Final Exam

Thomas Lux

December 12th, 2013

1. Confidentiality: Only people with permission can access data.

   (a) Breaking into a password storage file and stealing passwords.

   (b) Listening in on communcations between two other parties. Not to be confused with Man-in-the-Middle, strictly listening.

   (c) Any breach of encryption by a 3rd party.

   Integrity: Communication is clearly with a valid source.

   (a) Phishing: Sending out information saying that you are another source, and requesting users click on a link or type in their password.

   (b) Man-in-the-Middle attack, where a 3rd party intercepts all communications between two communicants and has seperate key value pairs with each.

   Availability: Users should have the ability to access their data.

   (a) (Distributed) Denial of Service, when a computer or set of computers is used to repeatedly access a server/site such that no other users are able to access the server/site. This is attack is used to overwhelm a host.

   (b) DNS poisoning, domain name services are overwritten or diverted so that when the name of a given website is given, the user is erdirected to the incorrect IP address.

2. (a) Yes, because the same keys will be used to encrypt and decrypt the message that is being delivered.

   (b) A = 1
   B = Ceaser Encryption Key
   M = 26, the length of the alphabet

   (c) $((C_n - B) \times$ Multiplicative inverse $A)$ % M

   (d) Check out the program "prob2partd.cc"

3. (a) The difference between symmetric key cryptographic systems and public-key cryptographic systems is the number of keys used for interactions. Symmetric key cryptography has a seperate key for every communication, where as public key uses the same key for all communicationss.

   (b) In a symmetric key cryptographic system, since the same key is used for encryption and decryption and every message must have it's own unique key to be secure, this means that we would need $O(n^2)$ keys in order to communicate securely with everyone. Take the internet for example, with k servers, and n people, we would need $\frac{(k\times n)\cdot((k\times n)-1)}{2}$, but that's way too many!

In the public key system, we no longer need unique keys for all communications, everyone else just needs our public key so they can encrypt, and then only we can decrypt. A theoretical problem with this system is that if someone does manage to break our private key, and can use it, then they can now decipher all information that is delivered with us as the intended recipient.

(c) A programmer would be primarily concerned with some other aspects of the encryption, primarily the time required to perform these actions. Symmetric key encryption is much faster for encryption and decryption, so it is frequently the case that asymmetric encryption is only used for initial communication, then from there on out symmetric key encryption is used.

(d) One possible solution to the runtime of the asymmetric key encryption is to come up with a formula that factors any given number in constant time!! Then the whole internet would be insecure and broken!

4. Hmmmm ... beep beep, boop, bop, bop, booooooop, ...
$n = (491 \times 397) = 194927)$
$p = 491$ $q = 397$
phi(n) $= (p-1)(q-1) = 194040$
Encryption key: 47

EEA:

n1	n2	q	x	y
491	397	1	-34	42
397	94	4	8	-34
94	21	4	-2	8
21	10	2	1	-2
10	1	10	0	1
1	0		1	0

Multiplicative inverse: -34 or 194006
Either 47 or 194006 could be the public key with the other as the private key.

5. (a) Check out the program "prob5parta.cc"

(b) No, the hash function has neither confusion, diffusion, nor a considerable amount of entropy. The relationship between input and output is very simplistic, small changes in the input do not change the output drastically, and the output does look different than the input, but that is only a format, that can be easily read to be the same thing.

6. (a) SSL and TLS are able to authenticate with a process similar to the following:

Phase 1: Hello, nice to meet you
Client hello, security capabilities, random nonce
Server hello, security capabilities, random nonce

Phase 2: Server side communication
certificate delivery (signed by some trusted 3rd party)
public key
(optional) certificate request
hello done message

Phase 3: Client side communication
(optional) certificate
client key
(optional) certificate verification

Phase 4:
Client - change cipher done
Server - change cipher done
Begin encrypted comunication

Take note that this establishes a shared symmetric key to make the encryption faster. The program will use the random nonce information to generate the shared symmetric key. The server computes the key and sends it encrypted with the client's public key so only the client can know the symmetric key.

(b) Some additional mechanisms that must be in place for all of this to work are valid and compatible security capabilities on both the server and the client side of the communication. This also depends on the third party that authenticates the certificates to be entirely secure.

7. Check out the program "prob7parta.cc"

8. (a) The purpose of these attacks was to gain access to a different account that has higher privilege than the current user. This could gain access to files that were locked before.

   (b) The element that allowed shells to be opened with root privilege was that root was the one calling the open-new-shell command. When the current user opens a new shell, it opens the shell as the current user. This is where much of the "code injection" became useful. It allowed us to *trick* the root into opening a new shell for us.

9. (a) In python an Java, all of the memory allocation is done by the language, this ensures that users (even programmers) cannot overflow an array of allocated memory. In C and C++ this flaw was escaped with canary values, used to pad the return addresses after function calls ensuring that the return address was not overwritten during the call.

   (b) The primary expense of these mechanisms is time, checks must be made to ensure that values were not overwritten, and in Python and Java, more checks must be made to make sure that arrays are not being overfilled. All of these checks weighs down programs, and makes execution take longer. There is also some portion of extra space that is required as well.