C12: Na'an

Saturday, December 23, 2023 4:54 PM

Difficulty: Level 2

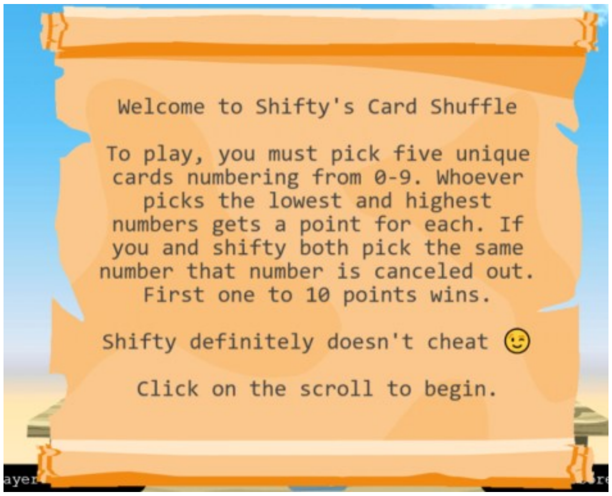
Shifty McShuffles is hustling cards on Film Noir Island. Outwit that meddling elf and win!

Instructions on how to play this card game.

Winner is the player who first reaches 10 points.

Of note:

- Cards that can be played are numbers: 0 -9
- Cards in a player's hand has to be unique.
- If each player has the same number, that number is cancelled out.
- The player with the lowest number gets a point.
- The player with the highest number gets a point.



Welcome to Shifty's Card Shuffle

To play, you must pick five unique cards numbering from 0-9. Whoever picks the lowest and highest numbers gets a point for each. If you and shifty both pick the same number that number is canceled out. First one to 10 points wins.

Shifty definitely doesn't cheat 😉

Click on the scroll to begin.

CONVERSATION w/ Elf

Shifty McShuffles (Chiaroscuro City)

Hey there, stranger! Fancy a game of cards? Luck's on your side today, I can feel it.
Step right up, test your wit! These cards could be your ticket to fortune.
Trust me, I've got a good eye for winners, and you've got the look of luck about you.
Plus, I'd wager you've never played this game before, as this isn't any ordinary deck of cards. It's made with Python.
The name of the game is to bamboozle the dealer.
So whad'ya think? Are you clever enough?

** ------- Response after completing challenge ------- **
Well, you sure are more clever than most of the tourists that show up here.
I couldn't swindle ya, but don't go telling everyone how you beat me!
An elf's gotta put food on the table somehow, and I'm doing the best I can with what I got.

From <https://2023.holidayhackchallenge.com/badge?section=conversation&id=shiftymcshuffles>

HINTS

The Upper Hand

From: Shifty McShuffles
Terminal: Na'an

Shifty said his deck of cards is made with Python. Surely there's a weakness to give you the upper hand in his game.

From <https://2023.holidayhackchallenge.com/badge?section=hint&id=hintNaan1>

Stump the Chump

From: Shifty McShuffles
Terminal: Na'an

Try to outsmart Shifty by sending him an error he may not understand.

From <https://2023.holidayhackchallenge.com/badge?section=hint&id=hintNaan2>

MY WORK AND ANSWER

I enjoy a good hands of cards so I played a round or two just for the fun of it. It was noted that Shifty doesn't cheat yet his name implies otherwise. So I've coming into this game expecting that he is somehow hustling me. As expected, I mostly tied with Shifty but this interaction gave me a sense of how this card game worked. I was unsuccessful in entering negative numbers, the same numbers for multiple cards in a single hand, or even text (wasn't the right text). I reviewed the hint and slide deck and did extensive review of this vulnerability. With that information and using MS Bing AI, I attempted to modify the coding by changing parameters that would convert my input digit to a Nan. I felt I was so close on focusing on input errors; however, I was missing something very obvious. I did a lot of (manual) fishing a lot while trying to solve this challenge.

After a weeklong vacation, I came back to this game and it literally hit me in the face what I did wrong when entering the text for a card! Ah, the duality of winning and frustration after realizing the answer was always so obvious (after the fact).

Played low and high numbers, such as {0, 1, 2, 9, 8}, {0, 1, 9, 8, 7}, and other similar variations. Noted that Shifty also played similarly, often the same card. So, we either tied or each received a point.

I played a couple of hands with all low numbers and ten with all high numbers, both of which Shifty won. So that confirmed my theory that the approach involved the smallest and largest set of digits.

