A brief look at the security of DeviceNet communication in industrial control systems

Pal-Stefan Murvay
Politehnica University of Timisoara
Timisoara, Romania
pal-stefan.murvay@aut.upt.ro

Bogdan Groza
Politehnica University of Timisoara
Timisoara, Romania
bogdan.groza@aut.upt.ro

ABSTRACT

Security is a vital aspect of industrial control systems since they are used in critical infrastructures and manufacturing processes. As demonstrated by the increasing number of emerging exploits, securing such systems is still a challenge as the employed fieldbus technologies do not offer intrinsic support for basic security objectives. In this work we discuss some security aspects of DeviceNet, a communication protocol widely used for control applications especially in the North American industrial sector. Having the Controller Area Network (CAN) protocol at its base, DeviceNet inherits all the vulnerabilities that were already illustrated on CAN in-vehicle communication. We discuss how the lack of security in DeviceNet can be exploited and point on the fact that these vulnerabilities can be modelled by existing formal verification tools and countermeasures can be put in place.

CCS CONCEPTS

• **Security and privacy** → **Denial-of-service attacks**; • **Information systems** → *Process control systems*; • **Computer systems organization** → Embedded systems;

KEYWORDS

industrial control systems, security, DeviceNet, attacks

ACM Reference Format:
Pal-Stefan Murvay and Bogdan Groza. 2018. A brief look at the security of DeviceNet communication in industrial control systems. In *Central European Cybersecurity Conference 2018 (CECC 2018), November 15–16, 2018, Ljubljana, Slovenia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3277570.3277575

1 INTRODUCTION AND MOTIVATION

Industry areas such as critical infrastructure and manufacturing have become increasingly dependent on industrial control systems (ICS) which have evolved and adapted to suit the needs of all industry sectors. Inevitably, this made ICSs a target for attackers leading to more and more attacks being reported in recent years. The well known Stuxnet worm [8] targeted specific systems involved in the process of uranium enrichment. In another instance the BlackEnergy malware was used to cause a considerable power outage in western Ukraine in 2015 [21]. Such attacks raise awareness on the security of ICSs in general and on the technologies used at the network level in particular

Multiple fieldbus technologies were developed to facilitate communication in ICSs. However, these industrial network protocols were not developed with security in mind enabling an attacker to mount successful exploits once gaining access to the internal process control network. Studying the protocol specification can help in revealing weaknesses and attacks that they can facilitate. This work focuses on the vulnerabilities that stem from the DeviceNet protocol specification, a member of the family of networks that uses the Common Industrial Protocol (CIP) at the higher-level. The CIP network adaptions (i.e. EtherNet/IP, DeviceNet, ControlNet and CompoNet) are developed and maintained by ODVA (Open DeviceNet Vendor Association). While having common specifications for the upper application layer, each of these protocols uses specific adaptations of CIP. DeviceNet, also known as CIP on CAN (Controller Area Network), is employed by millions of installed nodes according to ODVA [14]. Using CAN as the building block covering the physical and data-link layers makes DeviceNet susceptible to the same types of attacks that are possible on the basic CAN (e.g., DoS, spoof, replay). We will not insist on the particularities of these attacks since they were extensively studied as part of research done on the security of the automotive sector in which CAN is the most widely used network protocol. We focus on attacks that use these basic CAN vulnerabilities to exploit the DeviceNet protocol specification from the upper application layers.

The remainder of this paper is structured as follows. In section 2 we present related work on the security of industry-standard protocols. A basic description of the DeviceNet protocol is given in section 3 before presenting the identified attacks in section 4. Finally a discussion on vulnerability modelling and securing DeviceNet is included in sections 5 and 6 followed by the paper conclusion.

