CMP 201 – DATA STRUCTURES AND ALGORITHMS 1

Tia [redacted] - [redacted]

PROBLEM

- The fictional company ‘Phones2U’ have had a data breach in which names and passwords were leaked

- The Company has asked customers to check if their names and passwords were breached in the attack.

- They have asked to have two programs built using different algorithms to check the databases.

PROBLEM

- The database of names is around 70KB
- The database of passwords is around 133MB
- The algorithms need to be able to find the given name and password in both databases.

SOLUTION

- The algorithms will need to be able to search for a given username and password (pattern)
- The algorithms will need to be able to search through large amounts of data and strings to find these patterns.
- The algorithm will also need to be as quick and efficient as possible.

ALGORITHMS

- The first algorithm that has been chosen, is the Boyer-Moore algorithm

- The second algorithm that has been chosen, is the Rabin-Karp algorithm

- They are both string searching algorithms but use different methods of finding patterns within text.

BIG O NOTATION – RABIN-KARP

- Average and best running time for Rabin Karp is O(n+m)

- Worst case running time is O(nm).

- The worst case running time is caused when characters in the pattern and the text are the same hash values.
- Username is Aaron but the next name in the list is Aaren would be an example of O(nm).

BIG O NOTATION – BOYER MOORE

- Best case running time is O(n/m).

- Worst case running time is O(mn).

- Occurs when the text and the pattern characters are the same, for example "cwmcarn", "cwmcarnsophie", "cwmch17" where the password pattern is "cwmch17".

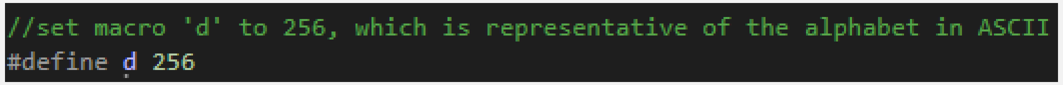
ABSTRACT DATA TYPES USED - MACROS

- Used in place of a constant integer

- Constant int stored in memory, so need to access memory each time variable is needed. This would make the program much slower.

- Macro is a fragment of code that is set when the program is compiled, it doesn’t need to access memory each time the variable is needed.

- This means that the program runs more efficiently and quicker.



```
//set macro 'd' to 256, which is representative of the alphabet in ASCII
#define d 256
```

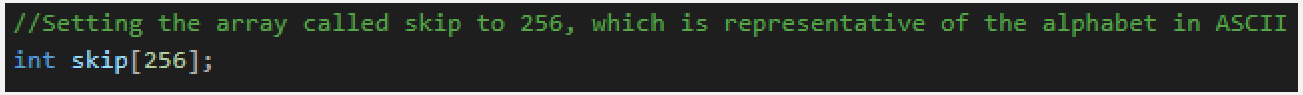
ABSTRACT DATA TYPES USED

- I have decided to use arrays and vectors within the programs.

- I was going to add extra ADT such as position and list, however there was no need for them and it would have caused the program to be less efficient.

ABSTRACT DATA TYPES USED - ARRAY

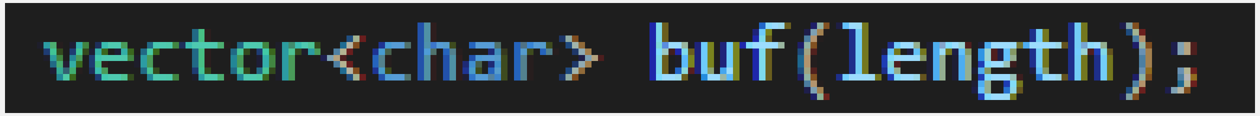
- The skip table variable is an array. This is the most suitable ADT for this requirement.
- The program is able to choose the position it needs to enter or retrieve data from the array.
- Because the array is being accessed in each skip, it makes sense to have an array as the time complexity is O(1).



```
//Setting the array called skip to 256, which is representative of the alphabet in ASCII
int skip[256];
```

ABSTRACT DATA TYPES USED - VECTORS

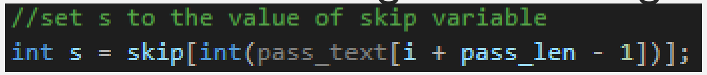
- A vector has been used for the function when opening the username and password files.
- The vector is a dynamic array which means it doesn't have a set size, which is why it's useful for storing the length of the text file!
- The vector being used within the functions have the time complexity of O(1).



```
vector<char> buf(length);
```

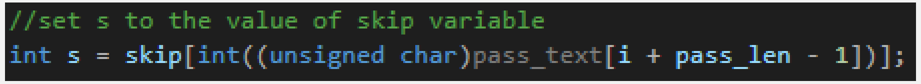
DEBUGGING AND FAULT TESTING – PROGRAM CRASHING

- Boyer Moore – putting in garbage data into password caused program to crash
- Error was occurring when setting s to skip variable.



```
//set s to the value of skip variable
int s = skip[int(pass_text[i + pass_len - 1])];
```

- Because some of the passwords contain symbols, some of the pass_text characters weren’t being represented as they were too big.



```
//set s to the value of skip variable
int s = skip[int((unsigned char)pass_text[i + pass_len - 1])];
```

- Using unsigned char meant that the value being written to memory is increased and can represent symbols.

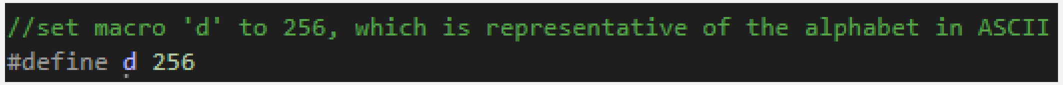
DEBUGGING AND FAULT TESTING - MACROS

- Used in place of a constant integer

- Constant int stored in memory, so need to access memory each time variable is needed. This would make the program much slower.

- Macro is a fragment of code that is set when the program is compiled, it doesn't need to access memory each time the variable is needed.

- This means that the program runs more efficiently and quicker.



```
//set macro 'd' to 256, which is representative of the alphabet in ASCII
#define d 256
```

EXPECTED RESULTS

- Before comparing the performance of the two algorithms, we need to look at the time complexities for both and how they will manage the task.
- Boyer Moore has a best case running time of O(n/m) and worst case running time of O(mn).
- Rabin Karp has an average and best running time of O(n+m) and a worst case running time of O(nm).
- Boyer Moore is likely to be the faster of the two and that it will be able to deal with matching characters better than rabin karp's matching hashes.

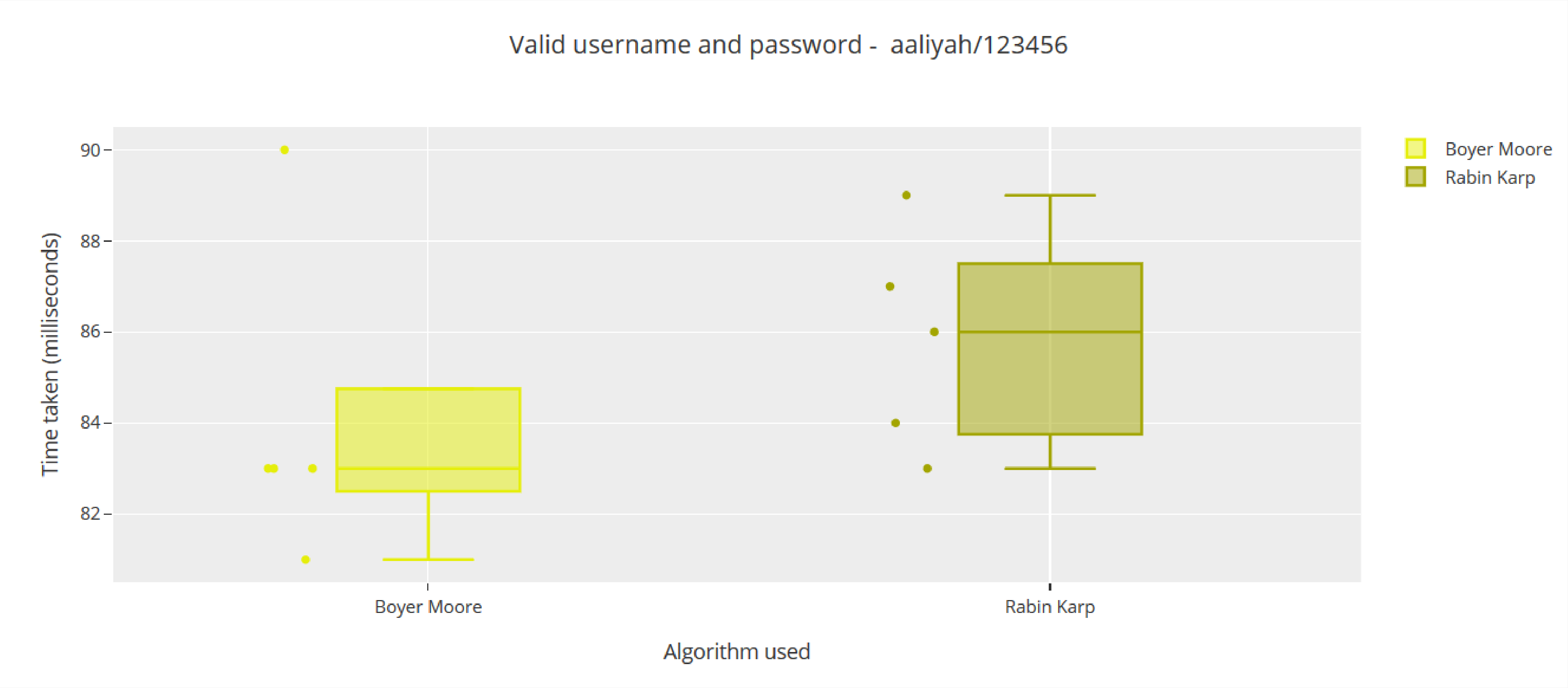
COMPARISONS

- The following graphs will inspect how quickly the algorithms will find:
- Valid username and password
- Valid username and invalid password
- Invalid username and valid password
- Invalid username and password

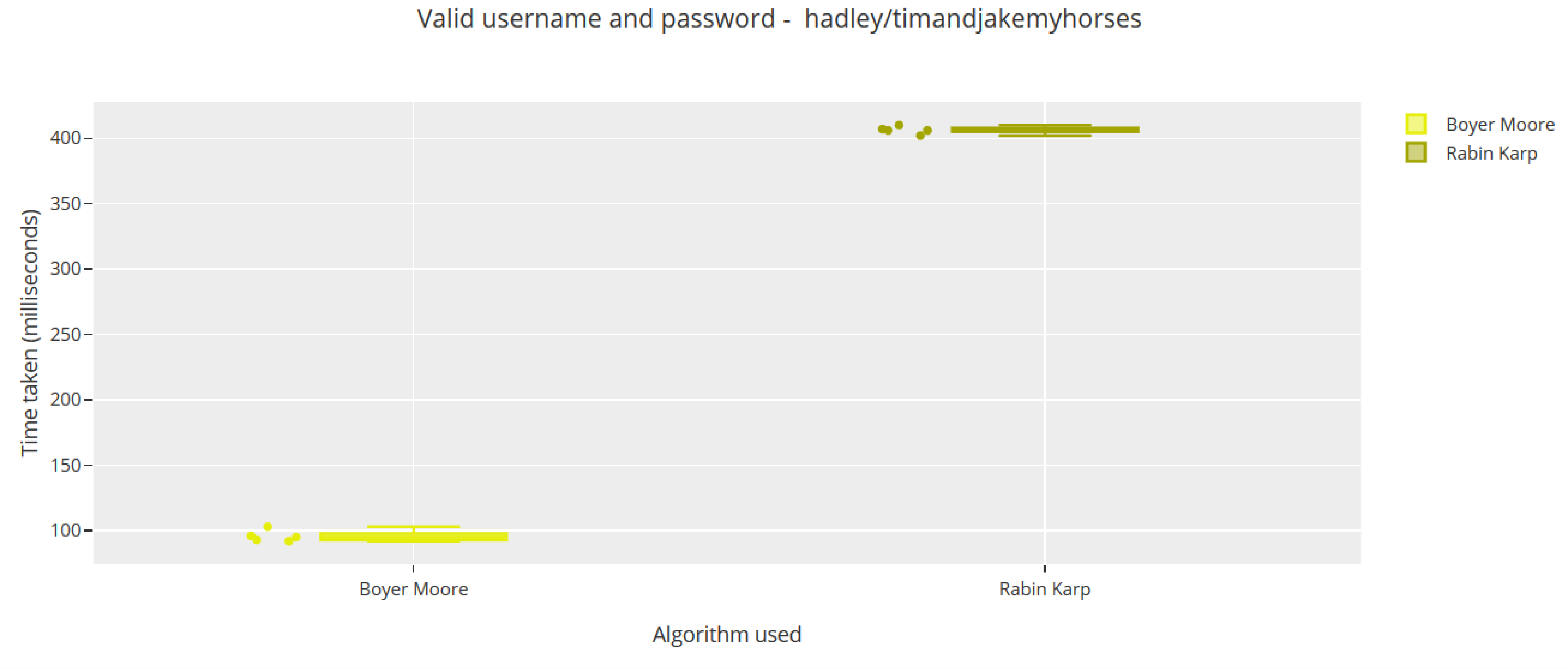
COMPARISONS – VALID USERNAME AND PASSWORD

- The username and passwords being used are:
- First entries in each file: “Aaliyah/123456”
- Entries at least ¼ of way into each file:“Hadley/timandjakemyhorses”
- Entries at least ¾ of way into each file:“Philippa/cwmcarnsophie”
- Last entries in each file:“Zylen/!”

