Cryptography - Attack Plan

lrfk99

Version 2020.12.29

1 Attack Plan

This is my attack plan The weakness of ECB mode is that it lacks diffusion. Since it encrypts identical plaintext blocks into identical ciphertext blocks, it does not hide data patterns well. As such, glocal patterns across the plaintext are still kept in the ciphertext.

Something else that I can exploit here is the fact that the key is 8 bytes long, yet the outcome (and therefore the input) is 16 bytes long, so the ciphertext is generated by using the same key twice on each half of the plaintext. Therefore we can look at each half of the ciphertext individually in order to try to work out the 8 byte long plaintext behind each. we know that the input was more than 8 bytes if padding was used and exactly 16 bytes if padding was not used. Either way we know that the key is repeated for longer inputs, as such we know that the same key was used to encrypt the first half of the message "90 34 08 ec 4d 95 1a cf" as the second half "ae b4 7c a8 83 90 c4 75".

The encrypt.exe file does not use padding, thereforee we can tell that the input is exactly 16 bytes long. There are 201 English words which only contain hexidecimal characters, e.g. 'abcdef'. The location is a combination of these, of length 14, since there are 2 '.' characters. Scrap that, what 3 words location included '.' seperators, which are not hexidecimal, therefore must be converting string to hex.

Input is a 32 character hexidecimal string, which means 16 letter. We can break ciphertext in half to work out each half of the plaintext at a time, so if the first eight letters of a what 3 words address such as "heavy.br" produces the first half of the ciphertext "90 34 08 ec 4d 95 1a cf" then we have found part of the address which was encrypted. encrypt.exe "heavy.br" == 90 34 08 ec 4d 95 1a cf I could narrow it down the around 7 million possibilities for the first half of the plaintext, using this I would then test out all seconmd half possibilities. However this would take a long time (60 days) due to the inbuilt delay in the encrypter.

Any attempt at running the executable file encrypt.exe multiple times will not work since the program has a built in 0.5 second delay in order to prevent brute force attacks.