.Sieťový analyzátor

Úvod do problematiky zadania:

Zadanie sa skladá z viacerých častí, ktoré sa prehlbujú v konkrétnosti. Prvou časťou zadania bolo zjednotenie základných informácií o zachytenej dátovej komunikácií. Následne sme sa venovali detailným výpis danej dátovej komunikácie a v poslednej časti sme už priamo kontrolovali správnosť vytvorenia spojenia medzi komunikátormi či iným dôkladným procesom pri komunikácií.

V rámci zachovania jednotnosti a jednoduchosti spracovania dátového toku, učinili sme rozhodnutie celú implementáciu nášho analyzátora realizovať v programovom jazyku Python pomocou IDE Visual Studio Code. Vzhľadom na náturu zadania usúdili sme, že najvhodnejší prístup implementácie kódu bude objektovo orientovaný. Vzhľadom na tento fakt, veľká časť našej implementácie využíva jednoduché koncepty OOP ako dedenie pri konkretizovaní technických detailov a špecifikácií, ktoré bolo nutné dopĺňať/obohacovať. Rovnako tak, referenčný prístup k objektom nám zaručil možnosť meniť premenné daných objektov aj v mohutnejších štruktúrach, kde boli dané objekty umiestnené.

Následne po zjednotení všetkých implementačných špecifikácií, sme vytvorili externý súbor vo forme YAML, ktorý nám bude slúžiť ako akási databáza (slovník) za účelom spájať celočíselné dekadické hodnoty špecifických bajtov, ktoré budeme kontrolovať. Dané číselno-hodnotové spojitosti sú roztriedené podľa zámeru do určitých kategórií, z ktorých počas spracovania údajov špecificky vyberáme potrebné dáta.

Diagram

Pre zovšeobecnenie problematiky celého zadania sme vytvorili grafickú reprezentáciu nášho riešenia, ktorá sa skladá z horizontálnej a vertikálnej časti. Vertikálna generalizuje základný postup získavania a triedenia daných údajov – začíname otvorením pcap súboru, ktorý otvárame pomocou knižnice scapy, tieto dáta hneď meníme na bajty a následne ich v cykle využívame ako parametre v metóde, ktorá začína jednoduchým určením dĺžky packetov, určenie cieľových a zdrojových MAC adries a určenie frametyp-u. V tomto bode začíname pri určovaní všetkých týchto generickým vlastností sa venujeme presne definovaným umiestneniam bajtom, ktoré dané údaje obsahujú. Obzvlášť následne venujeme pozornosť detailnejšiemu rozboru IPv4 packetov pri frametype Ethernet II, kde definujeme daný protokol a aj protokol aplikačnej vrstvy.

V rámci horizontálneho rozloženia sa venujeme implementácií 4. úlohy z nášho zadania. Kde na základe vybraného protokolu, vykonávame špecifické operácie, ktoré dopĺňajú všeobecné informácie o packetoch a kontrolujú správny priebeh komunikácie. Tieto procesy a metódy sú detailnejšie vysvetlené v ďalších kapitolách dokumentácie.

Technické špecifiká – metódy a triedy

V rámci logiky objektovo-orientovanej implementácie sme vytvorili viaceré triedy a metódy. Triedy používame na vytváranie objektov, ktoré reprezentujú „typy“ paketov, ktoré následne riešime. Vo všeobecnosti máme triedy pre „Generický packet“, „IEEE LLC & SNAP“, „IEEE LLC“, „IEEE Raw“, „Ethernet II“, „Handshakes“, „ICMP komunikácia“ a „TFTP komunikácia“.

Triedy

Generický packet ( generalPacket )

Táto trieda slúži na reprezentáciu najvšeobecnejšej formy zachytávanej komunikácie resp. účelom je udržiavať všetky údaje, ktoré budú potrebné pri všetkých detailnejších analýzach údajov – obsahuje ID daného packetu, dĺžky packetu, frame\_type, zdrojová a MAC adresa. Následne v rámci tejto triedy máme konštruktor, ktorý iniciuje novo-vytvorený objekt. Následne používame v našej implementácií daný objekt ako rodičovský objekt pri detailnejšej analýze packetu pre ether\_type a dodatočné informácie podľa frame\_type štandardu.

