Report 2-Analysis **160016114** **28 November, 2019** **Word Count: 2129**

Introduction:
For this part of the practical, the aim was to make a key logger demonstration system, and then perform analysis on certain hypothesis given in the specification. I chose Hypothesis 1,2 and 4.

Methodology:
Keylogger: (main.pyw & clone.pyw)
I built the key logger using a python where I used the Pynput library which allows mouse and keyboard input and monitoring activities. Furthermore, I use the event logging system via the logging module of Python to save the logged keys to a file, instead of printing out to the terminal. Separate CSV files are generated for keys pressed and keys released. Since this analysis is being performed keeping in mind that there would be at least one false user and one true user, I just clone my program to 2 separate Python files which allows me to separately log the false users and the true users keystrokes(pair of keys pressed and keys released). The logging of both shift keys on the key board and the caps lock key is disabled for purposes explained in the next part of the design which explains the method of analysis.

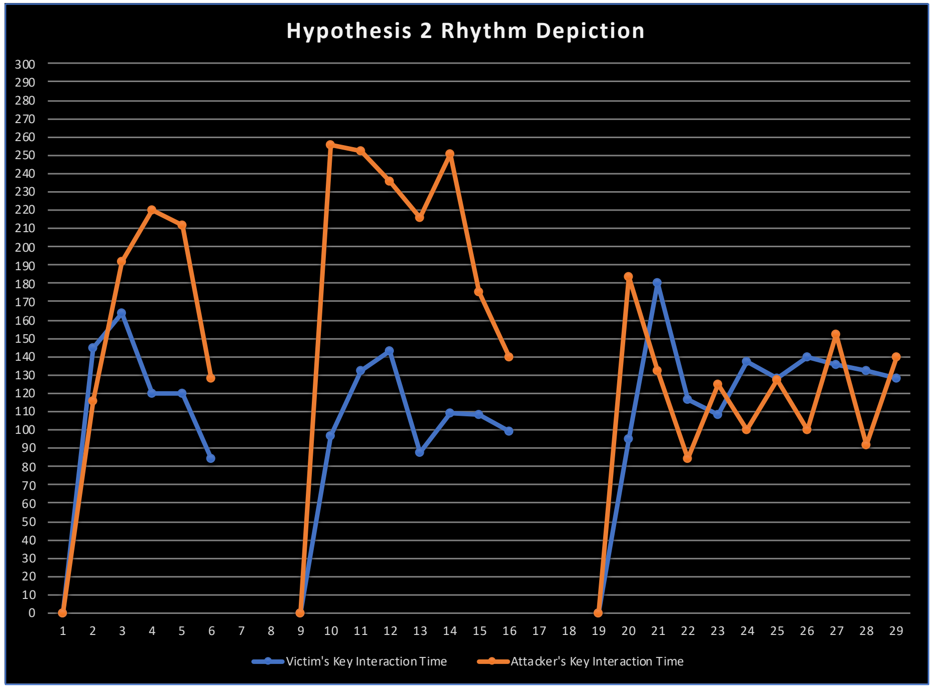
HypA.py:
This is where the numerical calculations are done to either refute or accept the examined hypothesis. Since the only keys that need to be kept in mind are the characters of either usernames or passwords, I choose to either disable the key logger (aforementioned), or I choose to delete the key, from the data frames generated. I use Spyder to do the analysis subsection since it allows me to input CSV type data into data frames, and see single line executions of the code if necessary to see what's happening when performing calculations.

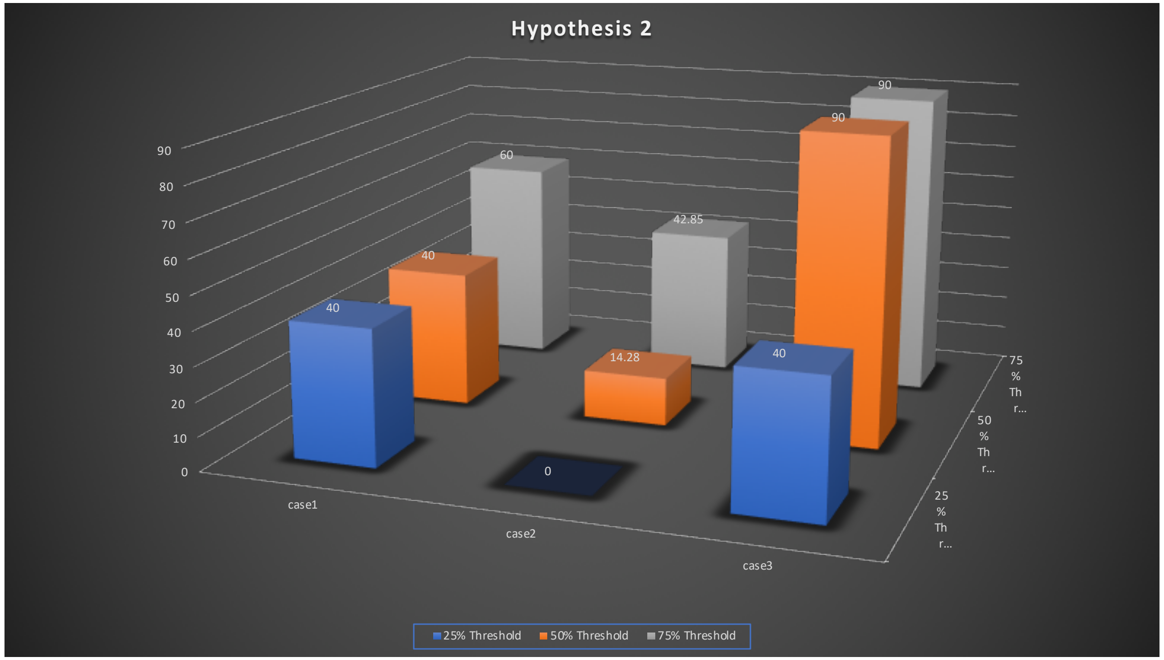
The pyw files generate four separate files for the keystrokes, two each for the false user and the true user. I load each input into 4 separate data frames, and calculate the difference between the key interaction between the false user and the key interaction between the true user. Key interaction here means the time taken to perform the action of pressing and releasing the key (time taken to press - time taken to release).
The three hypothesis require me to interact with the concept of **false acceptance occurrences.**

The false acceptance rates/false positive for the purpose of analysing the hypothesis is defined as when the false user takes roughly the same to interact with the key as the true user would. The false positives (represented in boolean form) are generated for each keystroke of the attacker in one input. This input could be a password or a username entry. A threshold value is set which is generated from the true user's key interaction time. That threshold value is added and subtracted to the true user's key interaction times to generate a range of tolerance. If the false users fall within that tolerance range, then that means the false user lies within the threshold and the system accepts that particular keystroke, making it a false positive. Each hypothesis has been analysed in a way that, each one has three test cases, where the threshold values have been set to 25%, 50% and 75% respectively for each test case. The cases are made by stating different rhythms or the length of the input. A rhythm is classified as a difference between a long keystroke and a short keystroke, or the necessary requirements of an input, for instance, typing in a way to form a certain shape on the keyboard or using certain keys, etc.
It has to be taken into account that for all these rhythms, inputs were taken from 2 different users, and both users had their own way of staying within the case rhythms specified before entering the password. The situation of human fallacy where sometimes the false user might enter an input value too fast or too slow has to also be kept in mind.
The discussion of the test cases of all 3 hypothesis is done below:

Experiment Discussion:
Each Hypothesis comes with 2 graphs, the rhythm depiction just shows graph comparing the interaction times of key presses (Y-axis), with the Key number that is pressed(X-Axis). The second one is the bar graph which highlights the FP rates and allows us to make deductions about the hypothesis. A common behaviour noted by having multiple users input values is that combination of key presses to access certain key values enlongates the interaction time for the user.