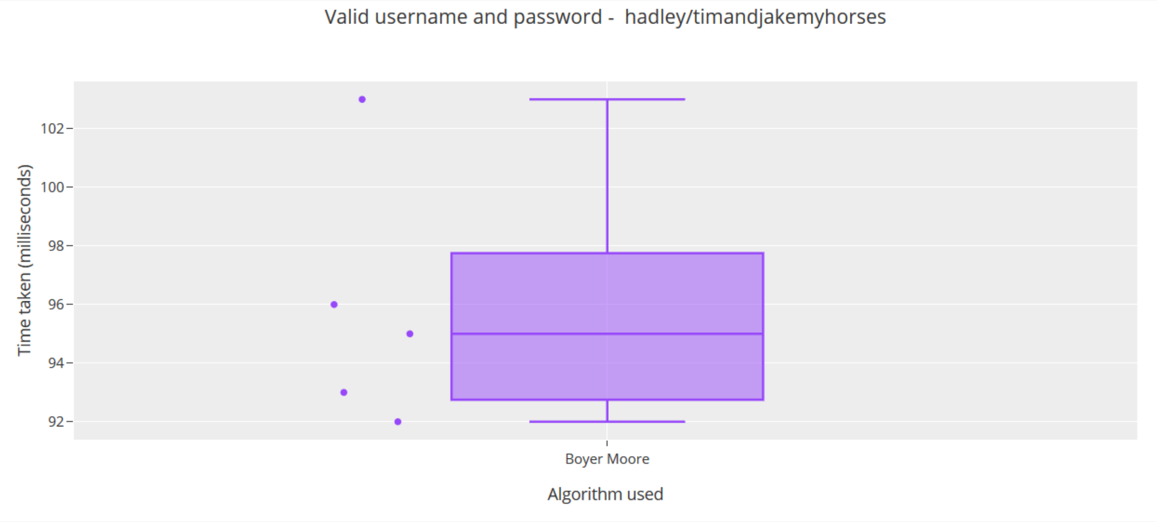
VALID USERNAME AND PASSWORD – AALIYAH/123456

