به نام خدا

تمرین 3 شبکه

سوال 1

الف:

Udp یک پروتکل connectionless است یعنی هیچ handshake ای بین فرستنده و گیرنده صورت نمیگیرد. و همچنین segment های این پروتکل به صورت مستقل از هم اداره میشوند. پس این پروتکل unreliable است. اگر بخواهیم کانال ان را امن کنیم باید از مکانیزم های خاصی مثل checksum استفاده کنیم.

پس امکان امن کردن کانال های udp تا حدی وجود دارد.

ب:

چه نوع خطاهایی توسط checksum و چه نوع خطاهایی توسط number sequence قابل تشخیص هستند؟

Checksum: تا حدودی میتواند تغییر بیت از صفر به یک و یا از یک به صفر را تشخیص دهد.

Sequence number: تشخیص خطاهای تغییر بسته ها و loss و duplicate

ج:

اگر خیلی کوچک فرض کنیم ممکن است ack یک بسته بعد از زمان مشخص شده برسد باید مجددا بسته ارسال شود با اینکه مشکلی ایجاد نمیکند ولی استفاده ی غیر بهینه از منابع است.

اگر خیلی بزرگ باشد واکنش ان به بسته های گمشده کند میشود و ارسال مجدد بسته ها دیرتر انجام میشود که این موضوع هم باعث اتلاف منابع میشود.

د:

مقدار Checksum Internet براي دو عدد زير را محاسبه كنيد.(نمایش اعداد بصورت hex میباشد)

0x7BE0 = 0111 1011 1110 0000

0x8653 = 1000 0110 0101 0011

Sum = 1 0000 0010 0011 0011 => add1 to number=> 0000 0010 0011 0100 =>

complement: 1111 1101 1100 1011

سوال 2

با توجه به تصوير زير مقادير پورتهاي مبدا و مقصد هر يک از بستههاي A تا D را بدست آوريد.

بسته	پورت مبدا	پورت مقصد
A	6507	6582
B	6582	6507
C	5301	6582
D	5333	6582

الف) پورت مبدا و مقصد در این بسته چیست؟

پورت مبدا: 397c

پورت مقصد: 01bb

ب)این بسته چندمین بایت را ACK می کند؟ آیا این مقدار، یک مقدار valid است؟

منتظر بایت b53d5600 هستیم و تا قبل از ان را Ack میکند.

50110203 = 0101 0000 0001 0001 0000 0010 0000 0011

A=1 => معتبر است

FIN = 1

Receive window = 0000 0010 0000 0011

ج)این بسته مربوط به کدام قسمت از یک کانکشن TCP می باشد؟

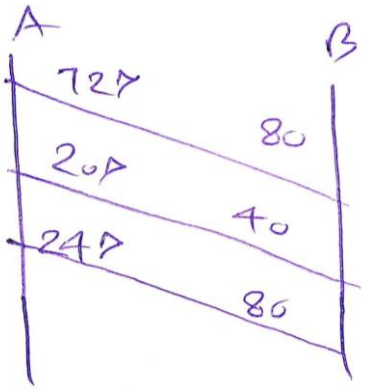
Flag FIN =1 پس این بسته شامل سیگنال کنترلی است و مربوط به قسمت قطع ارتباط است.

د) Sequence Number این بسته چه مقدار است؟ این عدد نشان دهنده چیست؟

Sequence Number=B6c14086

سوال 3

الف) در بسته سوم، مقدار number sequence و پورت مبدأ و پورت مقصد چیست ؟



Sequence number = 247

Source = 5445

Destination = 80

ب) درصورتی که بسته اول قبل از بسته دوم برسد، در بسته ACK اي که پس از دریافت بسته اول توسط گیرنده ارسال میشود، مقدار number ACK و پورت مبدأ و پورت مقصد چیست ؟

Ack = 207

Source = 80

Destination = 5445

ج) درصورتی که بسته دوم قبل از بسته اول برسد، در بسته ACK اي که پس از دریافت بسته دوم توسط گیرنده ارسال میشود، مقدار number ACK و پورت مبدأ و پورت مقصد بسته اي که اول رسیده چیست ؟

