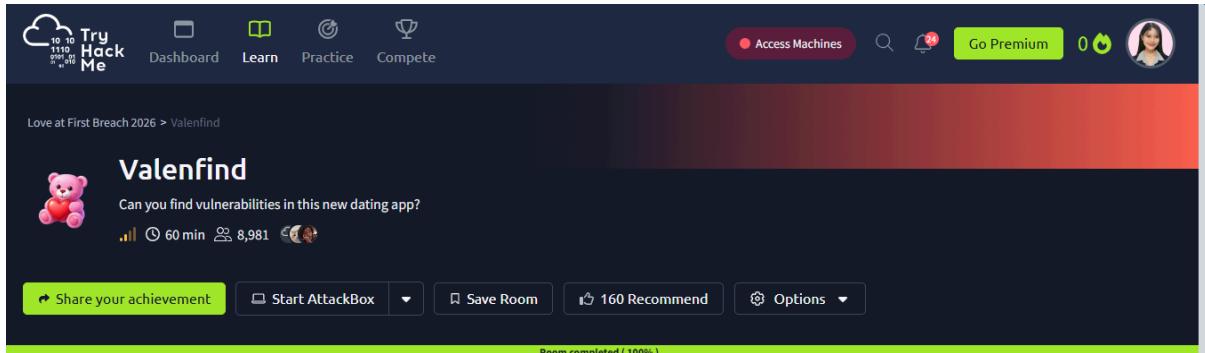


JASMINE OMANDAM

[TryHackMe | Love at First Breach 2026 Training](#)

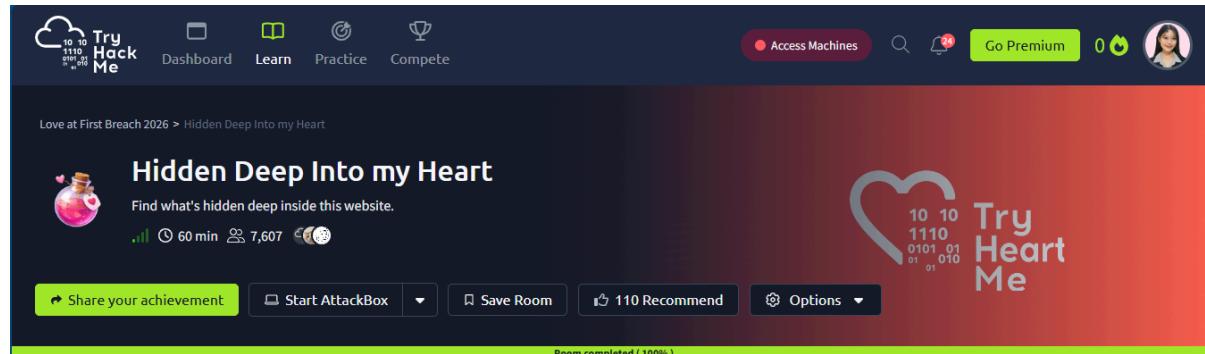
[TryHackMe | Valenfind](#)



A screenshot of the TryHackMe platform showing the 'Valenfind' challenge. The challenge title is 'Valenfind' with a pink teddy bear icon. Below it, the description reads 'Can you find vulnerabilities in this new dating app?'. It shows a progress bar at 100% completion. The challenge has a duration of 60 minutes and 8,981 solves. Buttons for 'Share your achievement', 'Start AttackBox', 'Save Room', 'Recommend', and 'Options' are visible.

In this challenge, I learned how beginner-coded web apps often contain simple but exploitable flaws. By exploring the dating app's inputs and endpoints, I noticed weak validation and insecure logic. I solved it through **systematic testing of parameters**, manipulating requests until I uncovered the vulnerability and retrieved the flag. This reinforced the value of **methodical probing, documenting each attempt, and thinking like both a developer and attacker** when approaching web exploitation.

