Enthymeme: A Zero-Trust, Formal Verification System for Cyber-Physical Networks

Francis Mendoza
CSE598: Safe Autonomy For Cyber-Physical Systems
Fall 2021
fmendoz7@asu.edu
Final Report

***Abstract*—My main category is implementation and research. Topic: "Enthymeme: CPS Security and Privacy within heterogeneous, high-scale, Byzantine environments" (modified version of the original CPS security & privacy project).**

***Index Terms*—Formal Verification, Cyber-Physical Systems, Consensus Protocols, Hybrid Automata, Timed Automata**

I. Introduction

As a software engineer and whitehat hacker who has worked on production-grade Internet-Of-Things (IoT) systems deployed at the edge, there are several grave implications regarding the lack of rigorous cybersecurity for Cyber-Physical Systems (CPS). With the global state of contemporary "greenfield" IoT device adoption (exclusive of legacy "brownfield" SCADA systems) reaching over 20 Billion Devices and owned by disparate parties in a functionally Byzantine environment, it is self-evident that the superbroad attack surface and contentious incentive schema prioritizing speed of adoption represents a looming threat to public safety and the future of the impending "Automation Revolution". Everything from mission-critical infrastructure (ie: Water Treatment plants) to consumer-grade smart devices is a potential target. Contributing to a more rigorous, high-scale security posture to prevent catastrophic cyberattacks is critical for the continued use of various "smart" technologies, such as autonomous vehicles & smart city infrastructure, integral to the "Automation Revolution".

The Internet Of Things is heavily divided into niche use-cases (ie: water treatment, power systems, autonomous vehicle fleets, etc.) dominated by various corporations that compete with each other either maliciously or semi-honestly. Certain use-cases may need to exchange data in a secure manner (ie: competing autonomous vehicle ride-share services to inform an overarching municipal traffic policy) in order to achieve an end result that benefits all parties involved, without leaking confidential information that puts them at a business or cybersecurity disadvantage. And although there is much prior art regarding formal verification for computer systems security at an individual device level, not much prior art exists at the network level within a Byzantine environment, for the overarching Cyber-Physical Network (CPN). This project aims to utilize temporal logic requirements, model-based verification, and consensus mechanisms to enable confidential, secure, and authentic data sharing on CPS models and overarching device utilization policy in a byzantine environment, with the Russia-EU-Ukraine natural gas pipeline system as a motivating example for our CPN.

II. Work Performed

A. *Overview: Methodology, Material, Instructions*

The original contributions of Enthymeme, as it is a modification of the original Russia-EU-Ukraine natural gas pipeline project proposed as a viable topic, are two Simulink-Stateflow (SLSX) models modeling an insecure version and a secure version of the natural gas pipeline. There is an additional SLSX model regarding a separate model demonstrating the CPN Consensus layer. The code, which is attached as part of the situation folder, comprises of SLSX files and a setup.m file. For my implementation, I utilized the Simulink and Stateflow libraries to supplement standard MATLAB.

In terms of methodology, temporal logic and hybrid automata were the mainstay of this particular project, as the gas pipeline's Flow Controller Systems (FCS) utilize timesteps to guide their execution in relation to the behavior of other gas plants.

B. *Original: (Insecure) Gas Pipeline*

The first contribution of this final project was an insecure SLSX implementation of the Russia-EU-Ukraine gas pipeline model. An insecure model was engineered on purpose to demonstrate both normal operational scenarios as well as various attacks that exploit the implicit information flow inherent in the model, which are elaborated further in the "Experiments" part of the report.

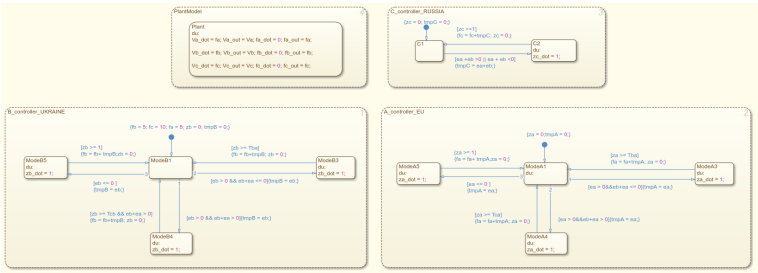

Fig. 1. SLSX Model Of Insecure Gas Pipeline

There are various variables of interest that are utilized in the SLSX models. T_x represents the required time for gas at x to be transferred. V_x represents the volume of gas at a particular plant x. f_x represents the flow rate of gas at x, where positive flow rate implicate more exports than imports, and vice versa if negative. e_x represents the change in user demand at x.