Ack = 127

Source = 80

Destination = 5445

د) اگر به دلیل شلوغی شبکه، ابتدا بسته سوم به مقصد برسد و سپس بسته اول و دوم به ترتیب به B برسند .دراینصورت ACK هایی که در هر مرحله ارسال می شود به چه صورت می باشند؟

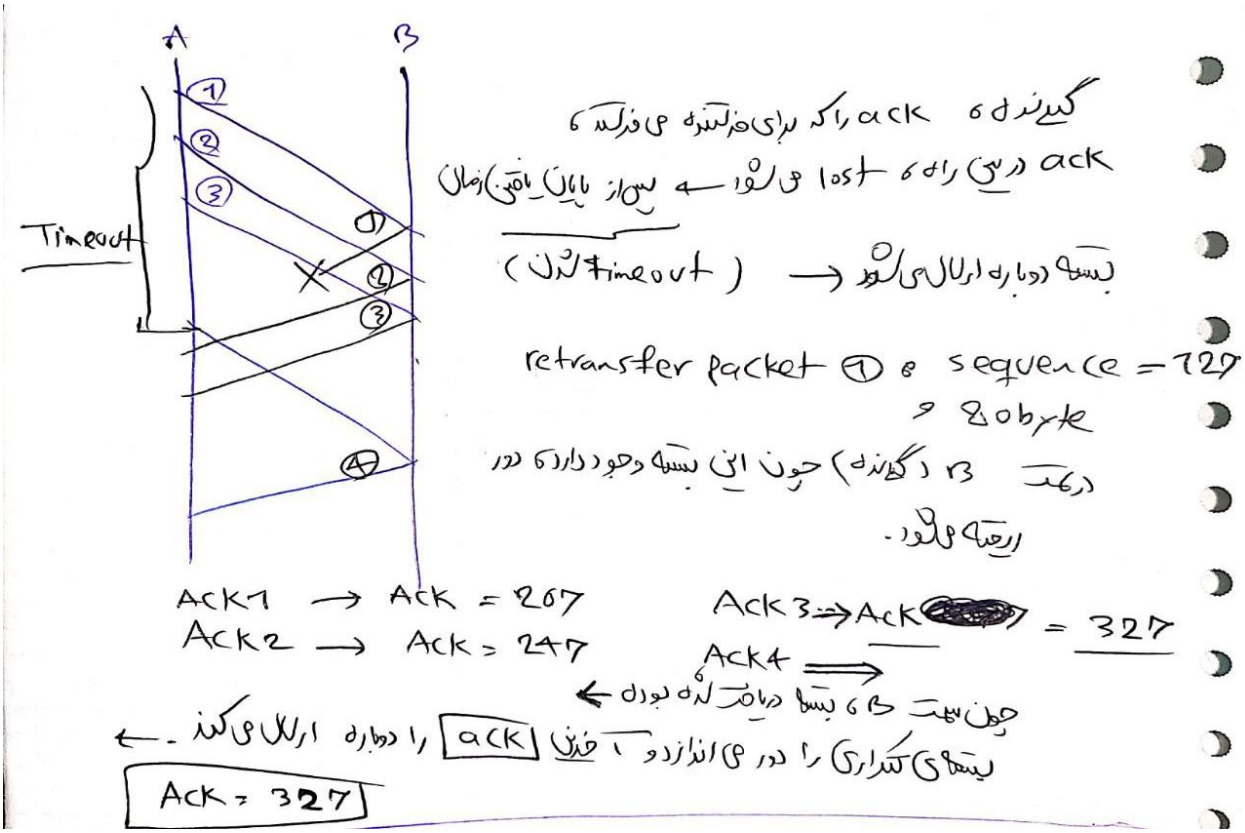
Ack = 127 =>1: Ack = 207 =>2:Ack = 327 (247+80)

ه)

