Assignment 1: TryHackMe: DNS In Detail

RENE OLUOCH
CS-CNS09-25030

Table of Contents

Introduction

I explored the foundational workings and practical applications of the Domain Name System (DNS), on the TryHackMe website(DNS in details), learning how it translates user-friendly domain names into machine-readable IP addresses. I gained insight into the structure of domain names, the function of various DNS record types, and the resolution process involving recursive, root, and authoritative servers. This theoretical knowledge was reinforced through an interactive simulation where I used tools like nslookup to perform real DNS queries. The hands-on experience solidified my understanding and highlighted the role of DNS in network communication and troubleshooting.

What is DNS?

In this room, I learnt that DNS (Domain Name System) serves as a fundamental component of internet communication by converting human-readable domain names into numerical IP addresses that computers use to identify each other on the network. Rather than having to memorize complex IP addresses like 104.26.10.229, DNS allows us to simply use easy-to-remember names such as tryhackme.com. This system functions much like a digital address book, streamlining the process of locating and communicating with websites or services online, and significantly enhancing user accessibility and efficiency when navigating the web.

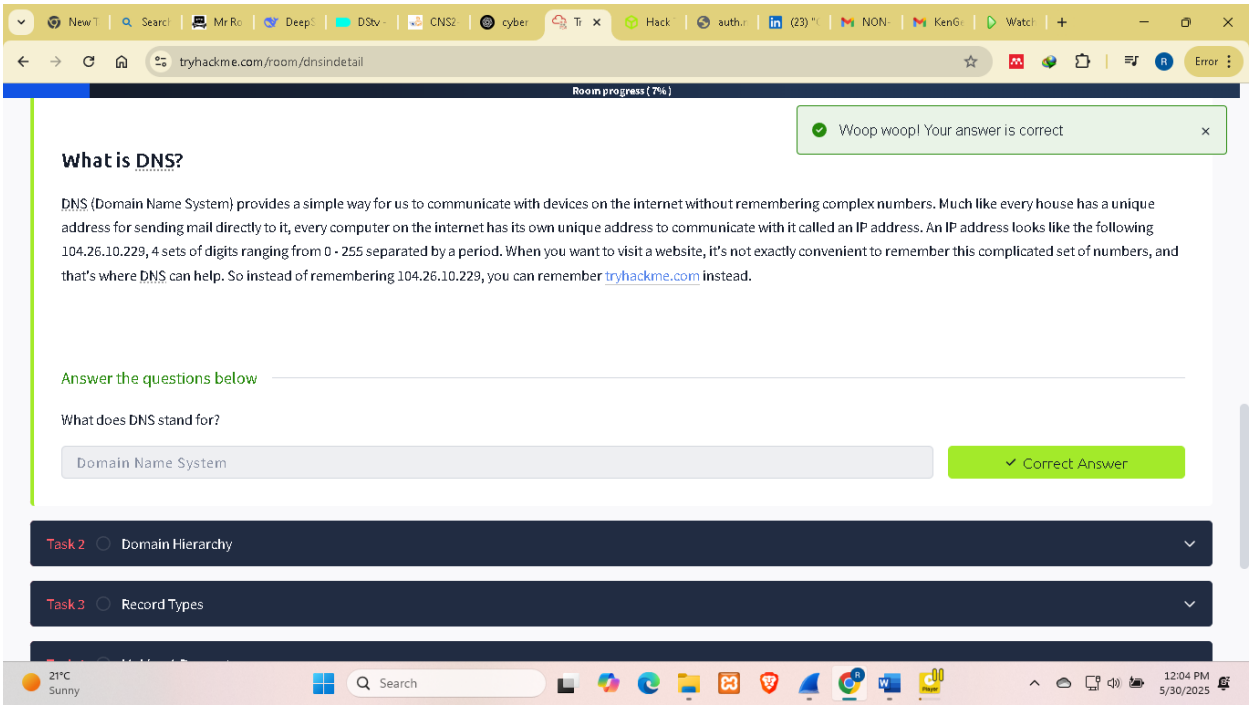

Fig 1.0 What is DNS

Domain Hierarchy

I learnt about the structure of domain names and how they are categorized. The Top-Level Domain (TLD) is the rightmost part of a domain name, such as ".com" in tryhackme.com. There are two main types of TLDs: generic TLDs (gTLDs), like .com, .org, and .edu, which traditionally indicate the purpose of a site, and country code TLDs (ccTLDs), such as .ca for Canada or .co.uk for the United Kingdom, which relate to geographic locations. With growing demand, many new gTLDs have been introduced, including .club, .online, and .biz. The Second-Level Domain (SLD) is the part directly before the TLD—like "tryhackme" in tryhackme.com—and follows certain naming conventions, allowing up to 63 characters, including letters, numbers, and hyphens, though not at the beginning or end or in consecutive positions. Subdomains are placed to the left of the SLD and separated by periods, like "admin" in admin.tryhackme.com. They follow similar character rules and can be layered (e.g., jupiter.servers.tryhackme.com), with a maximum full domain length of 253 characters. There is no limit to how many subdomains a domain can have, which allows for extensive customization and structuring of web addresses.

