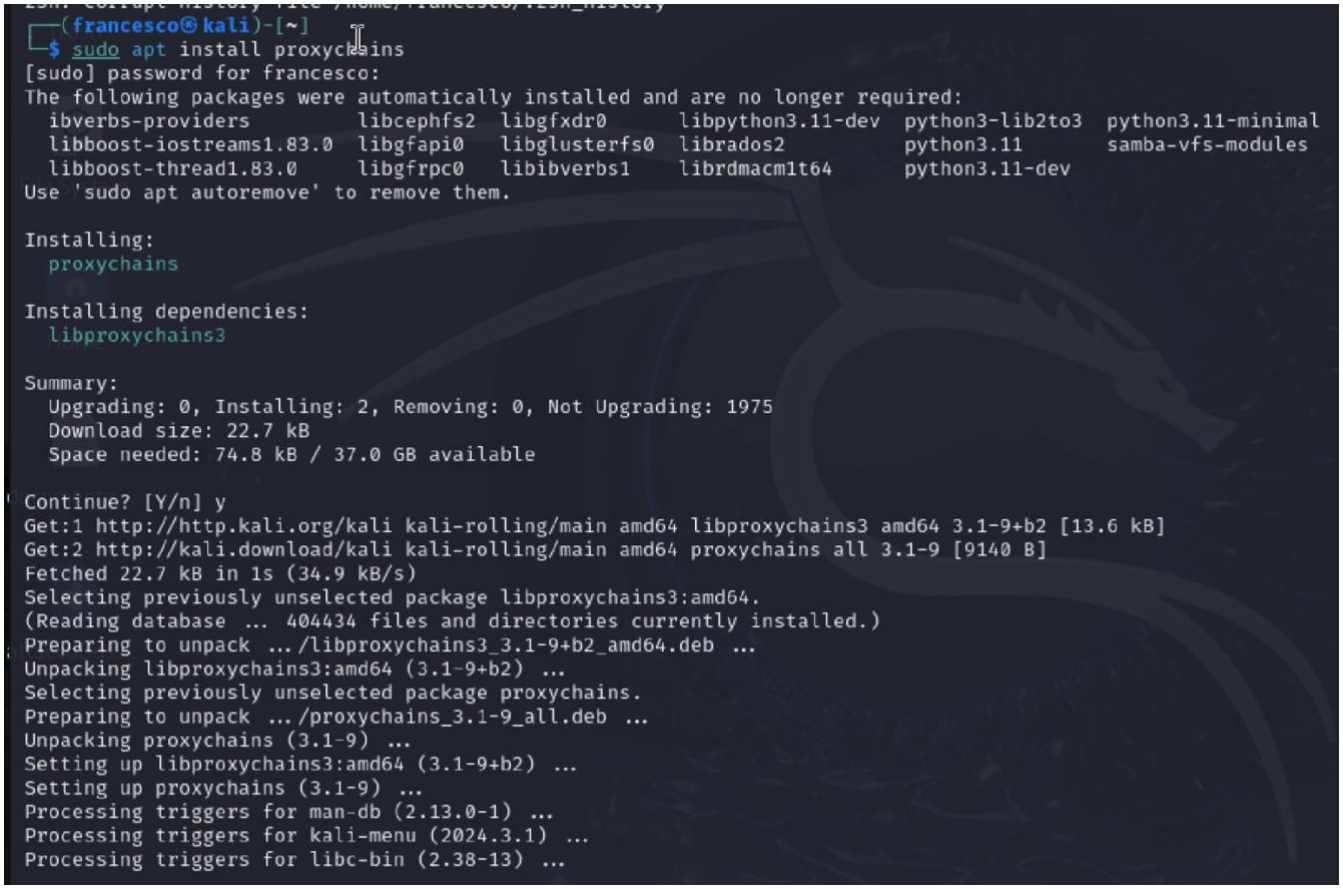
PROGETTO BONUS 2 S7L5

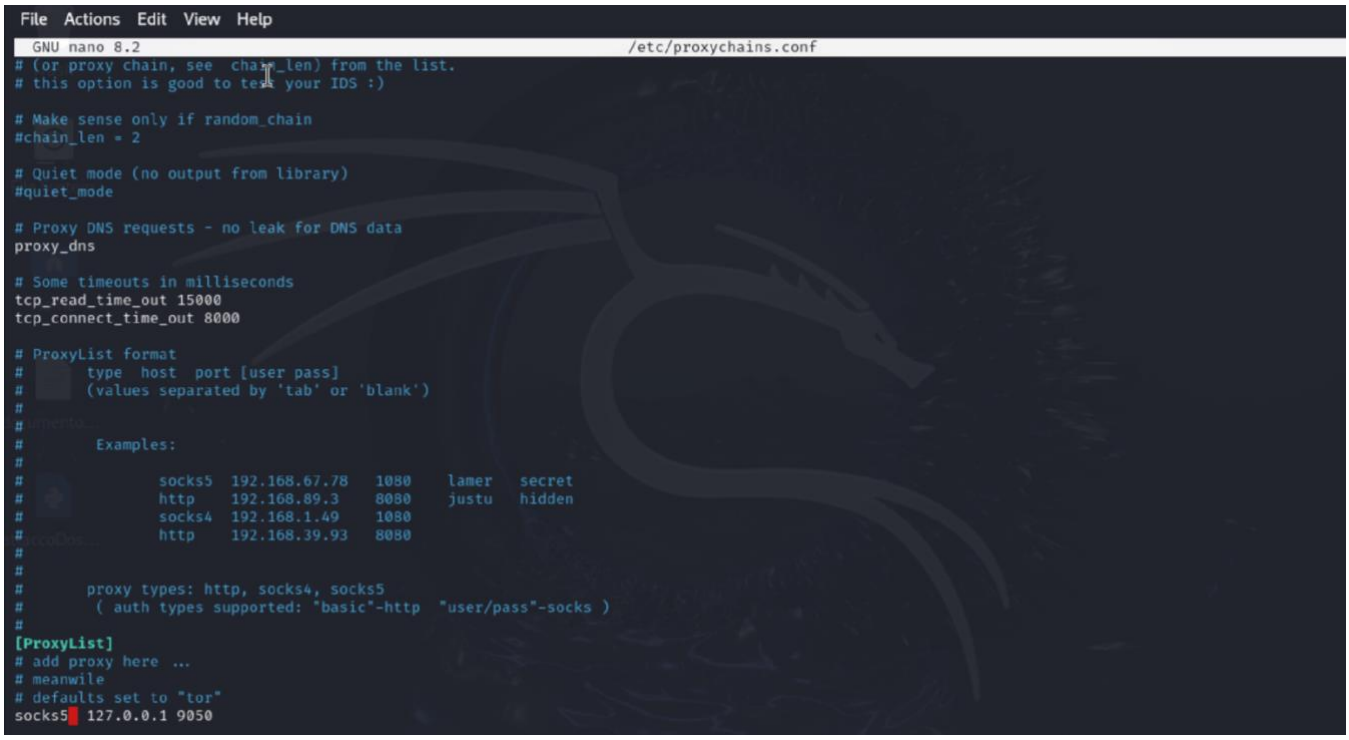
TRACCIA:

Effettuare una simulazione di un attacco al sito http://testphp.vulnweb.com/ passando da TOR.
Lo scopo dell'esercizio non è riuscire nell'attacco ma appunto attaccando dall'interno della rete TOR.
NOTA: attaccare il sito testphp.vulnweb.com è totalmente legale quindi non rischiate nulla, basta che non facciate un Ddos.

