Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science 317 (2015) 141–153

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Attacking a Feedback Controller

Ashish Tiwari [1,2]

Computer Science Laboratory
SRI International
Menlo Park, U.S.A

Abstract

We consider the problem of generating sensor spoofing attacks on feedback controllers. The attacker has the option of remaining in a stealth mode – wherein it spoofs some sensor but only by an amount that is indistinguishable from noise. Later, the attacker can launch a full attack and try to force the system to get into an unsafe region. Using bounded model checking on an example adaptive cruise controller, we show that (1) remaining in a stealth mode is not very benefecial for the attacker, and (2) there is a phase transition between two classes of attacks: attacks that are small and indistinguishable, but that are unable to make the system unsafe by themselves, and attacks that are large, and possibly easily detected, but that easily take the system to an unsafe state. The preliminary experiments suggest that a control system is most vulnerable when it is just engaged (at discrete switches). Moreover, if it is guarded with a safety envelope based monitor and can ignore sensor data that is outside the safety envelope, then it is relatively difficult to compromise safety of such a control system by just sensor spoofing.

Keywords: bounded model checking, feedback control, sensor spoofing, safety

1 Introduction

A control system consists of sensors, controllers, and actuators. Sensors gather data about the state of the system. Controllers use the sensor data to compute an appropriate control action. Actuators then perform the control action and thus effect a change in system state, which is then sensed by the sensors again in the next cycle. In such a feedback system, the controllers rely on sensors to provide good quality data to ensure that the system operates correctly and safely even in an uncertain environment.

We are constantly embedding such control systems inside larger cyber-physical systems that are increasing becoming *connected* to the outside world. As a result,

[1] This work was sponsored, in part, by Defense Advanced Research Projects Agency (DARPA) and Air Force Research Laboratory (AFRL), under contract FA8750-12-C-0284, and the National Science Foundation under grant CNS-1423298. The views, opinions, and/or findings contained in this report are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the funding agencies.

[2] Email: tiwari@csl.sri.com

http://dx.doi.org/10.1016/j.entcs.2015.10.014

the attack surface is growing swiftly, and it is not difficult to spoof sensors and inject wrong data into the system. There are known instances of GPS spoofing attacks on unmanned aerial vehicles [1]. Modern automobiles have also been shown to have several attack surfaces that an attacker can use to compromise computers/networks within a car and inject sensor spoofing attacks [5,2].

One way to improve security of any system would be to eliminate the attack vectors. This is an uphill battle since systems are increasingly becoming more connected and communicate extensively with other devices. Here, we take the view that it will be difficult to completely eliminate all attack surfaces, and inevitably, there will always be some channels available for an attacker to exploit.

We also assume that it may be possible to perform anomaly detection and build a safety monitor that can be used to detect attacks. Specifically, one could possibly compute a *safety envelope* for the nominal operation of the system, and then, at runtime, monitor that the system remains inside the safety envelope [13]. In such a setting, a sensor spoofing attack that sends "wild" sensor readings to the controller may get easily detected and countered. It may be beneficial for the attacker to change the sensor values only slightly. That way, the attacker could remain undetected and cause the system to reach some intermediate state before launching a full-blown attack. Would such a strategy be helpful for the attacker? Are there good deviation values for the attacker to use – ones that will guarantee that they remain undetected, but still push the system into an unsafe state? These questions form the starting point of our investigation in this paper.

Our goal here is to analyze sensor spoofing attack strategies in the presence of runtime monitors. Our approach is based on using formal verification tools on a model of a controller for adaptive cruise control. Specifically, we build a model of two vehicles moving in a 1-dimensional road where the rear vehicle's acceleration is controlled by a feedback law. The goal of the controller is to ensure that the rear vehicle never crashes into the car in front. We assume that an attacker can spoof the velocity sensor of the rear car. We create a timed relational abstraction of the system model and then use infinite bounded model checking to answer some of the questions posed above.

We formalize the problem of interest in Section 2, we present the results of our experiments in Section 3, and thereafter, in Section 4, we present details of our experimental setup.

