Patrick Austin

CS 445 Homework 1

10 February, 2018

1a & 1b.) First, note this diagram of the sample network topology on which all tests discussed in this document were run. The victim was specified as router 0 and the users and attackers were tested as routers 1-5. Source code and input files are attached for verifying my results with this topology.

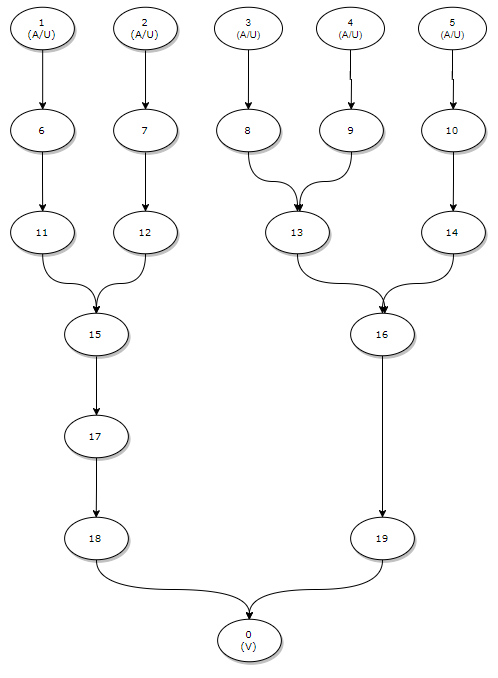


Figure 1: The sample network topology, consisting of 20 routers, used to generate the data gathered in this report. The victim was located at router 0 and tests were run with users and attackers variously located at routers 1 through 5.

In terms of results, I first observe that edge sampling was completely robust against cases with multiple users and a single attacker. Node sampling was robust in most cases, but for high values of p it was possible for the algorithm to get stuck for long periods as large amounts of marked traffic from the legitimate user at nodes close to the victim impeded the ability to sort the table of marked packets for the true attack path.

Due to these erroneous results causing large outliers in the data I have opted to show the more consistent single attacker, single user case in the graphs for this question. Single attacker single user was the case both algorithms were robust against and so produced the most clean and illustrative results.

In order to compare the accuracy of traceback (which I define as the total number of packets transmitted to the victim before my implementations of node and edge sampling identify the correct attack path) between node and edge sampling, with various choices of parameter values, see Figures 4 and 5.

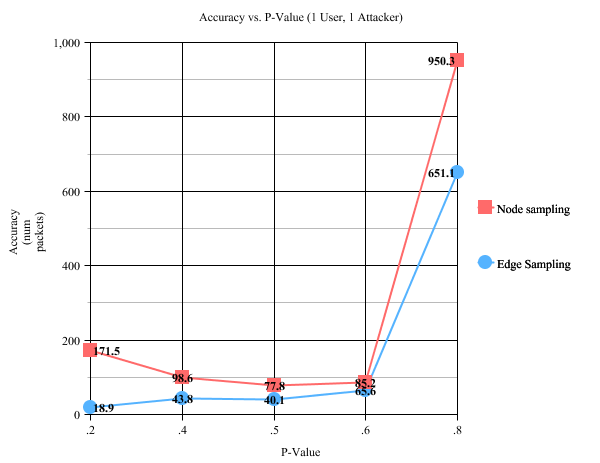


Figure 4: A comparison of traceback accuracy between node and edge sampling for varying values of p, the probability a node will be marked by a router. Accuracy values are averaged from 10 runs. Simulations with a single attacker at node 3 and a normal user at node 4, with a fixed x value of 100, were used to generate this graph.

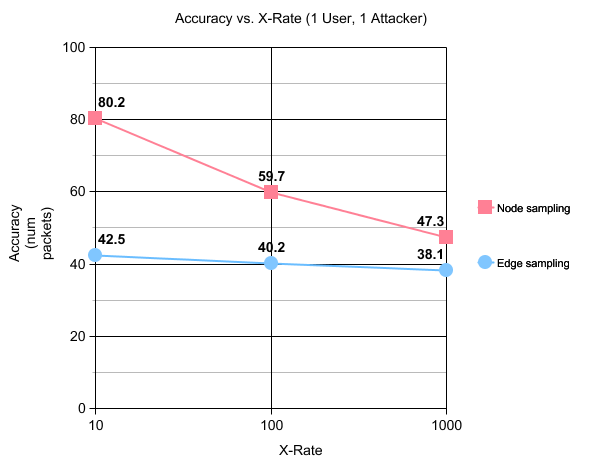


Figure 5: A comparison of traceback accuracy between node and edge sampling for varying values of x, the factor by which an attacker’s traffic exceeds that of a normal user. Accuracy values are averaged from 10 runs. Simulations with a single attacker at node 3 and a normal user at node 4, with a fixed p value of .5, were used to generate this graph.