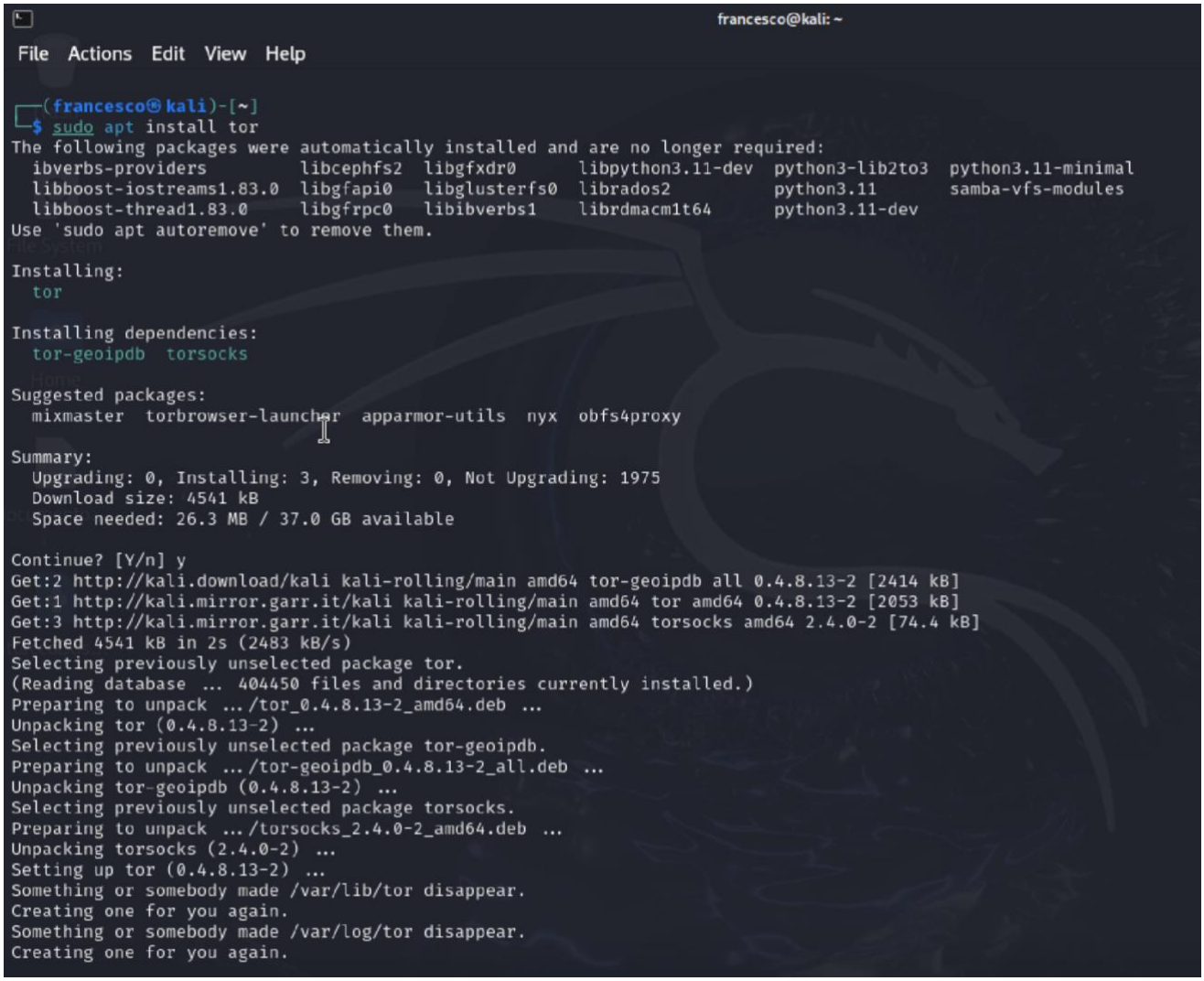
Per svolgere questo esercizio, utilizzerò proxychains, uno strumento che consente di indirizzare il traffico attraverso la rete TOR.
Per installarlo, basta eseguire il comando ***"sudo apt install proxychains":***