[TryHackMe | Hidden Deep Into my Heart](#)



A screenshot of the TryHackMe platform showing the 'Hidden Deep Into my Heart' challenge. The challenge title is 'Hidden Deep Into my Heart' with a heart-shaped bottle icon. Below it, the description reads 'Find what's hidden deep inside this website.'. It shows a progress bar at 100% completion. The challenge has a duration of 60 minutes and 7,607 solves. Buttons for 'Share your achievement', 'Start AttackBox', 'Save Room', 'Recommend', and 'Options' are visible.

In this challenge, I learned how important it is to look beyond the surface of a web application to uncover hidden content. The site contained clues buried deep inside its structure, and by carefully inspecting directories and responses, I was able to reveal the secret data. I solved it through **systematic exploration** of the website, testing paths and analyzing outputs until the hidden information was exposed. This reinforced the value of patience and thoroughness in web exploitation.

[TryHackMe | Signed Messages](#)

The screenshot shows the TryHackMe platform interface for the 'Signed Messages' challenge. At the top, there's a navigation bar with 'Access Machines', a search bar, a notification bell with 24 notifications, and a 'Go Premium' button. On the right, there's a user profile icon. Below the navigation is a banner for 'Love at First Breach 2026 > Signed Messages'. The main content area features a title 'Signed Messages' with a smartphone icon, a description 'Their messages are secret, unless you find the key.', and stats: '60 min', '4,188', and a lock icon. Below this are buttons for 'Share your achievement', 'Start AttackBox', 'Save Room', '72 Recommend', and 'Options'. A progress bar at the bottom indicates 'Room completed (100%)'.

In this challenge, I learned how trust systems in web applications can be broken if message verification is poorly implemented. The platform relied on signatures, but by analyzing how they were generated and validated, I discovered weaknesses in the process. I solved it by **testing the signing mechanism**, manipulating inputs, and eventually forging a valid message to bypass the system. This reinforced the importance of understanding both **web logic and cryptographic checks** when exploiting applications.

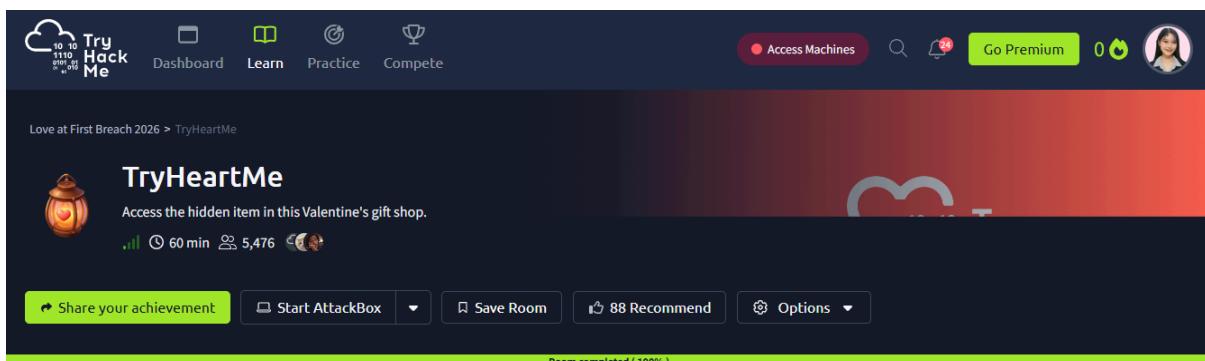
[TryHackMe | CupidBot](#)

The screenshot shows the TryHackMe platform interface for the 'CupidBot' challenge. The layout is similar to the previous challenge, with a navigation bar, a search bar, a notification bell with 24 notifications, and a 'Go Premium' button. The main content area features a title 'CupidBot' with a heart icon, a description 'This AI bot knows more than just love letters.', and stats: '60 min', '7,757', and a lock icon. Below this are buttons for 'Share your achievement', 'Show Split View', 'Start AttackBox', 'Save Room', '112 Recommend', and 'Options'. A progress bar at the bottom indicates 'Room completed (100%)'.

