Cyberattacks and Vociferous Implications on SECS/GEM Communications in Industry 4.0 Ecosystem

Shams A. Laghari[1], Selvakumar Manickam[2]*, Shankar Karuppayah[3], Ayman Al-Ani[4], Shafiq Ul Rehman[5]
National Advanced IPv6 Centre (NAv6), Universiti Sains Malaysia (USM), Pulau Pinang, Malaysia[1,2,3,5]
Department of Biomedical Engineering, Faculty of Engineering, University of Malaya, Kuala Lumpur, Malaysia[4]

***Abstract*—Information and communications technology (ICT) is prevalent in almost every field of industrial production and manufacturing processes at present. A typical industry network consists of sensors, actuators, devices, and services to connect, track, and manage production processes to increase performance and boost productivity. The SEMI Equipment Communications Standard/Generic Equipment Model (SECS/GEM) is SEMI's Machine-to-Machine (M2M) protocol for equipment-to-host data communications. It is the most popular and profoundly used M2M communication protocol operating in the manufacturing industry. With Industry 4.0 as a guiding factor, connectivity to business networks is required for accessing real-time data whenever and wherever needed. This openness of connectivity raises security concerns as SECS/GEM protocol offers no security, which endangers exposing the manufacturing industries' business secrets and production processes. This paper discusses the key processes involved in SECS/GEM communications and how potential attackers can manipulate these processes to obtain illegal or unauthorized access. The experiments' results indicate that the SECS/GEM processes are entirely vulnerable to numerous attacks, including DoS attack, Replay attack, and False-Data-Injection-Attack. Thus, the future direction involves developing a prevention mechanism that aims at securing the SECS/GEM processes in the industrial network. This study's findings are useful as preliminary guidance for the infrastructure owners to plan for appropriate security measures to protect the industrial network.**

Keywords—SECS/GEM; cybersecurity; industry-4.0; machine-to-machine communication; industrial internet of things (IIoT)

I. Introduction

Over the past few decades, technological advancements have progressed so rapidly that we have reached the fourth industrial revolution called Industry 4.0, commonly termed as the Industrial Internet of Things (IIoT) or smart manufacturing [1]. Cyber-physical structures have been accoladed to map the physical world into a virtual one with the advent of the technological revolution. Cybersecurity, robotics, cloud computing, 5G networks, big-data analysis, machine learning, Internet of Things (IoT), and additive manufacturing are the prime developments of this modern-day technology, which bring forth groundbreaking changes in the life of individuals, society, industry as well as economy [2]. Industry 4.0 is vying for enhanced connectivity, machine learning, real-time data collection, the interaction between machine-to-machine on novice mechanisms, automation and advanced robotics, increasing industrial productivity, and modernizing the production process. Even though businesses, companies, and organizations are entirely different in scale and scope, they all face the same problem, i.e., the need for connectivity and real-time insights through manufacturing processes, products, material movement, and resource utilization for timely decision making [3].

Historically, manufacturers have been more concerned with protecting their Operational Technology (OT) environment, often almost entirely neglecting Information Technology (IT) security. The perpetrators are fully informed of this negligence and realize how easy it is to infiltrate and hack industrial networks. In accordance with the awareness that important, confidential information such as product types, product recipes, material details, system configurations, comprehensive equipment logs, equipment's communication patterns, etc., is held valuable by industries and thereby opens up an increasingly attractive prospect for threat-actors. Hackers and threat-actors have realized that the supply chain is a large and complex process with ample vulnerabilities, and therefore it is the perfect environment for attempting an attack to infect numerous suppliers and organizations at a massive scale.

Recent incidents of security breaches and cybercrimes have reached the epidemic stage in the manufacturing industry, rendering the manufacturing industry the most vulnerable and targeted sector by attackers [4]. According to EEF, a recent survey reveals that 48 percent of manufacturers have been subject to a cybersecurity incident at any point, and half of those firms have suffered financial losses or market disruption. Another study carried out by Cybersecurity Ventures has identified that cybercrime will cost corporations worldwide $10.5 trillion a year by 2025, up from $3 trillion in 2015 [5], while manufacturing has been catching up quickly in recent years, as can be seen in the Verizon Data Breach Investigation Report 2019, which detailed 352 incidents, 87 among manufacturers. The Taiwan Semiconductor Manufacturing Company (TSMC) computer virus attack was Taiwan's biggest information security infringement in its history. It fully exposed the information security weaknesses at production plants as the manufacturing industry embraces the fourth industrial revolution, or industry 4.0, with increasing automation and data exchange [6].

This work embodies an overview of SECS/GEM standard and uncovers lacunas in SECS/GEM, which eases launching of

*Corresponding Author

attacks on factory equipment with the help of attacks like False-Data-Injection attack (FDIA), Replay attack, Denial of Service (DoS) attack. This paper, therefore, includes a brief explanation of the relevant aspects of the SECS/GEM standard in order to fully comprehend the mechanics of these cyberattacks on industrial networks.

This paper is arranged into the following sections: A brief literature review of security features of popular industrial M2M communication protocols is covered in Section 2. SECS/GEM's overview and connection establishment processes are explained in Section 3. Attacks on SECS/GEM processes are discussed in Section 4. Scenarios, attacks, and experimental results are discussed in Section 5. The conclusion and future outlook are provided in Section 6.

II. Literature Review

Within the landscape of Industry 4.0, cybersecurity plays a leading role in preventing the loss of companies' competitiveness. However, when seen in the perspective of the industrial automation industry, cybersecurity has concentrated more on protecting corporate and operational perimeters, i.e., restricting unauthorized access to the industrial network. There are numerous messaging communication protocols for IIoT and industrial communications such as SECS/GEM, OPC UA, CoAP, DDS, MQTT, and many more. Industry 4.0 has brought these protocols under the spotlight once again. Most of these protocols have limited security features. A brief discussion on security features offered in the above-mentioned M2M protocols is provided in the following sub-sections.