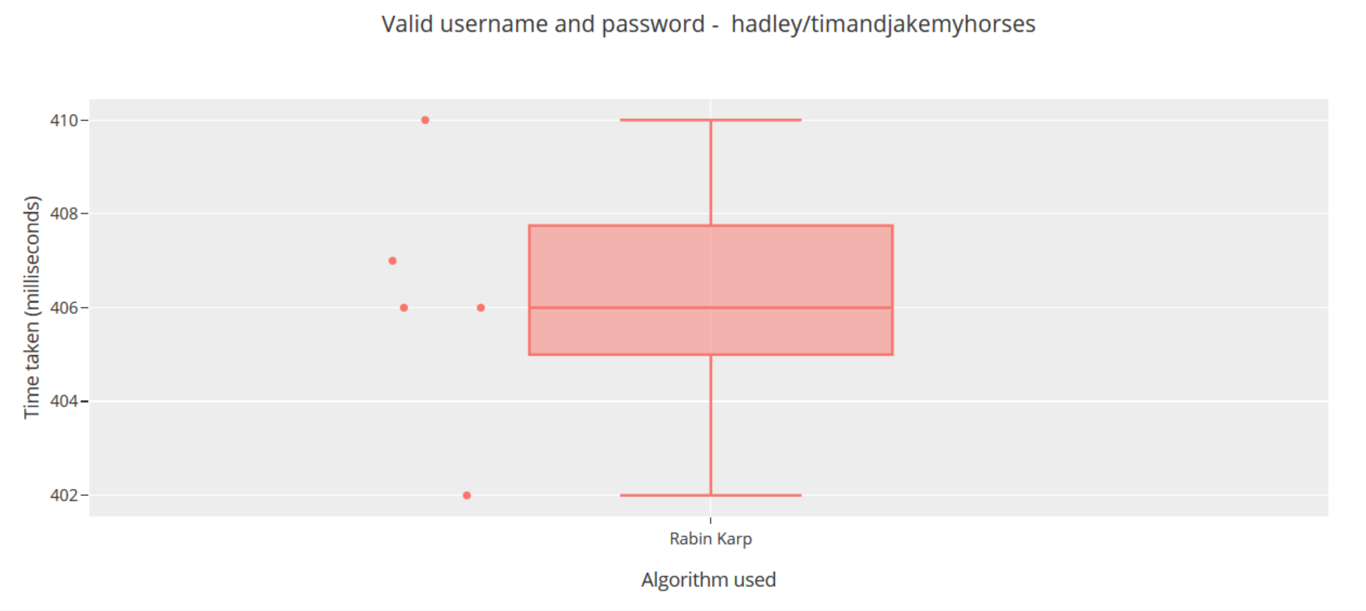


VALID USERNAME AND PASSWORD – HADLEY/TIMANDJAKEMYHORSES

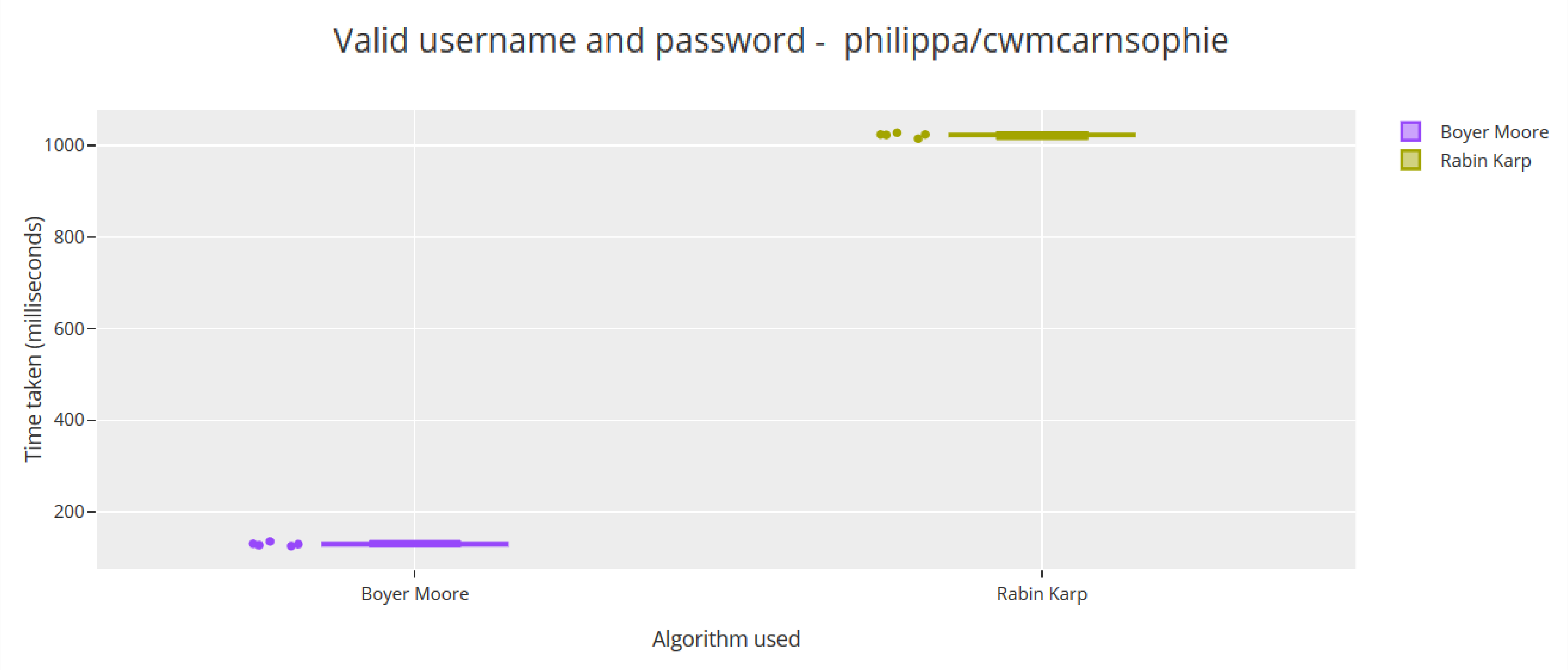


VALID USERNAME AND PASSWORD – HADLEY/TIMANDJAKEMYHORSES

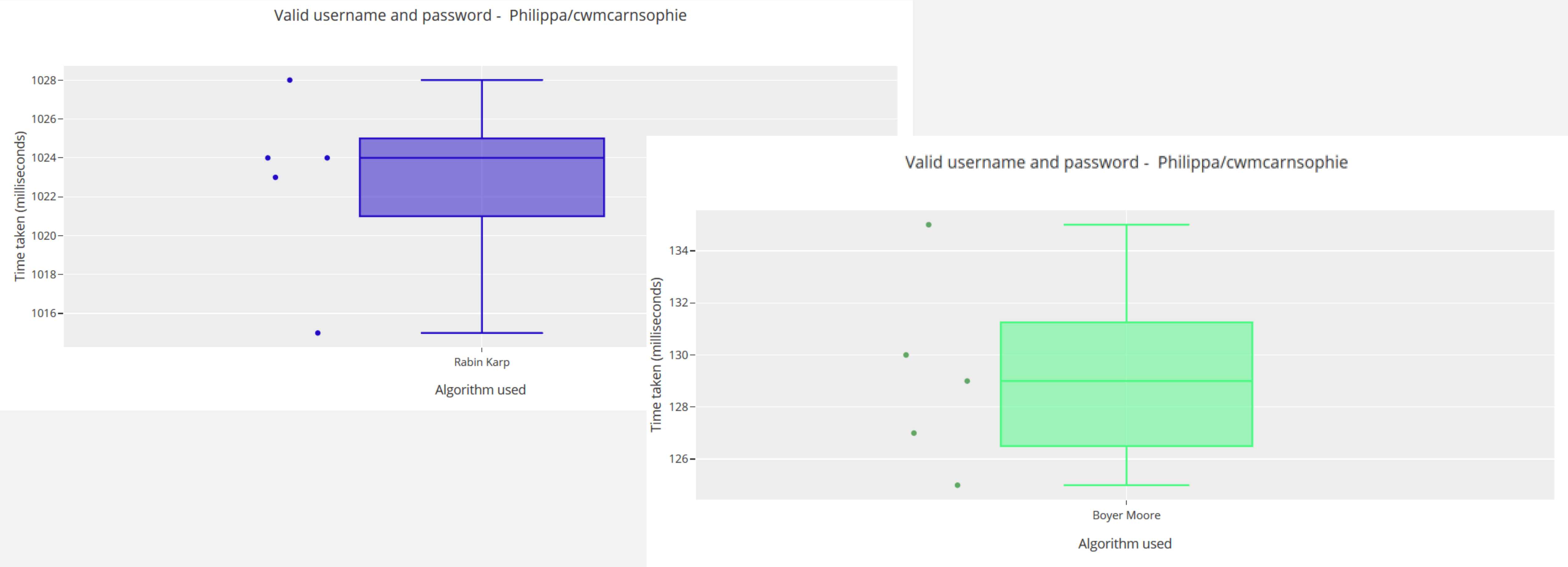




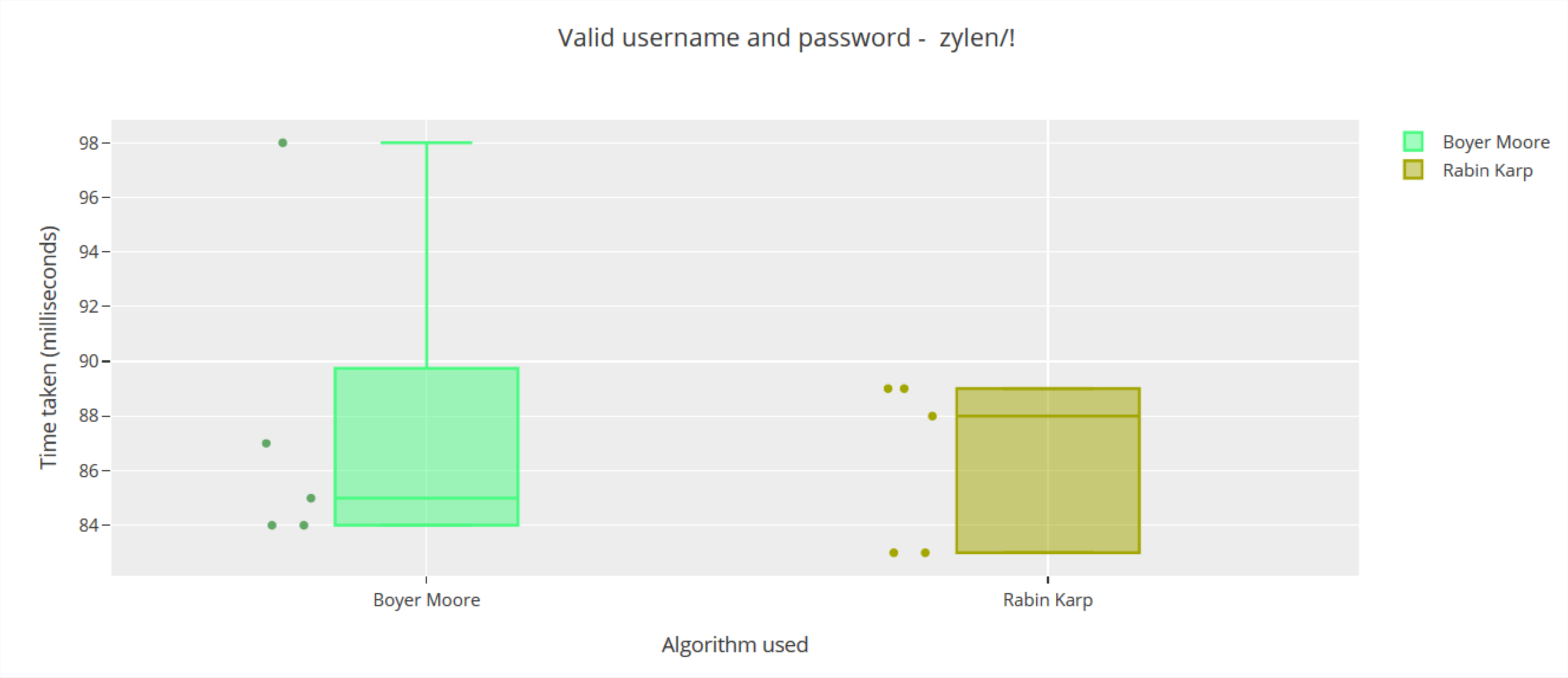
VALID USERNAME AND PASSWORD – PHILIPPA/CWMCARNSOPHIE



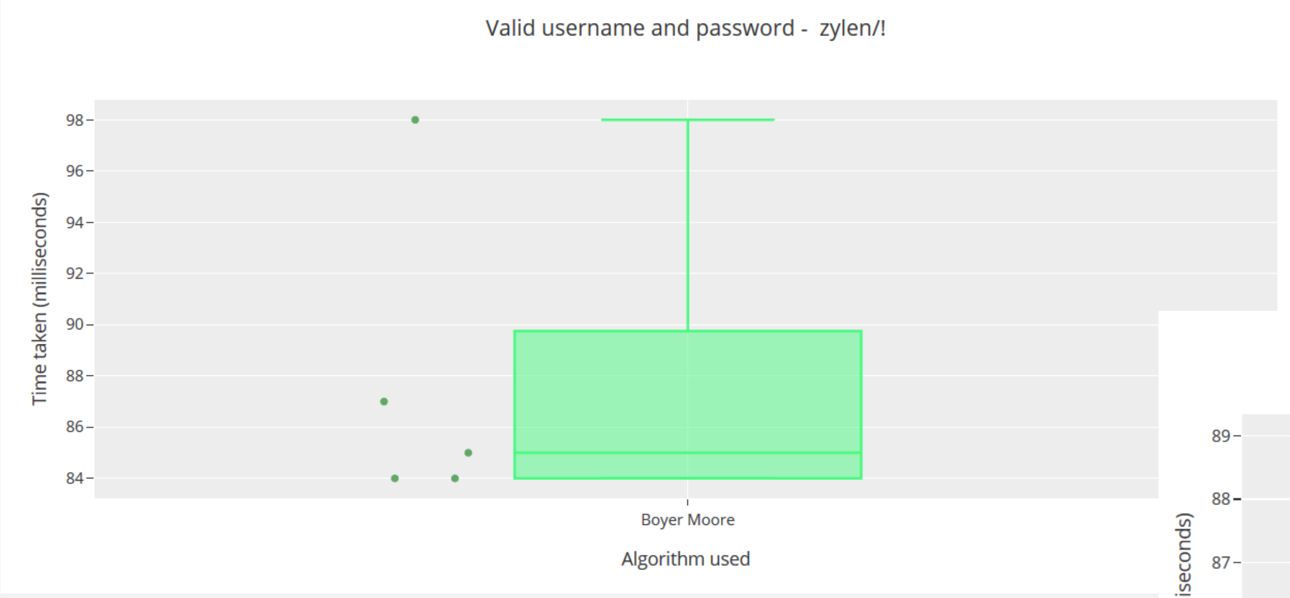
VALID USERNAME AND PASSWORD – PHILIPPA/CWMCARNSOPHIE

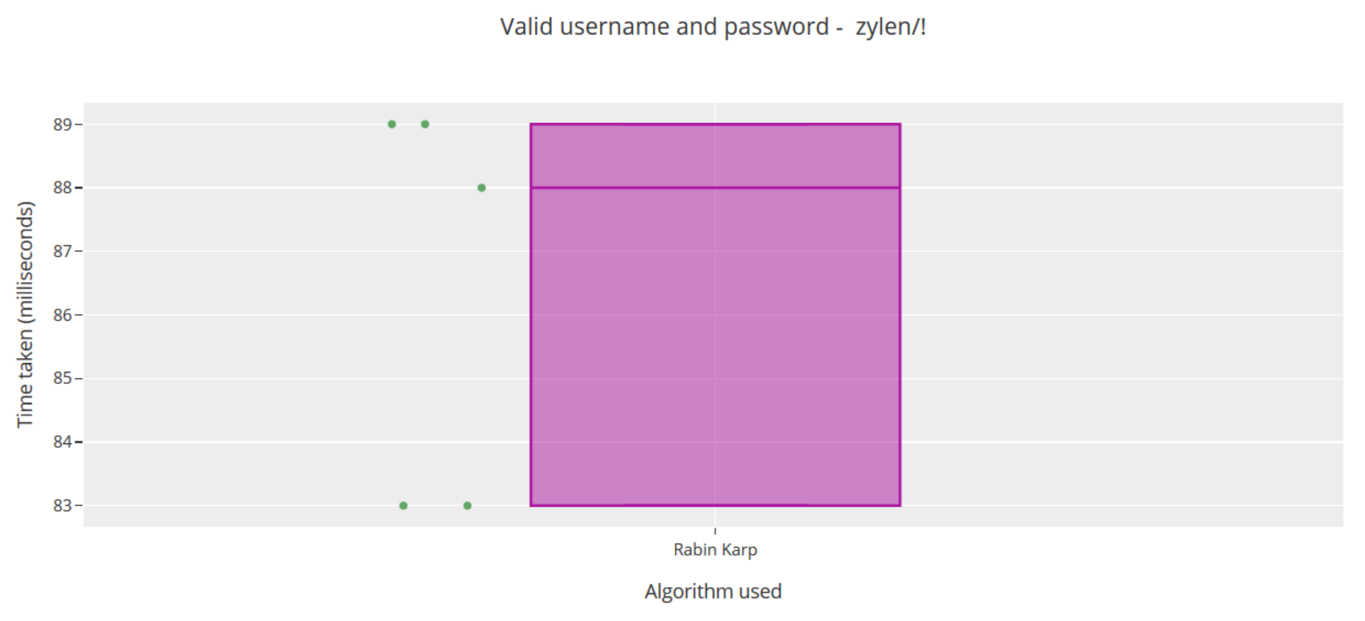


VALID USERNAME AND PASSWORD – ZYLEN/!

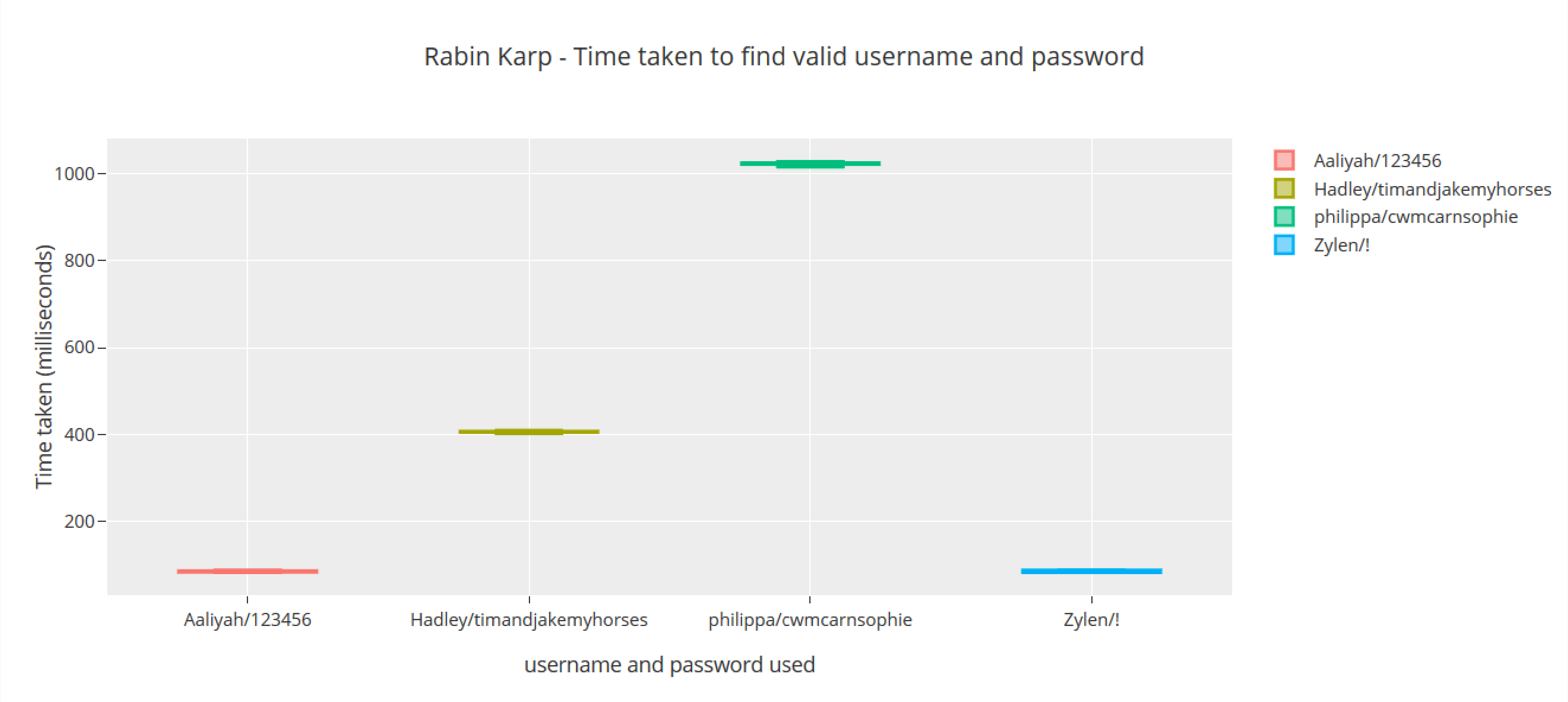


VALID USERNAME AND PASSWORD – ZYLEN/!

