Available online at www.sciencedirect.com
ScienceDirect
ICT Express 4 (2018) 1–5
www.elsevier.com/locate/icte

FPGA-based network intrusion detection for IEC 61850-based industrial network

Junsik Kim, Jaehyun Park*

Department of Information and Communication Engineering, Inha University, Incheon, 22212, Republic of Korea

Received 1 December 2017; received in revised form 7 January 2018; accepted 8 January 2018
Available online 1 February 2018

Abstract

This paper proposes an FPGA-based network intrusion detection system for the IEC 61850-based industrial network that is specially designed for substation automation. The proposed system uses the Shift-And algorithm for detecting malicious network packets within IEC 61850 messages. To implement a complex rule matching module with a limited memory size of FPGA, a specially designed rule matching module was proposed in this paper. For feasibility evaluation, a prototype with 265 regular expression matching modules was implemented using Xilinx Zynq-7030 FPGA and its performance is presented in this paper.

Keywords: Network intrusion detection system (NIDS); IEC 61850; Regular expression; Substation automation; FPGA

1. Introduction

Communication between intelligent electric devices within a substation automation system plays an important role in real-time operations of the whole electric power distribution network [1,2]. Considering the real-time control and monitoring of electric power network, IEC 61850 was proposed, which is widely adopted for substation automation [3,4]. Even though IEC 61850 is a key element to achieve interoperability in a modern power network, such as smart grid, adopting IEC 61850 that uses the standard switched Ethernet network also incidentally accompanies cyber security risk due to the vulnerability of IEC 61850 messages [5–7].

To reduce this network security risk, a network intrusion detection system (NIDS), which protects the computer system by analyzing the network packets is widely adopted [8]. NIDS distinguishes the malicious packets among the received network packets based on predefined rules that define the communication pattern of the malicious packets. Snort and Suricata are the most widely used intrusion detection software that utilize user-defined detection rules for deep packet inspection. Even though these software-based NIDS are flexible, packet inspection is performed by the software and the processing speed depends on the performance of the CPU core. This means that the detection speed becomes slower as the complexity of the rules grows. In addition, the NIDS for IEC 61850 network is required to be specially designed after understanding the physical infrastructure to determine whether the observed communication patterns are benign or malicious [9,10].

Several FPGA-based NIDS systems have been proposed in the past decades to achieve higher performance in a real-time control environment, such as in substation automation. In such systems, suspicious packets are filtered using predefined rules that are normally written in regular expression and processed by FPGA after being converted to the finite state machine (FSM). However, implementing a large size FSM generally requires significant memory blocks of FPGA. To improve the efficiency in memory usage, Hieu proposed a structure that combines redundant states into one [11], while Freire proposed a method using Huffman coding [12]. On the other hand, Bando proposed a method to avoid the overlapping issue [13]. Among

* Corresponding author.
E-mail address: jhyun@inha.ac.kr (J. Park).
Peer review under responsibility of The Korean Institute of Communications Information Sciences.

https://doi.org/10.1016/j.icte.2018.01.002

these architectures, the Extended Shift-And algorithm (ESA) is useful for dynamically reconfiguring patterns, such as regular expression matching. Kaneta implemented the ESA using an FPGA [14,15]. However, in their architecture, as the maximum number of states depends on the length of the register, it is challenging to apply a regular expression having many states. In addition, the number of states of FSM rapidly increases by the unrolling of constrained repetition. In a practical NIDS, several repetitions of the detection rules are used, resulting in a very large number of states. To overcome this memory size issue, a modified Extended Shift-And structure with counter module is proposed in this paper.

2. Related work

2.1. Pattern matching method

The regular expression is common language to specify a single pattern string to match a set of strings. It is primarily used for character searches, DNA pattern detection, and deep packet inspection in NIDS software. Table 1 summarizes the regular expressions used to define the packet pattern to be detected by the FPGA-based NIDS engine.

