UNIWERSYTET KARDYNAŁA STEFANA WYSZYŃSKIEGO

W WARSZAWIE

WYDZIAŁ MATEMATYCZNO-PRZYRODNICZY

SZKOŁA NAUK ŚCISŁYCH

Kacper Sądzyński

Krzysztof Gorzkowski

Kamil Chibowski

Jan Brzeziński

Tobiasz Jeska

IPv4 scanner for vulnerable protocols

WARSZAWA 2021

Spis treści

Wstęp

Program służy do poszukiwania adresów IPv4, które mogą posłużyć do ataków DDoS. Do napisania programu został wykorzystany język programowania Java. IDE wykorzystane do zbudowania tego projektu to Intellij IDEA ultimate 2021.1.

Wymagania systemowe programu

Minimalne

- Wymaga 64-bitowego procesora
- Procesor: Dual core 2Ghz+
- Pamięć: 4 GB RAM
- Miejsce na dysku: 1 GB dostępnej przestrzeni

Zalecane

- Wymaga 64-bitowego procesora
- Procesor: Quad core 3Ghz+
- Pamięć: 8 GB RAM
- Miejsce na dysku: 1 GB dostępnej przestrzeni

Funkcjonalność

Program służy do poszukiwania adresów IPv4, które mogą posłużyć do ataków DDoS. Program skanuje sieć używając następujących protokołów:

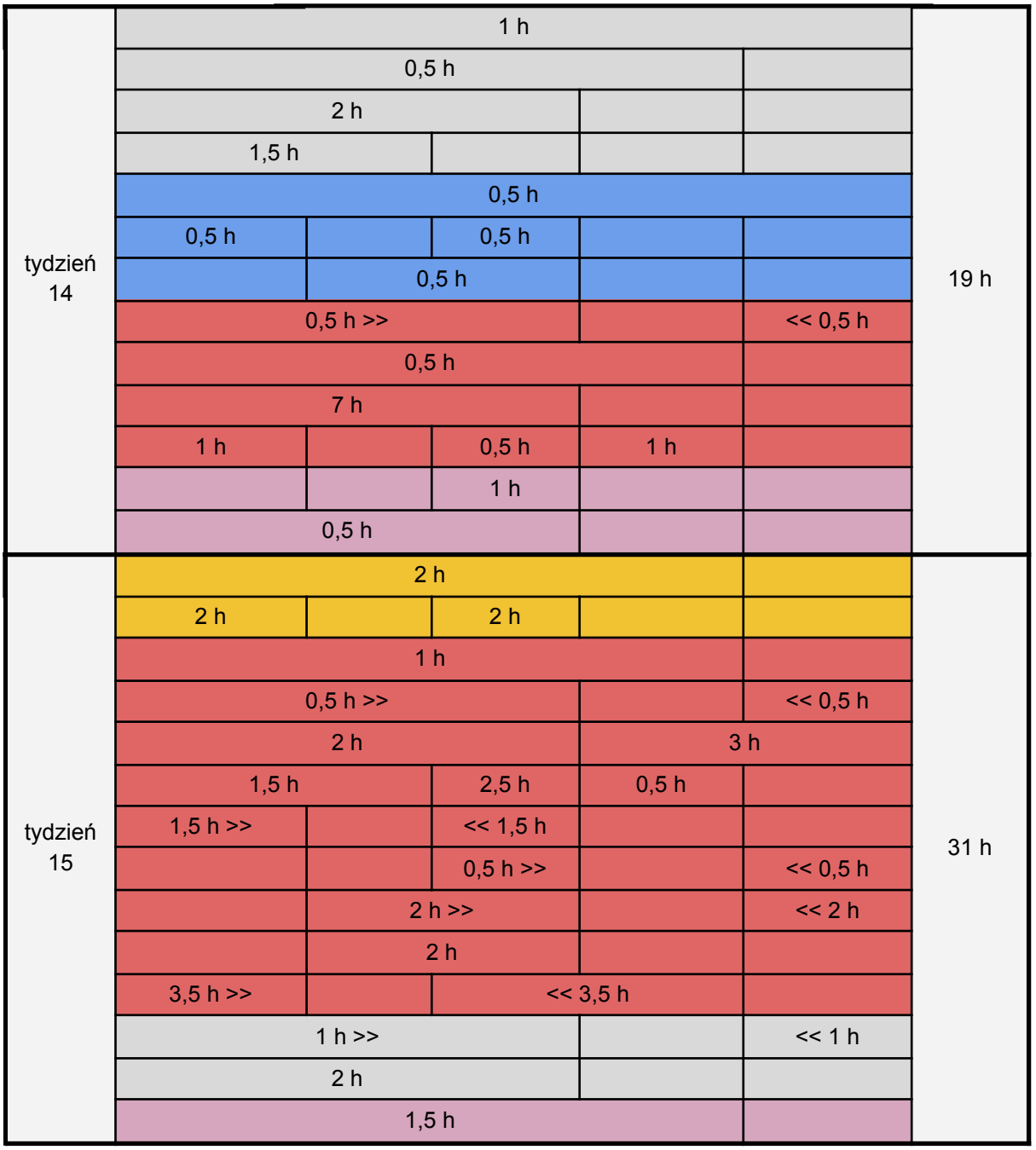
- DNS
- SNMP
- NTP
- MemCached

Program wywoływany jest z poziomu konsoli. W argumentach wywołania programu użytkownik może podać, jakich protokołów chce użyć oraz czy wynik chce zapisywać do pliku.
Za pomocą argumentu wywołania może również wyświetlić pomoc - listę dostępnych argumentów.
Za pomocą argumentów wywołania może uruchomić demonstracyjną wersję programu.

Diagram Gantta

	Krzysztof Gorzkowski	Tobiasz Jeska	Jan Brzeziński	Kacper Sądzyński	Kamil Chibowski	Czas łączny
tydzień 1	3 h					3 h
tydzień 2	9 h					9 h
tydzień 3	12 h					14 h
	2 h					
tydzień 4	4 h					43 h
	15 h		8 h	8 h	8 h	
tydzień 5	2,5 h					18 h
	4 h		2 h	3 h	3 h	
	2 h					
	1,5 h					
tydzień 6	0,75 h	1 h	2 h	1 h	2 h	17,5 h
	3,5 h					
	1,5 h	1 h	1,5 h	1 h	1 h	
			0,5 h	0,25 h	0,5 h	
tydzień 7	0,5 h					35,5 h
	2 h >>			<< 2 h		
	2,5 h					
	1,5 h					
	2,5 h	1 h	3 h	1,5 h	1 h	
		1 h				
	10 h				2 h	
	4 h					
	2 h				1 h	
tydzień 8	5,5 h		2,5 h			18,5 h
	2 h					
			3 h			
	1 h					
	1 h		1 h			
	2,5 h					
	1,5 h					

tydzień 9			1 h		1 h	12,5 h
		1,5 h				
	2 h >>		<< 2 h			
	2,5 h >>				<< 2,5 h	
	1 h			0,5 h		
	1 h					
	1 h				1 h	
tydzień 10	3 h					8,5 h
	0,5 h					
			1 h			
	3 h					
					1 h	
tydzień 11			2 h			14,5 h
	2 h					
			0,25 h			
	0,25 h					
	2 h					
	0,5 h					
	0,5 h					
				4 h	3 h	
tydzień 12	2 h					3,5 h
	1 h					
	0,5 h					
tydzień 13	0,5 h					10,5 h
	4 h					
	0,5 h >>				<< 0,5 h	
	2 h					
	1 h					
	1 h >>				<< 1 h	
	1,5 h					



LEGENDA
Czytanie o programowaniu sieciowym
Poznawanie platformy GitHub
Uczenie się Javy
Kodowanie
Dokumentacja
Testy

Kroki pośrednie

1. Utworzenie klasy Main oraz ArgAvailable, a także pliku args.txt, aby zdefiniować możliwe argumenty wywołania programu
2. Utworzenie generatora wszystkich adresów IPv4
3. Dodanie metody weryfikującej argumenty wywołania programu, przeniesienie generatora adresów do nowej klasy IPv4Addresses
4. Refaktoryzacja klasy DNSScanner - od teraz dziedziczy po klasie IPv4Addresses
5. Dodanie metody query w klasie DNSScanner umożliwiającej budowanie pakietu i wysyłanie zapytania
6. Dodano funkcjonalność pisania do pliku
7. Testowanie skanera DNS w celu znalezienia pakietów dających największą amplifikację - dodano funkcjonalność do testowania
8. Dodanie funkcjonalności do klasy SNMPScanner analogicznej do DNSScanner (konstrukcja pakietu, wysyłanie, słuchanie odpowiedzi), jednak wykorzystując paczkę snmp4j zamiast ręcznej budowy pakietu
9. Testowanie skanera SNMP
10. Refaktoryzacja DNSScanner - wielowątkowość+
11. Testowanie wielowątkowości
12. Refaktoryzacja DNSScanner - docelowy kształt
13. Dodanie funkcjonalności do klasy NTPScanner analogicznej do DNSScanner (konstrukcja pakietu, wysyłanie, słuchanie odpowiedzi)
14. Testowanie skanera NTP
15. Modyfikacja klasy IPv4Addresses w celu zwiększenia estetyki kodu
16. Dodanie funkcjonalności do klasy MemCachedScanner - budowa pakietu analogiczna do DNSScanner (skanowanie po UDP)
17. Dodanie nowej funkcjonalności do klasy MemCachedScanner - wysyłanie zapytania przy pomocy TCP
18. Refaktoryzacja klasy MemCachedScanner - utworzenie dwóch metod do skanowania odpowiednio po UDP i TCP. Metody te są wywoływane kolejno w metodzie query()
19. Stworzenie serwera MemCached na potrzeby testów
20. Testowanie skanera MemCached
21. Refaktoryzacja kodu
22. Umożliwienie uruchamiania programu z konsoli (stworzenie pliku jar)
23. Dodanie wersji demonstracyjnej
24. Dodanie paska progresu pokazującego aktualny stan programu
25. Usunięcie paczki snmp4j (problem z wielowątkowością)
26. Dodanie funkcjonalności do klasy SNMPScanner analogicznej do DNSScanner (konstrukcja pakietu, wysyłanie, słuchanie odpowiedzi) tym razem z ręczną konstrukcją pakietu
27. Końcowa refaktoryzacja i okomentowywanie kodu
28. Uzupełnienie plików help.txt i README.md

