CS 161- Homework#5

Student: Ninh DO
SID: 25949105

Problem 1

(a) The attacker can spoof a victim email and send millions of email to millions of dummy email addresses. Since these dummy addresses do not exist, the Stanford mail server will send millions of FDN messages back to the victim email.

(b) There are a couple of way to mitigate this issue:

i. The server can prevent spoofing by checking the match of the source email address (the attacker address) and the sender address (the selected victim address) in the sending request. If they do not match, drop the email.

ii. The server can recognize the pattern and use spam filter: if there is a huge amount of emails with the same content send to the non-existent addresses, filter all the emails.

(c) The attacker uses multiple botnets to send millions of emails to various addresses.

Problem 2

(a) The attacker N can stop the "No Such Domain" and replace it by a message that contains the attacker's server such as "This domain moved to www.evil.com". The victim may decide to try this name server.
Second attack is that the attacker sends his evil name server www.evil.com to the client, but the client resolver may reject it as it is not signed.
Third attack is that the attacker will create the same domain (with evil contents) that the victim requests and send the key of this domain to the victim. Finally, the attacker will stop the "No Such Domain" and replace it by the signed fake domain. The victim validates the signature by the key sent to him before, the victim also see that the domain main is what he requests without knowing that it is created by the attacker.

(b) 1. DoS attack: The attacker N can stop the signed message and send his own message of huge volume to overwhelm the client resolver.
2. The attacker N cannot launch the attack as in part (a), since the message is signed. If the attacker makes his own message and signs it using his key, the client's resolver cannot validate the signature and will reject the message.

(c) The server S returns "The hash of the requested name server comes after 80a4cb36… and c218f96a…"
The client will calculate the hash of abc.cs161.com and see that its hash is 99f3e2ba which falls between the above numbers.

(d) Yes, the attacker can determine whether the "names of interest" exist by conducting a dictionary attack by two ways:
Direct: try requesting names from the server S to see if they exist.
Indirect: Compute the hash values of all names of interest and inspecting the hash values returned by NSEC3 to determine whether the names of interest are in the domain.

(e) Changing the 'salt' value is to disable the computation of hash values of a batch of names offline. It also increases the work for the attacker, make the potential attacks more difficult. It maintains the protection against enumeration.

(f) The purpose of the iteration number is to make the computation of hash value more costly, thus deterring the attacker. It makes the attacker's enumeration more difficult.

(g) Too high iteration numbers can make the computation of hash values more expensive. Also, it can invite DoS attack against the server.

Problem 3

(a) The worm may change the number of bytes in the sequence other than 4, say 8-byte or 16-byte. In this case, the alarm does not work although there is a problem. The worm may also change the pattern of the 4-byte sequence after a time, then the alarm cannot recognize this new pattern.

(b) The architecture should be designed so that it recognizes the pattern and the number of bytes the worm generated repeatedly, then the box will alarm based on the collected patterns, instead of hard-coding the pattern and the number of bytes of the sequences into the hardware.
The design cannot completely eliminate the false negatives, since nothing is absolute, the worm can send varying-length packet over the link.

(c) We have 2^32 variation of 4 bytes (say, 4 bytes = 1 word).
In 1 sec, the traffic is 150 Mb = 150 * 2^20 bytes. Thus, there is 150 * 2^20 / 4 = 150 * 2^18 words.
The expected time is 2^32 / (150 * 2^18) = 0.01333 * 2^14 sec

(d) We have 2^64 variation of 8 bytes (say, 8 bytes = 1 word).
In 1 sec, the traffic is 20 Gb = 20 * 2^30 bytes. Thus, there is 20 * 2^30 / 8 = 20 * 2^27 words.
The expected time is 2^64 / (20 * 2^27) = 0.1 * 2^37 sec

(e) The worm author can use a random number generator coding in the worm so that each time the worm sends a packet, it generates a random number ‘n’ and send a n-byte packet.