SeungJi Lee
sl136

Homework 4

Problem 1

1. In the solution proposed in the paper, IDS reside inside physically but outside logically of the host.
2. VMM is a thin layer of software that runs directly on the hardware of a machine. It virtualized the physical hardware recourses allowing multiple machines to multiplex the resources of the physical machine. According to the paper, the security of the VMI IDS lies on the assumption that VMM is difficult to compromise.
3. The VMM has access to all the state of a virtual machine. Thus, VMM has the ability to inspect the hardware state of the virtual machine of the monitored host. VMM can obtain CPU state, all memory, and all I/O device state of the virtual machine.

Problem 2

1. Given physical access to the machine, it is easy to get the key and/or the plaintext. In the hibernate file, the page table of at the time of hibernation and the state of the encryption program must be stored. Then, examining the state of encryption program we can obtain the virtual address of the encryption key and/or the plaintext. Then, using the page table, we can obtain the physical address, and thus the location of the encryption key and/or the plaintext in the file.

   One possible way to fix this problem is to encrypt the hibernation file. When OS can encrypt the file with a key that is a hash of user password. Then, ask the user to authenticate when loading the hibernated states. This method should work since even if the attacker has the access to the physical machine, the attacker cannot decrypt the file unless the attacker has the user password or the key. Obtaining the key directly is almost infeasible. If the attacker has the user password, the attacker has access to the computer itself, so nothing can be done.

2. Like the above problem, we can encrypt the page that is being swapped. The encryption key can be obtained by cryptographic hashing the user password. This way the attacker cannot evaluate or obtain any useful information even if he gains the content itself if the attacker does not know the key or the password.
   Another way is to simply not use swap. If the page is faulted or needed to swap, simply check if any data in the page needs to be copied to the disc, and if not, simply dump it. This way, the attacker has no page file/swap to read.

3. We can simply make the OS to clean the memory block when it is released or is assigned to another process. Meaning we can simply write ‘0’ in the memory when the memory is released from a program, or is allocated to another process. This way, whenever the attacker’s process tries to access the memory, the OS will wipe out the data and assign it to the attacker’s process, preventing the leakage of the decryption key.

Problem 3

1. An adversary cannot forge a membership proof for an item that is not in the tree T, because the root hash is known and the root hash contains all information necessary for the entire file or sets of file. If the item is not in the tree, any membership proof of that item will not give the root

hash since the items at the first hashing is different, resulting different hashes for the consequent hashing.

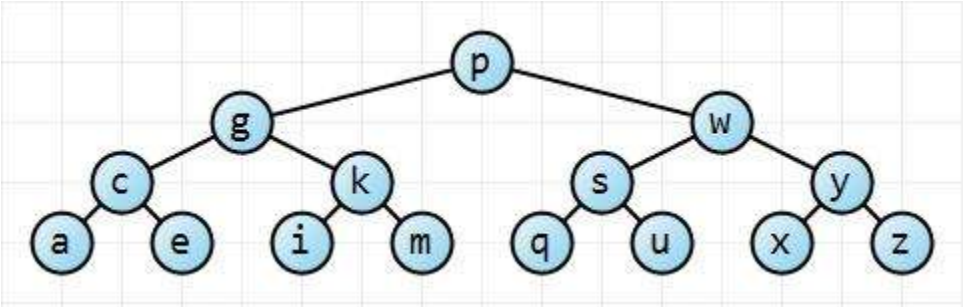
2. Here's the pseudocode:
   given a stack = [a, b, c, d], stack.pop() returns d, the last element.

   given stack = [(a(1), L/R), (a(2), L/R),.....,(a(k), L/R)]:
   int rank = 0;
   for(int i=k; I >= 1; i --){
   Let (h, D) <- stack.pop();
   if(D == L)
   rank += 2^(i-1);
   }
   return rank;

   The time complexity of this algorithm is O(k) since the algorithm goes through all elements in the stack, which must be in the order of k, the depth. All the operations in the for loop take constant time. Thus, the algorithm is in the order of k.

3. When we construct the hash tree, we can add empty leaf-nodes as many as the actual leaf-nodes. The empty leaf-nodes should be added to the end. Then, the hash tree will have one larger depth than the original tree. Now, any proof with rank greater than 2^(k-1) is non-membership proof.

Problem 4
A binary tree of symmetric-keys is maintained. Each key in the leaves is associated with one subscriber.



1. The subscriber maintains a key of its leaf-node, and the keys of the nodes in the path from its leaf to the root. Also the subscriber maintains the root key (the key associated with the root node).
2. The broadcast content is encrypted with the root key.
3. The broadcast content is decrypted with the root key.
4. When the subscriber is revoked, the keys associated with the subscriber's node, its path to the root, and the root are regenerated. Then they are encrypted with the keys associated with nodes that are not the subscriber's node, but has the same parent. Then those will be broadcasted. So if the subscriber's node is m, and thus has path, (m, k, g, p), then a message encrypted with w, a message encrypted with c, and a message encrypted with i will be broadcasted. Here the message encrypted is the new keys for nodes (m, k, g, p).

SeungJi Lee
sl136

5. The revoked subscriber cannot decrypt the content now since the root key is changed and thus the content is encrypted with different key. The subscriber may have received the encrypted messages with the new keys, but they are encrypted with keys that the subscriber do not have. Thus, subscriber cannot gain access to the content.

Problem 5

1. The employment of cross-realm authentication is useful in many settings. One of them is in the academic settings. Each university maintains its own library resources. With business contracts and other relations, the universities share their resources. It is beneficial to employ cross-realm authentication to share such resources.

2. The administrator can simply modify the ticket granting server in realm B to deny any tickets from realm A. This can be done since the cross-realm tickets contains information about the source realm.

3. There must be one shared keys per two realms. So there must be n(n-1)/2 keys.

4. This transitive trust causes weakening of the multi-realm Kerberos environments. For example, A and C do not have direct trust relationship. Yet, A can access resources in C through intermediate realm B. So, if A is compromised, both B and C can be compromised. It is also harder for C to block spurious tickets since B is not compromised and the ticket of A to C is granted from B.