2 RELATED WORK

Intrinsic vulnerabilities in the CAN protocol were first reported in [20]. Later experimental studies in the automotive domain has shown that replay and spoofing attacks launched

CECC 2018, November 15–16, 2018, Ljubljana, Slovenia

ACM ISBN 978-1-4503-6515-4/18/11…$15.00
https://doi.org/10.1145/3277570.3277575

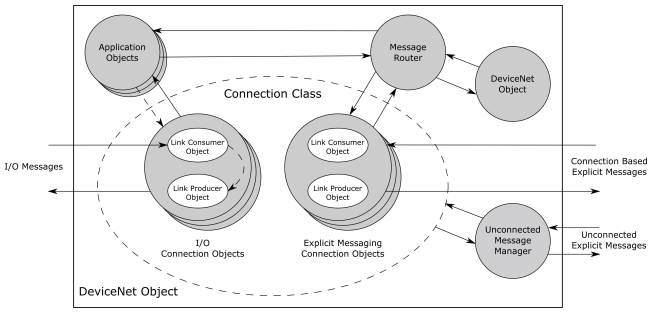

Figure 1: The DeviceNet object model according to [12]

from the application layer can be easily used to control the behavior of nodes in a CAN network [3, 9], while the protocol's arbitration and fault confinement mechanisms can be exploited to mount DoS attacks [10].

Research on the security of the CIP family of protocols has mainly focused on the EtherNet/IP protocol. A man-in-the-middle approach to attack EtherNet/IP communication is discussed in [19]. While effective in affecting the controlled process this type of attack is based on the assumption that the attacker has the ability of planting a device in the ICS network. The authors of [5] report a set of DoS attacks on EtherNet/IP nodes based on connection space exhaustion, flooding and TCP connection timeouts. The possibility to exploit vendor specific extensions of CIP to mount DoS attacks is investigated in [17].

With adaptations, some of these attacks could be mounted on DeviceNet communication given the shared CIP specification. However, to the best of our knowledge this is the first paper to investigate attacks on the DeviceNet protocol.

In depth specification analysis was done for other fieldbus protocols such as DNP3 and Modbus to build detailed attack taxonomies [4, 7].

3 DEVICENET PROTOCOL

The specification of DeviceNet, as provided by ODVA, consists of two main documents: the Common Industrial Protocol Specification [11] and the DeviceNet adaption of CIP [12]. The main body of the CIP specification covers the top layers of the protocol stack. CIP was designed as an object oriented protocol following a producer-consumer communication model. It uses object modelling to define the visible network behaviour of nodes which are viewed as collections of objects. Figure 1 depicts the object model of a DeviceNet node presenting object types along with their relationships and illustrates the class concept. A class is a set of objects with related services, attributes and behaviours. Services are the procedures that an object can perform while attributes are object characteristics. The CIP specification provides a so-called library of objects which defines objects that cover typical functionalities required in network nodes. For other more specific objects that could be needed the specification covers the possibility of defining vendor-specific objects.

CAN ID bits										Value range	Designated use
10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**		
0	ID^{G1}_{Msg}				Src. MAC ID						000-3ff	Message Group 1
1	0	Src./Dest. MAC ID					ID^{G2}_{Msg}				400-5ff	Message Group 2
1	1	ID^{G3}_{Msg}			Src. MAC ID						600-7bf	Message Group 3
1	1	1	1	1	ID^{G4}_{Msg}						7c0-7ef	Message Group 4
1	1	1	1	1	1	1	x	x	x	x	7c0-7ef	Invalid CAN Identifiers

Figure 2: CAN ID encoding in DeviceNet

While the main CIP specification is common to a family of protocols, special restrictions on the application of CIP are defined for each of these protocols. DeviceNet represents the implementation of the CIP on the widely used CAN technology. DeviceNet uses CAN as the basis for the physical and data link layers. The DeviceNet physical layer extends the classical CAN specification [15] by adding improved transceiver characteristics along with options for cabling and connectors [16]. At the data link layer DeviceNet uses the classical CAN protocol which specifies standard frames having an 11 bit ID field and a data payload of up to 8 bytes, therefore, excluding the use of extended frames. As an additional restriction, the use of remote frames is not allowed.

3.1 General communication principle

An addressing scheme is used for addressing specific objects within nodes as part of the communication process. Each node is assigned an unique integer identifier which, in the case of DeviceNet, is called MAC ID (Media Access Control Identifier). To assure uniqueness, the MAC ID of a node trying to gain network access is checked and access is only granted if this MAC ID is not already in use. Classes, object instances and attributes also have associated integer identifiers while services are represented by corresponding service codes.