IEEE LLC & SNAP ( ieeeLlcSnap)

Ako bolo vyššie avizované táto trieda dedí metódy a konštruktor s volaním vlastnej metódy. V rámci parametrov sa pridáva oproti „General packetu“ iba parameter PID. Následne po inicializovaní všeobecných údajov prechádzame na určenie PID. Tento údaje určujeme pomocou vlastnej metódy triedy „resolve“, ktorá prechádza dva možné miesta (21-22B alebo 47-48B), kde by sa daný údaj mal nachádza, avšak väčšiu prioritu má prvé možné umiestnenie. Táto metóda „resolve“ prehľadáva tieto dve možné lokácie a porovnáva daný údaj s našim externým súborom, kde máme uložené číselné kombinácie známych PID. Pri úspešnej zhode údajov, načítame z externého súboru zhodné označenie PID a priradíme ho k zodpovedajúcemu parametru objektu.

IEEE LLC (ieeeLlc)

Táto trieda kopíruje postup vyššie spomínanej „IEEE LLC & SNAP“ triedy, avšak so smenou, že dodatočný parameter už nie je PID, ale ide o SAP. Taktiež v korelácií s touto zmenou a prehľadávané bajty sú zmenené ( lokácia údaju SAP je na 16B ). Taktiež obsahuje rovnako pomenovanú metódu s rovnakým princípom určovania parametru SAP.

IEEE Raw (ieeeRaw)

Trieda „IEEE RAW“ je najjednoduchšia, pretože v tomto prípade nepredpokladáme žiadne dodatočné pridávanie údajov k danému štandardu, avšak tento taktiež dedí od rodičovskej triedy „General packet“.

Ethernet II (ethernetTwo)

Daná trieda, ako aj predošlé, dedí od rodičovskej triedy „General packet“ základné údaje, avšak v tomto prípade pridávame už oveľa viac údajov. V tejto triede pridávame parametre zdrojovú a cieľovú logickú adresu, protokol sieťovej vrstvy, protokol transportnej vrstvy, protokol aplikačnej vrstvy a zdrojový aj cieľový protokol. V rámci tejto triedy máme aj metódu, ktorá sa stará o pridelenie logických adries ( bajty, ktoré obsahujú údaje IP adries sú na 27-30 B a 31-34 B). V rámci metódy sa nám už dané IP adresy aj formátujú do štandardného tvaru.

Handshakes (threeHandshakes)

Trieda „Handshakes“ sa používa v rámci implementácie na reprezentáciu jednej celistvej komunikácie pri protokoloch TCP na transportnej vrstve. Ide o súčasť kontrolovania správneho vytvorenia komunikácie pred posielaním dát. Táto trieda ďalej nie je nikde dedená a jej parametre sú nasledovné: attempt\_to\_start (definuje či komunikácia sa snaží nadviazať spojenie), started (definuje či dané spojenie bolo zahájené platne podľa parametrov pravidiel three-way handshake), finished (definuje či spojenie bolo korektne ukončené) , communication\_data (ide o pole, do ktorého ukladáme každú úspešne filtrovanú komunikáciou vo forme objektov tried spomínaných vyššie, niekedy aj s dodatkovými parametrami), zdrojovú a cieľová IP adresa a porty (tieto parametre využívame na priame porovnávanie objektu tejto triedy s packetom, ktorý testujeme filtrom).

Metódy – postupná analýza

V rámci tejto dokumentácie opisujeme metódy chronologicky v poradí vykonávania procesov pri analýze packetov. Niektoré časti metód či metódy celé sú pomerne dlhé a v dokumentácií pôsobia pomerne neprehľadne, z tohto dôvodu sa v dokumentácií budeme odkazovať na súbor, ktorý je v rámci zadania odovzdaný – meno súboru je main.py. Cieľom tejto kapitoly je objasniť štruktúru nášho kódu a vysvetliť postup analýzy packetov.

