Reversing Zuma555's crackme #3

Introduction

Decided to get back into reversing after a very long hiatus, in which now I must revise my skills to get back in top form. Figured this was also a good time to write a tutorial in cryptographic keygenning for beginners. This tutorial assumes you have some basic reversing knowledge, as well as familiarity with unpacking basic packers with the ESP trick.

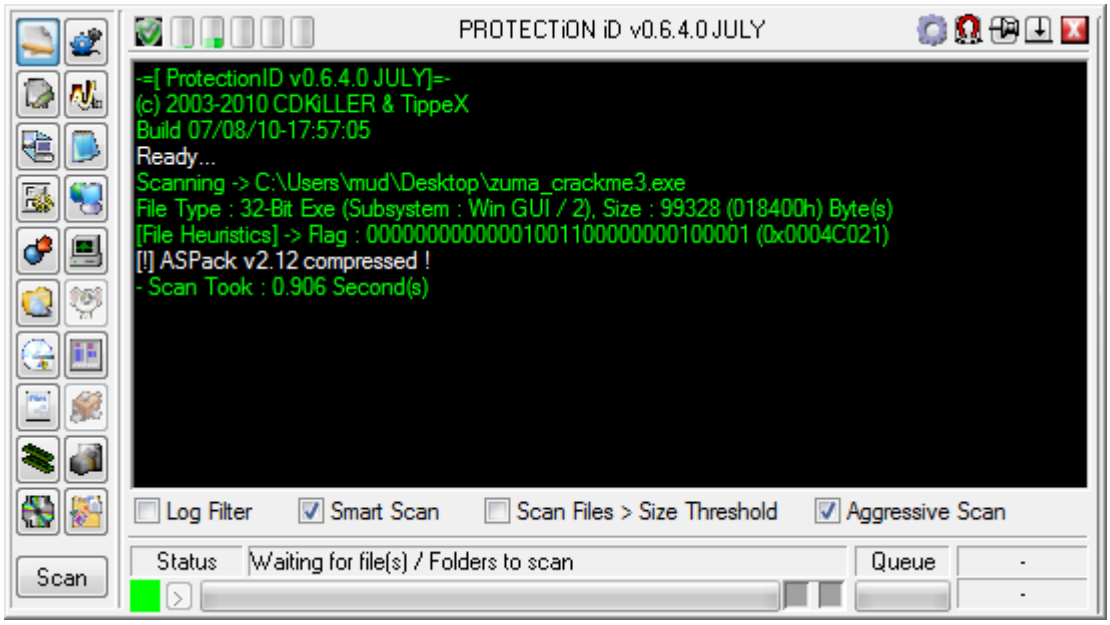
Tools

You will need the following:

- OllyDbg
- SND Reverser Tool
- Hash & Crypto Detector
- ProtectionID

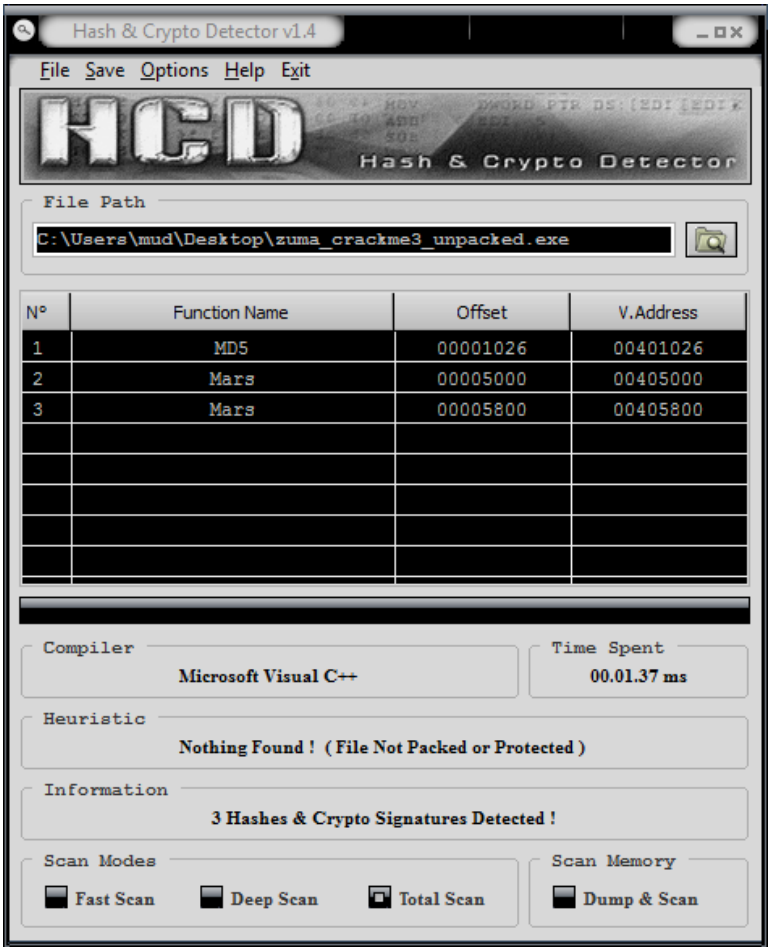
Analysis

First of all, it would be handy to get an idea of what we are dealing with here, so we use ProtectionID to scan for anything wrapping the crackme.



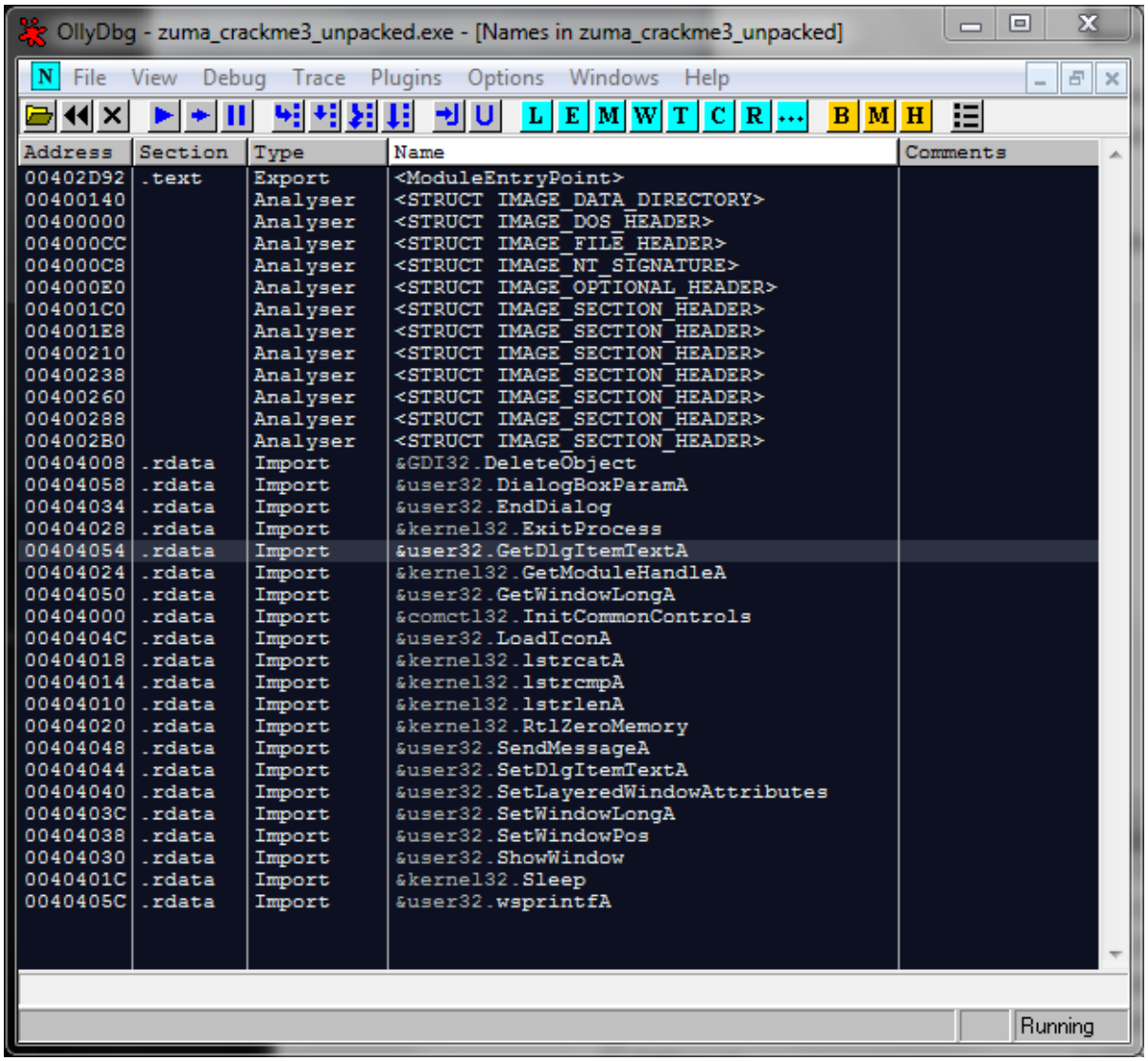
Okay, so it uses ASPack to pack the EXE. Unpacking is simple if you know how, if you know the ESP trick, it will work. So, then unpack the executable. Otherwise, feel free to look at the unpacked version instead.