In this challenge, I learned how **prompt injection vulnerabilities** can expose hidden data in AI-driven web applications. The chatbot stored multiple flags, and by carefully crafting inputs, I was able to bypass its intended behavior and extract them. I solved it through **systematic input manipulation**, testing different prompts until the bot revealed the hidden flags. This reinforced the importance of understanding how AI systems process instructions and how attackers can exploit that trust.

\

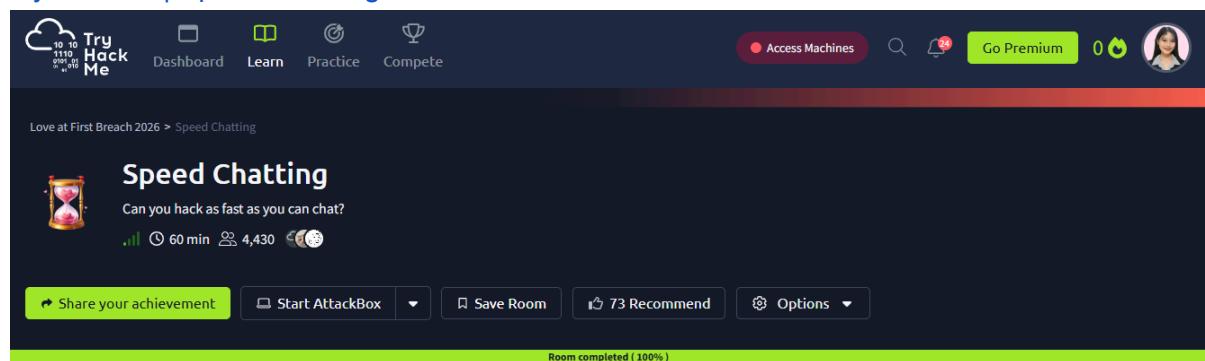
[TryHackMe](#) | [TryHeartMe](#)



The screenshot shows the TryHackMe interface for the challenge "TryHeartMe". The challenge title is "TryHeartMe" with a heart icon. The description is "Access the hidden item in this Valentine's gift shop." It includes a timer icon (60 min), a user count (5,476), and a difficulty rating (bronze). Below the challenge are several buttons: "Share your achievement", "Start AttackBox", "Save Room", "Recommend" (88), and "Options". At the bottom, a progress bar indicates "Room completed (100%)".

In this challenge, I learned how hidden items in web applications can be uncovered by carefully analyzing requests and responses. The shop contained a secret product, and by manipulating parameters and exploring the site's logic, I was able to access the hidden flag.

