Hindawi
Security and Communication Networks
Volume 2017, Article ID 2430835, 17 pages
https://doi.org/10.1155/2017/2430835

Research Article

Identification of ICS Security Risks toward the Analysis of Packet Interaction Characteristics Using State Sequence Matching Based on SF-FSM

Jianxin Xu[1,2,3] and Dongqin Feng[1,2,3]

[1] *Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China*
[2] *National Engineering Laboratory for Safety & Security Technology of Industrial Control System, Zhejiang University, Hangzhou 310027, China*
[3] *State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China*

Correspondence should be addressed to Dongqin Feng; dongqinfeng@zju.edu.cn

Received 3 November 2016; Revised 21 January 2017; Accepted 26 February 2017; Published 13 April 2017

Academic Editor: Mamoun Alazab

This paper discusses two aspects of major risks related to the cyber security of an industrial control system (ICS), including the exploitation of the vulnerabilities of legitimate communication parties and the features abused by unauthorized parties. We propose a novel framework for exposing the above two types of risks. A state fusion finite state machine (SF-FSM) model is defined to describe multiple request-response packet pair sequence signatures of various applications using the same protocol. An inverted index of keywords in an industrial protocol is also proposed to accomplish fast state sequence matching. Then we put forward the concept of scenario reconstruction, using state sequence matching based on SF-FSM, to present the known vulnerabilities corresponding to applications of a specific type and version by identifying the packet interaction characteristics from the data flow in the supervisory control layer network. We also implement an anomaly detection approach to identifying illegal access using state sequence matching based on SF-FSM. An anomaly is asserted if none of the state sequence signatures in the SF-FSM is matched with a packet flow. Ultimately, an example based on industrial protocols is demonstrated by a prototype system to validate the methods of scenario reconstruction and anomaly detection.

1. Introduction

Industrial control systems (ICS) are widely used at the core of national critical infrastructure and are increasingly exposed to public networks that are at risk from cyber-attacks. The control processes of oil and gas facilities, chemical processing plants, power plants, water treatment plants, traffic control systems, and so forth are becoming more vulnerable to threats originating from external networks [1, 2]. These threats not only affect the confidentiality, integrity, and availability of data but also harm the reliability, stability, and safety of physical devices. However, traditional information security solutions do not consider these situations and security may not effectively protect ICS by preventing abnormal data flows that would disrupt the manufacturing process or cause greater damage. Thus, it is necessary to promote the security level under the precondition that the data flows in the industrial control network remain unaffected and to implement some kind of passive security such as monitoring and an alert mechanism. A typical example of an attack of this nature is Stuxnet, which can be abstracted to general ICS attack process [3, 4]. The attacker firstly gains access to the control network and then, through scanning, accesses control process data. Lastly, the attacker assumes control of the ICS. Thus, the attacker would need to determine how to penetrate the ICS LAN and send instructions by industrial protocols as the key points to complete a successful attack. The research aims to address three problems related to the security of ICS. The first involves determining how to use unidirectional monitoring without affecting the original communication.

The second is to identify the inherent vulnerabilities of ICS applications and devices. The third entails detecting the function responsible for abusing the industrial protocols. Thus, we propose a framework to identify the static and dynamic risks, respectively, caused by application inherent vulnerabilities and the unauthorized behavior of intruders. We use the characteristics of a sequence of industrial protocol packets to identify the communication applications. An SF-FSM model is defined to describe the packet sequence using the state transition sequence. The state transition sequences of all the known applications using the same type of industrial protocol can be fused into one SF-FSM. Thus, we match the packet flows with a state transition sequence of the SF-FSM to identify a legitimate application to determine its vulnerabilities as well as the illegal behavior of an unauthorized application. A novel approach to perform rapid and accurate identification of a sequence of packets is also proposed.

2. Motivation and Contributions

This section presents our analysis of the key issues associated with ICS security and our approach to implementing the solutions. The contributions of this paper are also presented and then an outline is given.

2.1. Challenges and Issues of ICS Security. Because of the characteristics listed below, the implementation of ICS security solutions is highly challenging.

(A) Very few security features capable of withstanding advanced cyber-attacks are incorporated within ICS designs.

(B) Considering the importance of process-critical systems, blocking malicious data may not be a major priority.

(C) A sheer number of manufacturers, brands, series, models, and versions of ICS products with a variety of standards mean there is no unified solution.

(D) Industrial communication protocols generally are of three types: open, half-open, and proprietary. Some manufacturers attempt to promote the security level by nondisclosure protocols [5], but any implicit security protocol can be exploited easily by packet capture and reverse engineering.

(E) Physical safety is more important than information security for an online processing control system [6]. Defense procedures tend to focus on preventing physical damage rather than information leakage.

(F) Even if known vulnerabilities exist in the system, they are not easily patched, because the patches may influence the complex industrial on-site environment in uncertain ways.

(G) ICS has a long-term life cycle [7] measured in decades, after which most online systems are outside of their maintenance period or have been upgraded many times such that it would no longer be possible to integrate the documents and design principles.

(H) ICS is generally distributed in wide or dangerous areas; thus, it is not easy to research on-site data relating to the control process.

Firewalls, intrusion prevention systems, isolation gateways, and other traditional information security solutions are unsuitable given these abovementioned characteristics and risks [8]; therefore, proprietary technology and methods need to be proposed for both industrial security and safety.

Addressing ICS security problems requires the implementation of multiple phases [9] for both offline and online systems, particularly for the latter.

The core of ICS security is to protect physical assets and the production process [10], with the essential part being security-in-process. The main process medium of ICS between communication sites is data; thus, we need to overcome the risks resulting from data flow. In addition, the source and carrier of threats are also data. Therefore, data flow is important in ICS security research.

Current generic methods and ideas for specific ICS able to solve the above problems involve the characteristics of each component [11] of the system as well as discovering vulnerabilities and design flaws of products; however, from the point of view of data security, this method is equivalent to only obtaining static data relating to system security, rather than dynamic data of a system in processing. The current problem is the lack of support for on-site application data in the process of security analysis; thus, while we need to establish a channel for on-site data capturing, we also need to research the characteristics of the on-site data.

The ultimate goal of security implementation of ICS is to completely reinforce the system after exposing the system risk, of which the process is carried out in phases. The first step would entail effectively identifying both the known and unknown risks faced by the target system. The known risks are supposed to identify inherent vulnerabilities and improper configuration of the system, whereas unknown risks require data analysis based on audit, forensic, and other types of analyses to backtrack the source, trace the path, and locate the attack point of the risks, so as to generate a warning and alert mechanism that can provide evidence as motivation to strengthen the security even further.

Figure 1 shows the relationships among the ICS, data, risk, and security. The purpose of monitoring [12–14] is to identify threat behavior. The purpose of preventing is to block the threat behavior. The purpose of probing is to discover the risk.

2.2. Evolution of Security Implementation. ICS is a kind of process-critical system; hence, once the system is in the process of continuously running, usually without stopping for one year or more, even if the system poses risks, it is necessary to avoid or reduce changes to the original operating resources of the system. Therefore, when assessing the security level or attaching protection measures, the first consideration is to ensure that the changes or measures on the system have minimal impact. Additional protection could then be implemented as understanding of the target system intensifies.

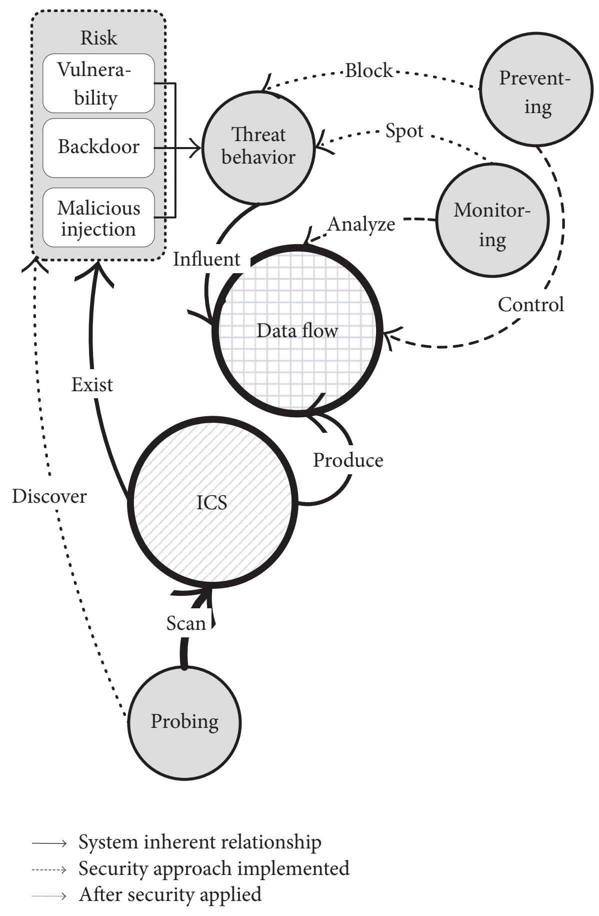


Figure 1: Relationships among ICS, data, risk, and security.

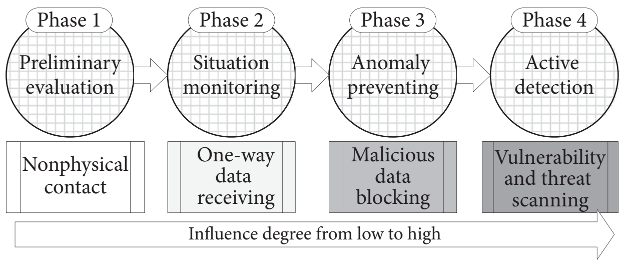


Figure 2: Security implementing process and influence on target system.

We divide the process of security implementation into four phases of which the degree of influence ranges from low to high, as shown in Figure 2.

Phase 1: Preliminary Evaluation. This phase evaluates the risks of the system using nonphysical contact in a static way. The risks mainly involve two aspects: inherent vulnerabilities and misconfiguration. The preliminary evaluation phase does not influence the target system.