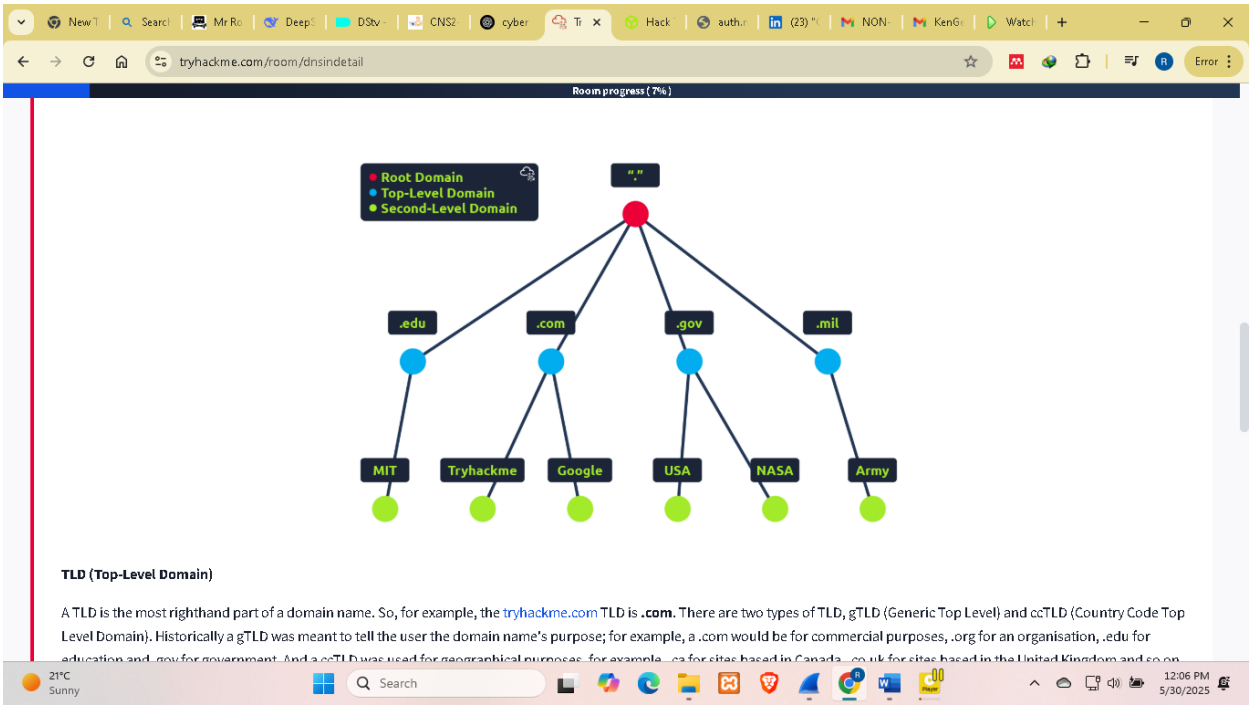

Fig 1.1 Domain Hierarchy

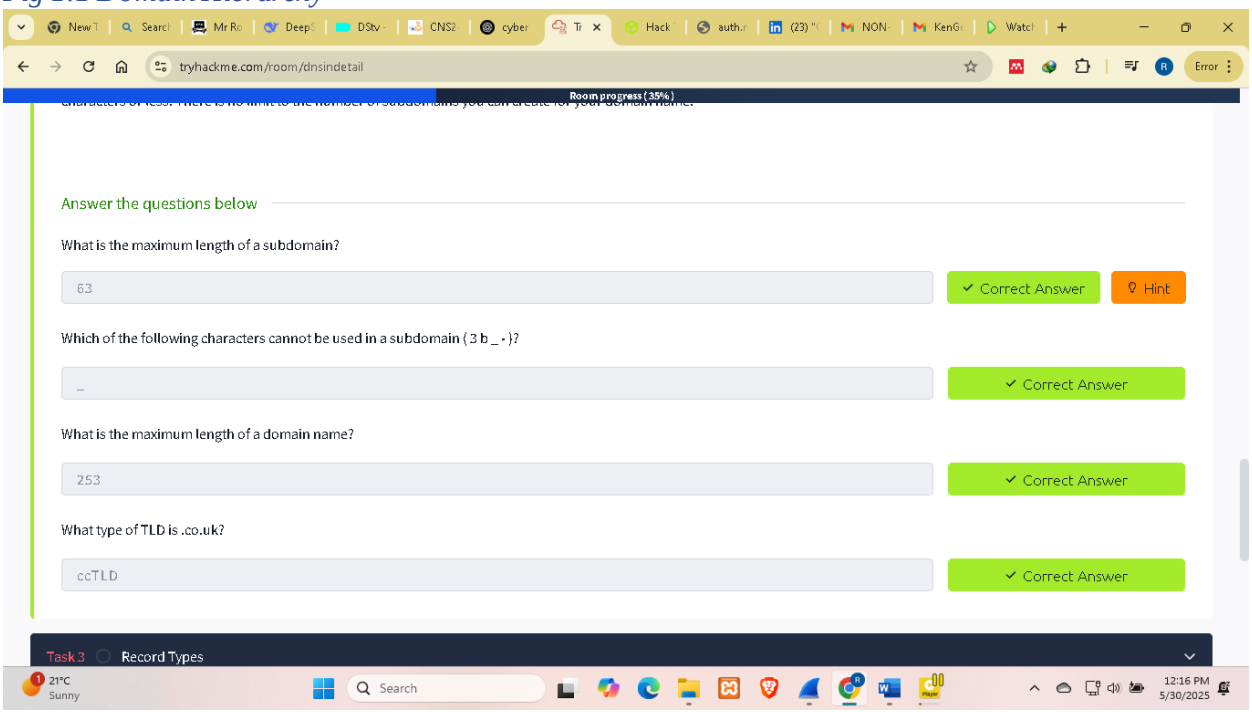

Fig 1.2 Domain Hierarchy Answers

DNS Record Types

DNS is not only used to translate domain names into IP addresses for websites, but it also supports multiple record types that serve different purposes in network communication. One of the most common is the A record, which maps a domain to an IPv4 address like 104.26.10.229. Similarly, the AAAA record performs the same function but for IPv6 addresses, such as 2606:4700:20::681a:be5. The CNAME record allows one domain name to point to another, effectively creating an alias—like store.tryhackme.com pointing to shops.shopify.com, which then requires another DNS lookup to resolve the IP address of the destination. MX (Mail Exchange) records are vital for email delivery, directing email traffic to the correct mail servers for a domain; they also include priority values to ensure messages are sent to backup servers if the primary one fails. Finally, TXT records allow domains to store arbitrary text data. These records are often used for email authentication, like listing which mail servers are allowed to send on behalf of the domain, and for verifying domain ownership during integration with third-party services. Overall, each record type plays a unique role in making the internet more functional and secure.

