VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IPK PROJEKT 2

Varianta ZETA: Sniffer paketů

Jakub Hlava (xhlava52)

Obsah

Úvod do problematiky

Cílem projektu je implementace síťového analyzátoru, který je schopný zachytávat pakety na uživatelem určeném rozhraní, tyto pakety filtrovat podle několika vlastností, získat základní informace o paketu a na základě nich jej korektně vypsat na standardní výstup.

Souhrn informací nutných pro realizaci

O drtivou většinu problematiky zachycení paketu se starají funkce knihovny `libpcap` samy o sobě.

Důležitá je poté struktura zachyceného paketu. Předpokládáme záchyt pouze paketů na ethernetové vrstvě, tedy každý paket začíná hlavičkou ethernetového rámce. Z této hlavičky nás nejvíce zajímá část `EtherType`, která se nachází na 13. a 14. bytu a lze z ní vyčíst, jaká data ethernetový rámec obsahuje. Pro účely našeho projektu nás nejvíce zajímají hodnoty `0x0800` (IPv4), `0x86DD` (IPv6) a `0x0806` (ARP). V případě detekce ARP protokolu je zajímavý pro výpis paketu i zbytek hlavičky, tedy MAC adresa cíle na 1. – 6. bytu a MAC adresa zdroje na 7. – 12. bytu.

Pokud při zpracování ethernetové hlavičky dojde k detekci protokolu IP, pak je nutné z jeho hlavičky získat další informace – adresu zdroje, cíle a protokol dat. Velikost a umístění těchto informací se liší podle verze IP protokolu, tu získáme, jak již bylo uvedeno výše, z `EtherType`.

V případě IP verze 4 jsou adresy 4bytové, v IP hlavičce se adresa zdroje nachází na 13. – 16. bytu a adresa cíle na 17. – 20. bytu, identifikátor protokolu se nachází v 10. bytu.

Hlavička IP verze 4 má proměnnou velikost, kvůli nepovinnému poli `Options` s proměnnou délkou. Tento fakt je nutno zohlednit při zjišťování zdrojového a cílového portu protokolů TCP a UDP, délku hlavičky lze získat pomocí pole `IHL` (Internet Header Length), které se nachází ve spodní polovině 1. bytu. Přítomnost pole `Options` značí hodnota IHL větší než 5 a obecně IHL udává počet 32bitových slov v hlavičce, velikost hlavičky v bytech tedy získáme jako IHL * 4.

Hlavička IP verze 6 má pevnou velikost, odpadá tedy nutnost výpočtu její velikost a stačí získat zdrojovou a cílovou adresu a protokol dat. IPv6 adresa má 16 bytů, v hlavičce se zdrojová adresa nachází na 9. – 24. bytu, adresa cíle na 25. – 40. bytu a identifikátor protokolu se nachází v poli `Next header` na 7. bytu.

Identifikátory protokolů jsou stejné pro IPv4 i IPv6 a pro účely projektu jsou důležité především: `0x01` (ICMP), `0x06` (TCP) a `0x11` (UDP).

Na závěr u protokolu TCP a UDP je nutné získat zdrojový a cílový port. Data TCP i UDP segmentů se nachází za IP hlavičkou, porty jsou 16bitová čísla na začátku TCP i UDP segmentu, tedy zdrojový port lze získat z 1. a 2. bytu, dílový z 3. a 4. bytu.

Implementace

U implementace této úlohy jsem se rozhodl pro jazyk C++ a knihovnu `libpcap`.

Program si nejprve zjistí všechna dostupná rozhraní v systému, aby je v případě, že uživatel nespecifikuje rozhraní pro záchyt paketů, mohl uživateli vypsat nebo v případě, že uživatel rozhraní zadá, aby mohl ověřit, že skutečně takové rozhraní existuje.

Poté přichází na řadu příprava filtrace paketů, k té využívám filtry z `libpcap` (volání `pcap_compile` a `pcap_setfilter`), proto příprava filtrů spočívá pouze ve složení řetězce podle spouštěcích parametrů snifferu a přeložení tohoto řetězce na filtrační program. Útržky těchto filtračních programů jsou uloženy v makrech v hlavičkovém souboru `main.hpp`.

Samotné zachytávání paketů se odehrává v cyklu, dokud se nezachytí uživatelem specifikovaný počet paketů. Po zachycení paketu se ze struktury vyčte čas přijetí, který je převeden na formát dle zadání pomocnou funkcí `make_timestring`. Poté se částečně manuálně, částečně pomocí pomocných funkcí `extract_ip`, `extract_ports` a `extract_arp` z paketu postupně získá `EtherType`, podle něj se získají IP adresy, příp. MAC adresy zdroje a cíle u ARP, poté se v případě IP protokolu získá protokol dat a v případě TCP a UDP ještě zdrojové a cílové porty.

U protokolu ICMP a ARP není v zadání uvedeno požadované chování výstupu, rozhodl jsem se tedy, že pro všechny protokoly dat v rámci IP bude vypisována zdrojová a cílová IP adresa, u protokolu TCP a UDP obohacená o zdrojový a cílový port dle referenčního výstupu v zadání. U ARP protokolu jsou IP adresy ve výstupu nahrazeny MAC adresou zdroje a cíle podle ethernetové hlavičky.

Všechny tyto získané informace, které tvoří hlavičku výpisu paketu, se složí a vypíší na standardní výstup. Následuje výpis paketu po bytech na standardní výstup a ukončení programu po dokončení přijímání všech paketů.

Pro testování a ladění je možné odkomentovat/zadefinovat v hlavičkovém souboru main.hpp makro DEBUG před překladem, poté bude sniffer vypisovat více ladících informací o filtrech a protokolech nad rámec běžného zadání. Tato funkcionalita mi sloužila především pro ladění a rozhodl jsem se v běžném režimu příliš neupravovat požadovaný výstup programu.

Testování

Program jsem testoval na referenčním virtuálním stroji za pomoci nástrojů `Wireshark` a `nping`.

Testování spočívalo v první fázi ve spouštění mého packet snifferu s různými parametry a filtry, zasílání paketů na různých portech a protokolech pomocí nástroje `nping` z mého počítače do virtuálního stroje a vizuální kontrola výstupů (počet paketů, formát, zachycení jen paketů podle filtru). V druhé fázi potom ve spuštění zachytávání paketů v programu `Wireshark` a v mém packet snifferu ve stejnou chvíli, sledování výstupu obou programů a následném porovnání výsledků, zda přijatý a vytištěný obsah paketu je v obou programech shodný a zda odpovídají informace zjištěné a vypsané o paketu (adresy, porty, protokoly), obdobně pak i s aplikací filtrů ve `Wiresharku` a v mém snifferu.

Zdroje

Ethernet frame. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/Ethernet_frame

EtherType. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/EtherType

IPv4. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/IPv4

IPv6. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/IPv6

IPv6 packet. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/IPv6_packet

List of IP protocol numbers. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers

Transmission Control Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

User Datagram Protocol. In: *Wikipedia: the free encyclopedia* [online]. San Francisco (CA): Wikimedia Foundation, 2001- [cit. 2021-04-22]. Dostupné z: https://en.wikipedia.org/wiki/User_Datagram_Protocol

Man page of PCAP [online]. [cit. 2021-04-22]. Dostupné z: https://www.tcpdump.org/manpages/pcap.3pcap.html