Phase 2: Situation Monitoring. This phase is mainly based on the structure of the target system remaining unchanged and capturing and collecting network communication data among hosts on the internal network from data access points that are deployed at key gateways or switches. Then, on the basis of data mining and forensic capability, we can analyze the security situation either manually or in automatic mode and the final result would be presented in the form of a flow statistic or anomaly alert [15]. The situation-monitoring phase only receives one-way data from the target system and therefore does not affect the target system other than deploying some bypass data access points.

Phase 3: Anomaly Preventing. This phase can use some resources and results of the previous phase, including data flow, anomaly warnings, and alerts; however, it prevents intrusion behavior and events in terms of predefined protection rules and policies, which can also be generated by the analysis of historical data and monitoring results and characteristics of situation awareness. The anomaly-preventing phase obtains the communication flow data from either bypass or trunk but needs to block the malicious packets from communication sessions once anomaly events are detected. Thus, we would have to deploy a data blocking point on the trunk that would change the network infrastructure [16] and have an impact on the data stream.

Phase 4: Active Detection. This phase can actively scan and discover potential vulnerabilities and malicious code that exists in the target system by communicating with key components of the system and finally expose risks and threats, with the ultimate ability to take the initiative to defend and repair the system. This process constructs and sends specific packets to the target system. These packets would corrupt software or devices and would be especially harmful for online systems without being tested sufficiently on the processing site. Thus, the impact level of the active detection phase is the highest and can usually be practiced in the process of product development and lifecycle testing.

The most important aspect in ICS security is to understand the risks posed by its own online processing system and to generate alerts in time when threats occur without affecting the system.

As mentioned above, the latter two phases would impact the target system, whereas the former two phases would not. The preliminary evaluation phase can be equated to static analysis as prior knowledge is generally explored manually, whereas the situation-monitoring phase can be equated to the automatic dynamic analysis of data flow. This means that the former two phases can meet the basic needs of ICS security by exposing risks and threats from both internal and external networks.

If these two phases are treated as one input/output system, then the inputs are the target system knowledge and the data flow generated by the system, and the outputs are the system risks and threat events.

2.3. Contributions and Outline. Traditional information security risk assessment [17] considers assets, vulnerabilities, and threats as the essential factors. The qualitative or quantitative risks of a target system are evaluated by analysis and calculation of these three factors. In this paper, a threat is defined as an unknown factor that only presents itself when anomaly events occur. Thus, we treat assets and vulnerabilities as input factors and as system knowledge, which can be acquired through design and deploy documents. However, when the methods in this paper are implemented as a mature product, it usually needs to occur automatically, especially when used in an unknown system; in other words, the assets and vulnerabilities need to be identified without any documents or other supported resources.

In this work, ICS assets are considered to mainly include workstations, servers, network equipment, HMIs, control devices, and field devices according to visible hardware classification, rather than on-site production equipment. The vulnerabilities that are discussed belong to these assets and are harmful to the assets only when exploited by threats.

Data flow is another important form of input to the method proposed in this paper, and it is generated by assets. For one specific system accessed by a monitor product, the only resources that can be obtained are data flow, which can be used for both identification of assets corresponding to vulnerabilities and detection of anomalies.

Overall, in the case of constrained resources, we need to reconstruct the ICS security scenarios including the characteristics of system assets to obtain full understanding of the target based on prior knowledge combined with field data and also monitor the data flow in order to be alerted to unexpected behavior.

The primary index of the reconstruction and detection process is the accuracy of the results, which means an extensive fingerprint library is required. The secondary index is the rate of identification of components and anomalies. Considering the limitation of field resources, an efficient approach needs to be proposed to solve these problems.

In this paper, we consider security risk evaluation and data flow monitoring as one of the major approaches to protect ICS in light of the key points discussed in Section 2 and show the relationships among risks, vulnerabilities, applications, and communication behavior.

A state fusion finite state machine (SF-FSM) model is proposed with the aim of analyzing the packet sequences; it is generally used to identify risks and illegal behavior, which are realized during scenario reconstruction and anomaly detection, respectively.

Several types of packet sequence are also defined and contrasted by the state machine model to describe the characteristics of industrial protocol flow and are regarded as the behavior attributes of the communication component in the software of each application.

An inverted index of keys composing the packets is applied to determine the packet type rapidly, and a set of states to each key is established to easily achieve the state triggered by the packet composed by the chained keys.

The outline of the paper is as follows. The basic communication model of ICS is abstracted and the main risks of ICS in the proposed framework, which illustrates the relationships between risks and data flow in the network of ICS, are highlighted in Section 3. Section 4 contains the data flow analysis model and a detailed description of industrial protocol packet sequence characteristics with the finite state machine model. An approach to address the accuracy and efficiency of the packet matching is also proposed. The contents and results of scenario reconstruction within the framework and model introduced in the previous sections are provided in Section 5. The criteria of anomaly behavior are discussed in Section 6. Then we demonstrate the monitoring system and the results of an application example in Section 7, followed by the conclusions in Section 8.

3. The Proposed Framework

In this section, we analyze the ICS protocol characteristics and risks in the communication process between control stations and devices. A framework composing scenario reconstruction and anomaly detection to manage risks is described.

3.1. Basic ICS Communication Model. An industrial control system mainly includes workstations, servers, embedded controllers, and other controlling and communication devices as its components and is regarded as hosts in this framework. These hosts set up their communication connections between each other in the supervisory control layer network, which is the primary on-site data source as the main study object.

An industrial protocol is the carrier of processing direction, data flow acquisition, resource scheduling, and distribution and it is initiated by the operating system or application software installed on the host we refer to above. Furthermore, one software program may have multiple functions and can be divided into a few communication components, of which each one performs one communication task using one protocol and sets up a session of the protocol in the operation of the system.

In summary, the ICS has four major parts: host, application, communication component, and network protocol from the point of view of the supervisory control layer network under this framework. Thus, the ICS communication model can be abstracted as in Figure 3.

3.2. Risks to Be Identified. The risks associated with one ICS system include its inherent vulnerabilities and features [18] that could be used by outsiders, often with the help of a violated client or server. A violated client can be used to manipulate the control devices and damage the controlled objects. A violated server can be used to deceive the operators and the HMI with a stealthy attack. Often the server in ICS exists in the form of a controller. This is also one of the main differences with traditional Internet security that treats all business as external untrusted behavior whereas ICS treats all business as internal trusted behavior.

Therefore, our work and aim focus on two aspects. The first is to identify vulnerabilities of the hosts in the communication model and the behavior that can exploit the vulnerabilities [19]. The other is to identify behavior associated with illegal feature utilization. The features in this paper refer to the functions that attackers can use, including directly manipulating the control points of the production

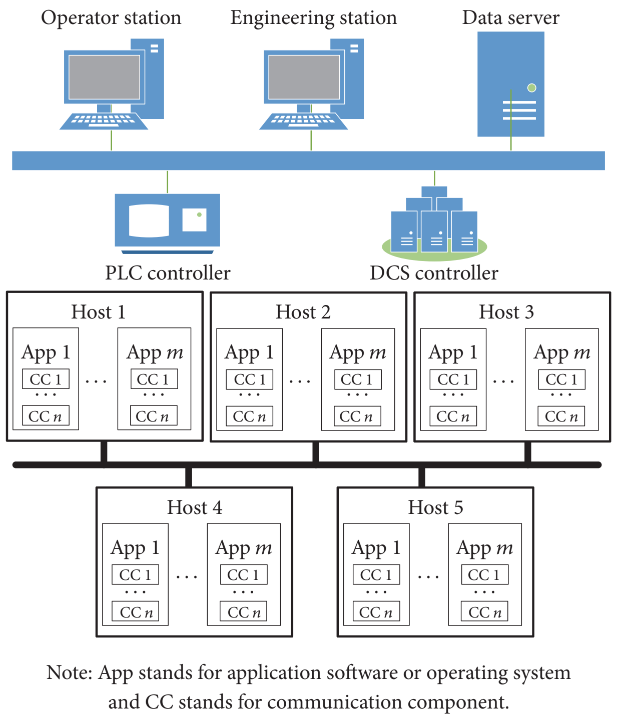


Figure 3: ICS supervisory control layer network and abstract model under proposed framework.

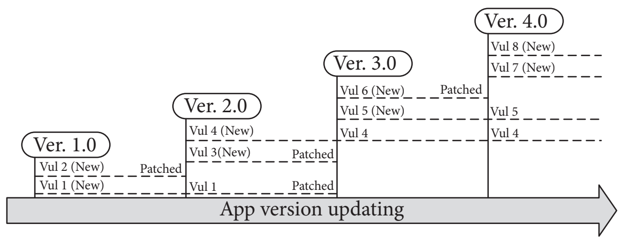


Figure 4: Vulnerability evolution with App updating.

equipment, such as read/write parameters from/to the control devices through specific protocol instructions, or indirect control by tampering with the configuration of a control device, such as rewriting the PLC program also through specific protocol instructions. This may not only ruin the integrity of data but also damage the physical equipment.

Vulnerability exists in software and new vulnerabilities will be revealed or be patched along with software updates [20], including those involving applications and the operating system, as shown in Figure 4. Thus, the software version needs to be identified before known vulnerabilities are listed.

Attackers can exploit the vulnerabilities of specific versions of software and can also take advantage of unprotected features to implement attacks on the ICS. Usually these features are applied through communication flows and packets, which require us to distinguish abnormal communication behavior from normal behavior. Figure 5 shows an example of normal and illegal communication behavior by means of excavating the differences between packet pair sequences. In Figure 5(a), the normal sequence is to read from var1 to var5, whereas in Figure 5(b), the illegal sequence is to read var1 and var5, although there is no limit to this in the standard.

3.3. *Rationale of the Framework.*

Provided the objects and risks that need to be identified have been confirmed, the framework of ICS security can be proposed as in Figure 6.