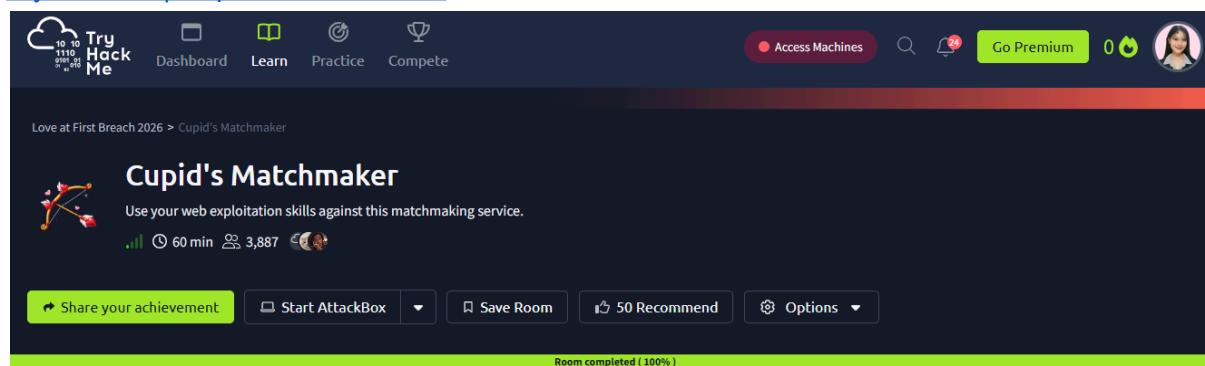
[TryHackMe](#) | Speed Chatting



The screenshot shows the TryHackMe interface for the challenge "Speed Chatting". The challenge title is "Speed Chatting" with a clock icon. The description is "Can you hack as fast as you can chat?". It includes a timer icon (60 min), a user count (4,430), and a difficulty rating (bronze). Below the challenge are several buttons: "Share your achievement", "Start AttackBox", "Save Room", "Recommend" (73), and "Options". At the bottom, a progress bar indicates "Room completed (100%)".

In this challenge, I learned how rushing applications into production without proper testing often leaves exploitable flaws. The messaging platform had weak security controls, and by probing its inputs and endpoints, I uncovered vulnerabilities that exposed sensitive data.

[TryHackMe](#) | Cupid's Matchmaker



The screenshot shows the TryHackMe interface for the challenge "Cupid's Matchmaker". The challenge title is "Cupid's Matchmaker" with a bow and arrow icon. The description is "Use your web exploitation skills against this matchmaking service.". It includes a timer icon (60 min), a user count (3,887), and a difficulty rating (bronze). Below the challenge are several buttons: "Share your achievement", "Start AttackBox", "Save Room", "Recommend" (50), and "Options". At the bottom, a progress bar indicates "Room completed (100%)".

In this challenge, I learned how matchmaking services can be vulnerable when user input isn't properly secured. By exploring the survey and submission process, I found weaknesses that allowed me to bypass the intended flow and uncover the hidden flag.

[TryHackMe | When Hearts Collide](#)

The screenshot shows the TryHackMe platform interface. At the top, there's a navigation bar with icons for Dashboard, Learn (highlighted in green), Practice, and Compete. On the right, there are buttons for Access Machines, a search bar, a notifications icon (24), a Go Premium button, and a user profile picture. Below the navigation is a red banner with the text "Love at First Breach 2026 > When Hearts Collide". The main content area has a dark background with a title "When Hearts Collide" and a subtitle "Will you find your MD5 match?". It includes a timer icon showing "60 min", a view count of "3,337", and a small profile picture. Below the title are several buttons: "Share your achievement", "Start AttackBox", "Save Room", "50 Recommend", and "Options". A progress bar at the bottom indicates "Room completed (100%)".

In this challenge, I learned how applications that rely on **MD5 hashing** can be exploited when the algorithm is used insecurely. The matchmaking system compared hashes, which made it possible to manipulate inputs and generate collisions. I solved it by **analyzing the hashing process** and crafting inputs that matched the expected MD5 values, allowing me to bypass the intended logic and reveal the flag. This reinforced the importance of avoiding weak hashing algorithms in web applications.

-
-  **Valenfind** 
Can you find vulnerabilities in this new dating app?
 -  **Hidden Deep Into my Heart** 
Find what's hidden deep inside this website.
 -  **Signed Messages** 
Their messages are secret, unless you find the key.
 -  **Corp Website** 
lafb2026-e7
 -  **CupidBot** 
This AI bot knows more than just love letters.
 -  **TryHeartMe** 
Access the hidden item in this Valentine's gift shop.
 -  **Speed Chatting** 
Can you hack as fast as you can chat?
 -  **Cupid's Matchmaker** 
Use your web exploitation skills against this matchmaking service.
 -  **Love Letter Locker** 
Use your skills to access other users' letters.
 -  **When Hearts Collide** 
Will you find your MD5 match?
 -  **Topic Rewind Recap**
Lock in what you learned with a recap. Earn points and keep your streak.