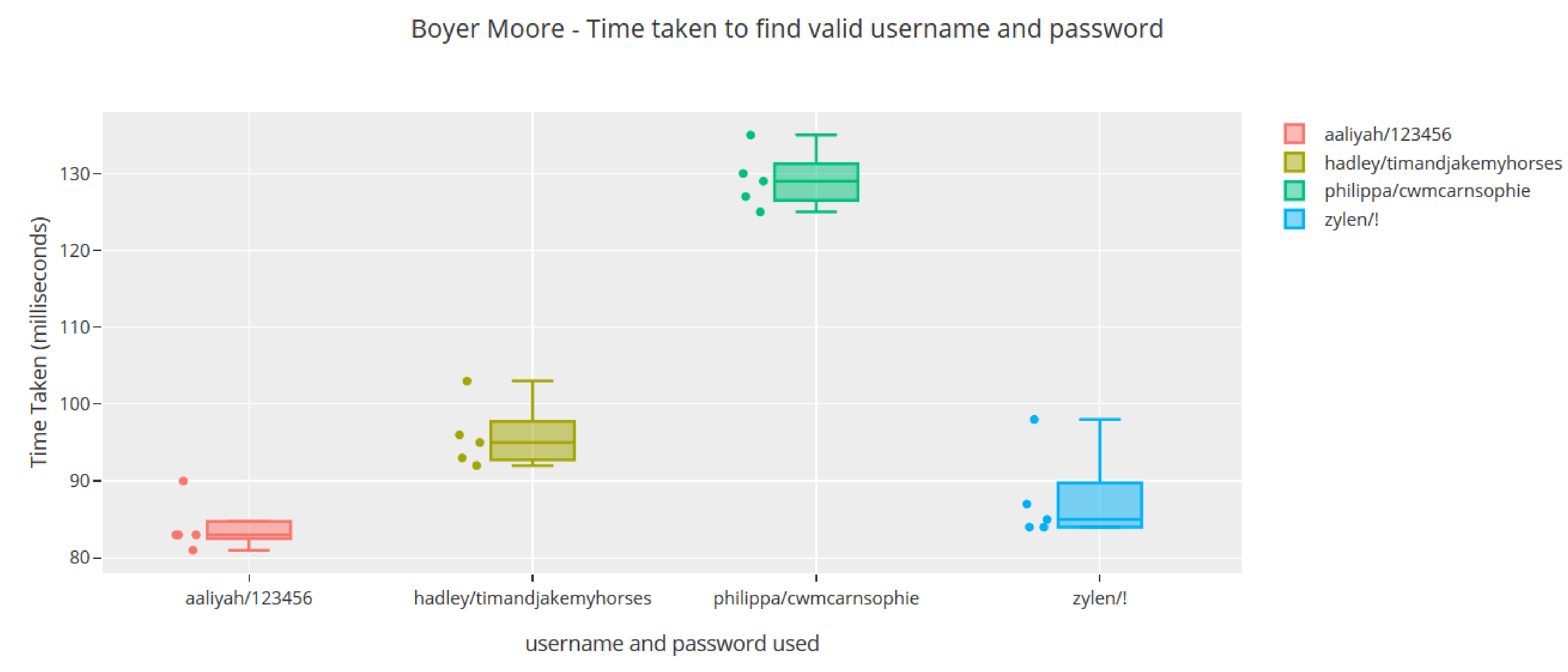




COMPARISONS – VALID USERNAME AND PASSWORD



COMPARISONS – VALID USERNAME AND PASSWORD

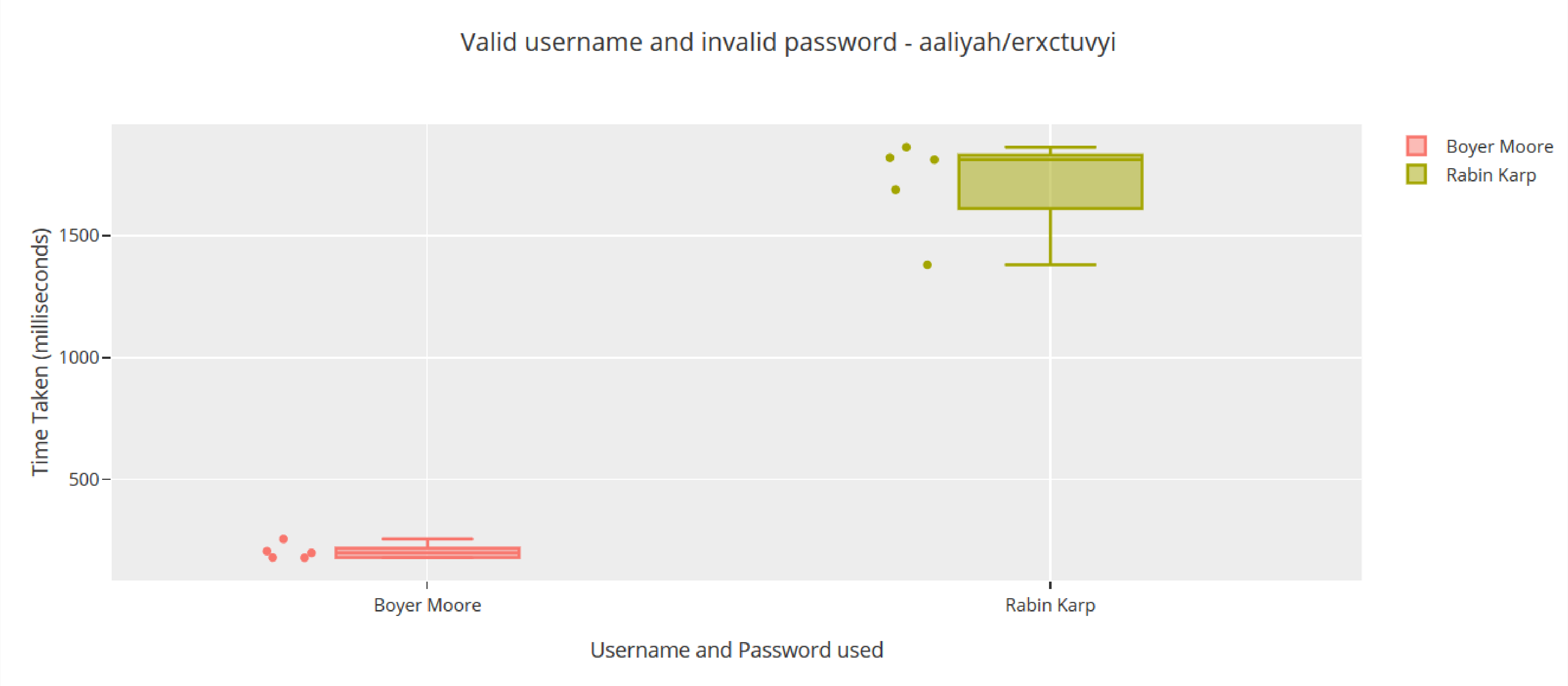


COMPARISONS – VALID USERNAME AND INVALID PASSWORD

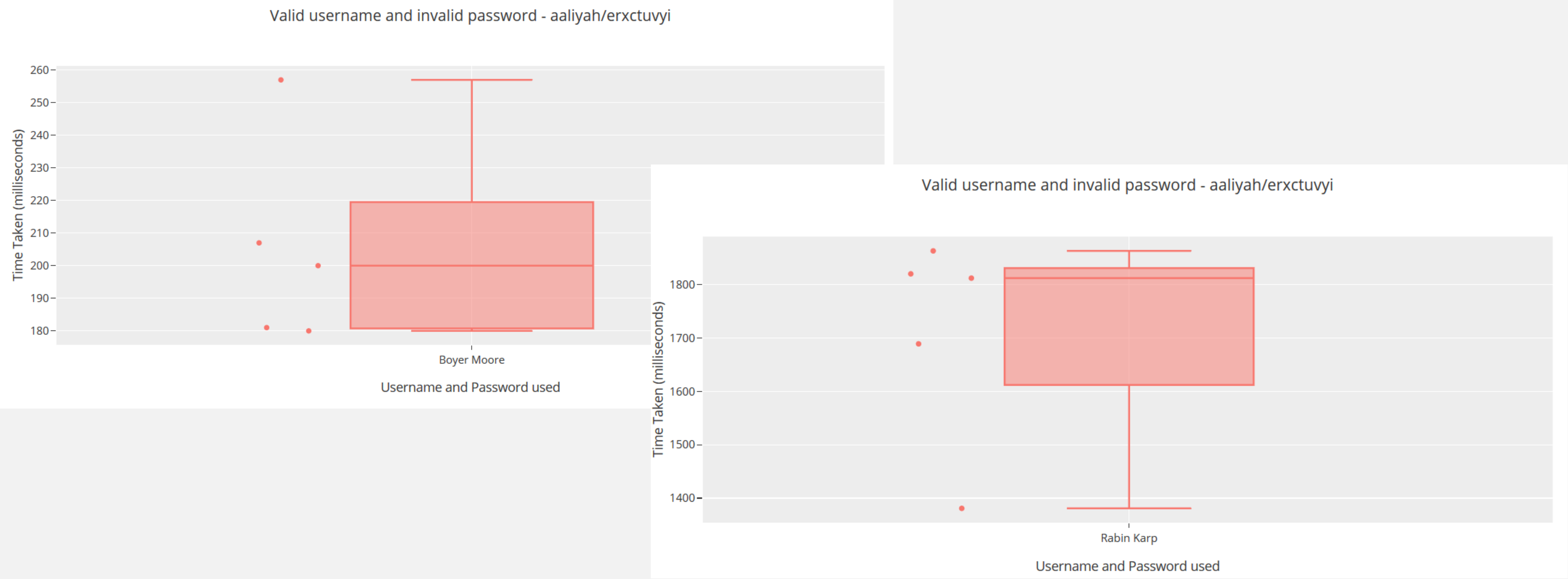
- The username and password's being used are:
- First entry in username file: "Aaliyah/ erxctuvyi"
- Entry at least ¼ of way into username file:"Hadley/ erxctuvyi"
- Entry at least ¾ of way into username file:"Philippa/ erxctuvyi"
- Last entry in username file:"Zylen/ erxctuvyi"

- For invalid entries, the following text will be used:"erxctuvyi" – It is junk text and is not in the password text file.

