SDNVD-SCADA: A Formalized Vulnerability Detection Platform in SDN-Enabled SCADA System

Jinjing Zhao[1](✉), Ling Pang[1], and Bai Lin[2]

[1] National Key Laboratory of Science and Technology on Information System Security, Beijing, China
zhjj0420@126.com, stissl@163.com
[2] Beijing Institute of System Engineering, Beijing, China
linbaipeking@126.com

Abstract. After the Stunex event in 2010, the security problems of SCADA reveal to the public, which abstract more and more researchers to design new security firms to address the security problems of SCADA. Especially, after the software defined network (SDN) arose, it has become a beneficial attempt to improve the SCADA security. In this paper, a formalized vulnerability detection platform named SDNVD-SCADA is presented based on the SDN technology, which can be used to find the most familiar vulnerabilities in SCADA design, implementation, deployment and action processes. A general security mechanism description language and a SCADA vulnerability pattern database are embedded in SDNVD-SCADA to achieve the ambition of automatic vulnerability detection.

Keywords: SCADA · Software defined network · Vulnerability detection

1 Introduction

Supervisory control and data acquisition (SCADA) networks perform critical tasks and provide essential services within critical infrastructure, which be considered to be the backbone of any country. Critical infrastructure, and in particular control systems, require protection from a variety of cyber threats that could compromise their ordinary operation. The impairment of SCADA networks could cause interruption of critical services, process redirection, or manipulation of operational data that could have serious consequences for the population.

In terms of security, SCADA systems have many problems that might cause attacks or security events [1], e.g.:

1) Insecure design and implementation:

- **No hardware authentication:** which makes it easier to connect non-authorized computers to the system.

D. Dong et al. (Eds.): ACA 2020, CCIS 1256, pp. 3–15, 2020.
https://doi.org/10.1007/978-981-15-8135-9_1

- **No access credentials:** several systems do not use access credentials, which means their security really only on the belief that no one (non-authorized) will get virtual or physical access to them.
- **No individual access credentials**: some systems only allow the setting of general passwords that quickly become known by too many people.
- **Uncontrolled access:** access limitations in control software are often not used.
- **Anonymous access allowed:** services like Telnet and FTP often allow for anonymous login.

2) **No system patching:** on the last years several efforts were made to improve the policy on patch management and from these efforts resulted some standards [2]. Nevertheless, once systems go into production, they will likely never be patched, due to the impossibility of having downtime on the production line, the fear that systems may become unstable, have limitations, lack support/updates by the vendor, among others.
3) **External network connections:** nowadays almost all the SCADA systems are, directly or indirectly, connected to external networks like internet, however it is often believed they are completely isolated from the outside world. This means that numerous connections are uncontrolled.

Many security firms have started designing solutions to address security problems of SCADA systems from different aspects. But there still has no efficient way to find its vulnerabilities automatically and correctly, because getting the security mechanisms of each entity and the whole network running state on time and on purpose are very hard to implement. But they are the fundamental factors of SCADA system security analysis.

SDN is an architecture that decouples forwarding functions (data plane) and network control (control plane), with the aim of introducing direct programmability into the network, to applications and policy engines alike. In recent years, many creative works have been done to deploy SDN in SCADA system [2, 6, 13]. The SCADA masters can also be operated as the SDN controllers or connect to a openflow switch just like the PLC. The network architecture can be designed as Fig. 1. With the help of SDN technology, the controllers can get all the entity states, including the security mechanisms and the traffic information of interactions between entities inside or outside the SCADA system.

In this paper, a formalized vulnerability detection platform named SDNVD-SCADA is presented based on the SDN technology, which can be used to find the most familiar vulnerabilities in SCADA design, implementation, deployment and action processes. In short, our paper makes the following contributions:

1) We propose the first SDN-enabled vulnerability detection platform in SCADA. We also implement a fast prototype on open source SDN controller Floodlight v1.0.
2) We investigate the familiar vulnerability patterns in SCADA and build the SCADA vulnerability database. The relations between these vulnerabilities and possible attacks are also be analyzed.
3) A SCADA security mechanism description language is designed in SDNVD-SCADA platform, which can be used to describe the security mechanisms of different

