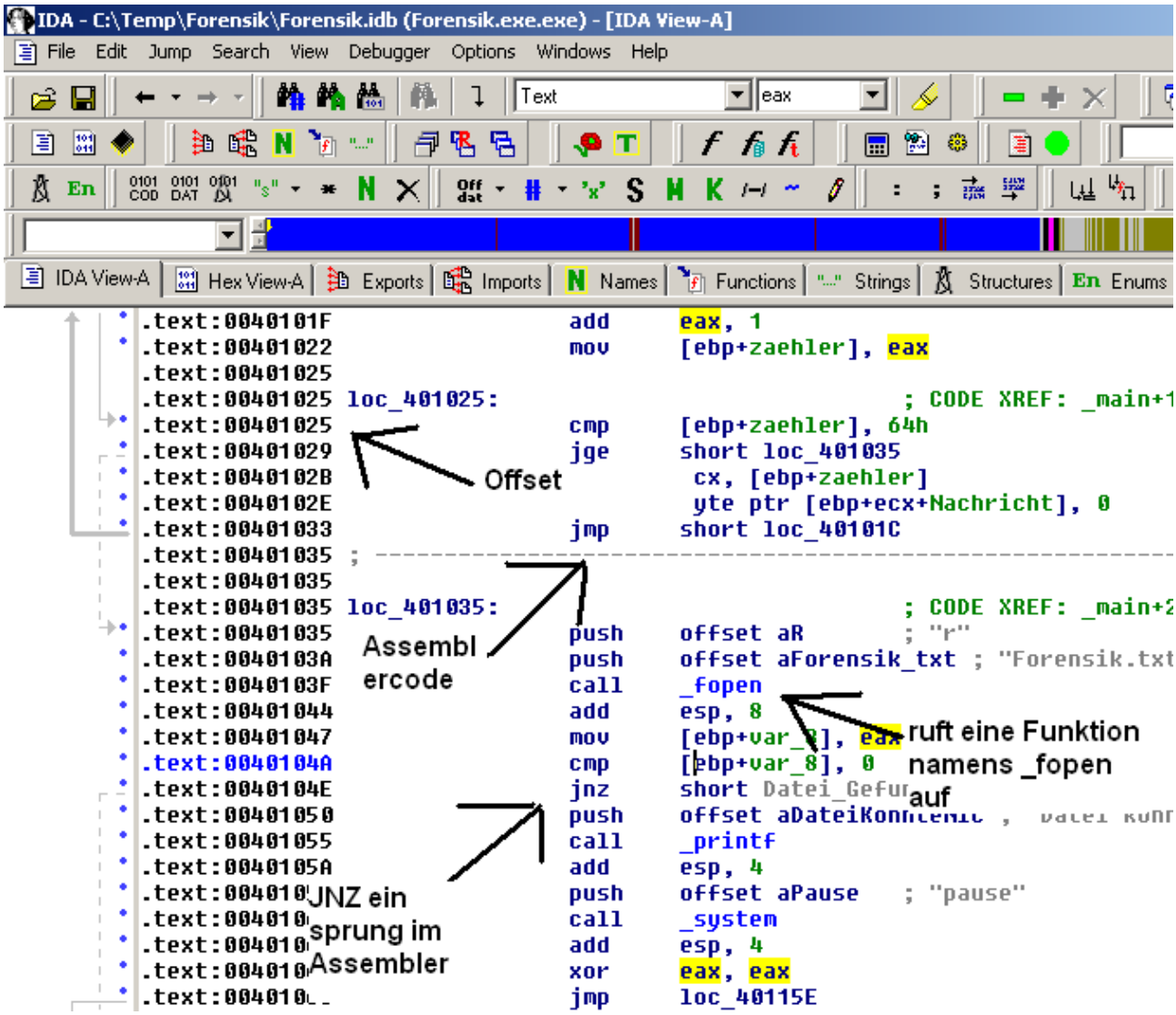


Im Disassembler siehst du das disassemblierte Programm „**Forensik.exe**“. Die für uns interessante Hauptroutine „**_main**“ befindet sich am ***Offset 00401000*** und endet am ***Offset 0040116B***. Im **Anhang** findest du eine Liste der **ASCII Codes** und ihre Darstellung im Decimal Format. Des Weiteren findest du dort eine kurze Erläuterung einiger **Assembler Befehle** und **C Funktionen** die für Dich **interessant** sein könnten. Diese Hilfsmittel sollen Dir helfen das Verfahren zur Ver- und Entschlüsselung zu erarbeiten und daraus das Passwort herzuleiten.