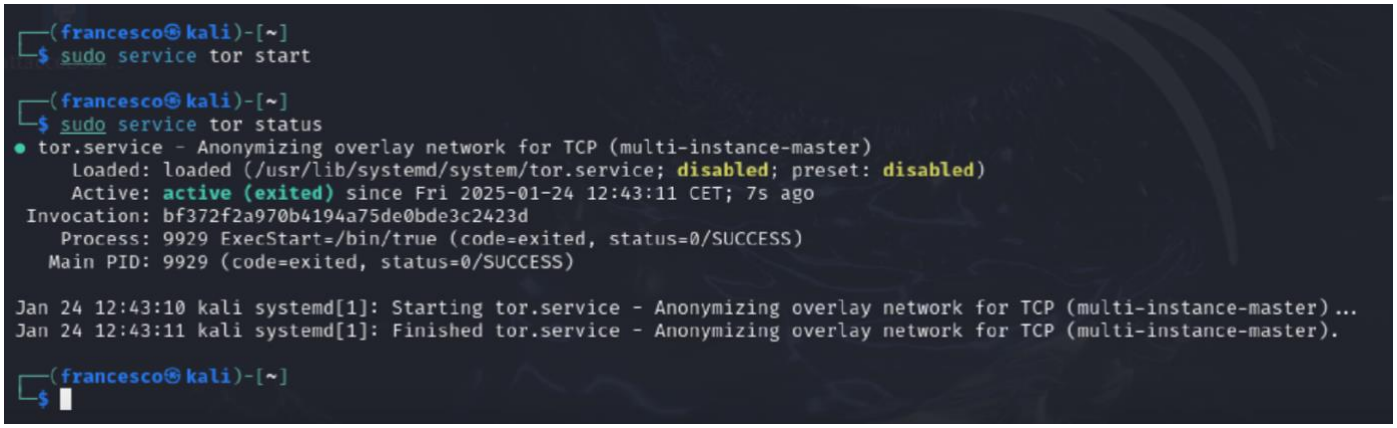
Una volta installato, posso procedere con la configurazione.
Per aprire il file di configurazione, utilizzo il seguente comando: ***"sudo nano /etc/proxychains.conf"***, successivamente scorro fino all'ultima riga, e come unica modifica cambio socks4 in socks5, salvo e chiudo il file.
Perché utilizzare il protocollo socks5?
Socks5 supporta l'autenticazione, è più sicuro e può gestire il traffico UDP, mentre socks4 è più vecchio e limitato.



Dopo aver installato e configurato nikto, procedo con l'installazione di tor:

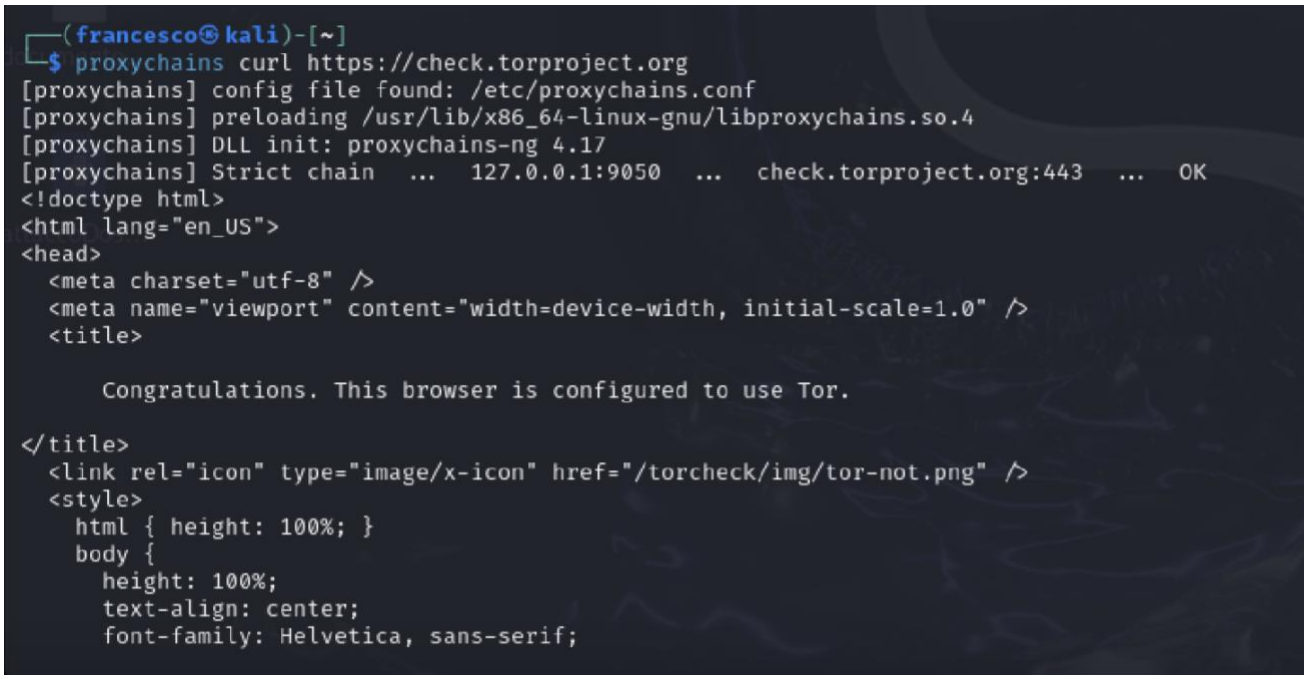


Successivamente avvio il servizio tor con il comando ***“sudo service tor start”*** e subito dopo controllo il suo stato con il comando ***“sudo service tor status”:***



Possiamo vedere che lo stato è “active”.

