Attack Graph Generation for Micro-service Architecture

Stevica Bozhinoski[1] and Amjad Ibrahim[2]

***Abstract*—**

I. Introduction

II. Related Work

What makes us different to previous work: - To the best of our knowledge, there is no work that has been done for attack graph generation for docker containers. - The paper(Automated Generation of Attack Graphs Using NVD) does not provide an example/ or clearly define the attack graph generation process. Their main focus is to evaluate attack rules. They dont use Clair as well. They do not offer any performance comparison between different topologies.

III. Architecture

1) Breadth-first search: We use a modification of the breadth first search algorithm to find the nodes and the edges.

The algorithm requires the goal container, the topology and a dictionary of the exploitable vulnerabilities as an input and the output is made up of the nodes and the edges that make the attack graph. The algorithm first initializes the nodes, edges, queue and the passed nodes. Afterwards it generates the nodes which are a combination of the image name and the exploitable vulnerability. Then into a while loop we iterate through every node, check its neighbours and add the edges. If the neighbour was not passed, then we add it to the queue. The algorithm terminates when the queue is empty.

```
Data: goal_container, topology,
      container_exploitability
Result: nodes, edges
nodes, edges, queue, passed_nodes = list(), dict{},
  Queue(), [];
queue.put(goal_container);
nodes = get_nodes();
while ! queue.isEmpty() do
    ending_node = queue.get();
    passed_nodes[ending_node] = True;
    cont_exp_end =
      container_exploitability[ending_node];
    neighbours = topology[ending_node];
    for neighbour in neighbours do
        if neighbour == ”outside” then
            edges.append(create_edges());
            continue;
        end
        if ! passed_nodes[neighbour] then
            queue.put(neighbour);
        end
        if neighbour == goal_container then
            continue;
        end
        edges.append(create_edges())
    end
end
```

Algorithm 1: Breadth-first search algorithm for generating an attack graph.

IV. Experiments

V. Conclusion

Acknowledgment

References

[1] Merkel, Dirk. ”Docker: lightweight linux containers for consistent development and deployment.” Linux Journal 2014.239 (2014): 2.