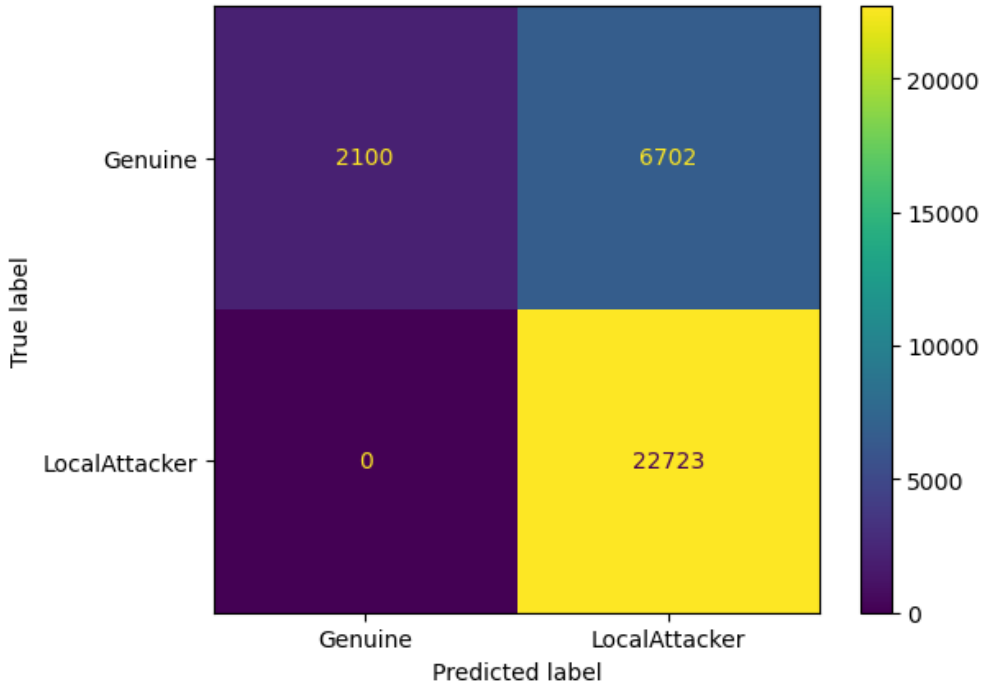
Report 2 - Marshall Thompson

Part 1



TOTAL MESSAGES=31525

True Positive, False Positive, False Negative Rates

```
TP: 23.831936631844794
FP: 0.10519856228631541
FN: 76.1680633681552
```

Part 2

Attack Type	True Positive	False Positive	False Negative
A5	100%	0.0032%	0.0%
A6	100%	0.0033%	0.0%
A7	100%	0.0%	0.0%
A8	100%	0.0%	0.0%
A9	100%	0.087%	0.0%
A10	100%	0.21%	0.0%
A11	100%	0.25%	0.0%
A12	99.65%	0.0%	0.34%

Attack Type	True Positive	False Positive	False Negative
A13	100%	0.0%	0.0%
A14	100%	0.59%	0.0%
A15	100%	0.21%	0.0%

Part 3

A

PositionConsistancyCheck:

```
double LegacyChecks::PositionConsistancyCheck(veins::Coord* curPosition,
    veins::Coord* oldPosition, double time)
{
    double distance = mdmLib.calculateDistancePtr(curPosition, oldPosition);

    if (distance < MAX_PLAUSIBLE_SPEED * time) {
        return 1;
    }
    else {
        return 0; //distance
    }
}
```

The PositionConsistancyCheck makes sure that a vehicle does not move more than should be possible for a Genuine car. If the distance that a car moves is more than the `MAX_PLAUSIBLE_SPEED * time`, then it is moving faster than a Genuine car should.

SpeedConsistancyCheck

```
double LegacyChecks::SpeedConsistancyCheck(double curSpeed, double oldspeed,
    double time)
{

    double speedDelta = curSpeed - oldspeed;

    //    double attFact = mdmLib.gaussianSum(1, 1 / 3);
    //    if (time >= 1) {
    //        attFact = time;
    //    }

    if (speedDelta > 0) {

        if (speedDelta < MAX_PLAUSIBLE_ACCEL * time) {
            return 1;
        }
```

```
        else {
            return 0; //distance
        }
    }
    else {

        if (fabs(speedDelta) < MAX_PLAUSIBLE_DECEL * time) {
            return 1;
        }
        else {
            return 0; //distance
        }
    }
}
```

This function makes sure that the acceleration/deceleration of a car is not greater than what is plausible for a Genuine car. If the acceleration/deceleration is greater/less than what is plausible, then it is flagged as possibly a non-Genuine car.

PositionSpeedConsistancyCheck

```
double LegacyChecks::PositionSpeedConsistancyCheck(veins::Coord* curPosition,
    veins::Coord* oldPosition, double curSpeed, double oldspeed, double time)
{
    if (time < params->MAX_TIME_DELTA) {
        double distance = mdmLib.calculateDistancePtr(curPosition, oldPosition);
        double curminspeed = std::min(curSpeed, oldspeed);
        //double theoreticalSpeed = distance / time;
        double retDistance[2];
        mdmLib.calculateMaxMinDist(curSpeed, oldspeed, time,
            MAX_PLAUSIBLE_ACCEL, MAX_PLAUSIBLE_DECEL, MAX_PLAUSIBLE_SPEED,
            retDistance);

        double addon_mgt_range = params->MAX_MGT_RNG_DOWN + 0.3571 * curminspeed -
0.01694 * curminspeed * curminspeed;
        if (addon_mgt_range < 0) {
            addon_mgt_range = 0;
        }

        double deltaMin = distance - retDistance[0] + addon_mgt_range;
        double deltaMax = retDistance[1] - distance + params->MAX_MGT_RNG_UP;

        if (deltaMin < 0 || deltaMax < 0) {
            return 0;
        }
        else {
            return 1;
        }
    }
    else {
        return 1;
```

```
        }
    }
```

This function combines the checks from the previous two and checks whether both speed and position remain consistent from the current message to the previous message.

Three Attacks These would be useful against

1. A7: If the vehicle is moving at a random speed, it could "choose" a speed so much greater or less than its current speed that it exceeds the maximum acceleration in the `SpeedConsistancyCheck`.
2. A8: In a similar vein to (1), if a vehicle is moving at a random speed, the distances between the current and previous reporting could be much farther apart than possible failing `PositionConsistancyCheck` or the `SpeedConsistancyCheck` if the change of speed(acceleration) is too great.
3. A15: DoSDistruptive floods the network with random previous messages. Since an element to A15 is random, this could lead to similar issues seen in (1) and (2) leading to the triggering `SpeedConsistancyCheck` or `PositionConsistancyCheck`. If both speed and position change(I.E. they do not remain consistent from the previous randomly chosen message and the current randomly chosen message) then the `PositionSpeedConsistancyCheck` would also fail.

B

If we decrease the threshold that means that it is easier for the checks to fail. I.E., fewer messages have to be plausibly an attacker for the check to fail. This threshold is a parameter that should be optimized over. If we decrease it too far, than everything would be an attacker, and we would have many false positives. However, if its properly tuned, sitting at a sweet spot of not too many false positives while still having a high rate of true positives and high accuracy, we would catch more attackers.