Homework #1

Tuesday, October 9, 2018 3:00 PM

Andrew Laux
email: andrew_laux@ucsb.edu
cs177 on T R @ 12:30pm

Task 1 – Goals of Computer Security **(5 points)**

Classify each of the following acts (not necessarily related to computing infrastructures) as a violation of confidentiality, of data integrity, source integrity, or availability, or some combination thereof:

a) A hacker obtains millions of Yahoo passwords.

b) Anthony accidentally cuts the electricity from the server room.

c) The NSA finds an efficient method to break AES.

d) Anna registers the domain name "JohnSmith.com" and refuses to let John Smith buy or use the domain name.

e) Some malware encrypts the victim's hard drive with a secret key, and a criminal asks for a ransom to decrypt it.

f) The NSA wiretaps the cell phone of a suspect in a criminal investigation.

g) A foreign state actor finds a zero-day vulnerability for voting machines used in the US.

Hint: Answers may not be unique. Explain your answers as well as possible!

a) This is a violation of all of the above. Confidentiality since the hacker has access to user's private communications. Data integrity since the hacker can alter details about the email accounts, or delete emails. Source integrity since emails sent from such accounts may be from the hacker. Availability since the hack is able to restrict access to the accounts if they wish.

b) This is a violation of availability since clients expect to be able to connect to the server.

c) This is a violation of confidentiality. Since AES is a method of encryption, breaking AES would allow the NSA the ability to intercept messages. As a consequence, however, this could result in all types of violations depending on the contents of the encrypted messages and the actions taken by the NSA.

d) This is potentially a violation of source integrity depending on the context. If Anna has a husband named John Smith whom she runs a website for then no violation has occurred. But if a user was connecting to a site www.google.com they have a reasonable expectation that google is the owner of that site. If not then a violation has occurred. If anna has no reason to be the owner of JohnSmith.com other than to prevent John Smith's ownership then, I think, that would be a violation of availability as well.

e) This is a violation of confidentiality since the criminal has access to the victims hard drive. This is a violation of availability since the victim does not have access to their data. The criminal, with access to the hard drive, could also take action to violate data integrity by altering data on the hard drive and also even violate source integrity by posing as the victim using information on the hard drive.

f) This is a violation of confidentiality since the suspect expects that their private communications are not being listened to or recorded by a third party.

g) This is a violation of data integrity since a zero-day vulnerability would allow a foreign state actor to alter the data of said voting machines which are expected to be accurate representations of US voting results. If the make the machines unusable that would violate availability.

The goal of this task is to decrypt some ciphertexts generated with historical encryption methods which are only weakly secure.

a) The Shift Cipher (also known as Caesar's cipher) is a very simple symmetric encryption scheme. The secret key is a random integer k in the range $\{0, 1, \ldots, 25\}$. Then, to encrypt a certain text, we map each letter $x \in \{A, B, \ldots, Z\}$ to a new letter by moving k steps ahead in the alphabetical order, and possibly wrapping around if we reach the end of the alphabet. E.g., if $k = 2$, A is mapped to C, and Z is mapped to B, etc.

The ciphertext at `http://www.cs.ucsb.edu/~tessaro/cs177/hw/cipher1.txt` has been encrypted using the Shift Cipher – find the corresponding plaintext!

b) The ciphertext at `http://www.cs.ucsb.edu/~tessaro/cs177/hw/cipher2.txt` has been encrypted using a mono-alphabetic substitution cipher – find the corresponding plaintext!

c) (5 Bonus points) The ciphertext at `http://www.cs.ucsb.edu/~tessaro/cs177/hw/cipher3.txt` has been encrypted using a mono-alphabetic substitution cipher – find the corresponding plaintext!

Warning: You will get a positive number of points **only if** the decrypted plaintext comes with a detailed explanation of the procedure you used to obtain it. You are not allowed to use tools available on the Internet. If you write some code to help you out, please submit this code with the assignment. (Further instruction for code submission will follow.)

a)
IFYOUTRYANDTAKEACATAPARTTOSEEHOWITWORKSTHEFIRSTTHINGYOUHAVEONYOURHANDSISANON
WORKINGCAT

see code in task2_a.py
I printed out the string computed with all values of K from 1 to 26 and simply looked for the one with recognizable english.

b)
thechiefdifficultyalicefoundatfirstwasinmanagingherflamingoshesucceededingettingitsbodytuckedawayc
omfortablyenoughunderherarmwithitslegshangingdownbutgenerallyjustasshehadgotitsnecknicelystraig
htenedoutandwasgoingtogivethehedgehogablowwithitsheaditwouldtwistitselfroundandlookupinherface
withsuchapuzzledexpressionthatshecouldnothelpburstingoutlaughingandwhenshehadgotitsheaddowna
ndwasgoingtobeginagainitwasveryprovokingtofindthatthehedgehoghadunrolleditselfandwasintheactofc
rawlingawaybesidesallthistherewasgenerallyaridgeorfurrowinthewaywhereversheWantedtosendthehed
gehogtoandasthedoubledupsoldierswerealwaysgettingupandwalkingofftootherpartsofthegroundalicseo
oncametotheconclusionthatitwasaverydifficultgameindeed

see code in task2_b.py
I began this one with by filling substitutions according to the common frequency shown in the class slides. Through trial and error I found that the best strategy was to fill in the next few most common characters with numerical values and then attempt to complete small words. First 'the' and 'that' then larger words like 'getting' 'expression'. As more words got filled in it became earlier.

