Availability Assessment of Embedded Systems with Security Vulnerabilities

Andrew J. Kornecki
ECSSE Department
ERAU
Daytona Beach, FL 32114, USA
kornecka@erau.edu

Janusz Zalewski
Dept. of Computer Science
Florida Gulf Coast University
Fort Myers, FL 33965, USA
zalewski@fgcu.edu

Wendy F. Stevenson
Garmin, Ltd.
Olathe, KS 66062, USA
wendy.stevenson@garmin.com

***Abstract*—The paper discusses modeling assessment aspects of incorporating security into an embedded system. The essential assumption in the approach and the model we propose is that a security breach may cause degradation of the service and ultimately a failure. The security model concentrates on the system's interaction with the environment via a communication channel. The cooperative adaptive cruise control (CACC) system is used as a case study. The results of the study obtained for availability assessment due to security lapses show that a modeling technique may be used to evaluate the need for appropriate mitigation mechanisms allowing the system to withstand the attacks still assuring desired level of availability.**

Keywords-security; embedded systems; real-time systems; software design; availability; cruise control.

I. INTRODUCTION[1]

Security of computing systems has been traditionally understood as protection of unauthorized access to information (data and software) in centralized business applications. With the advent and widespread use of embedded devices, both business and personal applications are no longer limited to run in isolated environments, where physical separation and password controlled access provide sufficient protection and control of vulnerable resources. On the contrary, devices containing valuable information or providing access to such information may be distributed over a wide area, such as a building, a factory or a plant in industrial control systems, be literally spread over the entire world, as in banking systems, or travel themselves as mobile systems in transportation [1-12].

Providing security in such circumstances involves applying new techniques corresponding to new types of threats related to the use of embedded devices in distributed environments. Before such new techniques can be developed, models of these new threats have to be built to allow understanding of the nature of these threats and the development of effective countermeasures. One remarkable trend is to build security into the process of designing software for such devices.

The objective of this work is to define a model of embedded system security, determine potential vulnerabilities and assess the reliability and availability properties of the system. We consider scenarios where the system may or may not be repaired when the failure, due to security violation, is detected. The rest of the paper is structured as follows. Section II presents an overview of security threats in embedded systems and their respective countermeasures. Section III outlines our approach to reliability assessment, and Sections IV and V discusses a case study and simulation experiments, respectively. The paper ends with a conclusion in Section VI.

II. SECURITY THREATS AND COUNTERMEASURES

In recent embedded systems literature, multiple review papers exist that deal with specific sorts of security threats in embedded systems and their respective countermeasures across an entire array of applications. In the most recent work, Gebotys [1] lists several types of vulnerable embedded applications, including: contact smart cards, contactless smart cards and RFID tags, cell phones and PDA's, automobiles, game stations, FPGA's and networks on a chip. Potential threats, called attacks, may involve: probing or penetration attack, monitoring attack, manipulation attack, substitution attack, replay attack, modification attack, and splicing attack.

Ravi et al. [2] look at embedded systems security from the point of view of typical requirements, in which they include: user authentication, secure network access, secure communication functions, secure storage, content security, and system availability. Among inadequacies of current design techniques to ensure proper security, they list the following: processing gap, battery gap, flexibility, tamper resistance, and assurance gap.

Khelladi et al. [3] list similar challenges as in [2], and discuss particular solutions to increase security of embedded systems by using specific software design methodologies, tamper-resistance techniques, secure architectures, energy efficient schemes, and secure networking. Both authors in [2]

[1] A part of this work has been funded by a grant SBAHQ-10-I-0250 to Florida Gulf Coast University from the U.S. Small Business Administration (SBA). SBA's funding should not be construed as an endorsement of any products, opinions, or services. All SBA-funded projects are extended to the public on a nondiscriminatory basis.

and [3] pay attention to the role of costs in providing security for embedded systems.