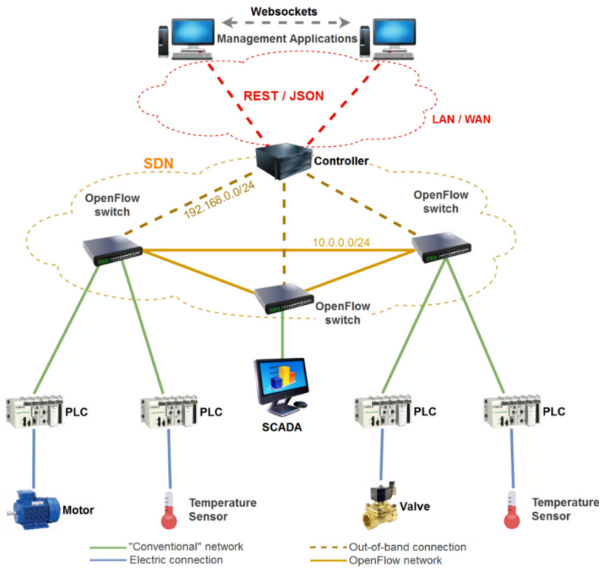

Fig. 1. A case of SDN deployed in SCADA system. The SCADA masters can also be operated as the SDN controllers or connect to a openflow switch just like the PLC.

entities based on their interaction modes. This makes the standardized and automated vulnerabilities analysis possible.

The remainder of this paper is organized as follows. The related works on SCADA security using SDN technology are listed in Sect. 2. The vulnerability detection method is presented in Sect. 3 with the introduction of SDNVD-SCADA architecture. The SDNVD-SCADA implementation is described in Sect. 4, including the SCADA security mechanism description language, SCADA vulnerability pattern database and SCADA vulnerability detector. An example is shown in Sect. 5 to illuminate how to use SDNVD-SCADA. Finally, we conclude in Sect. 6 with a brief summary and discussion.

2 Related Works

Using the SDN technology to improve the security of SCADA and ICS (Industrial Control System) is a new and creative attempt in recent years [2–14].

A approach is suggested by Machii et al. [4] as a way to minimize the attack surface by using SDN to dynamically segregate fixed functional groups within the ICS. This strategy reduces the time and spatial exposure to attacks (effectively creating a moving target) and also provides the means to isolate compromised devices.

Also related to dynamic configuration techniques, Chavez et al. [5] present a security solution based on network randomization, which also encompasses an IDS with near real-time reaction capabilities. This network randomization approach assigns new addresses to network devices in a periodic basis or by request, in order to protect them against attacks that rely on knowledge about the ICS topology (such as static device addresses).

Silva et al. [6] also describe a dynamic technique that makes use of SDN to prevent eavesdropping on SCADA networks. The intended goal is to deter attackers from collecting sequential data, which is essential for breaking encryption, identifying patterns, and retrieving useful information from the payload.

Genge et al. [7] propose two distinct SDN-based techniques to mitigate and block ICS cyber-attacks. The first technique, designed for single-domain networks, attempts to mitigate DoS attacks by rerouting traffic, using information from the SDN controller.

Song, Shin, and Choy [8] suggested using honeynets (networks set up with several honeypot devices) together with SDN technologies to detect scouting procedures and collect profiling information about attackers. Despite being a generic proposal, this solution can be easily ported to most ICS infrastructures.

Adrichem et al. [9] present a SDN network failover solution that should reduce the recovery time in multiple topologies. Being the speed of failure detection the main key for improving the recovery time, a short discussion on different failure detecting systems is presented and the best choice is presented in a detailed mode.

In [10], N. Dorsch et al. also describe their algorithms and different approaches based on SDN regarding: the efficiency improvement of network recovery time in case of link failure; the real-time processing of messages through the implementation of QoS mechanisms, based on the flexibility of SDN networks.

Rui Miguel's master degree dissertation [12] addresses the absence of proper management and security policies problems to improve SCADA ICS manageability, availability and security based on synergies between SDN and ICS domain.

In thesis [14], an SDN-assisted middleware is designed and implemented with open source platforms Open Network Operating System (ONOS) and Mininet, which not only enables real-time information exchange between two SCADA control centers but also supports multiple-to-multiple communications simultaneously.

