Tunelování datových přenosů přes DNS dotazy

Projektová dokumentácia

ISA – Sieťové aplikácie a správa sietí

Matej Koreň, xkoren10

13.11.2022

Obsah

Zadanie

Úlohou bolo vytvoriť klientskú a serverovú časť aplikácie, ktorá využíva tunelovanie dát prostredníctvom DNS otázok (použiteľné napríklad pri DNS data exfiltration útoku.)

Princíp tunelovania

V sieti, v ktorej sa nachádzame, beží DNS resolver, ktorý prekladá doménové mená z internetu. Za normálnych okolností by nám tento resolver nijako nepomohol, pretože by sme si síce mohli preložiť ľubovoľnú doménu na IP adresu, ale sieť nás s ňou odmietne spojiť. Pri DNS tunelovaní miestnemu DNS resolveru budeme posielať otázky, do ktorých zabalíme našu komunikáciu. Resolver ich predá nášmu autoritatívnemu DNS serveru, v domnienke, že cieľom je preložiť doménové meno na adresu. Náš špecializovaný autoritatívny server ale v skutočnosti nebude odpovedať bežným spôsobom, ale do odpovedi zabalí dáta, ktoré nám chce poslať. Keďže ide o štandardizovanú DNS komunikáciu, resolver nám informácie ochotne predá.

(Preložené a upravené z https://www.root.cz/clanky/tunelujeme-provoz-pomoci-dns-cesta-ven-ze-site/)

Implementácia

Klientska aplikácia – dns_sender.c

Klientska aplikácia odosiela dáta zo súboru / štandardného vstupu. Spustenie programu je podľa nasledujúceho predpisu:

dns_sender [-u UPSTREAM_DNS_IP] {BASE_HOST} {DST_FILEPATH} [SRC_FILEPATH]

pričom

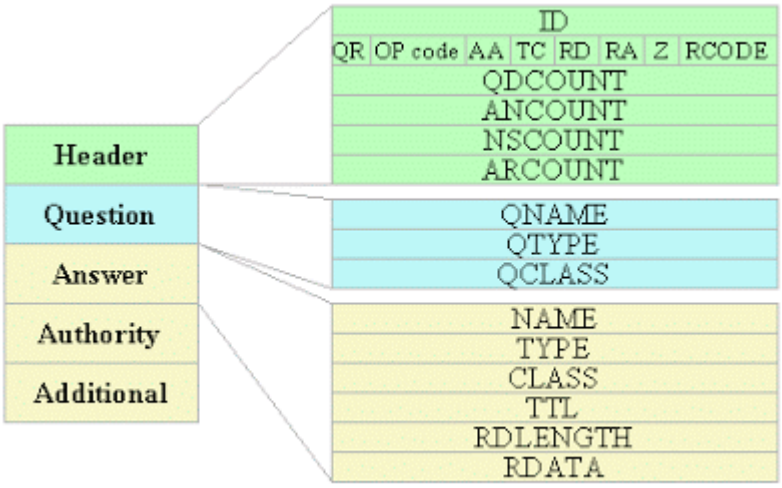
- Prepínač –u slúži na vynútenie vzdialeného DNS serveru (ak nie je zadaný, použije sa prvý implicitný DNS server zo súboru etc/resolv.conf
- BASE_HOST je parameter určujúci bázovú doménu prenosov
- DST_FILEPATH je cesta, pod ktorou sa dáta uložia na serveri
- SRC_FILEPATH je cesta k súboru na odoslanie (ak nie je zdaná, číta sa z STDIN)

Po spustení prebehne kontrola a načítavanie argumentov, ktoré sa uložia do premenných. Vytvorí sa socket na UDP komunikáciu [1]. Otvorí sa súbor na odoslanie a v cykle sa z neho číta určitý počet znakov, kým sa nenarazí na koniec súboru. Tento blok znakov sa následne pošle do funkcie na jeho zakódovanie do znakov povolených v DNS packete (*base_32_encode*). Už zakódované dáta posunú funkcii *send_chunk*, v ktorej sa vytvoria štruktúry *dns_header_t* a *dns_question_t* . Tie obsahujú hodnoty presne špecifikujúce druh paketu. Pred naplnením bufferu na odoslanie sa zakódované dáta musia rozdeliť na tzv. *labels*, ktoré môže mať maximálne 64 znakov. Dáta sa segmentujú a adekvátne sa označí ich dĺžka. Taktiež sa nami zvolená bázová doména prevedie do DNS formátu podobným

spôsobom, kde sa z nej odstránia bodky a nahradia sa dĺžkou znakov po nej nasledujúcich. Následne sa paket poskladá do tvaru *Hlavička-Zakódované dáta-Bázová doména-Otázka* [2]. Paket sa odosiela na server pomocou funkcie *send_to* a čaká sa na odpoveď. Podľa jej druhu sa pokračuje v odosielaní ďalšieho bloku zakódovaných dát alebo sa komunikácia ukončí [1].