A. *Message Queuing Telemetry Transport*

Message Queuing Telemetry Transport (MQTT) is an open-source messaging protocol developed by IBM for resource-constrained devices (i.e., devices having limited computational power, memory, storage, and battery backup). It adheres to the publish/subscribe interface paradigm and performs well in networks where communication requirements are less than optimal such as high latency with low bandwidth. It keeps bandwidth requirements to an absolute minimum, works ridiculously well with unstable networks, and is best suited for machine-to-machine (M2M) communications. MQTT is undoubtedly regarded as a de facto IoT normative protocol, and research on the offensive and defensive solutions to MQTT has attracted substantial interest from academia and industry for analysis on security issues [7].

The condition under which the MQTT protocol operates by default [18] can easily be linked to nearly all security concerns that occur. Since MQTT is meant to be lightweight, it does not encrypt the header or payload but rather exchanges data as plaintext, which is obviously a security problem. Thus, encryption must then be used as a separate function, such as by Transport Layer Security (TLS), which increases the computational overhead on the resource-constrained devices. Most MQTT brokers provide authentication support through the use of the Link control message. A few security solutions have been proposed to address MQTT's security concerns [8][9][10].

B. *Open Platform Communications – Unified Architecture*

Open Platform Communications – Unified Architecture (OPC-UA) is an M2M communication protocol developed by OPC Foundation for Industrial Automation [11]. OPC UA is built to ensure security and thus offers a wide variety of core security features, including authentication, integrity, confidentiality, and authorization. It provides different protection modes, such as no security, integrity only, integrity and confidentiality, as well as security policies predefining encryption and signature cryptographic algorithms for the establishment of secure communication. Depending on the message protection mode, the client chooses a security protocol to secure the messages exchanged during the handshake. Notably, one of the seven security policies available offers little security, and two have been discarded because of cryptographic primitives that are now vulnerable. Overall, OPC UA offers industrial automation with strong security features [12].

C. *Constrained Application Protocol*

Constrained Application Protocol (CoAP), as specified in RFC 7252[13], is an Internet Application Protocol specifically designed for resource-constrained devices and constrained networks [14]. It functions HTTP-like and follows REST-Architecture, and runs over User Datagram Protocol (UDP). To secure communication, CoAP does not use TLS since it runs over UDP, which is an insecure transport protocol, so DTLS is used by CoAP instead of using TLS as a security solution.

CoAPS is a security-enabled variant of the CoAP protocol, which is based on a TLS protocol with modifications needed to operate over unreliable UDP connections. Some of the TLS enhancements include features whereby CoAPS retains the connection in the event of missed or out-of-order packets and prevents link termination (i.e., teardown) as it is done in CoAP. There is a probability of retransmitting handshake messages as an example. With the exchanging of client and server 'hello' messages, the handshaking mechanism is quite similar to the one in TLS, but with the additional possibility for a server to send a verification query to ensure that the client sends its 'hello' message from the authentic source address. This added functionality helps in preventing DoS attacks.

D. *Data Distribution Service*

Data Distribution Service (DDS) [15] is an M2M protocol that follows the publish/subscribe pattern, and it is specifically designed to allow efficient, high-performance, interoperable data exchange for real-time systems. DDS can run on both TCP and UDP, so a security mechanism provides two distinct options for DDS. The security combination may be either TCP with TLS or UDP with Datagram Transport Layer Security (DTLS), depending on the underlined transport protocol. Both TLS and DTLS are not designed for constrained devices; thus, they incur too much computational overhead and are not suitable for a constrained environment. In order to solve the issue, OMG's proposed DDS security specifications describe a robust security architecture that is considered lightweight and acceptable for the IoT device.

Unlike all protocols mentioned above, SECS/GEM does not provide any security features. High-Speed SECS Message Services (HSMS) is a SECS/GEM's transport communication protocol and runs over TCP. It lacks security. Henceforth, data is not encrypted by any encryption algorithm. Moreover, credentials and certificates are not required for connection. Furthermore, validation of the connecting party is not carried out. However, obfuscation is applied to the data through the data packaging binary encoding process, which makes it incomprehensible for humans.

To our knowledge, no one has previously assessed SECS/GEM in terms of security features. In this paper, we have extensively analyzed the mechanism of SECS/GEM and have identified weak points that may be problematic if not patched in time would undoubtedly lead to catastrophic consequences, including financial and business losses.

III. SECS/GEM Processes and Connection States

SEMI (Semiconductor Equipment and Material International) is an association enjoying membership of more than two thousand companies worldwide. It deals with materials, services, and equipment required by the manufacturing industries. The SECS/GEM standard has been widely adopted in semiconductors, surface mount technology, electronics assembly devices, photovoltaic, and solar cell industries. SEMI has published several different specifications that coordinate communication between the host and the factory machinery, including E4, E5, E30, and E37. SEMI communication standards are collectively known as SECS/GEM. The SECS/GEM protocol is an industry-standard that is widely in operation across several industries worldwide [3][16][17]. It acts as a backbone of the semiconductor industry and is heavily used in the world's leading enterprises, including Intel, Samsung, TSMC, IBM, Qualcomm, Broadcom, UMC, SK Hynix, Micron, TXN, Toshiba, NXP, proving as a de facto communication protocol and control system since years[18].