3 SDNVD-SCADA Architecture

In the SDN-enabled SCADA system, SDNVD-SCADA can be installed on the SDN controller. Every important entity in SCADA which may influence the system security, should describe its security mechanisms and report them to the SDN controller by openflow switch. Based on the common description of security mechanisms that the SCADA entities submitted, SDNVD-SCADA dissects them from the global view and utilizes the FSM to discover the vulnerabilities caused by the absence of some security mechanisms. The inputs of SDNVD-SCADA are the standard description of entity security mechanisms and the network running states, while the outputs are variety of potential vulnerabilities. The design of SDNVD-SCADA architecture is shown in Fig. 2.

In order to describe the security mechanisms of entities formally, SDNVD-SCADA proposes a uniform description language, which is a high level abstraction of entity security mechanism. The entity security mechanism is divided into three levels: system-level, Internet-level and operation level. Based on the study of various SCADA vulnerability cases and data, SCADA vulnerability pattern database defines the vulnerability patterns as the missing the confidentiality, integrity or availability of the critical resources in

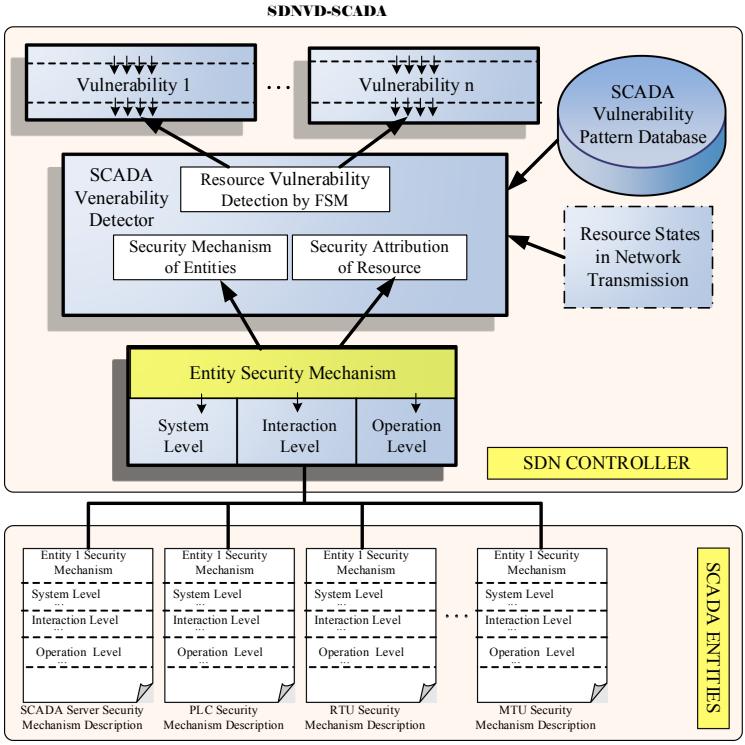

Fig. 2. The design of SDNVD-SCADA architecture, which can be separated into the controller part and entities part.

SCADA. The key characteristics of vulnerability are represented with the vulnerability name, triggered position and possible attack. In addition, SDNVD-SCADA sets the security sensitive resources as the analysis objects, extracts their security attributes from the SCADA security mechanisms of related entities, and build their finite-state machines (FSM) on their security states in the network transmission process. If the state of resource matches some items in the SCADA vulnerability pattern database, and the resource is sensitive to this vulnerability patterns, SDNVD-SCADA will deduce that the behavior of SCADA entities may cause a vulnerability.

The time cost of the process is composed of two parts. The one is the time of resources information generation, which is O(|Entity|) and the |Entity| is the entity number in SCADA which has the sensitive resources or has the right to read and write the resources; the other is the time cost on building the FSMs of all these entities. One FSM building time overhead is O(lg|VMD|), which is the course of searching and matching process in the SCADA vulnerability pattern database using the binary search algorithm. Consequently, the time overhead of all the resources FSM building is O(|R|*lg|VMD|). Thus, the entire time cost of the vulnerability detection is max (O(|Entity|), O(|R|*lg|VMD|)).

4 SDNVD-SCADA Implementation

In order to implement the SDNVD-SCADA system, the most important thing is how to solve the formalization problems to let the detection process automatically. Therefore, a SCADA security mechanism description language is designed to describe the security mechanism of different entities in SCADA system. A SCADA vulnerability pattern description syntax is described to construct the vulnerability pattern database. And the corresponding SCADA vulnerability detector is presented to make the detection automatically.