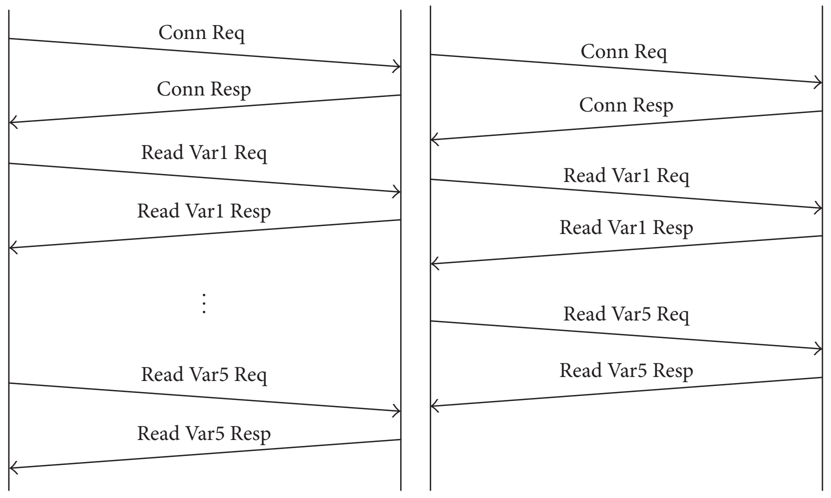
Firstly, the framework has two phases. The first phase is the identifying phase, namely, the reconstruction of security scenarios in the paper to identify applications and their version on each host. Each communication component of the application generates one or more data flows of one type of protocol that can be matched with the standard protocol. A combination of the matched data flows can be used to confirm the type and version of the application. A vulnerability list is then chosen according to this version. In this way, the scenario reconstruction phase is complete. The second phase is the monitoring phase, namely, anomaly detection to alert the user to anomalous behavior through communication data flow in the network. The anomalous behavior includes two types, that is, packets that exploit the identified vulnerabilities and the abuse of the protocol.

Secondly, the framework has two risk dimensions, including a static dimension that indicates the static risks of application inherent vulnerabilities and a dynamic dimension that indicates the dynamic risks of data flows generated by unexpected access programs or terminals. The static vulnerabilities can also be dynamically exploited by attackers constructing specific packets through the communication channel.

Thirdly, there are three essential libraries in the framework—the vulnerability library, exploitation signature library, and protocol characteristic library. The vulnerability library covers the mapping relationship between the specific version of application and vulnerabilities that need to be reconstructed in the security scenarios. Once the application version is determined, a list of vulnerabilities is selected accordingly. The exploitation signature library provides the signature of packets used by attackers to exploit the host; these signatures help us recognize a malicious packet. The protocol characteristic library maintains a characteristic table of the most common industrial protocols so as to be matched with each data flow generated by the communication component and mark it by attaching a protocol label. However, the construction of the libraries is not discussed in this paper as a prior knowledge referring to other related work.

The proposed framework, as a kind of input/output system, only accepts data flows captured in the network as input and uses applications with version information, vulnerabilities, and intrusion alerts as outputs.

Among the factors of the framework, the packet analysis of each data flow in the network, which is used for both scenario reconstruction and anomaly detection, is the most important and time consuming. This process not only considers the integrity and validity of a single packet but also checks the sequence of a series of packets.



(a) Normal (b) Illegal

FIGURE 5: Normal and illegal communication interaction sequence.

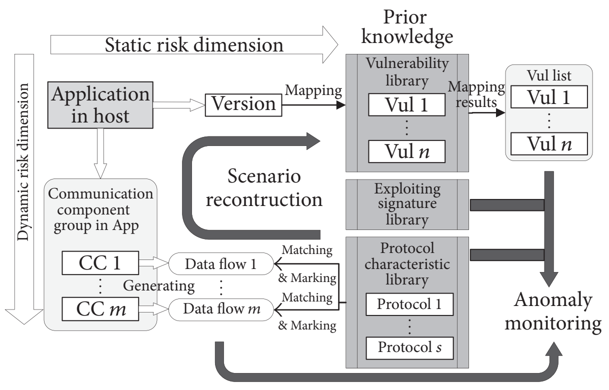


FIGURE 6: Risk identifying and monitoring framework for ICS security.

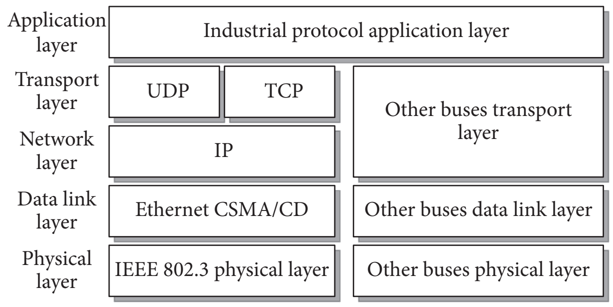

Application layer
Transport layer
Network layer
Data link layer
Physical layer
Industrial protocol application layer
UDP
TCP
Other buses transport layer
IP
Ethernet CSMA/CD
Other buses data link layer
IEEE 802.3 physical layer
Other buses physical layer

FIGURE 7: General industrial protocol stack structure.

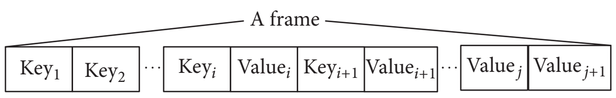


FIGURE 8: Typical industrial protocol packet structure.

A finite state machine is a frequently used method to analyze protocol and this paper proposes the SF-FSM model to describe the packet sequence and to manage the update of the industrial protocol version.

4. Model Analysis of Data Flow

This section presents our analysis of the industrial communication process and presents the SF-FSM to describe the characteristics of the industrial protocol and communication process. An SF-FSM state matching method is also proposed to promote the efficiency of packet sequence matching.

4.1. Industrial Protocol Data Flow. Industrial communication protocols [23] and their dynamic data flows have the following characteristics:

(I) The two communicating parties constitute a client/server model, and the messages between them interact in the request/response format. A request-and-response process usually does not rely on other processes.

(II) Industrial protocols based on Ethernet are normally over TCP/UDP (Figure 7). When using TCP to transport a message, which is generally translated from protocols based on a serial bus such as RS-485, industrial data flow is still transmitted in the form of packets, including a frame as unit instead of a stream, thus supporting compatibility. A frame of an industrial protocol is generally composed of a number of keys and values (Figure 8).

(III) Various versions of protocol standard differ from each other in terms of the addition or removal of functions, service code modification, and so on. In the case of one particular version, manufacturers, third-party developers, and open-source organizations [24] also customize their own communication components to achieve the desired messages.

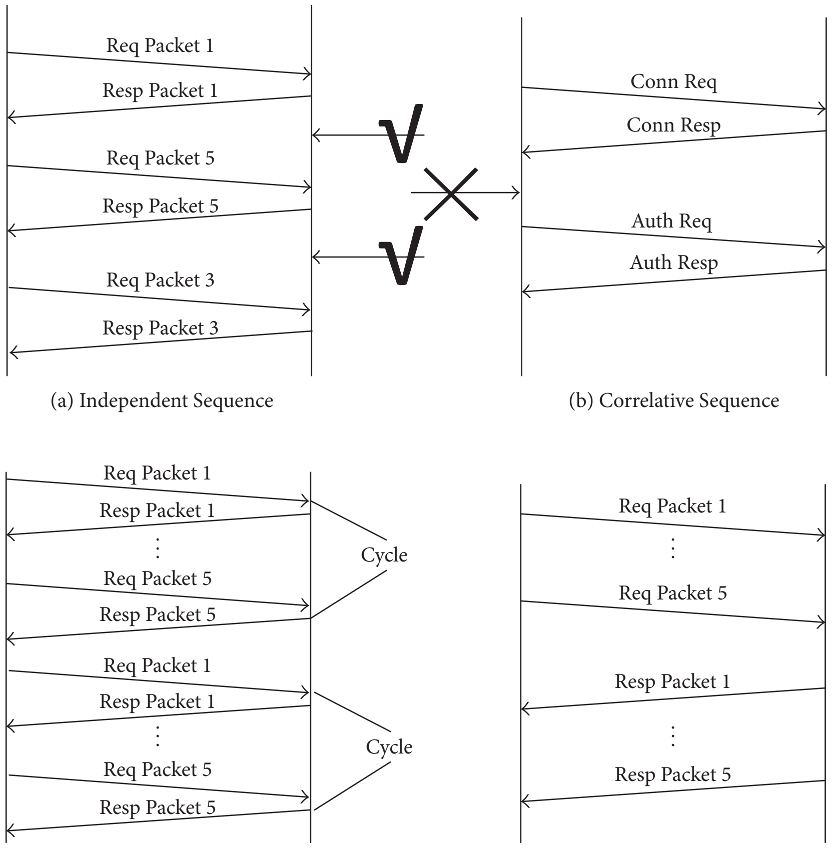


FIGURE 9: Typical sequences in industrial protocol.

(IV) From the perspective of access control, application layer connections, probably including authentication or encryption, are also incorporated in the TCP/UDP sessions.

Overall, a different combination of communication components with a specific version, of which the characteristics reflect those of a request-response packet pair and packet sequence, is used to identify the type and version of the application, whose vulnerabilities are consequently exposed.

Several packet sequences (Figure 9) are used in industrial protocols and can be described as follows:

(i) Independent sequence: an unrelated request-response packet pair, with none of the pairs depending on the other pairs, which means that this type of request-response pair sequence is not rigorously regulated by the protocol standard.

(ii) Correlative sequence: packets are transmitted in a consistent sequence by both communicating parties such as an authenticating sequence and a connection establishing sequence.

(iii) Cyclic sequence: one packet or a serial of packets with the same attributions are periodically transmitted finite or infinite times. This situation is common when reading or writing the I/O supervisory values.

(iv) Pipeline sequence: multiple request packets are continuously sent without waiting for a response packet. Pipeline requests are usually used when configuring the control devices to upload or download the program blocks.

For sequence type (ii) and type (iv), every manufacturer, third-party developer, or open-source organization has to abide by these rules, although they may take advantage of the independence of the pair of packets, for example, type (i) and type (iii), to construct their own facilitated sequences to achieve the expected features without implementing those that are undesired, which leads to the diversity of each application product and can also be used to distinguish illegal applications created by attackers.