Hypothesis 2: I checked for the false positive rates to see if the false user's way of typing (rhythm) produces an interaction time that is sufficient to land within the threshold value generated by the true user's interaction with the key. The false positive rates or the FP rates is being calculated by dividing the keystrokes of the false user within the threshold by the total keystrokes.





Case1: Rhythm: Going Completely right(right handed); Length: 5

FPRate(Threshold 25): 2/5 = 40%
FPRate(Threshold 50): 2/5 = 40%
FPRate(Threshold 75): 3/5 = 60%

Hyp2C1T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'e'	145	'q'	116	36.25	181.25	108.75	True
1	'd'	164	'd'	192	41	205	123	True
2	'r'	120	't'	220	30	150	90	False
3	'f'	120	'y'	212	30	150	90	False
4	't'	84	'j'	128	21	105	63	False

Case2: Rhythm: Alphanumeric(V-shaped(it starts with a number and it ends with a number)) (right hand); Length: 7

Hyp2C1T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'e'	145	'q'	116	72.5	217.5	72.5	True
1	'd'	164	'd'	192	82	246	82	True
2	'r'	120	't'	220	60	180	60	False
3	'f'	120	'y'	212	60	180	60	False
4	't'	84	'j'	128	42	126	42	False

Hyp2C1T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'e'	145	'q'	116	108.75	253.75	36.25	True
1	'd'	164	'd'	192	123	287	41	True
2	'r'	120	't'	220	90	210	30	False
3	'f'	120	'y'	212	90	210	30	False
4	't'	84	'j'	128	63	147	21	True

FPRate(Threshold 25): 0/7 = 0%
FPRate(Threshold 50): 1/7 = 14.28%
FPRate(Threshold 75): 3/7 = 42.85%

Hyp2C2T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'1'	97	'2'	256	24.25	121.25	72.75	False
1	'q'	132	'w'	252	33	165	99	False
2	's'	143	'd'	236	35.75	178.75	107.25	False
3	'x'	88	'v'	216	22	110	66	False
4	'd'	109	'g'	251	27.25	136.25	81.75	False
5	'r'	108	'y'	175	27	135	81	False
6	'5'	99	'8'	140	24.75	123.75	74.25	False

Hyp2C2T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'1'	97	'2'	256	48.5	145.5	48.5	False
1	'q'	132	'w'	252	66	198	66	False
2	's'	143	'd'	236	71.5	214.5	71.5	False
3	'x'	88	'v'	216	44	132	44	False
4	'd'	109	'g'	251	54.5	163.5	54.5	False
5	'r'	108	'y'	175	54	162	54	False
6	'5'	99	'8'	140	49.5	148.5	49.5	True

Hyp2C2T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'1'	97	'2'	256	72.75	169.75	24.25	False
1	'q'	132	'w'	252	99	231	33	False
2	's'	143	'd'	236	107.25	250.25	35.75	True
3	'x'	88	'v'	216	66	154	22	False
4	'd'	109	'g'	251	81.75	190.75	27.25	False
5	'r'	108	'y'	175	81	189	27	True
6	'5'	99	'8'	140	74.25	173.25	24.75	True

Case3: Rhythm: Alternating Alphabets & Special Characters; Length: 10
FPRate(Threshold 25): 4/10 = 40%

FPRate(Threshold 50): 9/10 = 90%
FPRate(Threshold 75): 9/10 = 90%

Hyp2C3T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'g'	95	'q'	184	23.75	118.75	71.25	False
1	'$'	180	'#'	132	45	225	135	False
2	'u'	117	'r'	84	29.25	146.25	87.75	False
3	'*'	108	'%'	125	27	135	81	True
4	'g'	137	't'	100	34.25	171.25	102.75	False
5	'^'	128	'&'	127	32	160	96	True
6	's'	140	'h'	100	35	175	105	False
7	'\\'	136	'*'	152	34	170	102	True
8	'f'	132	'i'	92	33	165	99	False
9	'#'	128	'('	140	32	160	96	True

Hyp2C3T50

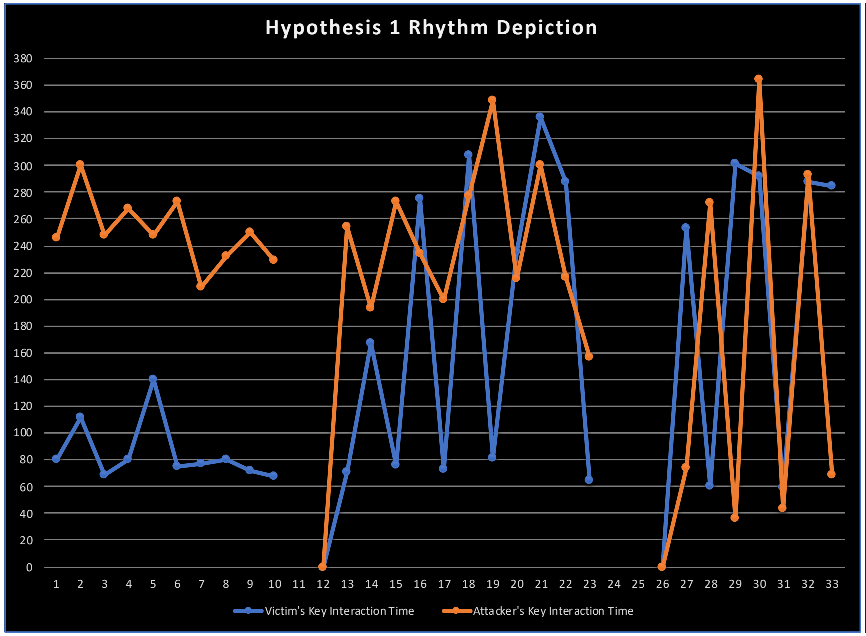
	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'g'	95	'q'	184	47.5	142.5	47.5	False
1	'$'	180	'#'	132	90	270	90	True
2	'u'	117	'r'	84	58.5	175.5	58.5	True
3	'*'	108	'%'	125	54	162	54	True
4	'g'	137	't'	100	68.5	205.5	68.5	True
5	'^'	128	'&'	127	64	192	64	True
6	's'	140	'h'	100	70	210	70	True
7	'\\'	136	'*'	152	68	204	68	True
8	'f'	132	'i'	92	66	198	66	True
9	'#'	128	'('	140	64	192	64	True