1) ***Problem: Implicit Information Flow****:* In the primary literature source referencing the gas pipeline for a static analysis tool, the first-authored paper by Luan Viet Nguyen[4] proposes a static analysis checker to determine information flows that could compromise a cyber-physical system. The authors seek to preserve the non-interference property

Definition 9 (Non-interference): A hybrid system H is noninterference secure if for every pair of initial state values $x_0, x_0^{'} \epsilon$ Init, and a pair of input signals $u, u^{'} \epsilon$ U for $x_0, x_0^{'}$ respectively, the following condition holds:

$$(x_0, u) = low(x_0^{'}, u^{'}) \Rightarrow \Sigma(x_0, u) = low\Sigma(x_0^{'}, u^{'})$$

The authors explained that values critical to the operation of a cyber-physical system (CPS), such as the volume of gas in a gas plant and the flow rate, are given generalized security values of **high** or **low**:

- **High**: Information that is kept secret
- **Low**: Information that is publicly observable

The authors also expounded upon two classes of vulnerabilities regarding information flow in order to preserve the non-interference property for hybrid systems: **Explicit flow** and **Implicit flow**.

- **Explicit Flow**: The value of a low variable is directly derived from the value of a high variable.
- **Implicit Flow**: The value of a low variable is updated indirectly due to information read from a high variable.

For example, assuming "h" is a variable with high security value and l is a variable with low security value, a guard condition or assignment such as $h = l$ yields explicit information flow, whereas a guard or assignment such as $l_1 = l_2$ or $l = l + 1$ does not yield implicit information flow (and is therefore safe), but a guard such as $(l > h) \rightarrow (l = 0)$ does, as it utilizes a high variable indirectly by referencing it within a conditional. In our particular example, the Russia-EU-Ukraine gas pipeline that is directly shown in [4] contains this implicit flow vulnerability.

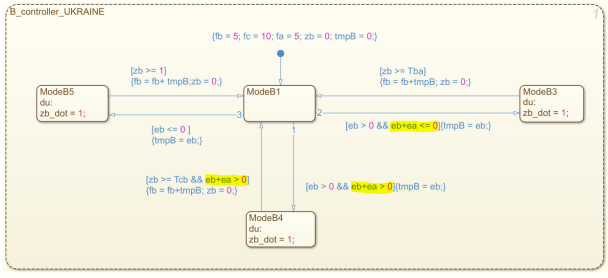

Fig. 2. Implicit Flow Discovered with Guards For Model B

It is given that model A (EU) and model C (Russia) are encrypted, while model B (Ukraine) is not and is hence publicly observable. Because of this, any attacker can observe critical variables assigned low security (such as f_b, e_b, and V_b) and thus infer information (such as the gas demand of the EU, e_A) and devise various attacks, such as a Gas Siphon attack, where B covertly steals gas from A, hoping to camouflage their actions amidst natural phenomena.

C. *Original: (Secure) Gas Pipeline*

The next contribution of the final project was a secure SLSX implementation of the Russia-EU-Ukraine gas pipeline model. A secure model was engineered with the necessary modifications to guards, states, and state variables in order to prevent implicit information flow at the Ukraine plant. This is important, as the attacker is no longer able to determine or probabilistically infer the state of high-security values, such as the change in demand for the EU (e_A), which is valuable information as to what types of attacks to launch and when. The more secure model is shown in Figure 3 below.

