Programujeme trojské koně aneb základy soketového programování

Vložil/a czokl [1], 19 Duben, 2006 - 19:36

- Networks & Protocols [2]
- Programming [3]
- Virus & Worms [4]

V tomto článku se Vám pokusím vysvětlit, jak jednoduše a efektivně vytvořit zadní vrátka do systému nebo jednoduchého trojského koně. Sice můžete namítnout, že takového softu existují tuny a máte pravdu, ale spoustě z nich je tak profláklých, že nastrčit jej někomu je zázrak hodný kouzelníka, jiné zase nemají potřebné funkce nebo jich mají zbytečně moc, případně hrozí, že daný prográmek může ovládat někdo jiný než vy atd.

Poznámka: pro plné zvládnutí článku by měl čtenář znát základy tcp/ip protokolů, jazyka c++ a programovací nástroje pro tento jazyk v operačních systémech windows/linux. jakákoli znalost BSD-sockets modelu je výhodou.

Návrh

Nejdříve je potřeba se zamyslet nad tím, co přesně chceme naprogramovat. Rozhodně nedoporučuji návrh podceňovat a u každého většího projektu (myslím tím projekt řádově v tisících řádcích kódu a více) je návrh jednou z nejvíce důležitých částí. Čím hůře navržený projekt, tím těžkopádnější bude programování a horší funkčnost výsledku, ale "teorie programování" sem nepatří, takže se radši vrhneme na náš problém.

Protokol

Jistě víte,že jakékoli dva subjekty či objekty komunikují na základě nějakých obecných pravidel, protokolů. Ne jinak je to s programy typu klient-server, který bude výsledkem našeho snažení. Jistě také víte, že internet funguje na základě protokolů TCP/IP, proto budeme volit i my protokol z této rodiny. Vybrat můžete prakticky jakýkoli protokol. Například pokud chystáte trojánka, který bude v krátkém limitu zasílat obrovské množství dat (např. video a audio přenosy), měli byste volit udp, pokud je situace opačná a záleží hlavně na tom, aby všechny packety dorazili k cíli, měli byste volit tcp. A nebo se vykašlat na konvence a použít exotičtější protokol. Výhoda tohoto postupu je lepší maskování, protože málokdo čeká aplikačních dat například v icmp protokolu (v některém dalším článku se chystám ukázat jak nakládat v raw sokety. Ukážu vám i jak takový icmp klient-server vytvořit).

Princip

Zde záleží na jediné věci - zda IP adresa vaše nebo oběti je veřejná (dostupná ze sítě internet). V případě, že hodláte "nakazit" trojánkem počítač s veřejnou IP (např. veřejný server), tak navazování spojení (u nespojovaných protokolů jako UDP se rozumí navazováním spojení ten, kdo první vyšle paket) necháme v programu klient, kterým budeme funkce trojana ovládat. Pokud ovšem oběť IP adresu veřejnou nevlastní, ale vy ano, necháme navazování spojení na klientu (použití tohoto principu u prvního případu není výhodné). V případě, že ani jeden z počítačů veřejnou IP nemá, musíme se obrátit na služby třetí strany (např. použití http serveru ve spojení s php). Nástin těchto situací si též ukážeme.

Jazyk a platforma

Neméně podstatnou částí při rozhodování jsou programovací nástroje a operační systém. Všechny moje příklady budou v jazyce c++, který je mým osobním favoritem, rozebírat výhody a nevýhody jednotlivých jazyků je daleko nad rámec tohoto článku, moje osobní doporučení je ovšem, že pro jednoúčelový trojan je lepší použít jazyk, který podporuje přímo API systému (např. c++ ve Windows), pro multifunkční nástroj zase nejlépe platformě nezávislý jazyk. Ale takové jazyky bývají převážně skriptovací, takže riskujete obtíže s interpretem. Jak už jsem předeslal dříve, moje příklady bodu v jazyce c++, které jsem programoval a zkoušel na systémech Windows XP a Linux Slackware 10.1.

Začínáme

Knihovna Winsock a "Socket API" v linuxu budou základními kameny pro programování TCP a UDP komunikace. Oba systémy vycházejí naštěstí ze stejného základu (BSD), takže budou odlišnosti mezi platformami minimální. Asi by nebylo od věci vysvětlit také co se myslí pojmem socket. Socket je zjednodušeně řečeno pomyslná roura, kterou proudí data z jednoho počítače na druhý, ovšem soket sám o sobě se o přenos nijak nestará, vytvoření soketu analogicky odpovídá například položení vodovodního potrubí. K používání takového potrubí potřebujeme připojit zdroj vody (server), který bude vodu distribuovat (funkce bind, u TCP navíc funkce Listen, jakési čidlo čekající na zájemce o vodu a Accept, který zapne přívod vody ze zdroje).také je potřeba zájemce o vodu (klient), který je-li potřeba o vodu požádá (opět se týká pouze TCP protokolu, funkce connect). A pak samozřejmě můžeme vodu už bez problémů přenášet od zdroje k žadateli (funkce sendto a recv). Dokonce v případě soketů je přenos dat oboustranný.

UDP

Poznámka: ještě než se pustíme do realizace, tak dvě poznámky. První je k organizaci a to, že nejdříve ve zkratce zopakuji co bude samotný kód dělat a pak popíšu slovy jak (nazval jsem to nástin postupu), další info najdete v bohatě okomentovaných zdrojových kódech. Druhá věc, činnost programu je velice jednoduchá a né příliš dokonalá, ale nechtěl jsem tam tahat další věci, které se netýkají soketů, proto servery "pouze" spustí program na hostitelském počítači pomocí fce shellexecute() resp. execl() a ukončí se.