Parameswaran and Wolf [4] list the following security related threats, calling them vulnerabilities of embedded systems: energy drainage (exhaustion attack), physical intrusion (tampering), network intrusion (malware attacks), information threat (privacy violation), introduction of forged information (authenticity violation), confusing or damaging of sensors/peripherals, thermal events (thermal virus or cooling system failure), and reprogramming of systems for other purposes (stealing). Then they categorize these attacks and describe briefly related countermeasures.

Dzung et al. [5] categorize potential attacks from the point of view of industrial communication systems, of which embedded systems are a crucial part. They list the following categories of attacks: denial of service, eavesdropping, man-in-the-middle attack, breaking into a system, and additionally distinguish between three categories related to malicious code (viruses, trojans, and worms). Among broad categories of countermeasures they list: cryptographic methods (crypto algorithms, message authentication, hybrid encryption, key distribution, and entity authentication), secure protocols at various layers of the ISO reference model (including firewalls, and intrusion detection systems), and secure architectures (including policies, weakest links, defense-in-depth, end-to-end security and similar methods).

What all these review papers have in common is that they list a whole array of potential security threats to embedded systems. Thus, to reasonably address an issue of embedded system security, one has to understand the big picture of embedded systems security and build a model, as a part of this picture, to define what particular issues are being approached and resolved, and how they may affect the entire view of making a system secure.

III. An Approach to Security Assessment

The Open Web Application Security Project (OWASP) maintains a list of known software vulnerabilities [13]. It has recently published the ten most critical web application security risks for 2010. These are ranked according to the risk they pose and they do not necessarily represent the ten most commonly occurring vulnerabilities. Some of these, which are relevant to embedded systems, include:

- Injection – hostile data are sent to system that can trick the system into executing unintended command.
- Broken Authentication/Session Management - attackers can compromise passwords, keys, session tokens, or exploit implementation flaws to assume other users' identities and thus access the system in a hostile way.
- Insecure Object References - a reference to an internal implementation object (file, directory, database key) so the attacker can manipulate these references to access unauthorized data.
- Incorrect Configuration – communication settings should be defined, implemented, and maintained since in many cases the system is not shipped with secure defaults.
- Insecure Cryptographic Storage - lack of appropriate encryption or hashing such that attackers may use weakly protected data to corrupt and misuse internal system data.
- Insufficient Transport Layer Protection - failure to encrypt network traffic necessary to protect sensitive communications (due to weak or incorrectly applied algorithms, expired or invalid certificates, etc.).
- Code Exposure - quality vulnerability where undetected software defects (e.g. uninitialized variables) may be exploited by attackers.
- Error Handling - improper cleanup after exception, empty exception blocks and uncaught exceptions may lead to security vulnerability.
- General Logic - faulty logic opening the system to misuse by attackers (like passing objects to untrusted methods, use of an easily guessed or visible temporary file, etc.).
- Input Invalidation - failure to adequately validate the data input.
- Protocol Errors - failure to check for certificate revocation or expiration, failure to protect stored data from modification.

Respective mitigation techniques include authentication, encryption and other means designed to avoid and/or detect intrusions. There is no single method to evaluate the security of a new mitigation technique before it is used in a system. This is partly due to the fact that security is relative to the currently known possible attacks. Thus, we can only say that a system is secure against these known attacks, whose nature, that is, a model is known.

The essential assumption in the approach and the model we propose is that a security breach may cause degradation of system services and ultimately a failure. For an embedded system working in a 24/7 regime, at any given point in time the system will be in one of the following states: (1) the normal state; (2) the failure state; or (3) one of the possible intermediate degraded states, depending on the occurrence of failures and the system's ability to detect and recover from those failures. This is true for any system whether it has security functionality or not.