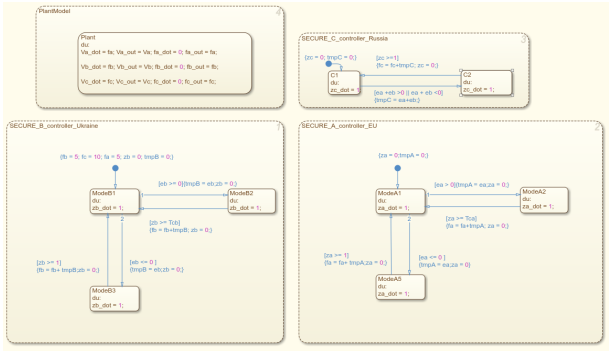

Fig. 3. SLSX Msodel of Secure Gas Pipeline

1) ***Solution****: Redefined Guards & State Variables:* In order to solve the implicit information flow problem, the entire state machine had to be rewritten where critical high-security variables (e in particular) were not referenced or used from any guard that was in a publicly observable (low-security) setting. Hence, model B was significantly revamped to reflect this.

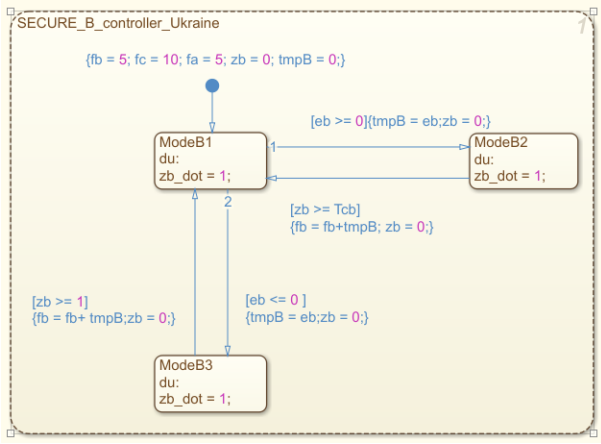

Fig. 4. Revised Guards Within Secure SLSX Model

D. ***Extra Credit***: *Cyber-Physical Network Consensus*

As an addendum to the original natural gas project, Dr. Fainekos approved the additional implementation of a communication layer and consensus algorithm for extra credit. The intent is to improve the security functionality of the preexisting system in a Byzantine (zero-trust) environment, where different cyber-physical systems are owned by various stakeholders that don't trust each other, but need each other's cooperation in pursuit of a larger goal that has far greater utility in cooperating than in competing. The intent of Enthymeme is to append functionality, particularly for a collection of CPS, to guarantee safety and the greater result despite the potential for malicious and semihonest behavior.

The motivating scenario in our project is to use Enthymeme's consensus layer to enforce a policy among a network of four insecure natural gas subpipelines, each similar in architecture to the original, where the flow rates of the publicly observable (and hence, vulnerable) Ukraine grid are all kept the same as part of the global state. This global state, in a practical setting, could be enforced by regulatory bodies due to environmental concerns, or business authorities due to unexpected fluctuations in supply. As Ukraine has been caught siphoning natural gas from Russia in 2015[4], distrust is assumed and an interoperable and clear means is necessary to enforce the rules of the policy while preventing exploitation by various parties. A high-level SLSX model of Enthymeme is shown in Figure 5 below.

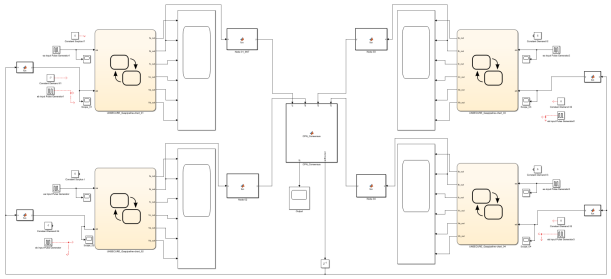

Fig. 5. SLSX Model for Enthymeme deployed on a gas pipeline CPN

1) ***Enthymeme***: *Novel Contributions:*

- **Consensus Layer**: Naive Voting-based consensus schema to check for discrepancies within global state values and enforce a predefined policy on critical parameters
- **Byzantine CPS Self-Correction**: Based on the data acquired from the consensus layer, Enthymeme is joined with additional Simulink blocks for self-correcting the inputs from malicious or semihonest peers for immediate policy enforcement and to prevent catastrophic failures
- **Backwards Compatibility**: Enthymeme can be deployed to any insecure CPS to limit the scope and extent of damage from attackers by enforcing policy on critical inputs, if the underlying logic of individual CPS cannot be changed

III. EXPERIMENTS

We performed two primary experiments in this paper: The Gas Siphoning Attack that was referenced in the original paper [4] as well as a test of both the consensus and self-correcting functionality that is part of Enthymeme. There are three SLSX models total: the first two models are for the original scope of the problem: one insecure and one secure. The third SLSX model details Enthymeme's functionality specifically.

