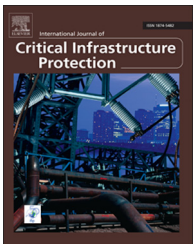
Available online at www.sciencedirect.com

ScienceDirect

www.elsevier.com/locate/ijcip



Flow whitelisting in SCADA networks

Rafael Ramos Regis Barbosa[a,*], *Ramin Sadre*[a], *Aiko Pras*[b]

[a]*Design and Analysis of Communication Systems, Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands*

[b]*Distributed and Embedded Systems, Department of Computer Science, Aalborg University, Selma Lagerlofs Vey 300, DK-9220 Aalborg, Denmark*

ARTICLE INFO

Article history:
Received 15 January 2013
Available online 20 August 2013

Keywords:
SCADA systems
Intrusion detection
Network flow whitelisting

ABSTRACT

Supervisory control and data acquisition (SCADA) networks are commonly deployed in large industrial facilities. Modern SCADA networks are becoming more vulnerable to cyber attacks due to the common use of standard communications protocols and increased interconnections with corporate networks and the Internet. This paper describes an approach for improving the security of SCADA networks using flow whitelisting. A flow whitelist describes legitimate traffic based on four properties of network packets: client address, server address, server-side port and transport protocol.

The proposed approach incorporates a learning phase in which a flow whitelist is learned by capturing network traffic over a period of time and aggregating it into flows. After the learning phase is complete, any non-whitelisted connection observed generates an alarm. The evaluation of the approach focuses on two important whitelist characteristics: size and stability. The applicability of the approach is demonstrated using real-world traffic traces captured at two water treatment plants and at an electric-gas utility.

1. Introduction

Supervisory control and data acquisition (SCADA) networks are commonly deployed to aid the operation of large industrial facilities such as water treatment plants and electric utilities. In the past, these networks were completely isolated and relied on purpose-specific hardware and software; but now they employ commodity hardware and are increasingly interconnected using standard network protocols such as TCP/IP. While this new scenario reduces costs and improves efficiency, the side effect is that the networks are exposed to a much wider range of attacks.

This paper proposes a flow whitelisting approach that seeks to reduce the number of attack vectors on SCADA networks that use TCP and UDP as their primary transport protocols. A "flow" is defined as a bidirectional sequence of packets with identical client address, server address, server-side port and transport protocol. Flow whitelists represent all the legitimate traffic based entirely on these four properties of network packets.

Flow whitelisting presents several advantages over deep packet inspection and host-level intrusion detection [7,8,10]. By not depending on the packet payload, flow whitelisting can handle proprietary protocols. Furthermore, it can operate at the network level; thus, it is not necessary to modify a host. This addresses the reluctance on the part of SCADA operators to make changes to their computing environments. Note that, although flow-level whitelists are not commonly used in traditional IP networks because the number of legitimate connections is too large to be manageable, they have been applied to specific problem domains such as reducing SPAM [5], avoiding phishing [9], guaranteeing access to important customers during DDoS attacks [15], and preventing various VoIP infrastructure attacks [6].

*Corresponding author.
E-mail address: r.barbosa@utwente.nl (R.R.R. Barbosa).

1874-5482/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijcip.2013.08.003

The main motivation for using whitelists is that most SCADA network traffic is generated by automated processes, such as the periodic polling of field devices. Also, SCADA systems are closed with very limited external access, if any. Moreover, changes are rare, in other words, SCADA hosts and services are infrequently added to or removed from the network.

Whitelisting has been recommended by several entities as a means for implementing SCADA security. For instance, the Norwegian Oil and Gas Association [12] suggests that "all access requests shall be denied unless explicitly granted. The U.S. National Institute of Standards and Technology (NIST) [14] recommends the "[blocking of] all communications with the exception of specifically enabled communications." However, to the best of our knowledge, the viability of whitelists has never been studied in real-world SCADA environments. In the previous work [2], we have shown that connection matrices are remarkably stable in SCADA networks, suggesting that whitelists are feasible in these environments.

This paper presents an approach for flow whitelisting in SCADA networks that assist network administrators in detecting illegitimate network traffic. In order to be viable, a whitelist must possess two characteristics. First, its size must be manageable. A very large list with millions of entries, as encountered in traditional IP networks, would make the approach infeasible to implement and manage. Second, the whitelist must be stable. If the list is unstable (i.e., changes frequently), it would require continuous updating by the network administrator or it would generate a large number of false alerts. The feasibility of the whitelisting approach is demonstrated using real-world traffic from two water treatment facilities and an electric-gas utility.

2. Flow whitelisting approach

Fig. 1 presents an overview of the flow whitelisting approach. The traffic in the SCADA network is captured, aggregated to connections and then aggregated to flows. A "connection" is defined as all packets with the same source/destination IP address, source/destination port and IP protocol, regardless of the direction in which the packets are sent. In the subsequent learning phase, the flows are observed over a certain period of time in order to create an initial flow whitelist. A flow whitelist contains entries of the form: client IP address, server IP address, server port, IP protocol. After the whitelist is generated, the connections are analyzed in the detection phase. All network traffic matching a whitelist entry is considered to be legitimate. Every connection that does not match a whitelist entry generates an alarm.