High-Speed SECS-II Messaging Service (HSMS) serves as a transport protocol for SECS/GEM communications in industrial networks [19][20][21]. The HSMS protocol is derived from Transmission Control Protocol / Internet Protocol (TCP/IP). It utilizes virtually the same methods of forming a link as specified in RFC-793 but with minor essential modifications [22]. In accordance with the guidelines prescribed in RFC 793, either side of the communication link can initiate a request to create the connection. However, the HSMS protocol limits the standard practice of the formation of connections as defined in the RFC; and distinguishes two distinct types of connection modes, namely active and passive. The devices configured in passive mode act as a server, open a port, and listen for incoming connections, while the responsibility of initiating a connection request is destined to the devices configured in active mode. Upon a successful connection has been formed between the two communicating entities (i.e., host and equipment), HSMS protocol passes binary encoded SECS-II messages. The HSMS protocol keeps the connection alive, and data moves back and forth until either or both entities are purposely rendered offline (i.e., firmware or software updates, add/remove machines in the production line, maintenance, etc.). The HSMS message format is shown in Fig. 1, and a brief overview of the HSMS header fields is discussed in Table I. HSMS messages are packed in binary format and are transported as byte streams, where the first four bytes of the message are used to calculate the overall message length. The minimum and maximum size of a message carried out over an HSMS protocol is 10 bytes and 4.3 gigabytes, respectively.

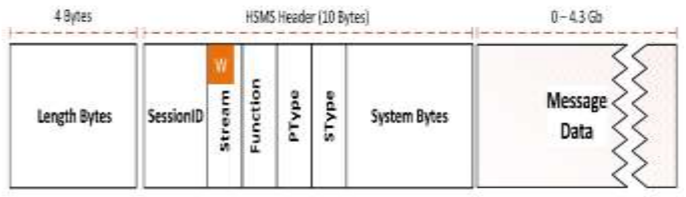


Fig. 1. HSMS Message Structure.

TABLE I. Brief Description of HSMS Header Fields Along with Possible Values

Bytes	Field	Description
0-1	Session ID	- Provides an association by reference between control messages and subsequent data messages
2	W-Bit & Stream	- Contains 0 or status code for control messages - When SType=0, it contains W-bit and Stream number for data messages
3	Function	- Contains 0 for control messages - When PType = 0, it contains SECS-II function
4	PType	- Contains 0 for SECS-II messages
5	SType	- 0 for data messages - 1-9 (8 is unused) for control messages
6-9	System Bytes	- Contains a unique value for each transaction - Same value for both Primary and its associated secondary (response) message

A. SECS/GEM Message Types

SECS/GEM messages are broadly divided into two categories, (1) control messages and (2) data messages. Control messages are used by the HSMS protocol for connection and link management. The SType header field of HSMS protocol may have values 1,3,5 and 9 to perform operations select.req, deselect.req, linktest.req, and separate.req, respectively. Each request message is acknowledged with a succeeding even number value for the paired control message except the separate.req, which does not require acknowledgment. In addition to the aforementioned control messages, there is an additional control message named reject.req with SType=7. This control message is used in response to any valid HSMS message received which is not supported by the receiver of the message or which is not valid at the time when that message was received [23] (for example, an HSMS message whose SystemBytes value is invalid or it does not match with any open transaction).

B. SECS/GEM Connections States

In order to establish a connection with equipment, it is important to understand the SECS/GEM connection state. There are two states at the fundamental level, NOT-CONNECTED and CONNECTED. As the name indicates, in the NOT-CONNECTED state, either an entity is listening on a

TCP port and waiting for incoming connection establishment requests, or all previously established connections have been terminated. Either way, an entity in a NOT-CONNECTED state does not engage in any communication. Upon a successful TCP connection establishment with the host, equipment enters the CONNECTED state. Two sub-states named NOT-SELECTED and SELECTED are in the CONNECTED state. The entities await the request for the HSMS link establishment upon entering the CONNECTED state.

Once the HSMS control message for the connection establishment is received, the entity enters the SELECTED state. Only when entities have reached the SELECTED state are the actual SECS-II messages and control messages exchanged between the two communicating entities. Fig. 2 depicts the HSMS State Model.

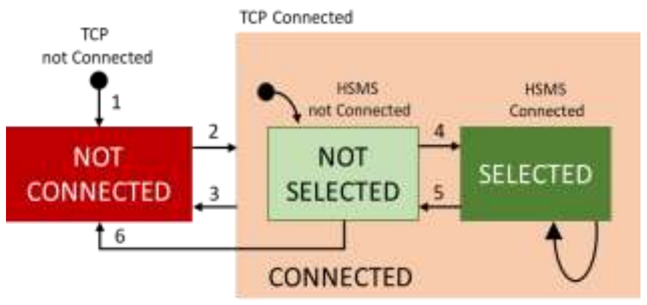


Fig. 2. HSMS Connection States.

C. HSMS Communication Processes

SECS/GEM messages exchanged between the host and equipment are highly important in order to control and monitor the equipment. In a normal communication mode, either side of the connection can request data. Upon receiving a request message, the target entity would reply with generated data for the specific request. SECS/GEM communication is carried out on HSMS protocol, which has a fixed header format and structure. If the SType field of the HSMS header in any message is 0, it means the message is a data message; otherwise, it will be considered a control message.

1) Connection establishment process: Fig. 3 represents the different SECS/GEM communications processes. The entity configured in active mode (i.e., usually host) must send a TCP request to the equipment entity that is typically configured in passive mode in order to initiate communication. Upon receiving a response from the equipment, the host will send an acknowledgment, and a TCP connection will be formed between the two communicating entities (i.e., Fig. 3(a)). The equipment will enter into the CONNECTED state at this point and wait for the establishment of an HSMS connection. At this stage, the host will request for the HSMS link establishment, and the equipment will respond to that message and change its state from NOT-SELECTED to SELECTED. Now, the link is complete and ready for the exchange of SECS-II messages (Fig. 3(e)). The process of creating TCP and HSMS connections is illustrated in Fig. 3(a) and Fig. 3(b).