The CIP messaging protocol is based on establishing connections between nodes. Each established connection is assigned a connection identifier (CID). Two CIDs are allocated if the connection involves bidirectional data transfer. The CID is transmitted in the CAN ID field which is used to encode four different message groups as depicted in Figure 2. Here ID^{Gi}_{Msg} represents the identifier of a message within group i. The CID is built by using a message ID in combination with a MAC ID resulting in a limited number of connections per MAC ID when also considering that some message IDs are reserved for predefined use.

The two main types of connections used in CIP networks are *explicit messaging connections* and *I/O connections*. Explicit connections are used for multi-purpose, point-to-point communication following a typical request/response approach within a client/server model. The client is the device that sends a request to a server, while the server node is expected to respond upon receiving the request. I/O connection are dedicated to specific-purpose, unicast or multicast communication following a producer/consumer model. A producer node places a message on the network to be consumed by one or several consumer devices.

3.2 Establishing connections

There are several methods for establishing connections, their usability, however, depends on the node state or CIP protocol adaptation. Connections can be established by using one of three mechanisms: (i) the *Unconnected Message Manager*, (ii) an already existing explicit messaging connection or (iii) the *Predefined Master/Slave Connection Set*.

Unconnected Message Manager. The Unconnected Message Manager (UCMM) is responsible for establishing explicit messaging connections in the absence of an already established connection that can be used for this purpose. Messages used in this case are called unconnected explicit messages as they have the form of an explicit message but they are not used over an established connection. In DeviceNet nodes UCMM supports two request services (for opening and closing connections) and five response services (for opening and closing connections as well as for device status notifications). Although not mandatory, the implementation of the UCMM in all DeviceNet nodes is highly recommended by the specification since devices without UCMM functionality bring additional burden on other nodes and on network bandwidth.

Connection Object. An already established connection between two nodes can be used to create subsequent connections by sending *create* service requests to the connection object class. Similarly, the *delete* service can be used to delete a specific connection or all existing connections within the class. However, the support for these two services is optional.

Predefined Master/Slave Connection Set. Some DeviceNet nodes have limited capabilities preventing them from employing the previously described approaches for establishing connections. To minimize the message processing load on this type of nodes, DeviceNet introduces the predefined master/slave connection set which provides 5 predefined connection types in a node, one for explicit messaging and four for I/O messaging. Nodes using predefined master/slave connection set messages communicate using a master/slave relationship. The master "takes control" over slaves by allocating their predefined connection set to its own MAC ID and gathers or distributes data to or from the controlled slaves. A slave can only allocate its predefined master/slave connection set to a single master at a given time. The master releases control over a slave by using the release service. The allocation and release of predefined master/slave connection sets is handled by the DeviceNet object which must have an instance per each physical network connection in a DeviceNet node.

4 ATTACKS ON THE DEVICENET PROTOCOL

In this section we will present attacks on the DeviceNet protocol identified by analysing the specification. For performing these attacks we considered the case of an attacker with direct access to the DeviceNet network that has the ability to listen to DeviceNet traffic and inject any forged message. Access to the DeviceNet network is gained either by compromising an already existing node or by connecting a new node. For a better understanding of the protocol vulnerabilities involved in each case the description of each attack is preceded by background details on the exploited feature.

(1) **Deny network access**

Background. Prior to initiating communication in a DeviceNet network a node has to gain network access by successfully passing the duplicate MAC ID check. This step involves sending the duplicate MAC ID check request at least two consecutive times (one time if using Quick Connect) without receiving a response. A response is sent by a network node that receives a check request for a MAC ID identical to its own. Both the request and response messages contain additional identification data about the sender nodes such as Vendor ID and serial number. If the duplicate MAC ID check fails the node enters a communication fault state which can be exited by manual intervention or a MAC ID change.

Attack. The network access procedure is similar to the IPv4 address conflict detection[1] making it prone to the same type of attacks. A DeviceNet node can be denied network access if an attacker always sends a response to the legit node's duplicate MAC ID check requests. The target node can be easily identified based on the vendor ID and serial number sent as part of the duplicate MAC ID check request. The DeviceNet specification does not require the use of a timeout for duplicate MAC ID check responses after the final request transmission. This omission along with the state machine describing the network access flow would suggest that it is possible to send false duplicate MAC ID check responses to nodes that have already established normal operation forcing them to transition in the fault state.