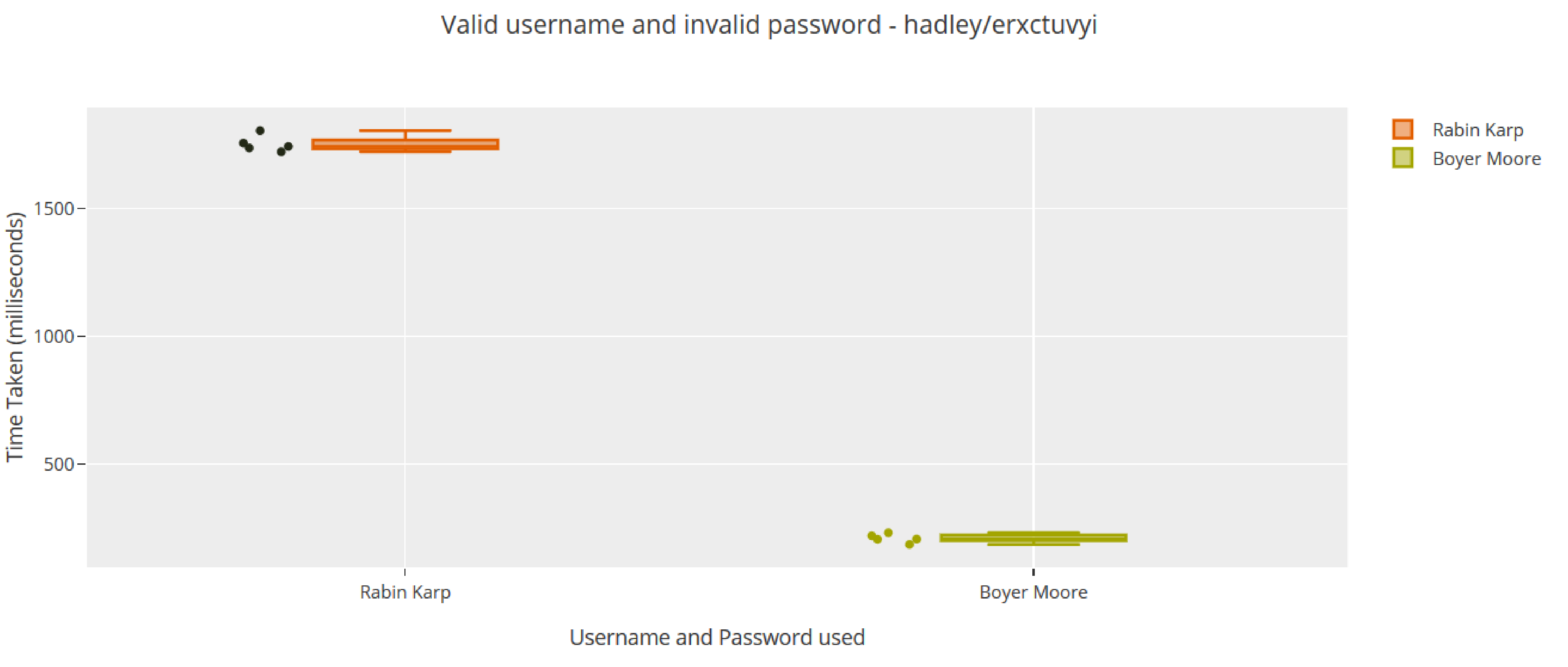
VALID USERNAME AND INVALID PASSWORD – AALIYAH/ERXCTUVYI



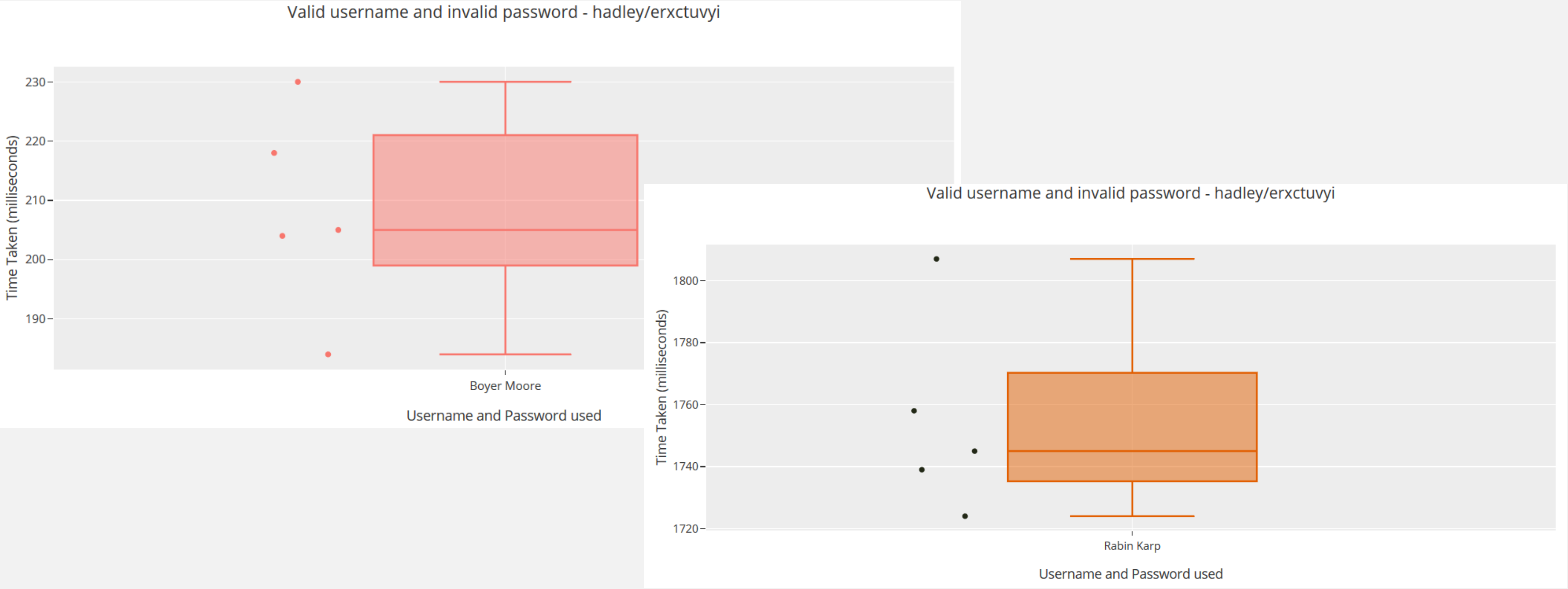
VALID USERNAME AND INVALID PASSWORD – AALIYAH/ERXCTUVYI



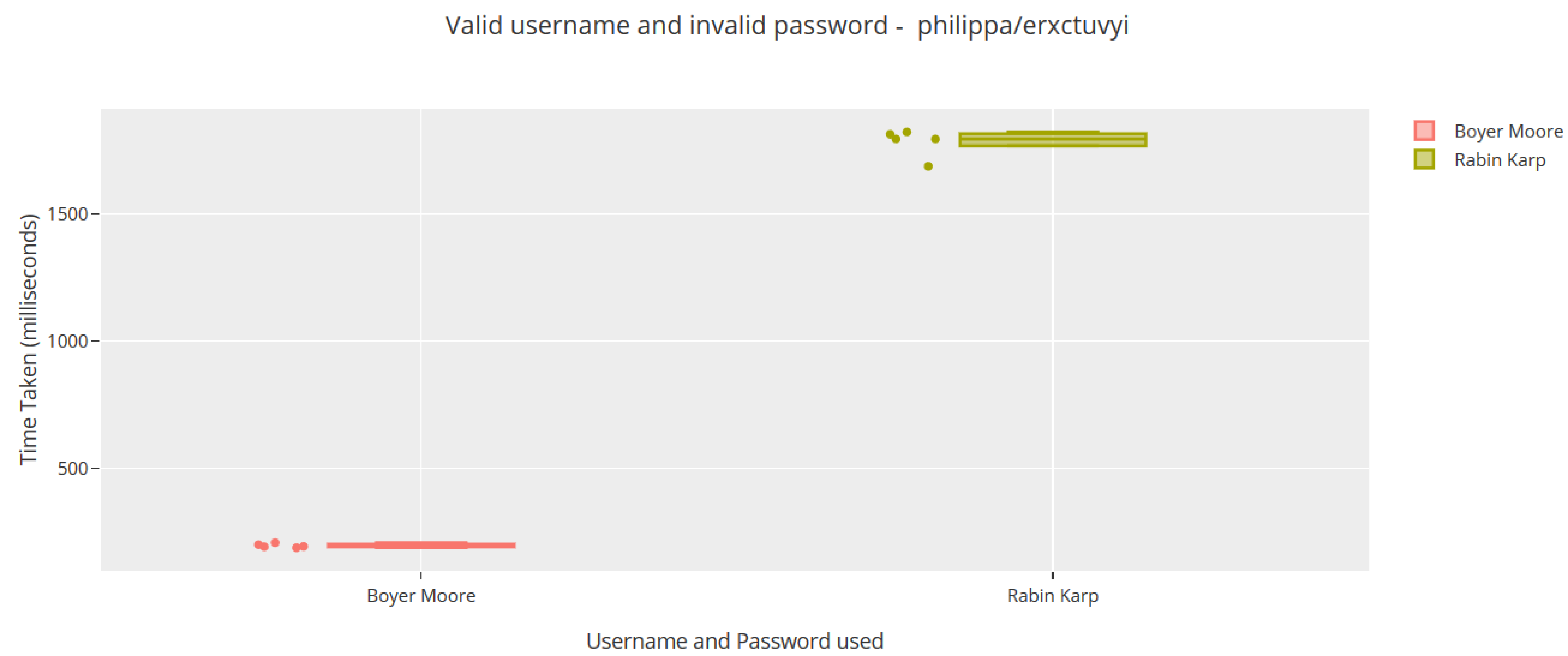
VALID USERNAME AND INVALID PASSWORD – HADLEY/ERXCTUVYI



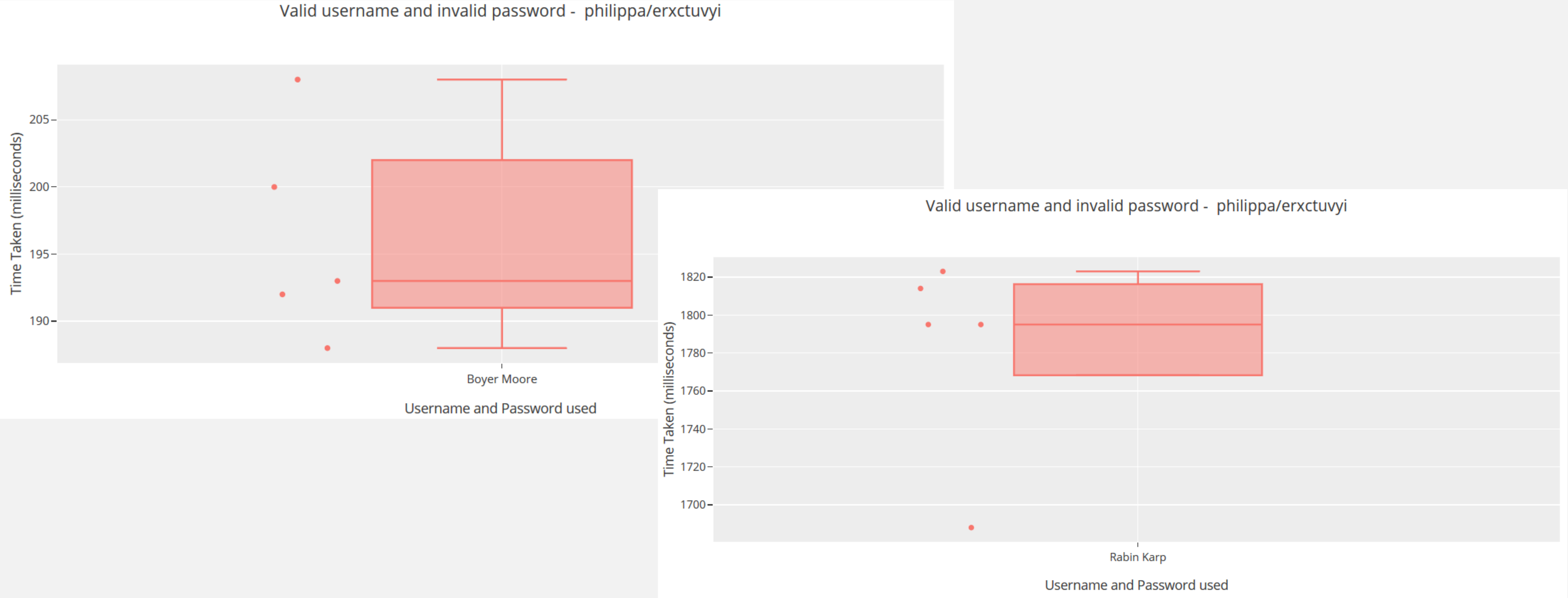
VALID USERNAME AND INVALID PASSWORD – HADLEY/ERXCTUVYI



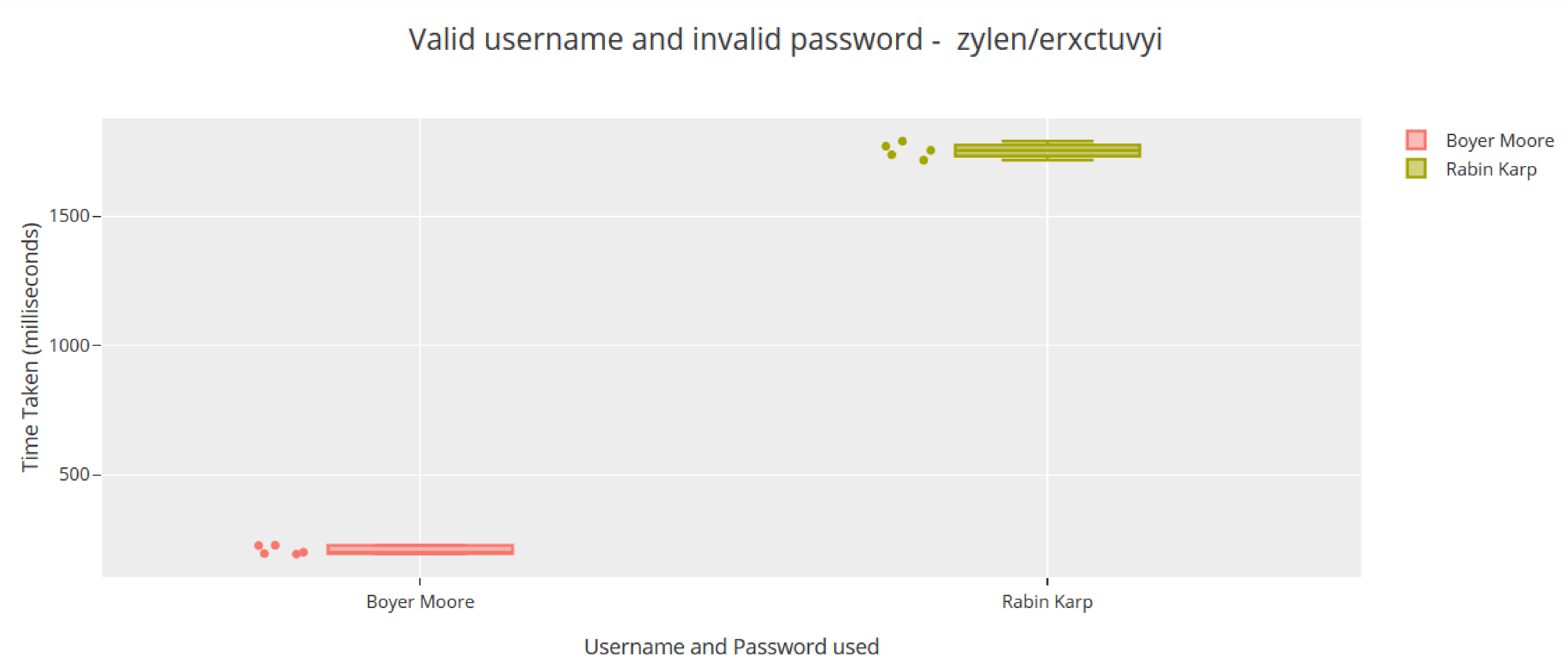
VALID USERNAME AND INVALID PASSWORD – PHILIPPA/ERXCTUVYI



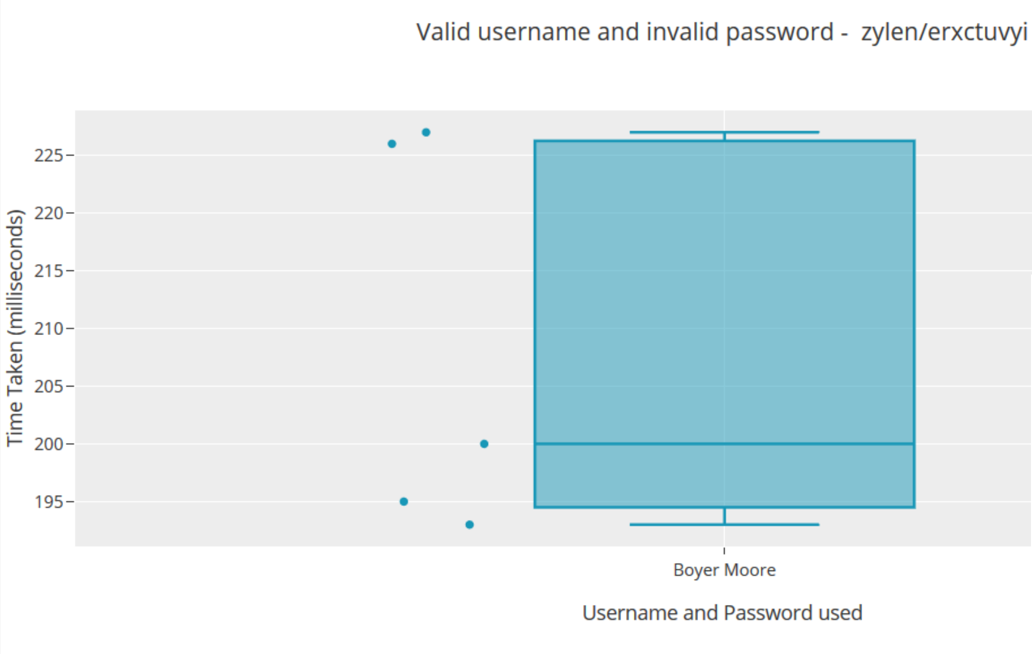
VALID USERNAME AND INVALID PASSWORD – PHILIPPA/ERXCTUVYI

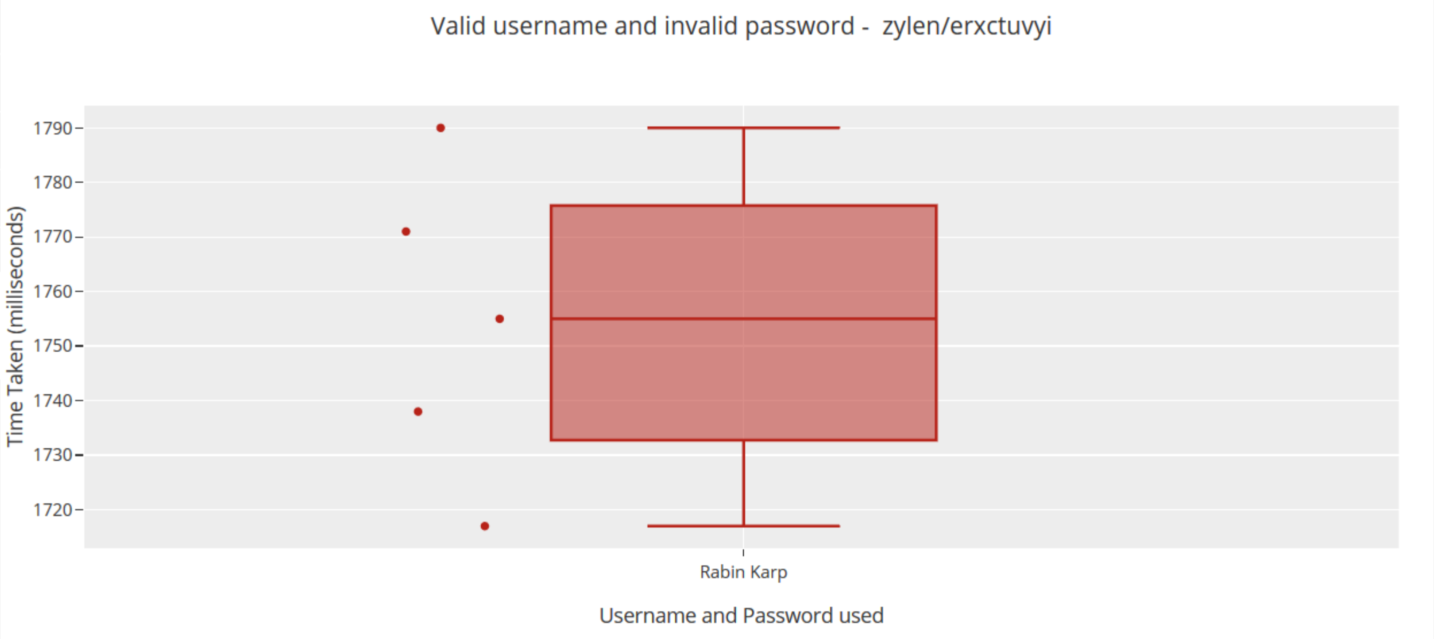


VALID USERNAME AND INVALID PASSWORD – ZYLEN/ERXCTUVYI

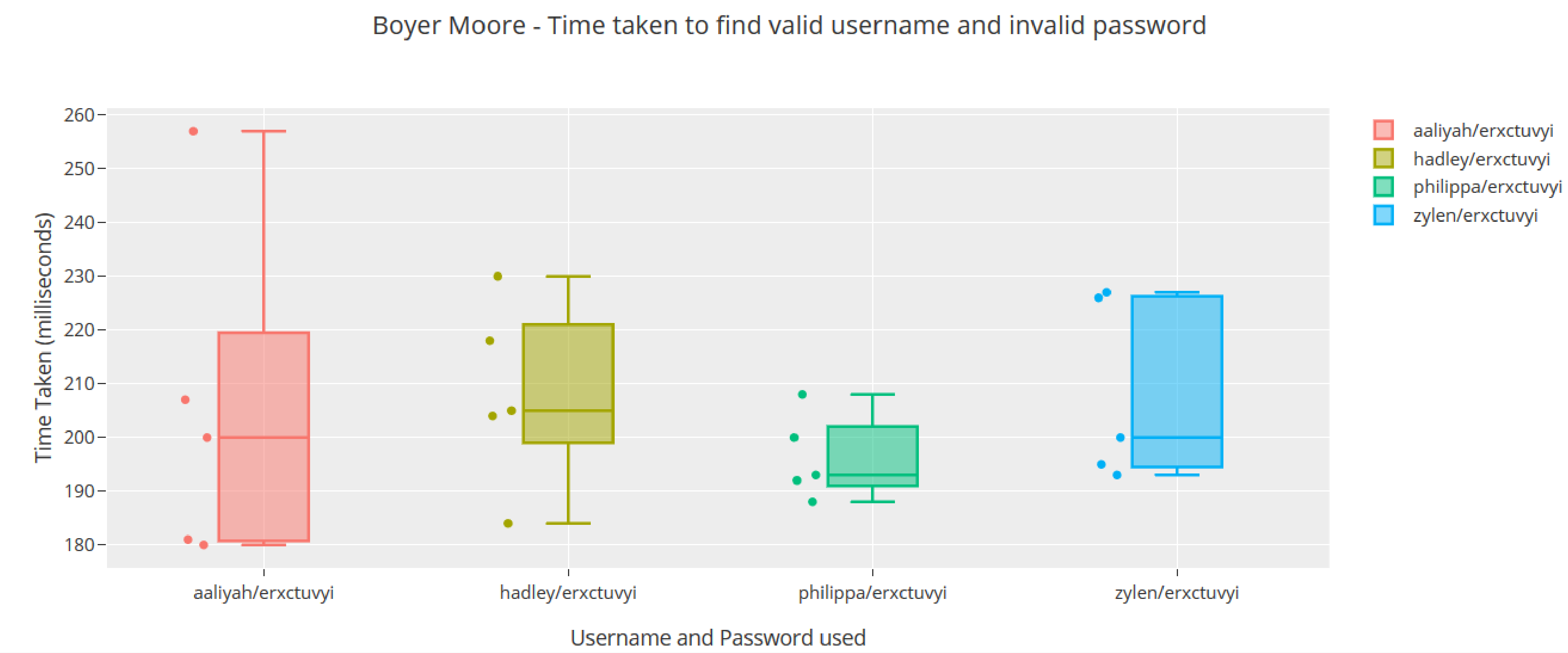


