Traffic-Locality-Based Creation of Flow Whitelists for SCADA Networks

Seungoh Choi, Yeop Chang, Jeong-Han Yun, Woonyon Kim

Chapter 6

TRAFFIC-LOCALITY-BASED CREATION OF FLOW WHITELISTS FOR SCADA NETWORKS

Seungoh Choi, Yeop Chang, Jeong-Han Yun and Woonyon Kim

Abstract The security of supervisory control and data acquisition (SCADA) networks has attracted considerable attention since the discovery of Stuxnet in 2010. Meanwhile, SCADA networks have become increasingly interconnected both locally and remotely. It is, therefore, necessary to develop effective network intrusion detection capabilities. Whitelist-based intrusion detection has become an attractive approach for SCADA networks. However, when analyzing network traffic in SCADA systems, general properties such as TCP handshaking and common ports are insufficient to create flow whitelists. To address the problem, this chapter proposes a methodology for locality-based creation of flow whitelists and conducts experiments to evaluate its effectiveness in seven SCADA systems. The experimental results demonstrate that the methodology generates effective whitelists for deployment in SCADA networks.

Keywords: SCADA networks, intrusion detection, whitelists, traffic locality

1. Introduction

Industrial control systems, especially supervisory control and data acquisition (SCADA) systems, are widely used in the critical infrastructure. SCADA networks were traditionally considered to be safe from cyber threats because they employed proprietary protocols and were generally isolated from information technology (corporate) networks. This situation has changed and, as Stuxnet has demonstrated, a highly protected and carefully isolated control network can be compromised by a simple USB drive. Malware targeting SCADA networks has proliferated in recent years with examples such as Duqu, Flame, Gauss, Mahdi, Shamoon and SkyWiper [8].

According to the U.S. Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) [6], SCADA vulnerabilities and incidents are in-

creasing rapidly. Meanwhile, SCADA networks are becoming interconnected with other networks, including the Internet, to link local and remote sites [7]; this greatly increases the ability of SCADA malware to propagate. Hence, it is necessary to develop effective intrusion detection schemes for SCADA networks used in the critical infrastructure.

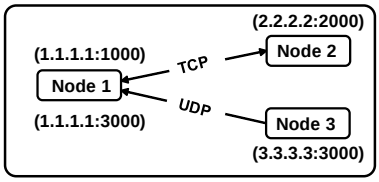
Whitelisting has been recommended for intrusion detection in SCADA networks because of their stable structure and predictable traffic [1, 9]. Several researchers have studied whitelisting and have proposed novel approaches [4, 5, 10]. However, a challenge when designing a whitelisting solution is to identify client-server relationships without requiring information from operators or manufacturers. Some whitelisting solutions (e.g., [4]) assume that the relationships are known; other researchers (e.g., [5, 10]) simply ignore the problem. Moreover, comprehensive evaluations of whitelist generation techniques and their experimental results have not been performed.

Barbosa et al. [2] have developed a state-of-the-art intrusion detection technique for SCADA networks using flow whitelisting. They inferred client-server relationships from TCP handshaking information (TCP flags) and well-known ports. However, when traffic from a real-world SCADA network is analyzed, problems still exist. First, three-way handshaking relationships cannot be identified when TCP sessions are maintained throughout a traffic collection period. Second, different ports numbers may be used for well-known services to enhance security. Third, network flows may not provide adequate operational information because SCADA systems frequently use proprietary protocols for communications and port assignments on devices that may not be publicly documented.

This chapter focuses on flow whitelisting based on the determination of client-server relationships in a SCADA network. A flow whitelist, which corresponds to a permissible access list, comprises IP addresses, port numbers and transport protocol information such as client-server relationships. This chapter describes a methodology for constructing flow whitelists for SCADA networks based on traffic locality, including degree centrality and locally frequently-used ports. The experimental results demonstrate that the whitelists are very effective at securing SCADA networks.

2. Background

This section provides background information. In particular, it discusses two important characteristics of SCADA network traffic flow. These characteristics, degree centrality and locally frequently-used ports, form the basis of the proposed methodology for extracting flow whitelists from SCADA network traffic.



(a) Directed graph.

Src.IP	Src.Port	Dst.IP	Dst.Port	Protocol
1.1.1.1	1000	2.2.2.2	2000	TCP
2.2.2.2	2000	1.1.1.1	1000	TCP
3.3.3.3	3000	1.1.1.1	3000	UDP

(b) Flows as five-tuples.

Figure 1. Flow representation.

2.1 Network Flow and Flow Whitelists

A network flow is a directional edge between two network nodes that is represented by pairs of IP addresses and port numbers. A directional edge identifies the flow protocol and direction of flow.

The directed graph in Figure 1(a) shows network traffic flows: a bidirectional TCP traffic flow between Node 1 and Node 2, and a unidirectional UDP traffic flow from Node 3 to Node 1. A directed graph represents flows using five-tuples (i.e., source IP address, source port number, destination IP address, destination port number, protocol) as shown in Figure 1(b).