Due to the dynamic state transition observed here, we consider Markov model which allows analysts to study events over time. The Markov model of a real system typically includes the state with all elements operating and a set of intermediate states representing partially failed conditions, leading to the state in which the system is unable to perform its function. The model may include repair transition paths, as well as failure transition paths. The corresponding failure and repair rates between states can be derived from system analysis. The Markov model equations describe situation where transition path between two states reduces the probability of the state it is departing, and increases the probability of the state it is entering, at a specified rate.

A Markov chain may be constructed to model the changes of the system state depending on the probability of certain security vulnerabilities and probabilities that the system can be

restored back to the regular state. The transition paths and the probability of transition between the various states define the system equations whose solution represents the time-history of the system. In a regular Markov model, the state is directly visible to the observer, and therefore the state transition probabilities are the only parameters.

In the system where deviations from regular state are due to potential external attacks, we can identify the attack's effects and their probabilities (e.g., percentage of messages corrupted, lost, incorrectly inserted, etc.). In turn, system transitioning to such degraded state can return to the regular state due to application of a mitigation mechanism or fail (also with assumed rate). Even after failure, we can consider system that can be repaired assuming certain repair rate. Such model is useful to quantify how a specific vulnerability, expressed in terms of probability of potential attacks, affects the system and what is the effective reliability and availability of the system. Respective simulations can give a basis for understanding how effective each mitigation method needs to be in order to ensure that the system meets or exceeds its required level of security.

IV. Case Study Description

The Cooperative Adaptive Cruise Control (CACC) system selected as the case study [14, 15], is an embedded control system for automobiles which automatically monitors and adjusts a vehicle's speed according to the traffic conditions in its immediate vicinity. The CACC system on a vehicle will receive the necessary data it needs from sensors on the vehicle, maintain communication with the CACC system in the vehicle directly in front of it and communication with a central Street Monitoring Data Center. In order to effect a change in the vehicle's speed, the system issues commands to the throttle device as well as to the brake pedal.

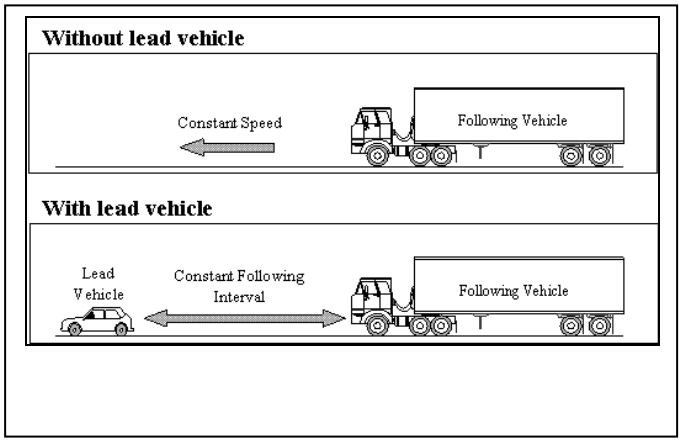

Figure 1. Operation of CACC.

The system is not completely autonomous in that the user sets the desired speed of the vehicle and has the option to turn off the CACC system. However, the long term trend is towards partial or even fully autonomous operation of a single vehicle, or even groups of vehicles. Therefore, safety and security are essential attributes of CACC systems.

The diagram in Fig. 1 shows the desired operation of the CACC system from the perspective of the truck. For the CACC architectural model used in this case study, the Constant Following Interval is a function of the current lead vehicle's speed, the condition of the road, the amount of traffic present and documented safe following distances. The user sets the desired speed when there is no vehicle immediately in front of them.

The overall context diagram of the CACC system is shown in Fig. 2. CACC is an embedded system with external interfaces prone to malicious attacks.

To enhance the level of security, changes have been made to the original CACC architectural model by adding authentication and secure communication components. The reliability and availability of the enhanced model have been analyzed by applying a Markov model.