4.1 SCADA Security Mechanism Description Language

SDNVD-SCADA provides a formalized approach to describe the security mechanism of different entities at various levels. Thus, SDNVD-SCADA provides a standardized and formalized language to analyze SCADA vulnerabilities.

The language provides the capability to describe the class specification in BNF paradigm syntax with XML format recursively. So, various types of security mechanisms at all levels can find the appropriate location in the SDNVD-SCADA framework. By language decomposes its complexity and highlighting the most essential features, which can effectively improve the analysis capabilities of SDNVD-SCADA model and establish the automated analysis foundation for SCADA vulnerability detection.

The entity security mechanism is divided into three levels: system-level, Internet-level and operation level. Based on the study of various SCADA vulnerability cases and data, we define the vulnerability patterns as the missing of resources confidentiality, integrity and availability.

Definition 1. X denotes an entity set, and R is a resource. If the entity in X cannot access the information on R, then define R for X is **confidential**.

Definition 2. Set X the aggregation of entities, and R is a resource. If all members in X trust R, then R has **integrity** for X.

Definition 3. Set X the aggregation of entities, and R is a resource. If all members of X can access R, then R for X with the **availability**.

System security mechanism involves confidentiality, integrity and availability. Confidentiality means that the security mechanism should prevent the resources from leaking to unauthorized entity. Integrity indicates that the security mechanisms must specify the authorized entities that can modify the resources. The security mechanism to describe the conditions and modes of the resource changes is called integrity strategy. Availability refers to that the security mechanism should describe the resources which must be provided. It defines the resources parameters and the access range.

SCADA security mechanism description language defines some basic elements in SCADA interaction, as shown in Table 1. The main contents are divided into two categories: the one is the basic definitions related to the interactive entities and processes; the other is the basic definitions of security mechanisms, which include resource access privileges, resource security features, and its security patterns.

Table 1. The elements definition of SCADA security mechanism description language

Elements Definition	
Entity ID	*Entity identification*
Duration	*Duration for an interaction process*
ResName	*Resource identification*
Data	*Data type of resource*
Information	*Information type of resource*
Service	*Service type of resource*
Confidentiality Sensitive	*Be sensitive for confidentiality*
Confidentiality Insensitive	*Be insensitive for confidentiality*
Integrity Sensitive	*Be sensitive for integrity*
Integrity Insensitive	*Be insensitive for integrity*
Availability Sensitive	*Be sensitive for availability*
Availability Insensitive	*Be insensitive for availability*
t_Readable	*Resource readable*
t_Writable	*Resource writable*
t_Executable	*Resource executable*
Authorization Pattern	*Entity authorization pattern, including Central pattern and distributed pattern, etc.*
Encryption Pattern	*Entity encryption pattern, including symmetrical pattern and dissymmetrical pattern, etc.*
Authentication Pattern	*Entity authentication pattern, including certification pattern and password pattern, etc.*

With the syntax definition above, SDNVD-SCADA provides a specification to describe the security mechanisms at different levels by dint of the BNF description methods. In BNF, "::=" denotes "definition", "|" means "or", angle brackets "<>" refers to a non-terminal symbol. The so-called non-terminal symbol is some abstract concept in language, and the terminal symbol is that can directly appear in the language.

Table 2 lists the descriptions for some security mechanisms. System level mechanism is mainly concerned about the security mechanisms used by the operating system. Interaction level mechanism is focused on the network layer and application inter-connection layer. Operation level security mechanisms is mainly used to describe the security mechanisms in business processes, including the information of released resources and its authorization, encryption and other information. If there is a new security mechanism, it can be added according to the syntax specifications.

4.2 SCADA Vulnerability Pattern Description

While the manifestations of vulnerabilities are very wide, their fundamental connotation just refers to that the confidentiality, integrity or availability of a resource is compromised. Thus, from the perspective of resource state, a vulnerability can be denoted as follows:

$$<\text{Vulnerability}> ::= R_{\neg Confidentiality|\neg Integrity|\neg Availability}$$