VALID USERNAME AND INVALID PASSWORD – ZYLEN/ERXCTUVYI

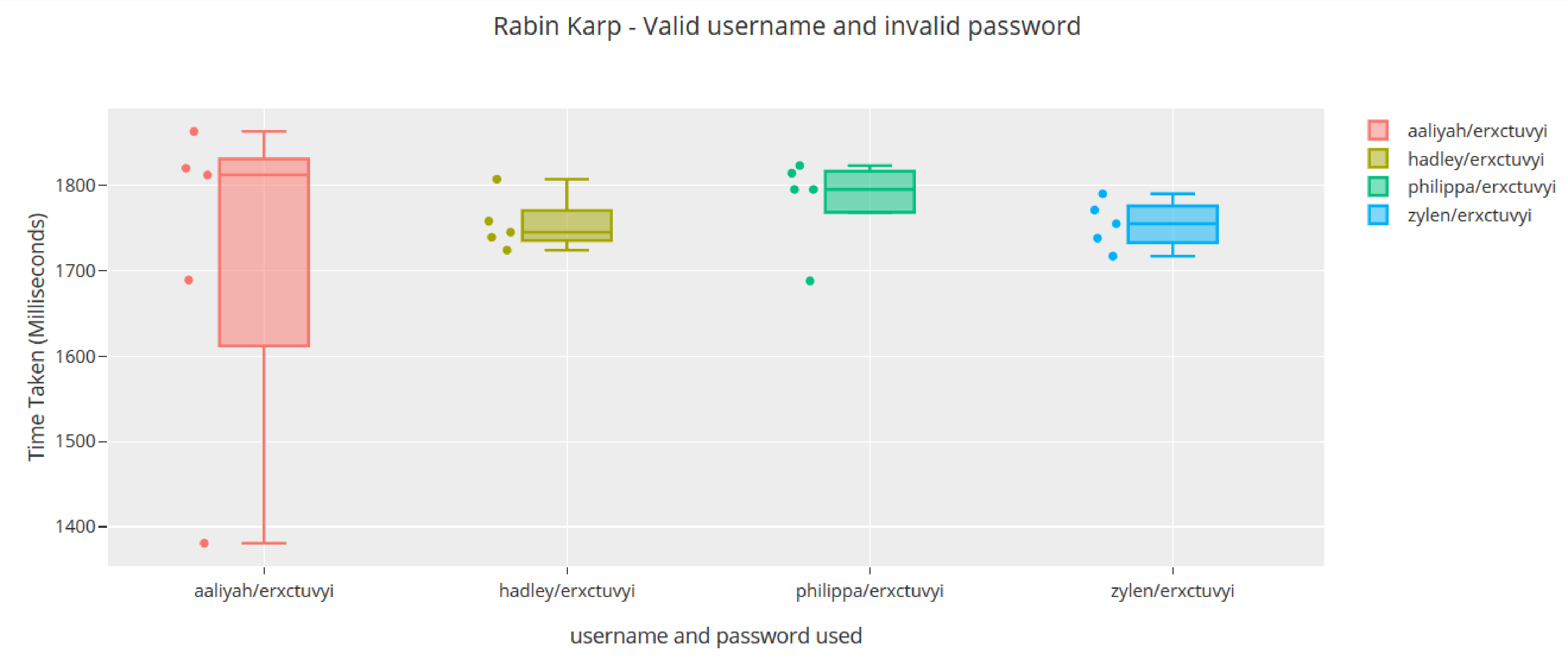




COMPARISONS – VALID USERNAME AND INVALID PASSWORD



COMPARISONS – VALID USERNAME AND INVALID PASSWORD

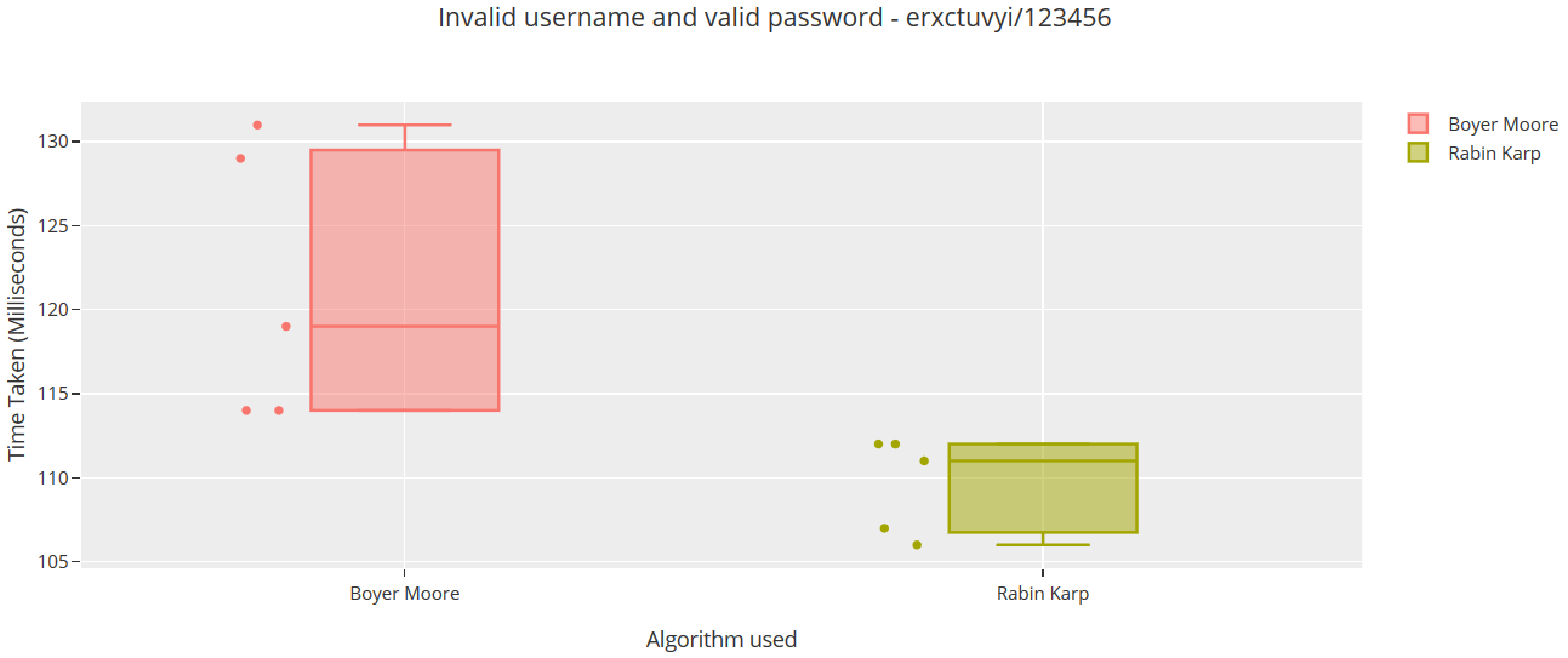


COMPARISONS – INVALID USERNAME AND VALID PASSWORD

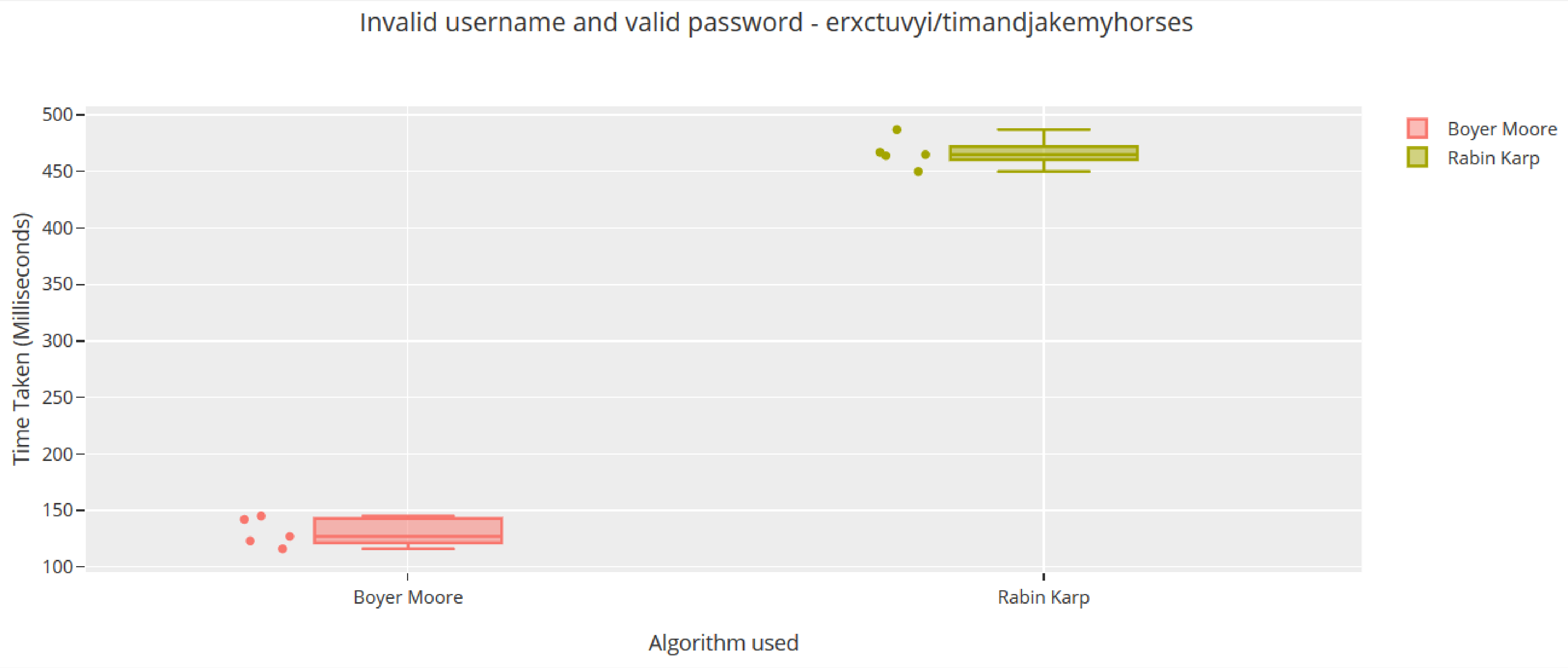
- The username and password’s being used are:
- First entry in passwors file: “erxctuvyi/123456”
- Entry at least ¼ of way into password file:“erxctuvyi/timandjakemyhorses”
- Entry at least ¾ of way into password file:“erxctuvyi/cwmcarnsophie”
- Last entry in password file:“erxctuvyi/!”

- For invalid entries, the following text will be used:“erxctuvyi” – It is junk text and is not in the username text file.