Klient - inicializuje a odesílá rozkazy

Nástin postupu:
(0) Inicializace Winsock knihovny - platí pouze pro windows)
1) Přebrání vstupních parametrů a převod hostname na ip adresu - funkce gethostbyname(), struktura hostent
2) Určíme kam se bude připojovat(typ protokolu, ip adresa, port) - naplnění struktury sockaddr_in
3) Vytvoříme soket UDP služby - funkce socket()
4) Přečteme příkaz a odešleme - cin.getline() a sendto()
5) Přijmeme výsledek - recvfrom()
6) Ukončíme práci - (WSACleanup - opět pro windows), closesocket() resp. close()

Server

- jak už bylo řešeno server bude přijímat komunikaci a vykonávat rozkazy, bude tedy umístěn u oběti a bude vyčkávat na připojení se klienta

Nástin postupu:
(0) Inicializace Winsock knihovny - platí pouze pro windows)
1) Určíme odkud se bude spojení přijímat(typ protokolu, ip adresa, port) - naplnění struktury sockaddr_in
2) Vytvoříme soket UDP služby - funkce socket()

3) Asociace serveru s portem - fce bind()
4) Vyčkáme až se klient připojí a přijmeme příkaz, který se má vykonat - recvfrom()
5) Příkaz vykonáme - fce shellexecute() resp. execl()
6) Odešleme potvrzení - sendto()
7) Ukončíme práci - (WSACleanup - opět pro windows), closesocket() resp. close()

Poznámka: Kód linux serveru je maličko složitější, aby se zbytečně nečekalo je tam použitá fce fork(), viz zdrojové kódy, kde je vše popsáno.

UDP klient windows verze [5]
UDP server windows verze [6]
UDP klient linux verze [7]
UDP server linux verze [8]

TCP

Klient i server budou dělat to samé co u udp, ale v samotném kódu bude několik změn.

Klient

Nástin postupu:
(0) Inicializace Winsock knihovny - platí pouze pro windows)
1) Přebrání vstupních parametrů a převod hostname na ip adresu - funkce gethostbyname(), struktura hostent
2) Určíme kam se bude připojovat(typ protokolu, ip adresa, port) - naplnění struktury sockaddr_in
3) Vytvoříme soket TCP služby - funkce socket()
4) Připojíme se na klienta - connect()
4) Přečteme příkaz a odešleme - cin.getline() a send()
5) Přijmeme výsledek - recv()
6) Ukončíme práci - (WSACleanup - opět pro windows), closesocket() resp. close()

Server

Nástin postupu:
(0) Inicializace Winsock knihovny - platí pouze pro windows)
1) Určíme odkud se bude spojení přijímat(typ protokolu, ip adresa, port) - naplnění struktury sockaddr_in
2) Vytvoříme soket UDP služby - funkce socket()
3) Asociace serveru s portem - fce bind()
4) Naslouchání - listen()
5) Vyčkáme až se klient připojí a připojení přijmeme - accept()
6) Přijmeme příkaz - recv()
5) Příkaz vykonáme - fce shellexecute() resp. execl()
6) Odešleme potvrzení - send()
7) Ukončíme práci - (WSACleanup - opět pro windows), closesocket() resp. close()

TCP klient windows verze [9]
TCP server windows verze [10]
TCP klient linux verze [11]
TCP server linux verze [12]

Tak a to je všechno. Doufám, že po přečtení tohoto článku budete schopni s pomocí manu a i-netu vytvářet jednoduché klient-server aplikace, mezi které patří i trojany. V dalším článku se pokusím předvést overproxy přístup, ale vzhledem k mému časovému vytížení nebude článek hned zítra, ale tak v horizontu jednoho měsíce. Dále chystám i další články o síťovém programování (již zmíněné icmp, raw sokety, možná něco o wininet, libcap/winpcap, ...), takže pokud vás toto téma zajímá určitě stay tuned, more to come :)

Reference

Winsock reference [13]
linux man

URL článku:
https://security-portal.cz/clanky/programujeme-trojsk%C3%A9-kon%C4%9B-aneb-z%C3%A1klady-soketov%C3%A9ho-programov%C3%A1n%C3%AD

Odkazy:
[1] https://security-portal.cz/users/czokl
[2] https://security-portal.cz/category/tagy/networks-protocols
[3] https://security-portal.cz/category/tagy/programming
[4] https://security-portal.cz/category/tagy/virus-worms
[5] http://data.security-portal.cz/trojan/udp_klient_win.cpp
[6] http://data.security-portal.cz/trojan/udp_server_win.cpp
[7] http://data.security-portal.cz/trojan/udp_klient_lin.cpp
[8] http://data.security-portal.cz/trojan/udp_server_lin.cpp
[9] http://data.security-portal.cz/trojan/tcp_klient_win.cpp
[10] http://data.security-portal.cz/trojan/tcp_server_win.cpp
[11] http://data.security-portal.cz/trojan/tcp_klient_lin.cpp
[12] http://data.security-portal.cz/trojan/tcp_server_lin.cpp
[13] http://msdn.microsoft.com/library/default.asp?url=/library/en-us/winsock/winsock/winsock_reference.asp