Table 2. The syntax of SCADA security mechanism description language

Syntax of SCADA Security Mechanism Description
<Security Mechanism> ::= <System Level Mechanism> | <Interaction Level Mechanism> | <Operation Level Mechanism>
<System Level Mechanism> ::= Entity ID | <Neighbor> | Authentication Pattern | Authorization Pattern | Encryption Pattern||...
*<Neighbor> ::= <Entity>**
<Entity> ::= PLC | MTU|RTU | Master|Switch Node|External Node…
<Interaction Level Mechanism> ::= <Interaction List>|...
*<Interaction List> ::= <Interaction>**
<Interaction> ::= <Src> <Des> <Path> <Msg> <Event>Duration
<Src> ::= PLC | MTU|RTU | Master|...
<Des> ::= PLC | MTU|RTU| Master|...
<Path> ::= <Src> Swith Node <Des>*
<Msg> ::= ResName+
<Event> ::= Read |Write | Copy| Cut | Create |Delete|...
<Operation Level Mechanism> ::= <Resource List>|<Authorization List>|...
*<Resource List> ::= <Resource >**
<Resource> ::= ResName <ResType> <ResFeather>
<ResType> ::= Data | Information | Service |...
<ResFeather> ::= Confidentiality Sensitive| Confidentiality Insensitive | Integrity Sensitive |Integrity Insensitive |Availability Sensitive | Availability Insensitive
<Authorization List> ::= <AuObject> <AuResource> <AuType>
*<AuObject> ::= Entity**
<AuResource> ::= ResName
<AuType> ::= R | W | E | RW| RE| WE| RWE
<R> ::= t_Readable
<W> ::= t_Writable
<E> ::= t_Executable

The SCADA vulnerability pattern in the database can be expressed as:

$$E \wedge \neg B \rightarrow V \rightarrow \mathrm{A}$$

It indicates that when the event E occurs, if the security mechanism B is invalid or missing, the vulnerability V will appear which may trigger attacks A. The event E is an interactive process, and is represented with a quad-ruple *f (Src, Des, ResName, Action)* to abstract the key properties of a resource. Action includes the actions of read, write, copy, cut, create, delete, and so on. When a resources is identification or encryption key, Action can include the identity and key creation, distribution and destroy. For example, if a user *X* requests resource R from server *Y*, event *E* can be denoted as *E = [X, Y, R, Read]*.

Security mechanism *B* is applied to the resource *ResName*. As the entity interaction happens between *Src* and *Des*, *B* can be expressed by a triple consisting of resources, methods, and entities, i.e. *B = g (Entity, ResName, Mechanism)*, *Entity∈[Src, Des]*, wherein mechanism may cover authorization, encryption, authentication, etc. If the

resource *ResName* is identity, key or other special types, *Mechanism* can include the key length, identity and encryption mode. As a demonstration, the server *Y* provides authorization mechanism *B* to its resource *R*, which can be expressed as *B = [Y, R, authorization]*.

The vulnerability *V* is denoted as seven kinds of forms enumerated as follows:

V = [(¬Confidentiality, Integrity, Availability), (Confidentiality, ¬Integrity, Availability), (Confidentiality, Integrity, ¬Availability), (¬Confidentiality, ¬Integrity, Availability), (¬Confidentiality, Integrity, ¬Availability), (Confidentiality, ¬Integrity, ¬Availability), (¬Confidentiality, ¬Integrity, ¬Availability)].

Attack *A* which may be incurred by vulnerability *V* includes all SCADA possible attacks, such as Sybil attack, denial of service attack, middle-person attack, etc.

For example, with the definitions above, when a client accesses the resources published by a server, if the server does not provide the authorization mechanism for resource accessing, the resources confidentiality may be compromised. This scenario can be represented in the following manner:

$$V1 : [USER,\ RP,\ R,\ \mathrm{Re}ad] \wedge \neg[RP,\ R,\ authorization] \rightarrow \neg Confidentiality \rightarrow Information\ leakage$$

When the server's resources are to be back upped to other servers, if there is no identity authentication for these servers, the resources integrity may not be guaranteed and Sybil attack perhaps takes place. This can be described with the following expression:

$$V2 : [RP1, RP2,\ R,\ Copy] \wedge \neg[RP1,\ R,\ ID\ Authentication] \rightarrow \neg Integrity \rightarrow Sybil\ Attack$$

4.3 SCADA Vulnerability Detector

In order to characterize the resource states changed with the SCADA entity interaction process, SDNVD-SCADA introduces a finite-state machine $M = (Q, \sum, \Delta, \delta, \lambda, q0)$, wherein:

- $Q = \{q0, q1, \ldots, q8\}$ is the collection of resource security state;
- $\Sigma = \{\sigma 0, \sigma 1, \ldots, \sigma n\}$ is the collection of input events;
- $\Delta = \{a0, a1, \ldots, an\}$ is the collection of output events;
- δ: $Qx\ \Sigma \rightarrow 2^Q$ *is* the state transformer function;
- λ: $Qx\ \Sigma \rightarrow \Delta$ is the output function;
- $q_0 \in Q$ is the resource initiate security state.