بسته 1: Sequenc = 127 , 80byte

بسته 2: Sequenc = 207 , 40byte

بسته 3: Sequenc = 247 , 80byte



سوال 4

سوال ۴

DevRTT(D) = (1−β) DevRTT + β × |R − E|

Estimated RTT (E) = (1−α) × E + α × R

Timeout (T) = E + 4D

$E_0 = 720$ $D_0 = 5$

$R_1 = 128 \rightarrow D_1 = \frac{1}{4}(3 \times 5 + |128 - 720|) = 51,75 ms$

$E_1 = \frac{1}{8}(720 \times 7 + 728) = 727$ ms

$R_2 = 735 \rightarrow D_2 = \frac{1}{4}(3 \times 5,75 + |735 - 727|) = 7,8725$ ms

$E_2 = \frac{1}{8}(7 \times 727 + 735) = 722,75 ms$

$R_3 = 747 \rightarrow D_3 = \frac{1}{4}(3 \times 7,8125 + |747 - 722,75|) =$

$E_3 = \frac{1}{8}(7 \times 722,75 + 747) = 725,03$ ms

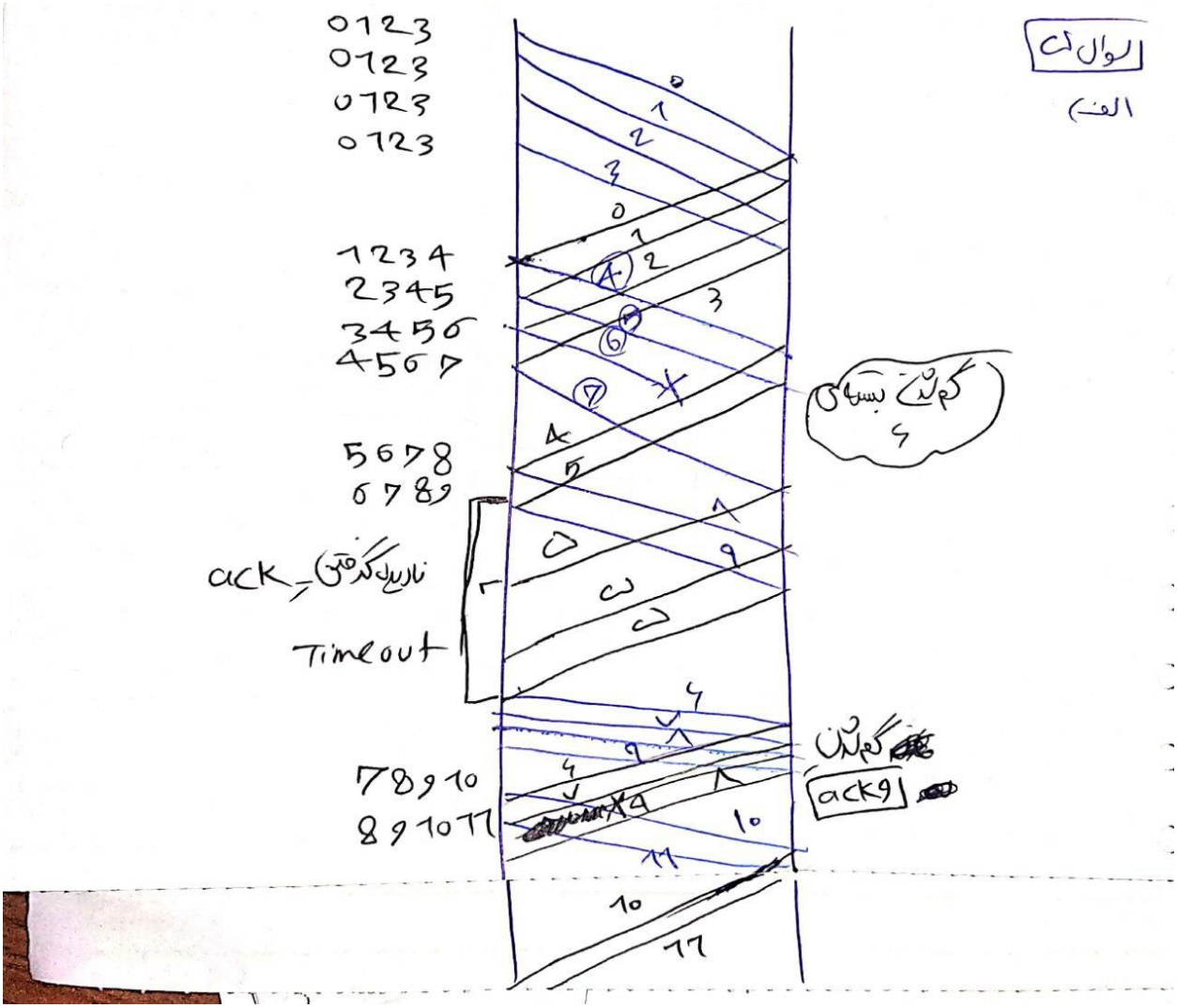
$R_4 = 770 \rightarrow D_4 = \frac{1}{4}(3 \times 10,42 + |770 - 725,103|) =$ 10,42 ms

