

CSPT0524IT – M1W4D4

Cybersecurity Analyst Progetto Finale M1

Report di Svolgimento del Progetto Finale M1

Requisiti e servizi:

- Kali Linux IP: 192.168.32.100
- Windows IP: 192.168.32.101
- HTTPS server: attivo
- Servizio DNS: attivo

Traccia:

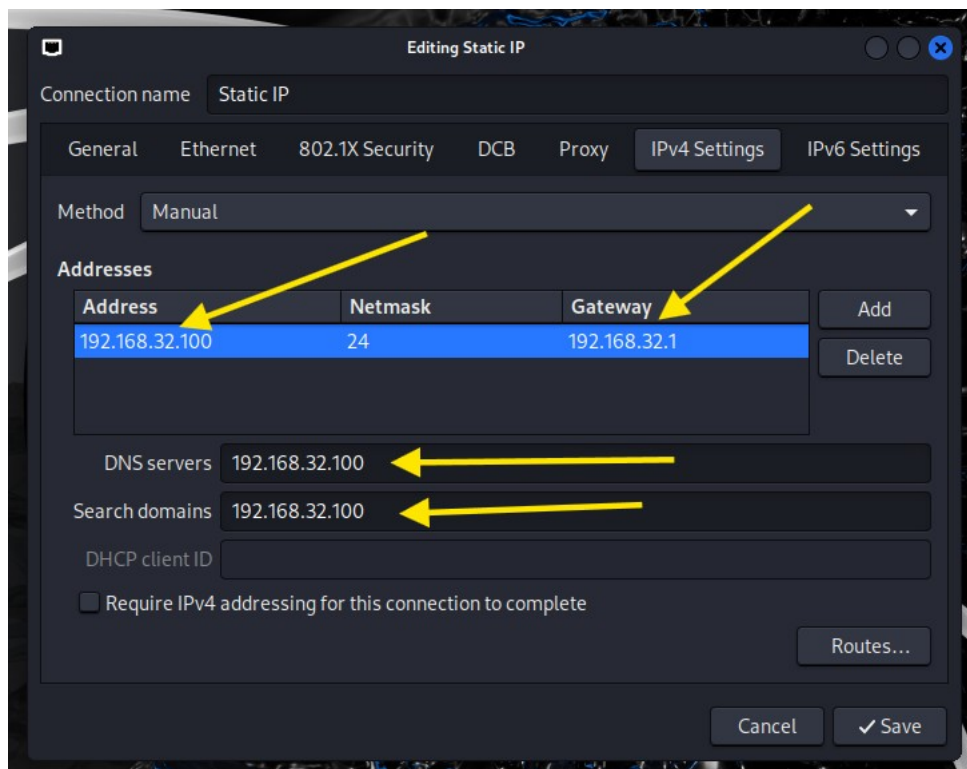
“Simulare, in ambiente di laboratorio virtuale, un’architettura client server in cui un client con indirizzo 192.168.32.101 (Windows) richiede tramite web browser una risorsa all’hostname epicode.internal che risponde all’indirizzo 192.168.32.100 (Kali). Si intercetti poi la comunicazione con Wireshark, evidenziando i MAC address di sorgente e destinazione ed il contenuto della richiesta HTTPS. Ripetere l’esercizio, sostituendo il server HTTPS, con un server HTTP. Si intercetti nuovamente il traffico, evidenziando le eventuali differenze tra il traffico appena catturato in HTTP ed il traffico precedente in HTTPS. Spiegare, motivandole, le principali differenze se presenti.”