A flow whitelist is a set of rules. A rule is also represented as a five-tuple (i.e., source IP address, source port number, destination IP address, destination port number, protocol). However, the source and destination ports are considered to be explicit ports or "ANY."

2.2 Traffic Summary

For the experimental evaluations, SCADA network traffic was captured using mirroring techniques from seven sites in two domains. The collection periods varied from 3 to 29 days. As shown in Table 1, there were distinct distribution differences between the TCP and UDP traffic from each site. A large proportion of traffic did not have many flows for both TCP and UDP communications, such as at Site A. However, Site B had an extremely large proportion of TCP traffic because Site B backed up the other sites using TCP.

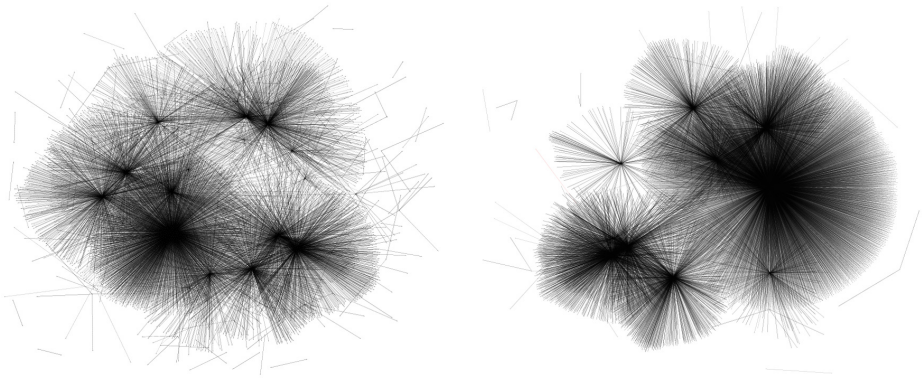
All the flows were analyzed to extract a flow whitelist. The analysis revealed two key characteristics, degree centrality and locally frequently-used ports (LFPs), which are discussed below.

2.3 Degree Centrality

Degree centrality is the number of links associated with a node [3]. Two types of degree centrality can be calculated for directional edges: (i) in-degree flow, which is the number of incoming links; and (ii) out-degree flow, which is the number of outgoing links. As shown in Figure 1, the degree centrality of Node 1 comprises an in-degree flow of two (incoming from Node 2 and Node 3) and an out-degree flow of one (outgoing to Node 2).

Table 1. Summary of captured SCADA network traffic.

Site	Size of Traffic (Proportion)		No. of Flows (Proportion)	
	TCP	**UDP**	**TCP**	**UDP**
A	450 GB	3 GB	27,402,062	25,385,051
	(0.9934)	(0.0066)	(0.5191)	(0.4809)
B	37 GB	0.1 GB	371,908	7
	(0.9973)	(0.0027)	(0.99998)	(0.00002)
C	409 GB	83 GB	4,405,006	4,793,932
	(0.8313)	(0.1687)	(0.4789)	(0.5211)
D	131 GB	76 GB	2,087,740	2,773
	(0.6329)	(0.3671)	(0.9987)	(0.0013)
E	370 GB	55 GB	1,479,011	1,113,938
	(0.8706)	(0.1294)	(0.5704)	(0.4296)
F	2,429 GB	193 GB	47,412,410	26,832,080
	(0.9264)	(0.0736)	(0.6386)	(0.3614)
G	277 GB	3 GB	357,399	24,729
	(0.9893)	(0.0107)	(0.9353)	(0.0647)

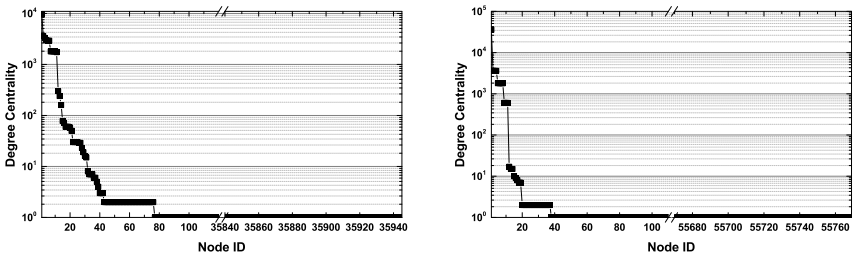


(a) TCP. (b) UDP.

Figure 2. Graphs of network flows during a five-hour period at Site C.

Figure 2 presents graphs of TCP and UDP flows seen during a five-hour period at Site C. Note that most of the flows are concentrated at specific nodes.

