Benny Tsai
600.424
Network Security
Assignment 0

1 Traceback

TracebackRouter extends the Router class. It modifies the forward() method to mark packets according to the edge sampling marking procedure, and then passes the packet to the original forward() method. The probability p of marking a packet is set to 1/25 (0.04).

Traceback is an Application plugin that performs edge sampling path reconstruction. The start_tb() method starts a trace by placing a BPF rule matching all incoming packets, and also initializes state variables. The stop_tb() method removes the BPF rule.

The heart of the operation lies in the traceback() method (can be called from UI via either “traceback” or “tb”). The method runs the path reconstruction algorithm and then outputs the results to the UI. The path reconstruction algorithm is detailed in the following pseudo-code:

```
for each distance d from 0 to max_distance
    for each packet p at distance d
        let z be the trace field of p
        if d = 0 /** z = origin of p */
            increment count of packets from z
            add edge in graph G from z to victim
            add z as a possible origin of attack paths
            add z to set of nodes at distance d
        else if d > 0
            for each node n at distance d – 1
                z = z XOR n /** XOR z with n to get origin of p */
                if z is an actual node in the network
                    increment count of packets from z
                    add edge in graph G from z to n
                    add z as a possible origin of attack paths
                    remove n as a possible origin of attack paths
                    add z to set of nodes at distance d
```

After this algorithm completes, we have the count of packets from each node, a set of nodes that are the origins of attack paths, and graph G which contains paths from these origins to the victim. traceback() then displays these paths, along with count of packets from nodes on the path, with help from TracebackPath. TracebackPath extends the Path class from Dr. Scheinerman’s graph package, modified to return an array of nodes on a path rather than printing them out.

Testing was done via the script tb1.script. The script launches attacks on PARC from MIT, JHU, and CMU. After these attacks conclude PARC uses traceback to reconstruct the attack paths.

2 Defeating Traceback

My attack was inspired by the paper's comment that while an attacker cannot fake edges between itself and the victim because of the distance incrementation, it is possible for an attacker to fake edges farther from the victim than itself. Without authentication mechanisms to check who marked a packet, in general it is possible for an attacker to fool the edge marking traceback scheme into reconstructing attack paths from the victim to any host whose path to the victim contains the path from the attacker to the victim. An example will hopefully make this more clear.

In the test.net topology, FLORAM sits on the only path from any host to PARC. If an attacker at FLORAM launches a flood attack on PARC, and marks some packets as if they were marked by the SPRINT or GOODNET routers, then PARC will be led to reconstruct an attack path originating from GOODNET. This creates the illusion that the attack originated from MIT, rather than FLORAM.

TracebackAttack, an Application plugin, implements only the simplest case of this setup, where the attacker is directly attached to the same router the victim is attached to. In the test.net topology this is demonstrated by the host pairs JHU+ISI and CERT+CMU. TracebackAttack's ta() method launches the traceback-defeating attack.

ta() takes the same arguments as PacketGenerator's flood() method. ta() first uses traceroute to determine the path from the attacker to the host who will appear to be the attacker on the fake attack path. Since the attacker is adjacent to the victim, this will be the same path as the path from the victim to the fake attacker. ta() then launches a flood attack on the victim, and simulates the fake attack path on the flood packets, so that they are marked exactly as if they traversed the fake attack path.

Testing was done via the script tb2.script. The script launches an attack on JHU from ISI. After the attack concludes JHU uses traceback to reconstruct the attack path, and it will appear as if the attack originated from FLORAM.