$E_4 = \frac{1}{8}(7 \times 725,103 + 770) = 723,75$ ms

Timeout interval = 723,750 + 4 × 11,57 = 769,43ms

سوال 5

GBN



الف:

Ackهای تکراری دریافت میکند و وقتی زمان تمام میشود دوباره بسته 6 و همه ی بسته های داخل window را مجددا ارسال میکند.

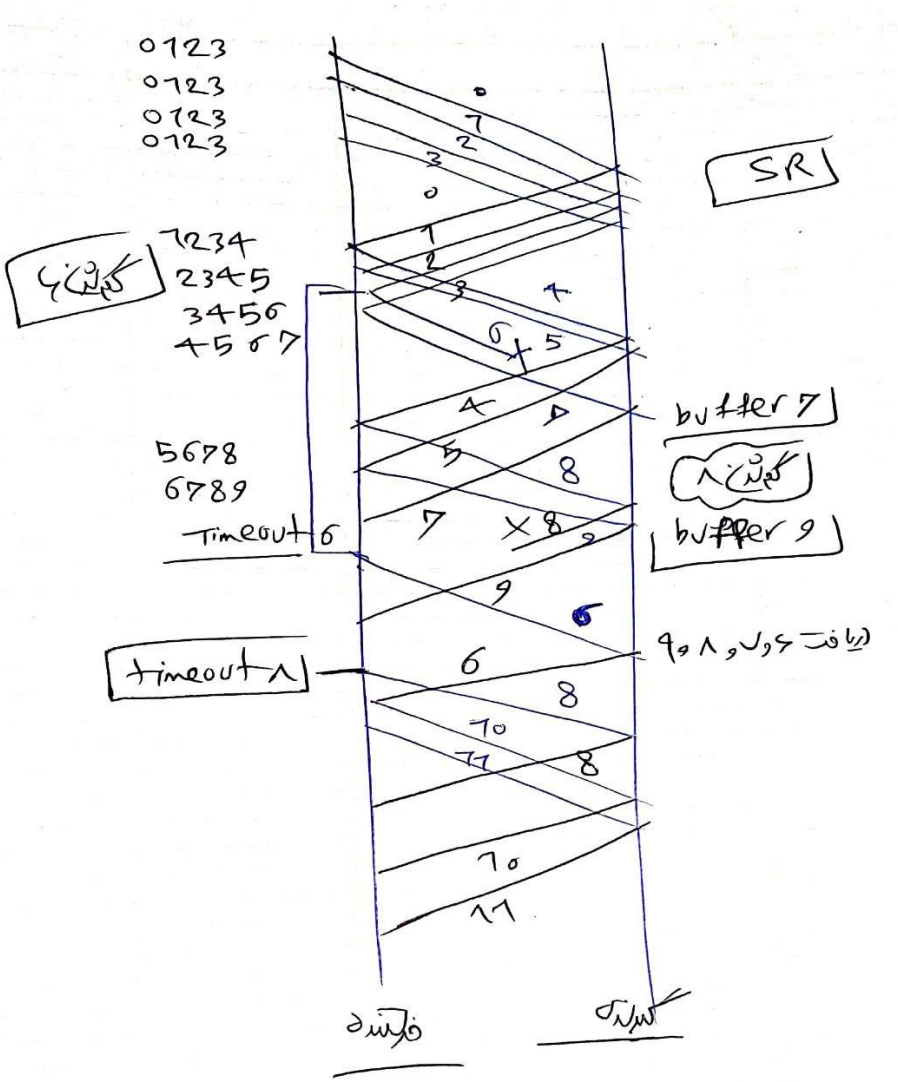
ب:

بسته ای دوباره ارسال نمیشود و اتفاقی نمی افتد چون ack9 به فرستنده میرسد

ج:

1.5RTT + 1RTT = 2.5RTT

SR



الف:

بعد از تمام شدن تاخیر مربوط به بسته ۶ ، این بسته دوباره ارسال میشود(در این نوع ،تنها همین بسته دوباره ارسال میشود) چون بسته های دیگر در بافر گیرنده ذخیره میشوند.

ب:

Ack ها تجمعی نیستند پس اگر این Ack گم شود پس از تمام شدن زمان مربوط به بسته ۸ ،دوباره ارسال میشود اما چون گیرنده ان را در بافر خود دارد ان را دور می اندازد و Ack را ارسال میکند.

ج:

1.5RTT

سوال 6

Nmap (Network Mapper) is a free and open-source network scanner created by Gordon Lyon (also known by his pseudonym Fyodor Vaskovich). Nmap is used to discover hosts and services on a computer network by sending packets and analyzing the responses.

Nmap provides a number of features for probing computer networks, including host discovery and service and operating system detection. These features are extensible by scripts that provide more advanced service detection, vulnerability detection, and other features. Nmap can adapt to network conditions including latency and congestion during a scan.

Nmap started as a Linux utility and was ported to other systems including Windows, macOS, and BSD. It is most popular on Linux, followed by Windows

features

Host discovery – Identifying hosts on a network. For example, listing the hosts that respond to TCP and/or ICMP requests or have a particular port open.

Port scanning– Enumerating the open ports on target hosts.

Version detection – Interrogating network services on remote devices to determine application name and version number.

OS detection – Determining the operating system and hardware characteristics of network devices.

Scriptable interaction with the target – using Nmap Scripting Engine (NSE) and Lua programming language.

Nmap can provide further information on targets, including reverse DNS names, device types, and MAC addresses.

Typical uses of Nmap:

Auditing the security of a device or firewall by identifying the network connections which can be made to, or through it.

Identifying open ports on a target host in preparation for auditing.

Network inventory, network mapping, maintenance and asset management.

Auditing the security of a network by identifying new servers.

Generating traffic to hosts on a network, response analysis and response time measurement.

Finding and exploiting vulnerabilities in a network.

DNS queries and subdomain search

TCP SYN (Stealth) Scan (-sS)

SYN scan is the default and most popular scan option for good reason. It can be performed quickly, scanning thousands of ports per second on a fast network not hampered by intrusive firewalls. SYN scan is relatively unobtrusive and stealthy, since it never completes TCP connections. It also works against any compliant TCP stack rather than depending on idiosyncrasies of specific platforms as Nmap's FIN/NULL/Xmas, Maimon and idle scans do. It also allows clear, reliable differentiation between open, closed, and filtered states.

SYN scan may be requested by passing the -sS option to Nmap. It requires raw-packet privileges, and is the default TCP scan when they are available. So when running Nmap as root or Administrator, -sS is usually omitted.

TCP ACK Scan (-sA)

This scan is different than the others discussed so far in that it never determines open (or even open|filtered) ports. It is used to map out firewall rulesets, determining whether they are stateful or not and which ports are filtered.