$Q = V \vee (Confidentiality, Integrity, Availability)$.
Σ and Δ are the collection of events E.
δ: $Q \times \sum \rightarrow 2^Q$ is the state transformer function,
λ: $Q \times \sum \rightarrow \Delta$ is the output function.

These two functions provide the basis rules for resources security state transformation. The definitions of δ and λ refer to the SCADA vulnerability pattern database. If the input events and the resource security mechanisms will lead to a vulnerability, LDS-IVDM will work on the resource security attributes of furthermore. If the resource is sensitive to this vulnerability, the security state of resource will switch to a fragile state; otherwise, the resource security state will remain, namely:

$$\delta, \lambda(\sigma_i, q_i) = \begin{cases} v, v \in V, & if\ \sigma_i \in E \wedge SecurityMechanism\ \in \neg B \wedge v \notin \text{Res}Feather \\ else & (Confidentiality, Integrity, Availability) \end{cases}$$

5 Example

In this section, we will show how to use SDNVD-SCADA to find vulnerabilities in SCADA by a typical example. Figure 3 exhibits the network topology of a SDN-enabled SCADA system used in the power grid. A SDN controller is deployed to control all the information transmission in the network. The entities in the SCADA include the OPC Server, PLC, MTU, RTU, Interface terminal etc. They transmit the control and data information between each other. For example, RTU uploads its collected information *Ri* about the power nodes to the OPC Server, PLC node read the industrial data *Ra* from the OPC Server, and PLC node transfer the control command *Rc* to the OPC Server.

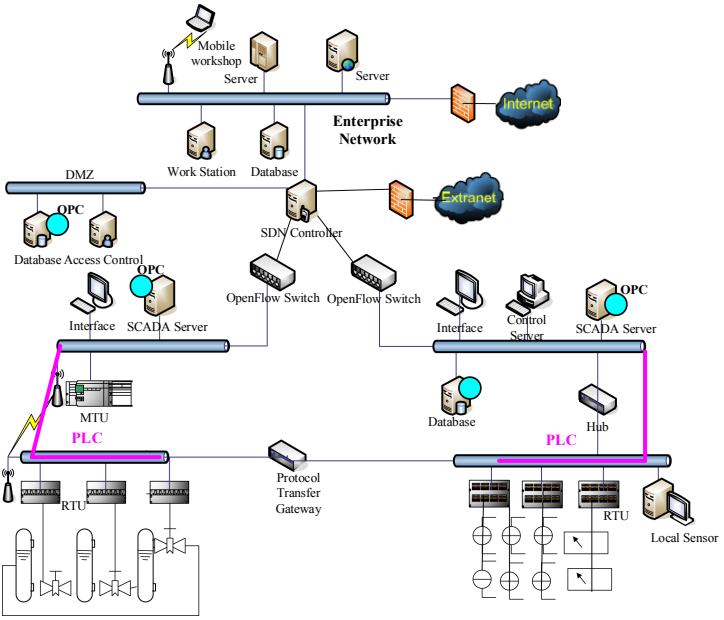

Fig. 3. Network topology of a typical SDN-enabled SCADA system used in the example. A SDN controller is deployed to control all the information transmission in the network. The entities in the SCADA include the OPC Server, PLC, MTU, RTU, Interface terminal etc.

The resource *Ri, Ra* and *Rc* are all sensitive the confidentiality, integrity and availability. Any action that may destroy the C-I-A of them is a vulnerability to the SCADA system. According to the applicable reality, *Ri* is a part of *Ra*, because OPC Server collect the *Ri* from all RTUs to construct the *Ra*. That means *Ri* has the same security feature of *Ra* (Fig. 4).

