


Simulated Cyber Attacks Performed on Manufacturing Systems Built in MATLAB's Simulink Software

 Manbir S. Sodhi, PhD.^α,  Marwan Abdaletti, PhD.^α, and  Zachary A. deWardener *^α

^α Mechanical, Industrial, and Systems Engineering, The University of Rhode Island, Kingston, USA.

ABSTRACT

In recent years, Industry 4.0 has taken over the manufacturing space. Newly developed Industrial Internet of Things (IIOT) devices are released and implemented into these increasingly autonomous processes every day. In classic near-sighted fashion however, these devices were built with very minimal defense capabilities against cyber attacks. the scope of our research in this field is to generate simulated machine process data for use in the fine tuning of an anomalous threat detection model built by Dr.Abdaletti to mimic the intrusion detection system developed by the DS2OS team.

KEYWORDS: Simulink; Simulation; liot; Industry 4.0; Simevents; Ids.

1 INTRODUCTION

In recent years, the use of IIOT technology in manufacturing industries has increased significantly, leading to a greater need for security measures to protect these systems from cyber attacks. Cyber attacks on manufacturing systems can have severe consequences, including loss of sensitive information, production disruptions causing financial losses, and even loss of life due to uncontrollable machinery.

In order to combat these malicious events, companies worldwide are committing vast amounts of capital towards programmatic solutions (i.e., Intrusion Detection Systems (IDS)) backed by Machine Learning (ML) algorithms used to spot malicious behavior. However, these algorithms must be trained on real-time data from the system both at normal working conditions and when an attack scenario is taking place. The scope of our research is to generate simulated data for an IDS to accurately detect and label anomalous behavior with the goal of providing companies a cost-effective way of building their own IDS in-house. Simulating a manufacturing system with cyber attacks in Simulink can provide valuable insights into the vulnerability of the system and the potential impacts of cyber attacks. In this report, we will describe the process of simulating a model-based manufacturing system (LabFab) in Simulink and incorporating different types of cyber attacks to test an anomaly detection ML model created by Marwan Abdelatti, PhD., initially tested on the DS2OS dataset (www.kaggle.com/datasets/francoisxa/ds2ostrafficttraces).

Our work utilized the SimEvents library, a discrete event simulations package created by MathWorks, to incorporate the discrete events that are typically present in manufacturing systems along with cyber attack processes.

2 SIMULATING THE MANUFACTURING SYSTEM

To emulate the model-based manufacturing system in Simulink, we created a complex system that includes many Simulink and SimEvents blocks working in conjunction (such as Entity Generators, Servers, Output/Input Switches, Simulink Function Blocks, etc.). Our final simulation consists

of five main processes, entity ("part") ordering subsystem, part routing subsystem, data logging functions, attack scenario subsystems, and a subsystem for each operation within the physical system. All of which can be seen in **Figure 1**.

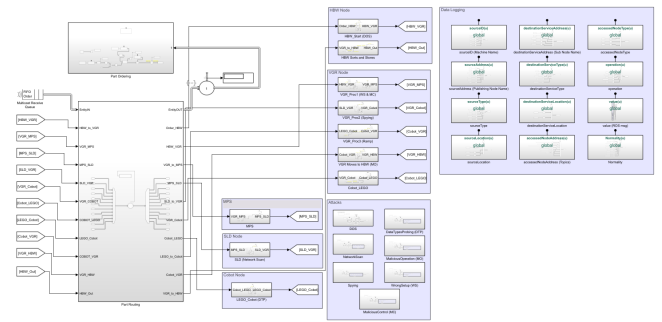


Figure 1: LabFab system simulation (full-view).

2.1 Part Ordering Process

Figure 2 shows an internal view of the part ordering process. Nine parts (entities) are created by the Entity Generator block titled "PartOrder", which are then sent through to an Entity Store block representing the High-Bay Warehouse (HBW). Once stored, a secondary Entity Generator titled "Order" sends an entity with attribute Part_ID which is randomly assigned a number (Part_ID = 1,2,...,9) which will travel into the Entity Selector block to be matched with one of the 9 primarily generated parts, simulating the process of an order being placed in the system. After which, entities pass through the Entity Gate block (which is set to allow passage at simulation start) and on to the first data-logging Entity Server block, which is set to send messages (highlighted in yellow) to the workspace via event action "entry" and "exit". Data logging process seen in **Figures 4 5**.