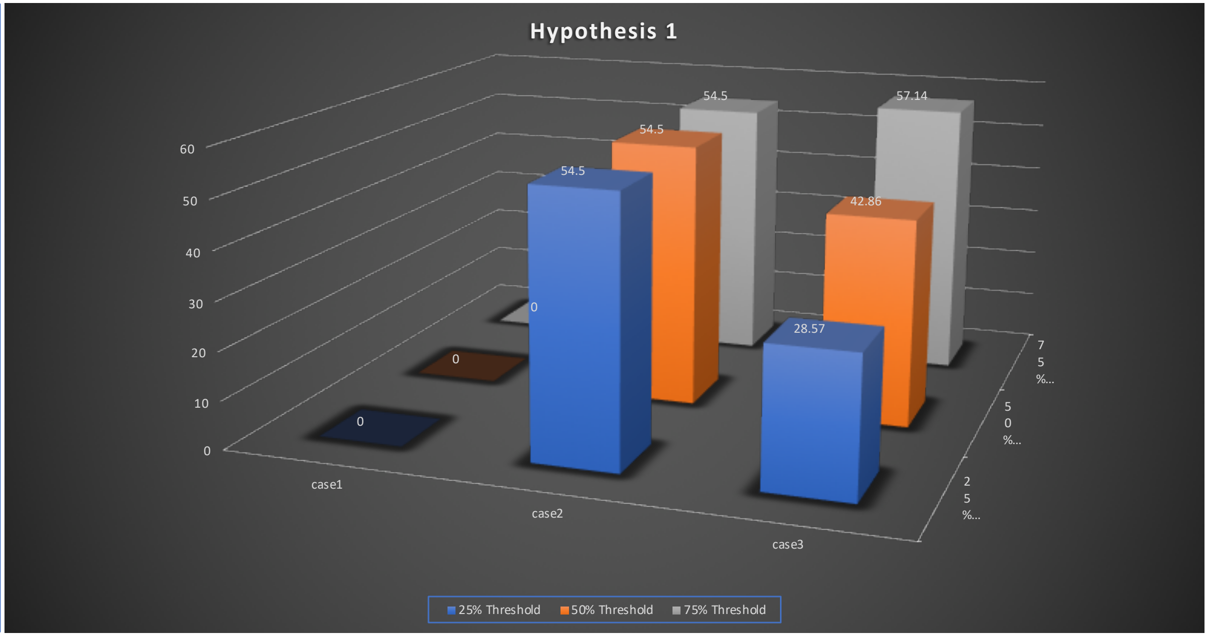
Hyp2C3T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'g'	95	'q'	184	71.25	166.25	23.75	False
1	'$'	180	'#'	132	135	315	45	True
2	'u'	117	'r'	84	87.75	204.75	29.25	True
3	'*'	108	'%'	125	81	189	27	True
4	'g'	137	't'	100	102.75	239.75	34.25	True
5	'^'	128	'&'	127	96	224	32	True
6	's'	140	'h'	100	105	245	35	True
7	'\\'	136	'*'	152	102	238	34	True
8	'f'	132	'i'	92	99	231	33	True
9	'#'	128	'('	140	96	224	32	True

As we can see from the graph above, The FP rate is the highest in case 3 and the lowest in case 2, so it is easier for the false user to break a pattern where the typing performed by both the true and false users would be relatively slow, due to the complexity of the input. For instance, the special characters might have been accessed by pressing the shift key before, so the reflexes of the true user would also slow down, giving the false user more time to stay within the threshold. Another explanation could also be that the false user just has a naturally faster typing speed compared to the true user. So the graph visualises the findings of analysing the hypothesis by showing that patterns regardless of their complexity to type can still be easily broken into, compared to an easier one.

Hypothesis 1: See if there is a decline or an incline in the False Positive rates. Get the True positive rates and then create TP:FP ratio. Since we're more focused on the relation between the fashion with which the true user and false user types, instead of what's being typed, the passwords being typed can be the same since that plays no role in determining the role of rhythm.
Assuming password is the same rhythm is different





Case 1:
(Input) Password: barca4ever
Rhythm for true user: only short keystrokes
Rhythm for false user: only long keystrokes

As shown that that there exists no false positives(0%) whatsoever at all, since the attacker would type slowly and guess the right password whereas the user wouldn't guess and type it fast.
The TP:FP ratio here would be 100:0 -
Notice that the false user's rhythm would be easily detected by the system as it is exactly opposite of what the true user's interaction rhythm.

Hyp1C1T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'b'	80	'b'	246	20	100	60	False
1	'a'	112	'a'	300	28	140	84	False
2	'r'	69	'r'	248	17.25	86.25	51.75	False
3	'c'	80	'c'	268	20	100	60	False
4	'a'	140	'a'	248	35	175	105	False
5	'4'	75	'4'	273	18.75	93.75	56.25	False
6	'e'	77	'e'	209	19.25	96.25	57.75	False
7	'v'	80	'v'	232	20	100	60	False
8	'e'	72	'e'	250	18	90	54	False
9	'r'	68	'r'	229	17	85	51	False

Hyp1C1T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'b'	80	'b'	246	40	120	40	False
1	'a'	112	'a'	300	56	168	56	False
2	'r'	69	'r'	248	34.5	103.5	34.5	False
3	'c'	80	'c'	268	40	120	40	False
4	'a'	140	'a'	248	70	210	70	False
5	'4'	75	'4'	273	37.5	112.5	37.5	False
6	'e'	77	'e'	209	38.5	115.5	38.5	False
7	'v'	80	'v'	232	40	120	40	False
8	'e'	72	'e'	250	36	108	36	False
9	'r'	68	'r'	229	34	102	34	False

Hyp1C1T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'b'	80	'b'	246	60	140	20	False
1	'a'	112	'a'	300	84	196	28	False
2	'r'	69	'r'	248	51.75	120.75	17.25	False
3	'c'	80	'c'	268	60	140	20	False
4	'a'	140	'a'	248	105	245	35	False
5	'4'	75	'4'	273	56.25	131.25	18.75	False
6	'e'	77	'e'	209	57.75	134.75	19.25	False
7	'v'	80	'v'	232	60	140	20	False
8	'e'	72	'e'	250	54	126	18	False
9	'r'	68	'r'	229	51	119	17	False

Case 2:
(Input) Password: q&a&q&a&Q&a
Rhythm for true user: Alternative long and short keystroke(start with short)
Rhythm for false user: long keystrokes

Assume that to make the situation more realistic as to why the user might have certain long keystrokes and the false user always because of the complex nature of the password, and even though the true user knows the password, their keystrokes for certain key values which require a combination of keys to pressed(eg: shift key + another key) might be slightly longer compared to other keystrokes which don't require that amount of effort.
The TP:FP ratio here would be 45.5:54.5 -
Notice that this because the password length is odd, the ratio could be reversed if there were one less long keystrokes in the true users rhythm and one more short keystroke. So here the TP:FP rate are also dependent on the length. Now the reason the TP:FP is almost split into half is because of the rhythm of the false user is a sub-part of the rhythm of the true user, and because of this, the false user's simple pattern would be easily detectable when compared to the true user's pattern.