To comment briefly on these results and explain their physical significance, I first need to explain how I defined the “stop state” of these algorithms. In my interpretation, node sampling stops when the top entries in the table of marked packets and router IDs, sorted into descending order of marked packets, matches the actual attack path. Edge sampling stops when all complete attack paths exist within the path tree that has been constructed from marked packets.

With this taken into account, it should first be noted that edge sampling outperforms node sampling for all tested values. This is an unsurprising result since node sampling is the more robust, sophisticated algorithm, and in some sense it has an easier stop condition to reach- it doesn’t need a correct hierarchy of sorted packets to reconstruct the path, as node sampling does.

I also note the more pronounced gaps in performance for high and low values of p, as well as low values of x. While node sampling prefers a p value between .4 and .6, edge sampling performs best with a very low p value such as .2. This makes physical sense, as node sampling needs a sortable hierarchy of marked packets, where edge sampling can make do with as little as one marked packet for each edge on the attack path. Both algorithms prefer higher x values, although edge sampling is more robust to low values for x- this, too, can be attributed to the design of the edge sampling algorithm, which in theory needs only singleton marked packets for each relevant edge in order to complete the attack path.

Both algorithms perform very poorly with high p values, which makes physical sense as well. Very high values of p mean that nodes nearer to the victim are very likely to mark and thus overwrite the data that might be coming in from more distant nodes; on average this leads to more packets that have to be sent before needed information can get through the “loud” nearby gateways without being overwritten by those gateways marking.

2. In some respects this question has already been treated above, I think, and the graphs show accuracy rates explicitly. Edge sampling outperforms node sampling in general, and I have discussed some reasons why this makes sense.

I do want to note, however, that while this may be the ‘accurate’ result, probabilistically speaking there is still some question of how confident one can be that these are the true attack paths. As the authors discuss in the paper we read for this class on edge and node sampling, reaching a high degree of confidence in the correctness of the traceback may require dramatically more packets to be received.

I have also set aside concerns about how attack paths should be recovered from the tree generated in edge sampling; it may be that I am overly optimistic about edge sampling due to the need for robust measures to retrieve attack paths from a tree in a far more complex real-world network scenario. If paths need to be extracted from the tree by e.g. counting marked packets as in node sampling, more packets will probably be needed and the algorithm will perform less favorably.

3. Adding a second attacker to the scenario immediately serves to show the weakness of node sampling. In fact, in my implementation node sampling proved utterly unable to resolve scenarios with multiple attackers. For that reason Figures 6 and 7 show only edge sampling; since node sampling never found the correct attack path no matter how much time it was given, we could call node sampling’s accuracy value infinite. Therefore I opted to omit it from the graphs entirely.

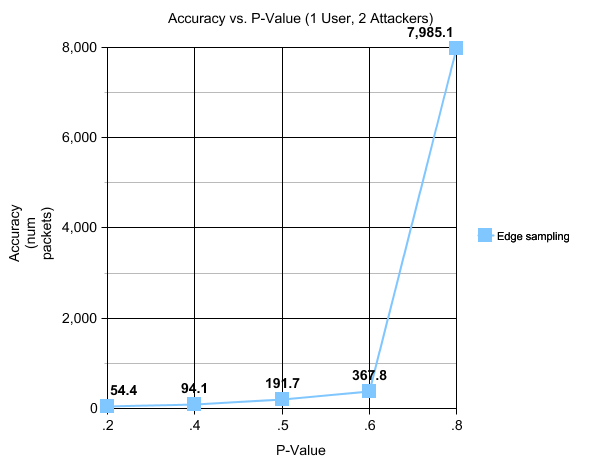


Figure 6: A graph showing traceback accuracy for edge sampling for varying values of p, the probability a node will be marked by a router. Accuracy values are an average of 10 runs. Simulations with attackers at nodes 2 and 3 and a user at node 4, with a fixed x value of 100, were used to generate this graph.