(2) **Disrupt UCMM connection process**

Background. The UCMM object facilitates the process of establishing explicit messaging connections. Such connections are established by using the Open Explicit Messaging Connection service. A request for this service contains the source and destination MAC IDs as well as the message ID and information on the messaging format to be adopted. Requests are answered with either a response to the same service or an error. The connection is considered as established on both sides if no error was issued and the response has been received by the request initiator.

Attack. The correct establishment of an explicit messaging connection using the UCMM functionality can be prevented if an attacker that detects the connection attempt transmits an error message before the legit answer is sent by the server node. As a result, the client will consider the connection attempt as failed following the reception of the error message and will drop the correct response that follows as it will not correspond to an ongoing request. Depending on the error code, the client may attempt to retry the connection request with adapted parameters. This retry could be prevented by setting the transmitted error code to Service Not Supported (0x08) suggesting that the requested service is not supported by the server. The connection will still be considered as established on the server side but unusable since any messages

[1] https://tools.ietf.org/html/rfc5227

sent to the client referencing an invalid CID are dropped. Moreover, if the server connection object uses the inactivity/watchdog functionality, the connection will be closed on its end after an inactivity period of at leas 10 seconds (the actual timeout value is configuration dependent).

(3) **Close connections via UCMM**
Background. The UCMM also provides a means for closing connections by implementing the Close Connection service. This service may be used to close any type of connection (i.e. explicit messaging or I/O). A close connection request will terminate the connection only on one of the connected endpoints, i.e. the one denoted by the MAC ID transmitted within the request message.
Attack. An attacker that wants to prevent messages from being transmitted or received over a certain connection can extract the CID from the monitored network traffic and send a close connection request to the targeted endpoint indicating the corresponding CID. The request receiver is forced to terminate the connection on its side ceasing to transmit or accept any messages over this connection.

(4) **Connection disruption via connection class**
Background. The connection class can provide services for creating and deleting connections over an existing explicit messaging connection. A device is only capable of providing this behaviour if the create and delete services are implemented by its connection class. When available, the use of these two services is similar to the use of the open and close services of the UCMM with the added ability of the create service to produce I/O connections. Moreover, the delete service can be used to delete all connections allocated on the device if, instead of addressing a particular connection, the delete request addresses the connection class.
Attack. Using these features an attacker can prevent new connections with an approach similar to the one used for disrupting the UCMM connection process, i.e. by sending error messages upon create connection requests. The same applies to closing connections with the added ability of closing all connections on one node with a single delete request.

(5) **Connection exhaustion**
Background. Each device can allocate a limited number of connections, due to the CAN ID encoding convention and internal device resources. CAN ID encoding, as illustrated in Figure 2, limits the total number of connections that can be initiated by a node to 27 over all message groups since some bit combinations are reserved for special-purpose messaging. The constraint is even greater if the usable message format is limited to a certain message group. Additionally, each device has a maximum number of connections which is established based on the device capabilities. A device that reaches its connection limits will refuse further connection requests.
Attack. An attacker could prevent a target node from achieving and using legitimate connections with other nodes by exhausting its connection capability. This can be done by forcing the target node to establish connections up to its configured limit. If the device's capabilities allows it to establish as many connection as allowed by the CAN ID encoding scheme the attacker needs to enssure that all the possible connections are covered. This might seem as an intensive task for the attacker if we consider that a DeviceNet network can hold up to 64 nodes each of which can initiate as much as 27 connections leading to a total of 1728 connections. It would take the attacker at most $1.8s$ to transmit the connection requests for all these connections at a baudrate of 125kbps not accounting for any computational bottlenecks and network delays caused by legit traffic. Even if legit nodes manage to establish connections with the node targeted by the attack, the connection can be closed on their end by employing the close connection attack. The same approach should be taken to leave any connections initiated by the target node open only on its side as they cannot be covered by the attacker by making connection requests. Another aspect to consider for such an attack is the connection inactivity/watchdog functionality that may force all the opened connections to close in the absence of any messages to use them. One of the parameters used to determine the behaviour of the inactivity/watchdog timer is the *expected_packet_rate* of the connection object. Setting it to 0 disables the use of the timer. This attribute defaults to 2500 (2.5s) for explicit messaging connections and 0 for I/O connections. Therefore, an efficient attacker should attempt to establish only I/O connections, otherwise additional actions are needed to disable the inactivity/watchdog functionality on nodes with which it establishes connections.

