Síťové aplikace a správa sítí

Přenos souboru skrz skrytý kanál

Vojtěch Šíma xsimav01

listopad 2021

Obsah

1 Spuštění

Samotné spuštění běhu programu probíhá z příkazové řádky dvěma způsoby. Pokud se jedná o serverovou část, která bude naslouchat a přijímat odeslané pakety, ta je spuštěna příkazem **sudo ./secret -l**, kde přepínač -l značí že se jedná právě o server. Klient, který se stará o odeslání souboru se spouští **sudo ./secret -s [adresa] -r [soubor]**.

2 Načtení argumentů

První věc, která je spuštěna po načtení programu je funkce *getopt()*, pomocí které dojde k načtění přepínačů a jejich hodnot. Jsou nastaveny boolean hodnoty pro **rflag, sflag a lflag** pro následnou kontrolu.

Poté dochází k rozdělení programu na základě toho, zda je spuštěn jako server nebo jako klient. Umístění obou částí v jednom souboru bylo zvoleno z důvodu lepšího přesouvání v kódu.

3 Klient

V klientské části po kontrole přepínačů dochází k volání vestavěných funkcí *getaddrinfo* a *inet_ntop()*. První uvedená nám do struktury **res** uloží hodnoty se kterými nadále budeme pracovat. Druhá převede IP adressu do textové podoby která je potřeba pro poslání paketu. Zde je použita převzená funkce *get_in_addr*[1], která vrací infromaci o tom, zda se jedná o IPv4 nebo IPv6 adresu.

3.1 Velikost dat

Následně je zvolena hodnota proměnné **maxvelpacketu**, která je nastavena na hodnotu 1424. Tato hodnota byla odvozena od maximální velikosti paketu, která může být 1500 bytů. Do této velikosti se musí vejít i IP a ICMP hlavičky a také Linux cooked capture. S ohledem na zmíněné proto bylo zvolena bezpečná velikost dat 1424 bytů, která je bezezbytku dělitelná hodnotou 16, která je zmíněna dále v dokumentaci. Na základě hodnoty **ai_family** ze struktury **addrinfo** je určena hodnota protokolu, který je následně použit ve funkci *socket*, jejímž voláním je vytvořen socket pro komunikaci.

3.2 Načtení souboru

Pomocí ifstream je načten celý soubor jako binární do proměnné **buffer**. Také je zjištěna délka celého souboru v bytech. Pokud soubor neexistuje nebo jeho otvírání selže, je vyvolána chyba na standartní chybový výstup.

3.3 Úvodní paket jako protokol pro přenos dat

Pro předání informací potřebných pro chod programu mezi klientem a serverem bylo zvoleno řešení, kdy je poslán speciální úvodní paket, ve kterém je uložena za klíčovým slovem **F1LEN4ME** informace o názvu souboru, celkovém počtu paketů které budou ze strany klienta poslány a o velikosti posledního paketu, která pro server bude jako jediná neznámá.

Tyto hodnoty jsou uloženy do textového řetězce a následně přetypovány a zašiforvány pro odeslání. Počet paketů je zjištěn a vypočítán na základě velikosti celého načítaného souboru a velikost posledního paketu je vypočítána jako zbytek po celočíselném dělení velikosti souboru a maximální velikosti paketu.

```
VelikostPoslednihoPaketu=VelikostSouboru % 1024; //Vypocet velikost posledniho paketu
```

3.4 Rozdělení načteného bufferu

Důležitým prvkem klienta je rozdělení načteného bufferu ze souboru na menší části, které pak mohou být odeslány v jednom paketu. K tomuto dochází v cyklu while, kde je tento původní buffer rozdělen postupně na menší části po 1424 bytech a jeden menší blok v případě posledního paketu.

3.5 Kryptovací funkce

Funkce, která se stará o zašifrování dat se jmenuje *CryptFunction* a její vstupní parametrem je část dat ze souboru a informace a jejich délce. Šifrovací klíč je nastaven funkcí *AES_set_encrypt_key* jako 128 bitů dlouhý a je založen na loginu. Jeho hodnota je uložena do proměnné **encryptkey**.

Následně je naalokována paměť pro výsledná zašifrovaná data, které budou postupně ukládána právě do proměnné **output** která je i výstupní hodnotou této funkce.

Před samotným šiforváním dochází k dalšímu rozdělení dat, nyní do bloků o maximální velikosti 16 bytů. Hodnoty jsou překopírovány do pomocné proměnné **input16**. Každý tento jednotlivý blok o velikosti 16 bytů je zašiforván pomocí funkce *AES_encrypt*. Postupně jsou takto zašiforvány všechny bloky které lze naplnit daty, až nakonec dojde i na "přebytečná"data, pro které je vytvořen menší blok dat a následně také zašifrován. Výstupem funkce jsou zašiforavná data, která jsou předána zpět klientovy pro finální poslání paketu.