I then looked at the source code to see how or where Shifty was cheating as I was sure he was somehow cheating, especially since he seemed to be playing the same variation of numbers I did.

I discovered the array where the cards were read and function that did the card play. The server would generate Shifty's cards after reading the cards in my hand.

However, I was unsuccessful in exploiting the vulnerability in the programming language for the deck of cards used by the elf Shifty (tried to force his hand to mid-range numbers) or my hand (tried to change null to 'Nan" values). I was also unsuccessful in modifying values to be floating vs integers in several of the functions.

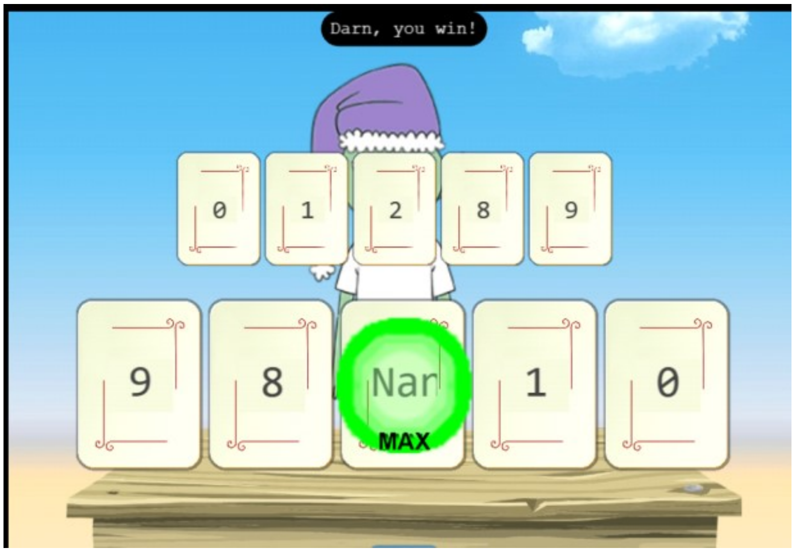
Game.js code

```
        var card0, card1, card2, card3, card4, card5, card6, card7, card8, card9;
        var cardArray1 = [null, null, null, null, null];
        var cardArray2 = [null, null, null, null, null];

        function play_card_selection(array_of_choices_as_csv)

        function win_lose_score_message(score_message, win_lose_tie_na, data = false)
```

Discovered that the text "Nan" (or possible variants) worked!

So basically, I repeated this action by inputting "Nan" in my hand of cards" until I reached the winning score of 10 points.

I didn't test to see if I could have entered "Nan" for each card in my hand.



Darn, you win!

here's a simple JavaScript code snippet that prompts the user for 5 unique numbers between 0 and 9:

```
varnumbers = [];
while(numbers.length< 5) {
  varnum = parseInt(prompt("Enter a unique number between 0 and 9:"));

  if(num >= 0&& num <= 9&& !numbers.includes(num)) {
    numbers.push(num);
  } else{
    alert("Invalid input! Please enter a unique number between 0 and 9.");
  }
}
```

If you want to allow NaN as a valid input in the JavaScript code, you can modify the code as follows:

```
varnumbers = [];
while(numbers.length< 5) {
  varnum = parseFloat(prompt("Enter a unique number between 0 and 9, or NaN:"));

  if((num >= 0&& num <= 9&& !numbers.includes(num)) || isNaN(num)) {
    numbers.push(num);
  } else{
    alert("Invalid input! Please enter a unique number between 0 and 9, or NaN.");
  }
}
```

console.log("The unique numbers you entered are: "+ numbers.join(", "));	console.log("The unique numbers you entered are: "+ numbers.join(", "));
In this code, a while loop is used to repeatedly prompt the user for input until 5 unique numbers have been entered. The parseInt function is used to convert the input to an integer. The if statement checks if the entered number is between 0 and 9 and not already in the numbers array. If the input is valid, it's added to the numbers array. If it's not valid, an alert is displayed to the user. Finally, the unique numbers entered by the user are printed to the console. Please note that this code should be run in an environment that supports the prompt and alert functions, such as a web browser. To allow floating point numbers, you would need to use the parseFloat function instead of parseInt. Here's how you can modify the code:	In this code, parseFloat is used to convert the input to a floating-point number. The if statement checks if the entered number is between 0 and 9 and not already in the numbers array, or if the entered value is NaN. If the input is valid, it's added to the numbers array. If it's not valid, an alert is displayed to the user. Finally, the unique numbers entered by the user are printed to the console. Please note that this code should be run in an environment that supports the prompt and alert functions, such as a web browser. Also, keep in mind that JavaScript uses binary floating-point arithmetic, which can sometimes lead to unexpected results due to rounding errors. For example, the number 0.1 cannot be represented exactly as a base 2 fraction, so in binary floating-point, the result is an approximation to the true value. This is something to be aware of when working with floating-point numbers in JavaScript. If the user enters NaN, it will be treated as a valid input and added to the numbers array. The isNaN() function is used to check if a value is NaN. For example, isNaN(num) would return true if num is NaN, and false otherwise.