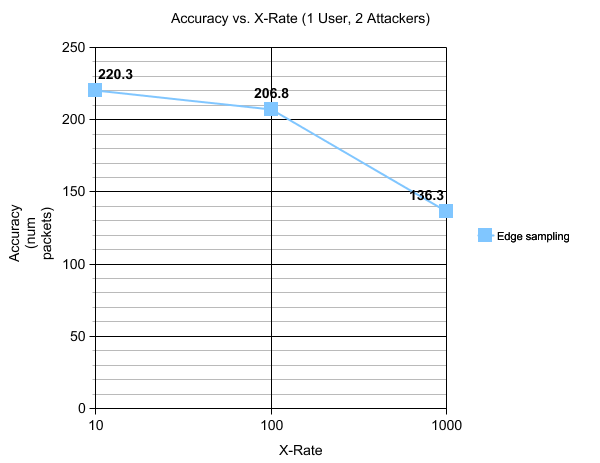


Figure 7: A graph showing traceback accuracy for edge sampling for varying values of x, the factor by which an attacker’s traffic exceeds that of a normal user. Accuracy values are an average of 10 runs. Simulations with attackers at nodes 2 and 3 and a user at node 4, with a fixed p value of .5, were used to generate this graph.

To comment briefly on these results and explain their physical significance, I first repeat the important point that node sampling’s inability to handle cases with multiple attackers has led to its omission from these graphs. Since most real world attack cases involve distributed attacks, this is something of a nail in the coffin for node sampling.

These graphs also bear repetition of some of the points mentioned in part 1. Edge sampling as I have implemented it prefers low values of p and high values of x. This makes physical sense due to even single marked packets for each needed edge being sufficient; the faster these can be delivered (higher x) and the less likely they are to be overwritten by nearby nodes (lower p), the faster the process will be able to complete.

To get to more novel results from these graphs, I first note that the two attacker cases took longer to resolve than the single attacker cases shown in Figures 3 and 4 for all matching p and x values- usually several times longer, in fact. Once more, this makes physical sense. More edges had to be received to build two complete attack paths instead of just one; more marked packets being necessary is quite logical. This exacerbates the problem at very high levels of p- more distant edges will need to get past the gatekeeping nearby nodes. And higher throughput, in terms of x value, will still lead to faster construction of the attack paths.

Since edge sampling is fully robust against cases with multiple users, as discussed in part 1, incorporating arbitrarily more users into the two attacker scenario can be done with minimal issue and only minor damage to accuracy. See Figure 8 for console output from a successful simulation run with 2 attackers and 3 users.

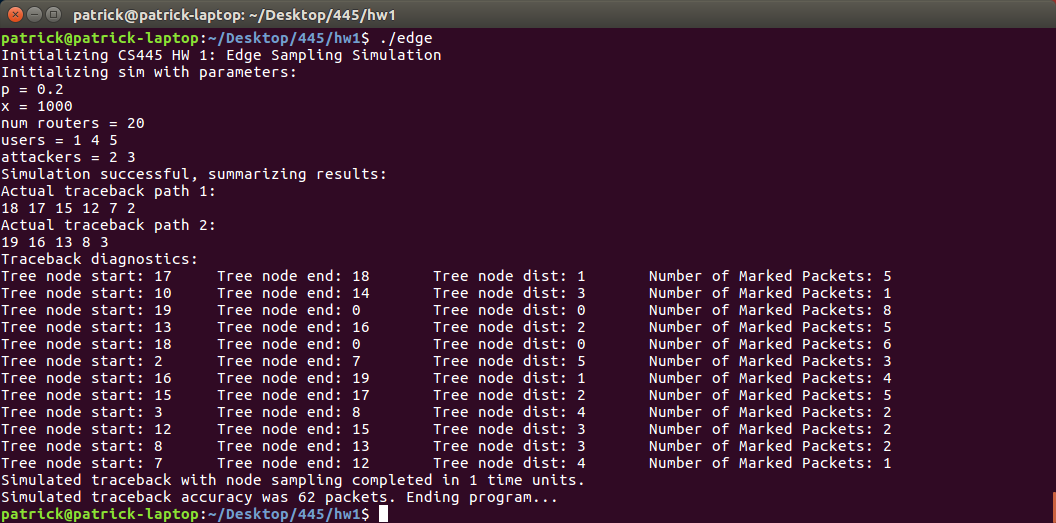


Figure 8: Simulation output for edge sampling with two attackers and three users. The edge sampling algorithm is not significantly hindered by the introduction of additional normal users.