Compared to the above scheme, the updated version of the protocol standard may add or remove protocol instructions, modify service codes, and so on, all of which can break the limitation of all sequence types and can also be compatible with the old versions.

To describe the law transforming packet sequences, this paper defines the SF-FSM model to describe the interactive state of the industrial protocol and the transition sequence.

4.2. SF-FSM Model. The finite state machine used for protocol analysis is divided into client state transition and server state transition, which are separately maintained in the client and server components.

In addition, because of characteristic (III) of industrial protocols described in Section 4.1, various components, either clients or servers, and versions implemented by numerous developers may also be given different FSM descriptions.

4.2.1. Baseline SF-FSM and State Transition Condition. The model proposed in this paper can be represented using only one finite state machine to solve the above two problems with state division and state extension.

Definition 1. A protocol packet flow FSM is mathematically modeled in this paper as a quintuple $(\Sigma, S, s_0, \delta, F)$, where

(i) S is a finite, nonempty set of states as

$$S = \{s_i\}, \quad 0 \le i \le N; \tag{1}$$

(ii) Σ is the input packet directory (a finite, nonempty set of packets) as

$$\Sigma = \{p_i\}, \quad 1 \le i \le N; \tag{2}$$

for each state s_i, there is an expected input packet directory as

$$\begin{aligned} \Sigma_{s_i} &= \{p_{s_i,j}\}, \quad 1 \le j \le N, \\ \Sigma_{s_i} &\subseteq \Sigma; \end{aligned} \tag{3}$$

(iii) s_0 is an initial state, an element of S;

(iv) δ is the state transition function, which can be represented by a set of state transition functions from one state to another as

$$\delta = \{\delta_{s_i}(p_{s_i})\}, \quad p_{s_i} \in \Sigma_{s_i}; \tag{4}$$

and the successor state s_{suc} with the input $p_{s_i,j}$ is

$$s_{\text{suc}} = \delta_{s_i}(p_{s_i,j}), \quad p_{s_i,j} \in \Sigma_{s_i}, \ 1 \le j \le N; \tag{5}$$

(v) F is the set of final states, a (possibly empty) subset of S.

The whole state machine processes the data flow as input, of which the packets trigger the state transition. Each state in the FSM has to receive and can accept two inputs, including a request packet and a response packet, both of which are required to meet the state requirement simultaneously to trigger its transition. Thus, one type of state input in the input set Σ can be represented as a packet pair:

$$p_i = (p_{\text{req},i}, p_{\text{resp},i}). \tag{6}$$

As for the condition of accepting only one request packet, we mark the state with a transient state, which becomes a steady state after accepting the response packet, and the

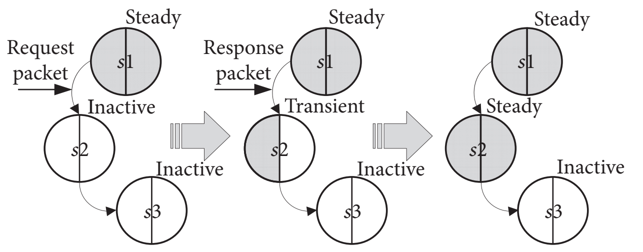


Figure 10: Transient and steady state transition.

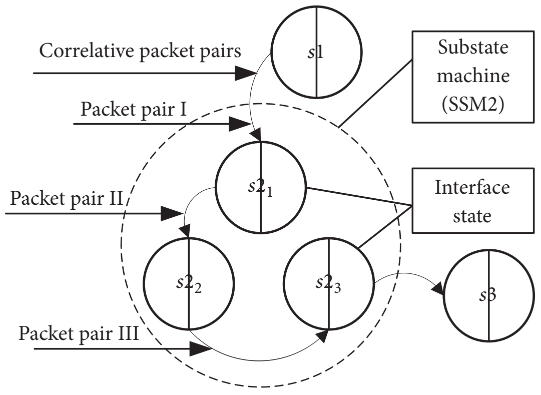


Figure 11: Correlative sequence and substate machine.

transition process is illustrated in Figure 10. Thus, we define the state with three substates:

$$s_i = \begin{cases} s_{ua,i}, & p_i = (\text{NULL}, \text{NULL}) \\ s_{tr,i}, & p_i = (p_{\text{req},i}, \text{NULL}) \\ s_{st,i}, & p_i = (p_{\text{req},i}, p_{\text{resp},i}). \end{cases} \tag{7}$$

4.2.2. Sub-FSM and Special State Transition of ICS Protocol. Considering packet sequence type (ii) described in Section 4.1, we propose the nested state machine as shown in Figure 11, in which substate machines are embedded into the master state machine to process the correlative packet pairs. The interface states are the only way the states in the substate machine can transition from or to the master state machine and no other states may contact the master state directly. Correlative packet pairs are input into the substate machine as a whole and each packet pair is regarded as input of the substate. The substate machine can also be integrated as a state of the master state machine and accept correlative packet pairs as state input.

Definition 2. The substate machine is defined as a quintuple:

$$\begin{aligned} &(\Sigma_{\text{sub}}, S_{\text{sub}}, s_{0,\text{sub}}, \delta_{\text{sub}}, F_{\text{sub}}), \\ &\qquad \Sigma_{\text{sub}} \subset \Sigma, \ S_{\text{sub}} \subset S, \ \delta_{\text{sub}} \subset \delta. \end{aligned} \tag{8}$$

Obviously, the states $s_{0,\text{sub}}$ and F_{sub} are interface states of the sub-FSM.

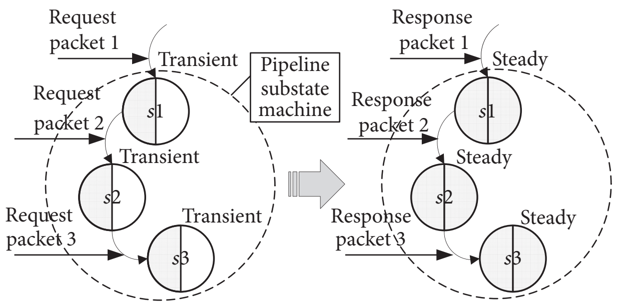
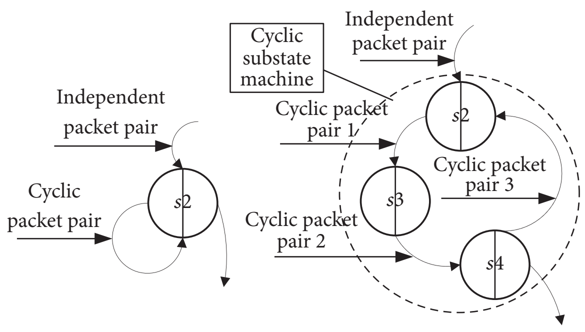


FIGURE 12: Dual state and pipeline state transition.



(a) Single state cycling (b) Multiple states cycling

FIGURE 13: Single and multiple cycling state transition.

(1) Pipeline Substate Machine. Using the dual state in Figure 10 and substate machine in Figure 11 enables us to manage the pipeline sequence and regard the packet pairs in the pipeline as input of the substate machine. Each request packet in the pipeline sequence triggers a substate transition to a transient state until all requests are input; then the response packets are input successively and trigger the steady states as shown in Figure 12. Therefore, another type of state input in the input set Σ can be represented as a list of packet pairs given by

$$p_i = \left(\left(p_{\text{req},i1}, p_{\text{req},i2}, \ldots, p_{\text{req},ij}\right), \left(p_{\text{resp},i1}, p_{\text{resp},i1}, \ldots, p_{\text{resp},ij}\right)\right). \tag{9}$$

(2) Cyclic Substate Machine. There are two types of cyclic packet sequences, including a single packet pair and multiple packet pairs; namely, each of the sequences, respectively, triggers single state cycling (Figure 13(a)) and multiple state cycling (Figure 13(b)), of which the latter can bundle the cyclic states into a cyclic substate machine in order to be described conveniently.

4.2.3. State Fusion Process. We fuse the FSMs of the protocol communication component with various versions by proposing an extension approach as shown in Figure 14.

Before we construct the FSM, we need to prepare some prior knowledge.

Prior Knowledge 1. The packet sequences of each type and version of communication components composing applications and the differences that are used to distinguish them from each other are needed in advance. Each client and server may have a different type or version of protocol implementation but they need to be compatible with each other before they can be represented in one FSM.

We first construct the baseline FSM by selecting the original version of the manufacturer applications, including the client and server. To extend the FSM, either a new state is added with new paths or a new path is added using the original states, and so on, as additional FSMs are fused. This happens when the version on either the client side or server side is updated or changed to another type of application. The changes can be summarized as the following aspects and consequently the FSM can be extended.

(A) If the trigger condition of one state to the follow-up state is changed (in other words, the packet pair is changed, mainly because of the content of the request or response being changed), we add a new state and appropriate paths to the FSM as in Figure 14(c).

(B) If an independent packet pair is inserted into an original packet sequence, then we add a new state and appropriate paths to the FSM as in Figure 14(c).

(C) If an independent packet pair is removed from an original packet sequence, then we add a new path from the predecessor to the successor of the removed state as in Figure 14(b).

The FSM extension process is to fuse two or more FSMs essentially.

