EE 418: Network Security and Cryptography (Fall 2017) Project 2 – Clock-Based Intrusion Detection in Controller Area Network (CAN)

Assigned: Nov 15 (Wed) 2017; **Due: Dropbox, 11:59pm, Dec 8 (Fri), 2017**

Prepared by Xuhang Ying
Network Security Lab (NSL)
Dept. of Electrical Engineering, University of Washington

1 Project Guidelines

1. Maximum allowed team size is **three**. By default, you will stay with your teammates from Project 1. If there is any change to your team, please notify the instructor and TA via email.
2. Submission method:
   - Submit both project report and source code via Dropbox.
   - No late submission is accepted.
3. On the front page of your project report, print:
   (a) Names and student IDs of team members
   (b) Clear description of each team member's contribution
4. Read the reference papers in order to start working on this project.
5. **WARNING**: This project has a lot of coding and reading tasks. Please get started ASAP.

2 Background

Contemporary automobiles are equipped with electronic control units (ECUs) for various functionalities such as vehicle maneuverability and fuel efficiency etc. In order to operate these ECUs properly, the information among ECUs is exchanged via in-vehicle network protocols, such as the Controller Area Network (CAN). However, such protocols were developed for closed networks that are isolated from the external environment. Based on the closed network assumption, in-vehicle protocols were not designed for security, and in particular do not provide encryption or message authentication.

Connected vehicles, however, have an increasingly large and diverse array of outward-facing components in order to provide safety, navigation, and entertainment, which violate the assumption of a closed operating environment. These external interfaces leave connected vehicles vulnerable to attacks in which an adversary compromises one or more outward-facing ECUs (e.g., CD players or cellular radio), gains access to the CAN bus [1], and then blocks messages sent by other ECUs (i.e., denial-of-service suspension attack), sends additional fabricated messages (i.e., fabrication attack) or sends spoofed messages that claim to be originated from legitimate ECUs such as steering or engine control (i.e., **masquerade attack**) [2].

In this project, we consider the masquerade attack on a CAN message that is transmitted every T sec periodically, as illustrated in Fig. 1. In practice, it is difficult to detect the masquerade attack, because the frequency of the targeted message is the same before and after the attack. Nevertheless, the state-of-the-art intrusion detection system (IDS) in [2] exploits the following facts for identifying ECUs: 1) ECUs operate on their local clocks without clock synchronization, and 2) the clock speed (i.e., **clock skew**) of an ECU is constant, and different from other ECUs. Hence, the clock skew is considered as the signature for ECUs.

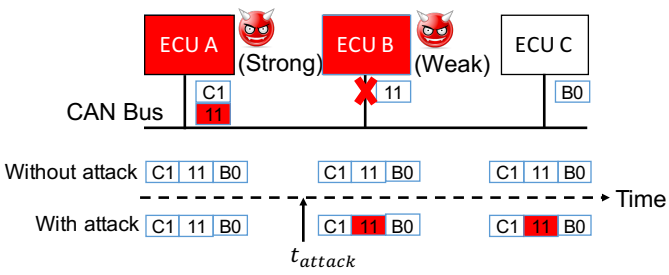

Fig. 1. Illustration of masquerade attack. In this example, ECU A is fully compromised by the strong attacker, and ECU B is weakly compromised by the weak attacker. Before the attack, ECU B transmits message 0x11 every T sec. At $t = t_{attack}$, the weak attacker suspends ECU B's transmission of message 0x11, and the strong attacker starts fabricating and injecting spoofed messages with ID=0x11 every T sec.

3 Clock-Based Intrusion Detection System

In Part I, you will implement a clock-based IDS to detect the masquerade attack in CAN. We first review clock-related concepts, and then introduce the state-of-the-art IDS [2] and the Network Time Protocol (NTP) based IDS [3].

