DHCPv6 Relay

Autor: Jan Vaculík (xvacul30)

Datum odevzdání: 18/11/19

Úvod

Program d6r slouží jako DHCPv6 relay. Je to vrstva mezi klientem a serverem, kteří mezi sebou jinak nemohou komunikovat.

Odchytává multicastové DHCPv6 zprávy od klienta a posílá je pomocí Relay-Forward zprávy serveru, poté vyčká na odpověď ve formě Relay-Reply zprávy a tu zprostředkuje klientovi.

V Relay-Forward zprávě dále posílá serveru, dle specifikace zadání, MAC adresu klienta.

Překlad a spuštění

Program se překládá příkazem `make` nebo `make all`. Pro překlad je použit kompilátor GCC s přepínačem `-lpcap`.

Program je rozdělen do dvou zdrojových souborů a dvou hlavičkových souborů. Soubor **d6r** obsahuje logiku programu a soubor **functions** pomocné funkce. V hlavičkových souborech jsou potom definice funkcí a definice struktur.

Program je nutné spouštět s právy super uživatele a může se spouštět například takto:

```
$ sudo ./d6r -s 2001:1200:1100:1000::10 -d -l
```

Přepínače

Při spuštění programu je možná předat v argumentu několik přepínačů:

-s
povinný přepínač, sloužící k určení serveru

-d
debugový přepínač, sloužící k výpisu informací na standardní výstup

-l
pomocí knihovny syslog zapisuje výše zmíněný debug výstup

-i
slouží k volbě síťového rozhraní, na kterém bude relay naslouchat, může být takto zvoleno pouze jedno síťové rozhraní, pokud takto není zvoleno žádné rozhraní, tak naslouchá relay na všech

-h
vypíše nápovědu ke spuštění na standardní výstup a ukončí program

Implementace

Funkcionalita programu byla testována na systému Ubuntu 18.04, přeložitelnost na serveru Merlin (CentOS 7). Kvůli bitovým operacím je správná funkcionalita zaručena pouze na procesorech s little-endian architekturou.

Jak již bylo zmíněno výše, tak se kód nachází ve dvou souborech. V souboru functions.c se nacházejí 3 funkce, které se používají v rámci programu:

int createSocket(*const char *iface_name*, *int port*)

Tato funkce slouží k vytvoření a konfiguraci síťového soketu. Jako argumenty bere jméno rozhraní na které se bude soket vázat (pokud je nastaveno na 0, soket se neváže na žádné rozhraní) a port, který se bude vázat na soket.

Nastavuje timeout interval při čekání na zprávu a nastavuje z jakého portu budou odcházet zprávy. Využívají se konkrétně porty 546 a 547, neboť na právě těchto portech probíhá DHCP komunikace. Funkce je implementována pomocí volání socket(), setsockopt() a bind() z knihovny sys/socket.

struct dhcpv6_relay_message_options* setOptions(*int option_len*, *int opt_num*)

Vytvoří strukturu obsahující hlavičku pro DHCPv6 options.V argumentech se zadává číslo a délka této option.

int lookForOptionType(*int option_no*, *const unsigned char* reply[1024], *int msg_lngth*, *int counter*, *int *option_length*)

Protože se jednotlivé typy options mohou v datagramu nacházet v nepředvídatelném (libovolném) pořadí, program je musí nejdříve najít, k tomu slouží tato funkce.

Poté co se jí předají v parametrech potřebné informace (číslo options které hledá, kde je hledá (datagram), délku prohledávaného úseku a kde má začít), tak ve while smyčce prohledává daný datagram.

Načte délku právě zpracovávaných option (aby funkce věděla, kam se přesunout, pokud na daném místě hledané options nenajde), tuto informaci ukládá do proměnné,

pomocí které ji programu vrací referencí. Prohledává do konce místa určení. Pokud zadané options najde, vrací číslo korespondující s indexem ve zprávě (proměnná counter), pokud jej nenajde vrací funkce -1.

Přepínače programu se kontrolují jako první věc v souboru d6r.c, hledají se platné argumenty ve smyčce for.
U přepínače **-s** se kontroluje jeho existence a zda za ním následuje platná IPv6 adresa. U přepínače **-i** se kontroluje zda za ním následuje platný název rozhraní. U zbylých přepínačů se zkoumá pouze jejich existence.