Štruktúra paketov

V klasickej DNS komunikácii majú pakety nasledujúci tvar:



Zdroj: https://amriunix.com/post/deep-dive-into-dns-messages/

V tomto projekte sú však využité len prvé dve časti – hlavička a otázka. Hlavička je naimplementovaná ako štruktúra bitových komponentov, ktoré sa plnia hodnotami pred odosielaním [3]. Menšia zmena je s časti *Question* – tá je v programe tvorená ako štruktúra obsahujúca len typ a triedu. *QNAME* je vytvorený zvlášť ako pointer v bufferi na odoslanie, ukazujúci na pozíciu za hlavičku. Zaň sa pridávajú zakódované dáta, bázová doména a nakoniec štruktúra *Question*.

Serverová aplikácia – dns_receiver.c

Server počúva na implicitnom porte pre DNS komunikáciu (port 53) a prichádzajúce dátové prenosy ukladá na disk vo forme súboru. Spustenie programu je podľa nasledujúceho predpisu:

dns_receiver {BASE_HOST} {DST_DIRPATH}

pričom

- BASE_HOST je parameter určujúci bázovú doménu prenosov
- DST_DIRPATH je cesta, pod ktorou sa dáta ukladajú na serveri

Po spustení prebehne kontrola a načítavanie argumentov, ktoré sa uložia do premenných. Vytvorí sa adresár špecifikovaný v *DST_DIRPATH* a socket na UDP komunikáciu [1]. Tento socket sa napojí na špecifikovaný port prostredníctvom funkcie *bind*. Podobne ako pri klientovi, *BASE_HOST* sa prevedie do DNS formátu, aby sme ho následne vedeli porovnávať s prichádzajúcimi dátami. Program sa dostáva do cyklu, v ktorom sa čaká na dáta. Ak funkcia *recv_from* zachytí DNS packet, skontroluje sa, či sa zhodujú bázové domény. Ak nie, pokračuje sa v čakaní na ďalší paket. Ak áno, jeho obsah (správa) sa z neho vyextrahuje podľa dĺžky jednotlivých segmentov (labelov). Dáta sa posúvajú funkcii *save_data*, kde sa dekódujú (*base_32_decode*) a spracujú podľa ich dĺžky a id hlavičky [1]. Na základe výstupu z ukladania sa klientovi odosiela adekvátna odpoveď [1].

Komunikačný protokol

Ako komunikačný protokol bol zvolený UDP. Je jednoduchší na implementáciu, avšak je nespoľahlivý. Túto skutočnosť implementácia rieši nasledovne:

Klient

Druh odosielaných dát závisí od identifikátoru v hlavičke a ich dĺžky (bez bázovej domény) :

- Id 0 - ide o názov súboru (DST_FILEPATH), do ktorého sa budú dáta ukladať
- Id > 0 a dĺžka dát nenulová - ide o dáta na zápis
- Id > 0 a dĺžka dát =0 - ide o posledný paket, ktorý značí koniec prenosu

Po odoslaní sa čaká na odozvu od serveru. Pri pozitívnej odozve sa pokračuje v posielaní dát, pri negatívnej nastala chyba na strane serveru a program sa ukončuje. V prípade, že odozva nepríde do 10 sekúnd, program končí s chybovou hláškou.