3.1 Clock-Related Concepts

In this project, we follow the Network Time Protocol (NTP) definitions of clocks [4]. Let us first define C_{true} as the "true" clock that runs at a constant rate, i.e., $C_{true}(t) = t$. Let $C_A(t)$ denote the time kept by clock A. The **clock offset** of C_A, denoted as $O_A(t)$, is the difference between the time reported by C_A and the "true" time, i.e., $O_A(t) = C_A(t) - C_{true}(t)$. The **frequency** of C_A at time t is given by $C'_A(t)$. The **clock skew** of C_A, denoted as $S_A(t)$, is the difference in the frequencies (or first derivatives) of C_A and C_{true}, i.e., $S_A(t) = C'_A(t) - C'_{true}(t)$.

A positive clock skew means that C_A runs faster than the true clock, while a negative clock skew implies that C_A runs slower than the true clock. The unit of skew is microseconds per second (μs/s) or parts per million (ppm). For example, if C_A is faster by 5μs every 10ms according to C_{true}, then its skew relative to C_{true} is 500ppm.

3.2 Offset Estimation

In order to use clock skew as signature for identification, an IDS (implemented at a receiving ECU) needs to first estimate average offset and accumulated offset from the arrival timestamps of a periodic message, based on which the clock skew is derived.

The average offset is estimated in batches, where each batch consists of N (e.g., 20) messages. Let the arrival timestamps in a batch be a_1, a_2, ..., a_N. We consider two ways of average/accumulated offset estimation: 1) heuristic-based [2], and 2) NTP-based [3].

– **Heuristic-Based Offset Estimation**: The average offset in the k-th batch is computed as:

$$O_{avg}[k] = \frac{1}{N-1} \sum_{i=2}^{N} \left[a_i - (a_1 + (i-1)\mu_T[k-1]) \right], \tag{1}$$

where $\mu_T[k-1]$ is the average inter-arrival time of the previous batch. The accumulated offset is

$$O_{acc}[k] = O_{acc}[k-1] + |O_{avg}[k]|, \tag{2}$$

where $|\cdot|$ in the second term of the right-hand side means the absolute value.

– **NTP-Based Offset Estimation**: The average offset is the k-th batch is given by

$$O_{avg}[k] = T - \frac{a_N - a_0}{N}, \tag{3}$$

where a_0 is the arrival timestamp of the last message in the previous batch. The accumulated average offset up to the last message of the k-th batch is given by

$$O_{acc}[k] = O_{acc}[k-1] + N \cdot O_{avg}[k]. \tag{4}$$

An example of heuristic-based and NTP-based accumulated offsets is provided in Fig. 5 in [3].

3.3 Clock Skew Estimation

The accumulated offset grows linearly as a function of time, and is modeled as $O_{acc} = S \cdot t + e$, where S is the regression parameter (i.e., clock skew), t the elapsed time, and e the identification error. To estimate the unknown parameter S, the Recursive Least Squares (RLS) algorithm is adopted:

1. It first computes the identification error $e[k] = O_{acc}[k] - S[k-1]t[k]$, where $S[k-1]$ is the estimated clock skew in the previous batch, and $t[k]$ is the elapsed time up to the last message of the k-th batch.
2. It then updates the gain $G[k] = \frac{\lambda^{-1}P[k-1]t[k]}{1+\lambda^{-1}t^2[k]P[k-1]}$, and the covariance $P[k] = \lambda^{-1}(P[k-1] - G[k]t[k]P[k-1])$, where $P[k-1]$ is the covariance in the $(k-1)$-th batch, and λ is the forgetting factor (e.g., 0.9995).
3. Finally, it updates the skew estimate $S[k] = S[k-1] + G[k]e[k]$.

3.4 Cumulative Sum (CUSUM) Detector

In a masquerade attack, the impersonating ECU has a clock skew different from the targeted ECU's, which would lead to significant identification errors. Hence, the identification error is considered as an indicator of whether an attack is taking place. The IDS tracks the normal clock behavior for messages with the target ID by tracking the mean and variance of the errors (denoted as e), μ_e and σ_e^2. To be robust against noise, μ_e and σ_e^2 are updated only if the new error sample e satisfies $|(e-\mu_e)/(\sigma_e)| < \gamma$, where γ is a given update threshold (e.g., 4).