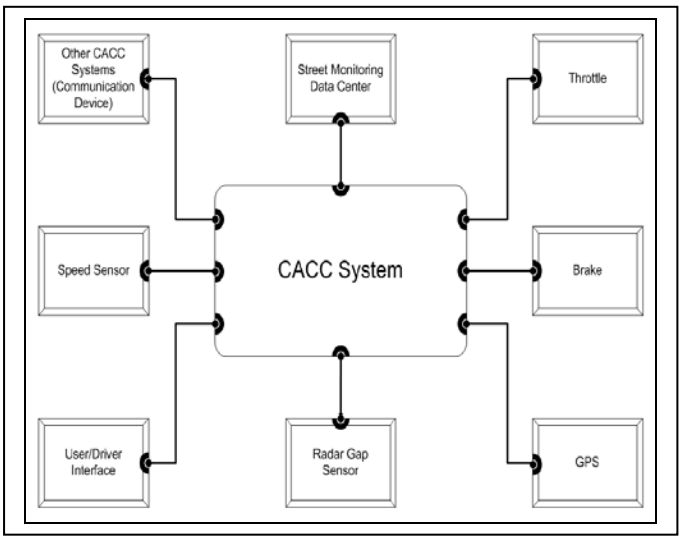

Figure 2. Context diagram of the CACC design.

The CACC system communicates via wireless network connection to other vehicles in the vicinity and to a centralized Street Monitoring Data Center (SMDC). These interfaces have the highest risk of malicious attack. Security violation can be resulting from external connection, i.e., by incorrect data from Street Monitoring Data Center or from external communication with other CACC equipped vehicles.

The specific vulnerabilities fall into four categories:

- Message Introduction: an untrue SMDC or Other CACC message is injected.
- Message Deletion: SMDC or Other CACC message is not received by the CACC system.
- Message Corruption: the contents of an SMDC or Other CACC message are altered before being received by the CACC system.
- Message Flooding: multiple frequent SMDC or Other CACC messages are received causing the CACC system to choke and not perform its required tasks within the

deadlines (this vulnerability occurs in conjunction with message introduction).

For a system with no security functionality, it will not be possible to transition from a degraded state back to the normal state. By adding security functionality, this transition will be made possible and so the resulting system reliability will be increased. The level of security added will determine the probability with which this transition will occur. As the probability for detection and recovery increases the overall reliability of the system increases.

For the purposes of this case study, the following assumptions and simplifications were made:

- It is assumed that only one security breach can occur at any given time.
- Only one message channel is modeled. To model both the SMDC and the Other CACC channels simultaneously, more states and associated transitions would be required.
- It is assumed that an undetected or unsuccessfully handled security breach leads to system failure even though the system may not physically fail. It may not be aware that a security breach has occurred.

V. Simulation Experiments

Relex [16] reliability analysis software was used to assess the reliability of the Markov model shown in Fig. 3. Each edge in the state machine is labeled with the probability that that particular state transition in the following figures will occur.

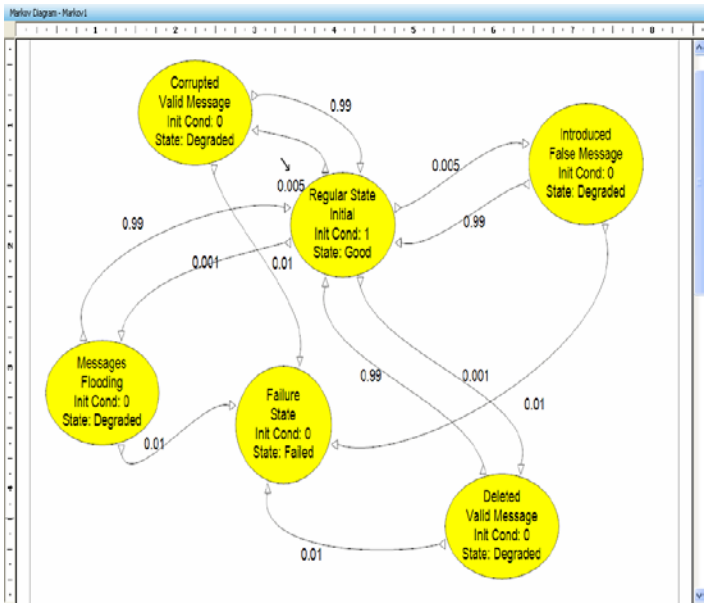

Figure 3. Markov Model of the CACC System (no repairs)