*✉ zdwardener@uri.edu

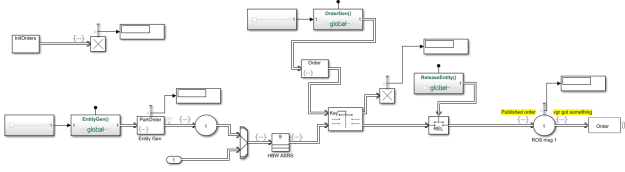


Figure 2: Internal view of the LabFab simulation part ordering process.

2.2 Part Routing Process

After leaving the Part Ordering Process, entities travel through the Part Routing process as seen in Figure 3 which updates the entity's *CurrentRoute* and *CurrentStep* attributes based on the predetermined *ServiceProcess* attribute. After every operation, entities pass through this process to determine the parts next target location. Inspiration for the setup of this portion was drawn from the MATLAB example model, 'simevents/StarRoutingExample'.

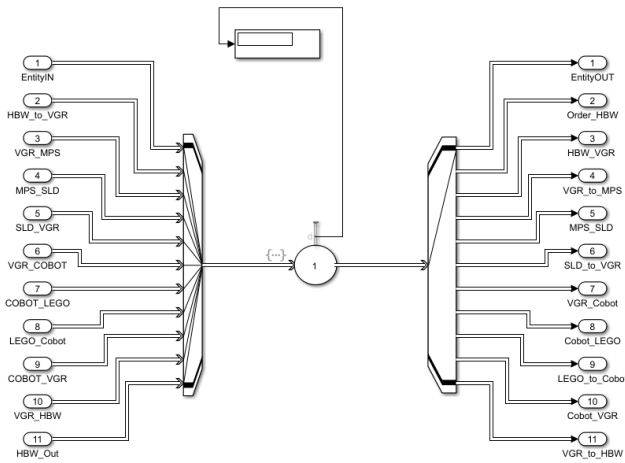


Figure 3: Internal view of LabFab simulation entity routing process.

2.3 Data Logging Process

Data logging and dataset creation is a crucial part of any simulation. As mentioned prior, data is logged via event action signals from within Entity Server blocks. Those signals are then sent to Simulink Function blocks found in the Data Logging area, as seen in Figure 4, and those signals are then translated into string constants sent to the workspace (in timeseries format) after passing through an Entity Output Switch whose path is dictated by a port number value (i.e., *Normality*(1)) as seen in Figure 5. After which, the data can be accessed through MATLAB's workspace where the data is subsequently transformed into timetables for readability, and then to tables to remove the "sec" from the Time column and lastly exported into an excel file with each column vector of data separated by sheet. The .mlx script for translating and exporting the data can be found under the name *ExportData2Excel.mlx*.

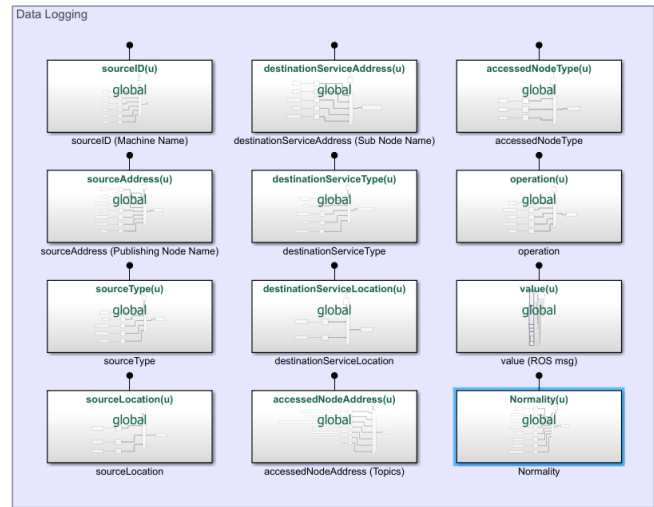


Figure 4: Data Logging area (external-view).

