Vaja 1 - Wireshark (ICMP, HTTP)

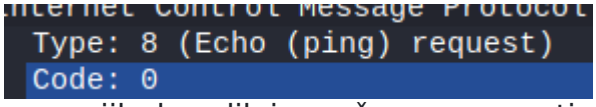
ICMP

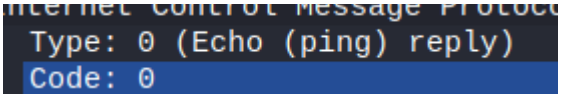
Koliko paketov se dejansko prenese med izvorom in ciljem?
Prenese se 20 paketkov, od tega 10 paketkov za zahtevo (oz. angl. “request”) in 10 paketkov za odgovor (oz. angl. “response”), sporočil v parih. Dolžina posameznega paketka v paru je 100 bajtov, od tega je 48 bajtov podatkov.

No.	Time	Source	Destination	Protocol	Length	Info
83	5.366016277	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=1/256, ttl=64 (reply in 84)
84	5.366699045	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=1/256, ttl=121 (request in 83)
104	6.367937350	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=2/512, ttl=64 (reply in 105)
105	6.368357247	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=2/512, ttl=121 (request in 104)
130	7.369757854	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=3/768, ttl=64 (reply in 131)
131	7.370437371	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=3/768, ttl=121 (request in 130)
151	8.371106914	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=4/1024, ttl=64 (reply in 152)
152	8.371771132	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=4/1024, ttl=121 (request in 151)
182	9.372570770	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=5/1280, ttl=64 (reply in 183)
183	9.373294009	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=5/1280, ttl=121 (request in 182)
213	10.374052004	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=6/1536, ttl=64 (reply in 214)
214	10.374692213	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=6/1536, ttl=121 (request in 213)
233	11.375827615	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=7/1792, ttl=64 (reply in 234)
234	11.376309828	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=7/1792, ttl=121 (request in 233)
250	12.376867568	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=8/2048, ttl=64 (reply in 251)
251	12.377292245	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=8/2048, ttl=121 (request in 250)
273	13.378681117	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=9/2304, ttl=64 (reply in 274)
274	13.379184134	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=9/2304, ttl=121 (request in 273)
290	14.380579535	164.8.208.205	164.8.8.99	ICMP	100	Echo (ping) request id=0xd3bb, seq=10/2560, ttl=64 (reply in 291)
291	14.381350582	164.8.8.99	164.8.208.205	ICMP	100	Echo (ping) reply id=0xd3bb, seq=10/2560, ttl=121 (request in 290)

Iz zgornje slike je razvidno število paketkov, vidnost parov med pošiljanjem ter začetni IP in končni IP naslov.

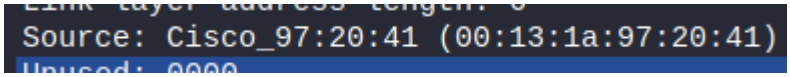
Kakšnega tipa so paketi za zahtevo in kakšnega tipa so paketi za odgovor?
Request paketki so tipa 8, response pa tipa 0.





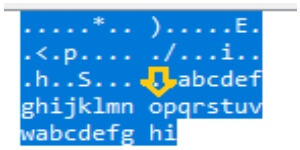
Iz zgornjih dve slik je možno prepoznati različne tipe paketkov. Levi je request paketek, desni pa response paketek.

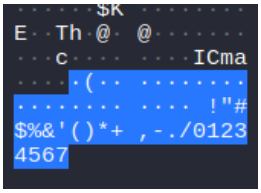
Zapišite MAC naslov mrežne kartice, ki se nahaja v ciljnem računalniku.
Mac naslov ciljnega računalnika se nahaja v source response paketka.



Iz zgornje slike je razvidno, da je Mac naslov 00:13:1a:97:20:41

Koliko zlogov se prenaša v polja "Data" in kakšna je vsebina polja?
V “Data” polju se na windowsih pošilja 32 bajtov podatkov, med tem ko pri Linuxu pa se ponavadi pošilja 48 bajtov podatkov, vendar je lahko količina podatkov dosti večja. Vsebina polja je pri windows sistemih v obliki abecede s ponavljajočo se abecedo na koncu. Linux pa je v obliki heksadecimalnih podaktov.





Prva slika je na windows sistemih, desna pa na linux.

Komentirajte razlike in podobnosti med izvedbo ukaza ping na sistemu Windows in Linux. Razlike utemeljite.

