

Analisi e confronto del traffico HTTP e HTTPS in ambiente virtuale con strumenti di monitoraggio.

INTRODUZIONE:

L'obiettivo di questa attività è mettere in pratica le competenze acquisite in questo mese nell'ambito della sicurezza delle reti e del monitoraggio del traffico. L'esercizio di oggi ci propone la simulazione di un'architettura client-server in un ambiente di laboratorio virtuale, in cui un client Windows accede tramite browser a una risorsa su un server Kali Linux utilizzando il nome di dominio `internal.epicode`, risolto tramite DNS. Per la realizzazione del servizio DNS si utilizza **INetSim**, che è un software utilizzato per emulare diversi servizi Internet. Attraverso l'uso di strumenti come **Wireshark**, invece verrà analizzato il traffico generato dalle richieste **HTTPS** e **HTTP**, confrontando le differenze tra i due protocolli in termini di sicurezza, visibilità dei dati e identificazione gli indirizzi **MAC** tra sorgente e destinazione.

Dati delle macchine utilizzate:

Windows: IP: 192.168.50.102

Kali linux: IP: 192.168.50.100

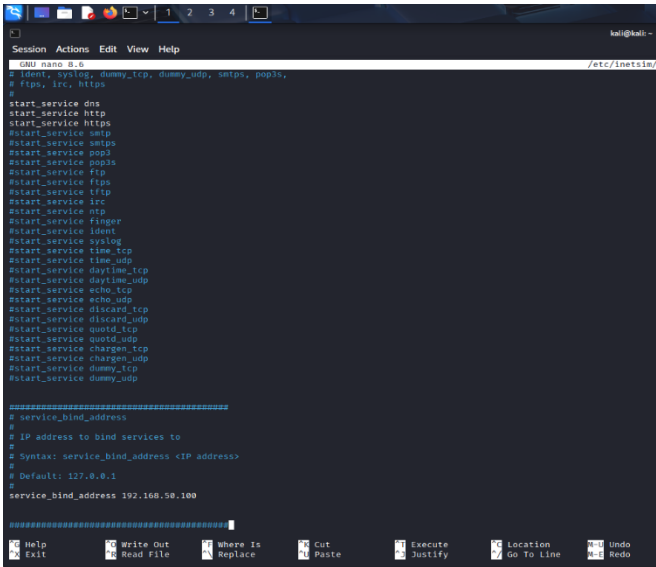
*Per comodità ho deciso di lasciare gli indirizzi già precedentemente utilizzati durante lo svolgimento delle lezioni/esercitazioni precedenti.

-In questa relazione scriverò anche tutti i problemi riscontrati durante lo svolgimento di questo esercizio con la relativa soluzione ad esso-

-Configurazione INetsim:

Per prima cosa nella macchina Kali linux, abbiamo configurato INetsim per la creazione del nostro servizio HTTP,HTTPS e DNS per configurare il tutto abbiamo utilizzato il comando sul terminale **"`sudo nano /etc/inetsim/inetsim.conf`"** un volta dentro il file, abbiamo deciso di modificare tutto ciò che ci interessava per l'esercizio, aggiungendo i commenti inserendo il **#** per i servizi non interessati e lasciando liberi quelli da utilizzare, che si trovano nella prima parte dalla voce `Start_service http`, `Start_service https` a `Start_service dns`, successivamente si impostano i parametri desiderati, nella voce `Service_bind_address` inseriamo IP di riferimento ovvero 192.168.50.100 dopo di che scendendo in basso, inseriamo i dati del dns nella voce `dns_static` con `epicode.internal` 192.168.50.100 una volta impostati questi dati possiamo salvare il tutto.

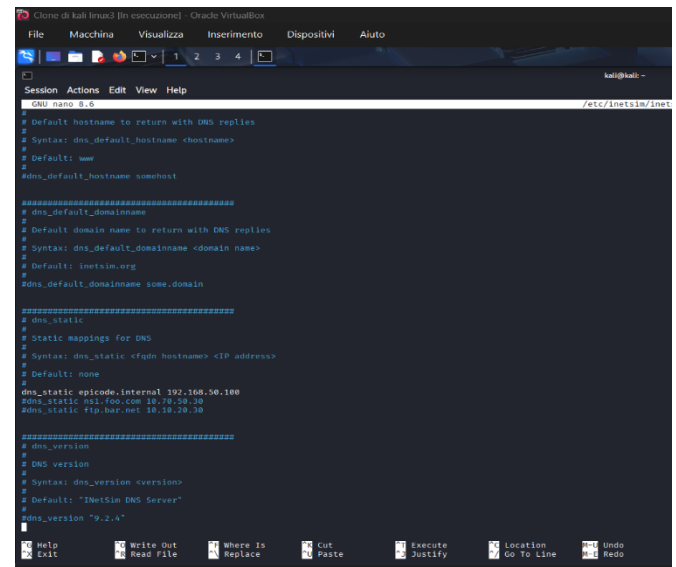
In basso le foto della configurazione di INetsim:



```
Session Actions Edit View Help
GNU nano 2.9.3 /etc/inetsim/inetsim.conf
# ident, syslog, dummy_tcp, dummy_udp, smtps, pop3s,
# ftps, irc, http,
#
start_service dns
start_service http
start_service https
start_service smtp
start_service smtps
start_service pop3
start_service pop3s
start_service ftp
start_service ftps
start_service irc
start_service ntp
start_service finger
start_service ident
start_service syslog
start_service time_tcp
start_service time_udp
start_service daytime_tcp
start_service daytime_udp
start_service echo_tcp
start_service echo_udp
start_service discard_tcp
start_service discard_udp
start_service quid_tcp
start_service quid_udp
start_service chargen_tcp
start_service chargen_udp
start_service dummy_tcp
start_service dummy_udp

=====
# service_bind_address
#
# IP address to bind services to
#
# Syntax: service_bind_address <IP address>
#
# Default: 127.0.0.1
#
service_bind_address 192.168.50.100

=====
#
# Help
# Exit
# Write Out
# Read File
# Where Is
# Replace
# Cut
# Paste
# Execute
# Justify
# Location
# Go To Line
# Undo
# Redo
```



```
Clone di kali linux3 (in esecuzione) - Oracle VM VirtualBox
File Macchina Visualizza Inserimento Dispositivi Aiuto
Session Actions Edit View Help
GNU nano 2.9.3 /etc/inetsim/inetsim.conf
#
# Default hostname to return with DNS replies
#
# Syntax: dns_default_hostname <hostname>
#
# Default: www
#
#dns_default_hostname somehost

=====
#
# Default domain name to return with DNS replies
#
# Syntax: dns_default_domainname <domain name>
#
# Default: inetsim.org
#
#dns_default_domainname some.domain

=====
#
# Static mappings for DNS
#
# Syntax: dns_static <qdn hostname> <IP address>
#
# Default: none
#
dns_static episode.internal 192.168.50.100
dns_static ns1.foo.com 10.70.50.30
dns_static ftp.bar.net 10.10.20.30

=====
#
# dns_version
#
# DNS version
#
# Syntax: dns_version <version>
#
# Default: "INetSim DNS Server"
#
#dns_version "9.2.4"
```