Definition 3. Consider the case in which two FSMs are fused; we provide the baseline FSM $(\Sigma_b, S_b, s_{b,0}, \delta_b, F_b)$ and the updated FSM $(\Sigma_u, S_u, s_{u,0}, \delta_u, F_u)$. Then for the new SF-FSM $(\Sigma_e, S_e, s_{e,0}, \delta_e, F_e)$, we obtain

$$\Sigma_e = \Sigma_b \cup \Sigma_u, \tag{10}$$

$$S_e = S_b \cup S_u, \tag{11}$$

$$s_{e,0} = s_{b,0} = s_{u,0}, \tag{12}$$

$$\delta_e = \delta_b \cup \delta_u, \tag{13}$$

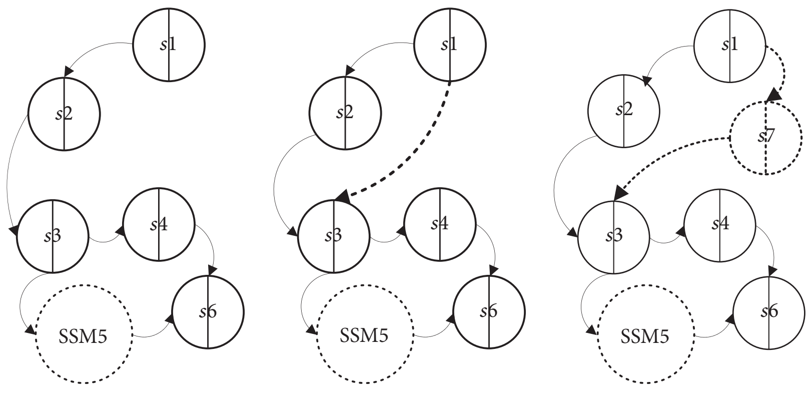
$$F_e = F_b \cup F_u. \tag{14}$$

Here it should be noted that δ_e in (13) may have two state transition functions δ_{b,s_i} and δ_{u,s_i} for one state s_i if the two functions are inconsistent.

In addition, changes in the states and paths of the substate machine do not affect the master state machine; hence, the substate and master state machines can be fused separately without affecting each other.

4.3. Relationship of States and Components. After the FSMs are constructed and fused, we need to establish the relationship between state sequences and the communication component type and version.

We create a mapping table such as Table 1 to assign a unique identification to each type and version of the communication component, which is associated with the states the



(a) Baseline FSM (b) Path extension (c) State and path extension

Figure 14: Baseline FSM and extension.

Table 1: Mapping of component and identification.

Communication component		Identification
Type	Version	
Own equipment manufacturer client	V1.0	1
	V2.0	2
Own equipment manufacturer server	V1.0	3
Third-party developer client	V1.0	4
Open source server	V1.0	5
⋮	⋮	⋮

FSM of the component of a specific type or version contains, while each state in the SF-FSM also maintains two sets of identifications mapped separately with the types and versions of the communication client and server.

Through every state transition we obtain two new sets of identifications. We ensure that a unique identification pair of the client and server can be obtained by taking the intersection of all the sets of a state transition sequence until a complete state transition signature sequence is successfully matched by the captured packet flow. The identification, that is, the type and version of the communication component, is not only related to the combination of the states in sequence, but also related to the order in which the states undergo transition; thus we choose the sequence of states as the signatures.

The two sets of identifications for each state are represented as follows:

$$\begin{aligned} \mathrm{ID}_{\mathrm{cli},s_i} &= \left\{\mathrm{id}_{\mathrm{cli},j}\right\}, \quad 1 \le j \le \max\left(\mathrm{ID}_{\mathrm{cli},s_i}\right), \\ \mathrm{ID}_{\mathrm{svr},s_i} &= \left\{\mathrm{id}_{\mathrm{svr},k}\right\}, \quad 1 \le k \le \max\left(\mathrm{ID}_{\mathrm{svr},s_i}\right). \end{aligned} \tag{15}$$

Theorem 4. *In an SF-FSM, $\forall id_{cli} \in ID_{cli}$ ($\forall id_{svr} \in ID_{svr}$), there is one and only one state transition signature sequence $C_{id_{cli}} = \overrightarrow{\{s_i\}}$ ($C_{id_{svr}} = \overrightarrow{\{s_i\}}$) mapping with id_{cli} (id_{svr}) and $C_{id_{cli}}$ ($C_{id_{svr}}$) is an ordered set.*

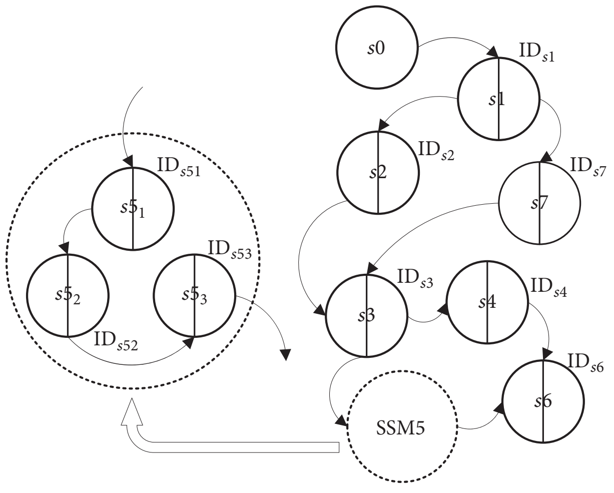


Figure 15: States with identification set.

In order to satisfy Theorem 4, we need to assure that each state transition signature sequence is unique after fusing the FSMs.

Theorem 5. *In an SF-FSM, the state transition sequence is a signature only when at least one δ of the sequence exists and does not exist in any other state transition sequence all of which can be mapped with a unique id.*

Actually, through a state transition signature sequence, both id_{cli} and id_{svr} can be confirmed at the same time. Hence, id_{cli} and id_{svr} can be arranged in pairs to each state in a state transition signature sequence as

$$id_{s_i} = \left(id_{cli,s_i}, id_{svr,s_i}\right). \tag{16}$$

So (15) can be replaced with (Figure 15)

$$\begin{aligned} ID_{s_i} &= \left\{id_{s_i,j}\right\} = \left\{\left(id_{cli,s_i,j}, id_{svr,s_i,j}\right)\right\}, \\ &\qquad 1 \le j \le \max\left(ID_{s_i}\right). \end{aligned} \tag{17}$$

Moreover, the state transition signature sequence in Theorem 4 can be represented by

$$C_{id} = C_{(id_{cli}, id_{svr})} = \overrightarrow{\{s_i\}}, \quad s_i \in S_e. \tag{18}$$

Similarly, C_{id} is also an ordered set.

Theorem 6. *In an SF-FSM $(\Sigma_e, S_e, s_{e,0}, \delta_e, F_e)$, for a state transition signature sequence $C_{id} = \overrightarrow{\{s_k\}}$, k is the index of s in C_{id} and K is the number of items in C_{id}; then the unique id mapping with the C_{id} is*

$$id = \left(\cdots\left(ID_{s_1} \cap ID_{s_2}\right)\cdots \cap ID_{s_{k-1}}\right) \cap ID_{s_k}, \quad 1 \le k \le K. \tag{19}$$

4.4. Packet Flow and State Matching. Using the SF-FSM model, we can determine the id of the communication component by matching the packet pair sequence and the state transition signature sequence.

Theorem 7. *Given a packet pair sequence $P = \overrightarrow{\{p_i\}}$ of a specific protocol and the corresponding SF-FSM, if a state transition signature sequence C_{id} is hit by the input P, then id of communication component which generates P is determined.*

Then (19) can be equivalent to

$$id = \left(\cdots\left(ID_{\delta_{s_0}(p_0)} \cap ID_{\delta_{\delta_{s_0}(p_0)}(p_1)}\right) \cap ID_{\delta_{\delta_{\delta_{s_0}(p_0)}(p1)}(p_2)}\cdots\right). \tag{20}$$

If the result is ununique or null, then the matching is considered to have failed.

A number of key words can be taken from the industrial protocol packet with reference to the packet structure in Figure 8. The efficiency of the packet flow matching is improved by constructing an inverted index list of keys to identify packets and the state sequence.

An inverted index is usually used in fast full text searches, of which the keys are not in a predictable position. However, for a packet, the offset of a key is definitely known; thus we can index the keys according to the offset.

In this paper, we are only interested in the packet type, which is determined by the keys composing the packet, rather than the other parts of the packet because we do not inspect the content. We can therefore regard the packet pair as well as the input of the state in SF-FSM as a set of keys.

Definition 8. The packet pair, namely, the expected input $p_i \in \Sigma_e$ of a state in an SF-FSM can be defined as

$$\begin{aligned} p_i &= \left\{\text{key}_{p_i,1}, \text{key}_{p_i,2}, \ldots, \text{key}_{p_i,j}\right\} \\ &= \left\{\left(\text{offset}_{p_i,1}, \text{value}_{p_i,1}\right), \left(\text{offset}_{p_i,2}, \text{value}_{p_i,2}\right), \right. \\ &\quad \left.\cdots\left(\text{offset}_{p_i,j}, \text{value}_{p_i,j}\right)\right\}, \quad 1 \le j \le N. \end{aligned} \tag{21}$$

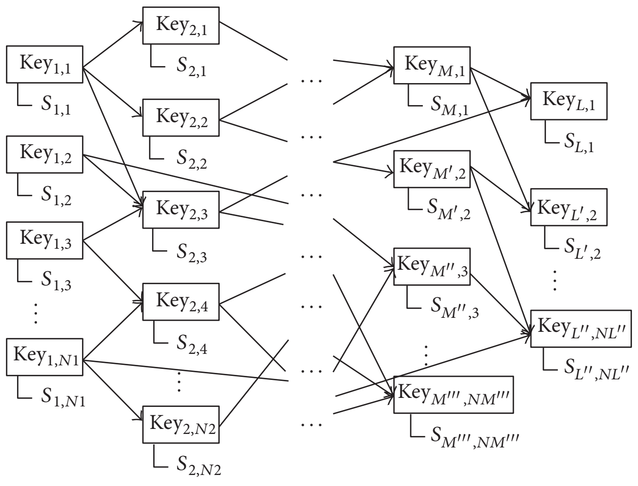


Figure 16: Chained inverted index of protocol keys.

For each state, $p_{s_i,j} \in \Sigma_{s_i}$ can be represented by

$$p_{s_i,j} = \left\{\text{key}_{p_{s_i,j},1}, \text{key}_{p_{s_i,j},2}, \ldots, \text{key}_{p_{s_i,j},k}\right\}. \tag{22}$$

As described above, if all the keys of an input packet pair are matched with (22), then the SF-FSM can be triggered from s_i to the successor state.