2.1. Connection and flow creation

Since the approach relies on IP packet header information, the packet headers have to be captured in the (sub)networks that are to be monitored by the whitelist. This paper only considers TCP and UDP packets. Connection creation involves the aggregation of the captured packets to connections. As mentioned before, a connection is defined as all packets with the same source/destination IP address, source/destination port and IP protocol, regardless of the direction of the packets. The end of a connection is determined either by using a TCP state machine or an inactivity timeout of 300 s. Our experiments used the `argus` open source tool to perform this task.

The flow creation step identifies the client and server sides of the connections and further aggregates the connections according to the four-tuple flow definition. Four rules are applied in sequence to identify the server side:

- Rule 1 applies to all TCP connections for which a three-way handshake is observed. The server is set to be the host that received the SYN packet or sent the SYN/ACK packet.
- Rule 2 is applied if a well-known port (below 1024) is observed: the host using such a port is set to be the server. Note that in the case of an active FTP session, where the originator of a data connection is the server, the source port 20 is set as a service port. In the case of protocols that use the same (well-known) port on both hosts (e.g., NTP), Rule 1 or Rule 4 is used for classification.
- Rule 3 is a heuristic. If the same protocol and port are reused by a host in multiple connections, then the host is set as the server and the protocol–port combination is used to identify the service. The heuristic relies on the fact that client ports normally vary with each connection and are less likely to be repeated. Rule 3 makes it necessary to keep every connection that is not classified by Rules 1 and 2 in memory until a second connection with a repeated host address, protocol and port is observed; this can potentially delay the analysis indefinitely. A timeout could be used in an online implementation, after which time the connection is classified using Rule 4. The offline implementation described in this paper employs an infinite timeout.

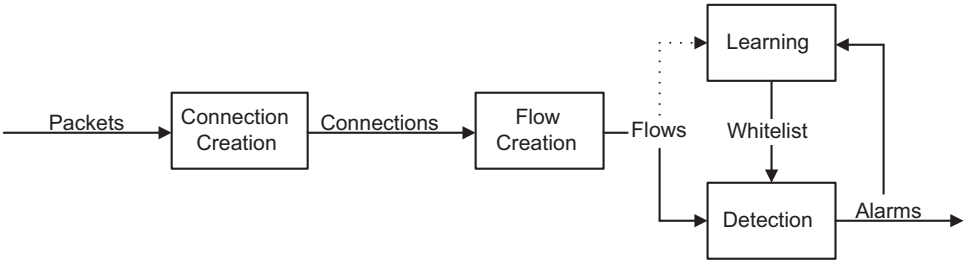


Fig. 1 – Flow whitelisting approach.

- Rule 4 is applied to flows that do not match one of the previous three rules. Rule 4 sets the server to be the destination of the first packet observed in the connection.

Note that the four rules implicitly assume that every transport port and IP protocol pair used by a server uniquely identify a service in the SCADA network. This definition is particularly problematic for network services that use dynamic port allocation (DPA) such as Microsoft's Active Directory. In this service, high ports (above 1024) are dynamically allocated for remote procedure calls [11]. We acknowledge this limitation, and discuss its effects when presenting our experimental results.

2.2. *Learning phase*

Ideally, a network administrator would know all the services deployed in a SCADA network and a flow whitelist could be constructed based on this knowledge. In practice, however, complete information is rarely available, partly due to the use of proprietary SCADA protocols.

The goal of the learning phase is to automatically create an initial whitelist from traffic collected over a period of time (learning period). Two assumptions are made: (i) all flows in the learning period are legitimate; and (ii) most legitimate flows can be observed during the learning period. The first assumption is valid because anomalous or malicious events are much rarer in SCADA networks than in traditional IP networks. In fact, no attacks were observed during the capture of the data sets used in our experiments. The second assumption is based on the expectation that most of the traffic in a SCADA network is automated, which implies that flows would be repeated fairly often. The approach used to set the duration of the learning phase is discussed later in this paper.

Note that all legitimate flows are not expected to be encountered during the learning phase. For example, manual changes to the configuration of programmable logic controllers (PLCs) could, depending on the setup, only happen rarely. Thus, flows related to this activity may not be present in the whitelist. For this reason, network administrators are provided with the opportunity to augment the whitelist during the detection phase.

2.3. *Detection phase*

The whitelist created during the learning phase is used in the detection phase to identify illegitimate flows. If a flow is whitelisted, nothing happens; otherwise an alarm is raised. In a real-world deployment, an administrator would either add the flow that raises an alarm to the whitelist (treating the alarm as a false positive) or block the flow (treating the alarm as a true positive). Note that, unlike traditional IT networks, where hosts are commonly put in quarantine in case of malicious activities, automatic blocking is not advisable for SCADA environments. This is because blocking legitimate SCADA traffic could have dire consequences such as an electricity blackout. This issue is discussed later in the paper.

