Cmpe344 Fall 2021 FF367
Experiment #7: Hardware Security

Task 1. Reading from Cache versus from Memory

- Fill the table with the number of CPU cycles of each element in each run.

Array Element\Run #	0	1	2	3	4	5	6	7	8	9
0*4096	101	65	99	114	320	22	83	83	129	179
1*4096	139	134	198	134	154	192	180	158	140	146
2*4096	140	446	152	223	134	142	175	146	154	150
3*4096	50	136	51	151	152	49	164	63	50	49
4*4096	136	209	150	152	152	154	152	1467	154	152
5*4096	136	142	138	138	152	254	158	166	133	153
6*4096	154	150	140	215	151	150	144	165	162	152
7*4096	50	142	217	53	53	156	152	132	55	56
8*4096	144	152	164	148	152	172	140	130	133	136
9*4096	294	146	189	177	150	172	170	161	134	154

* Add columns to the right if you run the program more than 10 times

- Is the access of array[3*4096] and array[7*4096] faster than that of the other elements? Describe your observations.

  Yes, in general access of array[3*4096] and array[7*4096] is faster than that of the other elements. Because we access those items before the loop and therefore add them to the cache memory.

- What is the threshold that you predict for memory vs cache access?

  When calculated, it can be seen that average number of CPU cycles for cache access is approximately ~100 cycles. For cache access it takes more higher than 100 cycles to complete. Therefore 100 cycles seems to be a appropriate threshold value.

Task 2. Using Cache as a Side Channel

- How many times did you run the program and how many times you were able to guess the secret correctly? Fill the table to trace your work.

  We used 100 as threshold as we explained in the previous question.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
T/F	T	T	T	T	F	T	F	T	T	T	T	T	T	F	T	T	T	T	T	T

  * Add columns to the right or copy/paste the table if you run the program more than 20 times

  3 False – 17 True

Task 3. Out-Of-Order Execution

- What happened when you changed *victim(i)* to *victim(i+20)*? Why do you think that happened?

It didn't produce any output, in other words, it didn't find any value in the cache (or couldn't find within the threshold). Because with i+20 we trained our CPU to not take the branch, therefore we couldn't utilize the out-of-order execution. In victim(i) case, we were able to train the CPU to take the branch.

- What happened when commented the 62nd line? Why do you think that happened?

  Again, it didn't produce an output. As stated in the description, if **size** is not in the CPU caches, it may take hundreds of CPU clock cycles before that value is read. If we don't clear the **size** from the cache memory, we can't utilize the out-of-order execution, simply because we don't need to. Checking the conditional statement takes much less than threshold value.