Based on this observation, it appears that degree centrality could be used to identify client-server relationships. To help identify the nodes, the results were ranked in descending order of degree centrality as shown in Figure 3. Note that, in the figure, the node ID is a unique IP address and port number pair. Most server nodes (IP_{server}, $PORT_{Server}$) had degree centrality values of more than 1,000, which covered approximately half of the flows for both TCP and UDP.



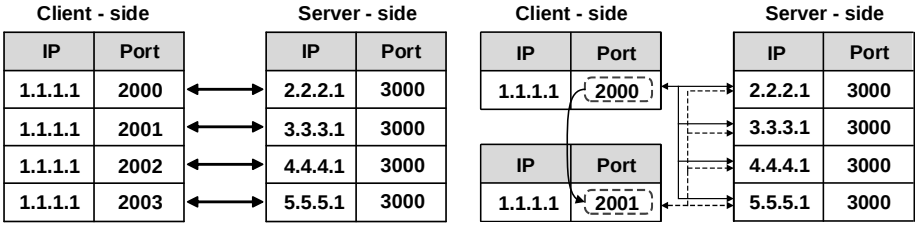
(a) TCP. (b) UDP.

Figure 3. Degree centrality during a five-hour period at Site C.

Upon analyzing the network traffic degree centrality, the fixed port numbers of each IP_{Server} were readily identified. Rules were then extracted from the flows associated with the server nodes. For example, if a server node (IP_{Server}, $PORT_{Server}$) had TCP incoming edges from nodes using the same IP_{Client}, the following rule could be extracted: (IP_{Client}, ANY, IP_{Server}, $PORT_{Server}$, TCP). In the case of TCP, there may be a counter rule, (IP_{Server}, $PORT_{Server}$, IP_{Client}, ANY, TCP), which is similar to the case of UDP.

2.4 Locally Frequently-Used Ports

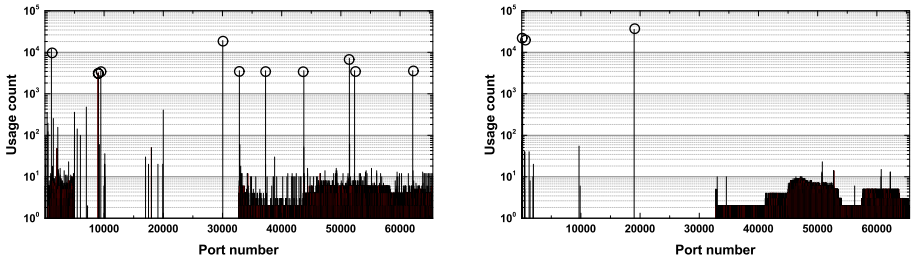
Although degree centrality helps extract rules that cover a considerable amount of flows, it cannot help extract all the rules for a flow whitelist. Moreover, degree centrality may yield incorrect rules in two situations. One is when one client communicates with many servers and the other is when communications occur via the same port.



(a) TCP. (b) UDP.

Figure 4. Limitations of degree centrality.

- **One Client to Many Servers:** A device, such as a front end processor (FEP), requires connections to other devices such as programmable logic controllers (PLCs) or remote terminal units (RTUs) with different IP addresses, but the same port number. The application of degree centrality does not yield rules in this case because no nodes have high degree centrality. As shown in Figure 4(a), TCP flows do not converge on a specific



(a) TCP. (b) UDP.

Figure 5. Port number usage during a five-hour period at Site C.

server IP address and server port pair. One port (3000) on the server side is frequently used, but no rules are generated due to the low degree centrality of every server.

This problem is more acute for UDP flows. As shown in Figure 4(b), one client opens a UDP port (2000) and sends packets to several servers with different IP addresses but the same port number. According to degree centrality, the client node (1.1.1.1, 2000) is a server node. When the client begins the next communications session using another UDP port (2001), another client node (1.1.1.1, 2001) is considered to be a new server node. To solve these problems, it is necessary to consider locally frequently-used server ports in SCADA networks.

- **Communications via a Fixed Port:** A client can use a fixed port. To communicate with programmable logic controllers, the client and server sides are configured to use fixed ports. In particular, some protocols, such as NetBIOS and SUN RPC, force the same port to be used by the client and the server. Therefore, it is necessary to also discover locally frequently-used ports that primarily use local sites because client-side ports should not be mistakenly identified as server-side ports. A locally frequently-used port can be inferred by the usage count, which varies per port as shown in Figure 5. In the figure, it is evident that there are roughly ten TCP ports and three UDP ports that are locally frequently-used ports when the usage count is greater than 1,000.

3. Flow Whitelist Creation

This section describes the methodology for creating a flow whitelist based on the observations discussed in the previous section.