- 1. Configurazione IP sulle macchine**
- 2. Creazione Regola Firewall e Test di comunicazione (PING)**
- 3. Avvio Server DNS su Kali**
- 4. Avvio Servizio HTTP/HTTPS su Kali**
- 5. Test HTTP/HTTPS su Browser**
- 6. Packet Sniffing con Wireshark**
- 7. Analisi dei Risultati e Osservazioni**

1. Configurazione IP sulle macchine

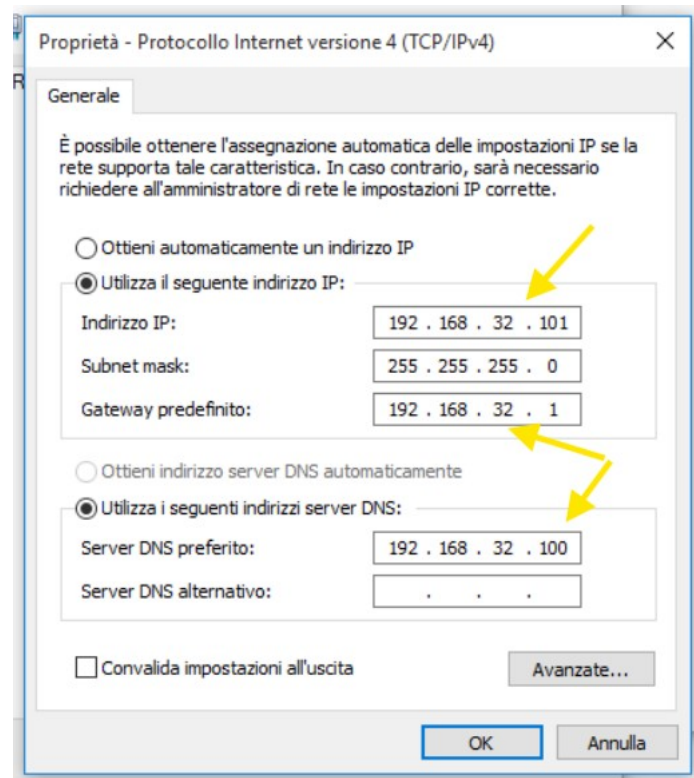
KALI-LINUX

- Ho assegnato l'IP address 192.168.32.100/24
- Gateway su 192.168.32.1 (Per una futura Installazione di PfSense).
- DNS Server e Search domains: 192.168.32.100 (perché la macchina kali farà l'host del server DNS).



WINDOWS 10

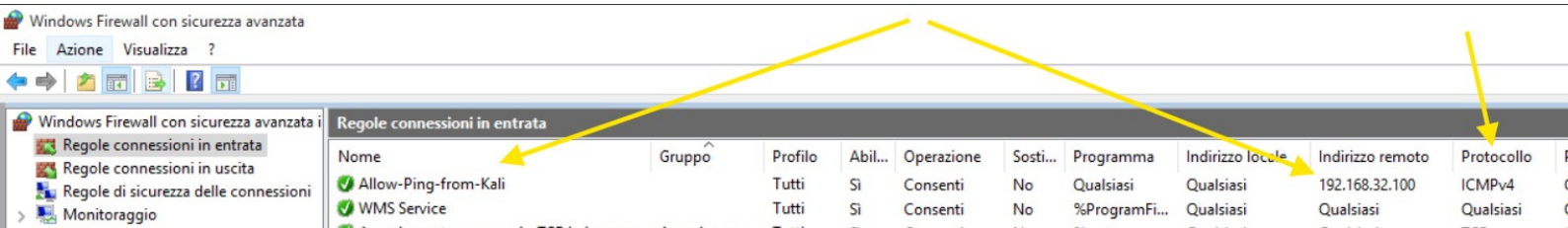
- Ho assegnato l'IP address 192.168.32.101/24
- Gateway su 192.168.32.1 (Per una futura Installazione di PfSense).
- DNS Server e Search domains: 192.168.32.100 (perché la macchina kali farà l'host del server DNS).



