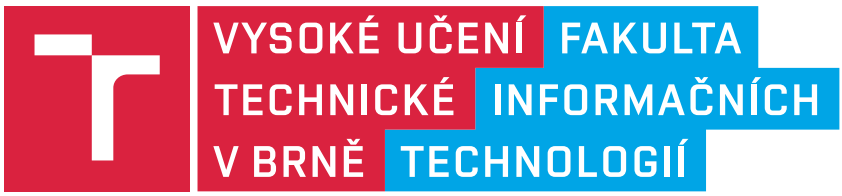

Dokumentace k projektu do předmětu ISA

FTP/SSH Honeypot

22. listopadu 2015

Autor: Sára Škutová, xskuto00@stud.fit.vutbr.cz

Obsah

1 Úvod

Tato dokumentace popisuje problematiku a řešení projektu FTP a SSH Honeypotu do předmětu Síťové aplikace a správa sítí. V první části se zaměří na vysvětlení termínu Honeypot a přiblížení problematiky FTP a SSH protokolů. Následně se popíše návrh a implementace jednotlivých části programu. Dále se vysvětlí spuštění a v závěru najdete seznam použité literatury.

2 Vysvětlení problematiky

V této části se popisuje pojem Honeypot. Dále jsou zde vysvětleny části FTP a SSH protokolů, které jsou důležité pro účely Honeypotu.

2.1 Honeypot

Honeypot je označení pro informační systém, jehož účelem je přitahovat potencionální útočníky a zaznamenávat jejich činnost (Wik, 2015).
Funguje tedy jako ochrana a detekce nežádocího přístupu (Malware, Spam . . .) Můžou se chovat jako klient či jako server, dále mohou mít plnou implementaci služby, kterou mají chrání a nebo pouze částečnou. V tomto projektu funguje jako server, který čeká až se na něho jednotliví uživatelé připojí. Následně si o nich zaznamená: čas ve kterém se připojili, jejich adresa, uživatelské jméno a heslo.

2.2 FILE TRANSFER PROTOCOL (FTP)

Tato sekce má pouze informativní charakter a nepopisuje celou dokumentaci FTP protokolu, více informací proto hledejte v RFC souboru daného protokolu (J. Postel, 1985).

Tento klient/server protokol slouží k přenosu souboru mezi počítači. Ke své funkčnosti využívá protokol TCP (protokol transportní vrstvy), který umožňuje mezi dvěma počítači vytvořit spojení, po kterém se komunikuje. V FTP se standartně využívájí 2 TCP spojení a to na portech 20 a 21. Port 20 slouží jako datové připojení – posílají se jim data, kdežto port 21 slouží k řízení přípojení – posílají se jim FTP příkazy. V tomto projektu se datové připojení nevytváří a nebudeme se jim zabývat.

K řízení připojení se požívá Telnet protokol, který defunuje syntax FTP příkazu. Každý příkaz je tzv. Telnet řetězec, kretý se skládá ze tří části: alfanumerického kódu, mezery a řetězce vázajícího se k danému kódu, to vše je ukončeno kombinací znaků <CRFL>, příklad: `USER <SP><username><CRLF>`. Klient posílá na server FTP příkazy a server na jednotlivé příkazy odpovídá stavovými kódy. Z pohledu Honeypotu jsou důležité tyto klientské příkazy: USER a PASS, kde v řetězci za USER se ukrývá přihlašovací jméno uživatele a za PASS se ukrývá heslo. Na klientské příkazy odpovídá server reply kódy, důležité v tomto projektu jsou kódy: 220, 331, 332 a 530. 220 pro přivítání uživatele na serveru, 332 pro žádost o login, 331 – žádost o heslo, 530 – o informování uživatele, že autentizace byla neúspěšná (toto chování je požadavek implementace Honeypotu). Reply řetězce jsou taktéž Telnet řetězce, ale klient využívá pouze daný kód, a řetězec obsažený za kódem má pouze informativní charakter pro uživatele.

2.3 The Secure Shell (SSH) Transport Layer Protocol

Tato sekce má pouze informativní charakter a nepopisuje celou dokumentaci SSH protokolu, více informací proto hledejte v RFC souboru daného protokolu (Ylonen, 2006).

SSH protokol umožňuje požívat zabezpečené připojení na nezabezpečeném médiu (Internet) a ke své funkčnosti využívá TCP/IP. Díky svému zabezpečení (šifrování dat, kryptografické autentizaci hosta . . .) umožňuje přeposílaní hesla aniž bychom se museli obávat možného odposlechu. Standardně server poslouchá na příchozí spojení na portu 22. SSH je možné využít místo protokolu Telnet, nebo umožní zabezpečit protokol FTP, má také mnoho dalšího využití. Spojení se serverem inicializuje klient.

Poté co se klient připojí, tak si se serverem vymění informace, s jakou verzi protokolu pracují (v tomto projektu se použila standartní verze SSH 2.0). Následně proběhne výměna informací o šifrovacím algoritmu a klíči, díky kterému se server a klient autentizují. Je také možné nevyužít šifrovací algoritmus, ale velmi se ne nedoporučuje, když pak použití zabezpečeného protokolu ztrácí smysl. Po výměně klíčů, bude moct server od uživatele získat potřebná data a to login a heslo.

3 Návrh aplikace

Honeypot má sloužit jako FTP nebo SSH server a zaznamenávat si informace o připojujících se uživatelích. To v jakém módě se má server spustit, bude zadáno v parametrech příkazové řádky. Aplikace je schopná pracovat jak v IPv4 tak v IPv6 – to jaký protokol se použije, se pozná dle adresy zadané v parametrech programu. Program je celkově rozdělen na tři části – zpracování parametrů příkazové řádky, FTP server, SSH server. Servery mají být konkurentní (umožňovat obsluhu více klientu najednou), to se docílí použití vláken, kde každý klient má své vlastní vlákno.

