CSCI 530: Security Systems
Lab 9: Confidential Communication in Tunnels, Encryption and VPN; Submission Due: December 6th 2024.
Name: Anne Sai Venkata Naga Saketh
USC Email: annes@usc.edu
USC ID: 3725520208

Question 1:
In the topology diagram for this experiment there are 2 pairs of arrows showing where an encrypted tunnel's endpoints might be placed-- either on nodes 1 and 3 or on nodes 1 and 4. The experiment included examples of both usages. ssh and stunnel did it the first way and OpenVPN used the second. Does it matter? Which tunnel endpoint option is "better," node3 or node4? Answer in two parts 1a and 1b per below.

a. Considering the fact that the application clients in this experiment are always housed on node4, choose one of the tunnel endpoint options and make the case for why it is the better one.

b. Imagine that diagram's upper network 10.1.1.0/24 were expanded to 100 or 200 nodes. Would any new or different rationales and arguments for placement of the tunnel endpoint arise? Considering the fact that the population of machines that could potentially benefit from secure communication is large, choose one of the tunnel endpoint options and make the case for why it is the better one.

Answer:

a. Here's the reason why tunnel endpoint on nodes 1 and 4 is the better choice: it ensures end-to-end encryption for all communication, including nodes 2 and 3. Since the application clients are on node 4, we can set up a tunnel directly to node 4, which means client-server communication is completely secure. If we set the tunnel endpoints between nodes 1 and 3, traffic between node 3 and node 4 would be unencrypted, leaving it vulnerable to eavesdropping or tampering. So, having the endpoint on node 4 gives us the best protection for keeping client data safe and private.

b. So, if the network expands from 10.1.1.0/24 to 100–200 nodes, the importance of putting the tunnel endpoint at node4 becomes even more crucial. With a bigger network, data has to travel through more hops, which means there's a higher chance of exposure if not encrypted end-to-end. If we use node3 as the endpoint, traffic between node3 and node4 would be exposed across these extra hops. But if we use node4 as the endpoint, encrypted communication can reach all potential client nodes, reducing risks even in a larger network.

Question 2:
We implemented 3 products that perform encryption but said little about how they do it. Do they all use the same cipher algorithm in common? If not, does each one use the same algorithm every time? Could we say that one product is more secure, in the encryption it applies, than another? Key sizes? Encryption modes? HMACs? Do a little bit of research, simply at the level of reading the man pages to find out what they have to say about this ("cipher" is a good search key). Then write a very short summary of how things work and what you find to be the salient issues discussed in the man pages. For your convenience here are the man pages: ssh sshd stunnel openvpn

You could go deep into this but that's not what we want. Just demonstrate having visited the man pages and awareness what the issues are.

Answer:

i. SSH uses a bunch of encryption methods, like AES, Blowfish, 3DES, and ARCFOUR. You can choose the key size, which goes up to 256 bits. It also checks data integrity using HMACs. You can change the ciphers and MACs by editing configuration files, so you can pick the encryption options that work best for you.

ii. SSHD (the SSH server) picks encryption ciphers for each session based on its settings. It doesn't just use one algorithm, it supports a bunch of ciphers like AES with keys up to 256 bits long. HMACs keep things safe, and admins can tell SSHD which ciphers they want to use in the configuration file.

iii. Stunnel uses SSL/TLS protocols to encrypt data. It supports ciphers like AES, DES, IDEA, RC2, and RC4 in modes like CBC and CFB. Stunnel also uses certificates for authentication and lets you choose the key size, but the exact details depend on how SSL/TLS is set up.

iv. OpenVPN uses OpenSSL to encrypt data, giving you options like AES, Camellia, Blowfish, and CAST. It also lets you choose different key lengths and uses HMAC for authentication. You can also customize the encryption settings in the configuration files to meet your specific security needs.

Question 3:

For servers in this experiment we used an echo server and an http server on node0. As tests of node4-to-node0 connectivity in the above instructions we sometimes tried echo, sometimes http. Sometimes we even tried ping. When it came to stunnel in particular, we tried http. We found it went through successfully. However we never tried echo nor ping. If we'd added those extra tests would they have gone through or not? Are you aware of any arrangement that could broaden coverage to *all* traffic seeking to pass from node4 to node0 so that, independent of service or protocol, anything addressed between the two nodes would be successfully encrypted and tunneled through?

Answer:

In this experiment, the stunnel setup is limited to HTTP traffic between node4 and node0. So, protocols like echo or ping, which don't use HTTP, won't be tunneled or encrypted through the stunnel. To support all traffic types, regardless of the protocol or service, we need a more comprehensive tunneling approach. One way to do this is to configure OpenVPN as a VPN tunnel. This would allow us to encrypt the IP level, which means it can encapsulate and secure all traffic between the nodes, including echo, ping, and other services.