Diagram klas

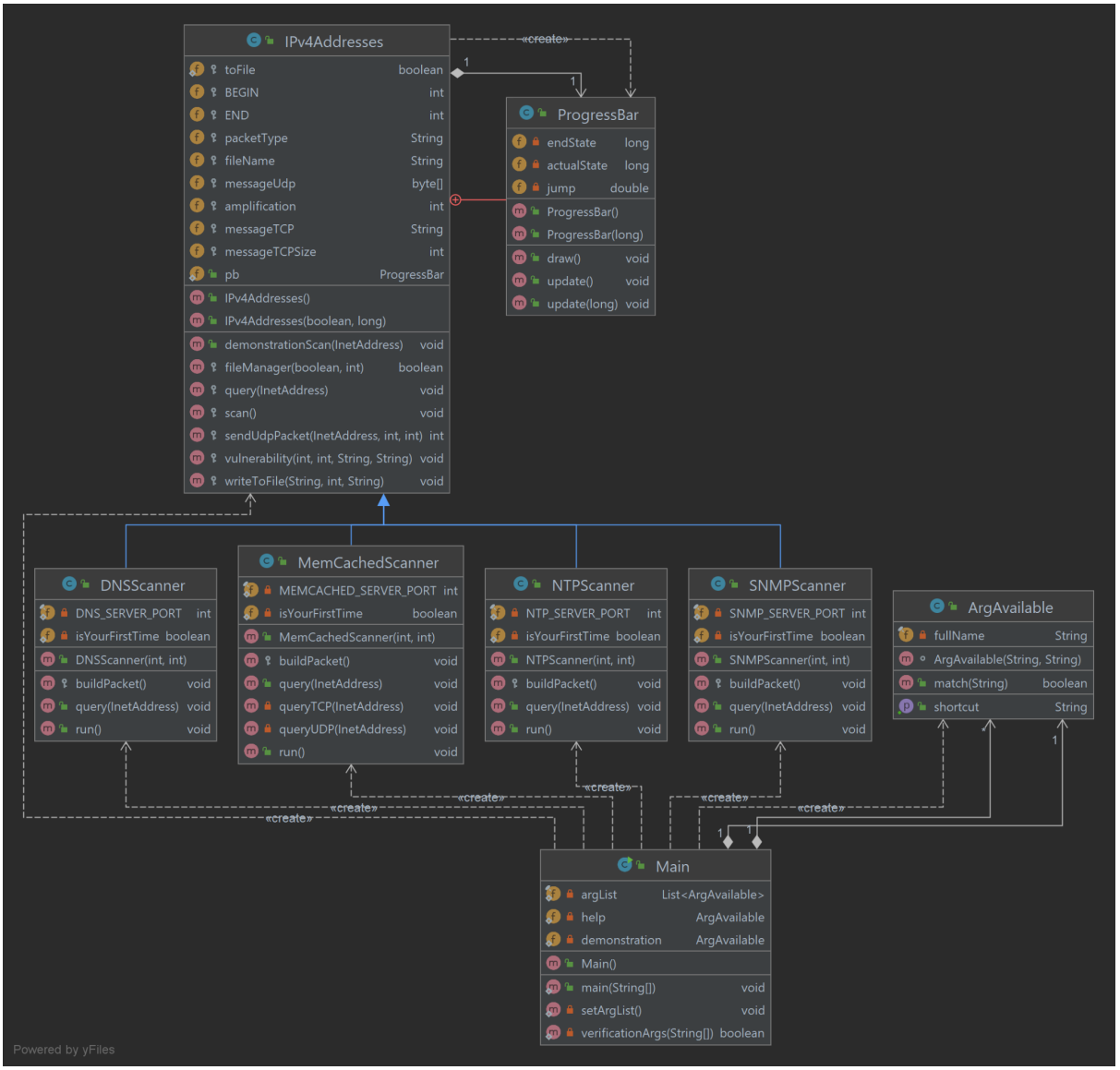


Diagram sekwencji