1.1 Related Work

Recently there has been increasing interest in studying control systems and cyber-physical systems under adversarial attack. There is considerable amount of work on attack-resilient state estimation, since if we can determine the correct state (even in presence of an attack), then the rest of the system can be used unchanged as it would just rely on the estimated state [4,9,7,8]. For linear systems, the state estimation problem in presence of attacks can be cast as an optimization problem [4,9]. Fawzi et al. [4] assume existence of redundant sensors and provide theoretical guarantees on the maximum number of attacks that can be handled. Extending the work in [4], an

attack-resilient state estimation procedure is presented that also handles presence of noise and modeling errors in [7]. It is shown that the attacker cannot destabilize the system by exploiting the difference between the model used for state estimation and the real physical system. A method for coding the sensor outputs to enable detection of false data injection attacks is presented in [6]. Assuming that the sensor itself is not compromised, the result in [6] is a low-cost alternative to encryption of sensor data. Both these solutions – coding sensor outputs and performing resilient state estimation – can potentially solve the sensor spoofing problem. However, it is still worthwhile to study the consequences of stealth attacks. First, a resilient state estimation may not be available because, for example, it may not be cost effective to have redundant sensors in the system. Second, algorithms used for coding sensor outputs may get compromised. In such a case, the questions raised, and partially answered, in this paper become relevant.

2 Problem Description

We assume we have a plant being controlled by a controller. The plant dynamics are given by

$$\frac{d\vec{x}}{dt} = P(\vec{x}, \vec{u}) \tag{1}$$

where $\vec{x}$ is the state space of the plant and $\vec{u}$ is the control input. (The actuator dynamics are included in the plant dynamics). A feedback controller is used to produce the values $\vec{u}$ from the sensor readings $\vec{y}$ as follows.

$$\vec{u} = C(\vec{y}) \tag{2}$$

Here C is some function. The sensor values $\vec{y}$ are a function S of the current state $\vec{x}$; when modeling sensor attacks, the function S can include the attack model.

$$\vec{y} = S(\vec{x}) \tag{3}$$

By putting the three equations above together, we get a model of the feedback control system under attack. Let `Sys` denote this model.

The controller under study is activated (or engaged) when certain conditions on the state space are met. Let `Init` denote these conditions.

$$\texttt{Init} \subset \mathbb{R}^{|\vec{x}|} \tag{4}$$

The system starts in some initial state in the set `Init`, and then generates an execution trace according to the Equation 1, Equation 2, and Equation 3. By a slight abuse of notation, let `Sys` also denote a function that maps a tuple consisting of a sensor model S, an initial state $\vec{x_0}$, and a time instance t to the state reached at time t.

$$\texttt{Sys}(S, \vec{x_0}, t) \in \mathbb{R}^{|\vec{x}|} \tag{5}$$

The function `Sys` is the semantics of the system.

The system is expected to remain inside some safe set of states. Let `Safe` denote the safe set.

(6) $\texttt{Safe} \subset \mathbb{R}^{|\vec{x}|}$

Ideally, we want $\texttt{Sys}(S, \vec{x_0}, t) \in \texttt{Safe}$ for all time $t \geq 0$, for all initial states $\vec{x_0} \in \texttt{Init}$, and for all reasonable sensor attacks S.

2.1 Sensor Attacks

Let us make things a bit more concrete by assuming that all the plant state variables $\vec{x}$ can be sensed (by different sensors). Hence, when there is no attack, the function S above would just be the identity function. Let us also assume, for simplicity, that the attacker just offsets the value of the sensor by some constant picked from a range. Hence, under these two assumptions, Equation 3 takes the form

(7) $\vec{y} = \vec{x} + \vec{c} \quad \vec{c} \in D$

where $D \subset \mathbb{R}^n$ is the domain of attack. If the attacker has the ability to spoof a single sensor, then D would contain vectors that have only one nonzero entry.

We assume the attack is carried out in two *modes*: a stealth mode and an evident mode. In the *stealth mode*, the offset in the sensor value introduced by the attacker is bounded by a small number. In the *evident mode*, this offset is much larger. Intuitively, the attacker tries to remain undetected in the stealth mode, and hence the attacker does not deviate the correct sensor value by a large amount. Now, Equation 3 takes the form

(8) $$\vec{y} = \begin{cases} S_{\texttt{stealth}}(\vec{x}) & \text{if in stealth mode} \\ S_{\texttt{evident}}(\vec{x}) & \text{if in evident mode} \end{cases}$$

Accordingly, Equation 7 will take the form

(9) $$\vec{y} = \begin{cases} \vec{x} + \vec{c} \quad \vec{c} \in D_{\texttt{stealth}} & \text{if in stealth mode} \\ \vec{x} + \vec{c} \quad \vec{c} \in D_{\texttt{evident}} & \text{if in evident mode} \end{cases}$$

Assume that exactly one sensor, say the sensor corresponding to index 1, is under attack. Let us fix the domains of attack.