3.6 Poslání paketu

Po zašifrování jsou data v proměnné **CryptedData** nakopírována do dříve vytořeného pole **packet** zvětšeného o velikost ICMP hlavičky. Následuje již pouze poslední krok, který je v programu proveden klientem a to poslání pomocí vytvořeného socketu na adresu, která byla zadaná jako vstupní parametr při spouštení. Pokud odeslání selže, dochází k ukončení běhu programu.

4 Server

Větev která obstarává chod serveru je vykonána na základě **lflag**, který je nastaven v případě zadaného příslušeného přepínače **-l**.

4.1 Výběr rozhraní a naslouchání

Pomocí funkce *pcap_open_live* je otevřeno rozhraní ANY, které bylo zvoleno z důvodu, aby mohl být program testován jako server tak, že jsou na něj data odesílána z klienta na loopback. Následně je nastaven **icmp or icmp6** filtr, díky kterému budou zachytávany pouze tyto pakety. Následně již rozhraní naslouchá a v případě, že přijde paket, dojde k jeho obsloužení ve funkci *mypcap_handler*.

4.2 Načtení dat

Na základě struktury **ip** je zjištěna velikost poslaných dat. Je zde potřeba počítat s 16 byty Linux cooked capture, která je na rozhraní ANY v linuxovém systému místo Ethernet vrstvy, 8 byty naší ICMP hlavičky a 20 byty IPv4 hlavičky. Celkem je tedy nutné přeskočit prvních 44 bytů, za kterými se budou nacházet data, která bude server dešifrovat. Při serverové části bylo čerpáno ze souboru **sniff-filter** který byl k dispozici jako příklad zveřejněný v datovém skladu k předmětu Síťové aplikace a správa sítí. [2]

```
unsigned char *data=(u_char*)(packet+16+20+8);  //Nacteni sifrovanych dat
int sizedata = ntohs(my_ip->ip_len) - 16 - 8 - 20; //Velikost nacitanych dat
```

4.3 Dešifrování dat

Dešiforávní dat je totožné, jako jejich šifrování. Zašiforvaná data jsou pomocí funkce *DecryptFunction* opět rozdělena na měnší bloky a ty jsou pomocí *AES_decrypt* postupně deširována a uložena do výstupní proměnné **output**, která tyto data přesune zpět k jejich uložení do souboru.

4.4 Kontrola prvního packetu

Je potřeba kontrolovat a správně odchytit první informační paket. K tomu dochází kontrolou prvních dešifrovaných znaků, kde se v úvodním paketu nachází naše klíčové slovo **F1LEN4ME**. Pokud na toto slovo narazíme, pomocí funkce *strtok* jsou postupně získávány z tohoho paketu zpět ty hodnoty, které do něj byly v klientské části uloženy.

4.5 Výstupní soubor

Výstupní soubor je na základě jména vytvořen a pak vyprázdněn pro případ, že by byl již vytvořen dříve. Hodnoty jsou do něj pak následně ukládány pomocí *write* ukládána tolikrát, kolikrát jsou přijaty pakety pro daný soubor.

```
myfilewrite.write((char *)data,sizedata);//Zapis desifrovanych dat do souboru
```

Velikost zapsaných dat je v případě posledního paketu závislá na základě velikosti posledního paketu, který byla přenesena v úvodním paketu. V ostatních případěch je velikost ukládaných dat 1424 bytů. Tyto správné velikosti jsou důležité proto, aby nedošlo při ukládání do souboru k uložení přebytečných dat, které se zde mohou během širování a dešifrování vyskytnout.

5 Nedostatky řešení

Není implemetováno v serverové části přijímání paketů zaslaných pomocí IPv6 adres a funguje tedy pouze přijímání IPv4 adres. Dalším nedostatkem je nepravidelný problém při úkládáních velikých souborů, především videí a větších obrázku. Při ukládání dojde k jejich částečné deformaci z důvodů, které bohužel nebyly objeveny.

6 Závěr

Řešení není kompletní, ale ty části které byly implementovány fungují tak jak mají a dle zadání jsou přenesny v zašiforvané podobě a následně je kopie souboru úspešně uložena v místě, odkud byl spouštěn server.

Reference

[1] Lorenz, T.: beejs-guide-to-network-samples. Listopad 2013. Dostupné z: https://github.com/thlorenz/beejs-guide-to-network-samples/blob/master/lib/get_in_addr.c

[2] Matousek, P.: sniff-filter.c. 2020. Dostupné z: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FISA-IT%2Fexamples&cid=14699