To run, a setup.m file is first run to load user-defined transfer rates for all plants, into all models. Then, one must go into the SLSX model of their choosing. Depending on the scenario they want to run, they can freely switch between a pulse-based input for e variables (for a sudden version of the Gas Siphon attack) or a constant input (for a sustained version of the Gas Siphon attack). Enthymeme output blocks for the consensus mechanism denotes y_0 of 1 as "Normal", 10 as "Under attack, but Fault Tolerance unbroken", and -1 as "Fault Tolerance breached". A video demonstration of the system in action has been attached to the folder to clearly walk through setup and testing, as navigating the models are quite complex to explain succinctly.

A. ***Original***: *Normal Operation & Attack Scenarios*

1) Normal Operating Scenario: In the normal operating scenario, constant inputs for the e values were given, all positive to denote their desire to consume more gas. The flow rate of A (EU) and B (Ukraine) increases in staggered jumps (as their request needs to be processed by the supplier), while plant C (Russia), who is the supplier, has flow rate increase linearly.

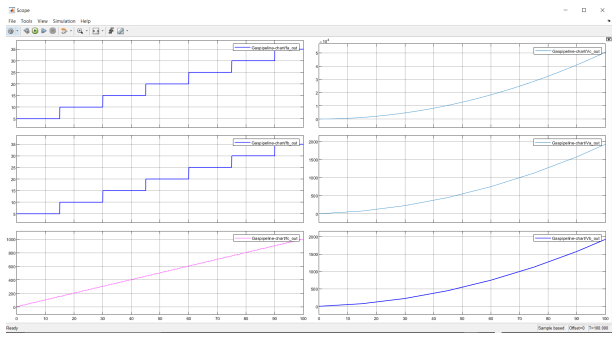

Fig. 6. Output Detailing Normal Operation

2) Gas Siphon Attacks: In the Gas Siphon Attack scenario, we assume the attacker has full observability and mutability of the input parameters for B, but cannot change the underlying logic of the model for B itself. The two parameters the attacker can control are the gas transfer time from Ukraine to EU (T_{ba}) and the change of demand (e_B). For the sudden method of gas siphoning, the attacker sets an abnormally high value for the gas transfer time from Ukraine to the EU in order for the balance of siphoned gas to be revealed all at once. Alternatively, for the sustained method of gas siphoning, the attacker does not change the gas transfer time. It is assumed that the gas transfer time from the EU to Ukraine is smaller

than the gas transfer time from Russia to Ukraine. The attacker also utilizes a pulse input that jumps to a negative version of the input for eA (deduced through frequency attacks) in order to snap up the momentary surplus. Figure 7 details setup, while Figure 8 details the results of the attack on the insecure model, in contrast to Figure 9, which details results of how it foiled the attack by immediately dropping the offending CPS to a constant rate. Negative flow rate denotes higher imports of gas than exports.