2) Connection management process: The connections in industrial networks are kept alive for several days or even weeks. There are possibilities that there is no communication carried out for a certain period of time between the two entities; therefore, it is required to test the connectivity beforehand. Fig. 3(d) illustrates a scenario when such an inactive period is detected when there is no communication between the two entities. The control messages linktest.req and linktest.res are exchanged periodically based on timer T3 value in order to keep the connection alive.

3) Connection termination (tear-down) process: Two ways to teardown a link are specified by SECS/GEM specifications, i.e., by sending deselect.req or separate.req control messages. The difference between these two messages is that without waiting for an acknowledgment, separate.req abruptly disconnects the connection. Fig. 3(c) and Fig. 3(f) demonstrate the teardown phase of the connection.

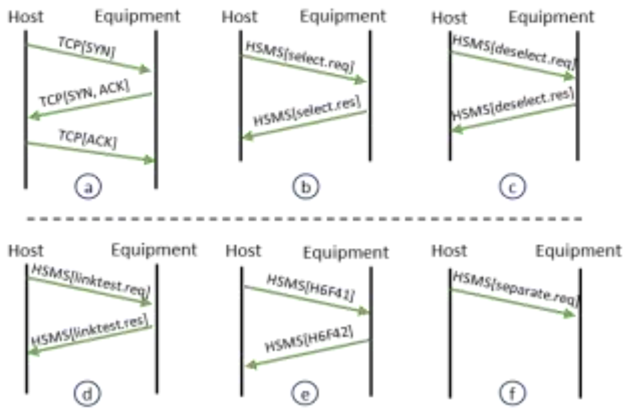


Fig. 3. SECS/GEM's Connection Establishment, Control and Data Message Processes.

IV. Cyberattacks on SECS/GEM

Numerous cyber-attacks, such as DoS attack, Replay attack, spoofing attack, side-channel attack, covert-channel attack, False-Data-Injection-Attack, etc., can be launched on industrial networks[24] where SECS/GEM is used for M2M communications. However, this study focuses on DoS attack, replay attack, and FDIA. Fig. 4 depicts a typical industrial machines network layout and production line on the shop-floor.

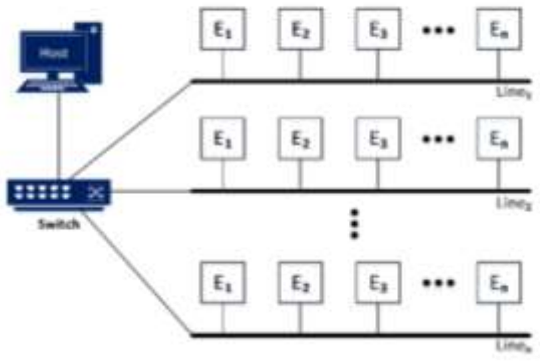


Fig. 4. A Simplified and Generalized OT Industrial Shop-Floor Network.

The first step that should be taken in order to carry out a successful attack on any network is to analyze network activity and communication patterns. A huge amount of data is usually generated by SECS/GEM-enabled machines, and this data is exchanged in plaintext and without any authentication. Fig. 5 shows the pattern of 20 hours of communication between a host and equipment that occurred in a production environment (for protection and privacy purposes, the organization's name and the machine's specifics are not discussed here). SECS/GEM messages are paired as primary and secondary messages, where request messages, and secondary messages are associated reply messages. Each primary message is usually responded with an associated secondary message (i.e., S6F11/S6F12 represents the pairing of Primary/Secondary messages), as shown in Fig. 5. It is important to note that a primary message can be sent at any time by both sides of the connection, and therefore it is essential to keep a record of the most recent SystemBytes value for a successful attack. The information presented in Fig. 5 also reveals that the preparators might not have yet carried out any successful attack on the industrial network; the intruders may still have the ability to effectively intercept data and steal sensitive product information by merely observing data shared between the host and the equipment.

A. Denial-of-Service (DoS) Attack

In industrial communication networks, DoS is characterized as a particular cyberattack in which attackers try to ensure that access to the intended users is either temporarily or permanently inaccessible for the desired service or resource. There are numerous ways to achieve a DoS attack in SECS/GEM communications. For example, the perpetrators may capture and forge a bogus message of a legitimate host/equipment and send a deselect.req message to terminate the communication link. Upon using the port stealing mechanism, the attack-actors may perform a man-in-the-middle attack and then initiate the DoS attack, in which case the host will never know the connection is terminated [25].

We have exploited the HSMS's control messages to launch a DoS attack. We used the control message separate.req in this study to execute a successful DoS attack on HSMS communications. As discussed in the previous section, an HSMS message with SType=9 is used to terminate HSMS communications immediately. With the exception of the SType value, separate.req control message is identical to the Deselect.req message. Its purpose is to end HSMS communications immediately and without exception or notifying the host. No response or acknowledge message is required.

Fig. 6 illustrates in detail how a DoS attack on SECS/GEM communications is carried out. As it happens in most cases, the preparator scans the network and passively tries to determine the nodes that are listening on port 5000, which is usually used for HSMS communications. The attacker captures and monitors the network traffic and captures the most recent reply message sent from the target machine. It is important to note that the chances of a successful attack carried out on response messages are higher than request messages because it usually takes a while to request another set of data in quick succession.

