Poster: Bypassing Control Invariants-Based Defenses in Autonomous Vehicles

Yinan Zhao
Waseda University
yinanzhao@nsl.cs.waseda.ac.jp

Ziling He
Waseda University
ziling@nsl.cs.waseda.ac.jp

Tatsuya Mori
Waseda University
mori@nsl.cs.waseda.ac.jp

***Abstract*—Autonomous Vehicles (AVs) have been increasingly attacked by hackers. However, the system security of AVs is of major importance, since any successful attack can lead to severe economic loss, equipment damage, or even loss of human life. A good security principle for evaluating new algorithms is to show that the proposal is resilient against a powerful adversary. So in this poster, we realize the worst type of of attacks, called stealthy attacks, against steering control system which is important to lateral control for AVs. The core of our proposed stealthy attacks is the use of Model Predictive Control (MPC), State-space Model (SSM), System Identification (SI) and Dynamic Time Warping (DTW) to allow attackers to accurately simulate system behavior-allowing them to perform undetectable attacks.**

I. Introduction

Autonomous vehicles (AVs) rely heavily on sensor data for navigation, raising concerns about their vulnerability to data manipulation and the need for robust data validation. To improve the security of AVs, Choi et al. [1] developed a control invariant (CI) framework to detect such threats. CI framework extracts control invariants by jointly modeling a vehicle's physical properties, its control algorithm and the laws of physics. Recognizing CI framework's reliance on physical models for defense, our research challenges these defenses by developing stealthy attacks. The proposed attack strategy is to apply perturbations that are undetectable by CI framework, striking a balance where the induced deviations do not exceed the anomaly detection threshold. At the core of our approach is the use of Model Predictive Control (MPC), System Identification (SI), Dynamic Time-warping (DTW) and State Space Model (SSM) to allow attackers to accurately simulate system behavior-allowing them to perform undetectable attacks on PI-IDS-secured AV systems.

In this preliminary study, we present our progress in how to successfully implement stealthy attacks against steering control system.

II. Attack Model

The attack model assumes that the attacker is able to

- Interact with the CAN network to read and write messages through the OBD-II port or other hardware attachments.
- Know the start point and destination of the ego vehicle and list all the possible routes the vehicle may follow to implement MPC control.
- Have full knowledge of SSM, MPC, SI and DTW technique.
- Inject tailored messages into the vehicle's network to stealthily control the vehicle.

Network and Distributed System Security (NDSS) Symposium 2024
26 February - 1 March 2024, San Diego, CA, USA
ISBN 1-891562-93-2
https://dx.doi.org/10.14722/ndss.2024.23045
www.ndss-symposium.org

III. Methodology

Stealthy attacks means the caused deviations are within the expected thresholds during an AV mission which do not cause any immediate unexpected behavior. The implementation of stealthy attacks against steering control mainly consists of the following steps. First, using MPC to identify the desired steer, current states and next expected states. Then, exploiting SI in MATLAB to instantiate the state space model. Finally, deciding the error threshold by using DTW technique. The more detailed procedure is as follows.

A. State Space Model

State space model [2] [3] describes the probabilistic dependence between the latent state variable and the observed measurement, which consists of the state equation (1) and the output equation (2):

$$x(t+1) = Ax(t) + Bu(t) \tag{1}$$

$$y(t) = Cx(t) + Du(t) \tag{2}$$

where $u(t)$, $y(t)$, $x(t)$, and $x(t+1)$ represents the system inputs at time t, system outputs at time t, control states at time t, and next expected states, respectively.

B. Model Predictive Control

MPC seeks to find control policy u for equation (1) by minimizing the deviation between desired trajectory and actual one. Here are the MPC parameters.

- Sample time: 0.02s
- Prediction horizon: 100
- Reference path: x∈(-10, 100), y = -5
- Weight matrix for states: [1.0, 1.0, 5.0, 10.0], weight matrix for control inputs: [0.1, 0.1]
- Constraints: kinematic model as shown in Fig.1 [4].

The inputs of the MPC controller are the current states of the ego vehicle and the reference path. The output of the MPC controller is the desired steering angle, u, and desired acceleration. The simulation of MPC is shown in Fig.2.

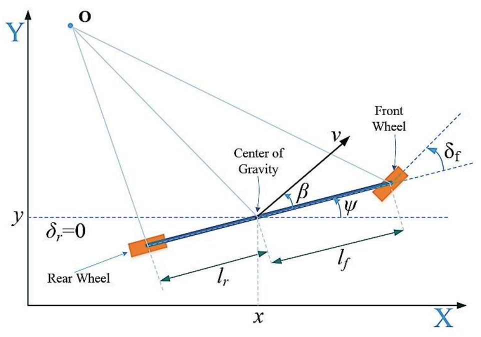


Fig. 1: Kinematic Model MPC algorithm subject to [4].