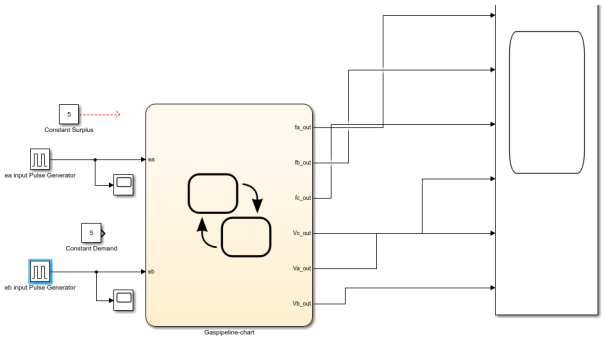

Fig. 7. Lower-Level SLSX Diagram Of Gas Siphon Attack

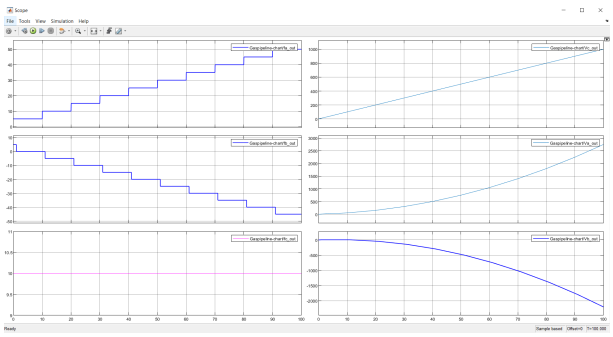

Fig. 8. Output Diagram of Sustained Gas Siphon Attack (Insecure)

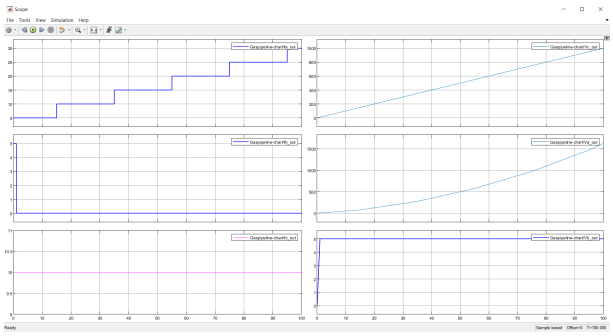

Fig. 9. Output Diagram of Failed Gas Siphon Attack (Secure)

B. ***Extra Credit***: *Cyber-Physical Network Consensus*

In the Enthymeme SLSX model, a CPN of four individual insecure gas pipeline models were joined together by a MATLAB function block that contained the naive-voting consensus logic. Different inputs for f_b, the parameter desired to be the same across the global state, can be altered individually for each natural gas CPS to be of any value and can either be pulsed or constant.