3.1 Overview

The flow whitelist creation methodology, which is presented in Figure 6, has three phases: (i) preparation; (ii) generation; and (iii) inspection. The initial

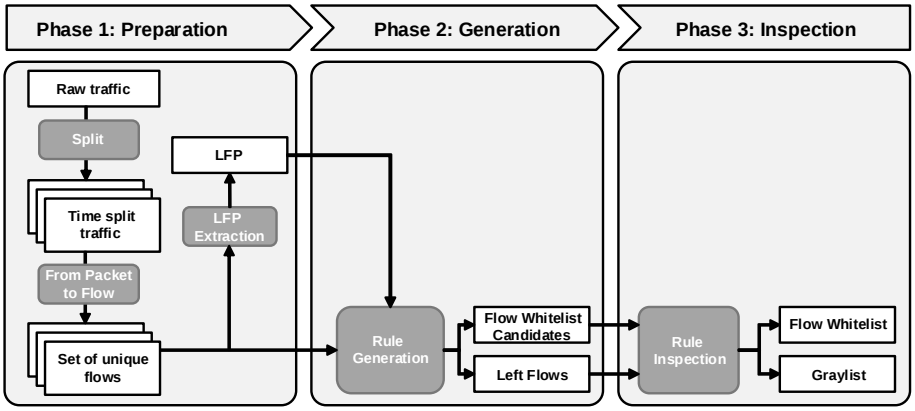


Figure 6. Flow whitelist creation.

preparation phase divides raw traffic into a set of unique flows in each hourly interval without redundant flows. In addition, it lists the locally frequently-used ports based on all flows. The generation phase selectively identifies candidates for the flow whitelist using a generation algorithm. The final inspection phase analyzes the flow whitelist candidates and omits flows that do not match the results of the algorithm executed in the previous phase. Thus, it helps administrators decide whether or not candidates should be placed in the final whitelist.

3.2 Phase 1: Preparation

The preparation phase processes raw traffic to make it appropriate as an input to the second phase. It splits the raw traffic according to a timeline, creates a set of unique flows and lists the locally frequently-used ports.

Raw traffic is first separated by time because flow whitelist generation examines flows within a given period. Next, a set of unique flows is produced; this set includes essential traffic information (i.e., represented as five-tuples) to reduce the processing burden. Finally, the ports that are frequently used with different IP addresses at a local site are listed.

Figure 7 presents the locally frequently-used port extraction process. Two threshold values, θ_{Unit} and θ_{Final}, are employed. The θ_{Unit} threshold is used to filter specific ports that are rarely found in a given set of unique flows. During the filtering, the number of a used port is set to zero as long as the number does not exceed the threshold; otherwise, the value is retained until the end of the process. The θ_{Final} threshold is used in the second phase to obtain the final locally frequently-used port list. In Figure 7, the thresholds used for θ_{Unit} and θ_{Final} are 5 and 25, respectively.

To overcome the lack of knowledge of flows in a SCADA network, the locally frequently-used ports serve as an informative reference to infer server-side ports

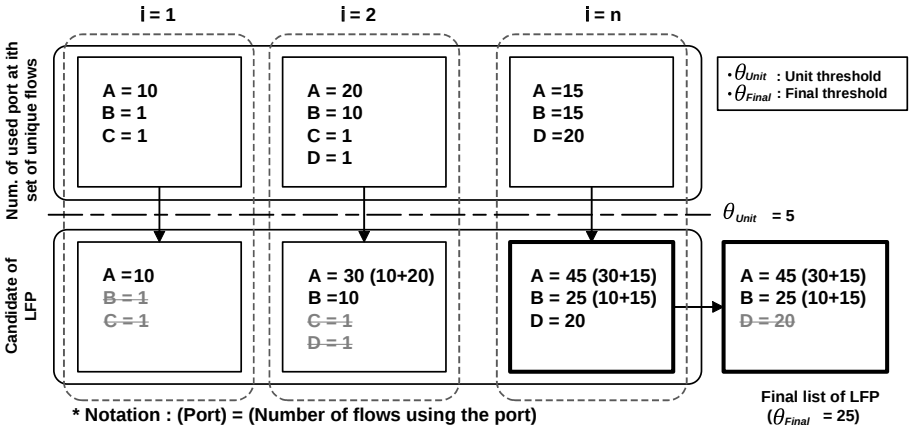


Figure 7. Locally frequently-used port extraction.

without mistakenly identifying the client side as the server side for TCP and UDP traffic. Moreover, based on the observations in Section 2.4, it can be an opportunity to quickly generate a flow whitelist and accurately use the locally frequently-used ports because many uses of server IP addresses are converted in advance to a flow whitelist.

3.3 Phase 2: Generation

The generation phase creates the flow whitelist. The set of unique flows is input to Algorithm 1, which uses two indicators, degree centrality and locally frequently-used ports, as discussed in Section 2.