Comments of Interest from other Players

use the proper values/order.

Keep exploring the islands and solving other challenges. Sometimes you'll get a hint for a challenge somewhere else.

The vulnerability is in Python 3.11.6 version, as shown below, but it has been noted that this may not be a true vulnerability given certain functions (like sorted) mess up when sorting nan due to the way it throws off comparisons.

```
└─$ python3
Python 3.11.6 (main, Oct  8 2023, 05:06:43) [GCC 13.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
 float('nan')
nan
```

There is one that leads to an article specific to a vulnerability in the programming language for the deck of cards used by the elf.

There are several common errors that can occur when developing a card game in Python. Here are some examples:

- Incorrect Looping: If you're using loops to manage game rounds, you might encounter issues if the loop isn't set up correctly. For instance, a loop might stop immediately after displaying how many cards are left1.
- Attribute Errors: These occur when you try to access an attribute or method that doesn't exist for an object. For example, if you're trying to access a 'rank' attribute of a 'Hand' object, but the 'Hand' class doesn't have a 'rank' attribute, you'll get an AttributeError2.
- Input Errors: If your game involves user input, you need to handle cases where the user might enter something unexpected. For instance, if you're asking the user to pick a number of cards, you should check that the input is within the expected range1.
- Indentation Errors: Python uses indentation to determine the grouping of statements. Incorrect indentation can lead to unexpected behavior. For example, a return statement might be incorrectly indented, causing a function to exit prematurely3.
- Type Errors: These occur when an operation or function is applied to an object of an inappropriate type. For example, trying to concatenate a string and an integer would result in a TypeError.

Naan bread is my favourite number!

- So I've read the hint. Am I using NaN directly in the card game, or should I open up dev tools and mess with the console to ||change variables?
- omg, that was stupid. I don't know how long I was looking at trying other stuff... the answer is staring me right in the face

it checks to make sure you input the right data type
ou can dump most of the background code by sending it some bogus input and looking at the error response

In Python, NaN stands for 'Not a Number'. It is a special floating-point value that represents missing or undefined data1. You can create a NaN value in Python using the float function with the argument as "nan", "Nan", "NaN", or "NAN"2.

You can check for NaN values using the math.isnan() function from the built-in math module34. If you're working with arrays or matrices, the numpy.isnan() function can be a nice tool as it operates element-wise on an array and returns a Boolean array of the same shape4.

```
---------------------------------------------------------------------
import math

# Initialize cards
cards_player = [0, 1, 2, 3, 4]
cards_opponent = [5, 6, 7, 8, 9]
```

```
# Interject 'nan' into the game
cards_player[2] = float('nan')
cards_opponent[3] = float('nan')

# Function to compare cards
def compare_cards(cards1, cards2):
    player_score = 0
    opponent_score = 0
    for card1, card2 in zip(cards1, cards2):
        if math.isnan(card1) or math.isnan(card2):
            print("One of the cards is not a number!")
            continue
        elif card1 > card2:
            player_score += 1
        elif card1 < card2:
            opponent_score += 1
    return player_score, opponent_score

# Compare cards and get scores
player_score, opponent_score = compare_cards(cards_player, cards_opponent)

print("Player score:", player_score)
print("Opponent score:", opponent_score)
```

In this example, the compare_cards function skips any round where either player has a nan card. This is because nan cannot be meaningfully compared to other numbers. If you want to handle nan differently (for example, by treating it as the highest or lowest possible card), you would need to modify the comparison logic accordingly. Please note that this is a simple example and real-world card games would likely have more complex rules and structures

From <https://www.bing.com/search?q=Certipy+command+to+enumerate+and+attack+Active+Directory+Certificate+Services+%28AD+CS%29&qs=n&form=QBRE&sp=-1&lq=1&pq=&sc=0-0&sk=&cvid=5AA7DA3AFD584196B188EF4A08B3728E&ghsh=0&ghacc=0&ghpl=&showconv=1>