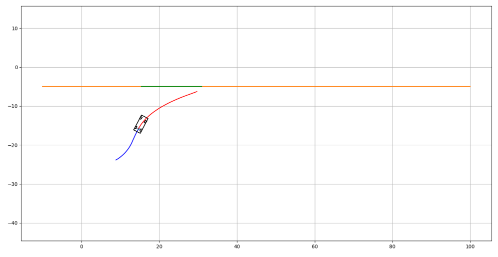


Fig. 2: MPC simulation.

The orange line represents the reference path, and blue line represents its actual path calculated by MPC algorithm.

C. Random Mission Generation

20 random missions are executed with different start points and the same destination. To simulate uncertainties and disturbances that might affect the actual system in real-world situations, noise is added to the control inputs, which means both desired steer and desired acceleration calculated from MPC are added to noise as actual steer, y_m, and actual acceleration, respectively. For the steering angle noise, the mean value is zero and the standard deviation is 1 rad. For the acceleration noise, the mean value is also zero and the standard deviation is 0.1 m/s^2. Take one mission as an example, the desired steering angle derived from MPC algorithm and the actual steering angle (y_m) after adding noise to the desired one are shown in Fig. 3.

Introducing noise to the control inputs helps to simulate the fact that in real world, the control inputs may not be perfectly executed due to various factors such as sensor noise, actuator limitations, or external disturbances. By adding noise to the control inputs during the simulation, MPC becomes more robust to uncertainties and can better handle the variability that presents in the actual system.

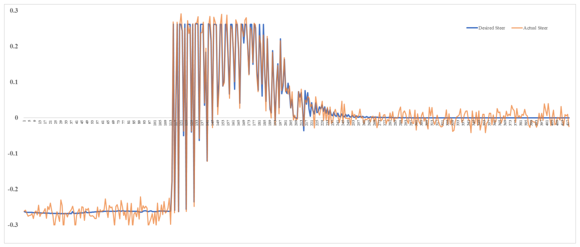


Fig. 3: Desired Steer (blue) and Actual Steer (orange).

D. System Identification

After finding u and y_m for the AV, we need to identify the values of the four matrices: A, B, C, and D for equation (1) and (2). We invoke MATLAB System Identification Toolbox [5], which is the same with the method using in [1] to instantiate the four matrices.

E. Dynamic Time-warping Technique

We then derive the error threshold using DTW. Assuming the real vehicle needs x seconds to make a turn. The model may take $x + w$ seconds to make the same turn. Therefore, our idea of determining the monitor window is to look for the maximum w in all the primitive operations [1]. Once the window is decided, the error threshold is then computed from the maximum observed model-induced errors within the window [1]. Here we use DTW to look for an order-alignment of the timestamps of the two sequences: y_p and y_m to get the detection window size w and error threshold τ.

F. Stealthy Attacks Execution

Algorithm 1 shows the algorithm to implement stealthy attacks against steering control system. To remain stealthy, the attacker should manipulate the steering angle such that the accumulated deviations $S(k)$ between y_m and y_p are lower than τ, namely $S(k) < \tau$, to avoid raising the alarm.

Algorithm 1 Stealthy Attacks Implementation

```
$y = Cx + D * targets$
$x = Ax + B * targets$
$dev = |y - angles|$
$dev_{sum} \ += dev$
if $dev_{sum} \ > \ threshold$
    $raise\_alarm$
```

IV. FUTURE WORK

In future work, we aim to integrate the attack framework into a realistic autonomous driving simulator, such as Carla or Autoware, to evaluate the overall attack framework. We will also work on developing countermeasures to mitigate these threats.

Acknowledgement A part of this work was supported by JSPS KAKENHI 22S0604 and JST CREST JPMJCR23M4.

REFERENCES

[1] H. Choi, W.-C. Lee, Y. Aafer, F. Fei, Z. Tu, X. Zhang, D. Xu, and X. Deng, "Detecting attacks against robotic vehicles: A control invariant approach," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 801–816.

[2] D. Koller and N. Friedman, *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[3] Z. Chen and E. N. Brown, "State space model," *Scholarpedia*, vol. 8, no. 3, p. 30868, 2013.