For the presented example, we initially assumed (Case 1) that the probability of corruption of a valid message or an injection of a false message to be five over 1,000 received messages. The probability of having a message deleted or getting a flood of false messages is assumed to be one over 1,000 messages. All these represent the probabilities of security breaches occurring. These values are dependent on the environment in which the system is to operate and depend on the system and the mitigation methods chosen. We repeated simulations for Case 2 where the probabilities of attack were one range higher, so that:

- Case 1: 5/1000 for corruption and insertion and 1/1000 for deletion and flooding
- Case 2: 5/100 for corruption and insertion and 1/100 for deletion and flooding.

To decrease the probability of failure, the probability of successful security breach detection and recovery would need to be increased. The related attack detection and recovery probabilities for the simulation experiments were 99%, 90%, and 50%. The respective failure probabilities (i.e. failed detection and/or mitigation of the attack) were: 1%, 10%, and 50%. The probabilities for the case of 99% recovery and 1% failure are listed on connections between degraded states either back to regular state or to the failed state shown on Fig.3. The experiment probability values are listed in Table I.

TABLE I. Probabilities of Transitions.

Probability	***Assigned Values***
Corrupted Valid Message	Case 1:0.005/Case 2: 0.05
Introduced False Message	Case 1:0.005/Case 2: 0.05
Deleted Valid Message	Case 1:0.001/Case 2: 0.01
Message Flooding	Case 1:0.001/Case 2: 0.01
Return to Regular (security mitigation working)	0.99/ 0.9/ 0.5
Failing to Return to Regular (security mitigation failure)	0.01/ 0.1/ 0.5
Return from Fail to Regular (Repair)	0 (no repair)/0.5/0.9

Respective simulation results are presented in Fig. 4. We assumed six hours simulation window. The baseline model does not have a transition from the failure state to the normal state, which corresponds to a system which has no possibility of repair after it has failed. This implies that the system availability will be equal to the system reliability which is shown to be the case (see Fig. 4). The resulting reliability is 93.0531% with mean availability of 98.9873%. Since the first time to failure is estimated to be 83.33 hours, we can deduce that in such scenario our system meets the expectations imposed on security.

It has to be noted that in general, an increase in this probability of successful attack detection and mitigation /recovery would result in a decrease in system performance because more effort, and hence execution time, is needed to increase the detection and recovery rate. A tradeoff, therefore, needs to be made, since performance is an important consideration for real-time embedded systems. Such analysis can be done with exploring the complexity of the applied mitigation methods including authentication, secure communication, etc. An appropriate timing analysis can help determine impact of the introduction of security measures on the system efficiency.