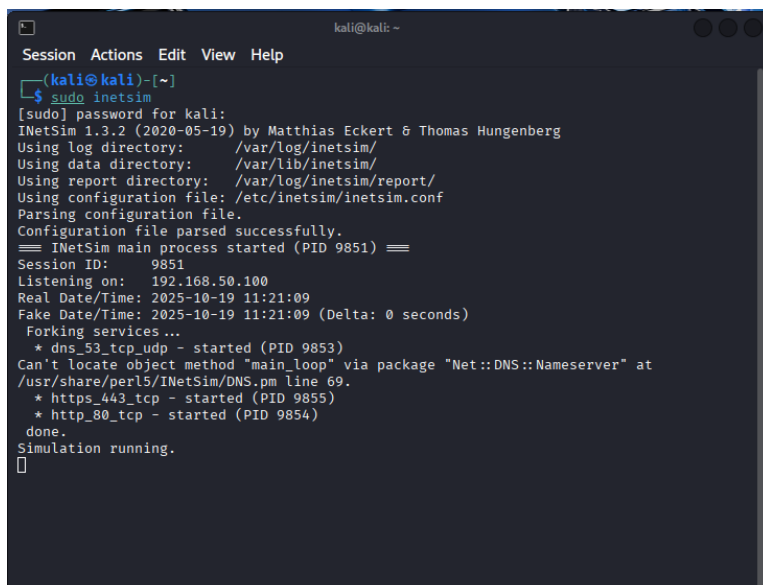
Una volta configurato INetsim possiamo

provare a farlo partire tramite in comando ***"sudo inetsim"*** e da qui notiamo subito una cosa il servizio parte tranquillamente ma il servizio dns sembrerebbe non rispondere correttamente, dando un **errore Can't locate object method "main_loop" via package "Net::DNS::Nameserver"** In sintesi, da quel che ho potuto leggere su alcuni forum su Reddit questo errore è dovuto a una incompatibilità del modulo **Perl** usato per il DNS da INetSim su Kali.

Quindi la versione di Kali utilizzate per il nostro esperimento non è compatibile con questa versione di perl, in questo caso si può procedere in due modi per risolvere il problema quella più semplice che prevede di installare una versione di Kali precedente esempio 2024, o quella più impegnativa ovvero procedere a un downgrade di perl.

Noi utilizzeremo quest'ultimo la soluzione più impegnativa.

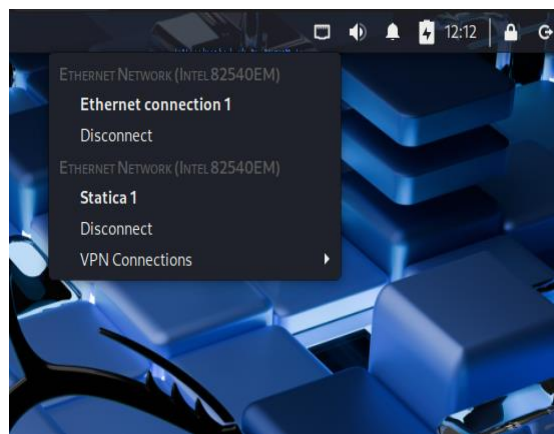
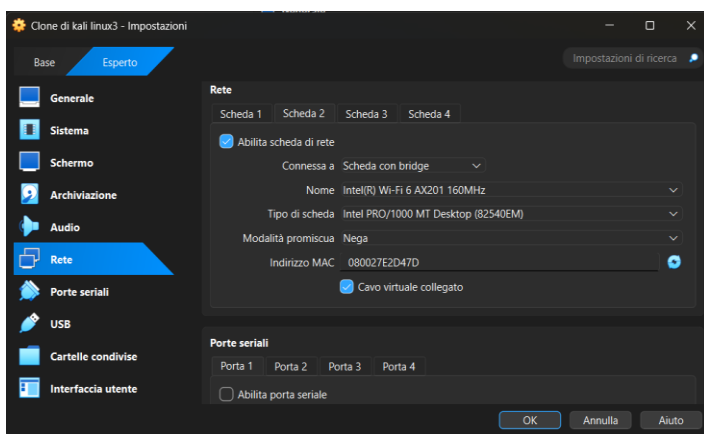
Foto dell'errore INetsim:



```
Session Actions Edit View Help
kali@kali: ~
$ sudo inetsim
[sudo] password for kali:
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory: /var/log/inetsim/
Using data directory: /var/lib/inetsim/
Using report directory: /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 9851) ==
Session ID: 9851
Listening on: 192.168.50.100
Real Date/Time: 2025-10-19 11:21:09
Fake Date/Time: 2025-10-19 11:21:09 (Delta: 0 seconds)
Forking services ...
* dns_53_tcp_udp - started (PID 9853)
Can't locate object method "main_loop" via package "Net::DNS::Nameserver" at
/usr/share/perl5/INetSim/DNS.pm line 69.
* https_443_tcp - started (PID 9855)
* http_80_tcp - started (PID 9854)
done.
Simulation running.
```