[4] W. Farag and Z. Saleh, "Mpc track follower for self-driving cars," 2019.

[5] MATLAB, "System identification toolbox," https://jp.mathworks.com/products/sysid.html.

Poster: Bypassing Control Invariants-Based Defenses in Autonomous Vehicles

Yinan Zhao[1], Ziling He[1], Tatsuya Mori[1,2,3]
[1]Waseda University, [2] RIKEN, [3] NICT

NDSS SYMPOSIUM · CREST · WASEDA University

Limitations of Current Attacks Research against AVs

Causing a major deviation in the intended path which is easy to be detected

Almost all attacks which mutate the sensor measurements or inject false messages to control systems are major, which can cause obvious anomalies

Stealthy Attacks: Cause Subtle and Minor Deviations

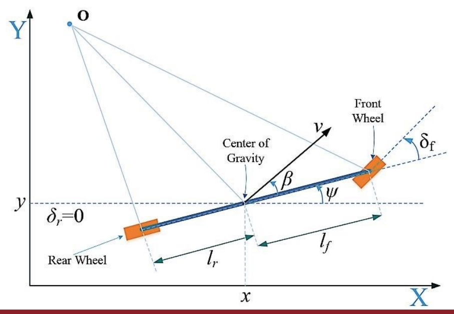
Key idea: Using Model Predictive Control (MPC) which is subject to kinematic model and State-space Model (SSM), System Identification (SI), and Dynamic Time Window (DTW) [1] to allow attackers to accurately simulate system behavior-allowing them to perform undetectable attacks on Control Invariants-secured AV systems.

Attack Model

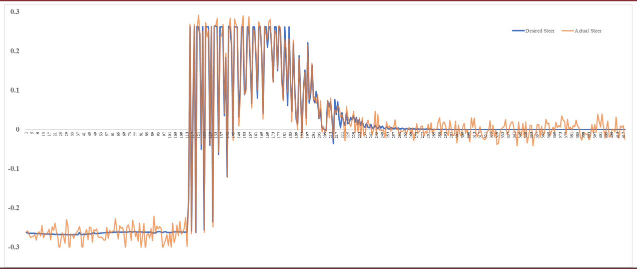
The attack model assumes that the attacker is able to

- Interact with the CAN network to read and write messages through the OBD-II port or other hardware attachments.
- Know the start point and destination of the ego vehicle and list all the possible routes the vehicle may follow to implement MPC control.
- Have full knowledge of SSM, MPC, SI and DTW technique.
- Inject tailored messages into the vehicle's network to stealthily control the vehicle.

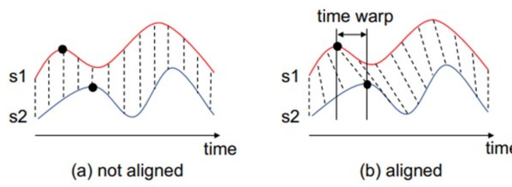
Kinematic Model



Desired & Actual Steering Angle



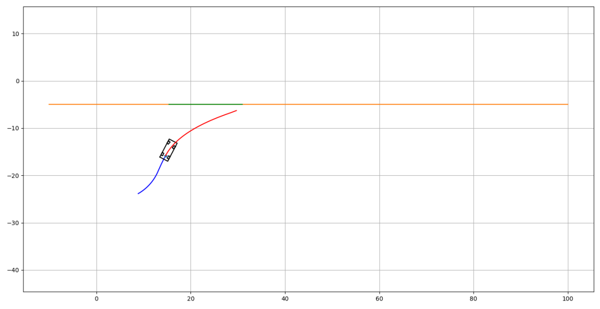
DTW Technique



Acknowledgement: A part of this work was supported by JSPS KAKENHI 22S0604 and JST CREST JPMJCR23M4.

Current Attacks: Cause Major Deviations

MPC Simulation in Python



MPC Simulation in CARLA

Stealthy Attacks Algorithm

```
Algorithm 1 Stealthy Attacks Implementation
  $y = Cx + D * targets$
  $x = Ax + B * targets$
  $dev = |y - angles|$
  $dev_{sum} \mathrel{+}= dev$
  if $dev_{sum} > threshold$
      $raise\_alarm$
```

Future Plan

- Implementation and comprehensive evaluation of the stealthy attack.
- Integrating the attack framework into a realistic autonomous driving simulator, such as Carla or Autoware, to evaluate the overall attack framework.
- Developing countermeasures to mitigate these threats.

[1] Choi, Hongjun, et al. "Detecting attacks against robotic vehicles: A control invariant approach." Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. 2018.