Project Proposal : CS3543 - Computer Networks 2
Mini-Nmap

Harsh Agarwal
cs15btech11019

S. Vishwak
cs15btech11043

1 Objective of the project

The objective of this project is to implement a native version of Nmap - focusing more on various kinds of host discovery and port-scanning algorithms.

2 Description of the project

2.1 Host Discovery

One of the very first steps in any network reconnaissance mission is to reduce a set of IP ranges into a list of active or interesting hosts. Scanning every port of every single IP address is slow and usually unnecessary. The idea in host discovery is to send *concurrent* `ICMP` messages to all the hosts in a network, and continue to do so until we have found a list of active IPs in the subnet. Host discovery is sometimes called ping scan. `Nmap` provides a lot more features to host discovery - which will take a lot of time to implement in entirety, and hence we are focusing on `ICMP` based host-discovery with certain options obtained by configuring the `ICMP` packet.

2.2 Port Scanning

A very popular method to perform port scanning is using the `TCP SYN` field, and is also called the `TCP SYN` scan. Here the host which is scanning (the scanner) sends a `TCP` packet with the `SYN` bit set and waits for either one of a `SYN/ACK` or a `RST` to detect open ports. We will be implementing this method along with other variants which are briefly described below:

- `FIN` scan ⇒Send `TCP` packet with `FIN` bit set and wait for FIN or RST response to detect open ports.
- `Xmas` scan ⇒Send `TCP` packet with `FIN`, `URG`, `PUSH` bit set and wait for `FIN` or `RST` response to detect open ports.
- `Null` scan ⇒Send `TCP` packet with no special / flag bits being set and wait for `FIN` or `RST` response to detect open ports.

There are also some indirect scanning techniques. The idea is to use the (spoofed) `IP` address of a third host to disguise the real scanning system. Since the scanned host will react by sending or not certain segments to the spoofed host, all that is needed is to monitor the `IP` activity of the spoofed host to know the results of the original scan. Our implementation will include three methods in indirect scanning:

- Dumb Scan ⇒Use a victim computer to scan target computer ports.
- Decoy Scan ⇒Send many packets with spoofed `IP` addresses and 1 genuine address. This makes it hard to pin-point the “attacker” IP.
- `identd` Scan ⇒By exploiting the Identification Protocol, retrieve the username (Userid) that owns the daemon running on a specified port.

We also plan on implementing an ARP poisoning technique. This tool is divided into 3 parts.

- The ARP Poisoner module responds back to every ARP request made by the target with scanner's own MAC address.
- The Packet Sender sends UDP packets with source port as the port being scanned.
- The Sniffer captures all the UDP and ICMP packets coming to the scanner and predicts whether port is open or closed.

3 Contributions from members

- Harsh: TCP SYN scan, Xmas scan, Dumb scan, ARP Poisoner and Sniffer, identd Scan.
- Vishwak: Host Discovery techniques using ICMP, FIN scan, Decoy scan and ARP packet sender.

Please note that this list is temporary and could evolve during the course of the project. Since we are using GitHub as a VCS, evaluators will be able to note contributions from there.

4 Technologies that will used and deliverables

Technologies

For testing purposes, we will create a private subnet on an OpenStack setup and spawn many VMs on this subnet. We will open multiple ports on all VMs and test our tool on them. For version control, we will be using GitHub, and WireShark and/or tcpdump for capturing packet traffic and behaviourial aspects. We hope to also use Prometheus as a monitoring system. For coding, we will preferably stick the Linux network API in C / C++ and occasionally use Python as well. Paper references are specified in the end of this document.

Deliverables

At the end of the project, we will submit the source code. Using WireShark and/or tcpdump, submit flow graphs of conversation taking place between scanner and victim. Prometheus will be able to provide information regarding traffic ingress and egress on the victim. We will also provide a small report discussing about the accuracy of various port scanning techniques, since modern OSes have checks to prevent canonnical port-scanning techniques.

5 Scope for Improvement and future work

All the above listed items will be implemented definitely as a part of the project. In addition - provided time permits, we hope to implement a few ideas specified below:

- Port Scanning Detection techniques ⇒how to find out if an open port is being scanned?
- Stack Fingerprinting and OS Detection ⇒detecting operating systems using the format of the reply obtained from the client. This is possible due to certain (un)specifications in the RFCs.

References

M. Dabbagh, A. J. Ghandour, K. Fawaz, W. E. Hajj, and H. Hajj. Slow port scanning detection. 2011.

Marco de Vivo, Le Ke, Germinal Isern, and Gabriela O. de Vivo. A review of port scanning techniques. 1999.

Sumit Kumar and Sithu D Sudarsan. An innovative udp port scanning technique. 2014.

Gordon Fyodor Lyon. *Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning.* 2009.