-Risoluzione problema ("Net::DNS::Nameserver"):

Come già citato nel paragrafo precedenti proveremo risolvere questo errore dovuto alla versione attuale di Perl, per prima cosa per poter fare una in downgrade alla versione precedente è necessario connettere la macchina alla rete esterna, quindi andiamo nelle impostazioni di rete di VM e apriamo la rete (2) e la impostiamo su bridge, questo ci serve per poter dare la possibilità alla macchina di connettersi a Internet, una volta fatto questo avviamo nuovamente la macchina kali, selezioniamo la rete 2 come rete preferita.



Da questo momento essendo connessi alla rete possiamo procedere con il downgrade di Perl. Apriamo il terminale ed eseguiamo in ordine prima il comando per rimuovere la versione di perl attuale ***“sudo apt remove libnet-dns-perl”*** (Foto 1)

poi si procede il comando

“ wget <https://cpan.metacpan.org/authors/id/N/NL/NLNETLABS/Net-DNS-1.37.tar.gz>”

per scaricare la versione interessata ovvero la 1.37 (Foto 2) a seguire estraiamo l'archivio scaricato con il comando ***“tar -xvzf Net-DNS-1.37.tar.gz”*** e ***“cd Net-DNS-1.37”*** (Foto 3) una volta dentro nella directory installiamo con il comando ***“perl Makefile.PL”*** (Foto 4), dopo di che si compila il modulo e si installa eseguendo prima il comando ***“make”*** e in seguito ***“sudo make install”*** (Foto 5 e 6) a questo punto abbiamo finito.

Se desideriamo verificare la versione installata possiamo inserire il comando

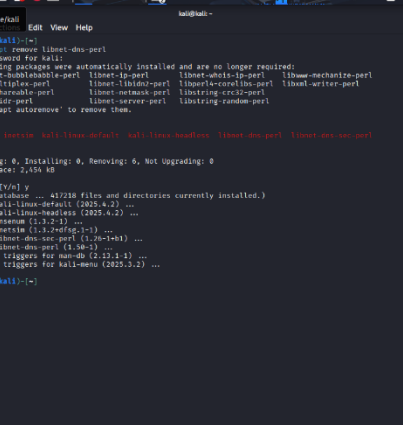
“perl -MNet::DNS -e 'print \$Net::DNS::VERSION, "\n";'” e vediamo che la 1.37 quella che serve a noi per far funzionare il tutto (Foto 7).

Adesso avendo eliminato il libnet-dns-perl, dobbiamo reinstallare INetsim con i comandi

“sudo apt update” e ***“sudo apt install Inetsim”***

Una volta eseguiti tutti questi passaggi il nostro servizio DNS su INetsim funzionerà in modo corretto