2. Creazione Regola Firewall e Test di comunicazione (PING)

Windows Firewall

- Su Windows 10 ho creato una regola si Firewall per ricevere pacchetti ICMP da kali.
Per questioni di sicurezza, ho preso l'iniziativa di aprire la connessione solo all'ip remoto di kali
come in figura



-Test di Comunicazione

- Sulla sinistra Kali che esegue con successo il Ping su IP di Windows10: 192.168.32.101
- Sulla destra Windows 10 che esegue con successo il Ping su IP di Kali: 192.168.32.100

```
(kali@kali)-[~]
$ ping 192.168.32.101
PING 192.168.32.101 (192.168.32.101) 56(84) bytes of data:
64 bytes from 192.168.32.101: icmp_seq=1 ttl=128 time=1.84 ms
64 bytes from 192.168.32.101: icmp_seq=2 ttl=128 time=1.49 ms
64 bytes from 192.168.32.101: icmp_seq=3 ttl=128 time=1.84 ms
64 bytes from 192.168.32.101: icmp_seq=4 ttl=128 time=1.22 ms
64 bytes from 192.168.32.101: icmp_seq=5 ttl=128 time=1.84 ms
64 bytes from 192.168.32.101: icmp_seq=6 ttl=128 time=1.73 ms
64 bytes from 192.168.32.101: icmp_seq=7 ttl=128 time=1.34 ms
64 bytes from 192.168.32.101: icmp_seq=8 ttl=128 time=1.72 ms
64 bytes from 192.168.32.101: icmp_seq=9 ttl=128 time=1.56 ms
64 bytes from 192.168.32.101: icmp_seq=10 ttl=128 time=1.54 ms
64 bytes from 192.168.32.101: icmp_seq=11 ttl=128 time=1.44 ms
64 bytes from 192.168.32.101: icmp_seq=12 ttl=128 time=1.56 ms
64 bytes from 192.168.32.101: icmp_seq=13 ttl=128 time=1.36 ms
64 bytes from 192.168.32.101: icmp_seq=14 ttl=128 time=1.32 ms
^C
— 192.168.32.101 ping statistics —
14 packets transmitted, 14 received, 0% packet loss, time 13025ms
rtt min/avg/max/mdev = 1.219/1.556/1.843/0.201 ms

C:\Windows\system32>ping 192.168.32.100

Esecuzione di Ping 192.168.32.100 con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata=2ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64


Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
    Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 1ms, Massimo = 2ms, Medio = 1ms

C:\Windows\system32>
```

3. Avvio Server DNS su Kali

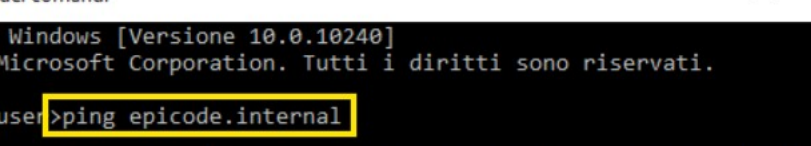
-Per avviare un servizio DNS su Kali mi sono servito del tool preinstallato **DNSChef** con il seguente comando:

```
dnschef --fakedomains epicode.internal --fakeip 192.168.32.100 --nameservers 192.168.32.100
--interface 192.168.32.100
```



```
kali@kali: ~  
File Actions Edit View Help  
  
(kali@kali)-[~]  
$ dnscchef --fakedomains epicode.internal --fakeip 192.168.32.100 --nameservers 192.168.32.100 --interface 192.168.32.100  
  
[version 0.4]  
[iphelix@thesprawl.org]  
  
(17:02:10) [*] DNSChef started on interface: 192.168.32.100  
(17:02:10) [*] Using the following nameservers: 192.168.32.100  
(17:02:10) [*] Cooking A replies to point to 192.168.32.100 matching: epicode.internal
```

- Mi sono assicurato di poter eseguire con successo un ping `epicode.internal` prima di proseguire



The screenshot shows a Windows Command Prompt window with the title "Prompt dei comandi". The text inside the window is as follows:

```
Microsoft Windows [Versione 10.0.10240]
(c) 2015 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\user>ping epicode.internal

Esecuzione di Ping epicode.internal [192.168.32.100] con 32 byte di dati:
Risposta da 192.168.32.100: byte=32 durata=2ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=3ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64
Risposta da 192.168.32.100: byte=32 durata=1ms TTL=64

Statistiche Ping per 192.168.32.100:
    Pacchetti: Trasmessi = 4, Ricevuti = 4,
    Persi = 0 (0% persi),
Tempo approssimativo percorsi andata/ritorno in millisecondi:
    Minimo = 1ms, Massimo = 3ms, Medio = 1ms

C:\Users\user>
```

4. Avvio Servizio HTTP/HTTPS su Kali

- Per avviare i servizi HTTP/HTTPS ho utilizzato inetsim come in lezione.