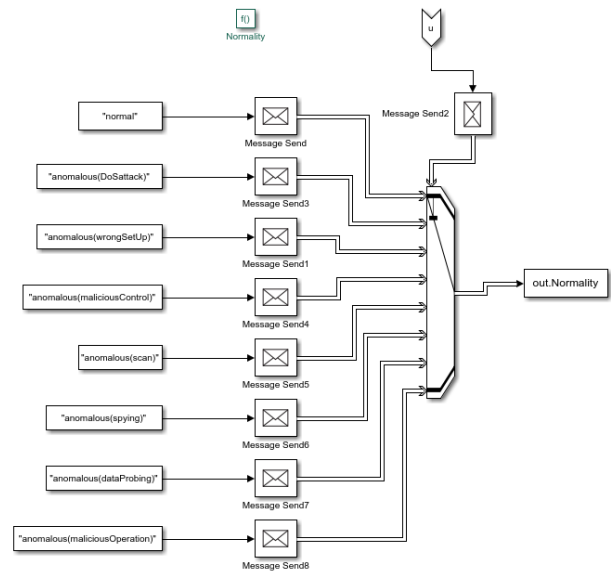


Figure 5: Inside the data logging Simulink Function blocks (Normality).

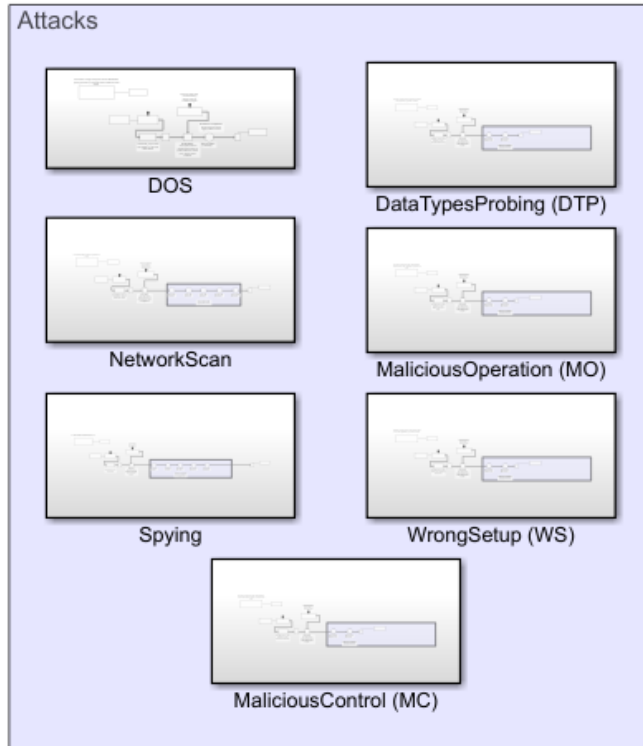


Figure 6: Simulink Subsystems containing cyber attack scenarios (external-view).

2.4 Incorporating Cyber Attacks

Once the manufacturing system was simulated, we incorporated different types of cyber attacks, as seen in Figure 6. The seven attacks we considered are those highlighted in the manuscript published by Dr. Abdelatti titled, An Intelligent Anomaly Detection System for Industrial Internet of Things. Those attacks include the following:

1. **Network Scan:** When an attacker scans a number of active services in the system. It can be seen in the DS2OS dataset as too many requests from the same source service to many random destination services. Simulating this attack on our smart manufacturing system involved the single source service /sld_node accessing multiple random destination services like /stus_update, /lego_node, /hbw_node, /listener(cobot node), and /vgr_node.

2. **Spying:** An attacker attempts to read operation values of active services. The DS2OS dataset shows this type of attack as when too many read requests from one source to many destination services of the same type, not random like the scan attack. This attack was simulated on the smart manufacturing system using the /hbw_node as a single source service sending read requests to motion sensors found in the VGR, LEGO Sorting Line, and HBW stations.

3. **Data Types Probing:** Involves sending data of different types than the one assigned to the target. Per contra to the previous two types, this attack involves only one destination service accessed by one source for only write operations to write data of type different from the original destination service data type. For the smart manufacturing system, this can be represented by the /lego_node service writing a random

string value to the motion sensors within the /listener service rather than reading pre-set, boolean, response values like Tue or False.

4. **Malicious Operation:** Involves one source service attempts to access a destination service with improper operation type, like writing to a sensor service while sensor services are created for read only. Similarly, a malicious operation in the smart factory context is shown by the single source service, /vgr_node, attempting to write a motion sensor state value to the /hbw_node rather than the normal operation of reading sensor values.

