Agentless Introspection of Bare Metal Servers using BMI
Sprint - 2

Vision and Goals:
Create an end to end agentless introspection system for the bare metal environment. Goals of the project are:

- To create a system that would extend the existing agentless crawler to get information of the bare metal systems within a multi-tenant environment.
- Introspection of the frames (system information) generated by the crawler to check possible vulnerability or security threats in a system.
- Providing the introspected data to various annotators that are implemented by IBM.

Personas/Users of the Project:

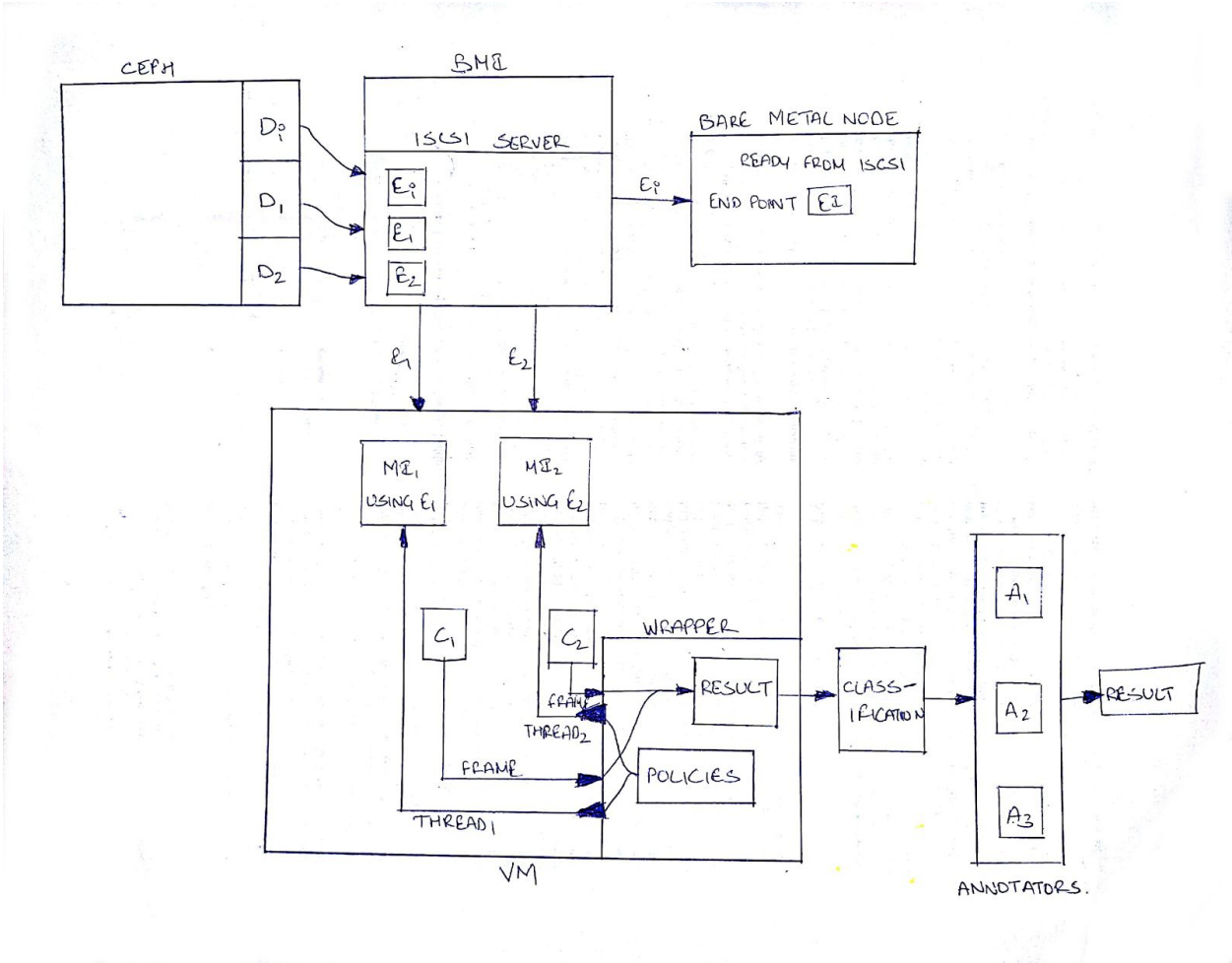
- Bare metal Agentless Introspection would be used by bare metal service providers to gather information regarding user environment on these servers without introducing any noise within data.
- Users in the bare metal environment can register for this service and obtain the introspected data to identify any vulnerability within their systems.

Implementation:

- Main components :
    - Ceph Storage
    - Disk Snapshots using BMI
    - TGT
    - Wrapper
        - Python System Crawler
    - Implementation diagram
    - Introspection

- Ceph Storage :
    - It provides a highly scalable object, block, and file-based storage under a unified system.
- BMI :
    - We use this technology to get the snapshot of a remote disk of a Bare Metal System that can be hosted anywhere.
- TGT :
    - Linux SCSI Target Framework acts as a layer between ceph storage and our application.
    - It allows us to get the disk images from Ceph as ISCSI End point and mount it on the VM where this service is running.
    - It also gives us the path where these images are mounted.

