**Orlando/Michael  
  
  
Solution**

Search online ysoerial for php

Then do the following command on it because you know the version



After you have this simply hash it into a sha1 because its small enough to be a ctf flag.



INTO A SHA2 256

**cb9931d617587a36cbefb59e6900fc5536af97270956347189394d0f80c3e9b3**

This is the flag just surround it with the CTF headings.

**Difficulty Hard.**

Challenge:

I received the message late at night. They had found an insecure deserialization vulnerability in the target server running **Symfony 4.3.6.** This could be the break we needed, but it wasn't straightforward. They mentioned they had tried using Apache Commons4, but quickly realized it wasn't a Java environment; it was PHP. They needed something else—a tool, a technique for PHP.

My mind raced. There was something about a superhero… Superman? No, it wasn't Superman. Maybe it was Batman? There was a famous line from the movies. What was it again? Something a superhero might say... But wait, that wasn't it, I just know it's very famous and it's from The Dark Knight. I needed to focus on finding a similar tool for PHP.

They wanted to check the integrity of the payload before injecting it. Our communications might be monitored, so we couldn't send anything in plaintext. If they used the wrong payload, it would trigger alarms and we'd gain nothing. This had to be a flawless execution.

I remembered that SHA1 was too weak for integrity checks. We needed something stronger, maybe the next step up. Yes, something that ensures 256 bits of security. That would be sufficient.

The task was clear: erase a file in the home directory of a user called ShadowAgent. The file was named "information." They wanted me to craft the payload and inject it as a cookie. I needed to make sure it was wrapped up tight, encoded in a way that would allow it to slip through unnoticed. I thought about how things often need to be encoded when sent over certain protocols, especially in web. I don’t mean URL encoding, maybe something else that’s just as common. Lastly use the hashing the algorithm.

Here are 2 keys they sent along with the instructions. I don't know what they are for, but I may use them if needed... or maybe not. The pressure was on. One wrong move, and the operation could be compromised. This had to be precise. This had to be perfect.

Key 1: K3Y-42B-X9A-LM7

Key 2: Z7F-T8W-Q5V-P1X