ACK scan is enabled by specifying the -sA option. Its probe packet has only the ACK flag set (unless you use --scanflags). When scanning unfiltered systems, open and closed ports will both return a RST packet. Nmap then labels them as unfiltered, meaning that they are reachable by the ACK packet, but whether they are open or closed is undetermined. Ports that don't respond, or send certain ICMP error messages back, are labeled filtered.

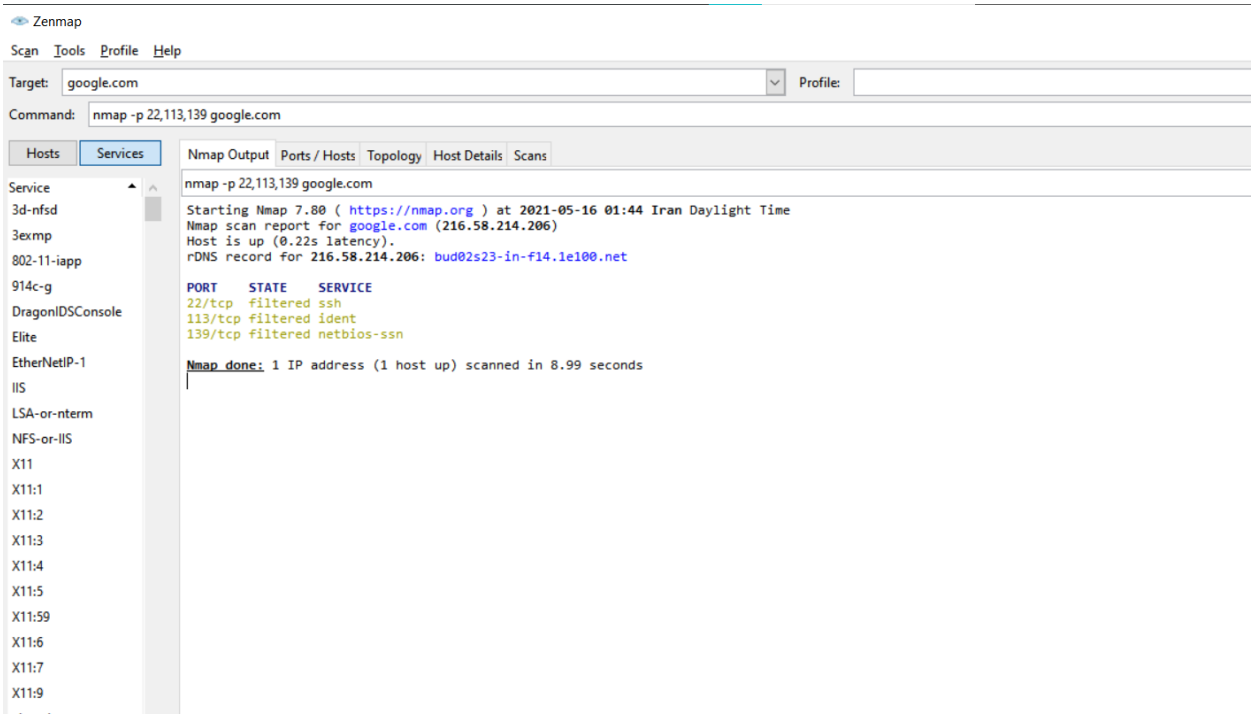
TCP Window Scan (-sW)

Window scan is exactly the same as ACK scan except that it exploits an implementation detail of certain systems to differentiate open ports from closed ones, rather than always printing unfiltered when a RST is returned. It does this by examining the TCP Window value of the RST packets returned. On some systems, open ports use a positive window size (even for RST packets) while closed ones have a zero window. Window scan sends the same bare ACK probe as ACK scan

This scan relies on an implementation detail of a minority of systems out on the Internet, so you can't always trust it. Systems that don't support it will usually return all ports closed. Of course, it is possible that the machine really has no open ports. If most scanned ports are closed but a few common port numbers (such as 22, 25, and 53) are open, the system is most likely susceptible. Occasionally, systems will even show the exact opposite behavior. If your scan shows 997 open ports and three closed or filtered ports, then those three may very well be the truly open ones.