For detection, the CUSUM method, which derives the cumulative sums of deviations from the norm behavior, is implemented. Letting $\theta_e = \frac{e-\mu_e}{\sigma_e}$, the upper and lower control limits L^+ and L^- (for detecting a sudden positive or negative shift) are updated for each new error sample as:

$$L^+ = \max(0, L^+ + \theta_e - \kappa), L^- = \max(0, L^- - \theta_e - \kappa), \tag{5}$$

where κ (e.g., 8) is a sensitivity parameter. If either control limit exceeds a detection threshold Γ (e.g., 5), a sudden shift is detected, and the IDS declares an intrusion.

4 Your Assignment

- **Dataset**: We will use the dataset collected from a real vehicle called EcoCar [5], which contains all CAN messages transmitted within 50 min (a sample screenshot is provided in Fig. 2). For our purpose, we are interested in three messages:
  - `0x184`: a 10-Hz message transmitted by ECU A.
  - `0x3d1`: a 10-Hz message transmitted by ECU B.
  - `0x180`: a 10-Hz message transmitted by ECU C.

  For simplicity, we have parsed the original dataset, and generated three files (`184.txt`, `3d1.txt` and `180.txt`), which only contain the arrival timestamps of the three messages, respectively.
- **Scenarios**: We consider two scenarios in this project.
  - **Scenario 1 – masquerade attack**: ECU A is fully compromised and ECU B is weakly compromised. The adversary stops A's transmission of `0x184` and uses ECU B to transmit spoofed messages `0x184` every 0.1 sec instead. In practice, ECU B is transmitting `0x3d1`, but we treat it as `0x184` [1].
  - **Scenario 2 – intelligent masquerade attack (i.e., cloaking attack)**: ECU A is fully compromised and ECU C is weakly compromised. A is prevented from transmitting `0x184`, and C transmits spoofed message `0x184` (in practice, it is 0x180 being transmitted, but treated as `0x184`). Different from Scenario 1, the adversary is intelligent and manipulates its clock skew (as seen by the detector) by adding a small time delay ΔT into message inter-departure time. That is, the adversary uses ECU C to transmit `0x184` every $(0.1 + \Delta T)$ sec, where $\Delta T = -0.000029$ sec or -29μs.
- **Source Code**:

  For Python, there are two files: `ids.py` and `simulation.py`.
  - `ids.py`: In this file, class `IDS` is implemented, which contains both the state-of-art and the NTP-based IDSs.
  - `simulation.py`: This file contains the main function and the following functions.
    - `import_data(file=None)`: Import data from file.
    - `plot_acc_offsets(ids, mode)`: `ids` is a dictionary that contains IDS instances, and `mode` is either `'state-of-the-art'` or `'ntp-based'`.
    - `simulation_masquerade_attack(mode)`: Simulate masquerade attack and plot control limits.
    - `simulation_cloaking_attack(mode)`: Simulate cloaking attack and plot control limits.

  For MATLAB, there are the following files:
  - `IDS.m`: Class IDS.
  - `simulation.m`: Main file for simulation.
  - `import_data.m`: Import data from file.
  - `plot_acc_offsets.m`: Plot accumulated offset curves as function of elapsed time.
  - `simulation_masquerade_attack.m`: Simulate masquerade attack and plot control limits.
  - `simulation_cloaking_attack.m`: Simulate cloaking attack and plot control limits.

[1] This is mainly because ECU A is a stock ECU, and we do not want to force it to stop transmission, in order to avoid any possible damage to the real vehicle.