Versuche diese Fragen zu beantworten:

1. Wie ist der Name der Datei die das Programm „Forensik.exe“ öffnet ?
   Antwort: _____
2. Was passiert wenn die Datei nicht geöffnet werden kann?
   Antwort: _____
3. Wie viele Zeichen werden aus der Datei gelesen ?
   Antwort: _____
4. An welchem Offset wird das Passwort eingelesen ?
   Antwort: _____
5. Was wird mit dem Passwort nach dem Einlesen gemacht ?
   Antwort: _____
6. An welchem Offset findet die Entschlüsselung der Nachricht statt ?
   Antwort: _____
7. Wie wird die Nachricht Entschlüsselt ?
   Antwort: _____
8. Schaue dir die Datei die von „Forensik.exe“ geöffnet wird an. Suche die ersten beiden Buchstaben in der ASCII Liste. Welcher Abstand zum nächsten Buchstaben würde Sinn machen, wenn das Verfahren aus Frage 7 angewendet wird.
   Antwort: _____
9. **Wie lautet das Passwort ?**
   Antwort: _____
10. Wie lautet das verschlüsselte Wort „Reverse“ wenn man das selbe Verfahren und Passwort anwendet ?
    Antwort: _____

Anhang:


IDA - C:\Temp\Forensik\Forensik.idb (Forensik.exe.exe) - [IDA View-A]
File Edit Jump Search View Debugger Options Windows Help
IDA View-A Hex View-A Exports Imports Names Functions Strings Structures Enums
.text:0040101F add eax, 1
.text:00401022 mov [ebp+zaehler], eax
.text:00401025
.text:00401025 loc_401025: ; CODE XREF: _main+1
.text:00401025 cmp [ebp+zaehler], 64h
.text:00401029 jge short loc_401035
.text:0040102B cx, [ebp+zaehler]
.text:0040102E yte ptr [ebp+ecx+Nachricht], 0
.text:00401033 jmp short loc_40101C
.text:00401035 ; ---
.text:00401035
.text:00401035 loc_401035: ; CODE XREF: _main+2
.text:00401035 push offset aR ; "r"
.text:0040103A push offset aForensik_txt ; "Forensik.txt
.text:0040103F call _fopen
.text:00401044 add esp, 8
.text:00401047 mov [ebp+var_8], eax
.text:0040104A cmp [ebp+var_8], 0
.text:0040104E jnz short Datei_Gefur
.text:00401050 push offset aDateiKonnte ; "Datei konn
.text:00401055 call _printf
.text:0040105A add esp, 4
.text:0040105D push offset aPause ; "pause"
.text:00401062 call _system
.text:00401067 add esp, 4
.text:0040106A xor eax, eax
.text:0040106C jmp loc_40115E
Offset
Assembl
ercode
ruft eine Funktion
namens _fopen
auf
JNZ ein
sprung im
Assembler

ASCII Tabelle:

000	NUL	033	!	066	B	099	c	132	ä	165	Ñ	198	ã	231	þ
001	Start Of Header (SOH)	034	"	067	C	100	d	133	à	166	ª	199	Ã	232	Þ
002	Start Of Text (STX)	035	#	068	D	101	e	134	å	167	º	200	╚	233	Ú
003	End Of Text (ETX)	036	$	069	E	102	f	135	ç	168	¿	201	╔	234	Û
004	End Of Transmission (EOT)	037	%	070	F	103	g	136	ê	169	®	202	╩	235	Ù
005	Enquiry	038	&	071	G	104	h	137	ë	170	¬	203	╦	236	ý
006	Acknowledge (ACK)	039		072	H	105	i	138	è	171	½	204	╠	237	Ý
007	Bell	040	(	073	I	106	j	139	ï	172	¼	205	═	238	¯
008	Backspace (BS)	041	)	074	J	107	k	140	î	173	¡	206	╬	239	´
009	Horizontal Tab	042	*	075	K	108	l	141	ì	174	«	207	¤	240	-
010	Line Feed **(LF)**	043	+	076	L	109	m	142	Ä	175	»	208	ð	241	±
011	Vertical Tab	044	,	077	M	110	n	143	Å	176	░	209	Ð	242	‗
012	Form Feed **(FF)**	045	-	078	N	111	o	144	É	177	▒	210	Ê	243	¾
013	Carriage Return **(CR)**	046	.	079	O	112	p	145	æ	178	▓	211	Ë	244	¶
014	Shift Out	047	/	080	P	113	q	146	Æ	179	│	212	È	245	§
015	Shift In	048	0	081	Q	114	r	147	ô	180	┤	213	ı	246	÷
016	Dataline Escape (DLE)	049	1	082	R	115	s	148	ö	181	Á	214	Í	247	¸
017	DC 1 (XON)	050	2	083	S	116	t	149	ò	182	Â	215	Î	248	°
018	DC 2	051	3	084	T	117	u	150	û	183	À	216	Ï	249	¨
019	DC 3 (XOFF)	052	4	085	U	118	v	151	ù	184	©	217	┘	250	·
020	DC 4	053	5	086	V	119	w	152	ÿ	185	╣	218	┌	251	¹
021	Negative Acknowledge (NAK)	054	6	087	W	120	x	153	Ö	186	║	219	█	252	³
022	Synchronous Idle	055	7	088	X	121	y	154	Ü	187	╗	220	▄	253	²
023	End Of Transmission Block	056	8	089	Y	122	z	155	ø	188	╝	221	¦	254	■
024	Cancel	057	9	090	Z	123	{	156	£	189	¢	222	Ì	255	
025	End Of Medium	058	:	091	[	124	|	157	Ø	190	¥	223	▀		
026	Substitude	059	;	092	\	125	}	158	×	191	┐	224	Ó		
027	Escape **(ESC)**	060	<	093	]	126	~	159	ƒ	192	└	225	ß		
028	File Seperator	061	=	094	^	127 (DEL)	⌂	160	á	193	┴	226	Ô		
029	Group Seperator	062	>	095	_	128	Ç	161	í	194	┬	227	Ò		
030	Record Seperator	063	?	096	`	129	ü	162	ó	195	├	228	õ		
031	Unit Seperator	064	@	097	a	130	é	163	ú	196	─	229	Õ		
032	*SPACE* (SP)	065	A	098	b	131	â	164	ñ	197	┼	230	µ		

C Funktionen:

fopen(DateiName,Öffnungsart) – öffnet eine Datei
fread(FilePointer,Anzahl,Größe,ZielBuffer) – liest Daten aus einer Datei
scanf(Typ,ZielBuffer) – liest Text aus der Konsole
printf – gibt Text in der Konsole aus
system - gibt einen Befehl an das Betriebssystem weiter

Assembler Funktionen:

32 Bit Register: eax,ebx,ecx,edx,...
16 Bit Register: ax, bx, cx, dx, … (Teilregister von eax,ebx,ecx,edx)
8 Bit Register: al,bl,cl,dl,..... (Teilregister von ax,bx,cx,dx,....)

call - ruft eine Unterroutine auf
mov a,b - setzt a gleich b
movsx a,[b] - setzt a auf ASCII Zeichen aus b
cmp a,b - vergleicht a mit b
jg xxx - springt zu xxx wenn b grösser a
jnz xxx - springt zu xxx wenn a ungleich b
jz xxx - springt zu xxx wenn a gleich b
jmp xxx - springt zum Offset xxx
add a,b - addiert den Wert b auf a
sub a,b - subtrahiert den Wert b von a

Parameter Übergabe in Assembler:
fopen(DateiName,Rechte) wird im Assembler so dargestellt:

```
push Rechte
push DateiName
call fopen
```