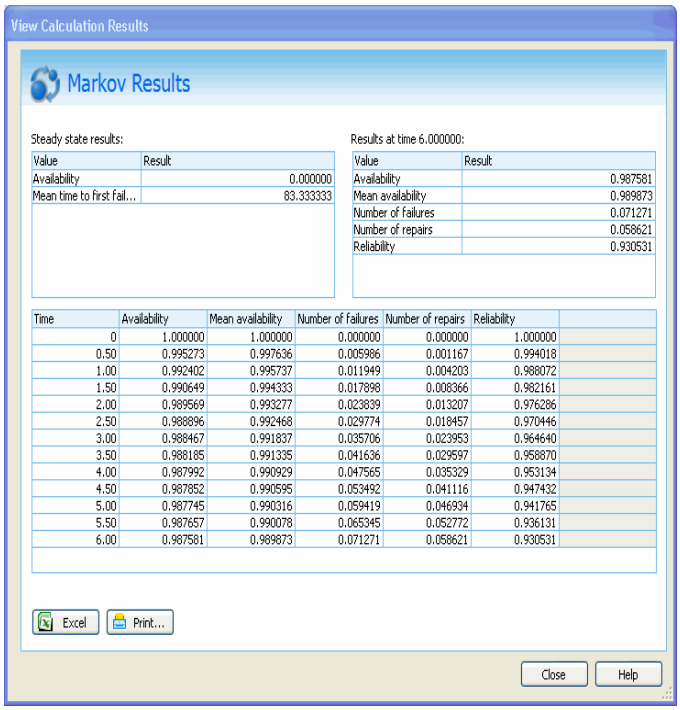

Figure 4. Markov Simulation Results (Case 1, 99%, no repairs).

In the Markov model, the failure state was modeled as a terminal state. However, availability of the system could be increased by adding possibility of a repair. In the CACC system, the monitor process is able to reset the system, if it detects a failure. This would be equivalent to a transition from the failure state back to the normal state.

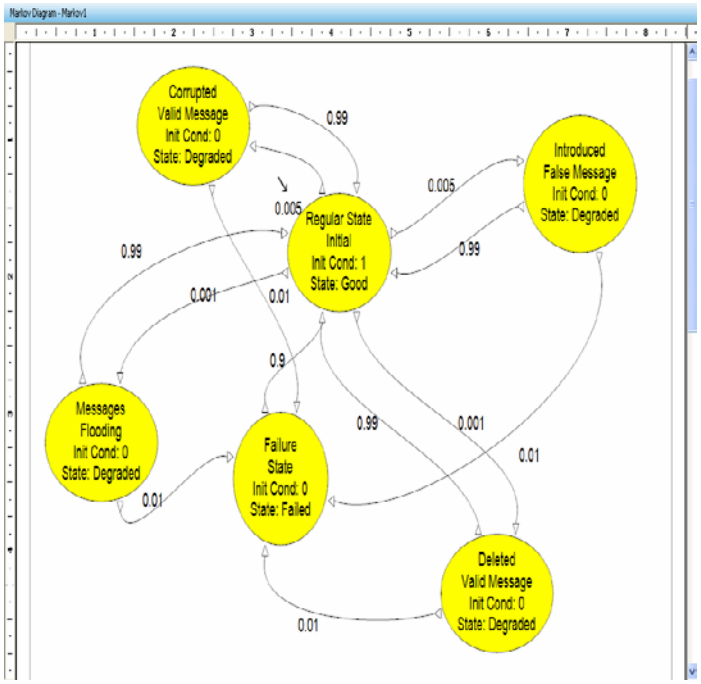

Figure 5. CACC Markov Model Showing Repair from the Failure State.

The updated Markov Model is shown in Fig. 5. The repair rate of the system is a parameter one would like to explore. Two cases, one with repair rate 0.9 and other with rate of 0.5, are shown in Fig. 6 and 7 respectively.