Hyp1C2T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'q'	71	'q'	254	17.75	88.75	53.25	False
1	'&'	167	'&'	193	41.75	208.75	125.25	True
2	'a'	76	'a'	273	19	95	57	False
3	'&'	275	'&'	234	68.75	343.75	206.25	True
4	'q'	73	'q'	200	18.25	91.25	54.75	False
5	'&'	308	'&'	277	77	385	231	True
6	'a'	81	'a'	349	20.25	101.25	60.75	False
7	'&'	235	'&'	215	58.75	293.75	176.25	True
8	'Q'	336	'Q'	300	84	420	252	True
9	'&'	288	'&'	216	72	360	216	True
10	'a'	64	'a'	157	16	80	48	False

Hyp1C2T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'q'	71	'q'	254	35.5	106.5	35.5	False
1	'&'	167	'&'	193	83.5	250.5	83.5	True
2	'a'	76	'a'	273	38	114	38	False
3	'&'	275	'&'	234	137.5	412.5	137.5	True
4	'q'	73	'q'	200	36.5	109.5	36.5	False
5	'&'	308	'&'	277	154	462	154	True
6	'a'	81	'a'	349	40.5	121.5	40.5	False
7	'&'	235	'&'	215	117.5	352.5	117.5	True
8	'Q'	336	'Q'	300	168	504	168	True
9	'&'	288	'&'	216	144	432	144	True
10	'a'	64	'a'	157	32	96	32	False

Hyp1C2T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'q'	71	'q'	254	53.25	124.25	17.75	False
1	'&'	167	'&'	193	125.25	292.25	41.75	True
2	'a'	76	'a'	273	57	133	19	False
3	'&'	275	'&'	234	206.25	481.25	68.75	True
4	'q'	73	'q'	200	54.75	127.75	18.25	False
5	'&'	308	'&'	277	231	539	77	True
6	'a'	81	'a'	349	60.75	141.75	20.25	False
7	'&'	235	'&'	215	176.25	411.25	58.75	True
8	'Q'	336	'Q'	300	252	588	84	True
9	'&'	288	'&'	216	216	504	72	True
10	'a'	64	'a'	157	48	112	16	False

Case 3:
(Input) Password: DogsFire
Rhythm for true user: Alternative long and short keystrokes(Start with long)
Rhythm for false user: Alternative long and short keystrokes(Start with short)
Here it is realistic to **assume** that both the user and the false user could take some time to different keystrokes, since both the true and the false user would apply the same rhythm but in an inverted fashion. As typing consecutive keys with alternative speeds could be a confusing task, so discrepancies might occur more in calculating the FP rates. That is why I have calculated the average FP rate given from the graph of the 2 cases and then taken the TP:FP ratio.
Average FP = (28.57+42.86+57.14)/300 = 0.428 = 42.8%
Hence the True positive rate here would be 57.2%
The TP:FP ratio here would be 57.2:42.8

There is an incline separately in the FP' ratio here compared to case 1 and 2 where the FP ratio remains stagnant. So this proves the hypothesis that a simple rhythm can be more easily determined than a patterned one.

Hyp1C3T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'D'	253	'D'	74	63.25	316.25	189.75	False
1	'o'	60	'o'	272	15	75	45	False
2	'g'	301	'g'	36	75.25	376.25	225.75	False
3	'F'	292	'F'	364	73	365	219	True
4	'i'	59	'i'	44	14.75	73.75	44.25	False
5	'r'	288	'r'	293	72	360	216	True
6	'e'	285	'e'	69	71.25	356.25	213.75	False

Hyp1C3T50

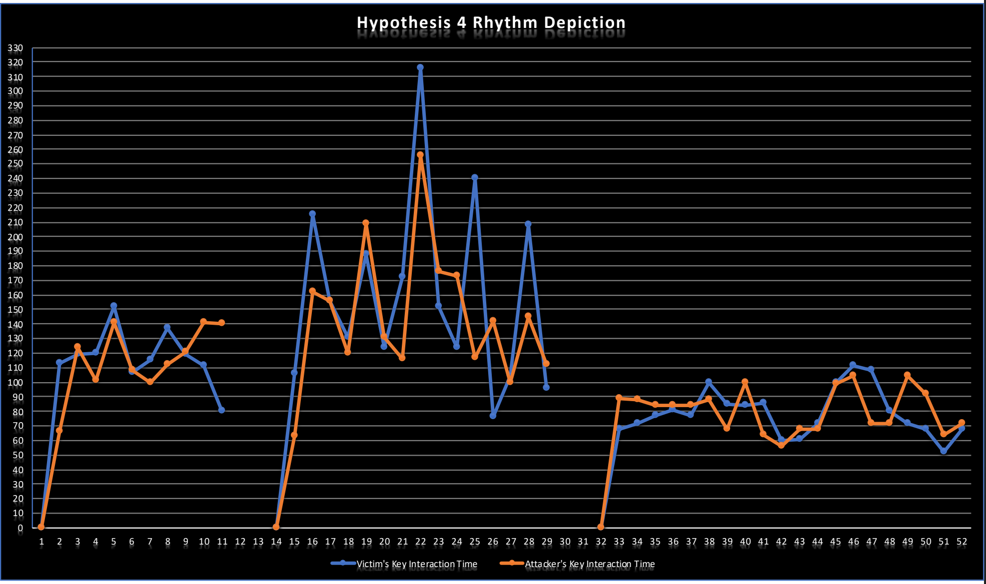
	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'D'	253	'D'	74	126.5	379.5	126.5	False
1	'o'	60	'o'	272	30	90	30	False
2	'g'	301	'g'	36	150.5	451.5	150.5	False
3	'F'	292	'F'	364	146	438	146	True
4	'i'	59	'i'	44	29.5	88.5	29.5	True
5	'r'	288	'r'	293	144	432	144	True
6	'e'	285	'e'	69	142.5	427.5	142.5	False