5. **Wrong Setup:** Is an access attempt from one source to a destination service in the wrong location. In the smart factory context, this happens when the /vgr_node service tries to read a motion sensor state within the LEGO Sorting Line station via /lego_color_pub topic.

6. **Malicious Control:** Occurs when one request from a source service sends a request to an irrelevant destination service. This attack type is a generalized version of the wrong set up attack where the irrelevance is not necessarily in the source /destination location. This was simulated in the smart factory as the /vgr_node tries to activate the Cobot unit at the wrong time.

7. **Denial of Service (DoS):** In this attack, the attacker overloads the system with requests, making it difficult for the system to respond. A DoS attack on the smart factory system was simulated by having the /order source service sending too many part order requests to the /hbw_node which may lock up the strafing motor, in turn halting the rest of the systems operations entirely.

2.5 Simulated Attack Process Flow

The Process Flowchart shown in Figure 9 (Page 5) describes the process used to simulate a Denial Of Service attack in Simulink within our LabFab simulation. The same general format was also used to simulate the remaining six attack patterns with slight differences in data logging to match the descriptions provided above. The lower portion of the Process Flowchart is representative of the Simulink subsystems containing the attack entities seen in Figure 8 (Page 4). The upper-portion of the Flowchart is describing a simplified version of the process seen in Figure 7 (Page4).

3 FUTURE WORK

In terms of future work on the simulation side of this project, some additional features could be added in order to bring a further depth to the model. For instance, adding in sensor readings from all light gates and motor outputs to provide the system administrator and IDS algorithm a dataset with a higher level of complexity. Another possible route for this project could be implementation using a comparable simulation software that allows for ROS communication. Since Simulink's ROS library would not allow us to publish string messages in a reliable way, we were forced to rely on String Constant blocks containing the messages that are meant to be transmitted via ROS connection.

4 CONCLUSION

The simulations performed using the Simulink and SimEvents libraries demonstrate the importance of addressing the security of manufacturing systems. The results show that these systems are vulnerable to common types of cyber attacks and that the consequences can be severe. To ensure the security and reliability of these systems, it is essential to invest in security measures and continuously evaluate and improve them. Overall, the use of the Simulink and SimEvents libraries for simulating cyber attacks on manufacturing systems provides valuable insights into the vulnerabilities and risks associated with these systems. This information can be used to inform the development of security measures and to ensure the continued protection of these systems from cyber attacks.

DATA AVAILABILITY

Access Simulink model and data processing .mlx script via [GitHub/Zdewardener/LabFab_DS2OS_Simulink](https://github.com/Zdewardener/LabFab_DS2OS_Simulink); see: https://github.com/Zdewardener/LabFab_DS2OS_Simulink
Make sure to use MATLAB R2022a

COPYRIGHT NOTICE

© The Author(s) 2023. This article is distributed under the terms of the [Creative Commons Attribution 4.0 International License](#), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

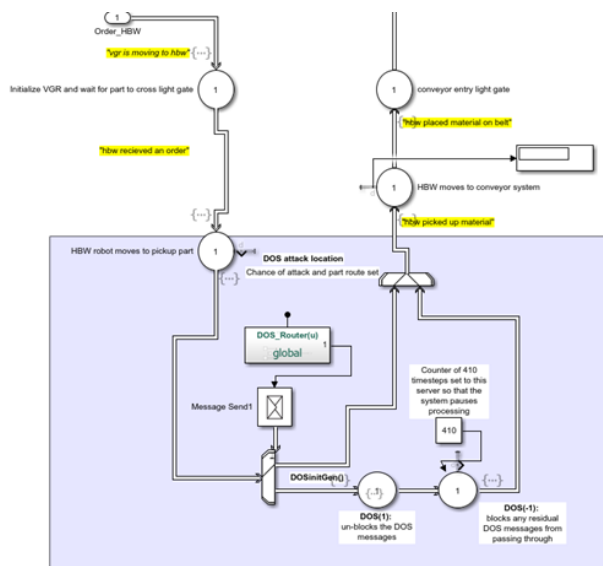


Figure 7: DOS attack routing found in HBW_Start subsystem (Initializing DOS attack).