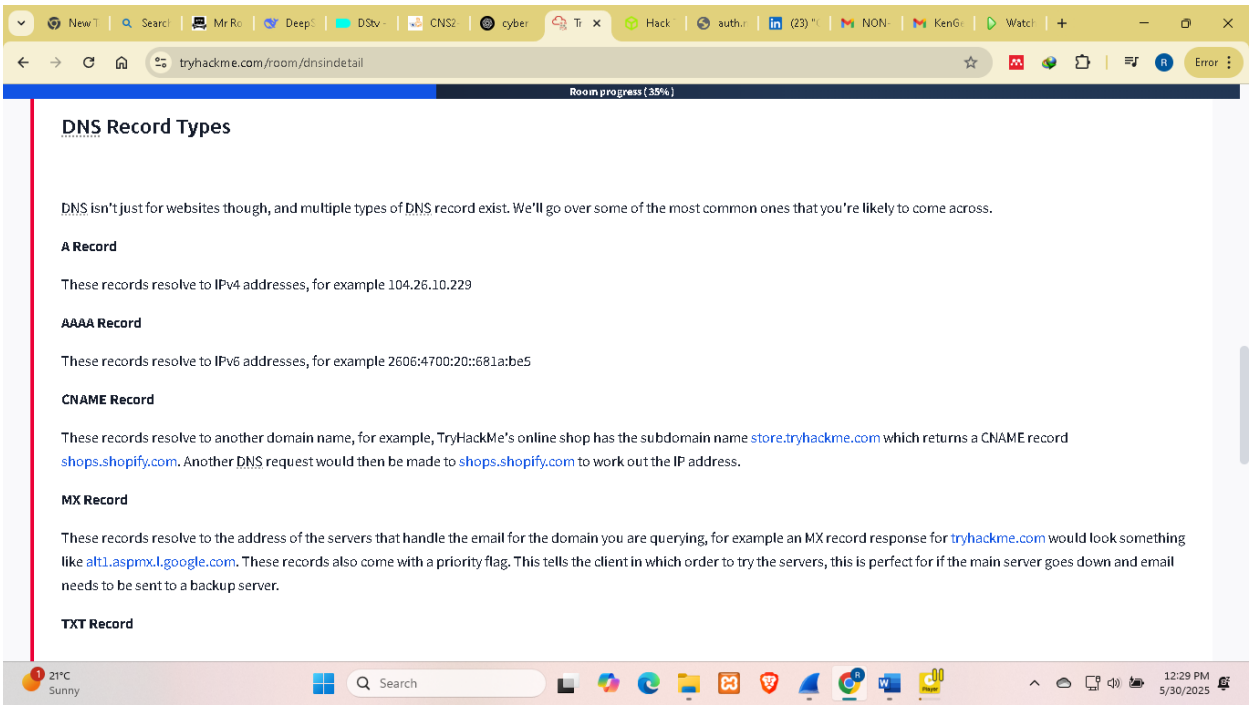

Fig 1.3DNS Record Types

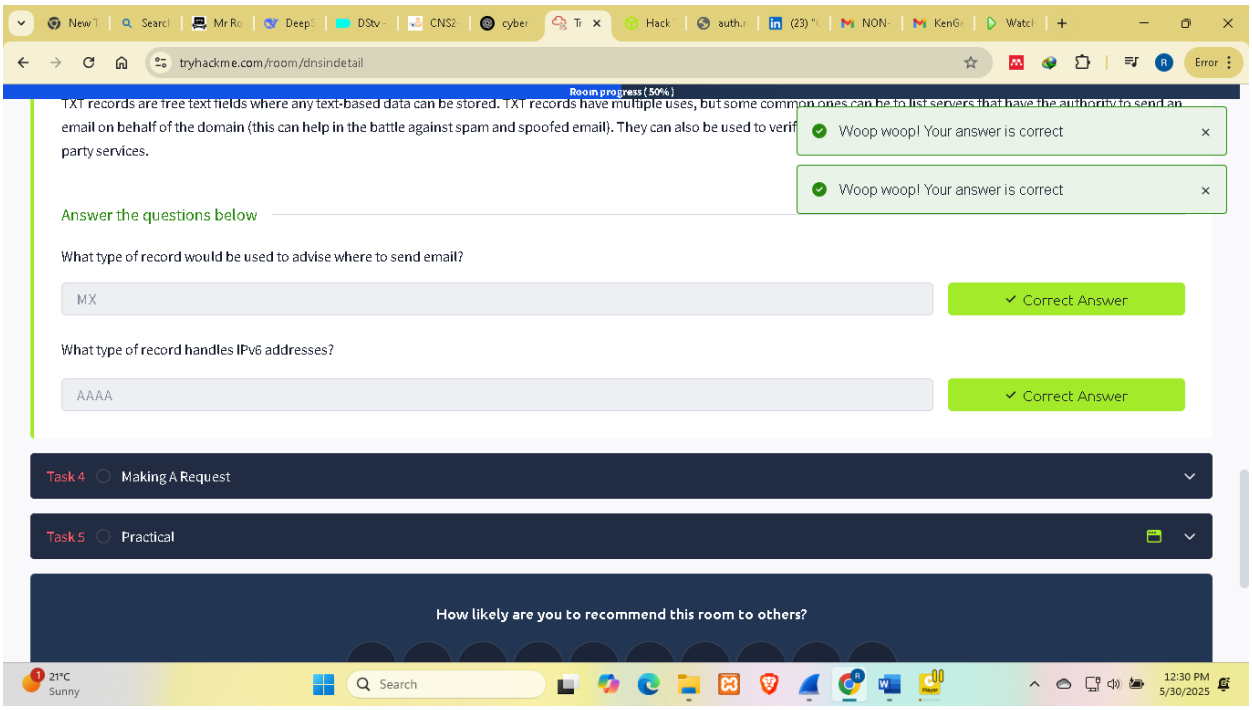

Fig 1.4DNS Record Types Answers

Making a request

What happens when you make a **DNS** request?