To apply the ESA [14], regular expressions should be converted to non-deterministic finite automata, as defined in Eq. (1).

$$A = (Q, \Sigma, I, F, \Delta) \tag{1}$$

where Q is the entire state set, Σ is all the characters in the regular expression, I is the start state, F is the end state, and Δ is the transition function. The transition function, Δ, determines the next state. When a new character α is received by NFA, the next state is determined according to the transition function as shown in Eq. (2). D is a specific function that determines the next state depending on the input character and the current state.

$$\Delta = \left\{(q, \alpha, q'), q \in Q, \alpha \in \Sigma \cup \{\epsilon\}, q' \in D(q, \alpha)\right\}. \tag{2}$$

NFA can move from one state to several states, thus multiple states can be simultaneously active at the same time. However, to apply the Shift-And algorithm, the NFA should have a property of moving only in one direction.

2.2. GOOSE pattern example

The IEC 61850 standard uses the Generic Object-Oriented Substation Event (GOOSE) message for time critical operations within a substation. As the GOOSE message is encoded in the Application Protocol Data Unit (APDU) of an Ethernet packet, a network attack using the GOOSE message is possible. For example, a regular expression

$$RE = GOOSEID.\{0, 20\}\,[\backslash x50 - \backslash x58] \tag{3}$$

can detect the packets with a TAG number, an octet located at the 21st place after the GOOSEID string, is between 80 and 88. By combining the MAC address of a station, a series of GOOSE messages from a specific station(s) is(are) easily defined using the above regular expressions. If a packet with a tag number 84 is to be transmitted from a device with a specific MAC address, packets with an unacknowledged address can be blocked by comparison with a predefined MAC address. Table 2 shows the transition table of the example of GOOSE message shown in Eq. (3). The items in the table are used for the NFA transition function of the ESA, and the operation of the algorithm is described in the following section.

Table 1
Regular expression description.

Symbol	Description
.	Matches any one character
[,]	It is the same as selecting one among several characters and using multiple |. A range can be specified with the "-" symbol.
(,)	Multiple expressions can be bound together.
*	Zero or more characters
{m, n}	m times or more, and n times or less
?	0 or 1 occurrence
+	More than once
|	Including one of the two

Table 2
Transition table: $RE = GOOSEID.\{0, 20\}[\backslash x50\text{-}\backslash x58]$.

Bit position	1	2	3	4	5	6	7	8	9
Init	1	0	0	0	0	0	0	0	0
Accept	0	0	0	0	0	0	0	0	1
Move[D]	0	0	0	0	0	0	1	1	0
Move[E]	0	0	0	0	1	0	0	1	0
Move[G]	1	0	0	0	0	0	0	1	0
Move[I]	0	0	0	0	0	1	0	1	0
Move[O]	0	1	1	0	0	0	0	1	0
Move[S]	0	0	0	1	0	0	0	1	0
Move[\x50-\x58]	0	0	0	0	0	0	0	1	1
Move[.]	0	0	0	0	0	0	0	1	0
Repeat[.]	0	0	0	0	0	0	0	1	0
START	0	0	0	0	0	0	1	0	0
END	0	0	0	0	0	0	0	1	0
SPACE	0	0	0	0	0	0	1	1	0
CR	0	0	0	0	0	0	0	1	0

2.3. Extended Shift-And algorithm

In this paper, the ESA proposed by Navarro is used to process the regular expression [14]. As the active state is expressed by single bit in ESA, state transitions can be performed through a shift operation. Thus, if the register length is w bits, w consecutive strings can be detected.

The NFA of the algorithm is linear and the state transition proceeds stepwise. When B has the transition conditions of all characters Σ in the memory table of word size m, the input character α is used as the address of the memory. If the current state of the NFA is D, and a new character arrives, the next state D_n is updated by Eq. (4).

$$D_n \leftarrow ((D \ll 1) \mid 0^{m-1}1)\&B\,[\alpha]\,. \tag{4}$$

In Eq. (4), the current state D is compared to $B(q, \alpha)$ after the shift operation to determine the next state. Pattern matching