c)
THEYREDREADFULLYFONDOFBEHEADINGPEOPLEHERETHEGREATWONDERISTHATTHERESANYONELEFTAL
IVE

see code in task_2c.py
Same as the last one but required much more patience with the limited length of the string.

a) Does the following table describe a valid block cipher?[1] Justify your answer! (Rows and columns correspond to keys and plaintexts, respectively, and the table entry for row K and column X is the ciphertext $E(K, X)$.)

K / X	000	001	010	011	100	101	110	111
00	000	101	010	111	011	110	001	100
01	111	010	110	000	011	101	100	001
10	101	000	001	011	110	010	111	100
11	010	100	001	101	011	000	111	110

b) Consider the function $E' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ such that $E'(K, X) = X$ for all $X, K \in \{0,1\}^n$. Show that E' is a block cipher.

c) Consider the function $E'' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ defined as $E''(K, X) = \overline{X} \oplus K$ for all $X, K \in \{0,1\}^n$, where $\oplus$ denotes bit-wise xor, and where $\overline{X}$ is obtained by flipping all bits of X. Show that is E'' a block cipher.

d) Are E' and E'' secure block ciphers when $n - 128$? Justify your answer!

a) The table describes a valid block cipher, Enc(K, X) : $\{0,1\}^2 \times \{0,1\}^3 \to \{0,1\}^3$, that is the table takes 3 bit blocks and encodes them to 3 bit blocks over 4 different permutations, 1 for each key. The table is also one to one meaning that for a given key each block of plaintext maps to a unique block from the encryption alphabet so that Dec(K , C) results in X.

b) This is a function defined as E' = $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ It can be described asa block cipher where k (the number of bits in the key) = n and E'(K,X) = X so for all keys the function maps X to itself it is still one to one which statisfies the requirement for a block cipher. Recoving the the plaintext given the key is trivial since the ciphertext is the plaintext.

c) This function definedas E'' = $\{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ where Enc(K, X) = (!X) ^ K = C This is also a valid block cipher where k(the length of the key in bits = n) and the finction E'' is one to one because the bitwise XOR is a one to one function:

^	0	1
0	0	1
1	1	0

For a given bit bitwise XOR with either 1 or zero produce a unqiue output.
Decrypting using the Key is as simple as bitwise XOR and getting the compliment of the result: !(C ^ K) = X

d) E' would not be secure because the cipher text is the plaintext. E'' would be secure because with a K ∈ $\{0,1\}^n$
!X ^ K could be any a block C in the alphabet with a probability of 1/2^n.

Task 4 – Playing with AES (5 points)

We want to develop a better sense of the pseudorandomness of the ciphertexts generated by the AES block cipher. In particular, we will focus on the most commonly used variant with 128-bit keys. Let X be the 16-byte string consisting of

$$X = 10\ 04\ 20\ 18\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00$$

in hex-format.

a) What is the value of $\mathsf{AES}_X(X)$? Write the result in hex-format. Here, $\mathsf{AES}_K(M)$ is the ciphertext generated by AES on input key $K$and plaintext M.

b) Find a 16-byte plaintext M with the property that the last byte of $C = \mathsf{AES}_X(M)$ is equal to 00.
Explain how you have found it!

c) Find a 16-byte key K with the property that the last byte of $C = \mathsf{AES}_K(X)$ is equal 00.
Explain how you have found it!

Hint: Instructions on how to evaluate AES using Python are available on Piazza. Solutions using other programming languages are possible, as AES would behave in the same way.

a) 24f3dc26071110ad5258a4556714d01d

b) b'00000000000000000000000000000082'
I started with the plaintext of all bytes 00 and ran a loop which added 1 to the integer representation of the text. Then for each iteration I encrypt and check the last byte of the ciphertext. This plaintext was identified on the 347th iteration.
See task4_b.py for code used.

c) b'0000000000000000000000000000004b'
Same as above but this time incrementing the integer representation of the key by 1. This key was found on the 74th iteration.
see task4_c.py for code used.

Task 5 – There is only so much one can do (5 points)

Mr. Cipher is a deep undercover agent from the Republic of Cryptonia. Every day, he sends one of two messages back to base:

- M_1 ="Nothing to report",
- M_2 ="Meet me tomorrow at the rendez-vous point, I have information".

[1]Note that we are not asking whether the block cipher is *secure!*

To send these messages, Mr. Cipher uses counter-mode encryption based on AES with a 128-bit key. In particular, it encrypts the current date (using 8 bytes), followed by the ASCII encoding of one of the above two messages. Mr. Cipher uses the same secret key every day.