A DNS request works by following a step-by-step journey that begins with my computer and ends with retrieving the correct IP address for the domain I'm trying to access. The first thing that happens is my computer checks its own local DNS cache to see if it already knows the IP address for the requested domain. If the address is not cached locally, the request is sent to a Recursive DNS Server, which is usually provided by my ISP but can be configured manually as well. This server also checks its cache; if the answer isn't there, the real process of resolution begins. The recursive server first contacts one of the internet's root DNS servers, which are the foundational layer of DNS. These servers don't provide the final answer but direct the request to the appropriate Top-Level Domain (TLD) server based on the domain suffix—like .com, .org, or .net. Once the TLD server is reached, it points the recursive resolver to the authoritative name server that holds the actual DNS records for the domain in question. In the case of tryhackme.com, for example, these would be kip.ns.cloudflare.com and uma.ns.cloudflare.com. The authoritative server sends back the requested DNS record—such as an A record pointing to an IP address—along with a TTL (Time To Live) value that dictates how long the answer should be cached by the recursive server and the client. This caching significantly speeds up future requests to the same domain by reducing unnecessary lookups. Through this process, I gained a detailed understanding of the layered, efficient way DNS resolves domain names and ensures fast and reliable internet communication.

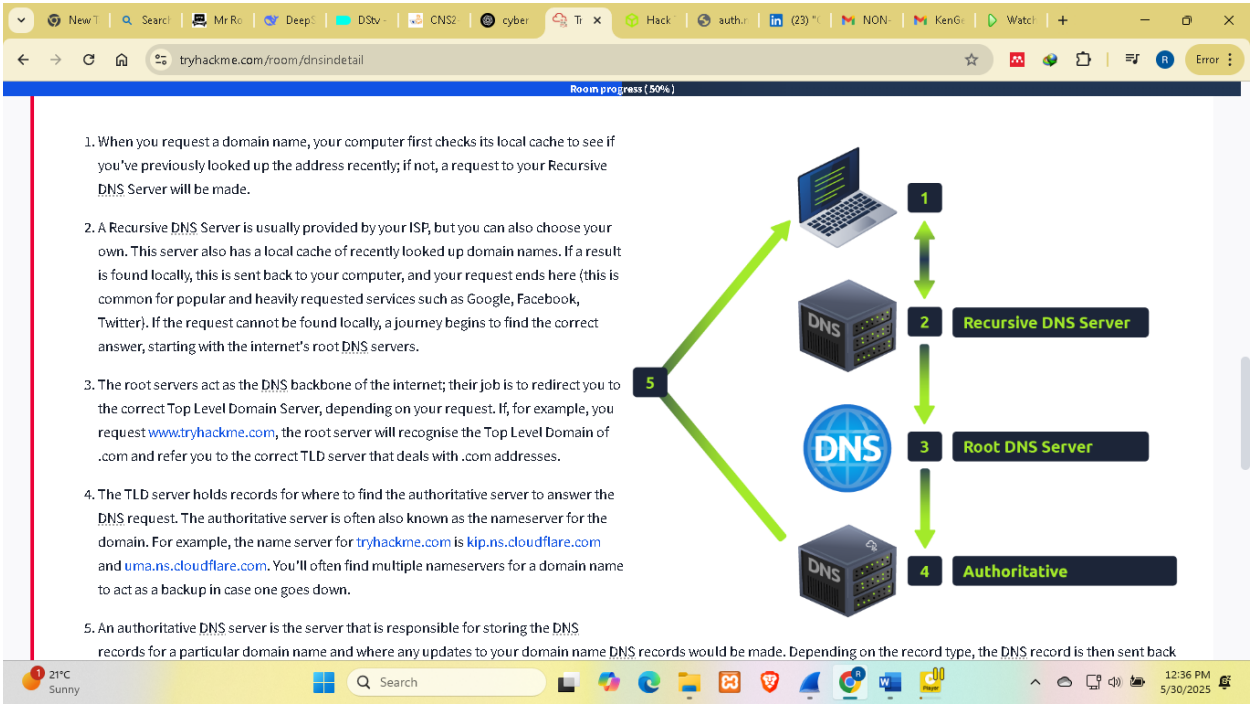

Fig 1.5Making a DNS request

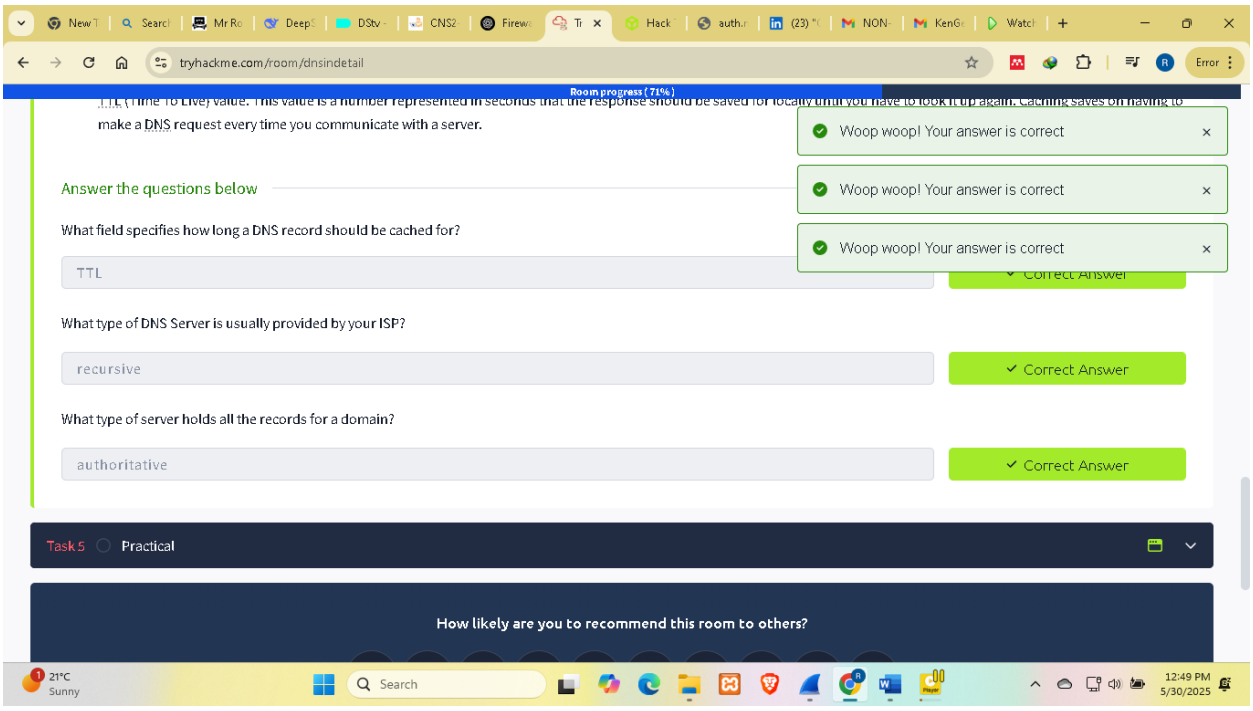

Fig 1.6Making a DNS request Answers

Practical part

As part of the practical exercise in this module, I explored how to perform DNS queries through an interactive browser-based virtual environment. Rather than typing commands manually, I was provided with a dropdown menu that allowed me to construct and execute different DNS requests. This setup simulated a real command-line environment and made it easier to focus on understanding the functionality of each record type. To begin, I used the dropdown to select and execute a CNAME query for the subdomain shop.website.thm. The result showed that the canonical name for this domain was shops.myshopify.com, confirming a redirect via a CNAME record.

Next, I queried the TXT record of the domain website.thm, and the result displayed a string: THM{7012BBA60997F35A9516C2E16D2944FF}. This is often used to verify domain ownership or for various authentication mechanisms. Following that, I selected an MX record query for the same domain and found that the mail exchanger had a priority value of 30, pointing to alt4.aspmx.l.google.com. Finally, I retrieved the A record for www.website.thm, which resolved to the IP address 10.10.10.10. These practical tasks allowed me to understand and visually confirm how different DNS records function, reinforcing the theoretical knowledge I had gained earlier in the module.