Assumption 9. For all $p_i \in \Sigma_e$, any key set of p_i in a specific sequence differs from other sequences.

Prior Knowledge 2. On the basis of Assumption 9, p_i is unique but the key is not unique. Different packets may have the same key at the same offset or different offset and, similarly, the same offset may have a different key.

Theorem 10. *We set up a chained inverted index of keys (Figure 16), of which each one points to the next keys in all p_i and has an attribute of state information, which is represented as $S_{key} = \{s_i\}$, $1 \le i \le N$. From each head key to the tail key through the chain, we can obtain p_i that triggers the SF-FSM from the current state to the next, where the next state is the intersection of S_{key} of all the keys traversed in the chain.*

Normally, if a $\text{key}_{p_{s_i,j},k}$ chain such as that in (22) is matched, then $p_{s_i,j}$ is determined and the next state can be achieved by $s_{\text{suc}} = \delta_{s_i}(p_{s_i,j})$ from (5).

However, referring to Theorem 10, the successor state is

$$s_{suc} = \left(\cdots\left(S_{key_{p_{s_i,j},1}} \cap S_{key_{p_{s_i,j},2}}\right)\cdots \cap S_{key_{p_{s_i,j},k-1}}\right) \cap S_{key_{p_{s_i,j},k}}. \tag{23}$$

Thus, we can use the inverted index of keys instead of the transition function to achieve the successor state in the SF-FSM.

Based on all of the above, in the case of a new type or version of the component presented, extension of the current

Table 2: Expected results of reconstruction.

Host	Application	Communication component	Vulnerability
H A	App a App b	CC 1 CC 2	Vul (1) Vul (2) Vul (3)
H B	App c App d	CC 3 CC 4 CC 5	Vul (4) Vul (5)
H C	App e	CC 6 CC 7	Vul (6) Vul (7)
	App f	CC 8	Vul (8) Vul (9) Vul (10)

SF-FSM only requires extension of the set of keys and their link relationships.

5. Scenario Reconstruction

As discussed in Section 3.2, scenario reconstruction aims to identify vulnerabilities of a target industrial control system by analyzing the data flow captured in the supervisory control layer network and the advantages of this method are passive, accurate, and extendable.

The results (Table 2 as an example) of scenario reconstruction mainly include vulnerabilities of applications in every host as illustrated in Figure 3.

Table 2 reveals clear correspondences between the applications and vulnerabilities, whereas there is no correspondence between the communication components and the vulnerabilities. Each combination of components corresponds with an application of determined type and version.

Prior knowledge, including prior knowledge 1 in Section 4.2 and prior knowledge 2 in Section 4.4, is necessary to accomplish the expected security scenarios of a specific ICS.

Prior Knowledge 3. The protocols and the communication components the various applications of ICS use is prior knowledge to identify the applications on the hosts.

Prior Knowledge 4. Each packet structure of the protocol the application uses is prior knowledge and the keys and the key sets used to distinguish the packets are also prior knowledge as the protocol characteristic library in Figure 6. However, how to extract the keys is beyond the range of this paper.

Prior Knowledge 5. The vulnerability lists of the specific type and version of applications are prior knowledge as the vulnerability library in Figure 6. The vulnerability information can be achieved on CVE (Common Vulnerabilities and Exposures) or other databases.

Before reconstruction, we need to make some assumptions to facilitate the process in order to adapt the SF-FSM in Section 4.2.

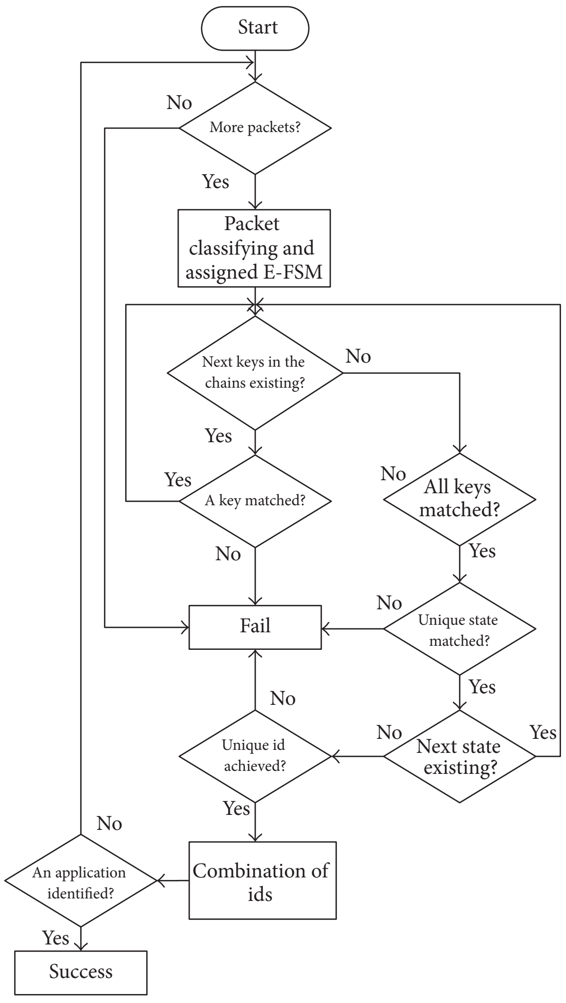


Figure 17: Flow chart of scenario reconstruction.

Assumption 11. Each host has an independent function and none of the hosts carry two or more applications for internal communication. This assumption assures that all the data flows exist in the network and can be captured.

Assumption 12. Each session of data flows can be captured from the first packet; in other words, the SF-FSM can be matched from the first state transition.

Assumption 13. Each session of captured data flows can cover a whole packet pair sequence that triggers the state transition and achieves a unique id indicating the component type and version.

Given an integrated packet flow, the process of scenario reconstruction is as shown in Figure 17.

6. Anomaly Detection

After the scenario reconstruction process, the type and version of each application on a host are determined and, accordingly, the list of vulnerabilities (prior knowledge 5) and the expected packet sequence signatures (Theorem 4) are determined. These are the necessary conditions to implement anomaly detection in Section 3.3.

To protect the ICS from potential risks caused by vulnerabilities, we need to know the approach used by the intruder exploiting the vulnerabilities as prior knowledge.

Prior Knowledge 6. The characteristics of packets, often known as PoC (proof of concept), the intruder use to exploit the vulnerabilities are prior knowledge needed as the exploitation signature library in Figure 6.

In this way we obtain the following anomaly criterion.

Theorem 14. *An application of which the type and version have been determined and the exploitation packet signatures are then determined according to prior knowledge 6. An alert is triggered if a captured packet is matched with the exploitation packet signatures.*

We protect the ICS from potential risks caused by abuse of the packet sequence by judging whether the packet flows are legal in terms of the level of expected packet sequence matching.

Theorem 15. *The type and version of an application can be determined after the packet sequence of each of the communication components used by the application is determined according to Theorem 4. Then the key chains are achieved as Figure 16. An alert is triggered if any of the following conditions are met:*

(a) *None of the keys is matched in the chains.*

(b) *None of the next keys in the chains is matched unless the tail of the chain is reached.*

(c) *All the keys in the chains are matched but the state determined by the matched keys is not the expected one in the state transition sequence.*

Ultimately, under the framework proposed in this paper, two aspects of anomalies are monitored:

(a) *Illegal applications, including clients, servers, and abused features, are discovered by analyzing the data flow.*

(b) *Vulnerability exploitation is detected with a lower cost of signature matching for the known existing vulnerabilities without regard to nonexisting ones.*

7. Experiment

In this section, we introduce a monitoring system and demonstrate a few case studies and experiments to verify the availability of the methods in this paper. Attention is fixed on the accuracy of application identification, which is the chief factor affecting the results of scenario reconstruction and anomaly detection.

7.1. Prototype System and Verification Environment. We present a prototype system matching the criteria of the intrusion detection system [25]. The system is connected to the supervisory control layer of an ICS network and access the

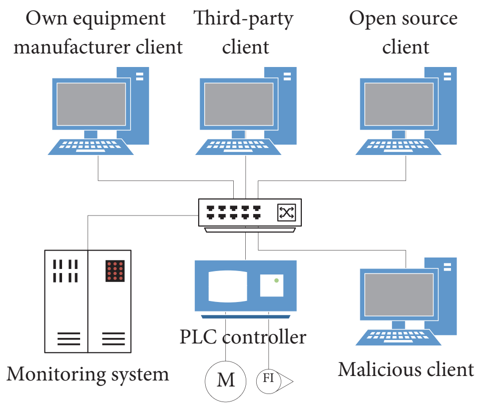


Figure 18: Experimental system and environment.

network traffic unidirectionally that can minimize the impact on the target system under monitoring.

The prototype system is based on an industrial controlling computer, on which CentOS, an operating system based on the Linux kernel, is installed. The application software framework is based on ntop [26], which is a monitoring system supporting hundreds of application layer protocols but not industrial protocols. We added some plugin modules of typical industrial protocols. The prototype system provides two interfaces, one for configuration management and for showing the results of monitoring and the other connected to the mirroring-interface of the switch for achieving data flows in the supervisory control layer network.

The prototype system implements the main functions and verifies the methods proposed in this paper as follows.

(a) The communication parties in the network can be identified from the basic session information, including the protocol type, client address, and server address. Then a virtual network relationship model can be constructed.

(b) The type of client can be identified according to packet interaction sequences in the respective session. The types of clients include own equipment manufacturer ones, certified third-party clients, and open source clients. The client version and the type and version of server are not scheduled for evaluation in this section. Then the vulnerabilities of the identified client can be assessed on the vulnerability publishing platform, such as CVE [27] and ICS-CERT [28].

(c) We construct an attack process by developing a client that meets the protocol specification to evaluate the detection capability of the monitoring system.

The structure of the verification environment is shown in Figure 18. We added a motor and a flowmeter as the controlled plant. The validation process includes two steps.