The algorithm initially checks if a unique flow matches an already-generated rule. When a flow is covered by a rule, the flow is ignored and a new rule is not generated. If a flow is not covered by a rule, then the locally frequently-used port list is examined for a port associated with the source or destination IP address. Three cases exist: the locally frequently-used port list includes: (i) only a source-side port; (ii) only a destination-side port; or (iii) both. As mentioned above, a port is designated as "ANY" if it is not present in the locally frequently-used port list. Thus, the inference of the server side based on the locally frequently-used ports is prioritized when generating the flow whitelist.

Next, the degree centrality with direction is analyzed to handle nodes in which the flow is centralized. Node in-degree and out-degree centrality values are computed by counting the numbers of incoming and outgoing links, respectively. When the in-degree or out-degree centrality of a node exceeds the port threshold (θ_{Port}) defined by an administrator, rule generation is triggered, which depends on the direction of flow at the node. For example, a rule is generated at Lines 14–17 of the algorithm if the node in-degree centrality exceeds the port threshold because many flows head to the destination. In the rule

Algorithm 1 : Flow whitelist generation.

Main:

```
for each file in files do
    MakeUniqueFlows (...)
    for each flow in flows do
        RuleGeneration(protocol, dIP, dPort, sIP, sPort)
    end for
end for
```

RuleGeneration:

```
if IsInRule() == True then return
end if
if sPort and dPort ∈ LFP then
    AddRule(sIP, sPort, dIP, dPort)
else
    if sPort ∈ LFP then
        AddRule(sIP, sPort, dIP, ANY)
    else
        if dPort ∈ LFP then
            AddRule(sIP, ANY, dIP, dPort)
        end if
    end if
end if
if In–degree of DestinationNode > θ_Port then
    Find sIPs connected with dNode
    AddRules(sIPs, ANY, dIP, dPort)
end if
if Out–degree of SourceNode > θ_Port then
    Find dIPs connected with sNode
    AddRules(sIP, sPort, dIPs, ANY)
end if
if AddRule == True then
    Remove all flows matched with the rule
end if
```

generation process, all flows that match a rule generated based on the locally frequently-used ports and degree centrality are eliminated.

3.4 Phase 3: Inspection

The inspection phase confirms if a generated (candidate) rule is whitelisted. Because of the use of the threshold in rule generation, remaining flows may exist. In fact, the inspection phase could generate rules flow by flow. However, not all flows in a SCADA network are legitimate; such flows arise due to link failures or problems with field devices. Therefore, guidelines are provided to serve as a graylist for the remaining flows as well as for the inspection of whitelist candidates. The following are some example guidelines:

Table 2. Preparation phase results.

Site	Volume of Raw Traffic	Volume of Unique Flows	Number of LFPs ($\theta_{Unit} = 25, \theta_{Final} = 100$)
A	453 GB	3.9 GB	49
B	37.1 GB	0.027 GB	11
C	492 GB	0.65 GB	39
D	207 GB	5.8 GB	215
E	425 GB	0.18 GB	6
F	2,622 GB	0.2 GB	77
G	280 GB	0.028 GB	19

- **Unidirectional Rule:** A unidirectional rule represents a one-way flow. It can result from the operational objective of a device or a problem situation. For example, a node may need to send or receive data from other nodes due to an operational objective. However, a packet may not reach its destination because of a problem in a SCADA device or network.

- **Usage Rule Frequency:** It is possible to identify a small rule usage count when a flow is not created often. At times, an unpopular rule may be meaningful if the rule deals with an essential service in a device. Otherwise, it is necessary to inspect the rule to identify if it is normal and can be entered in the whitelist.

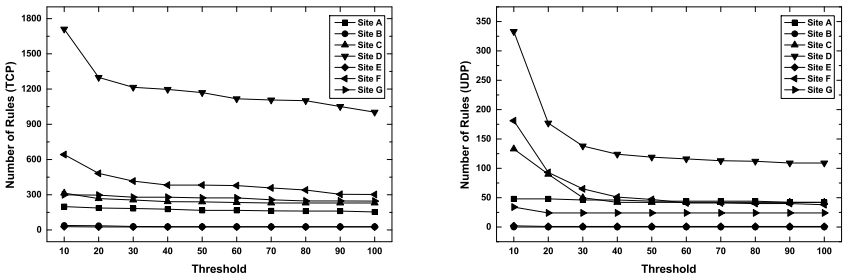
- **Dynamic Port:** A port is dynamically assigned by a service such as FTP. In this case, the flow whitelist is not generated because the flows do not satisfy the degree centrality threshold. Therefore, a rule must be created with "ANY" for the source and destination ports if there are many flows between two nodes after inspecting the remaining flows.

4. Experimental Results

This section presents the results obtained during the preparation, generation and inspection phases of the experiments.