(10) $$D_{\texttt{stealth}} = \{\vec{c} \mid |c_1| \leq \texttt{stbound},\ c_i = 0 \text{ for } i \neq 1\}$$
$$D_{\texttt{evident}} = \{\vec{c} \mid |c_1| \leq \texttt{evbound},\ c_i = 0 \text{ for } i \neq 1\}$$

Here, $\texttt{stbound}$ and $\texttt{evbound}$ are two parameters such that $\texttt{stbound} < \texttt{evbound}$.

Let us also assume that the attacker stays in stealth mode for $\texttt{t}_{\texttt{stealth}}$ time and then switches to the evident mode. Let $\texttt{t}_{\texttt{toAttack}}$ denote the time taken by the attacker *in the evident mode* to push the system into an unsafe state.

(11) $$\texttt{t}_{\texttt{toAttack}} = \inf\{t \mid \texttt{Sys}(S_{\texttt{evident}}, \vec{x_1}, t) \notin \texttt{Safe}, \ \vec{x_1} = \texttt{Sys}(S_{\texttt{stealth}}, \vec{x_0}, \texttt{t}_{\texttt{stealth}}),\ \vec{x_0} \in \texttt{Init}\}$$

Objective: The goal of the paper is to study how the the parameters $\texttt{stbound}$, $\texttt{evbound}$, $\texttt{t}_{\texttt{stealth}}$, influence the value $\texttt{t}_{\texttt{toAttack}}$.

Remark 2.1 *Generality of Considered Attack:* Although we say that the attacker just offsets the value of some sensor by some constant c, this constant *can change*

with time. The only constraint is that c remains inside the domain D of attack. So, we indeed consider rather general forms of attacks.

Remark 2.2 *Safety Runtime Monitor:* The safety monitor is implicit in the above formulation in the form of the definition of the stealth attack. More precisely, we are assuming that the safety monitor is checking, at runtime, if the difference between a sensor output and the expected sensor value is at most `stbound`. As a result, in the stealth mode, the attacker is allowed to change (the compromised) sensor value by at most `stbound`: if the attacker changes a value by more than that bound, then the safety monitor will detect the attack and take corrective action.

Remark 2.3 *Noise:* The value `stbound` is a reflection of the fact that there is *noise* in sensor outputs, and the feedback controller is designed to tolerate that noise. We will choose the value `stbound` to guarantee that, if the error in sensor output is bounded by `stbound`, then the system remains safe. So, *a priori*, it is guaranteed that the attacker can not cause the system to enter an unsafe state by being in the stealth mode always.

3 Main Results

We postpone the detailed description of the plant and the controller to later. We first present the experiments and the results of the experiments.

3.1 No Stealth Mode

We first considered the scenario where the attacker spends zero time in stealth mode, that is, $\mathtt{t_{stealth}} = 0$. In Figure 1, we plot the time taken to reach unsafe state $\mathtt{t_{toAttack}}$ (on y-axis) against the strength of the evident attack `evbound` (on x-axis).

As expected, Figure 1 shows that the time taken to reach an unsafe state decreases as we increase the range of values that can be used to offset the value of a compromised sensor. Note that sharp fall in the plot followed by the gradual decline later. At `evbound` $= 2.1$, the attack failed to take the system to an unsafe state, and at `evbound` $= 2.2$, the system was compromised in just 6 seconds. The drop in the value of $\mathtt{t_{toAttack}}$ is more gradual as `evbound` increases from 2.2 to 6.5.

If a safety envelope based monitoring approach is implemented for this control system, then a value of 2.1 would be a good choice for defining the safety envelope. The controller is able to ensure safety for deviations in sensor values less-than 2.1; and a safety monitor could "throw away" sensor values that are off by more than 2.1. Such a safety monitor then could almost guarantee safe operation of the system. In fact, pursuant to Remark 2.2 and Remark 2.3, we will choose `stbound` to be less-than or equal to 2.