- Wrapper :
    - This is will allows us to run the Agentless System Crawler on a target mounted image and get features.
    - The Crawler takes in parameters like
        - Mount Image url: location of stored image, we get it from TGT
        - Features: `os,disk,process,connection,metric,package,file,config`
        - Frequency: time after which crawler should scan for information again
        - And many more ….
    - All these parameters are decided on basis of the Policies (conditions) which exist inside the wrapper.
    - Wrapper can run multiple crawler at a time in a multithreaded environment and makes sure that nothing is failing and if it does then re running it.
    - Wrapper also collect data from all these Crawlers and combines the results.
    - All combined result are then classified into categories that are sent to different annotators discussed in previous section.
- Implementation Diagram :



- Introspection Parameters :

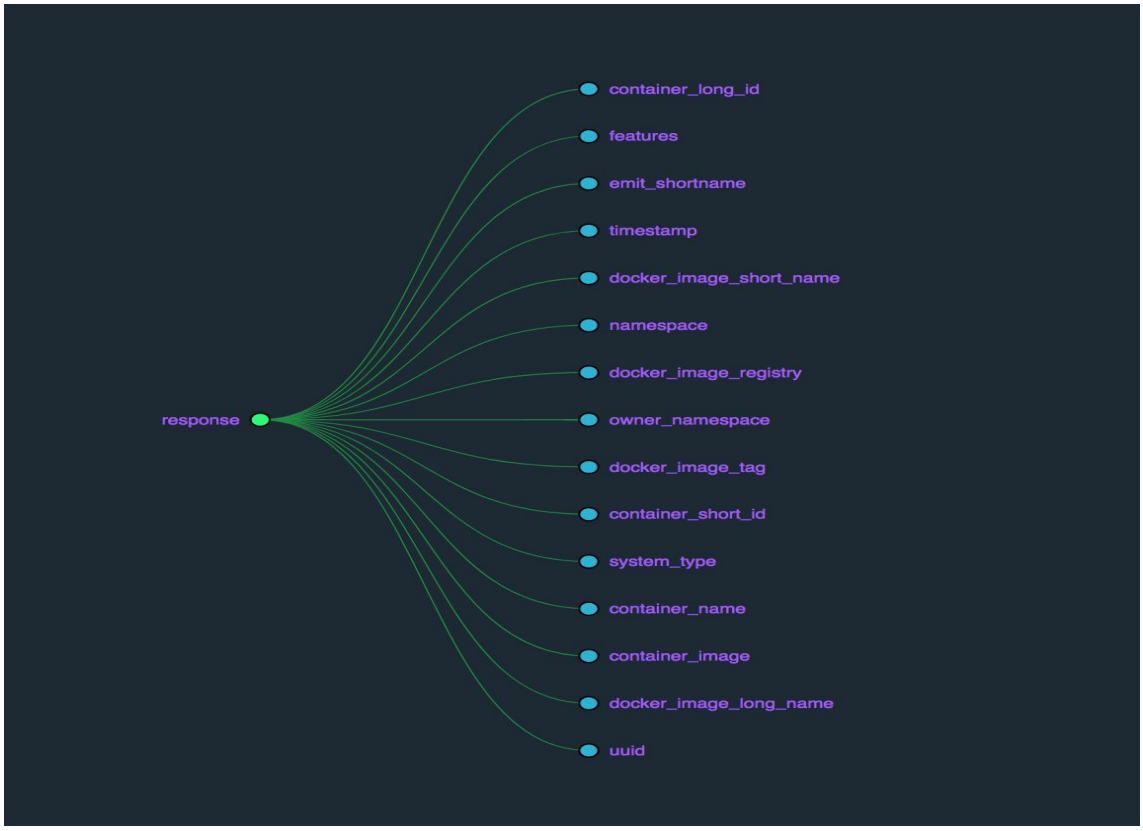


Figure: 3

- Figure 3 depicts all the parameters that we are crawling for introspection. We are crawling Operating system name, version, packages, network information of the disks that we have mounted from CEPH storage.

Output snippet of crawler:

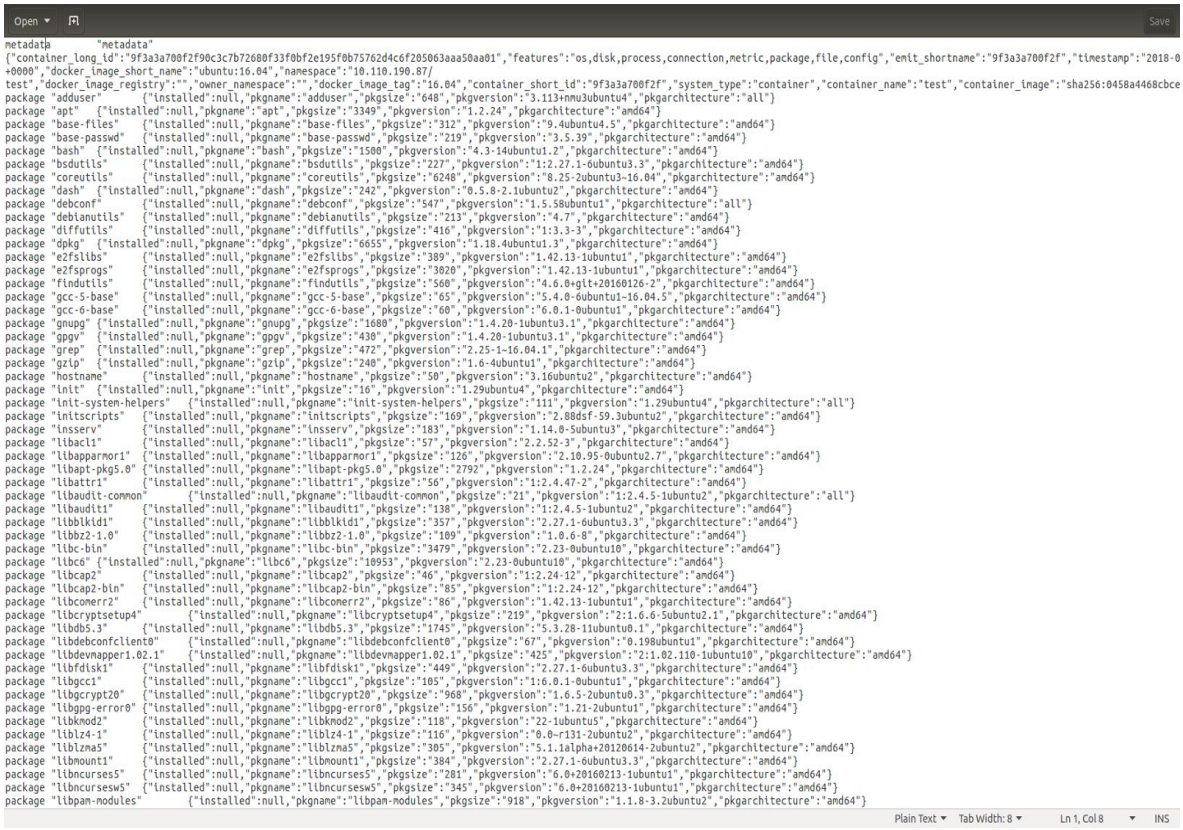


Figure: 4

- Figure 4 depicts the output of the crawler. We have used container to run the crawler image and we are getting packages and system information of the current VM that is installed in the machine to get the understanding of the agentless system crawler and how can we use the crawler to abstract information from the disks.
- After collecting all the system information the data will be passed to annotators like config annotators, vulnerability annotator, compliance annotator and password annotator
- For example if the system is mis-configured based on the information that we have provided then that system information will go to secure configure analysis for further analysis. IBM has this introspection system implemented for us.

1. Work till now:
    - Right now, we do not have access to CEPH storage. We are working on getting the access. Right now we are trying to create our own iSCSI target and mount it to another machine for further analysis of the disks.

- In order to try mounting images at a destination we used QEMU which is a emulator and allows us to mount a raw image as a disk image and then we can crawl it by the given crawler script. In this step we got results as shown.


```
{
    "partitionname": "/dev/sda1",
    "freepct": 31.400000000000006,
    "fstype": "ext4",
    "mountpt": "/cgroup",
    "mountopts": "ro,relatime,errors=remount-ro,data=ordered",
    "partitionsize": 8319852544,
    "feature_type": "disk",
    "namespace": "10.0.2.15/unruffled_raman"
}
```


Release Planning

SPRINT 1:

- Understanding the goals and features of the project.
- Understand the motivation behind building an agentless system.
- Read and understand the research papers to grasp the concepts of the existing architecture of Bare Metal Imaging system.

SPRINT 2:

- Work on installing and executing agentless crawler for the virtualized environment on a local system.
- Explore the frames and understand the various parameters acquired from the crawled data.
- Work with QEMU and Docker containers to understand operations of the crawler.

SPRINT 3:

- Understand and get access to TGT iSCSI target endpoint on BMI.
- Connect with the iSCSI endpoint on BMI to mount the image on a virtual machine.

SPRINT 4:

- Create a wrapper for the agentless crawler to run crawler instances on each mounted image.
- Decide the various policies to enforce on the wrapper for crawler execution.
- Perform testing on the wrapper and ensure that it executes properly on every mounted image.

SPRINT 5:

- Implement multi-threading in the wrapper to crawl multiple disk images in parallel.
- Perform testing on the built system to validate the resultant frames.

SPRINT 6:

- Classify and consolidate the frames acquired from various crawled disk images and pass them to different annotators for introspection.

SPRINT 7:

- Test the end to end system on a set of sample disk images and examine their results.
- Deploy the project on the MOC and prepare a final project demo.