From here we need to verify what sort of encryption algorithms, if any, is used on the executable. We use Hash & Crypto Detector for this:



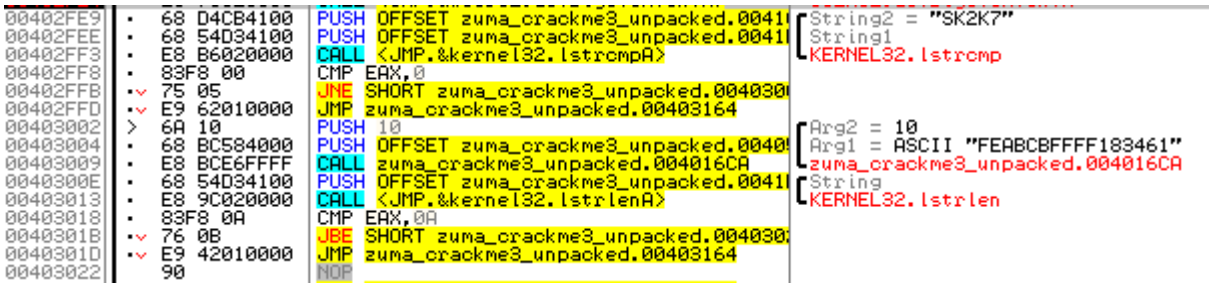
According to HCD, we get two hits for the MARS encryption algorithm, and one for MD5. Since there were 2 hits for MARS, I'd go with that in this case. We can double-check later if it's not ;),

Now we can fire up the target in OllyDbg, and place a breakpoint on any API that handles text input…