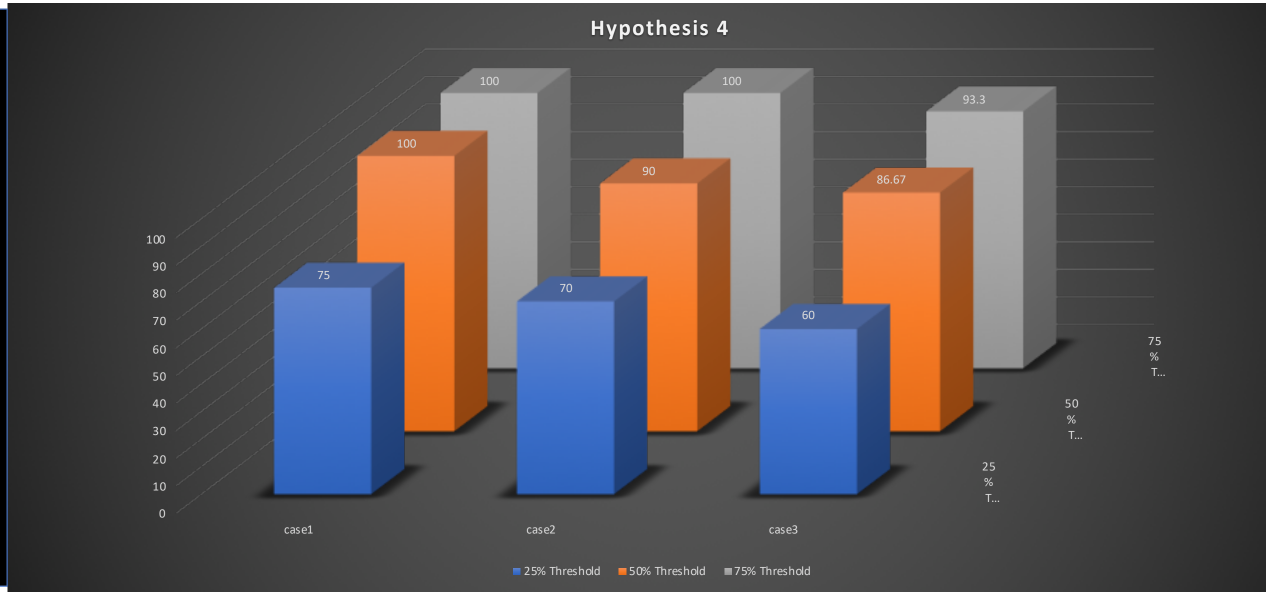
Hyp1C3T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'D'	253	'D'	74	189.75	442.75	63.25	True
1	'o'	60	'o'	272	45	105	15	False
2	'g'	301	'g'	36	225.75	526.75	75.25	False
3	'F'	292	'F'	364	219	511	73	True
4	'i'	59	'i'	44	44.25	103.25	14.75	True
5	'r'	288	'r'	293	216	504	72	True
6	'e'	285	'e'	69	213.75	498.75	71.25	False

Case 2 & Case 3 were quite complex to implement for the users as it requires reflexes to be used in a pattern someone might not be used to, i.e the true user might get a bit confused whilst typing the password, if they just started using that password, or the attacker who's just trying to guess would type it in a way that would cause tardiness. That's why human fallacy would come into play in some of the keystrokes that would fall into the threshold or fall out of the threshold.

Hypothesis 4: The data was taken from 2 different users





Case1:
Rhythm: Both typing short keystrokes
Length: 20char
password1(True User): dogfireyellowdetroit
password2(False User): wasteplanthorseblade
It is **assumed** that both users have been typing on the keyboard for a long time.

Hyp4C1T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'd'	68	'w'	89	17	85	51	False
1	'o'	72	'a'	88	18	90	54	True
2	'g'	77	's'	84	19.25	96.25	57.75	True
3	'f'	81	't'	84	20.25	101.25	60.75	True
4	'i'	77	'e'	84	19.25	96.25	57.75	True
5	'r'	100	'p'	88	25	125	75	True
6	'e'	85	'l'	68	21.25	106.25	63.75	True
7	'y'	84	'a'	100	21	105	63	True
8	'e'	86	'n'	64	21.5	107.5	64.5	False
9	'l'	60	't'	56	15	75	45	True
10	'l'	61	'h'	68	15.25	76.25	45.75	True
11	'o'	72	'o'	68	18	90	54	True
12	'w'	100	'r'	99	25	125	75	True
13	'd'	111	's'	104	27.75	138.75	83.25	True
14	'e'	108	'e'	72	27	135	81	False
15	't'	80	'b'	72	20	100	60	True
16	'r'	72	'l'	104	18	90	54	False
17	'o'	68	'a'	92	17	85	51	False
18	'i'	52	'd'	64	13	65	39	True
19	't'	68	'e'	72	17	85	51	True

Hyp4C1T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'd'	68	'w'	89	34	102	34	True
1	'o'	72	'a'	88	36	108	36	True
2	'g'	77	's'	84	38.5	115.5	38.5	True
3	'f'	81	't'	84	40.5	121.5	40.5	True
4	'i'	77	'e'	84	38.5	115.5	38.5	True
5	'r'	100	'p'	88	50	150	50	True
6	'e'	85	'l'	68	42.5	127.5	42.5	True
7	'y'	84	'a'	100	42	126	42	True
8	'e'	86	'n'	64	43	129	43	True
9	'l'	60	't'	56	30	90	30	True
10	'l'	61	'h'	68	30.5	91.5	30.5	True
11	'o'	72	'o'	68	36	108	36	True
12	'w'	100	'r'	99	50	150	50	True
13	'd'	111	's'	104	55.5	166.5	55.5	True
14	'e'	108	'e'	72	54	162	54	True
15	't'	80	'b'	72	40	120	40	True
16	'r'	72	'l'	104	36	108	36	True
17	'o'	68	'a'	92	34	102	34	True
18	'i'	52	'd'	64	26	78	26	True
19	't'	68	'e'	72	34	102	34	True

Hyp4C1T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'd'	68	'w'	89	51	119	17	True
1	'o'	72	'a'	88	54	126	18	True
2	'g'	77	's'	84	57.75	134.75	19.25	True
3	'f'	81	't'	84	60.75	141.75	20.25	True
4	'i'	77	'e'	84	57.75	134.75	19.25	True
5	'r'	100	'p'	88	75	175	25	True
6	'e'	85	'l'	68	63.75	148.75	21.25	True
7	'y'	84	'a'	100	63	147	21	True
8	'e'	86	'n'	64	64.5	150.5	21.5	True
9	'l'	60	't'	56	45	105	15	True
10	'l'	61	'h'	68	45.75	106.75	15.25	True
11	'o'	72	'o'	68	54	126	18	True
12	'w'	100	'r'	99	75	175	25	True
13	'd'	111	's'	104	83.25	194.25	27.75	True
14	'e'	108	'e'	72	81	189	27	True
15	't'	80	'b'	72	60	140	20	True
16	'r'	72	'l'	104	54	126	18	True
17	'o'	68	'a'	92	51	119	17	True
18	'i'	52	'd'	64	39	91	13	True
19	't'	68	'e'	72	51	119	17	True

Case 2:
Rhythm: True User typing a right moving password with left hand and the other typing a left moving password with right hand
Length: 10 Char
password1(True User): qwdfvbhjio
password2(False User): pokjnbgfre
It is **assumed** that both users are typing in a way that would be equally challenging for them, so both users are typing with their dominant hands.