RTU Security Mechanism

```
<RTU Security Mechanism>
<System Level>
    <PersonalID ID="RTU",IDType="local">
    <NeighborID ID="OPC Server">
</System  Level>
<Interaction Level>
    <Interaction List>
    <I1 Src="RTU", Des="OPC Server",
Path="[RTU,OPC Server]", Msg="Ri", Event
="Write", Time="10">
    </Interaction List>
</Interaction Level>
<Operation Level>
     <Resource List>
     <Re1 ResName="Ri",ResType="Information",
ResFeather="Default">
     </Resource List>
</Operation  Level>
</RTU Security Mechanism>
```

PLC Security Mechanism

```
<PLC Security Mechanism>
<System Level>
    <PersonalID ID="PLC",IDType="local">
</System  Level>
<Interaction Level>
    <Interaction List>
    <I1 Src="PLC", Des="OPC Server",
Path="[PLC...OPC Server]", Msg="RA", Event
="Read", Time="10">
    <I1 Src="PLC", Des="OPC Server",
Path="[PLC...OPC Server]", Msg="RC", Event
="Write", Time="10">
    </Interaction List>
</Interaction Level>
<Operation Level>
     <Resource List>
     <Re1 ResName="RC",ResType="Command",
ResFeather="Default">
     </Resource List>
</Operation  Level>
</PLC Security Mechanism>
```

OPC Server Security Mechanism

```
<OPC Server Security Mechanism>
<System Level>
    <PersonalID ID="OPC Server",IDType="local">
    <NeighborID ID="RTU">
</System  Level>
<Interaction Level>
    <Interaction List>
    <I1 Src="OPC Server", Des="RTU", Path="[OPC Server,RTU]", Msg="RC", Event ="Write", Time="10">
    </Interaction List>
</Interaction Level>
<Operation Level>
     <Resource List>
     <Re1 ResName="RA",ResType="Information",ResFeather="Default">>
     </Resource List>
</Operation  Level>
</OPC Server Security Mechanism>
```

Fig. 4. Security mechanism description of SCADA entities. The security mechanisms of RTU. PLC and OPC are described in SCADA security mechanism description language.

The process proceeding of SDNVD-SCADA is illustrated as follows. Firstly, SDNVD-SCADA pre-installed on the SDN controller analyzes the security mechanism files submitted by RTU, PLC and OPC Server. Secondly, it extracts the related mechanism of *Ra* and *Rc*. At the same time, The SDN controller monitors all the transformation in the network and builds the FSMs of *Ra* and *Rc* security states, which are illustrated

in Fig. 5. The events related to *Ra* and *Rc* include *OPC Server Create Ra, OPC Server Write Ra from RTU, PLC Read Ra from OPC Server, PLC create Rc, PLC Write Rc to OPC Server, OPC Server Write Rc to RTU*. Finally, SDNVD-SCADA detects the possible vulnerabilities and work out the potential attacks according to the part in the corresponding items in SCADA vulnerability pattern database. When the states of *Ra* and *Rc* transfer to the R2 and R3, they might occur a security problem.

1) To the R2 vulnerability state, it may be occurred by the remote anonymous log in, which is be permitted by the default deployment of many SCADA device corporations. So the attackers from the Internet or external network could access the OPC Server CLSID on the OPC Server, and analyze the Controller type and software information of the SCADA system. The confidentiality of the system is broken.
2) To the R3 vulnerability state, it may be occurred by the absence of the encryption mechanism in the OPC communication course. After the attackers connect in the SCADA network, it can listen and analyze the important information, or disguise itself as a OPC client and send the fake packets to the OPC server to destroy the accurate control process of SCADA.

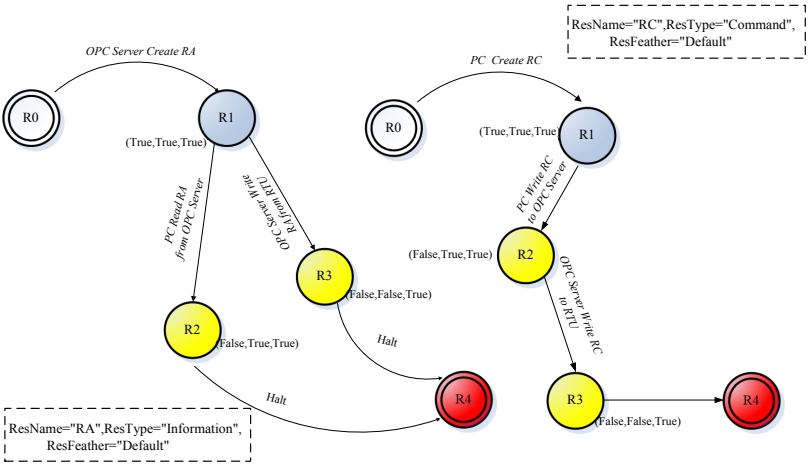

Fig. 5. FSMs of *Ra* and *Rc* Resource Security State. When the states of *Ra* and *Rc* transfer to the R2 and R3, they might occur a security problem.