Manjše razlike so v uporabi ukaza, vendar je glavna funkcionalnost enaka. Največja razlika je v načinu zapisa in količini podatkov. Če želimo na Linux sistemih uporabiti -n končnico, ki je v windowsih rezervirano za količine paketov nam stvar ne bo delala. Težava je namreč, da ima linux že rezervirano -n za “no dns name resolution”, kar v praksi pomeni, da se bodo paketki vračali brez domene, temveč samo z IP naslovom. -C pa naredi točno to kar želimo in omeji količino poslanih paketkov na -c <število> (na windowsih -n <število>). Oba sistema pa omogočata širok nabor funkcionalnosti.

Ping Option	The output of the Command
a	It will give a sound when a peer can be reached.
b	It will allow you to ping broadcast IP addresses.
B	Prevents the ping to change the source address of the probe.
c (count)	It will limit you to send the number of ping requests.
d	It will set the SO-DEBUG option on the used socket.
f	This will Flood by sending hundreds of packets per second over a network.
i (interval)	This will inform you that how many successful packets have been transmitted into the specified time interval. By default value = 1 Second
I (interface address)	I will help you st set your source IP address to a specified interface IP address. It is required while pinging IPv6 link-local address. For this, use an IP address or name of the device.
l (preload)	I will define the number of packets you can send without waiting for a response. You can specify the value higher than 3 and by giving yourself superuser permissions.
n	This will display IP addresses as output rather than hostnames.
q	This will show you quiet output that will ping line displayed and summary of the ping command at the end.
T (TTL)	It will Set Time To Live.
v	It will give verbose output.
V	It will show the ping version and exit to a new command prompt line.
w (deadline)	Before you exist a ping command, it will specify the time limit, regardless of how many packets have been sent or received.
W (timeout)	It determines the time in seconds for which you need to wait for a response.

```
└─$ ping -n --help
ping: invalid option -- '-'

Usage
  ping [options] <destination>

Options:
  <destination>      dns name or ip address
  -a                 use audible ping
  -A                 use adaptive ping
  -B                 sticky source address
  -c <count>         stop after <count> replies
  -D                 print timestamps
  -d                 use SO_DEBUG socket option
  -f                 flood ping
  -h                 print help and exit
  -I <interface>     either interface name or address
  -i <interval>      seconds between sending each packet
  -L                 suppress loopback of multicast packets
  -l <preload>       send <preload> number of packages while waiting replies
  -m <mark>          tag the packets going out
  -M <pmtud opt>     define mtu discovery, can be one of <do|dont|want>
  -n                 no dns name resolution
  -O                 report outstanding replies
  -p <pattern>       contents of padding byte
  -q                 quiet output
  -Q <tclass>        use quality of service <tclass> bits
  -s <size>          use <size> as number of data bytes to be sent
  -S <size>          use <size> as SO_SNDBUF socket option value
  -t <ttl>           define time to live
  -U                 print user-to-user latency
  -v                 verbose output
  -V                 print version and exit
  -w <deadline>      reply wait <deadline> in seconds
  -W <timeout>       time to wait for response

IPv4 options:
  -4                 use IPv4
  -b                 allow pinging broadcast
  -R                 record route
  -T <timestamp>     define timestamp, can be one of <tsonly|tsandaddr|tsprespec>

IPv6 options:
  -6                 use IPv6
  -F <flowlabel>     define flow label, default is random
  -N <nodeinfo opt>  use icmp6 node info query, try <help> as argument

For more details see ping(8).
```

Leva slika windows, desna slika linux.

HTTP

Zapišite IP naslov vašega računalnika, IP naslov strežnika in različico HTTP protokola.

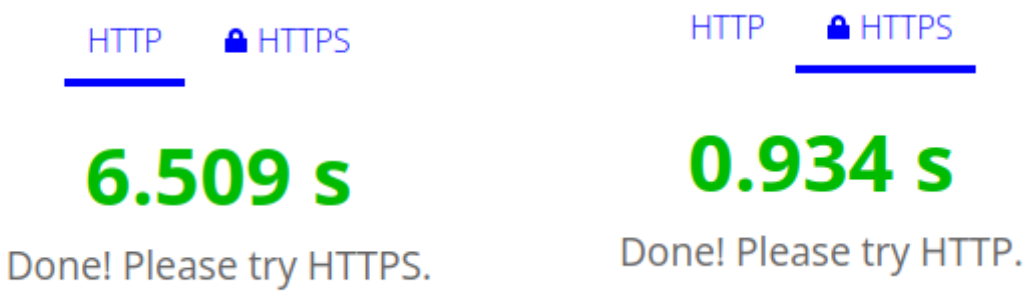
IP mojega: 164.8.161.219
IP strežnika: 45.33.7.16
HTTP protokol: persistentna povezava

```
476 6.160234985   164.8.161.219   45.33.7.16   HTTP   669 GET /check.png?16345554056171_16 HTTP/1.1
477 6.160517713   164.8.161.219   45.33.7.16   HTTP   669 GET /check.png?16345554056171_81 HTTP/1.1
478 6.160662948   164.8.161.219   45.33.7.16   HTTP   669 GET /check.png?16345554056171_10 HTTP/1.1
```

V katerih jezikih želi vaš spletni brskalnik sprejeti vsebino spletne strani? Kako se imenuje polje protokola HTTP v katerem so definirani jeziki?
Accept-Language: en-GB,en-US;q=0.9,en;q=0.8

Koliko zlogov je bilo prenešenih na vaš računalnik (vsebina HTML strani, CSS, JavaScript, Flash, slike, ...)? Koliko časa (v sekundah) je preteklo od prve zahteve vašega spletnega brskalnika do zadnje prenešene vsebine iz spletnega strežnika?