Hyp4C2T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'q'	113	'p'	66	28.25	141.25	84.75	False
1	'w'	119	'o'	124	29.75	148.75	89.25	True
2	'd'	120	'k'	101	30	150	90	True
3	'f'	152	'j'	141	38	190	114	True
4	'v'	107	'n'	108	26.75	133.75	80.25	True
5	'b'	115	'b'	100	28.75	143.75	86.25	True
6	'h'	137	'g'	112	34.25	171.25	102.75	True
7	'j'	119	'f'	121	29.75	148.75	89.25	True
8	'i'	111	'r'	141	27.75	138.75	83.25	False
9	'o'	80	'e'	140	20	100	60	False

Hyp4C2T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'q'	113	'p'	66	56.5	169.5	56.5	True
1	'w'	119	'o'	124	59.5	178.5	59.5	True
2	'd'	120	'k'	101	60	180	60	True
3	'f'	152	'j'	141	76	228	76	True
4	'v'	107	'n'	108	53.5	160.5	53.5	True
5	'b'	115	'b'	100	57.5	172.5	57.5	True
6	'h'	137	'g'	112	68.5	205.5	68.5	True
7	'j'	119	'f'	121	59.5	178.5	59.5	True
8	'i'	111	'r'	141	55.5	166.5	55.5	True
9	'o'	80	'e'	140	40	120	40	False

Hyp4C2T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	'q'	113	'p'	66	84.75	197.75	28.25	True
1	'w'	119	'o'	124	89.25	208.25	29.75	True
2	'd'	120	'k'	101	90	210	30	True
3	'f'	152	'j'	141	114	266	38	True
4	'v'	107	'n'	108	80.25	187.25	26.75	True
5	'b'	115	'b'	100	86.25	201.25	28.75	True
6	'h'	137	'g'	112	102.75	239.75	34.25	True
7	'j'	119	'f'	121	89.25	208.25	29.75	True
8	'i'	111	'r'	141	83.25	194.25	27.75	True
9	'o'	80	'e'	140	60	140	20	True

Case3: Rhythm : True User typing long keystroke on ‘O’ and ‘!’, and the other typing short keystroke on ‘A’ and ‘!’
Length: 15 Char
password1(True User):: t!ctOct!ctOct!c
password2(False User): n!cnAcn!cnAcn!c

Hyp4C3T25

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	't'	106	'n'	63	26.5	132.5	79.5	False
1	'!'	215	'!'	162	53.75	268.75	161.25	True
2	'c'	156	'c'	156	39	195	117	True
3	't'	131	'n'	120	32.75	163.75	98.25	True
4	'O'	188	'A'	209	47	235	141	True
5	'c'	124	'c'	131	31	155	93	True
6	't'	172	'n'	116	43	215	129	False
7	'!'	316	'!'	256	79	395	237	True
8	'c'	152	'c'	176	38	190	114	True
9	't'	124	'n'	173	31	155	93	False
10	'O'	240	'A'	117	60	300	180	False
11	'c'	76	'c'	142	19	95	57	False
12	't'	104	'n'	100	26	130	78	True
13	'!'	208	'!'	145	52	260	156	False
14	'c'	96	'c'	112	24	120	72	True

Hyp4C3T50

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	't'	106	'n'	63	53	159	53	True
1	'!'	215	'!'	162	107.5	322.5	107.5	True
2	'c'	156	'c'	156	78	234	78	True
3	't'	131	'n'	120	65.5	196.5	65.5	True
4	'O'	188	'A'	209	94	282	94	True
5	'c'	124	'c'	131	62	186	62	True
6	't'	172	'n'	116	86	258	86	True
7	'!'	316	'!'	256	158	474	158	True
8	'c'	152	'c'	176	76	228	76	True
9	't'	124	'n'	173	62	186	62	True
10	'O'	240	'A'	117	120	360	120	False
11	'c'	76	'c'	142	38	114	38	False
12	't'	104	'n'	100	52	156	52	True
13	'!'	208	'!'	145	104	312	104	True
14	'c'	96	'c'	112	48	144	48	True

Hyp4C3T75

	Victim's Key	Victim's Key Interaction Time	Attacker's Key	Attacker's Key Interaction Time	Threshold Val	Tolerance+	Tolerance-	Within Threshold
0	't'	106	'n'	63	79.5	185.5	26.5	True
1	'!'	215	'!'	162	161.25	376.25	53.75	True
2	'c'	156	'c'	156	117	273	39	True
3	't'	131	'n'	120	98.25	229.25	32.75	True
4	'O'	188	'A'	209	141	329	47	True
5	'c'	124	'c'	131	93	217	31	True
6	't'	172	'n'	116	129	301	43	True
7	'!'	316	'!'	256	237	553	79	True
8	'c'	152	'c'	176	114	266	38	True
9	't'	124	'n'	173	93	217	31	True
10	'O'	240	'A'	117	180	420	60	True
11	'c'	76	'c'	142	57	133	19	False
12	't'	104	'n'	100	78	182	26	True
13	'!'	208	'!'	145	156	364	52	True
14	'c'	96	'c'	112	72	168	24	True

Here I have deliberately chosen to plot them in said alignment because the graph itself is depicting that there is a left to right decrease(from case1 to case3), even though the length of case2 should have lower raised set of bars compared to case 3 as it is greater in length. Clearly, it isn’t the case that increase in length is affecting the FP rates because the average FP rate for case1 is 91%, for case 2 is 86% and for case 3 is 79.9%. This is due to the implementation of different types of rhythms and the combination of characters chosen for the password.
The analysis shows above that length plays no role in detection of false positives or false negatives, but it is the rhythm with which the user types that makes a difference. So Hypothesis 4 is refuted.

The Conclusion:

I chose 3 hypothesis the basis of which I understood well. The fundamentals of my analysis included creating unique rhythms, including unique combinations of characters, which highlight the essence of strong user credentials. Furthermore, the concept to use threshold value made it easy to identify False Positives and make comprehensible deductions by visualisation of the forms of graphs and tables. In essence, the observations laid out for one hypothesis could be used to prove another hypothesis as well, since I focused on making my parameters as flexible as possible(For instance, hypothesis 2 could also be used to prove hypothesis 4, given that there’s a constant increase in length of input characters).
Using python for keystroke analysis made the whole process faster. Given more time, I would work on the remaining hypothesis, come up with some of my own, and broaden my observation spectrum in terms of using multiple thresholds, and more rhythms and passwords.