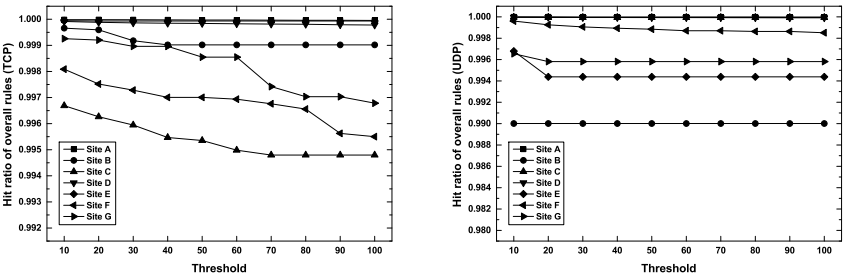
4.1 Phase 1: Preparation

Table 2 shows the results after processing raw traffic and running the locally frequently-used port (LFP) extraction procedure. As seen in the table, the volumes of the flows decreased dramatically compared with the raw traffic flow. In addition, several locally frequently-used ports were obtained when θ_{Unit} and θ_{Final} were defined as 25 and 100, respectively.



(a) TCP. (b) UDP.

Figure 8. Numbers of rules for varying degree centrality thresholds.

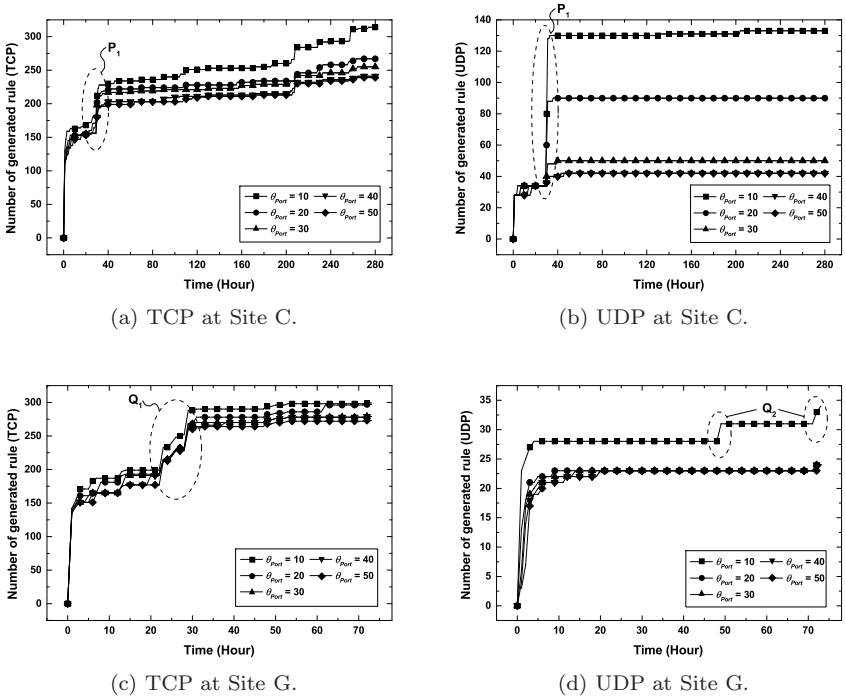


(a) TCP. (b) UDP.

Figure 9. Hit ratios of overall rules for varying degree centrality thresholds.

4.2 Phase 2: Generation

The experimental results were analyzed in terms of two factors: (i) degree centrality threshold; and (ii) time. Figures 8 and 9 present the numbers of generated rules and the hit ratios of generated rules for varying degree centrality thresholds for the seven sites. When the generated rule number was represented as in Figure 8, it was discovered that the number of rules decreased as the threshold increased. Although the extent of the decrease differed between sites, the numbers of rules for both TCP and UDP sharply decreased when the threshold rose from 10 to 20. Notably, all the hit ratios of the rules were close to one, as shown in Figure 9. The minimum hit ratios were 0.995 and 0.990 for TCP and UDP, respectively. However, unlike the UDP rules, the TCP rules were sensitive to an increasing threshold. The remaining hit ratios could possibly have been in the graylist, which must be inspected by an administrator. In summary, all the results decreased and nearly stabilized at particular values, regardless of the transport protocols and sites; only the amounts of the decreases differed.



(a) TCP at Site C.

(b) UDP at Site C.

(c) TCP at Site G.

(d) UDP at Site G.

Figure 10. Numbers of rules for varying degree centrality thresholds in a time series.

This enabled a reasonable threshold to be chosen so that a flow whitelist could be created for optimal performance with respect to protocol-specific or site-specific operations. For example, the appropriate threshold for TCP at Site C was 70 whereas it was 30 for UDP. In the case of TCP, the number of rules decreased to between 10 to 50 for the threshold, and the hit ratio noticeably decreased to 70 for the threshold. Thus, 70 was deemed to be the suitable threshold for TCP at Site C. Otherwise, the number of rules for UDP rapidly decreased until a threshold value of 30 was reached; however, the hit ratio did not change significantly over the range of thresholds. Consequently, it was determined that a threshold of 30 would be most appropriate for UDP at Site C.