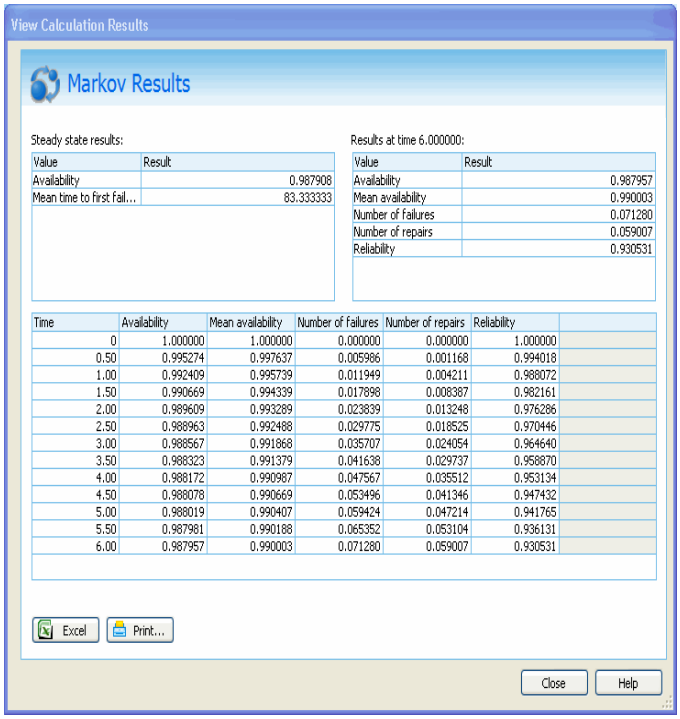

Figure 6. Markov Simulation Results (Case 1, 99%, repair rate of 0.5)

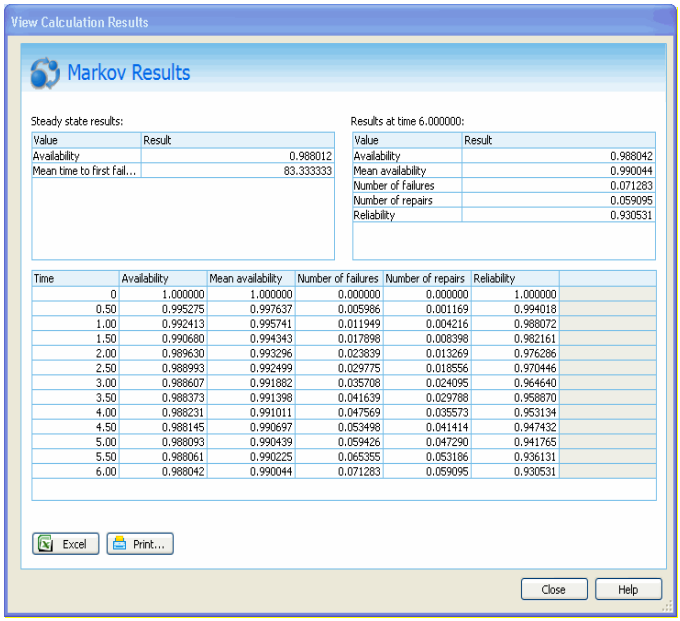

Figure 7. Markov Simulation Results (Case 1, 99%, repair rate of 0.9)

The results are presented in Table II for two case scenarios. It is shown that reliability of the system is not affected by the addition of system repair. However, the repair increases the availability of the system. A higher repair rate results in only a marginal increase in the system availability. For a system with

higher attack rate (Case 2), when only half of attacks are mitigated, introducing 0.9 probability of repair increases availability by 8.56%. Similar scenario when up to 90% of attacks are mitigated gives availability increase of only 1.65%. We may infer that much effort needs not be expanded in designing system with a repair capability. For higher attack rates, more value may be obtained in increasing the detection and recovery rates for the specific attacks, i.e., security mitigation rather than expanding resources on system repair.

TABLE II. SIMULATION RESULTS.

Attack rate	Security Mitigation	Repair Rate	Reliability After 6 hrs	Mean Availability	Estimated first failure
Case 1	50%	none	93.0531	97.7408	83.33 hrs
		0.5		98.3877	
		0.9		98.5872	
	90%	none		98.7572	
		0.5		98.8874	
		0.9		98.9275	
	99%	none		98.9873	
		0.5		99.0003	
		0.9		99.0044	
Case 2	50%	none	48.6751	80.4847	8.33 hrs
		0.5		85.7505	
		0.9		87.3755	
	90%	none		88.7127	
		0.5		89.8385	
		0.9		90.1779	
	99%	none		90.6804	
		0.5		90.7943	
		0.9		90.8290	

VI. CONCLUSION

Security issues in embedded systems may look on the surface very similar to traditional protection of enterprise systems. However, in fact, they are very different due to the addition of system's distribution aspects and the existence of sensors. Consequently, different models for assuring security in embedded systems have to be built.