(6) **Deny predefined connection set ownership**
Background. The predefined master/slave connection set of a slave must be allocated to a master node before being used. For this the master uses the allocate service to request ownership over the slave. The slave is not allowed to use connections from the predefined set with a node other than its current master until the master releases the ownership using the corresponding release service request.
Attack. This can be exploited by an attacker that allocates ownership over a node's predefined connection set to itself and never releases it preventing the legit master from performing its normal tasks. In case the legit master has already claimed ownership over the target slave, the attacker can forge the release request using the legitimate master's identity making the slave accept new requests for connection set allocation.

(7) **Change connection object attributes**
Background. Each connection object instance must implement a set of mandatory attributes which define the behaviour of the established connection. The values of certain attributes can be modified by using the Set_Attribute_Single service provided that the node implements it. Some of the parameters that can be modified while a connection is established refer to CIDs, packet data rate and timer configuration.
Attack. If implemented, this feature can be exploited to modify connection attributes with malicious intent. For example, an attacker could affect the efficiency of the process control algorithm by configuring a lower data rate for connections used to transport sensor data.

(8) **Disturb fragmented message transmission**
Background. Fragmented transmission is used for transmitting data which does not fit in a single 8 byte CAN frame. Using this feature each message fragment is sent along with sequence information needed to reconstruct the message. Sending a fragment which is out of order according to its sequence information results in all the previously received fragments being discarded. The only exception to this behaviour is the case of acknowledged fragmented transmission which accepts the reception of consecutive duplicated data.
Attack. An attacker can exploit this to prevent the correct reception of fragmented data by transmitting a forged fragmented message transmission with an out of order sequence information. While nodes sending acknowledged fragmented messages will detect the interrupted reception due to the missing acknowledgement, nodes involved in unacknowledged fragmented message transmission will be unaware of the problem in reception.

(9) **Spoof control data**
Background. The main body of the data transmitted in DeviceNet networks is used for process control functionalities either as inputs from sensors or control stations or outputs for actuators. A smaller percentage of traffic is used for establishing and maintaining connections between nodes.
Attack. An attacker listening to DeviceNet traffic can obtain information about device identities and purpose of established connection. Based on this information the attacker can send forged messages to either directly control the function of actuators or feed false sensor data in the process controller. Additional knowledge on the characteristics of the controlled process would increase the attacker's ability to control process output. Although this attack is generic to all CAN-based protocols and already studied in cited related work we include it here for completeness.

5 FORMAL MODELLING

Formal verification tools can be easily used to model the security of DeviceNet. We exemplify this by using the ASLan language which is the specification language of the AVANTSSAR platform [1]. For verification we use CL-Atse [18] which is one of the AVANTSSAR back-ends. The CL-Atse protocol verifier was previously used to model the security of industrial control systems in [18].

In ASLan, a protocol is modelled as a set of transitions from the left-hand side *LHS* to the right-hand side *RHS*, starting from an initial state. The *LHS* and *RHS* are a conjunction of positive and negative facts. Intruder abilities are modelled by the *iknows* predicate which embeds the intruder knowledge. Facts are non-persistent, in contrast, *iknows* is persistent as the intruder never forgets what he learned from protocol execution.

The simplified state diagram for DeviceNet network access is shown in Figure 3. We omit some transitions, e.g., in case of bus-off or quick-connect, due to space constraints. In principle, each node has to send a duplicate MAC check twice before entering the on-line state. In case a duplicate

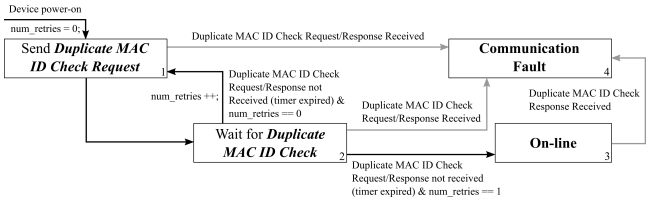

Figure 3: Simplified state machine of the network access mechanism

```
step send_req_duplicate_mac(X, MAC):=
assoc(X, MAC). state_a(X, 0)
 =>
iknows(MAC). iknows(X).
send_mac_request(X, MAC, 1).
send_mac_request(X, MAC, 2).
assoc(X, MAC).
state_a(X, 1)
```

Figure 4: ASLan code for modeling the *wait for duplicate MAC* state of network access transition diagram in DeviceNet

MAC response occurs, a communication fault is signalled. This state diagram is easy to model. A particular challenge however in symbolic executions is to model time (required here to check if a timer expired in state 2) but this can be done by concatenations of an abstract symbol associated to a clock tick.

