SWEN90010 - 2019_SM1 High Integrity Systems

Assignment_1

Student:
Bowen Bai 969899
Zeming Yao 962403

Answer:

Question1:

When the log proofs are used, one thing we need to confirm is that the root hash of the Merkle tree should be reliable for the user to verify. If the attack could trick the user to accept a fake root hash, they may make modifications on the leaves as well. For Mozilla's solution, the certificate which is included in the CT log not only signs a domain, the head of the Merkle tree which summarizes the SHASUM256 files will also be attached to that domain. So, this makes it possible for the verifier to compare the special domain's first two 16 bytes code to the root hash to confirm that it is the correct one.

Question2:

The whole process of Mozilla's System is based on the irreversibility of SHA256 or SHA512 which means that the attackers cannot easily work out the binaries from the SHA256SUMS files at least impossible now. What's more, another basic of this system is that there should not be one even more collisions, they could trick the verifier to accept fake leaves made by them.

Question3

When the verifier is trying to verify an update, the first step is to obtain the certificate for that update and check whether it is trustworthy. For each release, the certificate will contain the Merkle tree head for that release and an SCT as well. So that the verifier could compare the SCT he received and the SCT in the trusted log to confirm this is release or update which has been added to the log. This is not enough, cause the verifier is still not able to confirm that the metadata comes from Firefox, so the next step is to compute the Merkle tree head with the receiving file, and then compare it with the one in the certificate. Without the second step, if the attacker cheats to get an untrustworthy certificate of their fake Merkle tree head, the verifier might also get a malicious release.

Question4

Instead of directly accepting an update it receives, the Firefox will firstly look up the certificate transparency log, decrypt the certificate by the public key to check if the update files come from a trustworthy source. Then the Firefox need to check if the files are actually the files signed. To confirm that, the Firefox need to compute the Merkel tree head and compare it with the head

which is signed and added to the CT log. Different from computing the Merkle tree head for a release, only the update file will need to be proved in the Merkle tree. If these steps could pass, Firefox would accept this update.

Question5

Before accepting an update or installing a new release, Firefox needs to check two things, one is these files have truly been added to a trusted log, the other is these files are not replaced by attackers. To achieve the second goal, Firefox needs to compare the Merkle tree head to the first two 16 bytes value of the domain in the trusted log. However, if there comes a bug, making the browser not able to check that condition, which means the incorrect Merkle tree head will not be rejected. Thus, attackers could replace the SHA256SUM files, and the browser will always accept the update or new release.

Question6

If there is a cheating certificate authority which can simply replace the update files with fake files and get them added to the trusted log, Mozilla cannot detect this kind of cheating. What Mozilla does is to compute the Merkle tree head according to the update SHA256 file and then to compare it with the one on the trusted log. However, the values from these two parts are both provided by the attacker which is the cheating certificate authority. Under this condition, Firefox would also accept that update and confirm that this file is on the trusted log and hasn't been replaced. So if Firefox chooses that authority, that could be a disaster.

Question7

According to the analysis of question7, Mozilla will accept the fake update file which is replaced by a cheating certificate authority without awareness. So it could be difficult for Firefox to check whether it gets a fake or a genuine update if the file is signed by a single cheating certificate authority. However, Mozilla could introduce another certificate authority into this system. To be specific, two certificate authorities could be used to sign the updates, as a result, the cheating authorities would certainly sign for a fake file made by themselves which makes the signed information of two certificates differ. In this condition, Firefox could detect that there must be at least one cheating certificate authority signed for a fake update. Thus, Mozilla could take more actions to tell who cheated. If more authorities are introduced in this system, the more this system would be reliable.