-Foto dell'operazione di installazione e funzionamento di INetsim-



```

kali@kali: ~
File  Machine  Visualization  Insertions  Dispositives  Auto

[Icons] [1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17] [18] [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] [30] [31] [32] [33] [34] [35] [36] [37] [38] [39] [40] [41] [42] [43] [44] [45] [46] [47] [48] [49] [50] [51] [52] [53] [54] [55] [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66] [67] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80] [81] [82] [83] [84] [85] [86] [87] [88] [89] [90] [91] [92] [93] [94] [95] [96] [97] [98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108] [109] [110] [111] [112] [113] [114] [115] [116] [117] [118] [119] [120] [121] [122] [123] [124] [125] [126] [127] [128] [129] [130] [131] [132] [133] [134] [135] [136] [137] [138] [139] [140] [141] [142] [143] [144] [145] [146] [147] [148] [149] [150] [151] [152] [153] [154] [155] [156] [157] [158] [159] [160] [161] [162] [163] [164] [165] [166] [167] [168] [169] [170] [171] [172] [173] [174] [175] [176] [177] [178] [179] [180] [181] [182] [183] [184] [185] [186] [187] [188] [189] [190] [191] [192] [193] [194] [195] [196] [197] [198] [199] [200] [201] [202] [203] [204] [205] [206] [207] [208] [209] [210] [211] [212] [213] [214] [215] [216] [217] [218] [219] [220] [221] [222] [223] [224] [225] [226] [227] [228] [229] [230] [231] [232] [233] [234] [235] [236] [237] [238] [239] [240] [241] [242] [243] [244] [245] [246] [247] [248] [249] [250] [251] [252] [253] [254] [255] [256] [257] [258] [259] [260] [261] [262] [263] [264] [265] [266] [267] [268] [269] [270] [271] [272] [273] [274] [275] [276] [277] [278] [279] [280] [281] [282] [283] [284] [285] [286] [287] [288] [289] [290] [291] [292] [293] [294] [295] [296] [297] [298] [299] [300] [301] [302] [303] [304] [305] [306] [307] [308] [309] [310] [311] [312] [313] [314] [315] [316] [317] [318] [319] [320] [321] [322] [323] [324] [325] [326] [327] [328] [329] [330] [331] [332] [333] [334] [335] [336] [337] [338] [339] [340] [341] [342] [343] [344] [345] [346] [347] [348] [349] [350] [351] [352] [353] [354] [355] [356] [357] [358] [359] [360] [361] [362] [363] [364] [365] [366] [367] [368] [369] [370] [371] [372] [373] [374] [375] [376] [377] [378] [379] [380] [381] [382] [383] [384] [385] [386] [387] [388] [389] [390] [391] [392] [393] [394] [395] [396] [397] [398] [399] [400] [401] [402] [403] [404] [405] [406] [407] [408] [409] [410] [411] [412] [413] [414] [415] [416] [417] [418] [419] [420] [421] [422] [423] [424] [425] [426] [427] [428] [429] [430] [431] [432] [433] [434] [435] [436] [437] [438] [439] [440] [441] [442] [443] [444] [445] [446] [447] [448] [449] [450] [451] [452] [453] [454] [455] [456] [457] [458] [459] [460] [461] [462] [463] [464] [465] [466] [467] [468] [469] [470] [471] [472] [473] [474] [475] [476] [477] [478] [479] [480] [481] [482] [483] [484] [485] [486] [487] [488] [489] [490] [491] [492] [493] [494] [495] [496] [497] [498] [499] [500] [501] [502] [503] [504] [505] [506] [507] [508] [509] [510] [511] [512] [513] [514] [515] [516] [517] [518] [519] [520] [521] [522] [523] [524] [525] [526] [527] [528] [529] [530] [531] [532] [533] [534] [535] [536] [537] [538] [539] [540] [541] [542] [543] [544] [545] [546] [547] [548] [549] [550] [551] [552] [553] [554] [555] [556] [557] [558] [559] [560] [561] [562] [563] [564] [565] [566] [567] [568] [569] [570] [571] [572] [573] [574] [575] [576] [577] [578] [579] [580] [581] [582] [583] [584] [585] [586] [587] [588] [589] [590] [591] [592] [593] [594] [595] [596] [597] [598] [599] [600] [601] [602] [603] [604] [605] [606] [607] [608] [609] [610] [611] [612] [613] [614] [615] [616] [617] [618] [619] [620] [621] [622] [623] [624] [625] [626] [627] [628] [629] [630] [631] [632] [633] [634] [635] [636] [637] [638] [639] [640] [641] [642] [643] [644] [645] [646] [647] [648] [649] [650] [651] [652] [653] [654] [655] [656] [657] [658] [659] [660] [661] [662] [663] [664] [665] [666] [667] [668] [669] [670] [671] [672] [673] [674] [675] [676] [677] [678] [679] [680] [681] [682] [683] [684] [685] [686] [687] [688] [689] [690] [691] [692] [693] [694] [695] [696] [697] [698] [699] [700] [701] [702] [703] [704] [705] [706] [707] [708] [709] [710] [711] [712] [713] [714] [715] [716] [717] [718] [719] [720] [721] [722] [723] [724] [725] [726] [727] [728] [729] [730] [731] [732] [733] [734] [735] [736] [737] [738] [739] [740] [741] [742] [743] [744] [745] [746] [747] [748] [749] [750] [751] [752] [753] [754] [755] [756] [757] [758] [759] [760] [761] [762] [763] [764] [765] [766] [767] [768] [769] [770] [771] [772] [773] [774] [775] [776] [777] [778] [779] [780] [781] [782] [783] [784] [785] [786] [787] [788] [789] [790] [791] [792] [793] [794] [795] [796] [797] [798] [799] [800] [801] [802] [803] [804] [805] [806] [807] [808] [809] [810] [811] [812] [813] [814] [
```

[illegible][illegible]

```
(kali㉿kali)-[~]
└─$ cd Net-DNS-1.37

(kali㉿kali)-[~/Net-DNS-1.37]
└─$ perl Makefile.PL

(kali㉿kali)-[~]
└─$ cd Net-DNS-1.37

(kali㉿kali)-[~/Net-DNS-1.37]
└─$ perl Makefile.PL
Duplicate specification "noonline-tests" for option "noonline-tests"
Activating Non Fatal Online Tests ...

Activating IPv6 Tests ...

Warning!

Online tests depend on conditions beyond the control of Net::DNS. The tests
check for the expected results when both Net::DNS and the outside world are
functioning properly. In case of failure it is often undecidable if the error
lies within Net::DNS or elsewhere.

Checking if your kit is complete ...
Looks good
Generating a Unix-style Makefile
Writing Makefile for Net::DNS
Writing MYMETA.yml and MYMETA.json

(kali㉿kali)-[~/Net-DNS-1.37]
└─$
```

[illegible]

```

Appending installation info to /usr/local/lib/x86_64-linux-gnu/perl/5.40.1/perllocal.pod
(kali@kali) [~/Net-DNS-1.37]
$ perl -MNet::DNS -e 'print $Net::DNS::VERSION, "\n";'
1.37