Další věcí je nastavení snifferu, k jeho realizaci je použita knihovna pcap/pcap. Příchozí traffic se odchytává pouze na portu 546.

Hlavní část programu se nachází ve smyčce while, která pomocí funkce pcap_next() zpracovává příchozí datagramy. Parsuje se Ethernetová hlavička, ze které čteme MAC adresu, dále IPv6 hlavička, ze které se dozvíme délku zprávy a adresu klienta. Kontroluje se, že se jedná o UDP datagram. Jednotlivé hlavičky se ukládají v případě další práce s nimi do knihovnami předem definovaných struktur.

Poté co program takto zpracuje získanou zprávu se dostáváme do samotného těla UDP zprávy, zde se na prvních bajtech nachází číselná hodnota, která říká jaký typ DHCPv6 zprávy klient zaslal. V případě podporovaných zpráv se vytvoří Relay-Forward hlavička a Relay-Message hlavička ke které se připojí klientova zpráva.

Zpráva se ukládá do unsigned char array pomocí bajtové kopie (funkce memcpy()) - poté se ke zprávě připojí Client Link-Layer Address option obsahující MAC adresu klienta. Zpráva se dále zasílá serveru na předem určenou adresu a čeká se na odpověď.

Příchozí zpráva se (bez Relay-Forward hlavičky) zasílá klientovi. Poté se pomocí funkce lookForOptionType() hledají očekávané options - těmi jsou žádost o dočasnou adresu, žádost o permanentní adresu a zjišťování prefixu. Tyto informace jsou v případě této implementace programu d6r zjišťovány až ze strany serveru a nikoliv rovnou od klienta. Dále jsou tyto informace zjišťovány pouze v případě že chceme buď debug výpis, nebo chceme tyto informace logovat pomocí syslogu. V opačném případě je zpráva pouze přeposlána.

Po skončení tohoto procesu se program vrací na začátek smyčky while a zpracovává případnou další klientskou zprávu.

Návratové kódy

- **11** - chybí přepínač **-s**, nebo je adresa serveru zadána v nesprávném formátu
- **12** - přepínač **-i** zadán více než jednou
- **13** - rozhraní zadané v přepínači **-i** neexistuje, nebo nemá IPv6 adresu
- **14** - s tímto návratovým kódem končí program, pokud je použit přepínač **-h**
- **21** - zařízení nemá žádná síťová rozhraní
- **-1** - vnitřní chyba programu

Testování

Program byl testován v prostředí tří virtuálních strojů (všechny běželi na virtuálním obraze Ubuntu 18.04). Byly vytvořeny dvě lokální sítě - klientská a serverová. Relay potom nahlížel skrze dvě rozhraní do obou sítí a umožňoval komunikaci. Jakmile v tomto testovacím prostředí byla zajištěna funkcionalita, tak se serverový stroj odpojil a za pomoci FIT VPN relay přeposílal zprávy na poskytnutý server s IPv6 adresou 2001:67c:1220:80c::93e5:dd2.

TESKA, Ales. IPv6 parsing in C. Stack overflow [online]. [cit. 2019-11-16]. Dostupné z: https://stackoverflow.com/questions/2962664/ipv6-parsing-in-c

RFC 6939. HALWASIA, G., W. BHANDARI a S. DEC. Client Link-Layer Address Option in DHCPv6. Internet Engineering Task Force (IETF) [online]. Cisco Systems, 2013 [cit. 2019-11-16]. Dostupné z: https://tools.ietf.org/html/rfc6939

RFC 8415. MRUGALSKI, T., M. SIODELSKI, B. VOLZ, A. YOURTCHENKO, M. RICHARDSON, S. JIANG, T. LEMON a T. WINTERS. Dynamic Host Configuration Protocol for IPv6 (DHCPv6). Internet Engineering Task Force (IETF) [online]. 2018 [cit. 2019-11-16]. Dostupné z: https://tools.ietf.org/html/rfc8415

Funkcionalita snifferu byla inspirována dostupným demonstračním souborem od doktora Matouška - sniff.c.