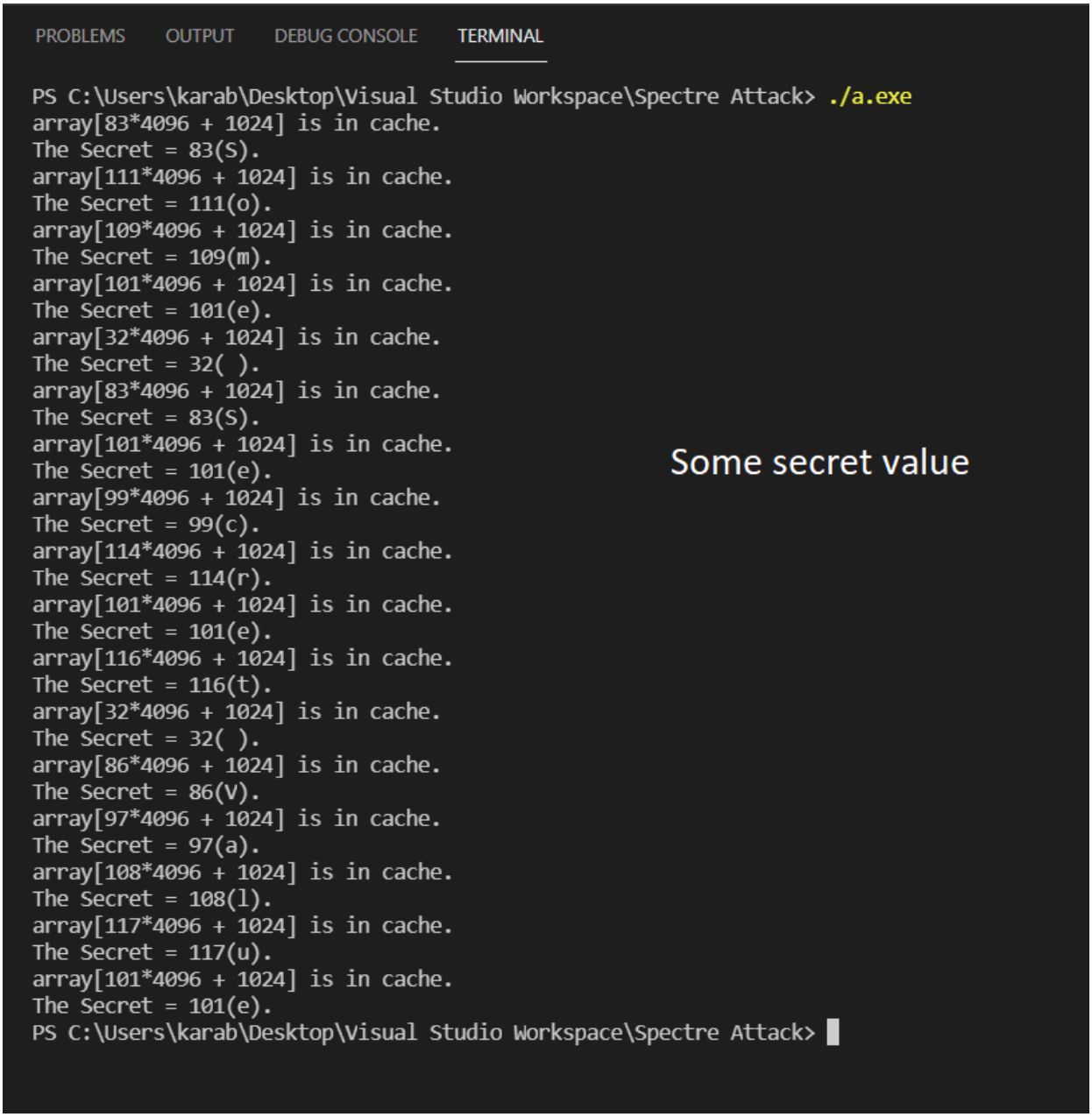
Task 4. Execute the Attack

- Describe your observation and note whether you are able to steal the secret value. Also state how many executions it took you to steal the secret value, if you did.

Yes, we were able to steal the first character of the secret value (S). It took 2 times to steal the secret value. In the first try, we encountered with (0).

Task 5. Steal the Entire Secret String

In the previous task, we just read the first character of the secret string. In this task, we need to print out the entire string using the Spectre attack. Please write your own code or extend the code in Task 4; include your execution results in the report. We added a loop to the main function in order to reach each character of the secret value. After successive tries, we successfully extracted the secret value. Screenshot regarding the successful output of the program can be found below:


PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PS C:\Users\karab\Desktop\Visual Studio Workspace\Spectre Attack> ./a.exe
array[83*4096 + 1024] is in cache.
The Secret = 83(S).
array[111*4096 + 1024] is in cache.
The Secret = 111(o).
array[109*4096 + 1024] is in cache.
The Secret = 109(m).
array[101*4096 + 1024] is in cache.
The Secret = 101(e).
array[32*4096 + 1024] is in cache.
The Secret = 32( ).
array[83*4096 + 1024] is in cache.
The Secret = 83(S).
array[101*4096 + 1024] is in cache.
The Secret = 101(e).
array[99*4096 + 1024] is in cache.
The Secret = 99(c).
array[114*4096 + 1024] is in cache.
The Secret = 114(r).
array[101*4096 + 1024] is in cache.
The Secret = 101(e).
array[116*4096 + 1024] is in cache.
The Secret = 116(t).
array[32*4096 + 1024] is in cache.
The Secret = 32( ).
array[86*4096 + 1024] is in cache.
The Secret = 86(V).
array[97*4096 + 1024] is in cache.
The Secret = 97(a).
array[108*4096 + 1024] is in cache.
The Secret = 108(l).
array[117*4096 + 1024] is in cache.
The Secret = 117(u).
array[101*4096 + 1024] is in cache.
The Secret = 101(e).
PS C:\Users\karab\Desktop\Visual Studio Workspace\Spectre Attack>
Some secret value