```

FOTO INetsim installazione:

```
kali@kali ~  
$ sudo apt update  
[sudo] password for kali:  
Get:1 http://kali.mirror.garr.it/kali kali-rolling InRelease [34.0 kB]  
Get:2 http://kali.mirror.garr.it/kali kali-rolling/main amd64 Packages [20.9 MB]  
Get:3 http://kali.mirror.garr.it/kali kali-rolling/main amd64 Contents (deb) [51.8 MB]  
Get:4 http://kali.mirror.garr.it/kali kali-rolling/contrib amd64 Packages [115 kB]  
Get:5 http://kali.mirror.garr.it/kali kali-rolling/contrib amd64 Contents (deb) [258 kB]  
Get:6 http://kali.mirror.garr.it/kali kali-rolling/non-free amd64 Packages [187 kB]  
Get:7 http://kali.mirror.garr.it/kali kali-rolling/non-free amd64 Contents (deb) [591 kB]  
Get:8 http://kali.mirror.garr.it/kali kali-rolling/non-free-firmware amd64 Packages [11.3 kB]  
Get:9 http://kali.mirror.garr.it/kali kali-rolling/non-free-firmware amd64 Contents (deb) [28.4 kB]  
Fetched 74.2 MB in 17s (4,367 kB/s)  
813 packages can be upgraded. Run 'apt list --upgradable' to see them.  
  
$ sudo apt install inetsim  
The following packages were automatically installed and are no longer required:  
  libnet-ip-perl libnet-netmask-perl libnet-whois-ip-perl libstring-random-perl libwww-mechanize-perl libxml-writer-perl  
Use 'sudo apt autoremove' to remove them.  
  
Installing:  
  inetsim  
  
Installing dependencies:  
  libnet-dns-perl libnet-dns-sec-perl  
  
Recommended packages:  
  libnfsqueue-perl  
  
Summary:  
  Upgrading: 0, Installing: 3, Removing: 0, Not Upgrading: 813  
  Download size: 663 kB  
  Space needed: 2,332 kB / 63.5 GB available  
  
Continue? [Y/n] y  
Get:1 http://kali.download/kali kali-rolling/main amd64 libnet-dns-perl all 1.53-1 [368 kB]  
Get:2 http://http.kali.org/kali kali-rolling/main amd64 inetsim all 1.3.2+dfsg-1-1 [247 kB]  
Get:3 http://http.kali.org/kali kali-rolling/main amd64 libnet-dns-sec-perl amd64 1.26-1+b1 [48.2 kB]  
Fetched 663 kB in 1s (650 kB/s)  
Selecting previously unselected package libnet-dns-perl.  
(Reading database ... 416867 files and directories currently installed.)  
Preparing to unpack .../libnet-dns-perl_1.53-1_all.deb ...  
Unpacking libnet-dns-perl (1.53-1) ...  
Selecting previously unselected package inetsim.  
Preparing to unpack .../inetsim_1.3.2+dfsg-1-1_all.deb ...  
Unpacking inetsim (1.3.2+dfsg-1-1) ...  
Selecting previously unselected package libnet-dns-sec-perl.  
Preparing to unpack .../libnet-dns-sec-perl_1.26-1+b1_amd64.deb ...  
Unpacking libnet-dns-sec-perl (1.26-1+b1) ...  
Setting up libnet-dns-perl (1.53-1) ...
```

Funzionamento INetsim:

```
Applications  
Session Actions Edit View Help  
  
(kali@kali)-[~]  
$ sudo inetsim  
[sudo] password for kali:  
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg  
Using log directory: /var/log/inetsim/  
Using data directory: /var/lib/inetsim/  
Using report directory: /var/log/inetsim/report/  
Using configuration file: /etc/inetsim/inetsim.conf  
Parsing configuration file.  
Configuration file parsed successfully.  
== INetSim main process started (PID 7648) ==  
Session ID: 7648  
Listening on: 192.168.50.100  
Real Date/Time: 2025-10-19 13:22:58  
Fake Date/Time: 2025-10-19 13:22:58 (Delta: 0 seconds)  
Forking services ...  
  * dns_53_tcp_udp - started (PID 7650)  
  * https_443_tcp - started (PID 7652)  
  * http_80_tcp - started (PID 7651)  
done.  
Simulation running.
```

In conclusione, dopo aver reinstallato INetSim e risolto le difficoltà incontrate, ora il sistema funziona regolarmente e consente di proseguire il lavoro come previsto.

Adesso possiamo sniffare i pacchetti utilizzando Wireshark provando a collegare la macchina Windows da Kali tramite il servizio http,https e dns di INetsim

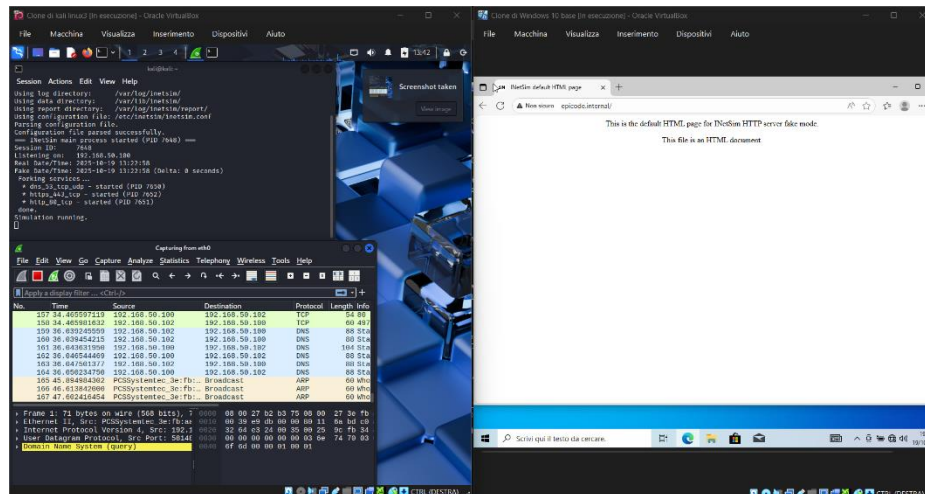
-WIRESHARK E INetsim:

Adesso proviamo a far comunicare le macchine utilizzando il servizio INetsim e cerchiamo di vedere tutti i pacchetti che transitano attraverso la rete.

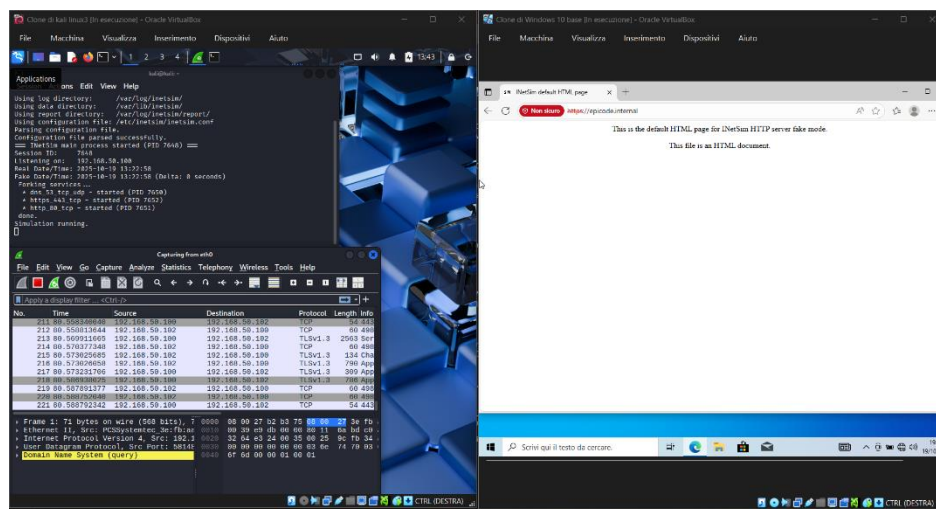
Settiamo nuovamente le macchine su rete interna e avviamo kali e windows, una a volta avviate le macchine dalle impostazioni di rete di windows impostiamo il DNS su 192.168.50.100 , da kali invece avviamo il servizio di INetsim e Wireshark.

