Cyber Risk Assessment Project - Sniffer

Deadline: published in Moodle

In this project, you will practice working with network protocols, network stack, and individual packets. You will create a packet sniffer to affect data flow and a solution for mitigating the sniffing. For this assignment, it is recommended to use Python, version 2.7.x only, with the Scapy module[1] or PyPCAP[2]. See also the documentation for handling raw sockets[3].

For submission, create a *.zip* archive containing the final source code and a *.pdf* file with explanations and screenshots that demonstrate each task.

Task 1

Create a network service that listens on UDP port 12321, which accepts connections on that port and echoes complete lines (using a CR/LF sequence as line separator) back to clients. No elaborate error handling is required. For the purposes of testing, it is only necessary to support connections from localhost (127.0.0.1, or perhaps ::1). Logging of connection information to standard output is recommended.

Create a client connecting to this service via UDP. Every 3 seconds the client sends some data (random text, a poem or whatever you choose) and waits for the server to echo it. The data to be sent is split into packets of small size — e.g., at most 100 bytes. Each UDP packet contains a sequence number and the appropriate data itself.

Create an adversary which listens (on localhost) to the connection between the client and the server, and prints the packet contents (sequence number and data). Note that the adversary is not a part of the connection, and hence must sniff packets using raw sockets or suitable packet capture library.

Task 2

Note that an active adversary may prevent packets with chosen sequence numbers from arriving to their destination.

The following instructions provide a solution for the client to mitigate this kind of attack.

1. Client and server fix a random number d.
2. The client calculates $e = x_{\pi_1} \oplus x_{\pi_2} \oplus \cdots \oplus x_{\pi_d}$ where the π_i are the packet indexes.
3. The client sends a packet containing e and the indexes as a payload in addition to each d original packets. In this protocol, if a packet is missing it can be calculated using the already received packets.

Modify the client and the server in order to implement this solution. Show with appropriate screenshots that if the adversary denies from specific packet to its destination, the server can retrieve the missing data packet. Packet loss can be simulated in the client.

---

[1] http://www.secdev.org/projects/scapy/doc/usage.html
[2] http://pypcap.readthedocs.io/
[3] https://docs.python.org/library/socket.html

Task 3

Implement the client, server and the adversary as separate bridged VMs on different IPs. Illustrate this in the submitted report.