4 Implementace

Zde se popisuje implementace jednotlivých části programu. Server je napsán v jazyce C/C++, objektově orientovaný návrh využit nebyl, použily se pouze některé objekty jazyka C++. Program byl vyvíjen a testován na virtuálním počítači Ubuntu 14.04 LTS.

4.1 Zpracování parametrů příkazové řádky

Parametry příkazové řádky jsou zpracovávány ve funkci `getParams()`, která vrací jednotlivé parametry ve struktuře pro parametry. Ve funkci `getParams()`, se o zpracování parametru stará funkce `getopt()`, která v případě špatně zadaných nebo chybějících parametrů, nastaví indikaci chyby. To samé, i pokud se nějaký parametr opakuje. V případě indikace chyby u parametrů se vypíše chybové hlášení a nápověda. Mód je možno zadávat ve formě FTP nebo ftp. Dále je ověřováno, zda číselné parametry (port, maximální počet najednou připojených klientů, počet pokusu o zadání hesla) jsou ve správném rozsahu a/nebo nejsou v záporných hodnotách. Na závěr se kontroluje, zda je možno vytvořit/otevřít soubor pro ukládání záznamů.

4.2 FTP server

FTP server je naprogramován pomocí BSD soketů. To zda se použije IPv4 nebo IPv6 protokol se rozhodne na základě funkce `getaddrinfo()`. Následně se vytvoří soket, nakonfiguruje se server a spustí se čekání na příchozí připojení od klientů. (Hall, 2015).

Po připojení klienta se kontroluje zda není překročen maximální počet obsluhovaných klientů, pokud ne, je klientovi vytvořeno vlákno a začíná jeho obsluha, v případě, že limit je překročen, tak se danému klientu uzavře soket, čímž se klient odpojí.

Ještě před spuštěním vlákna se získá adresa klienta pomocí funkce `getpeername()`. Pro obsluhu klientů je zde funkce `ftpCom()`, která řídí celou FTP komunikaci (posílá a přijímá zprávy/reply kódy): `220, 331, 332, 530`. To který kód se má požít se rozhodně na základě stavového automatu. Poté co server získá heslo, uloží si aktuální čas v požadovaném formátu (požije se funkce `strftime()`), následně se záznam uloží do souboru, odpojí klienta a ukončí vlákno.

Pokud vše proběhlo v pořádku, tak se v daném logovacím souboru objeví celý záznam. Pokud se klient odpojil od serveru před zadáním hesla, tak v záznamu bude chybět položka heslo a pokud se odpojil ještě před zadáním uživatelského jména – login a heslo budou chybět.

4.3 SSH server

SSH server je naprogramován pomocí knihovny libssh(ISA).

Příprava serveru na příjem klientů je podobná BSD soketům, zde se pouze využívají funkce z libssh knihovny. A podobně jako u FTP serveru i zde se stejným způsobem kontroluje maximální počet klientů. Dále se stejným způsobem získává adresa klienta, která se pak společně s dalšími daty posílá do obslužné funkce.

Pro ssh obsluhu klientů je zde funkce `sshCom()`, která na začátku provede s klientem výměnu klíčů a následně v nekonečném cyklu komunikuje s klientem pomocí zasílání a příjmu zpráv. Komunikace je ukončena pokud se klient odpojí, nebo se využijí veškeré pokusy na zadání hesla.

Po každém úspěšném pokusu o zadání hesla se zjistí čas, ve kterém bylo dané heslo zadáno (viz. FTP server) a záznam se uloží do logovacího souboru. Pokud se klient neočekávaně odpojil, bude v daném souboru záznam bez loginu a hesla.

5 Použití aplikace

Program by měl být spuštěn s právy roota. Pořadí parametrů je libovolné. Chybné parametry vypíší nápovědu.
`./fakesrv -m mode -a addr -p port -l logfile [-r rsakey] [-c maxclients] [-t maxattempts]`

- `--m mode`: určuje zda se program spustí jako ftp nebo ssh server
- `--a addr`: adresa na jaké ma server naslouchat, může být IPv4 nebo IPv6
- `--p port`: port na jakém má server naslouchat, od 0 do 65535
- `--l logfile`: logovací soubor, pokud nebude existovat, tak se vytvoří, jinak se budou nové záznamy přidávat na konec
- `--r rsakey`: **Povinný pouze u SSH**, určuje soukrommý RSA klíč
- `--c maxclients`: počet maximálně najednou obluhovaných klientů, musí být větší než 0
- `--t maxattempts`: **Nepovinný, vystupuje pouze u SSH** maximální počet pokusů o zadání hesla

Literatura

libssh 0.7.0 [online]. [cit. 22. 11. 2015]. Dostupné z: http://api.libssh.org/master/libssh_tutorial.html.

Honeypot [online]. 2015. [cit. 22. 11. 2015]. Dostupné z: https://cs.wikipedia.org/wiki/Honeypot.

HALL, B. *Beej's Guide to Network Programming Using Internet Sockets* [online]. 2015. [cit. 22. 11. 2015]. Dostupné z: http://beej.us/guide/bgnet/output/html/multipage/index.html.

J. POSTEL, J. R. *RFC 959: FILE TRANSFER PROTOCOL (FTP)* [online]. 1985. [cit. 22. 11. 2015]. Dostupné z: https://www.ietf.org/rfc/rfc959.txt.

YLONEN, T. *RFC 4253: The Secure Shell (SSH) Transport Layer Protocol* [online]. 2006. [cit. 22. 11. 2015]. Dostupné z: https://www.ietf.org/rfc/rfc4253.txt.