Una volta avviati, da Windows apriamo il browser e proviamo a collegarci su <http://epicode.internal/> e <https://epicode.internal> con in esecuzione anche wireshark in modo tale da monitorare i pacchetti in transito.

In http:



In https:



Adesso rechiamoci su Wireshark e verifichiamo quali pacchetti hanno transitato mentre stavamo provando a collegarci al nostro servizio

Allego foto dei pacchetti catturati:

Cattura indirizzo MAC:

261	1317.8578891...	192.168.50.102	192.168.50.100	TCP	66 49933 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
262	1317.8579221...	192.168.50.100	192.168.50.102	TCP	66 80 → 49933 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
263	1317.8592815...	192.168.50.102	192.168.50.100	TCP	66 49933 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
264	1317.8825053...	192.168.50.102	192.168.50.100	DNS	76 Standard query 0xea4b A epicode.internal
265	1317.8840343...	192.168.50.102	192.168.50.100	HTTP	532 GET / HTTP/1.1
266	1317.8840841...	192.168.50.100	192.168.50.102	TCP	54 80 → 49933 [ACK] Seq=1 Ack=479 Win=64128 Len=0
267	1317.8846683...	192.168.50.102	192.168.50.100	TCP	66 49934 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM
268	1317.8847034...	192.168.50.100	192.168.50.102	TCP	66 80 → 49934 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
269	1317.8856316...	192.168.50.102	192.168.50.100	TCP	60 49934 → 80 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
270	1317.8972964...	192.168.50.100	192.168.50.102	TCP	204 80 → 49933 [PSH, ACK] Seq=1 Ack=479 Win=64128 Len=150 [TCP PDU reassembled in 271]
271	1317.8998075...	192.168.50.100	192.168.50.102	HTTP	312 HTTP/1.1 200 OK (text/html)
272	1317.9016884...	192.168.50.102	192.168.50.100	TCP	60 49933 → 80 [ACK] Seq=479 Ack=410 Win=262144 Len=0
273	1317.9016887...	192.168.50.102	192.168.50.100	TCP	60 49933 → 80 [FIN, ACK] Seq=479 Ack=410 Win=262144 Len=0
274	1317.9017315...	192.168.50.100	192.168.50.102	TCP	54 80 → 49933 [ACK] Seq=410 Ack=480 Win=64128 Len=0
275	1317.9037868...	192.168.50.100	192.168.50.102	DNS	92 Standard query response 0xea4b A epicode.internal A 192.168.50.100

La cattura mostra inizialmente la risoluzione DNS del nome "epicode.internal" che viene correttamente associato all'indirizzo IP 192.168.50.100. Successivamente, si sviluppa una comunicazione HTTP tra il client (192.168.50.102) e il server (192.168.50.100), con la classica sequenza di handshake TCP, richiesta GET da parte del client e risposta 200 OK dal server. Tutte le sessioni si concludono con la regolare chiusura delle connessioni TCP.

Qui possiamo vedere la traccia nel dettaglio: (Riga 264 a 275)

Applications

Wireshark - Packet 264 - eth0

- Frame 264: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface eth0, id 0
- Ethernet II, Src: PCSSystemtec_3e:fb:aa (08:00:27:3e:fb:aa), Dst: PCSSystemtec_b2:b3:75 (08:00:27:b2:b3:75)
- Destination: PCSSystemtec_b2:b3:75 (08:00:27:b2:b3:75)
- Source: PCSSystemtec_3e:fb:aa (08:00:27:3e:fb:aa)
- Type: IPv4 (0x0800)
- [Stream index: 0]
- Internet Protocol Version 4, Src: 192.168.50.102, Dst: 192.168.50.100
 - 0100 ... = Version: 4
 - ... 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 62
 - Identification: 0xea5f (59999)
 - 0000 ... = Flags: 0x0
 - ... 0000 0000 0000 = Fragment Offset: 0
 - Time to live: 32
 - Protocol: UDP (17)
 - Header Checksum: 0x6a34 [validation disabled]
 - [Header checksum status: Unverified]
 - Source Address: 192.168.50.102
 - Destination Address: 192.168.50.100
 - [Stream index: 0]
- User Datagram Protocol, Src Port: 54832, Dst Port: 53
 - Source Port: 54832
 - Destination Port: 53
 - Length: 42
 - Checksum: 0x6175 [unverified]
 - [Checksum Status: Unverified]
 - [Stream index: 32]
 - [Stream Packet Number: 1]
 - [Timestamps]
 - UDP payload (34 bytes)
 - Domain Name System (query)
 - Transaction ID: 0xea4b
 - Flags: 0x0100 Standard query
 - Questions: 1
 - Answer RRs: 0
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - [Response ID: 275]