We make two remarks that hold for *all* plots in this paper. First, the values of $\mathtt{t_{toAttack}}$ did not vary continuously – the experimental setup forced the values of $\mathtt{t_{toAttack}}$ to be even numbers. Second, the value 24 was used as a substitute for any number larger than 24, including ∞.

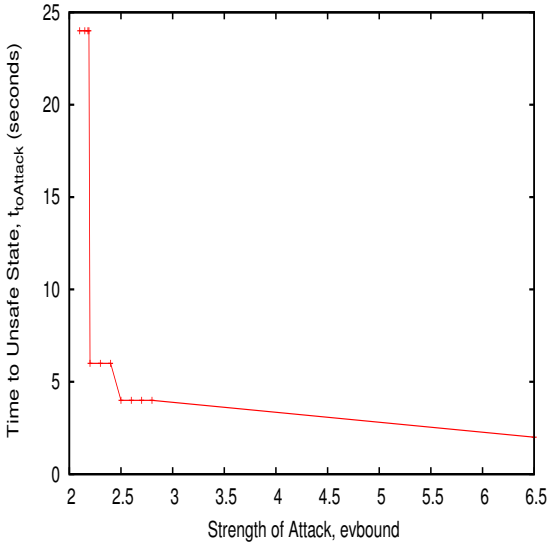

Fig. 1. Time taken to reach unsafe state $\texttt{t}_{\texttt{toAttack}}$ (on y-axis) against the strength of the evident attack `evbound` (on x-axis). There was no stealth mode. As we increase `evbound`, the value of $\texttt{t}_{\texttt{toAttack}}$ decreases.

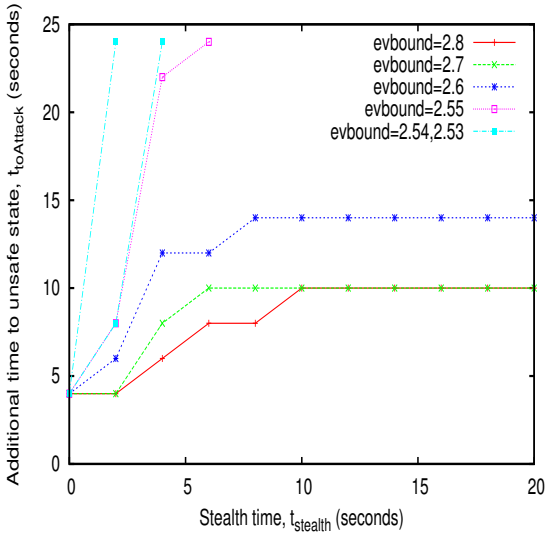

Fig. 2. Time taken to reach an unsafe state (y-axis, in seconds) in evident mode plotted against the time spent in stealth mode (x-axis, in seconds). As we increase the stealth time, it becomes progressively more difficult to get the system into an unsafe region. Here, the attacker is inactive in the stealth mode.

3.2 Attacker Silent in Stealth Mode

For the second set of results, we considered the scenario where the attacker spends nonzero amount of time in stealth mode ($\texttt{t}_{\texttt{stealth}} \neq 0$), but the attacker does not change sensor values in stealth mode ($\texttt{stbound} = 0$). For this scenario, in Figure 2, we plot the time $\texttt{t}_{\texttt{toAttack}}$ taken to reach unsafe state (on y-axis) against the time $\texttt{t}_{\texttt{stealth}}$ spent in stealth mode (on x-axis). We have one line in the plot for each

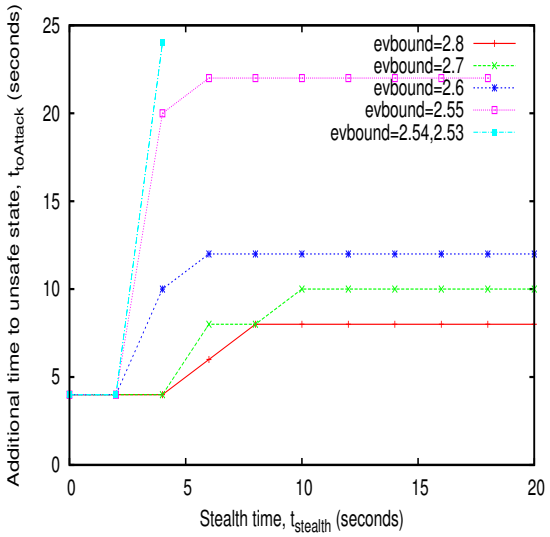

Fig. 3. Time taken to reach an unsafe state (y-axis, in seconds) in evident mode plotted against the time spent in stealth mode (x-axis, in seconds). As we increase the stealth time, it becomes progressively more difficult to get the system into an unsafe region. Here, the attacker is active in the stealth mode, but introduces only "small" deviations in sensor outputs ($\texttt{stbound} = 1$).

value of the strength `evbound` of the evident attack.