Nel file “**/etc/inetsim/inetsim.conf**” ho commentato tutti i servizi lasciando attivi solo i servizi HTTPS e HTTP e modificato il bind_address in “**service_bind_address 192.168.32.100**” (Kali IP), infine ho avviato inetsim con “**sudo inetsim**”

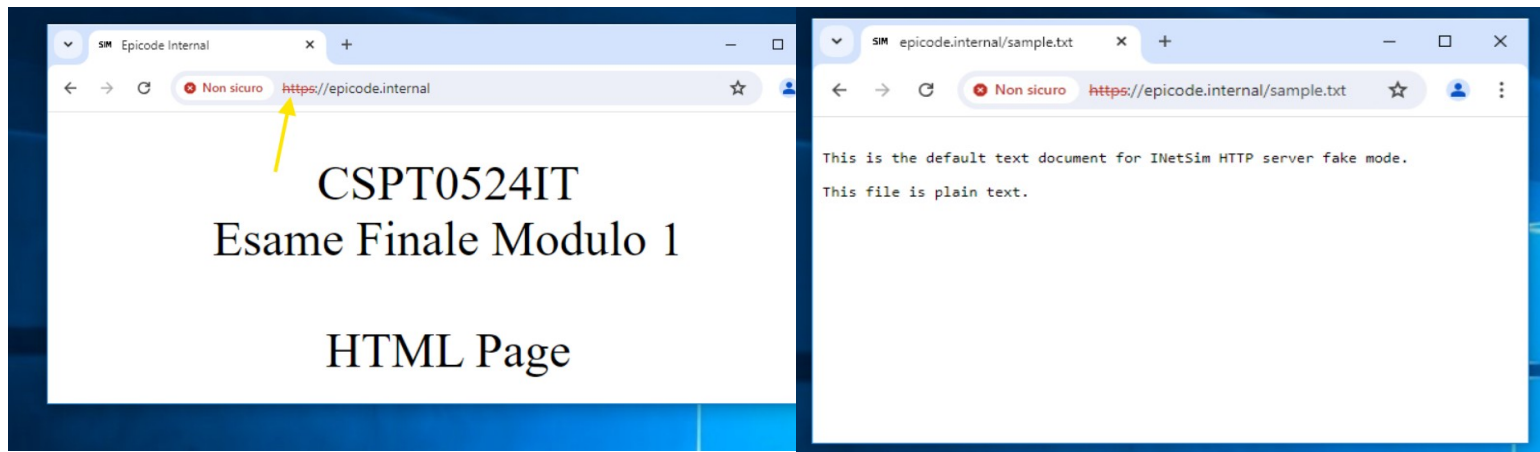
```
(kali㉿kali)-[/etc/inetsim]
$ sudo inetsim
INetSim 1.3.2 (2020-05-19) by Matthias Eckert & Thomas Hungenberg
Using log directory:      /var/log/inetsim/
Using data directory:     /var/lib/inetsim/
Using report directory:   /var/log/inetsim/report/
Using configuration file: /etc/inetsim/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
== INetSim main process started (PID 987271) ==
Session ID:      987271
Listening on:    192.168.32.100
Real Date/Time:  2024-11-29 17:34:39
Fake Date/Time:  2024-11-29 17:34:39 (Delta: 0 seconds)
Forking services ...
  * https_443_tcp - started (PID 987565)
done.
Simulation running.
```