Appendix:

1. **Source Code for analysis: HypA.py:**

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 27 04:05:23 2019

@author: gopaljuneja
"""
import pandas as pd
import os
import numpy as np
from datetime import datetime as dt



def deleteRow(df, colName, val):
    #delete last row from dataframe
    if df[colName].str.contains(val).any():
        df = df.drop(df.index[-1])
    return df


def readLogFile(df, path):
    df = pd.read_csv(path,names=["DateTime", "Time(Milliseconds)", "Key"], dtype={'Time(Milliseconds)': str})
    return df

def extractTime(df):
    #Extracting time out of DateTime into a separate column
    df['Time'] = df['DateTime'].str.split(' ').str[1]
    #Combining the time column with milliseconds column
```

```
    df['Time'] = df['Time'] + "." + df["Time(Milliseconds)"].map(str)
    df = df.drop(['DateTime'], axis =1)
    return df

def toMSformat(df, label1):
    #column for key pressed time
    df[label1] = df["Time"]
    df[label1] = df[label1].astype(str)
    #converting the type to timedelta from object
    df[label1] = df[label1].apply(pd.to_timedelta)
    #converting time to milliseconds
    df[label1] = df[label1].astype('timedelta64[ms]')
    df[label1] = df[label1].apply(lambda x: '%.f' % x)

    return df[label1]


def calcTimeDiff(df1, df2):
    #df_TimeDifference = pd.DataFrame()
    df_TimeDifference = pd.DataFrame()

    df_TimeDifference["timeKeyPressed(ms)"] = toMSformat(df1, "timeKeyPressed(ms)")
    df_TimeDifference["timeKeyReleased(ms)"] = toMSformat(df2, "timeKeyReleased(ms)")
    #calculating the time difference
    df_TimeDifference['time_diff'] = df_TimeDifference['timeKeyReleased(ms)'].astype(float) -
df_TimeDifference['timeKeyPressed(ms)'].astype(float)
    #Adding the key values to dataframe
    df_TimeDifference['Key'] = df1['Key']
    return df_TimeDifference


def formatDF(df, filepath):
    df = pd.DataFrame()
    df = readLogFile(df,filepath)
    df = deleteRow(df, 'Key', 'Key.esc')
    df = extractTime(df)
    df['Time(ms)'] = toMSformat(df, 'Time(ms)')
```

```
    return df

df_attackerKeyPress = pd.DataFrame()
df_attackerKeyPress = formatDF(df_attackerKeyPress,r'/Users/gopaljuneja/Desktop/CS4203/key_ErnielogPress.csv')

df_attackerKeyRelease = pd.DataFrame()
df_attackerKeyRelease = formatDF(df_attackerKeyRelease,r'/Users/gopaljuneja/Desktop/CS4203/key_ErnielogRelease.csv')

df_victimKeyPress = pd.DataFrame()
df_victimKeyPress = formatDF(df_victimKeyPress,r'/Users/gopaljuneja/Desktop/CS4203/key_PandalogPress.csv')

df_victimKeyRelease = pd.DataFrame()
df_victimKeyRelease = formatDF(df_victimKeyRelease,r'/Users/gopaljuneja/Desktop/CS4203/key_PandalogRelease.csv')


#Getting Interaction time with each key: which is key released - key pressed
df_timeTakenA = calcTimeDiff(df_attackerKeyPress, df_attackerKeyRelease)
df_timeTakenA['Key'] = df_attackerKeyPress['Key']
df_timeTakenV = calcTimeDiff(df_victimKeyPress, df_victimKeyRelease)
df_timeTakenV['Key'] = df_victimKeyPress['Key']

#this is checking the time taken to press each key by the victim and the attacker and then creates a threshold checking attackers logs
#under tolerance rates
#Interaction Time: The time taken to press and release a key

def hyp2a(threshold):
    df_verifyHypothesis = pd.DataFrame()
    df_verifyHypothesis["Victim's Key "] = df_timeTakenV["Key"]
    df_verifyHypothesis["Victim's Key Interaction Time"] = df_timeTakenV["time_diff"]
    df_verifyHypothesis["Attacker's Key"] = df_timeTakenA["Key"]
    df_verifyHypothesis["Attacker's Key Interaction Time"] = df_timeTakenA["time_diff"]


    df_verifyHypothesis["Threshold Val"] = (df_timeTakenV["time_diff"] * threshold)/100

    df_verifyHypothesis['Tolerance+'] = df_verifyHypothesis["Victim's Key Interaction Time"] + df_verifyHypothesis['Threshold Val']
    df_verifyHypothesis['Tolerance-'] = df_verifyHypothesis["Victim's Key Interaction Time"] - df_verifyHypothesis['Threshold Val']
```

```
    df_verifyHypothesis['Within Threshold'] = np.where((df_verifyHypothesis["Attacker's Key Interaction Time"] > df_verifyHypothesis['Tolerance+']) | (df_verifyHypothesis["Attacker's Key Interaction Time"] < df_verifyHypothesis['Tolerance-']) , 'False', 'True')
    return  df_verifyHypothesis

#################### Case1 hypothesis 1

df_Hypothesis1C1T25 = hyp2a(25)
countC1T25 = df_Hypothesis1C1T25['Within Threshold'].value_counts()
df_Hypothesis1C1T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C1T25.csv')

df_Hypothesis1C1T50 = hyp2a(50)
countC1T50 = df_Hypothesis1C1T50['Within Threshold'].value_counts()
df_Hypothesis1C1T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C1T50.csv')

df_Hypothesis1C1T75 = hyp2a(75)
countC1T75 = df_Hypothesis1C1T75['Within Threshold'].value_counts()
df_Hypothesis1C1T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C1T75.csv')

#################### Case2 hypothesis 1
df_Hypothesis1C2T25 = hyp2a(25)
countC2T25 = df_Hypothesis1C2T25['Within Threshold'].value_counts()
df_Hypothesis1C2T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C2T25.csv')

df_Hypothesis1C2T50 = hyp2a(50)
countC2T50 = df_Hypothesis1C2T50['Within Threshold'].value_counts()
df_Hypothesis1C2T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C2T50.csv')

df_Hypothesis1C2T75 = hyp2a(75)
countC2T75 = df_Hypothesis1C2T75['Within Threshold'].value_counts()
df_Hypothesis1C2T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C2T75.csv')

#################### Case3 hypothesis 1
df_Hypothesis1C3T25 = hyp2a(25)
countC3T25 = df_Hypothesis1C3T25['Within Threshold'].value_counts()
df_Hypothesis1C3T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C3T25.csv')
```

```
df_Hypothesis1C3T50 = hyp2a(50)
countC3T50 = df_Hypothesis1C3T50['Within Threshold'].value_counts()
df_Hypothesis1C3T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C3T50.csv')

df_Hypothesis1C3T75 = hyp2a(75)
countC3T75 = df_Hypothesis1C3T75['Within Threshold'].value_counts()
df_Hypothesis1C3T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp1C3T75.csv')

