Programming Assignment 2 Security Brennan Giles

The assignment doesn't seem to allocate any points for the first 6 steps but just in case it does, I flipped a bit in the sentence so that it went from cow to cov and this is the difference between the two

SHA256(cow)= 9ed7b218f3402376f0d2f77b89093704515f68f82857f17c9ba8dc4348ecf81f

SHA256(cow)= ef1ff199c23605d0162c87872830ed1ed5aabc39ad23ddb90e43a6b3dd317d09

3.2

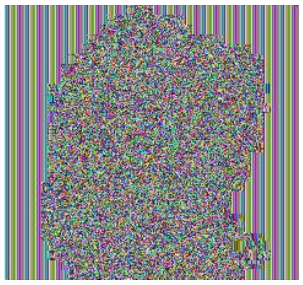
HMAC-SHA256(cow)= 43308c6aaf2a8d653b2b60bc8b94a3955b24cbad66be35798b9dabfd2e600ce7

HMAC-SHA256(cow)= 0d4a944abbb912d5d2dda1f2869933b4d9bf78914e9c394c9e64eead91ea7c58

HMAC-SHA256(cow)= 2dfae075aec316ea4a7d36841d7386de91fcb56b89a9653945f47926e3e4a506

These are the results of the HMAC-SHA256 for keys with size 128, 160, and 256 bits. The key size can be anything you want but it is not necessary to use a hashing algorithm that uses more bits than the size of the key.

HMAC algorithm is really quite flexible, so you could use a key of any size. However, if you only use a 128-bit key then there is no point using a 256-bit hash; you might as well use a 128-bit hash like MD5.3.3


Ecb

cbc

plain

Ecb has patterns which could allow the attacker to discern the origins of the image. Cbc is completely random, and required an IV unlike ecb.

3.4

1. No it does not, because encrypt2 is the encryption of encrypt 1. That means the data will simply be obfuscated further, and changed entirely

2. Still no, same reason as for 1

3.5 The Executable has a header, which can be corrupted so that windows handles it as a DOS program. This allows attacker to xor in their own code into the first 16 bytes of data into the executable file.

2. CFB mode, which allows for this block to be edited

3. It is too malleable and allows attackers to insert their own code into a single block, and potentially more

4. I would have used aes on the whole thing to begin with with key based on user password and iv based on timestamp

3.5

1. There is an old DOS EXE header that usually gets ignored, but if you flip a bit to make windows think the executable is no longer an application it will execute the dos stub. Attacker can xor 16 bytes of malicious code into fifth block of exe file where the dos stub should be and now hes got an avenue of attack.
2. CFB or cipher feedback mode has self correction, but when a block is edited by bit flipping it makes the next block garbage.
3. Described above. Definitely to blame is windows as well however for not cleaning up old header information that leads to this.
4. An authenticated encryption mode such as GCM.
5. Code-reuse attack could be used here to exceed byte limit.