Figure 4 shows a transition from the ASLan code that we used for modelling the transitions for DeviceNet network access according to the state machine depicted in Figure 3. Principal X associated to adress MAC sends two duplicate MAC requests on the bus (this is according to the DeviceNet specification which mandates for two such requests before going on-line). Fact *assoc(X, MAC)* is duplicated from the *LHS* on the *RHS* since principal X retains his address after the transition. From the transition, the intruder learns the name of the principal and his MAC which is modelled by *iknows(X). iknows(MAC)*. Figure 5 shows the attack trace as output by CL-Atse. First, principal a, in order to enter the network, sends a request for a duplicate MAC. From this request the intruder learns his name a and the value of his MAC represented here by natural number 15. In the second transition, the intruder knowing his name and his MAC value, sends a duplicate MAC response. This triggers the principal in the 3-rd transition to enter in a communication fault state.

6 SECURITY COUNTERMEASURES

The current version of the CIP protocol includes security enhancements based on proven technologies employed in Ethernet-based communication which are intended for EtherNet/IP implementations. Most of these solutions are clearly out of the reach of DeviceNet nodes which are resource constraint devices and use CAN as communication layer (not Ethernet). ODVA promises to include generic methods for

```
ATTACK TRACE
 (a,4) -> i:  15.a
              & Remove state_a(a,0); Remove assoc(a,15); Add state_a(a,1);
              & Add sent_mac_request(a,15,1); Add sent_mac_request(a,15,2);
              & Add assoc(a,15);
              & Built from send_req_duplicate_mac

 i -> none:  15
 none -> i:  {}
             & Remove state_a(i,0); Add sent_mac_response(15);
             & Built from send_duplicate_mac

 i -> (a,4):  {}
 (a,4) -> i:  {}
              & Remove state_a(a,1); Remove sent_mac_response(15);
              & Remove sent_mac_request(a,15,1); Remove assoc(a,15);
              & Add state_a(a,2); Add assoc(a,15);  Add comm_fault(a);
              & Built from wait_for_mac
```

Figure 5: Attack trace output by CL-Atse on the DeviceNet network access model

providing authorization, integrity, non-repudiation and availability in future protocol versions [13].

There is an extensive literature that addresses the security of the CAN bus which stands at the base of DeviceNet. A survey on CAN bus security is available in [6]. The challenge still remains in integrating security in a backward compatible manner for DeviceNet. To add security, authentication tags (regular cryptographic Message Authentication Codes - MACs) have to be embedded in existing frames or sent as separate frames. Embedding the tags along with the message suffers from the limited size of the CAN frame, i.e., 64 bits. According to current standards for automotive security [2] that address CAN frames, 24 bits of security for authentication tags is the recomended security level. This amount can be embedded in the 64 bits along with the message.

In the case of the previously discussed network access steps, adding additional authentication frames should not represent a problem since the network access should not be done so often. DeviceNet has a set of reserved IDs that are invalid CAN identifiers as depicted in Figure 2. These IDs can be used to carry additional authentication information. For regular message, sending the authentication tag in separate frames still adds some limitations since the a tag has to be associated to a particular message, messages need to be buffered until the authentication tag is received and the bandwidth will also be increased. Addressing these issues may remain as future work for us provided that we find a suitable scenario and experimental setup.

7 CONCLUSIONS

We illustrated some attack possibilities in DeviceNet networks as identified based on protocol specification analysis. While the applicability of attacks on optional features depends on the actual instantiation of a DeviceNet node it is clear that even a minimal protocol implementation is prone to DoS and spoofing attacks. The practical consequences of these attacks on industry standard DeviceNet nodes along with the identification of other implementation specific vulnerabilities remains to be determined as a future work. Numerous security solutions have been proposed for the CAN bus and these should fit the requirements of the DeviceNet protocol. Formal modelling may help in assessing security but challenges in modelling cyber-physical systems still persist.