#################### Case1 hypothesis 2
df_Hypothesis1C1T25 = hyp2a(25)
H2countC1T25 = df_Hypothesis1C1T25['Within Threshold'].value_counts()
df_Hypothesis1C1T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C1T25.csv')

df_Hypothesis1C1T50 = hyp2a(50)
H2countC1T50 = df_Hypothesis1C1T50['Within Threshold'].value_counts()
df_Hypothesis1C1T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C1T50.csv')

df_Hypothesis1C1T75 = hyp2a(75)
H2countC1T75 = df_Hypothesis1C1T75['Within Threshold'].value_counts()
df_Hypothesis1C1T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C1T75.csv')

#################### Case2 hypothesis 2
df_Hypothesis2C2T25 = hyp2a(25)
H2countC2T25 = df_Hypothesis2C2T25['Within Threshold'].value_counts()
df_Hypothesis2C2T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C2T25.csv')

df_Hypothesis2C2T50 = hyp2a(50)
H2countC2T50 = df_Hypothesis2C2T50['Within Threshold'].value_counts()
df_Hypothesis2C2T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C2T50.csv')

df_Hypothesis2C2T75 = hyp2a(75)
H2countC2T75 = df_Hypothesis2C2T75['Within Threshold'].value_counts()ftttttttttttttttttt7y]
df_Hypothesis2C2T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C2T75.csv')

#################### Case3 hypothesis 2
df_Hypothesis2C3T25 = hyp2a(25)
```

```
H2countC3T25 = df_Hypothesis2C3T25['Within Threshold'].value_counts()
df_Hypothesis2C3T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C3T25.csv')

df_Hypothesis2C3T50 = hyp2a(50)
H2countC3T50 = df_Hypothesis2C3T50['Within Threshold'].value_counts()
df_Hypothesis2C3T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C3T50.csv')

df_Hypothesis2C3T75 = hyp2a(75)
H2countC3T75 = df_Hypothesis2C3T75['Within Threshold'].value_counts()
df_Hypothesis2C3T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp2C3T75.csv')

#################### Case1 hypothesis 4
df_Hypothesis4C1T25 = hyp2a(25)
H4countC1T25 = df_Hypothesis4C1T25['Within Threshold'].value_counts()
df_Hypothesis4C1T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C1T25.csv')

df_Hypothesis4C1T50 = hyp2a(50)
H4countC1T50 = df_Hypothesis4C1T50['Within Threshold'].value_counts()
df_Hypothesis4C1T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C1T50.csv')

df_Hypothesis4C1T75 = hyp2a(75)
H4countC1T75 = df_Hypothesis4C1T75['Within Threshold'].value_counts()
df_Hypothesis4C1T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C1T75.csv')

#################### Case2 hypothesis 4
df_Hypothesis4C2T25 = hyp2a(25)
H4countC2T25 = df_Hypothesis4C2T25['Within Threshold'].value_counts()
df_Hypothesis4C2T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C2T25.csv')

df_Hypothesis4C2T50 = hyp2a(50)
H4countC2T50 = df_Hypothesis4C2T50['Within Threshold'].value_counts()
df_Hypothesis4C2T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C2T50.csv')

df_Hypothesis4C2T75 = hyp2a(75)
H4countC2T75 = df_Hypothesis4C2T75['Within Threshold'].value_counts()
df_Hypothesis4C2T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C2T75.csv')
```

```
################### Case3 hypothesis 4
df_Hypothesis4C3T25 = hyp2a(25)
H4countC3T25 = df_Hypothesis4C3T25['Within Threshold'].value_counts()
df_Hypothesis4C3T25.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C3T25.csv')

df_Hypothesis4C3T50 = hyp2a(50)
H4countC3T50 = df_Hypothesis4C3T50['Within Threshold'].value_counts()
df_Hypothesis4C3T50.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C3T50.csv')

df_Hypothesis4C3T75 = hyp2a(75)
H4countC3T75 = df_Hypothesis4C3T75['Within Threshold'].value_counts()
df_Hypothesis4C3T75.to_csv(r'/Users/gopaljuneja/Desktop/CS4203/Hyp4C3T75.csv')
```

2. Source Code for Keylogger:

a. Main.pyw

```
import pynput

from pynput.keyboard import Key, Listener
#vanilla
import logging

#make a logfile
log_dir1 = ""
log_dir2 = ""

def setup_logger(logger_name, log_file, level=logging.DEBUG):
    l = logging.getLogger(logger_name)
    formatter = logging.Formatter('%(asctime)s, %(message)s')
    fileHandler = logging.FileHandler(log_file, mode='w')
    fileHandler.setFormatter(formatter)
```

```
    l.setLevel(level)
    l.addHandler(fileHandler)


setup_logger('log1', log_dir1 + "key_PandalogPress.csv")
setup_logger('log2', log_dir2 + "key_PandalogRelease.csv")
logger_1 = logging.getLogger('log1')
logger_2 = logging.getLogger('log2')


def on_press(key):
    if (str(key) == 'Key.shift' or str(key) == 'Key.caps_lock' or str(key) == 'Key.shift_r'):
        logger_1.disabled
    else:
        logger_1.info(str(key))

def on_release(key):
    if (str(key) == 'Key.shift' or str(key) == 'Key.caps_lock' or str(key) == 'Key.shift_r'):
        logger_2.disabled
    else:
        logger_2.info(str(key))
        if key == Key.esc:
            # Stop listener
            return False


#this says, Listener is on
with Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```

b. Clone.pyw

```
import pynput
from pynput.keyboard import Key, Listener
#vanilla
import logging

#make a logfile
log_dir1 = ""
log_dir2 = ""

def setup_logger(logger_name, log_file, level=logging.DEBUG):
    l = logging.getLogger(logger_name)
    formatter = logging.Formatter('%(asctime)s, %(message)s')
    fileHandler = logging.FileHandler(log_file, mode='w')
    fileHandler.setFormatter(formatter)

    l.setLevel(level)
    l.addHandler(fileHandler)


setup_logger('log1', log_dir1 + "key_ErnielogPress.csv")
setup_logger('log2', log_dir2 + "key_ErnielogRelease.csv")
logger_1 = logging.getLogger('log1')
logger_2 = logging.getLogger('log2')

def on_press(key):
    if (str(key) == 'Key.shift' or str(key) == 'Key.caps_lock' or str(key) == 'Key.shift_r'):
        logger_1.disabled
    else:
        logger_1.info(str(key))

def on_release(key):
    if (str(key) == 'Key.shift' or str(key) == 'Key.caps_lock' or str(key) == 'Key.shift_r'):
```

```
        logger_2.disabled
    else:
        logger_2.info(str(key))
        if key == Key.esc:
            # Stop listener
            return False

#this says, Listener is on
with Listener(on_press=on_press, on_release=on_release) as listener:
    listener.join()
```