Process Params (process\_params)

Táto metóda sa spúšte hneď po tvorení daného pcap súboru a následnom premenení údajov na bajty. Jednotlivé packety prechádzame pomocou for cyklu. Táto metóda má 3 argumenty packet, id, port – packet sú bajtové údaje packetu, id je iterovaný identifikátor vo for cykle, ktorý opisuje špecifický packet a port je označenie pre protocol, ktorý zadávme ako flag/switch pri spustení kódu. Používame názvoslovie „port“ z dôvodu, že pojem protokol sa vyskytuje v našom kóde často a takýmto spôsobom ho jasnejšie odlišujeme. Táto metóda zisťuje základne údaje o packete cieľovú (1-6B) a zdrojovú (7-12B) ip adresu, dĺžku packetu, následne volá metódu „frame\_type\_function“, ktorá určí hodnotu frame\_typu na (13-14B) a na základe tejto hodnoty určí frametype. Následne sa táto hodnota vracia do premennej a všetky tieto údaje sa následne spracujú už v podobe objektu „General packet“ triedy. Ďalej spracujem hexadecimálny kód celého packetu a ten si zapíšeme do globálneho slovníka na základe ID prechádzaného packetu. Táto metóda končí zavolaním metódy final\_resolve.

Ethertype solver (final\_resolve)

Táto metóda na základe vyššie určeného frametypu vytvára špecifický typ objektu na základe príslušnej triedy - Ethernet II , 802.3 LLC / LLC & SNAP / RAW. Následne po vytvorení tohto objektu považujeme túto vrstvu za ukončenú a venujeme sa vyššej vrstve, kde pozeráme, aký špecifický ethertype používa. Taktiež sa určuje táto hodnota na základe externého súboru, pod podmienkou, že frametype bol definovaný ako Ethernet II. Pozeráme na už spomínaný 13-14 B, ktorý nám následne určí o ethertype pri zhode s údajom z externého súboru. Ďalej na základe vybraného ethertypu sa volajú špecifické funkcie – pri IPv4 pozeráme na ďalšiu vrstvu a zistime aký protokol sa na vyššej vrstve používa, ak ide o ARP kotrolujem ARP requesty a ARP komunikáciu pomocou metódy „arp\_check\_comm“ a v prípade, že sme to zadali do switchu -p pri spúšťaní súboru a v prípade, že nejde ani o ARP ani o IPv4 zavoláme metódu „yaml\_creator“, ktorý zodpovedá za vytvorenie generické yaml, kde vypíše komunikáciu na základe filtra.

Protokol Transportná Vrstva (solver\_inner\_protocol)

Táto metóda sa venuje definovaniu protokolu na štvrtej vrstve, z dôvodu dĺžky danej metódy nie je v dokumente uvedená, avšak nachádza sa na riadku 232 v súbore main.py. Na začiatku danej metódy si uložíme dĺžku hlavičky IPv4 packetu a následne zadefinujem protokol na štvrtej vrstve obdobne, ako pri iných, kde určujem na špecifickom bajte hodnotu daného protokolu (24B) a následne ju porovnávam s externým súborom. Pri zhode pozerám či sa nejedná o UDP alebo TCP protokol v takomto prípade metóda určí aj protokol, ktorý sa používa na aplikačnej vrstve. Prehľadá bajty, ktoré opisujú cieľový a zdrojový port služby, následne oba prechádzajú podobným procesom pomocou externého súboru, kde sa mapuje nájdená hodnota so službou v externom súbore. V prípade, že sme nechceli komunikáciu filtrovať, tak v tomto bode analýza končí a súbor sa vráti do metódy, ktorá vytvorí yaml súbor.

Avšak v prípade, kde využívame filter sa v prípade TCP komunikácie volá metóda na kontrolu TCP komunikácie „check\_communication“, v prípade že ide o UDP komunikáciu protokolu TFTP, tak sa volá metóda na kontrolu UDP komunikácie „tftp\_check“, ďalší prípad nastáva, keď ide o ICMP protokol, v takomto prípade sa volá metóda na kontrolu ICMP packetov „icmp\_check\_come“.