Figure 10 shows the rule counts for varying thresholds for TCP and UDP at several sites. In particular, Figures 10(a) and 10(b) show that the number of generated rules for every threshold steadily increased, except for the marking period of P_1 at Site C because new services began during the period. On the other hand, during the marking period of Q_1 and Q_2 in Figures 10(c) and 10(d), the reason for a significant increase was different from that for P_1. Specifically, the degree centrality during the period finally exceeded its threshold, causing a rule to be created. In fact, the percentage of flows for UDP (0.06%) was much

lower than that for TCP at Site G. Thus, the number of flows for UDP was continuously maintained when the threshold was greater than or equal to 20.

4.3 Phase 3: Inspection

Similar results were observed during the inspection phase regardless of the sites. An interesting result was obtained for Site C, which was much more indicative in terms of what can be discovered via inspection. Figure 11 lists the usage of the rules, including the number of rules used during each hourly interval and the total hours for each rule over 281 hours at Site C. The total number of generated rules was 447. For convenience, each rule was arranged in descending order according to the rule usage frequency. A dot denotes at least one flow that matches the rule at the given time, the top polygonal graph represents the number of rules used in the time interval and the right polygonal graph represents the number of hours during which the rule was used.

- **Flow Whitelist:** As shown in the upper graph of Figure 11, 203 rules (Rule ID 1 to Rule ID 203) were used during all the time intervals; the subsequent 140 rules (Rule ID 204 to Rule ID 343) were frequently used in 89% of the entire timeline. More than 84% of the rules were always active after 30 hours. This was explained in Section 4.2 as being due to new services that began during the period. Therefore, most generated rules were recommended for inclusion in the flow whitelist.

- **Graylist:** All the generated rules may not be included in a flow whitelist. In the experiments, some generated rules were rarely or ever used within one hour. In addition, there were one-way TCP rules between some IP addresses. These rules were determined to be graylist rules. The graylist could be processed without any assistance from a vendor: it was easy to identify unusual rules; moreover, the number of rules was adequate for inspection. Administrators are encouraged to use graylists to investigate SCADA networks and devices.

- **Left Flows:** Analysis revealed 1,062 left flows for TCP and 20 flows for UDP. In the case of TCP, 790 flows (approximately 53%) were identified; they were inferred as a dynamic port and twelve unidirectional TCP flows. Thus, 260 TCP flows and 20 UDP flows remained.

4.4 Discussion

Rules in a flow whitelist have interesting characteristics, correlations and periodicities. As shown at the bottom of Figure 11, a deactivation of Rule Group A and an activation of Rule Group B occurred at hour 209. The rules in the two groups were compared to ascertain the reason for the rule transition. The conclusion was that the transition was due to the service port assigned at the time. Although the two rule groups were both valid, the simultaneous appearance of Rule Groups A and B may have been abnormal. If the

