|  |  |
| --- | --- |
| Logo  Description automatically generated | Challenge Writeup  NoCrypt Keys  by Kouretas Panagiotis |

# Main Concept

# Traverse the enigmatic world of NoCrypt, where Jason's Legacy holds the key to unlocking arcane data vaults. Compose the sequel to this cryptic tale by decoding the messages concealed in plain sight.

# Exploitation

# NoCrypt Keys

First of all, we deploy the challenge from CTFLib.

Then, as seen in Figure 1.1, we are sent to the challenge's home page at <http://localhost:4242/> which looks like a common login page.

A screenshot of a computer screen

Description automatically generated

Figure 1. 1 - Challenge’s Home Page / Login Page.

With a quick look around at the page’s source code (pressing “CTRL + U” on the page) I couldn’t any helpful information or hints.

So, I tried testing some possibly valid credentials to bypass this login page. I used many combinations (“Username: admin” and “Password: admin” etc.) and every time I got the same error response as depicted in Figure 1.2.

A screenshot of a login box

Description automatically generated

Figure 1 2 - Trying to login without an account.

When I tried clicking the “Forgot your credentials?” button, this is the response I got as seen in Figure 1.3.

A black screen with white text

Description automatically generated

Figure 1 3 - Functionality error message

So far, we have no guidelines to move forward. However, upon closer inspection of the challenge description, we find that a few keywords seem to be implying something helpfull.

Challenge description:  
"Traverse the enigmatic world of NoCrypt, where Jason's Legacy holds the key to unlocking arcane data vaults. Compose the sequel to this cryptic tale by decoding the messages concealed in plain sight."  
  
There are three keywords in this description “NoCrypt”, “Jason” and “sequel”.  
Further consideration of these terms leads us to the NoSQL injection vulnerability since NoSQL databases like MongoDB or other NoSQL databases interpret JSON-style queries.

Let's now look for NoSQL injection payloads we can use to bypass this login page.

I discovered a GitHub repository with a variety of payloads that can be useful in our situation.

<https://github.com/swisskyrepo/PayloadsAllTheThings/tree/master/NoSQL%20Injection>

After running a few tests, I ended up with the following payload:

{"username": {"$ne": null}, "password": {"$ne": null}}

Let's break it down a little bit.

{"username": {"$ne": null}} is a query that can be used in NoSQL databases to specify a condition where the username field should not be equal to null ($ne which stands for "not equal"). When this is part of a login query, it essentially says "find me a user whose username is anything but null". Since most usernames in a database are indeed not null, this condition will be true for all users.

{"password": {"$ne": null}} applies the same logic to the password field. This means "find me a user whose password is anything but null". Again, this will be true for all users with a password set.

A screenshot of a login box

Description automatically generated

Figure 1 4 - Payload execution.

A message on a computer screen

Description automatically generated

Figure 1 5 - Hidden Flag Successfully Retrieved.

GOT IT, we successfully bypassed the login page and retrieved the hidden FLAG:

CTFLIB{N0$QL\_R2v3il3d\_D@t@-V@ult$}