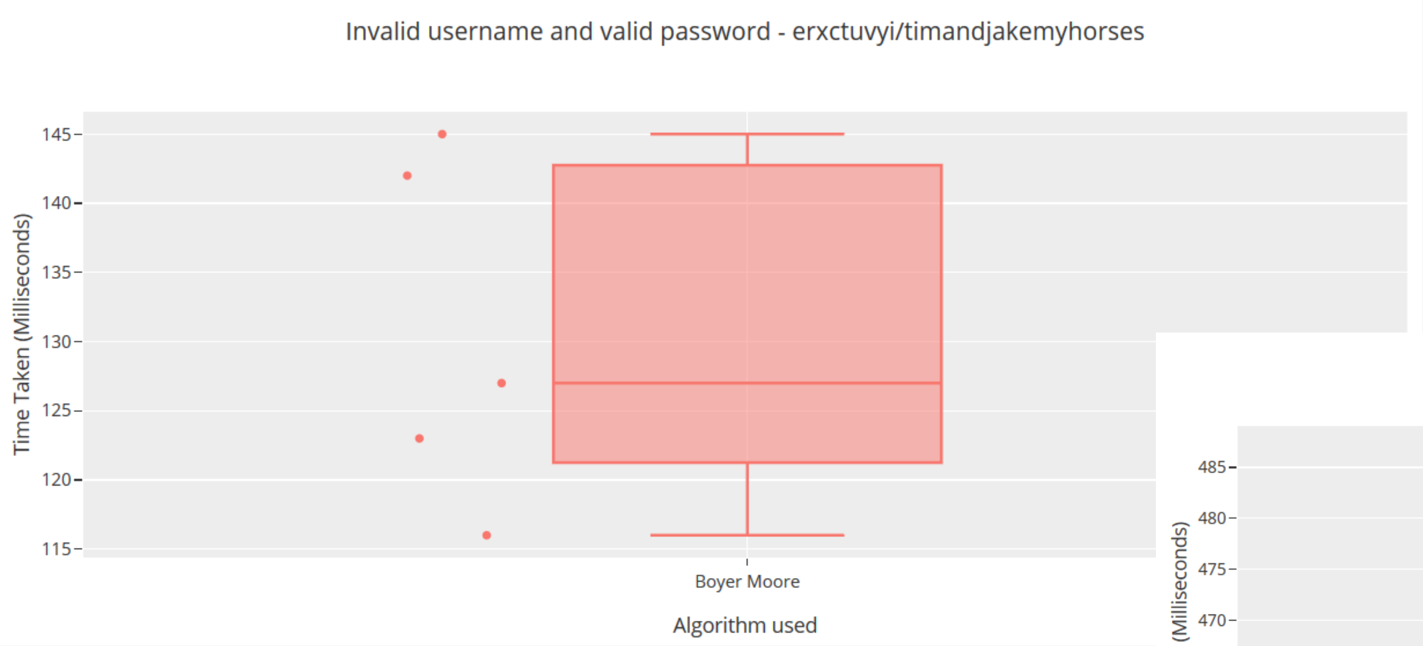
INVALID USERNAME AND VALID PASSWORD - ERXCTUVYI/123456

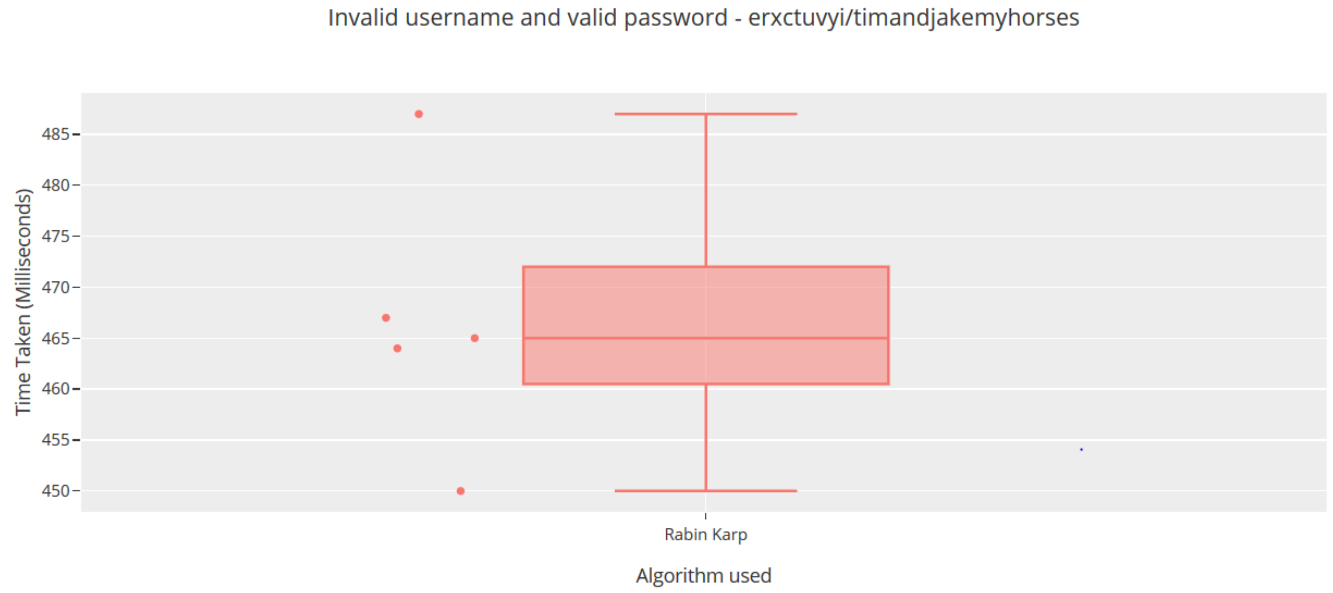


INVALID USERNAME AND VALID PASSWORD - ERXCTUVYI/TIMANDJAKEMYHORSES

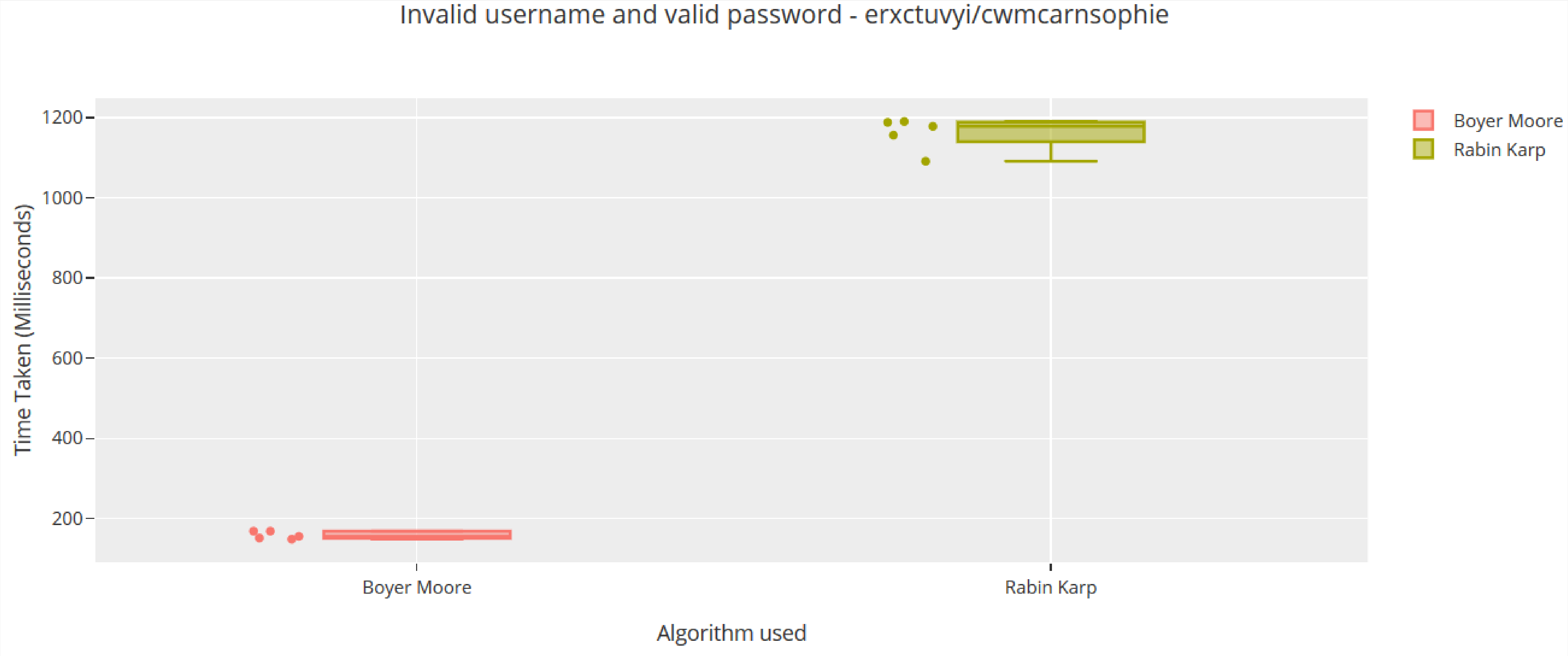


INVALID USERNAME AND VALID PASSWORD - ERXCTUVYI/TIMANDJAKEMYHORSES

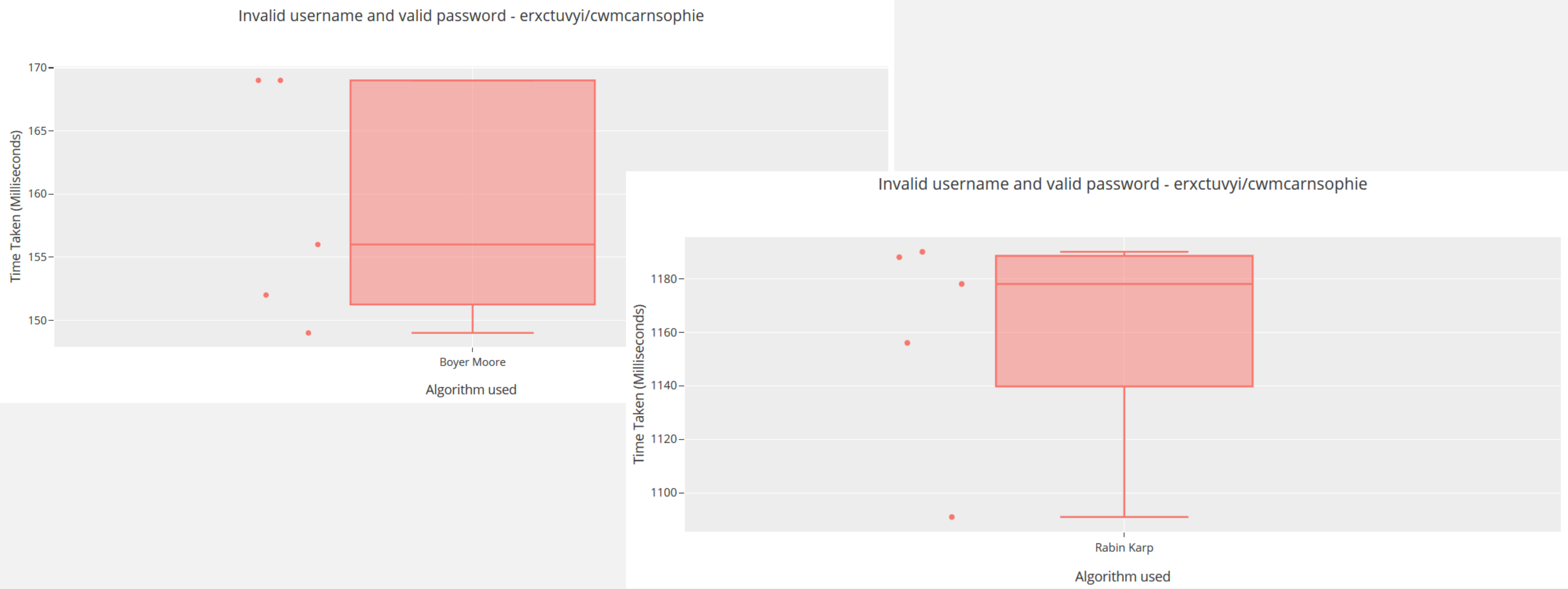




INVALID USERNAME AND VALID PASSWORD – ERXCTUVYI/CWMCARNSOPHIE



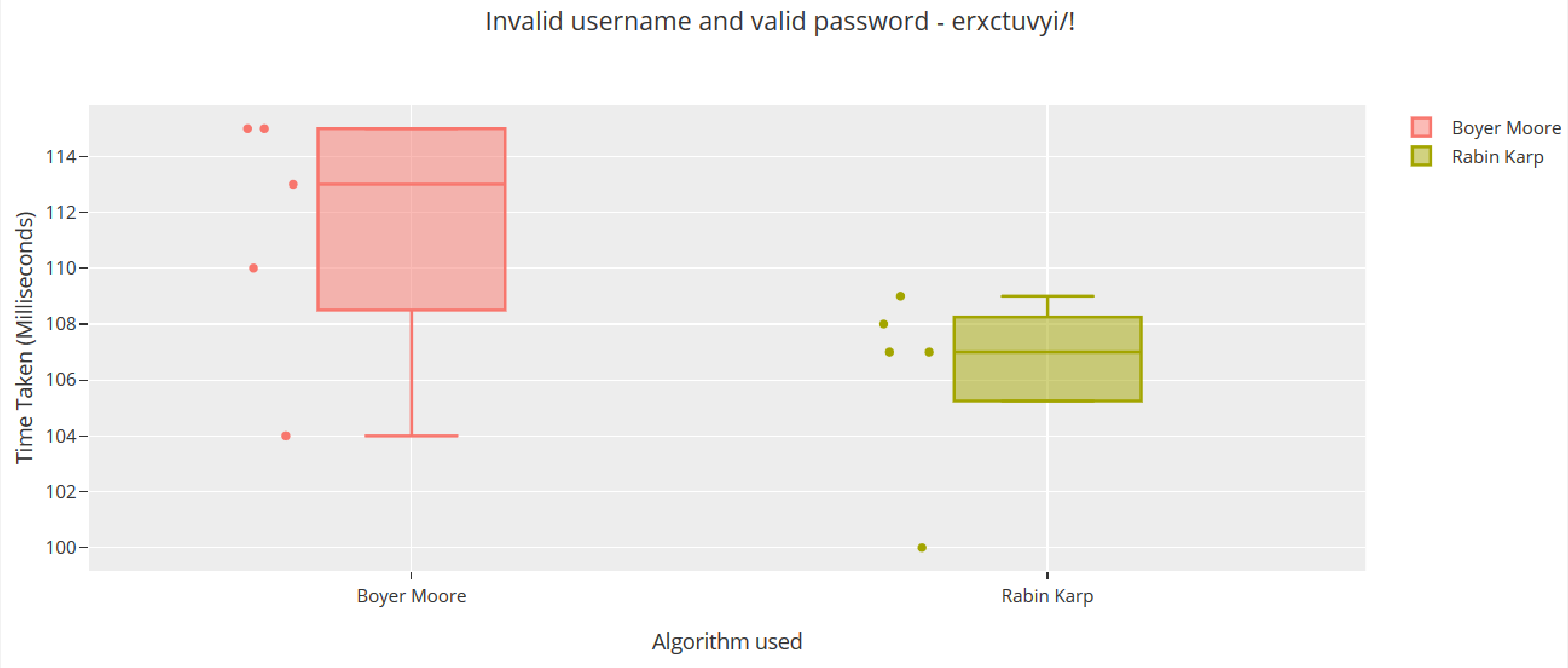
INVALID USERNAME AND VALID PASSWORD – ERXCTUVYI/CWMCARNSOPHIE



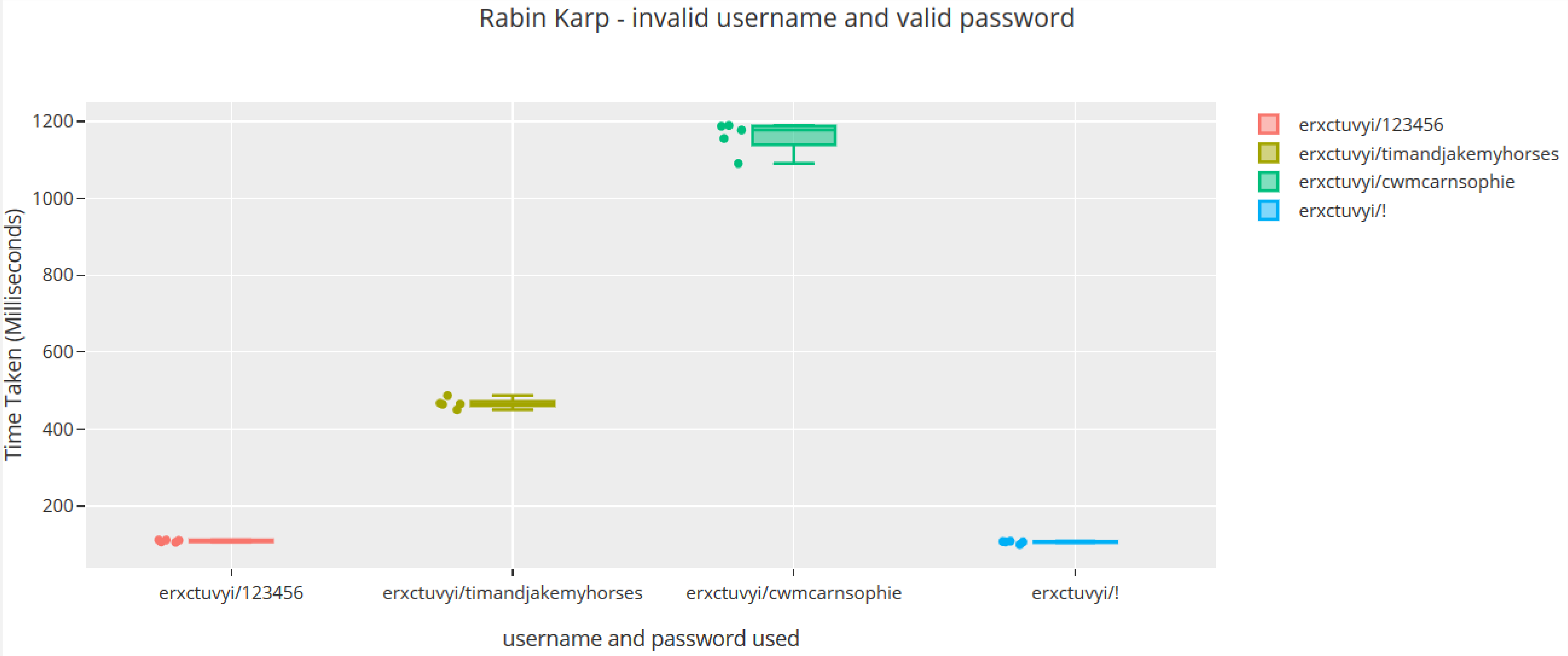
Invalid username and valid password - erxctuvyi/cwmcarnsophie