After the monitoring system is linked to the network, the first step is to construct the normal data flows in the network by using each client to configure the PLC controller and upload the configuration of the PLC controller. Then we manipulate the motor by writing the specific output of the

Table 3: Multiple types of Ethernet/IP client.

Protocol	Own equipment manufacturer client	Third-party client	Open source client
Ethernet/IP	RSLogix 5000	Molex EIP_Tools [21]	EPICS Tools [22]

I/O module and acquire the flowmeter value by reading the corresponding input of the I/O module. The result of the step is to identify the type of each client by the monitoring system.

The second step links the malicious client to the network and then executes the same operation as the other clients. The result of this step is to identify the malicious client and trigger the alert.

The packets can be captured and saved in the monitoring system to enable us to verify the results.

We provide an example of a typical industrial protocol and construct the circumstance to validate the methods and prototype system in the next two subsections.

7.2. Case Study with Ethernet/IP Protocol. In this circumstance, we provide an example of Ethernet/IP protocol supported by the Open DeviceNet Vendor Association (ODVA) [29]. The clients to be identified are listed in Table 3. The model of the PLC controller is a ControlLogix 1756-L62 and the communication module is 1756-ENBT/A.

Accessing data within the controller by the client using a protocol message typically contains the following address information:

(i) Device network address
(ii) Class ID
(iii) Instance ID
(iv) Attribute ID
(v) Service code (describing the action/service required)

When we download or upload the configurations of the PLC controller, we write or read the value of various combinations of the above address information. Thus for each client, the combinations and sequences of the address information may be different.

7.2.1. Signature Extraction for Client Identification. Before we use the SF-FSM to formalize the packet sequences of the clients, we extract the communication characteristics of each client with the Ethernet/IP protocol in terms of the four types discussed in Section 4.1 and provide one example of each type with the packet captured. The type of sequence in Ethernet/IP is related to the class ID, instance ID, attribute ID, and service code. Figure 19 shows the captured sequences (screenshot from Wireshark) generated by different clients.

(a) Independent Sequence. Most classes (such as Class 0x01, and 0x69) and instances used according to a specific service code (such as 0x03, and 0x52) in Ethernet/IP are independent and thus the client can access any class by an independent request packet. This characteristic exists in the communication interaction of all the clients. Except for RSLogix 5000, all the independent sequences can be regarded as a list of correlative sequences in the configuration process because the configuration downloading process cannot be segmented into separated steps.

(b) Correlative Sequence. This sequence signature can be regarded as a list of service codes and classes with an independent relationship combined compulsively by the client or the manufacturer customized service codes combination. An example of the latter case is illustrated in Figure 19(a). This sequence is the process of authentication between RSLogix 5000 and 1756-ENBT/A, and the service codes (0x5C, 0x4B, 0x4C) and class ID (0x8E, 0x64) used in this process are also customized by the manufacturer. Thus this sequence can be regarded as a signature of the RSLogix 5000 application.

(c) Cyclic Sequence. If the client needs to keep the linkage with the PLC controller, it would acquire the status of the PLC cyclically. Figure 19(b) shows RSLogix 5000 requiring the status of 1756-L62 using the service code of a multiple service request (0x0A) and multiple class IDs (0x01, 0x69, 0xAC, 0x70, etc.) and the process is repeated constantly until we quit the application. However, the other clients do not need to maintain the connection and do not have this sequence signature.

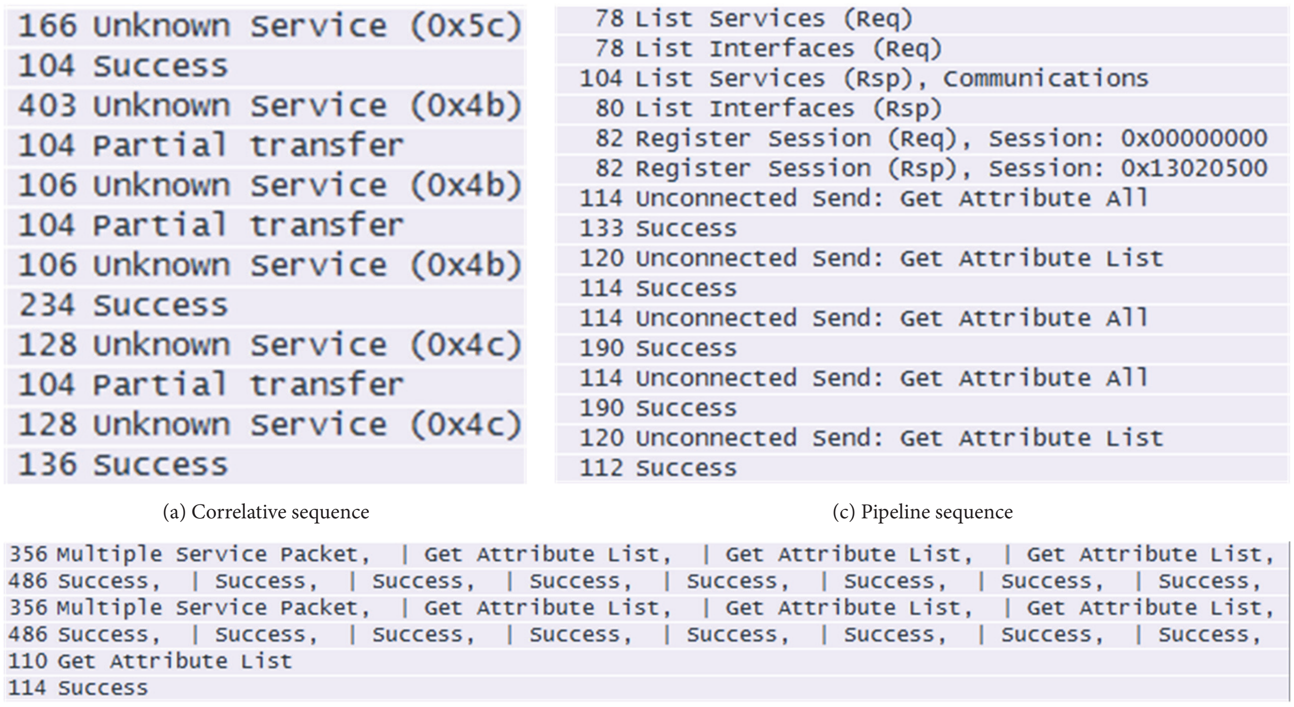
(d) Pipeline Sequence. Figure 19(c) also shows a case of the RSLogix 5000 signature. The client sends two requests consecutively to start the process of communication. However, the other clients would not respond. Another special type of pipeline sequence in the Ethernet/IP protocol is a multiple service request with the code 0x0A, which is implemented to process a few service requests simultaneously by RSLogix 5000.

Based on the classification of the sequence signatures belonging to different clients, we summarize the minimum set of each type of signature with the specific client in Table 4 and provide examples of the signatures in Table 5. The basic principle is that there is no intersection of signatures among the clients.

7.2.2. Experiment Results. Based on the general communication process of the Ethernet/IP protocol, we constructed the baseline SF-FSM in the monitoring system and extended the SF-FSM with the states that are exclusive for each client according to the respective sequence signature. The key chains of service codes, class IDs, instance IDs, and attribute IDs in the signature sequence packets were also constructed and linked with the state in the SF-FSM.

The experiment network architecture is deployed as shown in Figure 18. The IP address of the PLC (1756-L62) is 192.168.0.1.

We use the three legitimate clients listed in Table 3 to execute the operation of reading an attribute defined in the configuration of the PLC. The IP addresses of the three clients are 192.168.0.2, 192.168.0.3, and 192.168.0.4, respectively.



```
166 Unknown Service (0x5c)
104 Success
403 Unknown Service (0x4b)
104 Partial transfer
106 Unknown Service (0x4b)
104 Partial transfer
106 Unknown Service (0x4b)
234 Success
128 Unknown Service (0x4c)
104 Partial transfer
128 Unknown Service (0x4c)
136 Success
```

(a) Correlative sequence

```
 78 List Services (Req)
 78 List Interfaces (Req)
104 List Services (Rsp), Communications
 80 List Interfaces (Rsp)
 82 Register Session (Req), Session: 0x00000000
 82 Register Session (Rsp), Session: 0x13020500
114 Unconnected Send: Get Attribute All
133 Success
120 Unconnected Send: Get Attribute List
114 Success
114 Unconnected Send: Get Attribute All
190 Success
114 Unconnected Send: Get Attribute All
190 Success
120 Unconnected Send: Get Attribute List
112 Success
```

(c) Pipeline sequence

```
356 Multiple Service Packet,  | Get Attribute List,  | Get Attribute List,  | Get Attribute List,
486 Success,  | Success,  | Success,  | Success,  | Success,  | Success,  | Success,  | Success,
356 Multiple Service Packet,  | Get Attribute List,  | Get Attribute List,  | Get Attribute List,
486 Success,  | Success,  | Success,  | Success,  | Success,  | Success,  | Success,  | Success,
110 Get Attribute List
114 Success
```

(b) Cyclic sequence

Figure 19: Sequence types in Ethernet/IP.

Table 4: Sequence signatures statistics of clients.

Type of sequence signature	RSLogix 5000	Molex EIP_Tools	EPICS Tools
Independent sequence signature	**0**	**1**	**1**
Correlative sequence signature	**3**	**2**	**1**
Cyclic sequence signature	**2**	**0**	**0**
Pipeline sequence signature	**1**	**0**	**0**

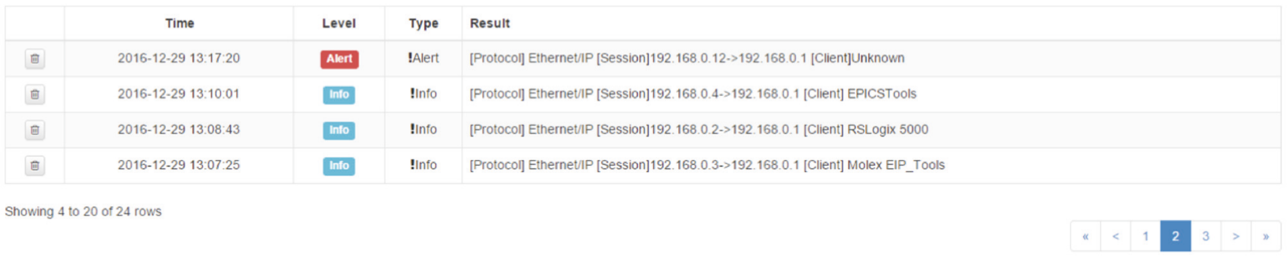
One malicious client developed by Visual studio 2010 was installed on the host with IP address 192.168.0.12. The malicious client was assumed to be used by the attacker who intended to write the attribute value in the configuration of the PLC. The attribute altered by the attacker is used for alias tag definition in 1756-L62. If the alias tag is renamed and quoted by the following code text, the attack could manipulate the desired control point of the physical equipment without affecting the surveillance point, which might be displayed on the HMI, so as to implement a stealth attack.

