Skaner sieci IPv6

Twórcy: Olaf Gontarczyk, Konstantin Ostapenko, Hubert Redeł

Podsumowanie:

Stworzona przez nas aplikacja ma na celu wygenerowanie adresów IPv6 oraz sprawdzenia czy owy adres jest podatny na ataki DDoS poprzez nawiązywanie połączenia z siecią i sprawdzenie czy jest ona zabezpieczona czy nie. Sama aplikacja została wykonana przy wykorzystaniu C++ z implementacją biblioteki WinSock2 oraz WS2tcpip.

Użytkowanie:

First of all it's quite important to add the "ws2_32.lib" library to the project. There are two ways to do it:

~ Visual Studio > right mouse click on the project > properties > Linker > Input > Additional Dependencies > Edit > ws2_32.lib.

~ In the main file of the project add the line:

```
#pragma comment(lib, "ws2_32.lib")
```

The #include <Windows.h> should be preceded with the #define WIN32_LEAN_AND_MEAN macro, because "Windows.h" defaults to including "Winsock.h" which doesn't support ipv6. However, the WIN32_LEAN_AND_MEAN macro prevents the Winsock.h from being included, so now it's possible to include WinSock2.h.

Next step is check if IPv6 available:

~ win + x > Windows PowerShell as Admin > ipconfig /all

~ win > type cmd > right click on Command Prompt > as Admin > ipconfig /all

Before the next step, it's mandatory to refresh network settings (doesn't work for the static IP):

~ netsh winsock reset

~ netsh int ip reset

~ ipconfig /release

~ ipconfig /renew

~ ipconfig /flushdns

If there aren't IPv6 addresses, it's should be configured in the cmd:

~ Preferable solution : netsh interface ipv6 isatap set state state=enable / disable

~ Less preferable: Enable-NetAdapterBinding -Name "*" -ComponentID ms_tcpip6

~ Not recommended: netsh interface ipv6 6to4 set state state=enable/disable undoonstop=disable

~ And the last one to manage the protocol: net start tcpip6/net stop tcpip6

More related commands here: MSDN man

Po przygotowaniu środowiska opisanego w poprzednim akapicie, użytkowanie samej aplikacji jest proste. Po uruchomieniu jej, użytkownik zostanie poproszony o wprowadzenie liczby adresów jakie mają zostać wygenerowane oraz przeskanowane. Po wpisaniu liczby (ważne jest aby liczba była typu int, powód zostanie opisany w kolejnym dziale zajmującym się omówieniem klas) i zatwierdzeniu enterem, aplikacja wykona niezbędne operacje i zwróci na ekran użytkownika wiadomości jakie adresy zostały wygenerowane, z którymi możliwe jest nawiązanie połączenia i czy są podatne na ataki DDoS czy nie.

Opis Klas:

Klasa IPv6Address.h, zajmuje się bardzo prostą funkcjonalnością, mianowicie generuje oraz przechowuje adresy na których aplikacja pracuje.

Oczywiście „main.cpp" jest dużo bardziej rozbudowany. Na samym początku kodu dodawane są niezbędne definicje, deklaracje metod oraz biblioteki z których WinSock2 oraz WS2tcpip pozwalają nam łączyć się i dokonywać operacji na socketach. Sam Main zaczyna się od deklaracji sranda oraz inicjalizacji WinSock2 (aby tego dokonać wywoływana jest funkcja `WSADATA wsaData;),` a cała ta operacja jest oprawiona odpowiednim feedbackiem dla użytkownika w postaci informacji wyświetlanych na ekran użytkownika przez aplikacje. Kolejna część (linjka: 85 do 101) pobiera od użytkownika input ile adresów ma zostać wygenerowanych, niestety z powodu braku czasu i skomplikowania zadania, nie zaimplementowaliśmy kontroli poprawności wpisanych danych, ale jeżeli zaistnieje potrzeba zostaną one zaimplementowane na życzenie klienta. Ostatnim etap jest dodanie kilku adresów których istnienia jesteśmy pewni aby móc sprawdzać poprawne działanie programu na stałych danych. Ostatnia część maina (linijka 104 do 109) jest właściwym wykonaniem programu, który dla każdego adresu sprawdza czy możliwe jest nawiązanie z nim połączenia i czy jest to adres chroniony czy nie co jest podsumowywane przez informacje wyświetlone na ekranie użytkownika. Od linijki 115 zaczynają się implementacje zadeklarowanych na samym początku klas. Pierwsza z nich (linijka od 115 do 237) jest metodą pomocniczą którą używaliśmy do testów podczas pisania programu i zostanie usunięta podczas czyszczenia kodu. Natomiast funkcja rozpoczynająca się w następnych linikach (linijka od 239 do 367) wykonuje lwią część operacji. Pierwszym etapem (do linijki 265) jest stworzenie klienta oraz ustawienie odpowiednich paramentrów i rodzin aby skaner działał wyłącznie na sieci IPv6. W następnych linijkach nasza aplikacja stara się rozpatrzeć server address oraz port i jeżeli to się uda stara się połączyć z pierwszym adresem zwróconym przez „call to geraddrinfo" oraz sprawdza czy socket rzeczywiście jest właściwym obiektem. Od linijki 289, metoda łączy się z serwerem zwracając odpowiednie informacja dla użytkownika w przypadku wystąpienia błędu lub nieudane połączenia z serwerem po czym czyści WSA. Ostatnia partia linijek od 304, jeżeli zaistniało połączenie, tworzy początkowy buffer i rozpoczyna przesył danych. Na podstawie ilości wysłanych danych i tym czy server nie zamknie połączenia. Na podstawie tych danych zwracana jest informacja czy server jest podatny na atak DDoS czy nie, o czym informacja jest zwracana na ekran użytkownika.