ACKNOWLEDGEMENTS

This work was supported by a grant of the Romanian Ministry of Research and Innovation, CNCS - UEFISCDI, project number PN-III-P1-1.1-PD-2016-1198, within PNCDI III.

REFERENCES

[1] Alessandro Armando, Wihem Arsac, Tigran Avanesov, Michele Barletta, et al. 2012. The AVANTSSAR platform for the automated validation of trust and security of service-oriented architectures. In *Intl. Conf, on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 267–282.

[2] AUTOSAR 2017. *Specification of Secure Onboard Communication* (4.3.1 ed.). AUTOSAR.

[3] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, et al. 2011. Comprehensive Experimental Analyses of Automotive Attack Surfaces. In *USENIX Security Symposium*.

[4] Samuel East, Jonathan Butts, Mauricio Papa, and Sujeet Shenoi. 2009. A Taxonomy of Attacks on the DNP3 Protocol. In *Intl. Conf. on Critical Infrastructure Protection*. Springer, 67–81.

[5] Ryan Grandgenett, William Mahoney, and Robin Gandhi. 2015. Authentication Bypass and Remote Escalated I/O Command Attacks. In *Proc. of the 10th Annual Cyber and Information Security Research Conf.* ACM, New York, USA, 2:1–2:7.

[6] Bogdan Groza and Pal-Stefan Murvay. 2018. Security Solutions for the Controller Area Network: Bringing Authentication to In-Vehicle Networks. *IEEE Veh. Tech. Magazine* 13, 1 (2018), 40–47.

[7] Peter Huitsing, Rodrigo Chandia, Mauricio Papa, and Sujeet Shenoi. 2008. Attack taxonomies for the Modbus protocols. *Intl. Journal of Critical Infrastructure Protection* 1 (2008), 37–44.

[8] Ralph Langner. 2013. *To Kill a Centrifuge: A Technical Analysis of What Stuxnet's Creators Tried to Achieve*. The Langner Group.

[9] Charlie Miller and Chris Valasek. 2013. Adventures in automotive networks and control units. *DEF CON* 21 (2013), 260–264.

[10] Pal-Stefan Murvay and Bogdan Groza. 2017. DoS Attacks on Controller Area Networks by Fault Injections from the Software Layer. In *3rd International Workshop on Secure Software Engineering*.

[11] ODVA 2010. *The CIP Networks Library Vol. 1: Common Industrial Protocol (CIP), Edition 3.9*. ODVA.

[12] ODVA 2010. *The CIP Networks Library Vol. 3: DeviceNet Adaptation of CIP, Edition 1.10*. ODVA.

[13] ODVA. 2016. *Optimization of Industrial Cybersecurity: ODVA's Vision for Securing the Flow of Data in Industrial Networks*. ODVA.

[14] ODVA. 2016. *Technology Overview Series: DeviceNet*. ODVA.

[15] Robert Bosch GmbH 1991. *CAN Specification, Version 2.0, Part A*. Robert Bosch GmbH.

[16] Viktor Schiffer. 2016. *Common Industrial Protocol (CIP) and the Family of CIP Networks*. ODVA.

[17] Francisco Tacliad, Thuy D Nguyen, and Mark Gondree. 2017. DoS Exploitation of Allen-Bradley's Legacy Protocol through Fuzz Testing. In *Proceedings of the 3rd Annual Industrial Control System Security Workshop*. ACM, 24–31.

[18] Mathieu Turuani. 2006. The CL-Atse protocol analyser. In *Intl. Conf. on Rewriting Techniques and Applications*. Springer, 277–286.

[19] David Urbina, Jairo Alonso Giraldo, Nils Ole Tippenhauer, and Alvaro Cárdenas. 2016. Attacking Fieldbus Communications in ICS: Applications to the SWaT Testbed. In *SG-CRC*. Springer, 75–89.

[20] Marko Wolf, André Weimerskirch, and Christof Paar. 2004. Security in automotive bus systems. In *Workshop on Embedded Security in Cars*.

[21] Kim Zetter. 2016. Inside the cunning, unprecedented hack of Ukrain's power grid. https://www.wired.com/2016/03/inside-cunning-unprecedented-hack-ukraines-power-grid/. [Online; accessed 1-July-2018].