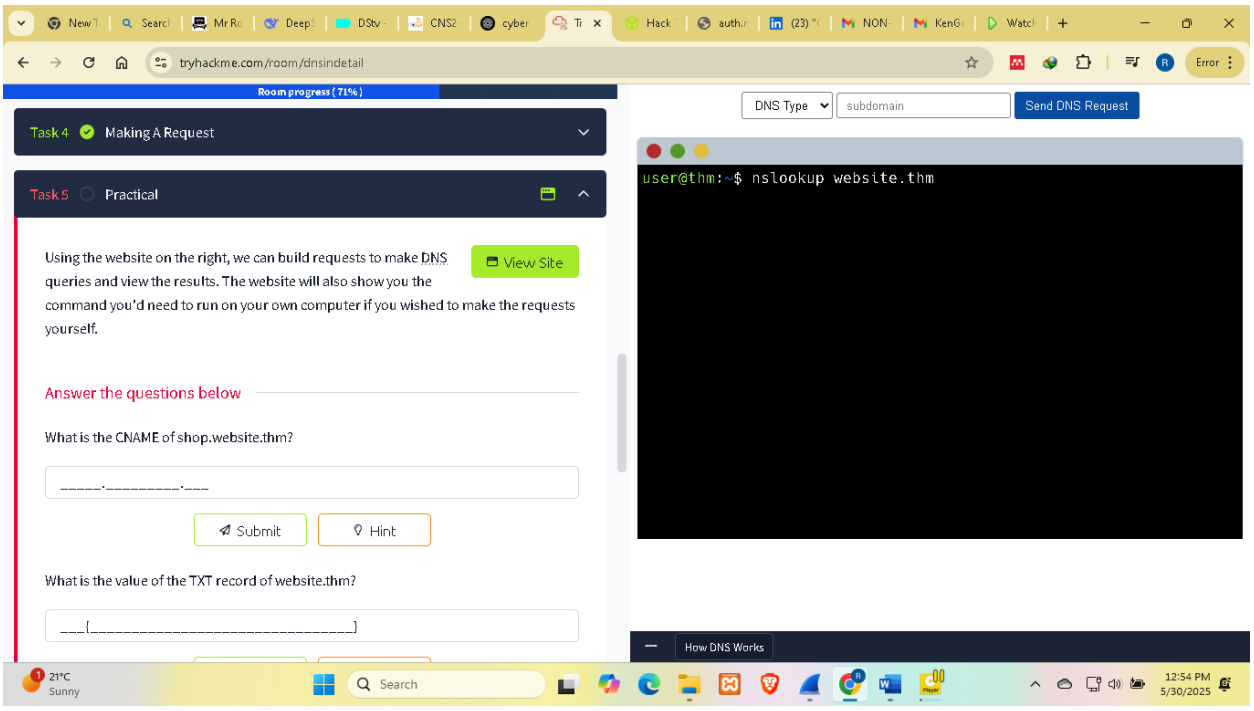

Fig 1.7Practical part with virtual terminal

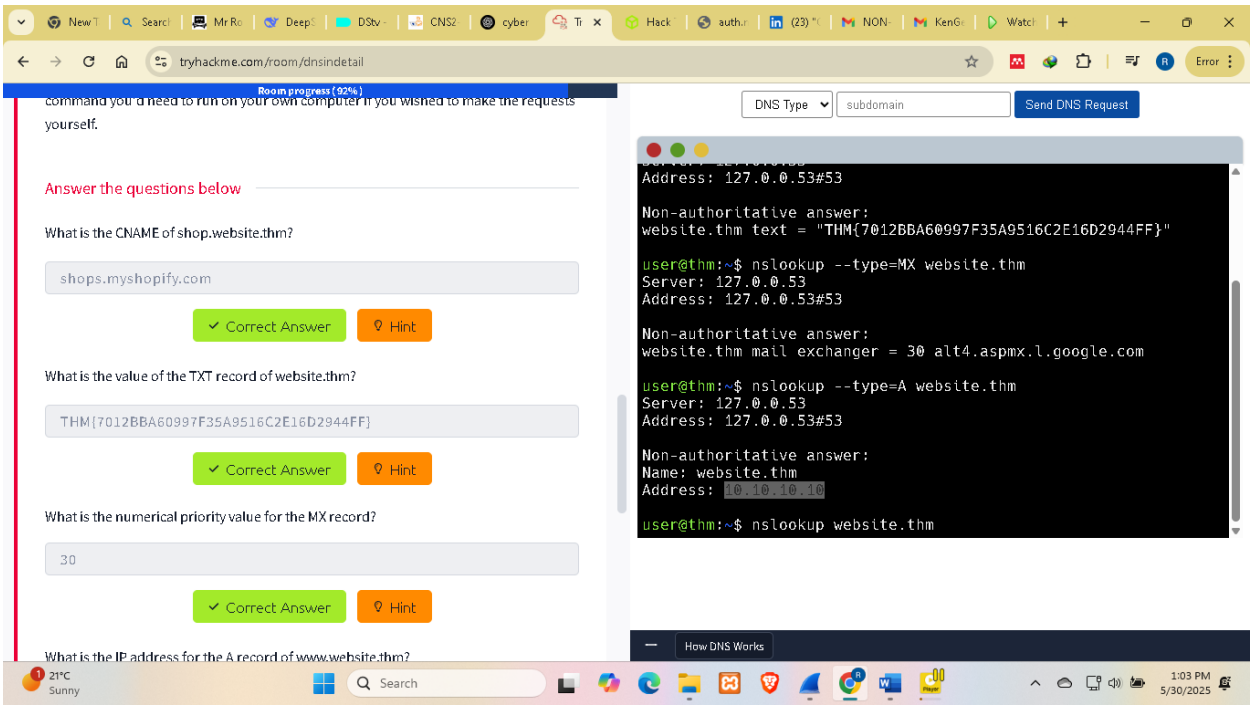

Fig 1.8Answers to the practical part

After finishing the room I was awarded a completion badge

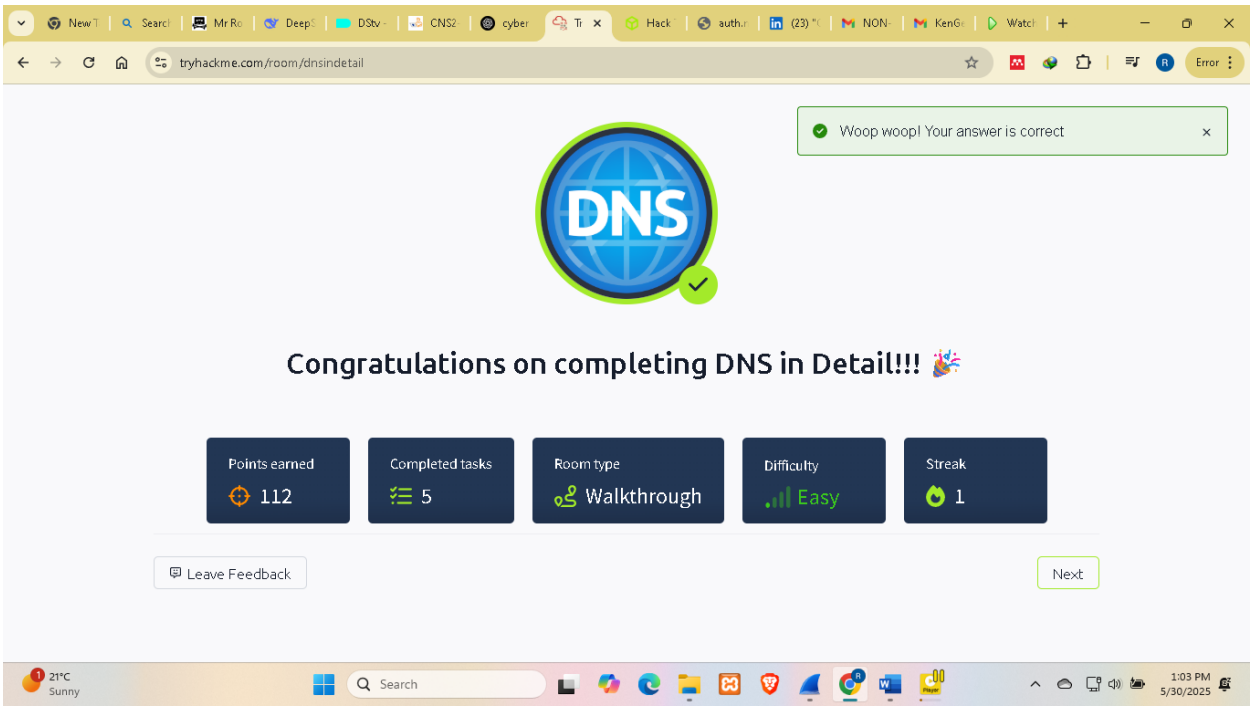

Fig 1.9Completion badge

Conclusion

By completing this module, I've gained both a theoretical and practical mastery of DNS operations. I can now confidently interpret domain hierarchies, distinguish between various record types like A, AAAA, CNAME, MX, and TXT, and understand their relevance in real-world scenarios such as website routing and email handling. The process of making a DNS request—once opaque—now makes sense to me from end to end, including the role of recursive resolvers, root servers, and authoritative name servers. The practical tasks solidified this knowledge by allowing me to execute queries and interpret responses in a simulated terminal. Overall, this module has equipped me with essential skills for network diagnostics, security assessments, and systems administration, making DNS less of a black box and more of a powerful tool in my technical toolkit.