Wireshark - Packet 275 - eth0

- Frame 275: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface eth0, id 0
- Ethernet II, Src: PCSSystemtec_b2:b3:75 (08:00:27:b2:b3:75), Dst: PCSSystemtec_3e:fb:aa (08:00:27:3e:fb:aa)
- Destination: PCSSystemtec_3e:fb:aa (08:00:27:3e:fb:aa)
- Source: PCSSystemtec_b2:b3:75 (08:00:27:b2:b3:75)
- Type: IPv4 (0x0800)
- [Stream index: 0]
- Internet Protocol Version 4, Src: 192.168.50.100, Dst: 192.168.50.102
 - 0100 ... = Version: 4
 - ... 0101 = Header Length: 20 bytes (5)
 - Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
 - Total Length: 78
 - Identification: 0x06d1 (1745)
 - 010 ... = Flags: 0x2, Don't fragment
 - ... 0000 0000 0000 = Fragment Offset: 0
 - Time to live: 64
 - Protocol: UDP (17)
 - Header Checksum: 0x4db3 [validation disabled]
 - [Header checksum status: Unverified]
 - Source Address: 192.168.50.100
 - Destination Address: 192.168.50.102
 - [Stream index: 0]
- User Datagram Protocol, Src Port: 53, Dst Port: 54832
 - Source Port: 53
 - Destination Port: 54832
 - Length: 58
 - Checksum: 0xe066 [unverified]
 - [Checksum Status: Unverified]
 - [Stream index: 32]
 - [Stream Packet Number: 2]
 - [Timestamps]
 - UDP payload (50 bytes)
 - Domain Name System (response)
 - Transaction ID: 0xea4b
 - Flags: 0x8500 Standard query response, No error
 - Questions: 1
 - Answer RRs: 1
 - Authority RRs: 0
 - Additional RRs: 0
 - Queries
 - Answers

No. 264: Time: 1317.882505389 Source: 192.168.50.102 Destination: 192.168.50.100 Protocol: DNS Length: 76 Info: Standard query 0xea4b A epicode.internal

✓ Show packet bytes Layout: Vertical (Stacked)

Close Help

No. 275: Time: 1317.903786861 Source: 192.168.50.100 Destination: 192.168.50.102 Info: Standard query response 0xea4b A epicode.internal A 192.168.50.100

✓ Show packet bytes Layout: Vertical (Stacked)

Close Help

Cattura HTTPS:

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

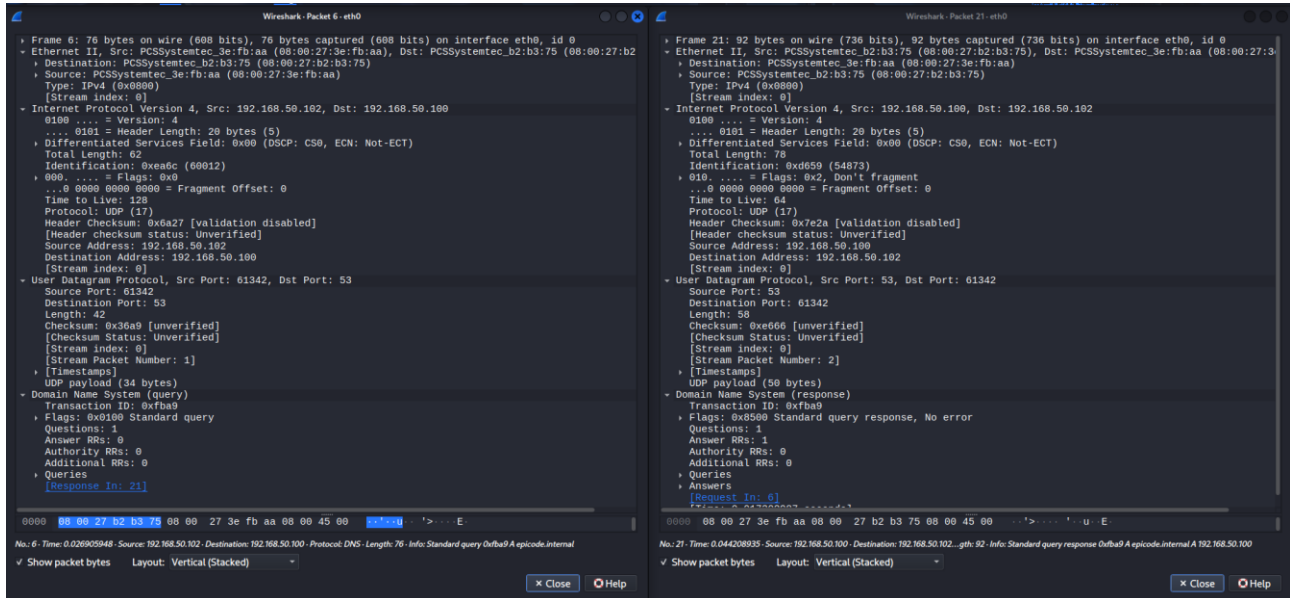
Apply a display filter ... <Ctrl>/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.50.102	192.168.50.100	TCP	66 49983 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
2	0.000033165	192.168.50.100	192.168.50.102	TCP	66 443 → 49988 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128	
3	0.001272583	192.168.50.102	192.168.50.100	TCP	60 49988 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0	
4	0.003547768	192.168.50.102	192.168.50.100	TLV1.3	2053 Client Hello (SNI=epicode.internal)	
5	0.003566700	192.168.50.100	192.168.50.102	TCP	54 443 → 49988 [ACK] Seq=1 Ack=1998 Win=62336 Len=0	
6	0.026905948	192.168.50.102	192.168.50.100	DNS	76 Standard query 0xfba9 A epicode.internal	
7	0.032322961	192.168.50.100	192.168.50.102	TLV1.3	2563 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data	
8	0.033215530	192.168.50.102	192.168.50.100	TCP	60 49988 → 443 [ACK] Seq=1998 Ack=2510 Win=2102272 Len=0	
9	0.034231902	192.168.50.102	192.168.50.100	TLV1.3	84 Change Cipher Spec, Application Data	
10	0.034232095	192.168.50.102	192.168.50.100	TCP	60 49988 → 443 [FIN, ACK] Seq=2028 Ack=2510 Win=2102272 Len=0	
11	0.037646301	192.168.50.102	192.168.50.100	TCP	60 49989 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
12	0.037679110	192.168.50.100	192.168.50.102	TCP	66 443 → 49989 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128	
13	0.038493877	192.168.50.102	192.168.50.100	TCP	60 49989 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0	
14	0.041335224	192.168.50.102	192.168.50.100	TCP	66 49990 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
15	0.041354461	192.168.50.100	192.168.50.102	TCP	66 443 → 49990 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128	
16	0.041957956	192.168.50.102	192.168.50.100	TCP	60 49990 → 443 [ACK] Seq=1 Ack=1 Win=262656 Len=0	
17	0.042414164	192.168.50.102	192.168.50.100	TLV1.3	2083 Client Hello (SNI=epicode.internal)	
18	0.042423456	192.168.50.100	192.168.50.102	TCP	54 443 → 49989 [ACK] Seq=1 Ack=2030 Win=62592 Len=0	
19	0.043129493	192.168.50.102	192.168.50.100	TLV1.3	1780 Client Hello (SNI=epicode.internal)	
20	0.043136431	192.168.50.100	192.168.50.102	TCP	54 443 → 49990 [ACK] Seq=1 Ack=1727 Win=62592 Len=0	
21	0.044200935	192.168.50.102	192.168.50.100	TCP	92 Standard query response 0xfba9 A epicode.internal A 192.168.50.100	
22	0.047176688	192.168.50.100	192.168.50.102	TCP	54 443 → 49988 [FIN, ACK] Seq=2510 Ack=2029 Win=63488 Len=0	
23	0.047919120	192.168.50.102	192.168.50.100	TCP	60 49988 → 443 [ACK] Seq=2029 Ack=2511 Win=2102272 Len=0	
24	0.060882075	192.168.50.100	192.168.50.102	TLV1.3	2563 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data	
25	0.060100935	192.168.50.102	192.168.50.100	TCP	60 49989 → 443 [ACK] Seq=2030 Ack=2510 Win=262656 Len=0	
26	0.070145853	192.168.50.102	192.168.50.100	TLV1.3	84 Change Cipher Spec, Application Data	
27	0.070772155	192.168.50.102	192.168.50.100	TCP	60 49989 → 443 [FIN, ACK] Seq=2060 Ack=2510 Win=262656 Len=0	
28	0.074996864	192.168.50.100	192.168.50.102	TCP	54 443 → 49989 [FIN, ACK] Seq=2510 Ack=2061 Win=642256 Len=0	
29	0.074859783	192.168.50.102	192.168.50.100	TLV1.3	2563 Server Hello, Change Cipher Spec, Application Data, Application Data, Application Data	
30	0.074925457	192.168.50.102	192.168.50.100	TCP	60 49989 → 443 [ACK] Seq=2061 Ack=2511 Win=262656 Len=0	
31	0.075421990	192.168.50.102	192.168.50.100	TCP	60 49990 → 443 [ACK] Seq=1727 Ack=2510 Win=262656 Len=0	
32	0.075494926	192.168.50.102	192.168.50.100	TLV1.3	84 Change Cipher Spec, Application Data	
33	0.076110515	192.168.50.102	192.168.50.100	TCP	60 49989 → 443 [FIN, ACK] Seq=1757 Ack=2510 Win=262656 Len=0	
34	0.077563217	192.168.50.102	192.168.50.100	TCP	66 49992 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM	
35	0.077581281	192.168.50.100	192.168.50.102	TCP	66 443 → 49992 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128	
36	0.078279307	192.168.50.102	192.168.50.100	TCP	60 49992 → 443 [ACK] Seq=1 Ack=1 Win=2102272 Len=0	
37	0.078754945	192.168.50.102	192.168.50.100	TLV1.3	1812 Client Hello (SNI=epicode.internal)	
38	0.078763319	192.168.50.100	192.168.50.102	TCP	54 443 → 49992 [ACK] Seq=1 Ack=1759 Win=62592 Len=0	
39	0.080100913	192.168.50.100	192.168.50.102	TCP	54 443 → 49990 [FIN, ACK] Seq=2510 Ack=1758 Win=62592 Len=0	
40	0.090257358	192.168.50.102	192.168.50.100	TCP	60 49990 → 443 [ACK] Seq=1758 Ack=2511 Win=262656 Len=0	

L'analisi del traffico di rete evidenzia la fase di connessione sicura HTTPS tra un client e un server della rete locale. In particolare, la schermata mostra l'iniziale handshake TCP

seguito dalla negoziazione TLSv1.3, tramite scambio di pacchetti "Client Hello" e "Server Hello". Durante la stessa sessione, il nome "epicode.internal" viene risolto tramite query DNS, ottenendo il corrispondente indirizzo IP 192.168.50.100. Successivamente avviene la trasmissione dei dati cifrati attraverso il canale HTTPS, che garantisce autenticità, integrità e confidenzialità delle informazioni trasmesse tra client e server.

Qui possiamo vedere la traccia nel dettaglio: (Riga 6 a 21)



-Conclusione:

In conclusione, dopo aver risolto le problematiche riscontrate nella configurazione di INetSim, il sistema adesso è perfettamente funzionante. L'analisi del traffico HTTP e HTTPS in ambiente virtuale ha mostrato le differenze significative tra i due protocolli in termini di sicurezza e visibilità dei dati. Utilizzando strumenti di monitoraggio come Wireshark, è stato possibile osservare in dettaglio il flusso dei pacchetti, e la corretta risoluzione DNS e le fasi di handshake, che ricordiamo essere il processo iniziale di negoziazione tra un client (come un browser) e un server che serve a stabilire una connessione sicura tramite TLS.

Spero che questa relazione vi piaccia e risulti utile per comprendere appieno l'analisi e le configurazioni che ho effettuato.

Data: 19/10/2025

Alunno: Alberto Sucato