6 Conclusion

In this paper, we focus on the security problem in SCADA system and propose a new solution on automatic vulnerability detection by adopting the SDN technology. The SDN controller implemented SDNVD-SCADA can centralized analyze the security mechanisms of important entities in SCADA and monitor all the information transferring about the security-sensitive resources in SCADA. By the formalized security mechanism

description language and SCADA vulnerability pattern database, a finite-state machine can be built of each security-sensitive resource, by which the vulnerable state of resource could be detected by SDNVD-SCADA.

The prototype of SDNVD-SCADA is under construction on one of the popular open source SDN controller, Floodlight v1.0. More SCADA vulnerability pattern should be collected and more experiments should be tested later.

References

1. Gresser, C.H.: Hacking SCADA/SAS systems. In: Seminar at Petroleum Safety Authority Norway (2006)
2. Dong, X., Lin, H., Tan, R., Iyer, R.K., Kalbarczyk, Z.: Software-defined networking for smart grid resilience: opportunities and challenges. In: Cyber Physical System Security (CPSS 2015), Singapore (2015)
3. Irfan, N., Mahmud, A.: A novel secure SDN/LTE-based architecture for smart grid. In: Proceedings of the 2015 IEEE International Conference on Computer and Information Technology, pp. 762–769 (2015)
4. Machii, W., et al.: Dynamic zoning based on situational activities for ICS security. In: Proceedings of the 2015 10th Asian Control Conference (ASCC), pp. 1–5 (2015)
5. Chavez, A., Hamlet, J., Lee, E., Martin, M., Stout, W.: Sandia National Laboratories, network randomization and dynamic defense for critical infrastructure systems, Albuquerque, New Mexico and Livermore, California, USA (2015)
6. Silva, E., Knob, L., Wickboldt, J., Gaspary, L., Granville, L., Schaeffer-Filho, A.: Capitalizing on SDN-based SCADA systems: an anti-eavesdropping case-study. In: 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), Ottawa (2015)
7. Genge, B., Haller, P., Beres, A., Sándor, H., Kiss, I.: Using software-defined networking. Securing Cyber-Phys. Syst. 305–329 (2016)
8. Simões, P., Cruz, T., Proença, J., Monteiro, E.: On the use of honeypots for detecting cyber-attacks on industrial control networks. In: Proceedings of the 12th European Conference on Information Warfare and Security, pp. 263–270 (2013)
9. Song, Y., Shin, S., Choi, Y.: Network iron curtain: hide enterprise networks with openflow. In: Kim, Y., Lee, H., Perrig, A. (eds.) WISA 2013. LNCS, vol. 8267, pp. 218–230. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-05149-9_14
10. Adrichem, N., Asten, B.J., Kuipers, F.A.: Fast recovery in software-defined networks. In: EWSDN 2014 Proceedings of the 2014 Third European Workshop on Software, Washington (2014)
11. Dorsch, N., Kurtz, F., Georg, H., Hagerling, C., Wietfeld, C.: Software-defined networking for smart grid communications: applications, challenges and advantages. In: IEEE International Conference on Smart Grid Communications (2014)
12. Queiroz, R.M.C.: Integration of SDN technologies in SCADA Industrial Control Networks. Masters' Degree in Informatics Engineering Dissertation, 15 January 2017
13. Rehmani, M.H., Davy, A.: Software defined networks based smart grid communication: a comprehensive survey. arXiv:1801.04613v4 [cs.NI], 27 March 2019
14. Beibei, L.: A Software-Defined Networking (SDN) Assisted Middleware Interconnecting Supervisory Control and Data Acquisition (SCADA) Systems. Thesis of the Degree Master of Science, Arizona State University, August 2018