You work for the counterintelligence – you *know* the messages M_1 and M_2, you know the exact location of the rendez-vous point, and you are intercepting Mr. Cipher's ciphertexts. Your task is to predict *when* Mr. Cipher will show up at the rendez-vous location.

a) Describe a strategy to obtain this information.

b) How could Mr. Cipher protect himself from your attack strategy?

a) Both messages will use 16 byte blocks of encryption. Message two will require more blocks simply because it is a longer message. Then the longer of the ciphertexts will indicate that Mr. Cipher is meeting the following day. I discovered this by running the ctr code provided in the lecture slides with both messages.

b) Mr. Cipher should either send messages of similar length by padding the shorter, or include gibberish so that the messages are never the same in length.

task2_a.py

```
ciphertext = b'KHAQWVTACPFVCMGCECVCRCTVVQUGGJQYKVYQTMUVJGHKTUVVJKPIAQWJCXGQPAQWTJCPFUKUCPQPYQTMKPIECV'

for K in range(0 , 26):
    M = [chr(((i - 65) + K) % 26 + 65) for i in ciphertext]
    F = ''.join(M)
    print(F)
```

Screen clipping taken: 10/12/2018 12:10 PM

task2_b.py

```
import operator
ciphertext = 'qsgxsogubouuoxztqdatoxgukzlbaquowpqjapolnalaholhsgwutanolhkpsgpzxxggbgbolhgqqolhoqpmkbdqzxrgbajadxknukwqar
alphabet = list('abcdefghijklmnopqrstuvwxyz')
english_freq_spaced = list('etainohdsglrwucfypbmvkzxj-') #This took some trial and error
english_freq = [i for i in english_freq_spaced if i != " "]
result = {}
for a in alphabet:
    result[a] = ciphertext.count(a)
sorted_result = sorted(result.items(), key = lambda kv: kv[1], reverse=True)
freq = [i[0] for i in sorted_result]
print(freq)
decipher = {}
for i in range(0, 26):
    decipher[freq[i]] = english_freq[i]
print(decipher)
text_list = list(ciphertext)
plaintext_list = [decipher[i] for i in text_list]
plaintext = ''.join(plaintext_list)
print(plaintext)
```

Screen clipping taken: 10/12/2018 12:10 PM

task2_c.py

```
import operator
ciphertext = 'HAFWKFTKFBTEQUUWEXSTXEDFAFBTCSPGFXGUFAFKFHAFPKFBHJXSTFKCVHABHHAFKFVBSWXSFUFEHBUCYF'
alphabet = list('ABCDEFGHIJKLMNOPQRSTUVWXYZ')
english_freq_spaced = list('ETHARNDLOFIYPGSBWUV4----------') #This took some trial and error
english_freq = [i for i in english_freq_spaced if i != " "]
result = {}
for a in alphabet:
    result[a] = ciphertext.count(a)
sorted_result = sorted(result.items(), key = lambda kv: kv[1], reverse=True)
freq = [i[0] for i in sorted_result]
print(freq)
decipher = {}
for i in range(0, 26):
    decipher[freq[i]] = english_freq[i]
print(decipher)
text_list = list(ciphertext)
plaintext_list = [decipher[i] for i in text_list]
plaintext = ''.join(plaintext_list)
print(plaintext)
```

Screen clipping taken: 10/12/2018 12:11 PM

task4_a.py

```
from Crypto.Cipher import AES
import binascii
import sys
import struct

key = b'\x10\x04\x20\x18\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
plaintext = b'\x10\x04\x20\x18\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
encryption = AES.new(key, AES.MODE_ECB)
ciphertext = encryption.encrypt(plaintext)
print(binascii.hexlify(ciphertext))
```

Screen clipping taken: 10/12/2018 12:11 PM

task4_b.py

```
from Crypto.Cipher import AES
import binascii
import sys
import struct

key = b'\x10\x04\x20\x18\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
plaintext = b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
encryption = AES.new(key, AES.MODE_ECB)
plaintext_int = int.from_bytes(plaintext, 'big')

for i in range(0,600):
    plaintext_int += 1
    plaintext = plaintext_int.to_bytes(16, 'big')
    ciphertext = encryption.encrypt(plaintext)
    if(ciphertext[15] == 0):
        print(binascii.hexlify(ciphertext))
        print(binascii.hexlify(plaintext))
        print("On iteration: " + str(i))
```

Screen clipping taken: 10/12/2018 12:12 PM

task4_c.py

```
from Crypto.Cipher import AES
import binascii
import sys
import struct

key = b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
plaintext = b'\x10\x04\x20\x18\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00'
key_int = int.from_bytes(key, 'big')

for i in range(0,600):
    key_int += 1
    key = key_int.to_bytes(16, 'big')
    encryption = AES.new(key, AES.MODE_ECB)
    ciphertext = encryption.encrypt(plaintext)
    if(ciphertext[15] == 0):
        print(binascii.hexlify(ciphertext))
        print(binascii.hexlify(key))
        print("On iteration: " + str(i))
```

Screen clipping taken: 10/12/2018 12:12 PM