All the data flows of each client corresponding to the operations were captured and analyzed by the monitoring system; this system then output the results of client identification and anomaly detection.

Table 5: Examples of signatures.

Type of sequence signature	Examples	Client belonging to
Independent sequence signature	List Identity (Req)	Molex EIP_Tools
Correlative sequence signature	Customized Service 0x5C (Req) Customized Service 0x4B (Req) Customized Service 0x4B (Req) Customized Service 0x4B (Req) Customized Service 0x4C (Req) Customized Service 0x4C (Req)	RSLogix 5000
Cyclic sequence signature	Multiple Service Packet, | Get Attribute List	RSLogix 5000
Pipeline sequence signature	List Services (Req) List Interfaces (Req)	RSLogix 5000

(1) Effectiveness Analysis. The monitoring results are shown in Figure 20 in the form of logs. Three "Info" level logs were maintained to indicate the legitimate clients, which were marked by the type of application software. These "Info" level logs can be regarded as the results of the scenario reconstruction process. In this experiment, we mainly analyzed the



	Time	Level	Type	Result
	2016-12-29 13:17:20	Alert	!Alert	[Protocol] Ethernet/IP [Session]192.168.0.12->192.168.0.1 [Client]Unknown
	2016-12-29 13:10:01	Info	!Info	[Protocol] Ethernet/IP [Session]192.168.0.4->192.168.0.1 [Client] EPICSTools
	2016-12-29 13:08:43	Info	!Info	[Protocol] Ethernet/IP [Session]192.168.0.2->192.168.0.1 [Client] RSLogix 5000
	2016-12-29 13:07:25	Info	!Info	[Protocol] Ethernet/IP [Session]192.168.0.3->192.168.0.1 [Client] Molex EIP_Tools

Showing 4 to 20 of 24 rows

Figure 20: Monitoring results.

client side, so the logs from the server side (namely PLC in the current circumstance) information were filtered out.

One "Alert" level log was maintained to indicate the malicious client, which was marked by unknown type.

The judgement principle of various clients is if all the sequence signatures belonging to each client as shown in Table 4 is hit by the communication session established by client and PLC server.

If none of the sequence signatures is hit by the session, the client can be marked with unknown client. If one or more but not all of the sequence signatures belonging to one client are hit by the session, the client can be marked with unknown client. If two or more sequence signatures belonging to different clients are hit by the session, the client can be marked with unknown client.

The matching hit of each signature is not displayed on the user interface of the monitoring system, but we can access the results through the background interface. We can ensure that the client with IP address 192.168.0.2 hits all the six sequence signatures of RSLogix 5000, the client with IP address 192.168.0.3 hits all the three sequence signatures of Molex EIP_Tools, and the client with IP address 192.168.0.4 hits all the two sequence signatures of EPICSTools. The client with IP address 192.168.0.12, on the other hand, hits two sequence signatures of RSLogix 5000 and one sequence signature of Molex EIP_Tools.

(2) Performance Analysis. The performance of the monitoring system can be measured from two aspects. One is the packet matching efficiency. The other is the client identification efficiency.

The packet matching efficiency is usually evaluated by the average time of packet handling. However, in this experiment, the packet generation rate by the clients is not very high and far below the handling rate of the monitoring system. Therefore, we did not achieve the average time, but we could achieve the average packet handling delay. We added an interface of the monitoring system to output the packets already handled. Then, we employed a PC to capture the input and output packets simultaneously and calculate each packet handling delay. An average delay of 0.132 ms was achieved.

The client identification efficiency depends on the clients' communication execution period. Once the session is finished or all the sequence signatures of one client are hit, the identification result can be generated immediately with the type of matched client. If the sequence signatures of two or more clients are hit, the identification result can also be generated immediately with the type of unknown client.

8. Conclusion

This paper proposed a state fusion finite state machine model in terms of the analysis of the characteristics of industrial data flow. In addition, a packet matching approach to expose the risks of a specific target industrial control system by identifying vulnerabilities and anomalous behavior was also proposed. A framework for the reconstruction of security scenarios and anomaly detection using the SF-FSM model was also conceptualized. A verification system was also developed to demonstrate the examples using the proposed methods in this paper.

In future studies, we aim to attach a larger number of attributes to the state of SF-FSM to promote the accuracy and efficiency of the packet sequence and propose an approach to address the prevention of risks. Further, the storage space of the knowledge libraries should also be considered to conserve hardware resources in the form of embedded systems in practical applications.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work is supported by National Natural Science Foundation of China under Grant 61433006.

References

[1] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 277–293, 2013.

[2] H. M. Leith and J. W. Piper, "Identification and application of security measures for petrochemical industrial control systems," *Journal of Loss Prevention in the Process Industries*, vol. 26, no. 6, pp. 982–993, 2013.

[3] https://ics-cert.us-cert.gov/content/overview-cyber-vulnerabilities.

[4] https://www.codeproject.com/KB/web-security/StuxnetMalware/Stuxnet_Malware_Analysis_Paper.pdf.

[5] K. Stouffer, J. Falco, and K. Scarfone, "GUIDE to industrial control systems (ICS) security," *NIST Special Publication*, vol. 800, no. 82, p. 16, 2011.

[6] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry, "Challenges for securing cyber physical systems," in *Proceedings of the Workshop on Future Directions in Cyber-Physical Systems Security*, p. 5, Newark, NJ, USA, July 2009.

[7] R. Leszczyna, "Approaching secure industrial control systems," *IET Information Security*, vol. 9, no. 1, pp. 81–89, 2015.

[8] INL, *Common Cyber Security Vulnerabilities Observed in Control System Assessments by the INL NSTB Program*, Idaho National Laboratory, Idaho Falls, Idaho, USA, 2008.

[9] R. R. R. Barbosa, R. Sadre, and A. Pras, "Flow whitelisting in SCADA networks," *International Journal of Critical Infrastructure Protection*, vol. 6, no. 3-4, pp. 150–158, 2013.

[10] N. Sayegh, A. Chehab, I. H. Elhajj, and A. Kayssi, "Internal security attacks on SCADA systems," in *Proceedings of the 3rd International Conference on Communications and Information Technology (ICCIT '13)*, pp. 22–27, IEEE, Beirut, Lebanon, June 2013.

[11] T. Vollmer, M. Manic, and O. Linda, "Autonomic intelligent cyber-sensor to support industrial control network awareness," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1647–1658, 2014.

[12] L. Wang, S. Ren, B. Korel, K. A. Kwiat, and E. Salerno, "Improving system reliability against rational attacks under given resources," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 44, no. 4, pp. 446–456, 2014.

[13] C.-W. Ten, G. Manimaran, and C.-C. Liu, "Cybersecurity for critical infrastructures: attack and defense modeling," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans*, vol. 40, no. 4, pp. 853–865, 2010.

[14] H. Janicke, A. Nicholson, S. Webber, and A. Cau, "Runtime-monitoring for industrial control systems," *Electronics*, vol. 4, no. 4, pp. 995–1017, 2015.

[15] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: techniques, systems and challenges," *Computers and Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[16] B. Galloway and G. P. Hancke, "Introduction to industrial control networks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 2, pp. 860–880, 2013.

[17] G. Stoneburner, A. Y. Goguen, and A. Feringa, "Risk management guide for information technology systems," Tech. Rep. SP 800-30, National Institute of Standards & Technology, Gaithersburg, Md, USA, 2002.

[18] S.-J. Kim, D.-E. Cho, and S.-S. Yeo, "Secure model against APT in m-connected SCADA network," *International Journal of Distributed Sensor Networks*, vol. 10, no. 6, Article ID 594652, 2014.

[19] N. Sayegh, I. H. Elhajj, A. Kayssi, and A. Chehab, "SCADA Intrusion Detection System based on temporal behavior of frequent patterns," in *Proceedings of the 17th IEEE Mediterranean Electrotechnical Conference (MELECON '14)*, pp. 432–438, IEEE, Beirut, Lebanon, April 2014.

[20] T. Nelso and M. Chaffin, "Common cybersecurity vulnerabilities in industrial control systems. Control Systems Security Program," Tech. Rep., Idaho National Laboratory, 2011, https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/DHS_Common_Cybersecurity_Vulnerabilities_ICS_2010.pdf.

[21] http://www.molex.com/mx_upload/superfamily/iccc/EtherNet_IPTool.html.

[22] https://github.com/EPICSTools/ether_ip.

[23] J.-D. Decotignie, "Ethernet-based real-time and industrial communications," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, 2005.

[24] A. Mahboob and J. A. Zubairi, "Securing SCADA systems with open source software," in *Proceedings of the 10th International Conference on High Capacity Optical Networks and Emerging/Enabling Technologies (HONET-CNS '13)*, pp. 193–198, December 2013.

[25] V. Jyothsna, V. V. Rama Prasad, and K. Munivara Prasad, "A review of anomaly based intrusion detection systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26–35, 2011.

[26] http://www.ntop.org/.

[27] https://cve.mitre.org.

[28] https://ics-cert.us-cert.gov.

[29] https://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00213R0_EtherNetIP_Developers_Guide.pdf.