5. Test HTTP/HTTPS su Browser

HTTPS

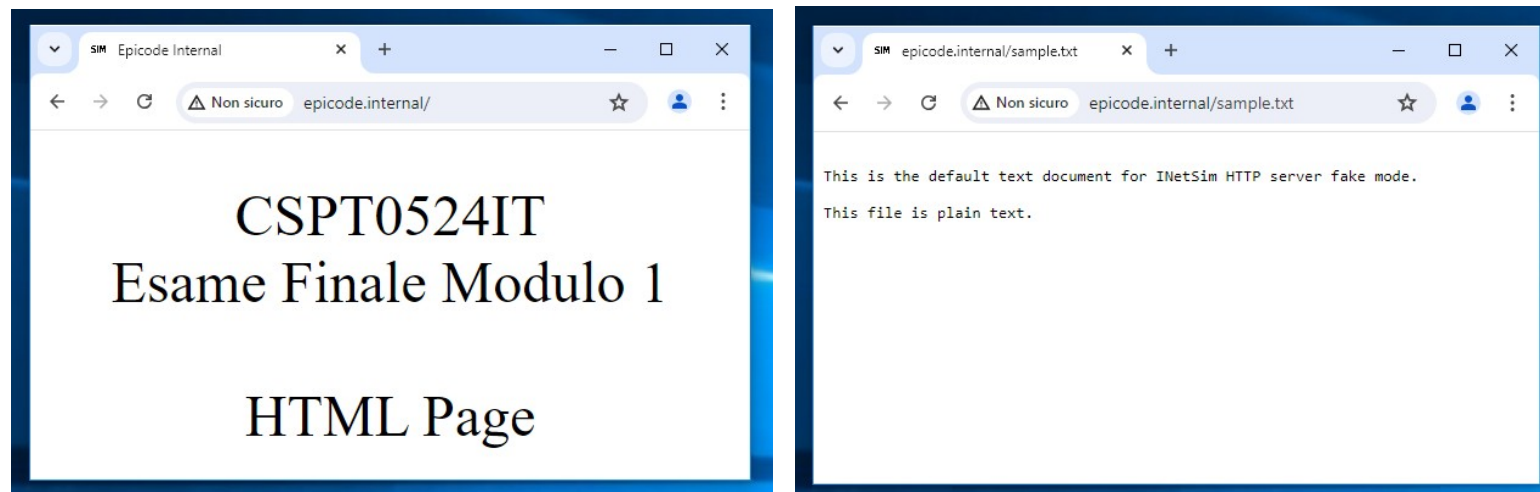
- Come si nota in figura il servizio HTTPS è attivo su porta 443.

Ho anche personalizzato la homepage dal file `/var/lib/inetsim/http/fakefiles/sample.html`



HTTP

- Si nota che non è presente il prefisso HTTPS perché la pagina è stata servita in HTTP (in chiaro).



6. Packet Sniffing con Wireshark

- Con Wireshark ho eseguito due scansioni:

- ◆ una scansione con filtro tcp.port == 443 per le connessioni HTTPS criptate
- ◆ una scansione con filtro tcp.port == 80 per le connessioni HTTP in chiaro

SCANSIONE HTTPS

Scansionando il servizio HTTPS pur essendo una connessione criptata, sono riuscito a ottenere diverse informazioni come:

- Nome e Tipo interfaccia utilizzata (Figura 1),

```
▼ Frame 56: 66 bytes on wire (528 bits)
  Section number: 1
  ▼ Interface id: 0 (eth0)
    Interface name: eth0
    Encapsulation type: Ethernet (1)
    Arrival Time: Nov 30, 2024 04:12:37
```

Figura 1: Nome-Tipo Interfaccia

- Data e Ora della richiesta HTTPS (Figura 2),

```
Arrival Time: Nov 30, 2024 04:12:37.320305272 EST
UTC Arrival Time: Nov 30, 2024 09:12:37.320305272 UTC
Epoch Arrival Time: 1732957957.320305272
```

Figura 2: Data-Ora Richiesta HTTPS

- Dominio tramite SNI (Figura 3),

```
[ACK] Seq=1 Len=1 Win=0
p (SNI=epicode.internal)
[ACK] Seq=1 Len=1 Win=0
```