As we increase the time spent in stealth mode, the time taken in evident mode to reach an unsafe state also increases. This matches with what one would expect, since here, in the stealth mode, we have $\texttt{stbound} = 0$, and hence the system is not under attack until time $\texttt{t}_{\texttt{stealth}}$. As a result, in the $\texttt{t}_{\texttt{stealth}}$ units of time after it is engaged, the controller is able to take the system into a "good region of state space" from where it becomes difficult for the attacker to move the system to an unsafe state.

It is worth comparing the results in Figure 2 with those in Figure 1. When the attacker could start the attack as soon as the controller was engaged, a value of 2.2 or greater for `evbound` was enough for the attacker to cause the system to enter an unsafe state within 6 seconds (Figure 1). But, if the attacker was even 2 seconds late in starting the attack (and the controller had just 2 seconds of "unattacked" time), it needed a value of `evbound` larger than 2.6 to get the system into an unsafe state in under 6 seconds. The tolerance of the feedback controller to attacks increases with time.

3.3 Attacker Conservative in Stealth Mode

For the third set of results, we considered the scenario where the attacker spends nonzero amount of time in stealth mode ($\texttt{t}_{\texttt{stealth}} \neq 0$), and the attacker launches a "relatively mild" attack in the stealth mode ($\texttt{stbound} = 1$). For this scenario, in Figure 3, we plot the time taken to reach unsafe state $\texttt{t}_{\texttt{toAttack}}$ (on y-axis) against the time $\texttt{t}_{\texttt{stealth}}$ spent in stealth mode (on x-axis). We have one line in the plot for each value of the strength `evbound` of the evident attack.

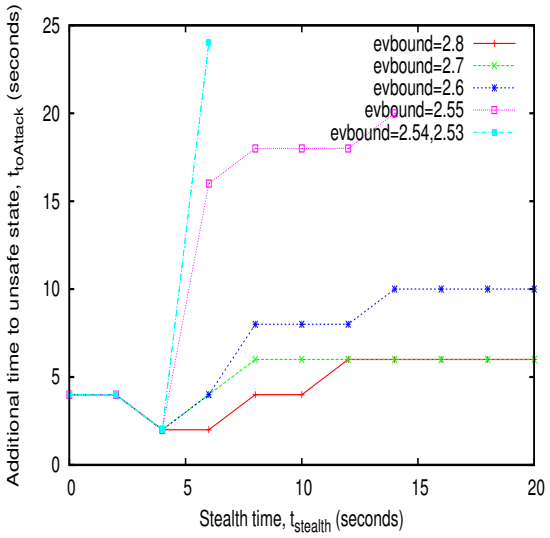

Fig. 4. Time taken to reach an unsafe state (y-axis, in seconds) in evident mode plotted against the time spent in stealth mode (x-axis, in seconds). As we increase the stealth time, it becomes progressively more difficult to get the system into an unsafe region. Here, the attacker is active in the stealth mode, and introduces "large" deviations in sensor outputs ($\texttt{stbound} = 2$).

One would expect that the attacker can use the time spent in stealth mode to drive the system into a state from where it can quickly force the system into an unsafe state in the evident mode. That is, we expect $\texttt{t}_{\texttt{toAttack}}$ to decrease as $\texttt{t}_{\texttt{stealth}}$ increases, assuming the other two parameters, `stbound` and `evbound`, remain unchanged. However, the plot in Figure 3 indicates otherwise. It shows that the attacker is unable to decrease $\texttt{t}_{\texttt{toAttack}}$ by increasing $\texttt{t}_{\texttt{stealth}}$. On the contrary, $\texttt{t}_{\texttt{toAttack}}$ increases as $\texttt{t}_{\texttt{stealth}}$ increases, much in the same way as it did when the attacker was silent in the stealth mode (Figure 2). In some sense, the feedback controller is able to "handle" the small attack launched by the attacker in the stealth mode.