Invalid username and valid password - erxctuvyi/cwmcarnsophie

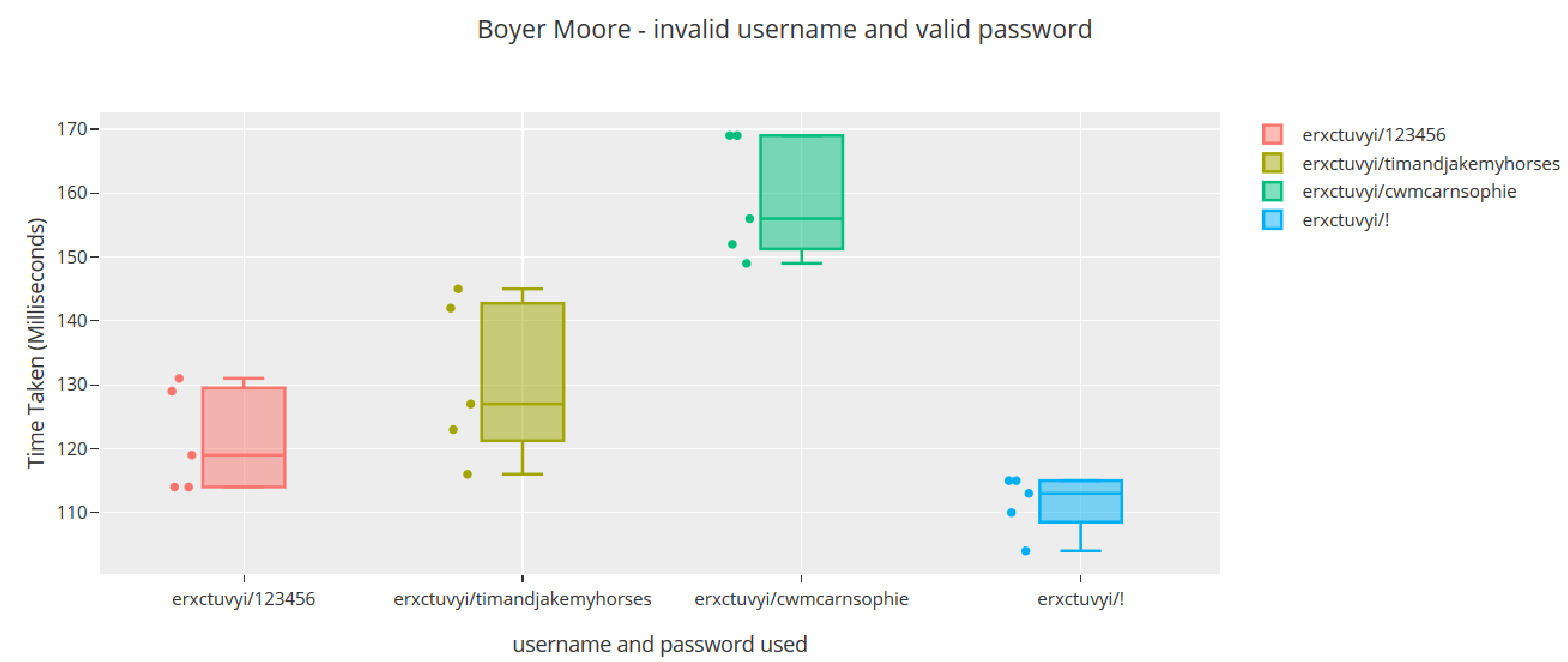
INVALID USERNAME AND VALID PASSWORD – ERXCTUVYI/!



COMPARISONS – INVALID USERNAME AND VALID PASSWORD



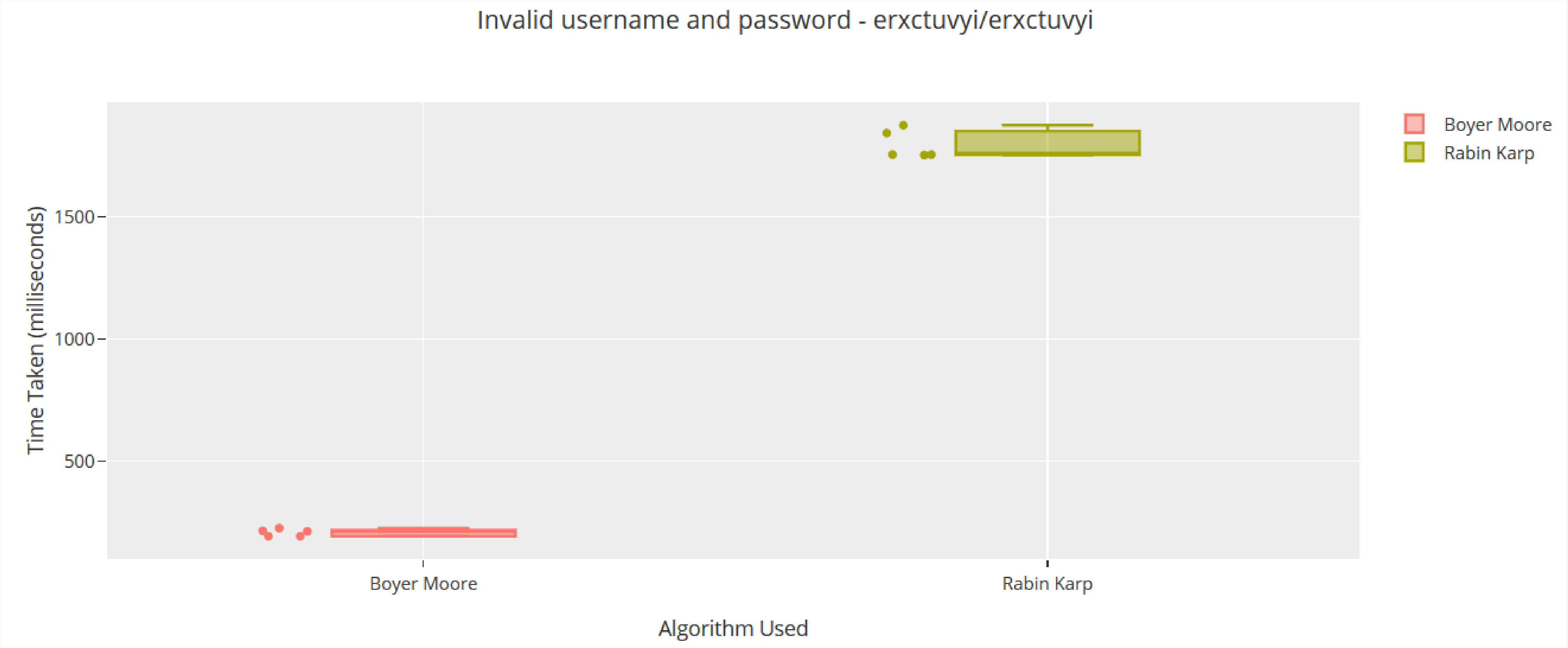
COMPARISONS – INVALID USERNAME AND VALID PASSWORD



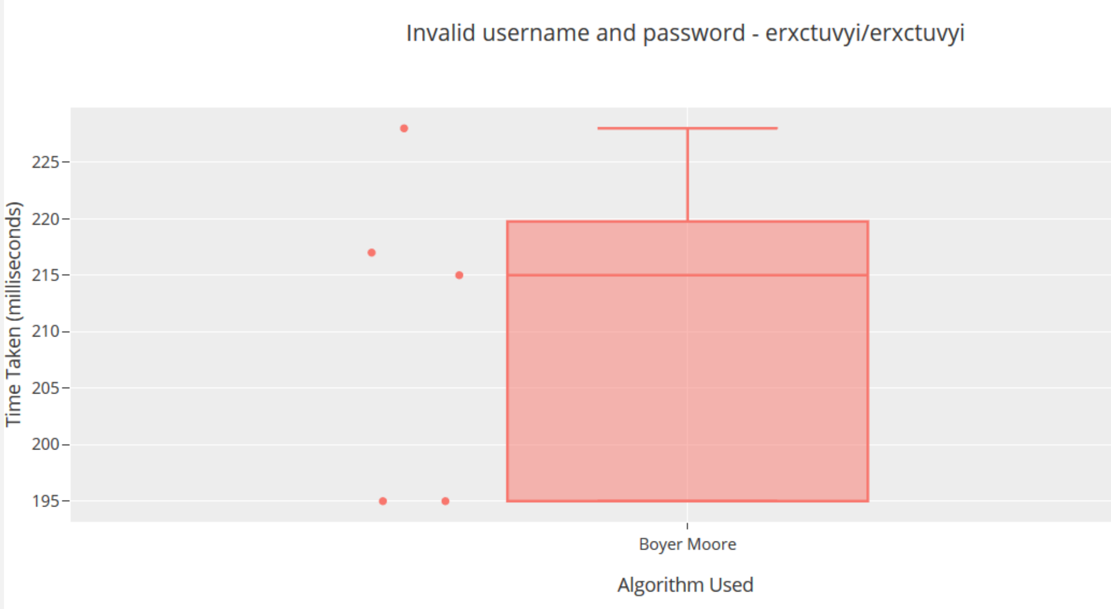
COMPARISONS – INVALID USERNAME AND PASSWORD

- To test the invalid username and password, we are just going to use the junk text.
- For invalid entries, the following text will be used: "erxctuvyi" – It is junk text and is not in the username or password text file.

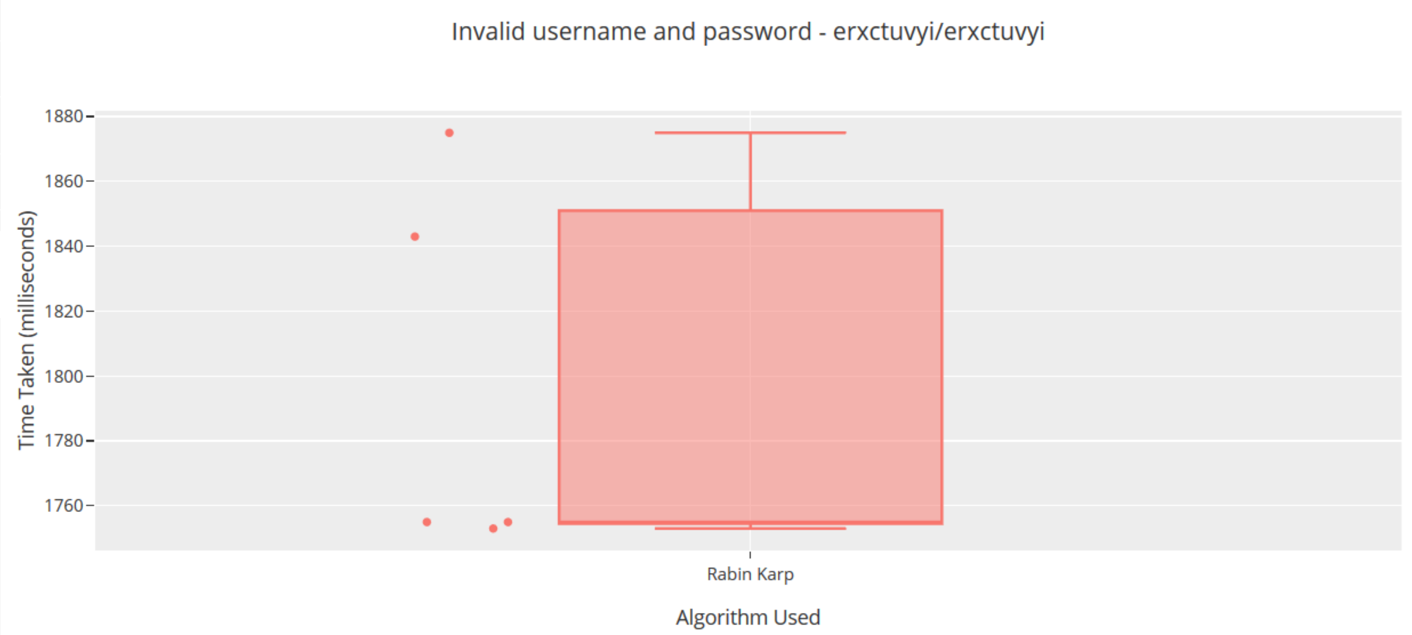
COMPARISONS – INVALID USERNAME AND PASSWORD



COMPARISONS – INVALID USERNAME AND PASSWORD



Invalid username and password - erxctuvyi/erxctuvyi



Invalid username and password - erxctuvyi/erxctuvyi

CONCLUSION - ACTUAL

- It is clear to see from the box plots that Boyer Moore was the fastest algorithm.
- Rabin Karp had similar times for some of the tests, however Boyer Moore was the fastest in all of the tests.
- Rabin Karp performed in it’s worst case time complexity for most of the tests, where Boyer Moore performed in it’s best or average case for most of the tests.

QUESTIONS?

Thanks for listening