# Report

**Design And Implementation:**

1) First of all I have created a hook in **execve** system call and find out the testcase process and I stored the process id in variable for further filtering.

2) I have created trace point on openat system call and filter out the testcase process with its stored pid. And then comparing filename if filename **= /tmp/ready_to_checkpoin**t.

Now first let see How am I storing the context then we will move to next step.

**How am  I stroring the context?**

I have created a hash map. The key will be virtual address and value will be a structure which contain string of 256 characters. Which will store the data.

3) I have created hook point at mmap system call and I saw that whenever anonymous memory will be allocated its file descriptor fd field have value = 4294967295. So I am filtering out anonymous memory allocated by mmap with this field.

**How to get range of mmap?**

I have created **sysenter mmap** and **sysexit_mmap** tracepoint. From sysenter I am capturing the **len** field which will say length in bytes.and in sysexit tracepoint I will capture the **return value** which is the start address from which address is allocated.

**How This 3 steps are running?**

Once I got the process id of testcase process I will set the flag which will filter out events which are done by testcase in each hook point (openat,sysenter_mmap and sysexit_mmap).

So whenever mmap called if it is anonymous memory I will calculate the range and I will store the address in a HashMap.

In hashmap I will take key = virtual address. And value = A structure which have one element a char array of 256 bytes. Which will store the content of address range [key,key+256).

Let say I got range [a,b) then I will store content on granularity of 256 bytes so will store content like this.

| Key | value |
|---|---|
| A | struct of char[256] initialized with 0 |
| A+256 | struct of char[256] initialized with 0 |
| ……. | |

So I got the entire range of address with its content initially I store the content as 0.

If I have more then one mmap anonymous memory then it will run more then one time and it will be stored in same hash map.

So before we go for ready for checkpoint we have all the anonymous virtuall address in our hashmap which is allocated by testcase and the content is initialized by 0.But in hashmap we have all the entries.

4) Now when ready to checkpoint file is created I will set a flag which will trigger one callback function for each element of this HashMap.

   Hashmap have (key,value) pair here key = virtual add of a process and value = structure which have one element char[256].

   Now in that callback function I will use bpf_probe_read_user function which will read from the key(vitual address) and write 256 bytes of data into value(char[256]).

   This callback function run for each element of the map.

5) now when it is done I will set the read done flag = 1, So It will trigger user space program to create checkpoint completed file.

6) Now when testcase process is ready to restore I have hook I will call again one callbackfunction on each element of this map.

   I will use bpf_probe_write_user function which will write the content of char[256](value) of 256 bytes into virtual address(key).

   After completing this callback on all the hash map elelement I will set one more flag which will give signal to testacase that restore is complete.

Challenges :

1) I faced issue in comparing the strings in bpf code because we can’t generally put arbitrary loop in bpf code it should be bounded.
2) I also faced some difficulties in using bpf_probe_read/write_user function.bpf code has limited stack size of 512B so first I have plan to do this logic in page granularity but I can’t allocate variables more the 512 bytes so that’s why I have char array of 256 bytes not 4096 bytes.
3) I also faced some difficulties to filter out what kind of memory is anonymous and how to detect it is anonymous. So I did strace on testcase process and find out that fd whenever mmap is called for anonymous memory it will set fd=-1 and in my system it will be set as 4294967295.