**SRI TEJA KALE**

**G01501801**

**HOMEWORK -3**

1. **Implementing transparent IP level encryption & authentication via divert socket (80 points)**

**Experiments**

Once you have developed such program, you need to run two instances of the FreeBSD VM: VM1 & VM2 and to make VM1 & VM2 have two different IP addresses (e.g., ip1, ip2). They should be able to ping to each other. On VM1, set up appropriate ipfw rules to divert incoming/outgoing traffic from/to VM2 to . Similarly, on VM2, set up appropriate ipfw rules to divert incoming/outgoing traffic from/to VM1 to <divert port>

1. **In order to verify the ip\_cryptAuthAll application, i installed two FreeBSD virtual machines (VM1 and VM2), both communicating with each other using encrypted and authenticated IP traffic.**

**Configuration Information:**

VM1 address: 192.168.109.128

VM2 address: 192.168.109.129

Divert port used on both VMs: 5000

Shared key: secretKey

Both machines used the same encryption key (secretKey) to ensure the successful decryption and authentication of packets.

**IPFW Rules**

To direct the appropriate traffic to the ip\_cryptAuthAll application, on each VM i installed and executed the following IPFW rules:

**On VM1 (192.168.109.128):**

ipfw add 100 divert 5000 ip from any to 192.168.109.129

ipfw add 110 divert 5000 ip from 192.168.109.129 to any

**On VM2 (192.168.109.129):**

ipfw add 100 divert 5000 ip from any to 192.168.109.128

ipfw add 110 divert 5000 ip from 192.168.109.128 to any

This setup allows for all incoming and outgoing IP packets between the two VMs to be intercepted and directed into the divert socket (port 5000) for the ip\_cryptAuthAll application to process.

Once we created the environment and executed the program on both VMs using the same key, we were able to decrypt all incoming packets to confirm that the keys matched on both ends and that the encryption and authentication mechanism worked properly.

To compile the ip\_cryptAuthAll program, the following gcc command was used:

**gcc -o ip\_cryptAuthAll ip\_cryptAuthAll.c md5.c ip\_checksum.c divertlib.c -lcrypto**

I have used port 5000 to divert the packets in VM1. The VM2 IP address is 192.168.109.129. In the below screenshot, we can observe that the incoming packets are being decrypted. This indicates that the encryption key used on both VMs is the same, confirming proper encryption, authentication, and decryption functionality.

Command: **sudo ./ip\_cryptAuthAll 5000 192.168.109.129 secretKey**

The outcome provides the following results:

* **Packets successfully processed**
* **Successful authentication**
* **Payloads decrypted correctly**

This shows that the ip\_cryptAuthAll is working correctly on VM1**.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**VM1**

I have used port 5000 to divert the packets in VM2. The VM1 IP address is 192.168.109.128. In the screenshot below, we can observe that the incoming packets are being successfully decrypted. This confirms that the key used in both VMs matches, ensuring that encryption and authentication are working correctly.

Command : **sudo ./ip\_cryptAuthAll 5000 192.168.109.128 secretKey**

The output from the terminal shows:

* **Successful processing of incoming packets**
* **Authentication success messages**
* **Decrypted payloads being correctly processed**

This confirms the correct setup of ip\_cryptAuthAll on VM2 and demonstrates that both VMs can securely communicate using the same shared key.

A screenshot of a computer

AI-generated content may be incorrect.

**VM2**

**b)** **For the second task, we purposely ran different encryption keys on VM1 and VM2 to see if decryption and authentication failed as intended when there are mismatched keys.**

On VM1 (192.168.109.128), the following command was executed:

**sudo ./ip\_cryptAuthAll 5000 192.168.109.129 secretKey**

On VM2 (192.168.109.129), a different key (secret) was used:

**sudo ./ip\_cryptAuthAll 5000 192.168.109.128 secret**

As shown in the screenshot captured below, the outgoing packets from VM2 are not authenticated and decrypted on VM1. This was expected behaviour and supports that the ip\_cryptAuthAll application is correctly implementing key-based authentication. Since VM1's keys are different than VM2's keys, authentication fails, and the payloads are not decrypted.

This result confirms the integrity and security of the system only allowing and processing traffic encrypted with the correct share key.

A screenshot of a computer

AI-generated content may be incorrect.

**VM1**

From the below screenshot we can observe that the Input packets are not getting Authenticated.

A screenshot of a computer

AI-generated content may be incorrect.

**VM2**

1. **Short Answer Question (10 points)**

**If two hosts are using the above transparent IP level encryption & authentication, would the communication between those two hosts subject to the man-in-the-middle attack? Why?**

Communication between two hosts using transparent IP-level encryption and authentication can still be vulnerable to man-in-the-middle attacks, especially if key management is poor or insecure.

Having encryption and authentication at the IP level ensures confidentiality and integrity of the data exchange but does not provide any protection against an attacker intercepting one or more packets if the attacker is on the same network path (e.g., a compromised router or a switch). If an attacker gains access to the shared encryption key, they are still able to decrypt and alter, and then re-encrypt any packet (message) before forwarding it, in a manner which is invisible to both hosts.

Transparent IP encryption also usually does not offer mutual authentication or secure key exchange, which may provide a defense against MITM attacks as both parties will know the other is who they say they are as well as that they use a shared secret, regardless of what that secret is. If an attacker compromises the key-sharing process, they can impersonate either host, and this could mean using a bad encryption key, or simply poor key distribution.

So much depends on what type of security additional to IP level encryption will be in use, such as a Public Key Infrastructure, secure key exchange protocols such as Diffie-Hellman, or Certificate Authentication, in which both hosts will assume they are safe from possible MITM attacks when used in addition to IP level encryption.

1. **Short Answer Question (10 points) If two hosts are using the above transparent IP level encryption & authentication, would the communication between those two hosts subject to the replay attack? Why?**

While it is always best to implement other measures, communication between two hosts using transparent IP-level encryption and authentication, can also be susceptible to replay attacks. Transparent IP-level encryption guarantees confidentiality, and authentication ensures integrity, but neither method prevents replay attacks from a strictly technical standpoint.

A replay attack occurs when a valid packet is captured and resent later by the attacker in an attempt to convince the recipient to accept old data as new. The receipt of the legitimate encrypted packet refers to the original packet with the assurance of confidentiality and integrity, however, it does not indicate faux packet freshness; again, there, no indications intelligence has built a reference model of received packets. Thus, the receiving host could accept the replayed packet as valid, and potentially maliciously, if the encryption key were unchanged, it is simple to understand how malicious actors can leverage replay attacks.

To mitigate against replay attacks, the protocol must leverage other attack detection mechanisms such as; sequence numbers, timestamps, nonces, or session tokens, which separate new packets from packets already seen for reception, and thus the duplicate packets will be rejected. Again, even if an IP packet was encrypted and validated, the encrypted and validated packet does not give the receiver the assurance that has never previously not been received, thus both the encrypted and validated purpose has misappropriation if a system does not actively prevent replayed packets from being exploited as legitimate system requests.

In summary, although transparent IP-level encryption and authentication create a protective layer, the protocol must contain additional methods of removing replayed attacks to ensure the fidelity of your residuals concerning packets being legitimately exploited.