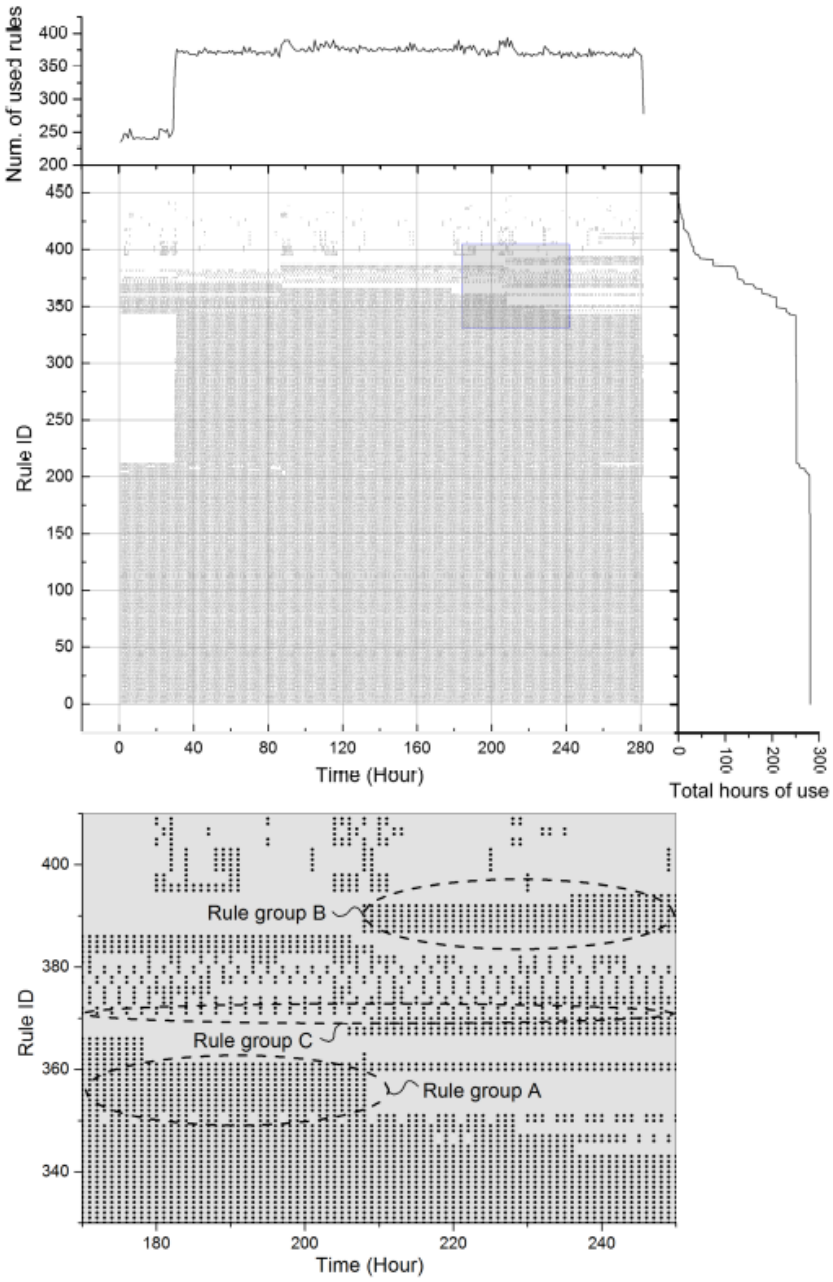


Figure 11. Usage of rules at Site C with varying times and rule IDs.

switchover from Rule Group A to B was sufficiently informative, an intrusion detection system that employs the flow whitelist could be programmed to alert an administrator.

In addition, periodicity was discovered in the flow whitelist. The rules in Rule Group C matched the flows for two hours. If these rules were to relate to an important regular activity (e.g., data backup), the disappearance of matched flows could be reported to an administrator, even if the issues were not security-related.

5. Conclusions

SCADA networks are often interconnected with other networks, including corporate networks and the Internet, greatly increasing the risk of intrusions by malicious entities. Whitelist-based intrusion detection is an attractive network security solution, but flow whitelist extraction from SCADA network traffic is a challenging task. The proposed methodology for creating flow whitelists using degree centrality and locally frequently-used ports addresses the challenges. Experimental results involving seven SCADA systems demonstrate that the methodology generates effective whitelists for deployment in SCADA networks. Indeed, the whitelists can be deployed in a variety of network devices, including switches, firewalls and intrusion detection systems. The approach also supports rule updating because a flow whitelist can indicate the services that have been added or removed by repeatedly updating itself; this enables administrators to easily handle the tasks of flow whitelist extraction and updating. Future research will focus on the deep inspection of SCADA network traffic to address non-disclosure characteristics.

References

[1] R. Barbosa, R. Sadre and A. Pras, A first look into SCADA network traffic, *Proceedings of the IEEE Network Operations and Management Symposium*, pp. 518–521, 2012.

[2] R. Barbosa, R. Sadre and A. Pras, Flow whitelisting in SCADA networks, *International Journal of Critical Infrastructure Protection*, vol. 6(3-4), pp. 150–158, 2013.

[3] L. Freeman, Centrality in social networks: Conceptual clarification, *Social Networks*, vol. 1(3), pp. 215–239, 1978-1979.

[4] Y. Jang, I. Shin, B. Min, J. Seo and M. Yoon, Whitelisting for critical IT-based infrastructure, *IEICE Transactions on Communications*, vol. 96(4), pp. 1070–1074, 2013.

[5] D. Kang, B. Kim, J. Na and K. Jhang, Whitelist generation technique for industrial firewalls in SCADA networks, in *Frontier and Innovation in Future Computing and Communications*, J. Park, A. Zomaya, H. Jeong and M. Obaidat (Eds.), Springer, Dordrecht, The Netherlands, pp. 525–534, 2014.

[6] National Cybersecurity and Communications Integration Center, ICS-CERT Year in Review: Industrial Control Systems Cyber Emergency Response Team, Department of Homeland Security, Washington, DC, 2013.

[7] National Cybersecurity and Communications Integration Center, ICS-CERT Monitor, Department of Homeland Security, Washington, DC, January – April 2014.

[8] A. Sood and R. Enbody, *Targeted Cyber Attacks: Multi-Staged Attacks Driven by Exploits and Malware*, Syngress, Waltham, Massachusetts, 2014.

[9] K. Stouffer, J. Falco and K. Scarfone, Guide to Industrial Control Systems (ICS) Security, NIST Special Publication 800-82, National Institute of Standards and Technology, Gaithersburg, Maryland, 2011.

[10] J. Yun, S. Jeon, K. Kim and W. Kim, Burst-based anomaly detection on the DNP3 protocol, *International Journal of Control and Automation*, vol. 6(2), pp. 313–324, 2013.