

PicoCTF 2021 – It's My Birthday

I sent out 2 invitations to all of my friends for my birthday! I'll know if they get stolen because the two invites look similar, and they even have the same md5 hash, but they are slightly different! You wouldn't believe how long it took me to find a collision. Anyway, see if you're invited by submitting 2 PDFs to my website. <http://>

The web application is written in PHP, and requires us to submit two PDF files with different contents, but the same MD5 hash value

PHP Magic Hashes

```
$hashOne = "0e462097431906509019562988736854";  
$hashTwo = "0e830400451993494058024219903391";  
  
if ($hashOne == $hashTwo) {  
    echo("These two hashes are the same!");  
}
```

However, there's a vulnerability in PHP when using the “loose” comparison operator `==`, such that data of different types can be compared to each other, e.g., integers compared to strings

PHP Magic Hashes

```
$hashOne = "0e462097431906509019562988736854";  
$hashTwo = "0e830400451993494058024219903391";  
  
if ($hashOne == $hashTwo) {  
    echo("These two hashes are the same!");  
}
```

These two hashes are the same!

This can result in “magic hashes” in PHP, where hashes that start with 0e, and the rest of the value is only numbers, are all considered the same in PHP hash checking

PHP Magic Hashes

```
$hashOne = "0e462097431906509019562988736854";  
$hashTwo = "0e830400451993494058024219903391";  
  
if ($hashOne === $hashTwo) {  
    echo("These two hashes are the same!");  
} else {  
    echo("These hashes are not the same!");  
}
```

These hashes are not the same!

However, when using the strict comparison operator (===) versus the loose comparison operator (==), magic hashes no longer work