The authors suggest viewing the attack and related security breach as a system failure, thus allowing to reason about security by modeling availability and reliability with existing tools. A model of a Cooperative Adaptive Cruise Control was used to determine system's sensitivity to security breaches. For the defined probabilities of attack and specified probabilities of detecting and handling the attack effects, Markov model allows estimating the system availability, thus providing valuable information on the level of security to be applied.

REFERENCES

[1] C.H. Gebotys, Security in Embedded Systems. Berlin: Springer-Verlag, 2010.

[2] S. Ravi, A. Raghunathan, P. Kocher, and S. Hattangady, "Security in Embedded Systems: Design Challenges," ACM Trans. on Embedded Computing Systems, vol. 3, no. 3, pp. 461-491, August 2004.

[3] L. Khelladi, Y. Challal, A. Bouabdallah, and N. Badache, "On Security Issues in Embedded Systems," Intern. Journal of Information and Computer Security, vol. 2, no. 2, pp. 140-174, 2008.

[4] S. Parameswaran, and T. Wolf, "Embedded Systems Security – An Overview," Design Automation for Embedded Systems, vol. 12, no. 3, pp. 173-183, 2008.

[5] D. Dzung, M. Naedele, T.P. von Hoff, and M. Crevatin, "Security for Industrial Communication Systems," Proceedings of the IEEE, vol. 93, no. 6, pp. 1152-1177, June 2005.

[6] P. Koopman, "Embedded System Security," IEEE Computer, vol. 37, no. 7, pp. 95-97, July 2004.

[7] Y.C. Kim, and J.T. McDonald, "Considering Software Protection for Embedded Systems," CrossTalk – The Journal of Defense Software Engineering, vol. 22, no. 6, pp. 4-8, September/October 2009.

[8] D. Serpanos, and J. Henkel, "Dependability and Security Will Change Embedded Computing," IEEE Computer, vol. 41, no. 1, pp. 103-105, January 2008.

[9] K. Stammberger, "Current Trends in Cyber Attacks on Mobile and Embedded Systems," Embedded Computing Design, vol. 7, no. 5, pp. 8-12, September 2009.

[10] S. Schulze et al., "On the Need of Data Management in Automotive Systems," Proc. BTW'2009, Datenbanksysteme in Business, Technologie und Web, Münster, Germany, March 2-6, 2009, pp. 217~226.

[11] A. Kornecki, and J. Zalewski, "Safety and Security in Industrial Control," Proc. CSIIRW 2010, 6th Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tenn., April 21-23, 2010.

[12] R.R. Brooks, S. Sander, J. Deeng, and J. Taiber, "Automotive System Security: Challenges and State-of-the-Art," Proc. CSIIRW'08, 4th Annual Workshop on Cyber Security and Information Intelligence Research, Oak Ridge, Tenn., May 12-14, 2008, Article No. 26.

[13] E. Chickowski, OWASP Top 10 - 2010: The Ten Most Critical Web Application Security Risks (release candidate). OWASP, Retrieved January 7, 2011. URL: http://www.channelinsider.com/c/a/Security/Top-10-Most-Critical-Web-App-Security-Risks-298234/

[14] Michigan State University, Computer and Software Engineering, CSE491-602 Class Projects, Fall 2006, Retrieved March 3, 2010. URL: http://www.cse.msu.edu/~chengb/RE-491/Projects/cacc_msu-ford.pdf

[15] W. Stevenson, "Evaluating the Impact of Adding Security to Safety Citical Real-Time Systems", Graduate Research Project, Embry Riddle Aeronautical University, Daytona Beach, Florida, Spring 2010

[16] Relex Reliability Prediction Tool, PTC Product Development Company, Needham, Mass. Retrieved January 7, 2011. URL: http://www.ptc.com/products/relex/reliability-prediction