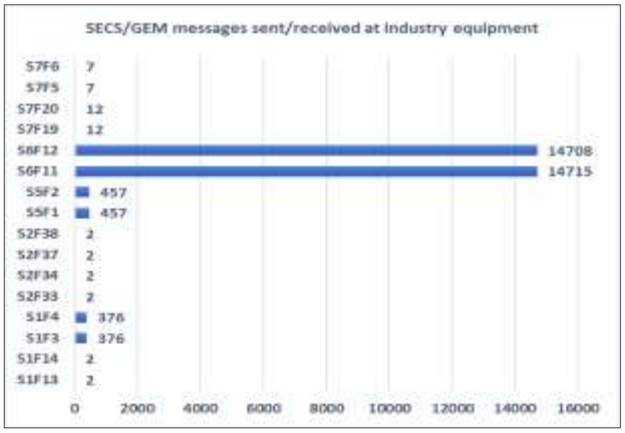


Fig. 5. Typical SECS/GEM Interaction Pattern as seen on a Factory Machine.

The DoS attack in Fig. 6 follows the same analogy, and the attacker waits for a linktest.req and linktest.res pair of messages exchanged between the host and equipment. At this stage, the attacker would intercept any control message (ex. linktest.res), modify header field SType=9 (which represents separate.req message) and increment SystemBytes value by 1 to launch DoS attack. The attacker may send these messages to both host and the equipment, which will teardown connections between the host and the equipment abruptly. The received message is considered legitimate because there is no mechanism available in SECS/GEM to validate the authenticity of the received separate.req message. The attacker may periodically or at random time intervals launch a replay-attack to converge it as a full-fledge DoS attack. Both the host and the equipment will remain unaware of the attack. The attacker then can send a connection establishment request on port 5000 to equipment to hold the connection. During this time period, the legitimate host would remain disconnect and unable to reestablish the communication link as SECS/GEM is a point-to-point communication protocol and only allows one active connection at a time.

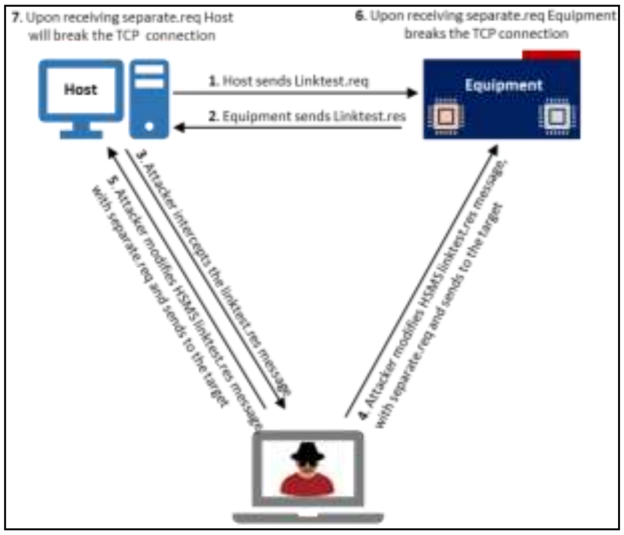
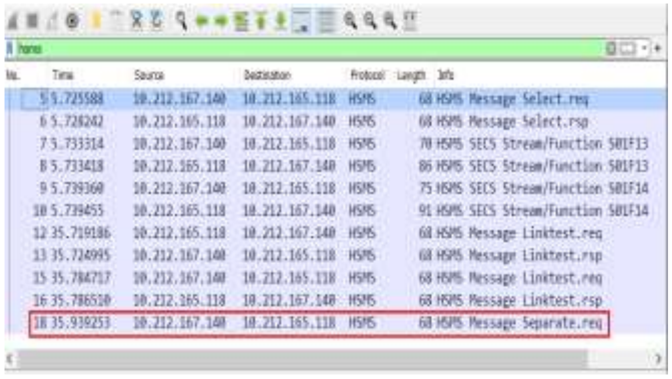

7. Upon receiving separate.req Host will break the TCP connection
6. Upon receiving separate.req Equipment breaks the TCP connection
Host
Equipment
1. Host sends Linktest.req
2. Equipment sends Linktest.res
3. Attacker intercepts the linktest.res message
5. Attacker modifies HSMS linktest.res message with separate.req and sends to the target
4. Attacker modifies HSMS linktest.res message with separate.req and sends to the target

Fig. 6. Scenario of Capturing, Intercepting & Attacking SECS/GEM Communications.

Alternatively, by using TCP's RST flag to tear down the TCP connection established between the two communicating entities (i.e., host and the equipment), we were able to launch a DoS attack on SECS/GEM communications successfully. The equipment would change its status back to a NOT-CONNECTED state when it detects the broken TCP connection, and it would start listening for new incoming connection requests on the specified port address (i.e., usually 5000). At this stage, the attacker would establish a connection on the open port and hold a connection for an unspecified time. During this time frame, any attempt of connection establishment by the legitimate host will be ignored by the equipment. In our experiments, we witnessed that a simple application such as NC (a.k.a., Netcat) running on an attacker's machine suffices the requirement to establish a TCP connection with the equipment. This is because HSMS communication starts after successfully establishing a TCP connection (i.e., three-way handshake). We also found that the equipment continued to send requests for the HSMS link establishment to the attacker machine that simply did not respond to any of the equipment's messages. However, during this attack, the attacker held the TCP session, and the equipment will not accept any other connection until the current session expires. Hence, the attacker is able to hold the connection and disrupt the communication between a host and equipment as long as it is desired.

