Integration Testing, Task 2.8

''' nisargpatel@lab1:~/tpm_attested_container$ python3 attest.py --init

tpm initialized successfuly

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 1
   1
   Instance ID: vm1
   Path to instance public key: id_tpm.pub
   SUCCESS: instance vm1 registered

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 1
   1
   Instance ID: vm2
   Path to instance public key: id_tpm.pub
   SUCCESS: instance vm2 registered

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 2
   2
   Instance ID: vm1
   instance vm1 verified

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 3
   3
   Instance ID: vm2
   Number of verification rounds: 3
   Delay between rounds (seconds): 1
   round: 1 succeeded
   round: 2 succeeded
   round: 3 succeeded
   Success: continous verification passed for instance vm2

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 4
   4
   Instance ID: vm1
   SUCCESS: Instance vm1 removed

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 2
   2
   Instance ID: vm1
   FAILURE: instance vm1 not registered

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 2
   2
   Instance ID: vm2

instance vm2 verified

Select an action:

1. Register instance
2. Verify instance once
3. Enforce continuous verification policy
4. Remove instance
5. Exit Enter option: 5
   5
   nisargpatel@lab1:~/tpm_attested_container$ ""

Security Analysis and Design Discussion

3.1 Security Benefits

Discuss two security advantages of TPM-backed attestation over traditional password-only authentication.

1. TPM backed attestation relies on hardware protected keys that can't be exported, this is advantageous over passwords because if an attacker got access to passwords they can compromise other systems. However since TPM keys cannot leave the TPM, the attack surface is smaller.
2. The challenge and response protocol with fresh nonces prevents replay attacks. If an attacker gets access to a password, they can be reused until the password is rotated. With TPM attestation, each verification uses a new nonce and so a new challenge. Capturing the previous signature is not worth anything.

3.2 Limitations and Possible Attacks

Discuss two realistic limitations or attacks that remain possible even when TPM-based attestation is deployed. Explain why these attacks are still feasible despite mitigations.

1. One limitation is that TPM attestation only proves that the instance holds a valid key. If the VM is compromised, and running malware the VM could still pass attestation. A PCR measured boot could detect changes in software as a mitigation though
2. In my implementation of VM attestation, theoretically a malicious VM could intercept the challenge, forward it to the real VM to sign, get the response and forward it to the TPM. The channel the VM and the TPM talk on isnt encrypted.

3.3 Realistic Deployment Recommendation

Scenario: Your organization must deploy TPM-based attestation in production across 1,000 VMs.

Question: Propose one improvement that would strengthen security or operability of thisdeployment. Explain how you would implement it, outline the key trade-offs, and estimate theimplementation complexity with a brief justification

Binding attestation to PCR values would address verifying boot integrity. The PCR registers can measure the firmware, bootloader, kernel, and other configs so the TPM can verify the VM booted to a known configuration

Implementation: we can use tpm2_quote to generate a signed quote that includes PCR values along with the nonce. The verifier maintains the golden set, on attestation, the TPM can compare the quoted PCR with the expected value and if they dont align then the boot was altered.

Trade offs: Management at scale is a challenge, with every update to the kernel or firmware, or any config changes there will be updates to the PCR values. This must be accounted for across the 1000 VMs.

Complexity: It should not be too complex, the swtpm supports quotes and reading PCRs. The harder part is the operational side of things. There has to be a server to manage expected PCR values, an integration with the pipeline to build 1000 VMs and rollout to update expected values before pushing software updates so there arent unexpected issues booting.