Figura 3: Dominio

- Indirizzo Mac Sorgente e Mac di Destinazione (Figura 4),

```
▼ Ethernet II, Src: PCSSystemtec_e9:03:5e (08:00:27:e9:03:5e), Dst: PCSSystemtec_ad:25:87 (08:00:27:ad:25:87)
  ▼ Destination: PCSSystemtec_ad:25:87 (08:00:27:ad:25:87)
    Address: PCSSystemtec_ad:25:87 (08:00:27:ad:25:87)
    ....0. .... = LG bit: Globally unique address (factory default)
    ....0. .... = IG bit: Individual address (unicast)
  ▼ Source: PCSSystemtec_e9:03:5e (08:00:27:e9:03:5e)
    Address: PCSSystemtec_e9:03:5e (08:00:27:e9:03:5e)
    ....0. .... = LG bit: Globally unique address (factory default)
```

Figura 4: Indirizzi Mac

- Indirizzo IP del Server HTTPS e IP Client (Figura 5),

```
Source Address: 192.168.32.101
Destination Address: 192.168.32.100
```

Figura 5: Indirizzi IP

- Porte utilizzate da Server e Client (Figura 6),

```
Source Port: 49465
Destination Port: 443
```

Figura 6: Porte

SCANSIONE HTTP

Durante la scansione del servizio HTTP, oltre alle informazioni ottenute in precedenza in HTTPS, sono riuscito a ottenere anche l'header della richiesta GET con l'user-agent, l'URL della richiesta, il tipo di Sistema Operativo, il Browser utilizzato e inoltre ho ottenuto anche il contenuto del messaggio nella risposta 200OK da parte del server HTTP:

- Richiesta GET da parte del client Windows (Figura 7)

```
GET / HTTP/1.1\r\n
Host: epicode.internal\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/126.0.0.0 Safari/537.36\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7\r\n
Accept-Encoding: gzip, deflate\r\n
Accept-Language: it-IT,it;q=0.9,en-US;q=0.8,en;q=0.7\r\n
\r\n
[Full request URI: http://epicode.internal/]
[HTTP request 1/1]
[Response in frame: 79]
```

Figura 7: Richiesta GET

- Contenuto in chiaro del messaggio (Figura 8)

Il contenuto mostra perfettamente in chiaro il codice che ho modificato in precedenza nel file `/var/lib/inetsim/http/fakefiles/sample.html`

```
Line-based text data: text/html (9 lines)
<html>\n
<head>\n
  <title>Epicode Internal</title>\n
</head>\n
<body>\n
  <p></p><br>\n
  <p align="center"><font size=50>CSPT0524IT<br>Esame Finale Modulo 1<br><br>HTML Page</font></p>\n
</body>\n
</html>\n
```

Figura 8: Contenuto Messaggio in Chiaro

7. Analisi dei Risultati e Osservazioni

-Il protocollo HTTPS utilizza SSL o TLS che implementa la crittografia e va a criptare i pacchetti in transito, quindi non è possibile scoprire il contenuto ma è possibile ottenere informazioni come

- Dominio
- IP del client e del server
- Porte utilizzate dal client e dal server
- Indirizzi MAC del client e del server
- Data e ora della richiesta

Pur essendo un protocollo criptato, le informazioni ottenute sono sicuramente abbastanza per il lavoro che svolgeremo nei mesi che verranno.

-Il protocollo HTTP, a differenza di HTTPS, non utilizza SSL/TLS e di conseguenza i pacchetti in transito non vengono criptati ma viaggiano in chiaro; in effetti sono riuscito a trovare più informazioni rispetto alla scansione HTTPS, che oltre al Dominio, gli IP, le porte e i Mac, mi ha permesso di raccogliere più informazioni come

- Tipo di browser utilizzato
- Sistema operativo del client
- Tipo di browser utilizzato
- URL completo della richiesta
- Contenuto del messaggio trasmesso tra client e server

OSSERVAZIONE:

Sono riuscito a intercettare la richiesta GET del Client e la risposta 200OK del Server ottenendo le informazioni sopra citate, in un ambiente controllato, questo mi permetterebbe di intercettare anche pacchetti di richieste POST che esegue il Client e ottenere informazioni sensibili e private. È fondamentale, per garantire la sicurezza informatica, che i servizi utilizzino HTTPS con certificati validi per evitare l'intercettazione da parte di malintenzionati.

Francesco Rinaldi