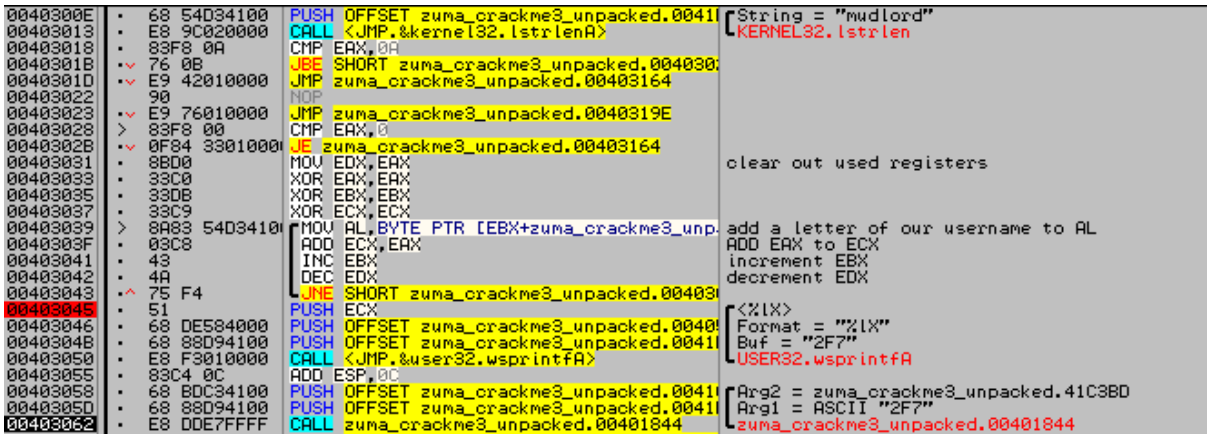


Clearly, GetDlgItemTextA() looks quite suspicious, so we place BPs on that import and any reference to it. Bam! We land…..

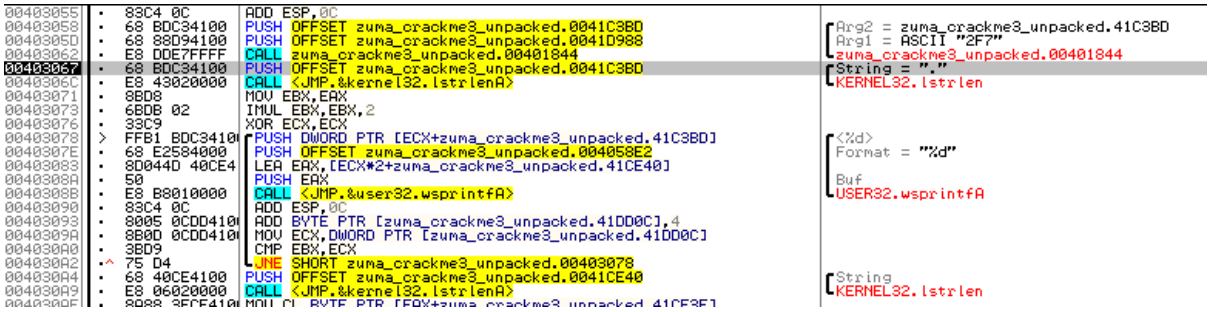
….right into the code for getting our username, in this case “mudlord” and shoving the result into a buffer.



It then does a check against the string SK27. If it matches that, it jumps to the badboy, which of course is something we don’t want. So we give a name that’s anything other than that. Then, once our name is accepted, it seems that the string “FEABCBFFFF183461” is passed to a function which seems rather important…and I’ll let you know very, very soon why.