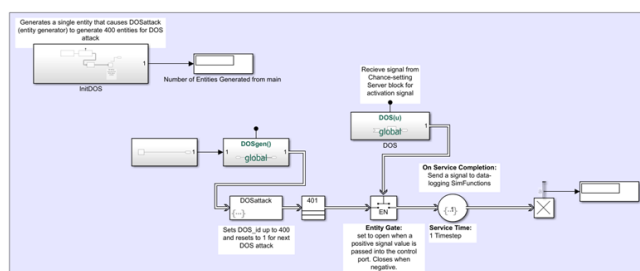


Figure 8: DOS attack Subsystem structure (Generating DOS messages).

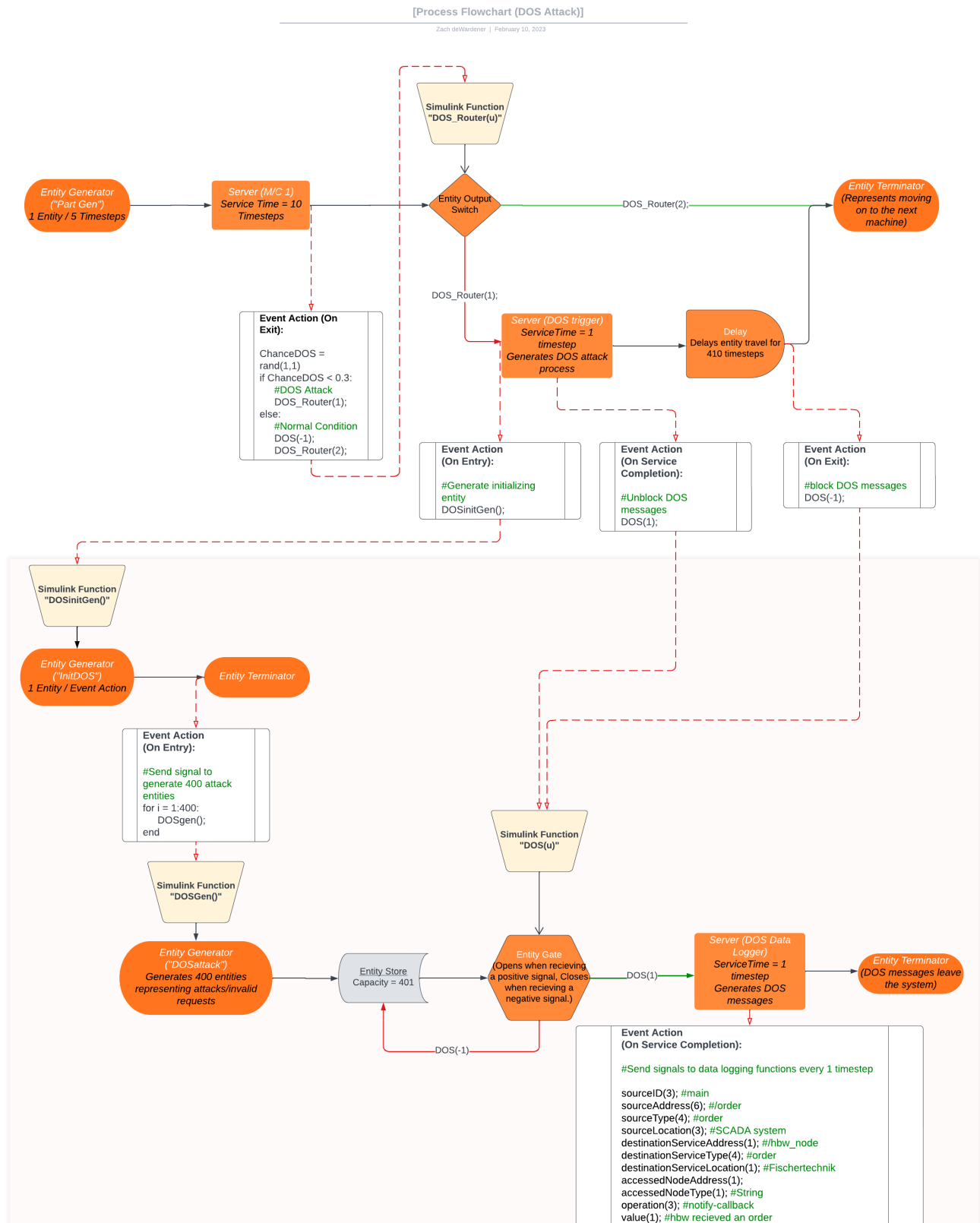


Figure 9: Process Flowchart for a DOS attack within the simulated model.