Server

Server po kontrole bázovej domény ukladá dáta podľa rovnakých pravidiel:

- Id 0 - ide o názov súboru -> Vytvorí sa súbor a čaká na zápis
- Id > 0 a dĺžka dát nenulová – Zápis do súboru
- Id > 0 a dĺžka dát =0 – súbor sa zatvára, čaká sa na ďalšiu komunikáciu

Pri úspešnom výstupe z funkcie *save_data* sa ako odpoveď klientovi odošlú rovnaké dáta, avšak s pozmenenými príznakmi:

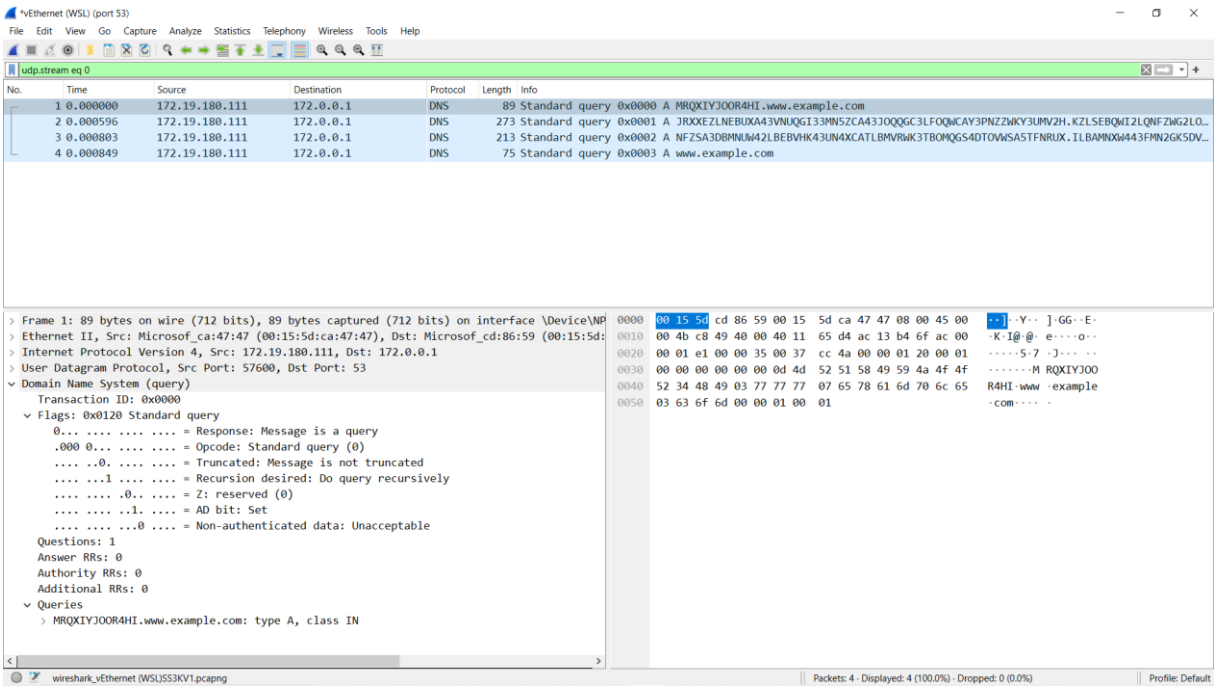
- Pozitívna odpoveď – V hlavičke sa nastaví druh dotazu response (1)
- Negatívna odpoveď - V hlavičke sa nastaví druh dotazu na response (1) a *response code* (2)

Obmedzenia a možné rozšírenia

Hlavným obmedzovacím faktorom je použitie UDP – pre bezpečnejšiu komunikáciu cez vopred vytvorený kanál by bolo vhodnejšie použiť TCP. Ďalším obmedzením je veľkosť správ a DNS paketov – pri odosielaní väčšieho súboru je práca s ním zložitejšia a vyžaduje viac odoslaných paketov. To zvyšuje šancu straty paketov alebo ich poškodenia. Pri lokálnom testovaní je ťažké overiť celkovú spoľahlivosť zvoleného protokolu, keďže poskytuje iba základné prvky sieťovej komunikácie. Medzi možné rozšírenia by určite patrila podpora IPv6, ošetrenie ďalších možných problémov (zmena poradia prichádzajúcich paketov, podpora rôznych typov odosielaných súborov a iné.