È arrivato il momento di testare Proxychains con Tor e verificarne il funzionamento.
Per farlo, utilizzo il comando ***"proxychains curl https://checktorproject.org":***

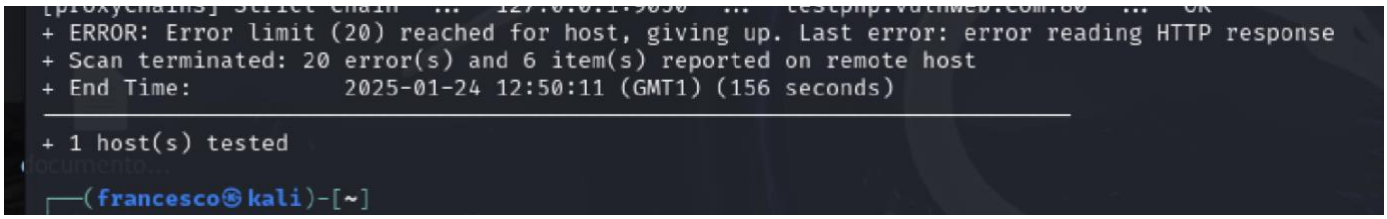


Si può intuire che la prova è andata a buon fine per aver ricevuto la risposta ***"Congratulations. This browser is configured to use Tor".***

Ora che tutto è pronto, posso utilizzare Nikto per scansionare il sito http://testphp.vulnweb.com/ passando dalla rete TOR.
Utilizzerò il seguente comando: ***"proxychains nikto -h http://testphp.vulnweb.com/".***

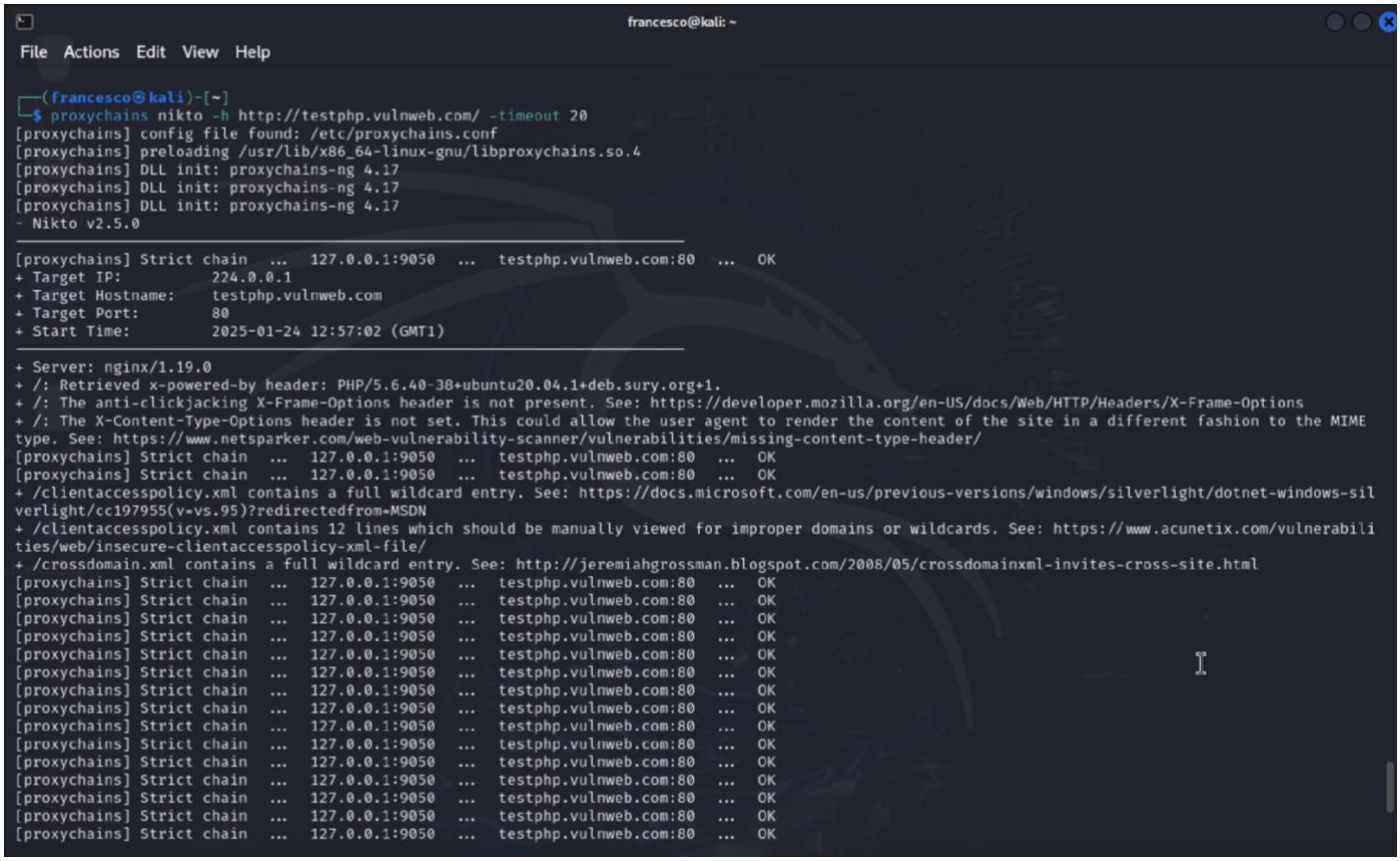
Proxychains: reindirizza il traffico attraverso TOR
Nikto: avvia Nikto
-h: specifica l'host da scansionare (in questo caso, testphp.vulnweb.com).