The DoS attack can also be launched on SECS/GEM communications using data messages. This is true because the HSMS standard limits HSMS connection to process messages as 1Hz, that is, one message per second. The attacker can send any data messages during this time to throttle that equipment's processing capabilities. Fig. 7 indicates the effective injection of the Separate.req message occurred in the HSMS connection establishment between the host and equipment, and subsequently, the equipment terminated the connection.



```
No.  Time        Source          Destination     Protocol Length Info
   5 5.725588    10.212.167.140  10.212.165.118  HSMS     68 HSMS Message Select.req
   6 5.728242    10.212.165.118  10.212.167.140  HSMS     68 HSMS Message Select.rsp
   7 5.733314    10.212.167.140  10.212.165.118  HSMS     70 HSMS SECS Stream/Function S01F13
   8 5.733418    10.212.165.118  10.212.167.140  HSMS     86 HSMS SECS Stream/Function S01F13
   9 5.739360    10.212.167.140  10.212.165.118  HSMS     75 HSMS SECS Stream/Function S01F14
  10 5.739455    10.212.165.118  10.212.167.140  HSMS     91 HSMS SECS Stream/Function S01F14
  12 35.719186   10.212.165.118  10.212.167.140  HSMS     68 HSMS Message Linktest.req
  13 35.724995   10.212.167.140  10.212.165.118  HSMS     68 HSMS Message Linktest.rsp
  15 35.784717   10.212.167.140  10.212.165.118  HSMS     68 HSMS Message Linktest.req
  16 35.786510   10.212.165.118  10.212.167.140  HSMS     68 HSMS Message Linktest.rsp
  18 35.939253   10.212.167.140  10.212.165.118  HSMS     68 HSMS Message Separate.req
```

Fig. 7. The Successful Injection of Separate.Req Attack Message.

Detecting a DoS attack is difficult to traceback as it is never initiated by a rogue device; often, they are carried out by the footprints of a legitimate device from within the network.

B. Replay Attack

SECS/GEM messages exchanged between the host and equipment are highly important in order to control and monitor the equipment. In a normal communication mode, either side of the connection can request data. Upon receiving a request message, the target entity would reply with generated data for the specific request. SECS/GEM communication is carried out on HSMS protocol, which has a fixed header format and structure. If the SType field of the HSMS header in any message is 0, it means the message is a data message; otherwise, it will be considered a control message. Irrespective of the message type, an attacker may sniff the message and replay it with a changed payload to inflict damage or interrupt the overall production. HSMS has a sequence number field in its header called SystemBytes, which is monotonically increased for each transaction. A pair of request and response messages from host and equipment or vice-versa is called a transaction, and each transaction is identified with a unique sequence number (header field SystemBytes). However, each successive transaction will have a successive number of the previous transaction. This predictability enables attackers to launch a replay attack by just modifying SystemBytes contents. The replay attack would be devesting for the overall communication and may bring down overall communication, or it may adversely impact the equipment's behavior.

Upon receiving a connection establishment request, the attacker may intercept any control message (ex. select.res), modify header field SType=9 and increment SystemBytes header fields by 1 to launch a replay-attack, which will teardown connection between host and the equipment abruptly. The attacker may periodically or at random time intervals launch a replay-attack to converge it as a full-fledge DoS attack. Both the host and the equipment will remain unaware of the attack. At this stage, it will be very difficult to trace out such an attack on the equipment side; however, the careful investigation at the host will reveal that no such teardown command was initiated from the host, which will ultimately unfold the truth that the equipment was under DoS attack.

C. False Data Injection Attack

Originally in the smart grid domain, the definition of FDIA[4] was first introduced. Although the term sounds normal, it indicates that an intruder exploits sensor readings in such a way that undetected errors are inserted into state variables and calculated results. FDIA is not limited to the manufacturing industry only; cyber-criminals are involved in leveraging related attacks in other application areas such as healthcare, financial institutes, defense. Thus, FDIA has been one of the highest priority problems to contend with within today's highly perilous cyber world of dynamic adaptive systems.

Cyber-criminal can launch FDIA and inject falsified data into blind-trusted communication occurring between two SECS/GEM entities. The receiving entity cannot differentiate between a legitimate and a bogus message, and it will process it nevertheless. The impact and intensity of the falsified data injected depend on the purpose of the attack; however, it can be imagined that under such attack and ill-intention data injected into the production system might disrupt the overall processing or at least will undermine the quality of the product.

V. Experiments and Results

In this section, the various attacks are launched on different SECS/GEM communication processes carried out between a host and equipment. The details of hardware specifications are shown in Table II, and the Testbed attack scenario is illustrated in Fig. 8. The SECS/GEM communication has been started between a host and equipment E2 in production $Line_n$.

The experiments were conducted to observe the SECS/GEM behavior during various attacks. The Python implementation of SECS/GEM host and equipment was used to emulate SECS/GEM entities' behavior and functionality. The necessary code level changes have been adapted to mimic the attack behaviors in the host-side of the emulator. In order to successfully carry out cyber-attacks on SECS/GEM communications, it is important to eavesdrop and capture the packets while in transit and then modify them to launch attacks such as DoS attack, replay attack, and FDIA. For this purpose, the Python-based Scapy tool is used to capture, modify, and transmit forged packets to the target entity [26]. The attacks were repeated 30 times for each attack scenario, and results were measured for each process. The equation (1) below is implemented to measure the ability of the SECS/GEM processes in preventing the attacks carried on SECS/GEM communications [27]:

$$ATTACK_{SR} = 1 – F / N \quad (1)$$

where, $ATTACK_{SR}$ represents the specific attack's success rate, N represents the number of times forged messages are injected, and F signifies the specific attack type failed a number of times. According to the $ATTACK_{SR}$ definition, it was found that if $ATTACK_{SR}$ is 1, this means that the attack is successful. Nevertheless, if $ATTACK_{SR}$ is 0, this indicates that the specific attack on the SECS/GEM process can be susceptible to DoS. Table III illustrates the experimental results of each experiment conducted on SECS/GEM's processes, such as control messages and data messages.