Keeping the parameters `evbound` and $\texttt{t}_{\texttt{stealth}}$ fixed, we now compare the change in $\texttt{t}_{\texttt{toAttack}}$ with change in `stbound`. For that, we need to compare Figure 3 (where attacker did something in the stealth mode, $\texttt{stbound} = 1$) with Figure 2 (where the attacker did nothing in the stealth mode, $\texttt{stbound} = 0$). We note that the value of $\texttt{t}_{\texttt{toAttack}}$ when $\texttt{stbound} = 1$ is no more than the value of $\texttt{t}_{\texttt{toAttack}}$ when $\texttt{stbound} = 0$; and in many cases, it is slightly less.

3.4 Attacker Incautious in Stealth Mode

For the final set of results, we considered the scenario where the attacker spends nonzero amount of time in stealth mode ($\texttt{t}_{\texttt{stealth}} \neq 0$), and the attacker launches a "relatively bold" attack in the stealth mode ($\texttt{stbound} = 2$). For this scenario, in Figure 4, we plot the time taken to reach unsafe state $\texttt{t}_{\texttt{toAttack}}$ (on y-axis) against the time $\texttt{t}_{\texttt{stealth}}$ spent in stealth mode (on x-axis). We have one line in the plot for each value of the strength `evbound` of the evident attack.

In this case, we do get the interesting behavior wherein $\mathtt{t_{toAttack}}$ *decreases* as $\mathtt{t_{stealth}}$ increases. Specifically, when $\mathtt{stbound} = 2$ and $\mathtt{evbound}$ is any value in $\{2.53, 2.54, 2.6, 2.7, 2.8\}$, we observe that $\mathtt{t_{toAttack}}$ decreases as $\mathtt{t_{stealth}}$ increases from 2 to 4. In this case, the attacker can indeed benefit by remaining in stealth mode. However, $\mathtt{t_{toAttack}}$ rises with $\mathtt{t_{stealth}}$ for all values of $\mathtt{t_{stealth}}$ greather-than 4, so there is only a small window where an attacker can benefit from a stealth mode.

We again note that, comparing Figure 3 and Figure 4, increasing $\mathtt{stbound}$ causes $\mathtt{t_{toAttack}}$ to decrease, and the decrement in $\mathtt{t_{toAttack}}$ as we go from $\mathtt{stbound} = 1$ to $\mathtt{stbound} = 2$ is signficant. Note that the value $\mathtt{stbound} = 2$, where we start getting benefit of a stealth mode, is close to the "edge" of the safety envelope, which was defined by the value 2.1 from the plot in Figure 1.

4 Adaptive Cruise Controller

We now briefly present the model we used for performing the experiments described above.

As mentioned before, we use the adaptive cruise controller model from [10]. The state variables are $\mathtt{gap}, v_f, v_l, a_f, a_l$, where $\mathtt{gap}$ denotes the gap between the leader (subscript l) and the follower (subscript f), v_f, a_f are the velocity and acceleration of the follower, and v_l, a_l are the velocity and acceleration of the leader.

The plant model is simple:

$$\frac{d\mathtt{gap}}{dt} = v_l - v_f \tag{12}$$

$$\frac{dv_l}{dt} = a_l \tag{13}$$

$$\frac{dv_f}{dt} = a_f \tag{14}$$

$$\frac{da_f}{dt} = u \tag{15}$$

The variable a_l is an input (disturbance), but it is assumed that a_l is constrained to be within $-5m/s^2$ and $2m/s^2$. The velocities v_l, v_f are always non-negative. The variable u is the output of the controller.

The controller sets the variable u as follows [10]:

$$u = -3a_f - 3(v_f - v_l) + \mathtt{gap} - (v_f + 10)$$

The controller is engaged whenever the following condition holds; in other words, the following is the set $\mathtt{Init}$ of initial states [10].

$$\mathtt{gap} \geq 5, \;\; 0 \leq v_l, v_f \leq 30, \;\; \mathtt{gap} - 0.1(v_f^2 - v_l^2) - 10 - (v_f - v_i) \geq 0$$

Since there is a nonlinear term in the third expression above, in our analysis we used a linear *under-approximation* of it under the constraint $0 \leq v_l, v_f \leq 30$:

$$(v_f \geq v_l \wedge \mathtt{gap} - 10 - (v_f - v_l) - 0.1 * (v_f - v_l) * 60 \geq 0) \vee (v_f \leq v_l \wedge \mathtt{gap} - 10 - (v_f - v_l) \geq 0)$$

The set of *safe* states is defined by the constraint $\texttt{gap} \geq 0$.