Vediamo quindi qual è il risultato della scansione:



Mi accorgo però, che alla fine della scansione vengono riportati degli, e dopo 20 errori Nikto interrompe la scansione.

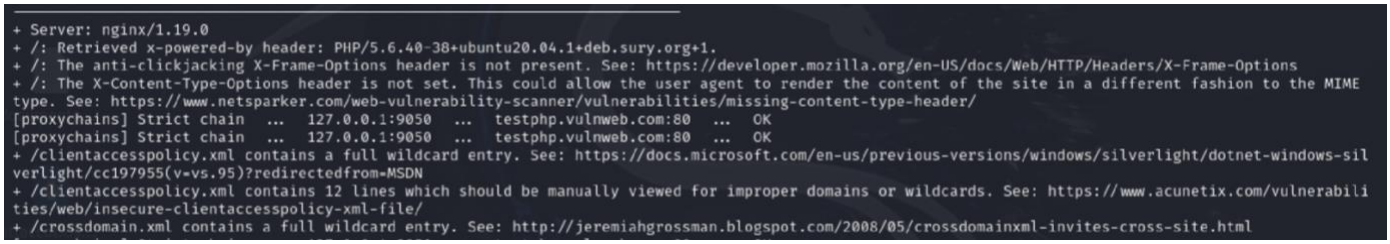
Per provare ad ottenere una scansione migliore, ho provato ad aumentare il timeout, quindi il tempo di attesa di Nikto per ricevere una risposta dal server, utilizzando il flag ***“-timeout”***.



Inoltre, si potrebbe togliere il limite di errori per fare in modo che Nikto continui la scansione nonostante i troppi errori riscontrati aggiungendo ***"- nolimit"*** al comando, o cambiare la porta per provare a forzarne un'altra.

In questo caso, lo scopo dell'esercizio di attaccare dall'interno della rete TOR è riuscito, ma c'è ancora una cosa che potrei fare.

Se l'attacco con Nikto è riuscito, l'output non significa che il sito è stato "violato" o compromesso, ma che Nikto è riuscito a individuare potenziali ***vulnerabilità***, ***configurazioni errate*** o ***problemi di sicurezza:***



Nell’immagine sopra, infatti, possiamo notare alcuni particolari come:
Server: nginx/1.19.0
Header mancanti:

- X-frame-options, la mancanza di questo header consente attacchi di clickjacking.
- X-Content-type-options, la sua assenza potrebbe permettere MIME sniffing.

Questi sono degli esempi delle vulnerabilità che ci fornisce Nikto dopo la scansione.

Conclusioni:

Nikto non esegue attacchi attivi, ma segnala possibili punti deboli. Con l’output ricevuto possiamo eseguire un’**analisi del rischio**, ulteriori **test**, **documentare i risultati** elencando le **vulnerabilità** e le **misure correttive**.