Síťové aplikace a správa sítí

Projekt 1:

Jednoduchý síťový TCP/UDP scanner

Matúš Kontra (xkontr00)

xkontr00@stud.fit.vutbr.cz

1.Úvod do probematiky

Při programováni nízkourovnovej sietovej komunikácie je kľúčová znalosť problematiky raw socketov, najma při odosielaní a dalej metóda zachytávania príchodzej komunikácie, v našom prípade knižnica libpcap. Dalej potrebujeme vediet jako funguje komunikácia protokolom tcp, udp, icmp a pakety ktore generujú.

1.1 BSD Raw Sockets

Na odosielanie IP datagramov budeme pouzivat socket vytvorený volaním socket knižnice BSD Sockets s vhodne yvolenými parametrami – jako protokol zvolime cislo protokolu ktorz chceme odosielat (IPPROTO_TCP, IPPROTO_UDP) a jako tzp socketu typ SOCK_RAW, ktorý zamedzí vytvoreniu hlaviciek vzssej vrstvy. Neskor musime v parametroch socketu nastavit aby nepribaloval hlavicku IP.

1.2 LibPcap

Na zachytavanie pouzivame kniznicu libpcap, ktorá nám umoznuje prijímat lubovolne pakety prichadzajuce na specifikovane rozhranie. Problémom je ze nevie filtrovat vsetky rozhrania naraz – musíme vediet odkiaľ budú prichádzať odpovede. Po nakonfigurovaní – čo obnáša vzhľadanie zariadenia (pcap_findalldevices), vytvorenie sessionu a nakonfigurovani filtra, sa prijate pakety spracúvajú v callbackovej funkcii ktorá je volaná pre každý príchodzí paket.

1.3 TCP, UDP, ICMP

Pri TCP SYN scanovaní je doležité aby sme vedeli ako prebieha handshake, a ký typ odpovede máme očakávať. Zaujímať nás budú tcp pakety s nastavenými príznakmi SYN a ACK, ktoré naznačujú že cieľ počúva a chce pokračovať v handshake.

UDP scan prevedieme odoslaním prázdneho udp paketu na adresu cieľa. Očakávaná odpoveď je žiadna alebo ICMP paket Destination port unreachable.

2. Návrh aplikácie

Aplikácia pozostáva z niekoľkých základných častí. Prou zložitjšou úlohou je sprarsovanie parametrov príkazového riadku a naplnenie zoznamov portov. Toto je vyriešené použitím metódy getopt a konečného automatu na spracovanie rozsahov portov. Dalej bolo treba vyriešiť prístup k socketu na odosielanie – na toto bola vztvorená samostatná trieda jazyka c++ ktorá ho encapsuluje. Prevedienie scanov tcp a udp je rozdelene do dvoch fukcii ktora sa kazda stará o vytvorenie threadov vykonavájúcich cinnost spojenu s odosielanim paketov na scanovanie. Prijímanie odpovedí je riešené v samostatnej metode. Po dokončení scanov sa zo zoznamov uchovavajucich zoznam portov a ich stav vypisu relevantné informácie.

3. Implementácia

Po naplnení zoznamov a konfigurácii pcapu sa spustí vlákno sniffera ktorý odchytáva pakety zo specifikovanej lokálnej adresy. Vlákná sú vytvárane knižnicou lpthread – štandardná unixová knižnica na prácu s vláknami. Po tomto sa zavolá metóda tcp syn scanu. Táto iteruje cez zoznam portov a

vztvára vlákna, kým není dosiahnutého maximálneho špecifikovaného v programe. Po tomto yačne blokovať, kým jedno z vlákien nedokončí cinnosť – metóda odblokuje a pokračuje dalej kým neprejde cez všetky porty. Potom čaká na dokončenie všetkých vlákien. Následne sa volá metóda, vykonávajúca udp scan, ktorá je založená na podobnom princípe. Po jej dokončení sa vypíšu obidva zoznamy spolu so stavmi portov. Následne dojde k zastaveniu vlákna sniffera, uvlneniu systémových zdrojov a ukončeniu aplikácie

4. Použitie

Parametrami pri spustení sú pt, a pu spolu s adressou ako je popísane v zadaní.

scan [-i <interface>] -pu <port-ranges> -pt <port-ranges> [<domain-name> | <IP-address>]

,kde:

- i špecifikuje rozranie kde sa ma sniffovat
- pu rozsah portov udp scanu
- pt rozsah portov tcp scanu
- adresu ciela

Odpoveď programu je vo formáte:

TCP:

port => stav

UDP:

port => stav

Podrobnejšie je volanie rozobraté v Readme

5. Záver

Vo výsledku sa jedná o použiteľnú a celkovo efektívnu aplikáciu port scannera, ktorá však okrem základných funkcii veľa neumožnuje takže jej nasadenie pri auditingu je nepravdepodobné.