4.1 Sensor Attacks

We assume that the velocity sensor of the following car can be spoofed. Hence, instead of seeing the velocity v_f, the controller gets a value $v_f + \texttt{attack}$ from the sensor. As a result, the controller actually computes the following control output:

$$u = -3 * ai - 3 * (v_f - v_l) + \texttt{gap} - (v_f + 10) - 4 * \texttt{attack}$$

In the *stealth* mode, the value of the variable `attack` is constrained to lie between $-\texttt{stbound}$ and `stbound`. In the *evident* mode, the value of the variable `attack` is constrained to lie between $-\texttt{evbound}$ and `evbound`.

We model the complete system in HybridSal [12]. The model has two modes. In the model, we also have a parameter $\texttt{t}_{\texttt{stealth}}$ and the model remains in stealth mode for time $\texttt{t}_{\texttt{stealth}}$. The complete HybridSal model is presented in the Appendix.

4.2 Abstraction and Bounded Model Checking

We generated the data for the plots shown in Section 3 using the two-step verification process supported by HybridSal. In the first step, the model is abstracted to an infinite-state discrete state transition system. In the second step, the abstract model is model-checked.

Since we are interested in finding attacks and the time required to take the system to an unsafe state, we used HybridSal to create a *timed relational abstraction* [14] of the system. Given a time duration Δ, a timed relational abstraction consists of a transition relation that relates two states if the second state is reachable from the first in exactly Δ units of time. The benefit of using them is that they are precise: relatively precise timed relational abstractions are computed by HybridSal. On the other hand, the drawback is that we may miss detecting a path to an unsafe state if every such path takes a duration that is not an integer multiple of Δ.

For our results reported in Section 3, we used $\Delta = 2$, and that is the reason why $\texttt{t}_{\texttt{stealth}}$ and $\texttt{t}_{\texttt{toAttack}}$ are both multiples of 2 always.

The abstract system is model checked using the SAL infinite bounded model checker [3]. The bounded model checker uses Yices as its constraint solver [11]. In our experiments, we used $\Delta = 2$, and $\texttt{t}_{\texttt{stealth}} + \texttt{t}_{\texttt{toAttack}}$ was mostly at most 40, so most of the bounded checking runs used a depth less-than or equal to 20. Only a few of these runs took more-than a few minutes of real time.

Remark 4.1 *Choice of sensor under attack:* The choice of the sensor under attack is insignificant for our results. If, for example, the sensor for `gap` was compromised (rather than the sensor for v_f), then we would obtain the same results, but for a constant factor in the values of `stbound` and `evbound`.

Remark 4.2 *Generalization:* Our observations here are based on analysis of a specific feedback controller. We can, of course, use the same tools to perform similar

study on other systems: in fact, HybridSal can model more general hybrid dynamical systems and thus model mode switching and controllers with finite memory. Moreover, it should be possible to perform a similar exploration theoretically (and more generally). Intuitively, we expect that we will obtain similar results.

5 Conclusion

We studied the problem of sensor spoofing in a feedback control system. We assumed a scenario wherein the control system was actively guarded by a runtime monitor that rejected sensor values that were ostensibly spurious. In our model, the attacker had the option of staying in stealth mode and launching mild attacks on sensor values before launching into a full-blown attack. We used a formal model of a cruise controller, abstraction and bounded model checking to find the relationship between different parameters and the existence of attacks. The main conclusion is that a feedback controller is most susceptible in a short time window soon after it is engaged – that is, at discrete switches of a supervisory controller. Moreover, weak attacks in stealth mode are not too helpful for the attacker since the feedback controller is able to handle them (as it handles noise). Strong attacks in stealth mode can help the attacker by reducing the time required in evident mode to reach an unsafe state, but they make the attacker more susceptible to detection even in stealth mode.

References

[1] Wikipedia, the free encyclopedia. en.wikipedia.org/wiki/Iran-U.S._RQ-170_incident.

[2] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. C. an, F. Roesner, and T. Kohno. Comprehensive experimental analyses of automotive attack surfaces. In *USENIX Security*, 2011.