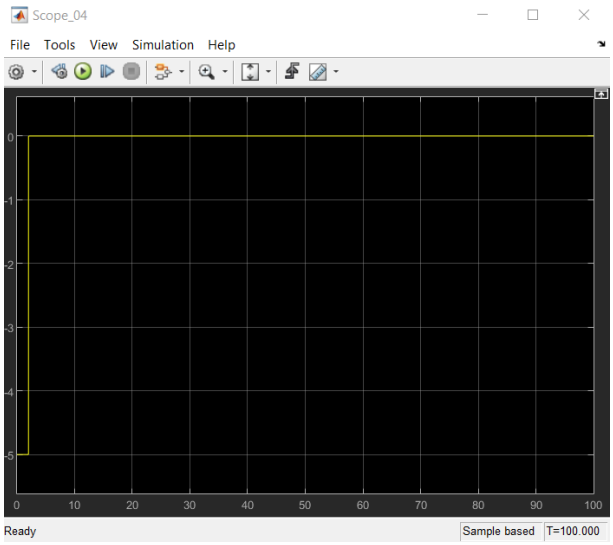

Fig. 10. Self-Correcting Behavior Modeled In Malicious, Insecure CPS

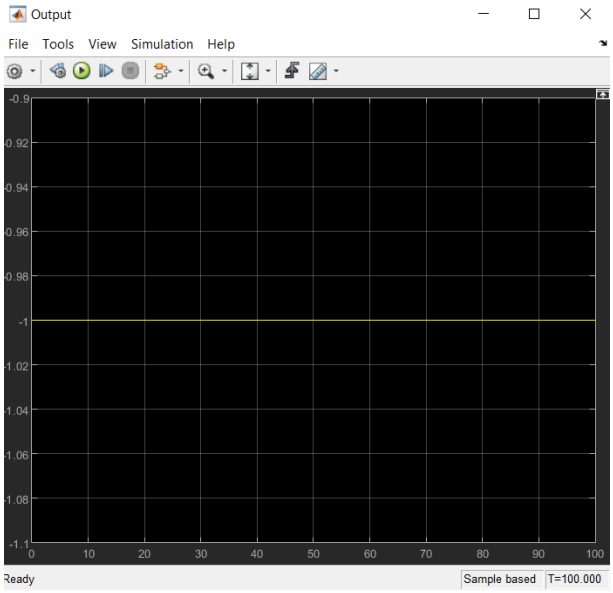

Fig. 11. Output Modeling Broken Fault Tolerance

The acting fault-tolerance of Enthymeme is 33%, akin to Practical Byzantine Fault Tolerance (PBFT), meaning that for a four-node network, only one can afford to act maliciously (in this case, having a different f_b). In this scenario, we have two f_b values differentiate from the other two, breaking fault tolerance and yielding a scope output of "-1". Once broken, the network no longer tries to collaborate as consensus is no longer reliable. However, I successfully replicated the secure CPS self-correcting behavior in the insecure CPS without altering

the underlying CPS logic, by forcing all f_b inputs to stay at a constant before consensus is reattempted. Hence, Enthymeme is able to provide a subset of similar security benefits as a secure CPS in a reactionary fashion to avaialble CPNs.

IV. Status and Lessons Learned

Enthymeme has gone through an unparalleled frequency of changes of wild magnitude for a class project, primarily due to incorrect assumptions regarding tooling and knowledge I could've only found out later. The final product for Enthymeme, in terms of desired method, has very little resemblance to the original plan for the method, although the desired core results are mostly similar. I had to start from scratch and abandon my prior work when I discovered the infeasibilty of modeling the CPS models through SimuPy, as there were blocks in Simulink and Stateflow that either don't exist or would be too complex to model. If I was to do the project again, I would immedatiely start with stateflow, simulink, and MATLAB and try to determine how to write the consensus algorithm using a MATLAB S-Block with C code, as MATLAB lacked many of the desired networking and cryptography functionality I wanted, with C being the closest compatible contender. I would've also use CORA for reachability analysis and Static Tool proposed in Nguyen et al regarding information flow analysis in order to provide comprehensive security guarantees if I had more time. If some other student was trying to do my project, I would advise against scope creep and encourage them to do full research on the extent of the capabilities of available tools so they don't waste any progress.

V. Relevance To Course

Since Enthymeme is a superset of the original CPS Security and Privacy project (an approved topic choice for this class), the primary material from the original project utilized involve temporal logic and the usage of formal verification through timed automata will be utilized. Enthymeme's contributions, however, will make use of hybrid automata and temporal logic as part of consensus for the overarching Cyber-Physical Network.

VI. Conclusion

Enthymeme is a superset of the original CPS Security and Privacy project (an approved topic choice for this class), which aims to produce novel contributions: namely, a consensus algorithm for an overarching Cyber-Physical Network. If I had more time, I would've tested CORA for reachability analysis as well as implementing a more rigorous type of consensus in C code in a MATLAB S block, while determining how to utilize network functionality.

References

[1] Kang, Eunsuk, et al. "Model-Based Security Analysis of a Water Treatment System." Proceedings of the 2nd International Workshop on Software Engineering for Smart Cyber-Physical Systems, ACM, 2016, pp. 22–28, doi:10.1145/2897035.2897041.

[2] Amin, Saurabh, et al. "Cyber Security of Water SCADA Systems-Part I: Analysis and Experimentation of Stealthy Deception Attacks." IEEE Transactions on Control Systems Technology, vol. 21, no. 5, IEEE, 2013, pp. 1963–70, doi:10.1109/TCST.2012.2211873.

[3] Nguyen, Luan, et al. "Detecting Security Leaks in Hybrid Systems with Information Flow Analysis." Proceedings of the 17th ACM-IEEE International Conference on Formal Methods and Models for System Design, ACM, 2019, pp. 1–11, doi:10.1145/3359986.3361212.

[4] Xiao, Feng, and Long Wang. "Asynchronous consensus in continuous-time multi-agent systems with switching topology and time-varying delays." IEEE Transactions on Automatic Control 53.8 (2008): 1804-1816.

[5] Fischer, Michael J., Nancy A. Lynch, and Michael S. Paterson. "Impossibility of distributed consensus with one faulty process." Journal of the ACM (JACM) 32.2 (1985): 374-382.