Based on Table III, the results revealed that the SECS/GEM communication is completely vulnerable to cyber-attacks. Following are our findings:

- The key issue with SECS/GEM protocol is that it does not provide any security mechanism to guard against cyber-attacks; thus, all data and control messages can be intercepted and tampered with by attackers, and it is up to the mercy of attackers to determine whether to initiate an effective attack by any manner they want.
- SECS/GEM messages are packed in the binary-encoded format to achieve high data density with little overhead. However, these messages are still readable and can easily be intercepted and converted into plaintext. Binary encoding thus does not offer any safeguard against attacks or hinders attackers from extracting meaningful information.
- The communication can be initiated by any host/equipment or rouge machine anytime without any validity or verification of the legitimacy. There is no authentication mechanism to check the legitimacy of the originator of the message.
- SECS/GEM offers a method to assess the freshness of the message by using SystemBytes; however, the SystemBytes value is monotonically incremented, and therefore it is simpler to estimate the next potential value from the current value. As a result, the receiver of the message cannot discern whether a message has been altered just by analyzing the contents of SystemBytes.
- SECS/GEM messages are paired as primary and secondary messages. Primary messages are request messages, whereas secondary messages are reply messages. SECS/GEM messages are binary encoded and exchanged in plaintext; however, they are still readable, and there is no way to detect if the messages are modified while in transit.
- Messages are not encrypted; therefore, the entire communication is susceptible, and the attacker can easily profile communication details and steal valuable information and business secrets.

TABLE II. TESTBED DEVICES AND THEIR SPECIFICATIONS

Device Role	Specifications	OS
Host	**CPU:** Intel(R) Ci7-9750H @ 2.60GHz x 8 **RAM:** 24 GB	Win10
Equipment	**CPU:** Intel® Ci7-3770M @ 3.4GHz x 8 **RAM:** 8 GB	Win10
Attacker PC	**CPU:** Intel(R) Ci3-330M @ 2.13GHz x 2 **RAM: 8 GB**	Ubuntu 2020.3
Switch	Cisco Catalyst 2960 Fast Ethernet	-

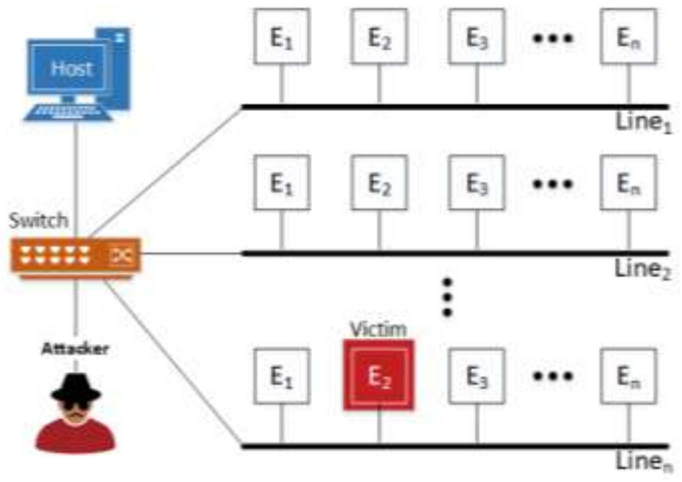


Fig. 8. Testbed Environment: An Attack Scenario.

TABLE III. EXPERIMENTAL RESULTS

Attack Type	Message Type	Experiment Count[N]	Attack Failure[F]	$ATTACK_{SR}$
DoS	Control TCP[RST]	20 20	0 0	1 1
Replay	Control Data	20 20	0 0	1 1
FDIA	Control Data	20 20	0 0	1 1

VI. Conclusion and Future Outlook

SECS/GEM has been seen as the cornerstone to semiconductor industries, and it has been in profound use since its inception. It aims to provide machine-to-machine communication, and it provides instantaneous (i.e., real-time) insights of factory equipment for effective decision making and performance boost up. SECS/GEM messages are binary encoded and exchanged plaintext without any authentication and encryption. As SECS/GEM does not offer any security mechanism, all these messages are subject to cyberattacks. The experimental results revealed that the SECS/GEM processes are vulnerable to cyberattacks. Therefore, there is a need for a comprehensive security framework that prevents these cyberattacks and offers authentication, confidentiality, and integrity for secure and reliable communication. The authentication mechanism will ensure that messages are exchanged between legitimate SECS/GEM entities. The integrity mechanism will ascertain that the contents within messages are not altered during transit, and the confidentiality mechanism will ensure that the messages are not readable while in transit. Our future direction is focused on all these issues and will provide a complete security framework that will protect SECS/GEM communications from cyberattacks discussed in this paper.

References

[1] E. Oztemel and S. Gursev, Literature review of Industry 4.0 and related technologies, *Journal of Intelligent Manufacturing*, Vol. 31, No. 1, pp. 127–182, January, 2020.

[2] G. *Culot*, F. Fattori, M. Podrecca, and M. Sartor, Addressing Industry 4.0 Cybersecurity Challenges, *IEEE Engineering Management Review*, Vol. 47, No. 3, pp. 79–86, September, 2019.

[3] S. A. Laghari, S. Manickam, and S. Karuppayah, A Review on SECS/GEM : A Machine-*to*-Machine (M2M) Communication Protocol for Industry 4 . 0, *International Journal of Electrical and Electronic Engineering & Telecommunications*, Vol. 10, No. 2, pp. 105–114, March, 2021.