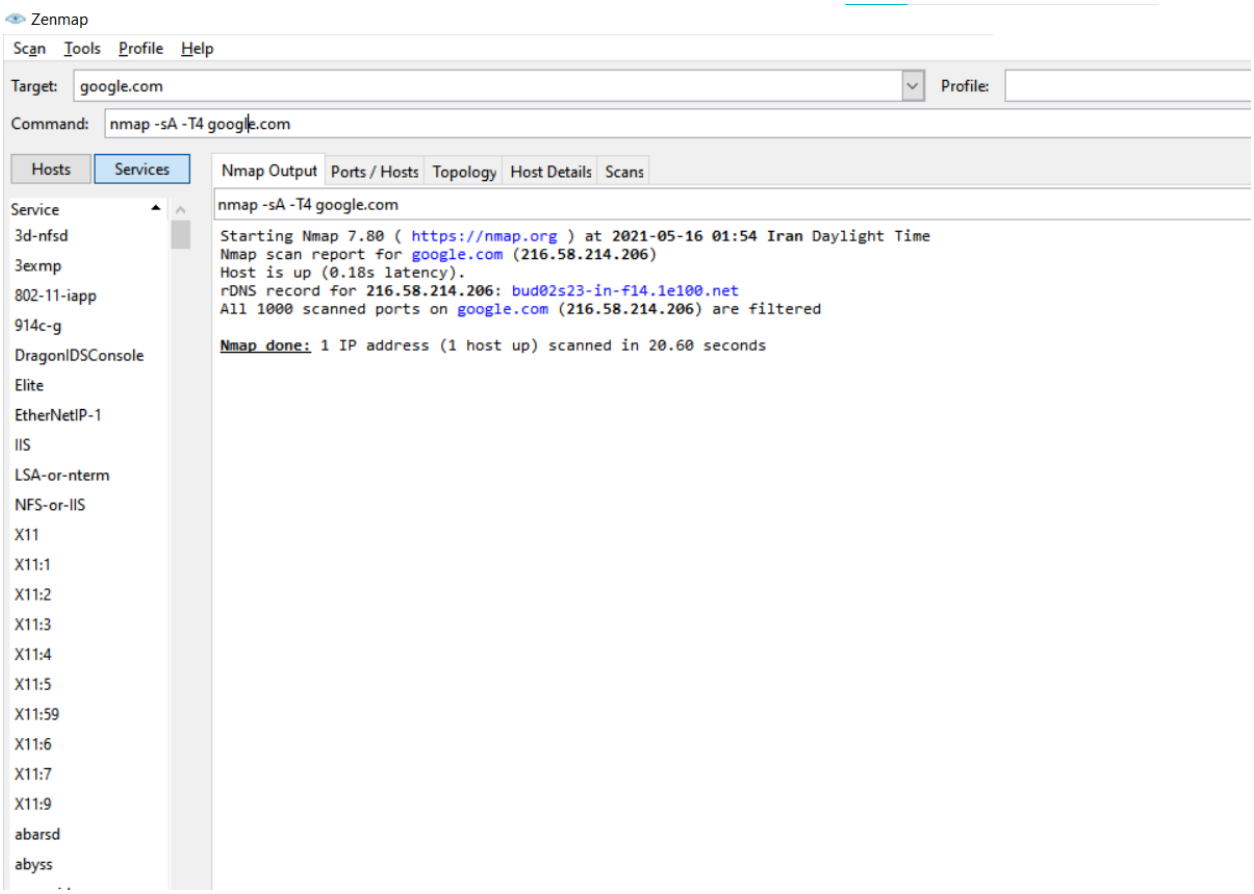
TCP SYN (Stealth) Scan (`-sS`)

nmap -p22,113,139 google.com



TCP ACK Scan (-sA)

nmap -sA -T4 google.com



TCP Window Scan (-sW)

nmap -sW -T4 geeksforgeeks.org