Testovanie a meranie

Softvér bol vyvinutý na WSL, čo so sebou prináša určité nevýhody. Ako prvý bol testovaný dns_sender.c, ktorý nečakal na odpoveď a rovno posielal DNS queries na špecifikovaný server:



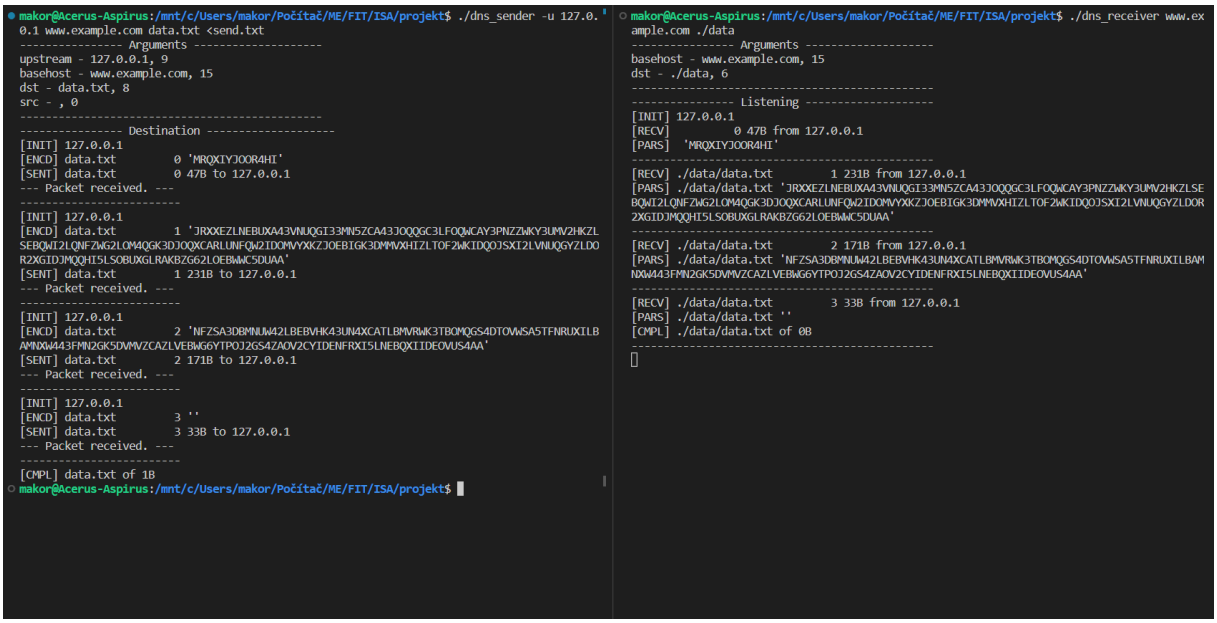
Obr.2 – otázky od klienta pre 127.0.0.1 (loopback)

Na tomto zázname môžeme vidieť, že poradie paketov súhlasí s použitým protokolom (prvý obashuje meno súboru, druhý a tretí kódované dáta a posledný len bázovú doménu)

Pre otestovanie komunikácie so serverom je nutné (kvôli WSL) zmeniť implicitný port na odosielanie. Komunikácia tak nebude špecifikovaná ako DNS, ale ako čisté UDP. Následne je však možné pozorovať ich vzájomnú komunikáciu:

	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	UDP	89	48342 → 1234 Len=47
2	0.000357816	127.0.0.1	127.0.0.1	UDP	89	1234 → 48342 Len=47
3	0.000595628	127.0.0.1	127.0.0.1	UDP	273	48342 → 1234 Len=231
4	0.000789054	127.0.0.1	127.0.0.1	UDP	273	1234 → 48342 Len=231
5	0.001006840	127.0.0.1	127.0.0.1	UDP	213	48342 → 1234 Len=171
6	0.001172841	127.0.0.1	127.0.0.1	UDP	213	1234 → 48342 Len=171
7	0.001306213	127.0.0.1	127.0.0.1	UDP	75	48342 → 1234 Len=33
8	0.001655361	127.0.0.1	127.0.0.1	UDP	75	1234 → 48342 Len=33

V terináloch:



Odkazy

[3] - https://www.ietf.org/rfc/rfc1035.txt

(base_32_encode) (base_32_decode) - https://github.com/google/google-authenticator-libpam/blob/master/src/base32.c

Ďalšie zdroje:

https://mislove.org/teaching/cs4700/spring11/handouts/project1-primer.pdf

https://moodle.vut.cz/pluginfile.php/502893/mod_folder/content/0/udp/echo-udp-client2.c?forcedownload=1

https://moodle.vut.cz/pluginfile.php/502893/mod_folder/content/0/udp/echo-udp-server2.c?forcedownload=1\