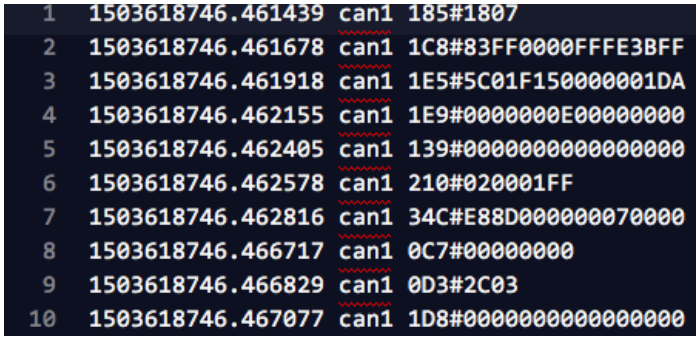

Fig. 2. Screenshot of collected CAN data from EcoCar [5]. Each row corresponds to a CAN message: the first entry is the arrival timestamp, followed by CAN interface ID; the last entry is the CAN message ID (in Hex) and CAN data (in Hex) that are separated by '#'.

4.1 Tasks and Questions

1. Implement the class `IDS`.
2. Plot accumulated offset curves as function of the elapsed time for the three messages (i.e., `0x184`, `0x3d1` and `0x180`) using the state-of-the-art IDS with batch size $N = 20$.
3. Repeat Task 2 for the NTP-based IDS with $N = 20$.
   - Compare the slopes of the curves from Tasks 2 and 3, and comment on the similarity of the three messages in terms of estimated clock skew from the perspective of the IDS.
4. Repeat Tasks 2 and 3 with $N = 30$.
   - Comparing the four figures from Tasks 2 through 3, comment on the consistency of clock skew estimation (i.e., the slope of the curve for the same message should be the same regardless of N) for the state-of-the-art and the NTP-based IDSs.
5. Simulate the masquerade attack in Scenario 1. In this simulation, we first feed 1000 batches of arrival timestamps of message `0x184` to the IDS, and then feed 1000 batches of arrival timestamps of message `0x3d1` to it (batch size is 20). That is, the masquerade attack occurs at the 1001-st batch, and is detected if either upper or lower control limit exceeds the threshold $\Gamma = 5$. Plot control limits as function of number of batches for the state-of-the-art and the NTP-based IDSs.
   - Which IDS can detect the masquerade attack? Why?
6. Simulate the cloaking attack in Scenario 2. In this simulation, we first feed 1000 batches of arrival timestamps of message `0x184` to the IDS, and then feed 1000 batches of arrival timestamps of message `0x180` to it (batch size is 20). Plot control limits as function of number of batches for the state-of-the-art and the NTP-based IDSs.
   - Which IDS can detect the cloaking attack? Why?
   - Comparing masquerade and cloaking attacks, comment on the limitations of a clock-skew based IDS.

Please include all necessary figures/plots, observations, and answers in your report.

4.2 Additional Questions

Read references [2] and [3], and answer the following questions.

1. Briefly explain how the adversary chooses ΔT for the cloaking attack on the clock skew detector.

2. What is Maximum Slackness Index (MSI), and what does it measure? Based on Fig. 8 of [3], briefly comment on the performance of cloaking attack on an IDS in terms of MSI.
3. Based on [2], explain under what circumstances, two messages are likely to be highly correlated. Based on the analysis in Section IV-C and Fig. 10 in [3], explain under what circumstances, two messages are likely to be highly correlated.
4. Based on [3], describe how to launch the cloaking attack on the correlation detector, and briefly explain why it works.

References

1. S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces." in *USENIX Security Symposium*. San Francisco, 2011.
2. K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection." in *USENIX Security Symposium*, 2016, pp. 911–927.
3. S. U. Sagong, X. Ying, A. Clark, L. Bushnell, and R. Poovendran, "Cloaking the clock: Emulating clock skew in controller area networks," *arXiv preprint arXiv:1710.02692*, 2017.
4. S. B. Moon, P. Skelly, and D. Towsley, "Estimation and removal of clock skew from network delay measurements," in *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, vol. 1. IEEE, 1999, pp. 227–234.
5. "UW EcoCar," http://uwecocar.com/, accessed: 2017-09-26.