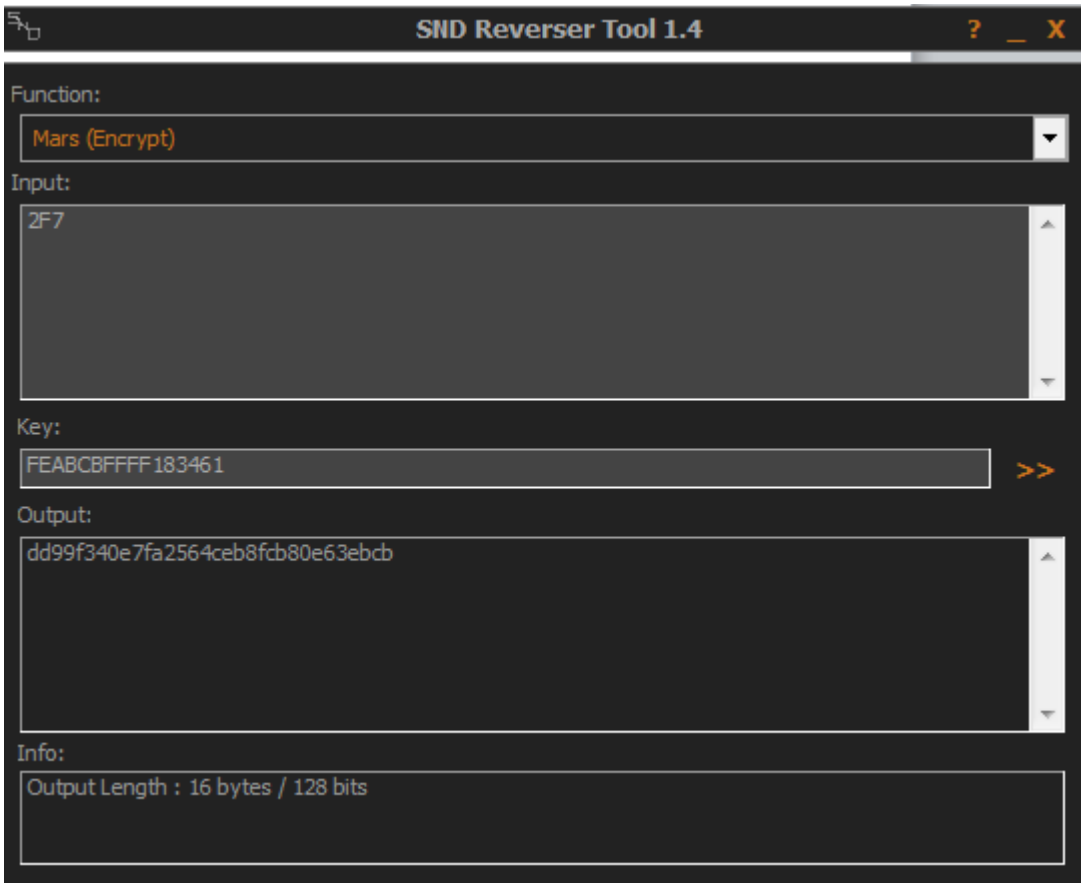
From there the length of our username, is checked. If it is greater than 10 letters/numbers, then it goes to the badboy too. If our username is right, then some funky stuff is done with our username, which the output goes to ECX. From there, it looks like the output of the ECX registered is formatted (in my case is 2F7) is past to a weird function.



This weird function in fact, seems to be the MARS encryption function! In fact, here’s a way we can prove it. Remember I told you to grab SND Reverser Tool? Now is the time to use it!

```
0041C3BD DD 99 F3 40 E7 FA 25 64 CE B8 FC B8 0E 63 EB CB
```

As you can see, that is the output of the function at 00403062, with the input “2F7”. Now then, MARS is what’s called a “block cipher”, a way to encrypt data. And like a padlock, MARS requires a key. Remember that weird string earlier? What if we use IT as a key?



..and there we go. Proof that this crackme uses MARS as a encryption method, and that the “weird string” we had earlier, was in fact, a encryption key. So we can assume, that the function that the key was passed to, was actually the function to set the key for the MARS encryption function. And if you tried MD5, it won’t give the same results, so MD5 is not used here.

So, with this new knowledge, we get to the main bulk of the key generation.

This might look like a lot of code but actually, it is quite simple to understand.

1. The length of the encrypted output is first checked
2. Then the output of the encryption is formatted into one long string.
3. Then, the length of the newly made string is checked, as you can see at 004030A9
4. After that, at 004030AE to 00403F6, a new smaller string is generated from the larger string by splicing portions of it.
5. At 004030FD, the first letter of the string “SK2K7” is read into the AL register. Respectively, the first letter of the string “ENCRYPTO//tKm” is read into CL.
6. Then, AL is XORed with CL and then multiplied by the first few characters in the string “2K7”
7. Finally, the result is then formatted.
8. The final serial is composed of two elements, the one element containing spliced portions of the large serial plus the serial made from step 5-7.

From there, a simple string compare is done in the crackme to check.

Files

src/mars.asm : assembler implementation of MARS encryption
src/zumacrackme3: key generation algorithm
keygen.exe : compiled keygen
zuma_crackme3.exe : original crackme by zuma555
zuma_crackme_unpacked.exe : unpacked crackme