Projekt do předmětu ISA

SSH/FTP Honeypot

Václav Bayer
xbayer05, 3BIT
22.11.2015

Obsah

1 Úvod 2
2 Implementace 2
2.1 Vstupní parametry 2
2.2 Vytvoření serveru 2
2.3 Komunikace s klientem 3
2.4 Semafor a sdílená proměnná 3
2.5 Ukončení spojení 3
3 Závěr 4
3.1 Test.sh 4
3.2 Statistiky 4

1 Úvod

Cílem projektu bylo vytvořit aplikaci, která bude formou SSH a FTP serveru sbírat přihlašovací údaje o klientech, kteří se snaží na daný server přihlásit. Při pokusu o přihlášení dojde k získání potřebných údajů a následuje výpis do logovacího souboru (mód serveru, datum s časem, jméno a heslo). Aplikace musí podporovat protokoly IPv4 a IPv6. Server musí být konkurentní.

Mód SSH požaduje autentizaci klienta dle normy RFC 4253, mód FTP pak dle normy RFC 959.

2 Implementace

2.1 Vstupní parametry

Ze všeho nejdřív je nutné získat parametry, se kterými je program spuštěn. K tomuto účelu slouží funkce *getopt(),* která jej roztřídí dle zadání na následující parametry:

- Mód server (ftp/ssh)
- Adresu, na které server naslouchá
- Port, na kterém server naslouchá
- Cestu k logovacímu souboru, do kterého budou zapisovány záznamy
- Cestu k soukromému RSA klíči (mód ssh)
- Nastavení maximálního počtu připojených klientů, kteří mohou být obsluhováni (defaultně 10 současně připojených klientů)
- Nastavení maximálního počtu pokusů o přihlášení (mód ssh) (defaultně 3 pokusy)

První z těchto čtyř položek jsou povinné, zbylé jsou individuální vzhledem k módu serveru. Po roztřízení parametrů je volána funkce *check_params(),* která má zajistit ošetření argumentů pro jednotlivé parametry. Probíhá zde například zjištění, zda je zadaná IP adresa ve správném formátu a rozhodnutí jedná-li se o IPv4 nebo IPv6 adresu. Dále je kontrolován rozsah portu v rozmezí 1-65535, rozhodnutí o jaký mód serveru se jedná atd. Nastane-li případ, že je některý z parametrů či jejich argumentů nevalidní, program je ukončen s vytisknutím chybového hlášení a nápovědy funkcí *print_msg(),* která má toto na starost po celou dobu běhu programu.

Do této doby vše probíhalo bez rozlišení módu, po jeho zjištění se program z hlavní funkce začne větvit dle jednotlivých módů.

2.2 Vytvoření serveru

Pro mód SSH je volána funkce *ssh_serv(),* pro FTP mód je to funkce *ftp_serv(),* která si volá pomocnou funkci pro nastavení parametrů *ftp_prepare_conn().* Pro dodržení konkurentnosti serveru se pro jednotlivé klienty generují procesy funkcí *fork().* FTP server je implementován pomocí BSD socketů. K implementaci SSH varianty je použito API knihovny libssh, u této varianty je také bezpečnostní odlišnost oproti FTP a to ve formě práce se soukromým klíčem, nastavení módu „ssh-rsa“ a cesty k tomuto klíči.

2.3 Komunikace s klientem

Po nastavení parametrů, alokaci potřebných dat funkcí *alloc_mem()* a vytvoření serveru, server naslouchá jednotlivým klientům a po navázání spojení dochází ke komunikaci a výměně zpráv tak jak to specifikují uvedené normy. Tuto činnost zajištuje funkce ssh_client_connection() u SSH serveru, v případě FTP je to funkce *ftp_communication().*

ssh_client_connection() – Vytvoří se záznam klienta a dojde k pokusu o výměnu klíčů s klientem. Pokud dojde k neúspěchu je klient odpojen a ukončeno spojení. V kladném případě je vygenerován čas připojení funkcí *get_time(),* získána IP adresa klienta a spuštěna nekonečná smyčka, ve které jsou zachytávány zprávy klienta, na které jsou generovány odpovědi. Jednu z prvních zpráv, které server zachytí, použije k získání jména (loginu) klienta. Další typem zpráv je pokus o přihlášení zadáním hesla na straně klienta (proběhne inkrementace počítadla pokusů o přihlášení a kontrola zda počítadlo nepřesáhlo hranici danou parametrem), v tomto případě dochází k získání hesla současným voláním funkce *ssh_get_user_info().* Při získání hesla je generován opět nový čas. Po každém cyklu smyčky je klientovi poslána odpověď a uvolněna zpráva od klienta.

ftp_communication() – Taktéž se vytvoří záznam klienta k uchování dat, z připojení je získána IP adresa klienta, vygeneruje se čas připojení a je-li všechno v pořádku, je dle normy poslána zpráva, že je zařízení k dispozici. Na tuto zprávu klient reaguje zprávou, která obsahu jeho jméno (login). Po zpracování jména je klient vyzván také k zadání hesla. Po odeslání hesla je ukončeno spojení. Po získání jména i hesla je vygenerován nový čas. V případě neznámé zprávy je ukončeno spojení. Zpracování přijaté zprávy má na starost funkce *ftp_recv_msg()*, která zprávu rozdělí na typ zprávy (první 4 písmena) a požadovaný obsah zprávy. Zasílání zpráv provádí funkce *ftp_send_msg()*.

2.4 Semafor a sdílená proměnná

V obou módech serveru je inkrementováno počítadlo přihlášených klientů, při překročení je taktéž klient odpojen (dekrementace počítadla). Tomuto počítadlu je k dispozici proměnná ve sdílené paměti, nad kterou provádí operace (funkce *counter_op()*). Pro zajištění synchronizace při zápisu do souboru byl ve funkci *alloc_mem()* vytvořen semafor, který je v každém kritickém úseku využíván.

2.5 Ukončení spojení

Od doby vytvoření serveru je naslouchán signálům SIGCHLD, SIGQUIT a SIGINT, které jsou nasměrovány do funkce *signal_handler()*. SIGCHL je používán při ukončení procesu klienta a to pro vyzvednutí jeho hodnoty aby se tak zabránilo zombie procesům. Signály SIGINT a SIGQUIT volají funkci *clear_mem()*pro uvolnění alokované paměti a zrušení semaforu a následně ukončí běh programu.

3 Závěr

3.1 Test.sh

K testování - spouštění serveru a více klientů současně, jsem naimplementoval bash skript. Tento skript má stejné spouštěcí parametry jako server, který chcete spustit pod určitým módem, navíc je potřeba přidat parametr „-n“ s argumentem vyjadřující počet klientů, kteří se pokusí připojit.

3.2 Statistiky

Seznam souborů:

1. fakesrv.cpp
   - Řádků: 695
   - Slov: 2103
   - Znaků: 17179
2. fakesrv.h
   - Řádků: 80
   - Slov: 164
   - Znaků: 1392
3. Makefile
   - Řádků: 41
   - Slov: 92
   - Znaků: 886

   Soubor byl převzat z předmětu IPK a následně upraven.
4. test.sh
   - Řádků: 79
   - Slov: 235
   - Znaků: 1425
5. README.txt
6. Manual.pdf

Počet souborů: 6