[4] N. Tuptuk and S. Hailes, Security of smart manufacturing systems, *Journal of Manufacturing Systems*, Vol. 47, pp. 93–106, April, 2018.

[5] S. Morgan, 2021 Report: Cyberwarfare In the C-Suite, January, 2021.

[6] Sydney Peng, The Real Reason Behind the TSMC Cyber Attack, *CommonWealth Magazine*, November, 2018.

[7] F. Chen, Y. Huo, J. Zhu, and D. Fan, A Review on the Study on MQTT Security Challenge, *SmartCloud 2020: IEEE International Conference on Smart Cloud*, Washington DC, USA, 2020, pp. 128–133.

[8] S. Hernández Ramos, M. T. Villalba, and R. Lacuesta, MQTT Security: A Novel Fuzzing Approach, *Wireless Communications and Mobile Computing*, Vol. 2018, 2018.

[9] C. Patel and N. Doshi, A Novel MQTT Security framework In Generic IoT Model, Procedia *Computer Science*, Vol. 171, pp. 1399–1408, January, 2020.

[10] A. Rahman, S. Roy, M. S. Kaiser, and M. S. Islam, A Lightweight Multi-tier S-MQTT Framework to Secure Communication between low-end IoT Nodes, *5th International Conference on Networking, Systems and Security* (NSysS), Dhaka, Bangladesh, 2018, pp. 1–6.

[11] N. Mühlbauer, E. Kirdan, M. O. Pahl, and G. Carle, Open-Source OPC UA Security and Scalability, 25th *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, Vienna, Austria, Vol. 1, 2020, pp. 262–269.

[12] Mühlbauer, Nikolas, Erkin Kirdan, Marc-Oliver Pahl, and Georg Carle, Open-Source OPC UA Security and Scalability, *25th IEEE International Conference on Emerging Technologies and Factory Automation*, Vienna, Austria, September, 2020, pp. 262-269.

[13] Z. Shelby, K. Hartke, and C. Bormann, The Constrained Application Protocol, RFC-7252, January, 2014.

[14] M. Iglesias-Urkia, A. Orive, and A. Urbieta, Analysis of CoAP Implementations for Industrial Internet of Things: A Survey, *Procedia Computer Science*, Vol. 109, No. 2016, pp. 188–195, January, 2017.

[15] M. Friesen, G. Karthikeyan, S. Heiss, L. Wisniewski, and H. Trsek, A comparative evaluation of security mechanisms in DDS, TLS and DTLS, *Kommunikation und Bildverarbeitung in der Automation*, Berlin, Heidelberg, 2020, pp. 201–216.

[16] F. T. Ewe, K. S. Yeo, S. C. Yeoh, and C. T. Khoh, Data Analysis on SMT Reflow Oven with SECS/GEM Communication Protocol, *ISCAIE 2020 - IEEE 10th Symposium on Computer Applications and Industrial Electronics*, Penang, Malaysia, 2020, pp. 118–124.

[17] F. Stoop, G. Ely, R. Menna, G. Charache, T. Gittler, and K. Wegener, Smart factory equipment integration through standardised OPC UA communication with companion specifications and equipment specific information models, *International Journal of Mechatronics and Manufacturing Systems*, Vol. 12, No. 3–4, pp. 344–364, 2019.

[18] A. Weber, B. Grey, B. Rubow, D. Francis, D. Lindsey, J. Lopez, J. Cravotta, M. Bennet, T. Hutchison, *Features & Benefits of GEM: A guide to some of the main features and benefits of GEM – SEMI Standard E30*, Cimetrix, 2019.

[19] G. Baweja and B. Ouyang, Data acquisition approach for real-time equipment monitoring and control, *13th Annual IEEE/SEMI Advanced Semiconductor Manufacturing Conference: Advancing the Science and Technology of Semiconductor Manufacturing*, Boston, MA, USA, 2002, pp. 223-227.

[20] A. B. Nelson, P. M. K. Chow, V. A. Snowball, S. Chung, and A. Majumdar, High-speed SECS message services (HSMS) pass-through including bypass. United States Patent US 8, January, 2012.

[21] L. Ma, N. Zhang, and Z. Zhang, Tool Efficiency Analysis model research in SEMI industry, *4th International Conference on Energy Materials and Environment Engineering (ICEMEE 2018)*, Selangor, Malaysia Vol. 38, 2018, pp. 1–5.

[22] J. Postel, Transmission control protocol, RFC- 793, September, 1981.

[23] T. Sauter and A. Treytl, Semiconductor equipment and materials international interface and communication standards: An overview with case studies, *Industrial Communication Technology Handbook*, Second Edition, CRC Press, 2015.

[24] J. E. Rubio, C. Alcaraz, R. Roman, and J. Lopez, Current cyber-defense trends in industrial control systems, *Computers & Security*, Vol. 87, p. 101561, November, 2019.

[25] S. Pfrang and D. Meier, On the detection of replay attacks in industrial automation networks operated with profinet IO, *3rd International Conference on Information Systems Security and Privacy*, Porto, Portugal, 2017, pp. 683–693.

[26] A. Al-Ani, M. Anbar, S. A. Laghari, and A. K. Al-Ani, “Mechanism to prevent the abuse of IPv6 fragmentation in OpenFlow networks,” PLoS One, vol. 15, no. 5, 2020.

[27] A. K. Al-Ani, M. Anbar, A. Al-Ani, and D. R. Ibrahim, Match-Prevention Technique Against Denial-of-Service Attack on Address Resolution and Duplicate Address Detection Processes in IPv6 Link-local Network, IEEE Access, Vol. 8, pp. 27122-27138, January, 2020.