0.62 MB total

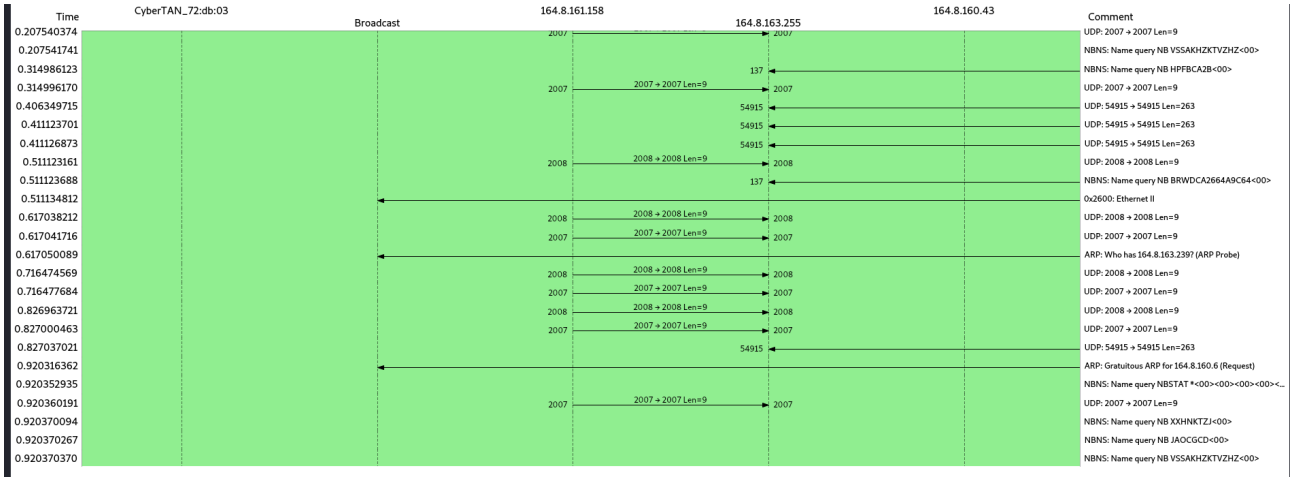


Preteklo je okoli 6.509 sekund, kar je izredno počasi. Če omogočimo HTTPS pa se čas pohitri. Povprečno se vsaj 7 krat pohitri. V praksi 10 krat.

Kakšne statusne kode in koliko le-teh je vrnil spletni strežnik?

Statusne kode so večinoma 200 OK, kar je prav. 4-x-x kode so kode z napakami na naši strani. Kode 5-x-x pa so napake na strani strežnika. 3-x-x so pa rezervirane za določene spremembe na spletni strani (preusemerjanje). Poslanih je bilo okoli 2000 paketov za cca 600 datotek.

Tvorite graf poteka prometa (flow graph) za protokol HTTP in pokomentirajte stanje.



Seveda se poveza desno ne končajo, vendar je zmanjkalo prostora na grafu. Na začetku HTTP spostavi povezavo in nato začne persistentno vlečiti dol datoteke.

Ocenjevanje časa HTTP prenosa

$d = 4$

$t_{d,f} = 0.75$ RTT

1 RTT = 120 ms

$p = 3$

$r = 2$

Nepersistentna povezava

$t_d = 2 + t_{d,f} + d(2 + t_{d,f}) = (1 + d) * (2+t_{d,f}) =$

$= 5 * 2.75 = 13.75$ RTT = 1650 ms

Nepersistentna povezava s paralelnimi povezavami

$t_d = 2 + t_{d,f} + r(2 + t_{d,f}) = (1 + r) * (2+t_{d,f}) =$

$= 3 * 2.75 = 8.25$ RTT = 990 ms

Persistentna povezava brez cevovodov

$t_d = 2 + t_{d,f} + d(1 + t_{d,f}) = 2.75 + 4 * 1.75 =$

$= 9.75$ RTT = 1170 ms

Persistentna povezava s cevovodi

$t_d = 3 + 2t_{d,f} = 3 + 1.5 = 4.5$ RTT = 540 ms