[3] L. de Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari. Sal 2. In R. Alur and D. Peled, editors, *Computer-Aided Verification, CAV*, volume 3114 of *LNCS*, pages 496–500. Springer, July 2004.

[4] H. Fawzi, P. Tabuada, and S. N. Diggavi. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Trans. Automat. Contr.*, 59(6):1454–1467, 2014.

[5] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, and S. Savage. Experimental security analysis of a modern automobile. In *Proceedings of the IEEE Symposium and Security and Privacy*, Oakland, CA, 2010.

[6] F. Miao, Q. Zhu, M. Pajic, and G. J. Pappas. Coding sensor outputs for injection attacks detection. In *53rd IEEE Conference on Decision and Control (CDC)*, 2014.

[7] M. Pajic, J. Weimer, N. Bezzo, P. Tabuada, O. Sokolsky, I. Lee, and G. J. Pappas. Robustness of attack-resilient state estimators. In *5th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2014.

[8] J. Park, R. Ivanov, J. Weimer, M. Pajic, and I. Lee. Sensor attack detection in the presence of transient faults. In *6th ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, 2015.

[9] F. Pasqualetti, F. Dorfler, and F. Bullo. Attack detection and identification in cyber-physical systems. *IEEE Trans. on Automatic Control*, 58(11):2715–2729, 2013.

[10] A. Puri and P. Varaiya. Driving safely in smart cars. In *Proceedings of the 1995 American Control Conference*, 1995.

[11] SRI International. *Yices: An SMT solver*. http://yices.csl.sri.com/.

[12] A. Tiwari. Hybridsal relational abstracter. In *Proc. CAV*, volume 7358 of *LNCS*, 2012. http://www.csl.sri.com/~tiwari/relational-abstraction/.

[13] A. Tiwari, B. Dutertre, D. Jovanovic, T. de Candia, P. Lincoln, J. M. Rushby, D. Sadigh, and S. A. Seshia. Safety envelope for security. In *3rd International Conference on High Confidence Networked Systems (part of CPS Week), HiCoNS '14, Berlin, Germany, April 15-17, 2014*, pages 85–94. ACM, 2014.

[14] A. Zutshi, S. Sankaranarayanan, and A. Tiwari. Timed relational abstractions for sampled data control systems. In *Proc. CAV*, volume 7358 of *LNCS*, pages 343–361, 2012.

A HybridSal Model

For completeness, we include the HybridSal model here.

```
PVAtt: CONTEXT =
BEGIN

tstealth: REAL = 4;     % time in stealth mode
stbound : REAL = 2;     % attack bound in stealth mode
evbound : REAL = 2.8;   % attack bound in evident mode
DeltaT  : REAL = 2;     % Sampling period for observation

control: MODULE =
BEGIN
 LOCAL gap, vf, vl, af, time: REAL
 INPUT al, attack: REAL
 INITIALIZATION
  time = 0;
  af  IN {x:REAL | -5 <= x AND x <= 2};
  vf  IN {x:REAL | 0 <= x AND x <= 30};
  vl  IN {x:REAL | 0 <= x AND x <= 30};
  gap IN {x:REAL | x >= 5 AND (
    (vf >= vl AND x - 10 - (vf - vl) - 0.1 * (vf - vl) * 60 >= 0) OR
    (vf <= vl AND x - 10 - (vf - vl) - 0.1 * (vf - vl) *  0 >= 0))};

 TRANSITION
 [
 vl >= 0 AND al >= -5 AND al <= 2 AND vl' >= 0 AND vl'-vl = al*DeltaT AND
 attack <= stbound AND attack >= -stbound AND
 time < tstealth AND time' <= tstealth -->
  afdot' = -3*af - 3*(vf - vl) + gap - (vf + 10) - 4*attack ;
  vfdot' = af ;
  gapdot' = vl - vf ;
  vldot' = al;
  timedot' = 1;
 []
 vl >= 0 AND al >= -5 AND al <= 2 AND vl' >= 0 AND vl'-vl = al*DeltaT AND
```

```
 attack <= evbound AND attack >= -evbound AND time >= tstealth -->
  afdot' = -3*af - 3*(vf - vl) + gap - (vf + 10) - 4*attack;
  vfdot' = af ;
  gapdot' = vl - vf ;
  vldot' = al;
  timedot' = 1;
 ]
END;

correct : THEOREM
  control |- G(gap >= 0 OR vf <= vl );

END
```