3. Experimental results

This section presents the results of four experiments used to evaluate the viability of flow whitelists in SCADA networks.

The first experiment was designed to verify if the whitelist is manageable by comparing the size of the complete whitelist with the number of hosts and communicating pairs in a network. The second experiment examined the ideal learning time. The third experiment focused on the classification of the alarms produced by the data sets. The fourth experiment analyzed the distribution of alarms.

In the experiments, it was necessary to simulate administrator intervention during the detection phase. This was accomplished by always adding a flow that causes an alarm to the whitelist, while also storing the alarm for post-processing. Thus, an alarm is never repeated, and the analysis can concentrate on the nature of the alarms rather their number.

3.1. *Data sets*

The experiments used network packet traces collected at three SCADA environments: two water treatment facilities (*water1* and *water2*) and one electric-gas utility (*electric-gas*). Two data collections were performed simultaneously at one of the water treatment facilities, one in the field sub-network (*water2-field*), consisting of PLCs, remote terminal units (RTUs) and field devices; and one in the control sub-network (*water2-control*), consisting of servers with different functions (e.g., polling PLCs, maintaining historical data and performing access control) and human machine interfaces (HMIs) (i.e., operator workstations). A single collection was performed at the other two facilities, each containing all the data for the two logical sub-networks. The SCADA data sets comprised full-packet `tcpdump/libpcap` traces and each collection was treated as a separate data set .

Two traditional IT network data sets were used for comparison purposes. One comprised a publicly available `tcpdump/libpcap` trace captured at an educational organization (*loc6*) [4]. Only a portion of the available data was used, approximately the first 7.5 days of the trace. The other traditional IT data set comprised fifteen days of NetFlow records collected at an internal router located at a university campus (*uni*). Table 1 presents an overview of the six data sets.

Note that the flow creation step cannot be applied to the *uni* data set. NetFlow records do not contain enough information to identify the host that initiates a TCP connection according to the three-way handshake, which is necessary to apply Rule 1. Instead, we used the techniques described in [13] to aggregate NetFlow records to connections. Therefore, the client side (termed as the originator in [13]) and, as a consequence, the server side and the service port of a connection were determined using these techniques.

3.2. *Whitelist size*

The first experiment sought to verify if the whitelist size is manageable. Specifically, it evaluated the sparseness of the

connection matrix, i.e., if the number of acceptable flows was small compared with the number of possible flows.

This characteristic was tested by extending the learning time to the full duration of each trace and counting the number of flows observed. This allowed the estimation of the size of the whitelist corresponding to a complete trace, assuming that no attacks were present in the data set. Although no attacks were reported during the capture of the SCADA data sets, malicious activities such as network scans are so common in traditional IT networks that their artifacts were almost certainly present in the *loc6* and *uni* data sets. The bias was minimized by only considering flows for which traffic was observed in both directions; this reduced the number of observed flows caused by network scans and other network anomalies.

Table 2 shows the whitelist size for each of the data sets. Note that the internal hosts column in the table corresponds to the number of observed hosts located inside the monitored networks and the host pairs column corresponds to the number of communicating host pairs. In order to make the different traces comparable, the metrics are provided as absolute values and as ratios based on the numbers of internal hosts (in parenthesis).

In most cases, the sizes of the whitelists for the SCADA data sets have the same order of magnitude as the corresponding numbers of internal hosts, suggesting that flow whitelisting is feasible in these environments. In comparison, the traditional IT network data sets have whitelists that are orders of magnitude larger than the corresponding numbers of internal hosts, illustrating why the approach does not scale to traditional network environments. Because of the massive whitelists obtained for the traditional IT data sets, we did not use these data sets in the remaining experiments.

Another observation is that the differences between the host pair and whitelist ratios are not very large, implying that, on the average, there are one or two services per server. This means that a whitelist without service information, which is less restrictive and, thus, considerably less secure, would not reduce the whitelist size in a significant manner.

The only exception to the results is that the *water2-control* data set has a whitelist that is one order of magnitude larger than the number of communicating host pairs. However, we show later in this paper that this difference is mostly caused by a traffic anomaly.

3.3. Training set size

The second experiment examined the effect of the learning time on the size of the learned whitelist. Fig. 2 shows the size of the learned whitelist as a percentage of the total number of flows versus the learning time for the four SCADA data sets. In the case of the *water1* and *water2-field* data sets, more than 50% of the flows were observed within the first hour of traffic, with only a few additions during the first day. This percentage is much lower for the *water2-control* and *electric-gas* data sets, which are around 10% and 15%, respectively. The *water2-control* data set shows a significant jump in whitelist size after around 7 days. The whitelist for the *electric-gas* data set grows steadily from day 10 to day 40. The reasons for this behavior are explained in the next section.

Despite the differences, one characteristic was shared by all the SCADA data sets: no additions were made to the whitelist during the second day of traffic. In fact, almost no additions were made up to the third day in the case of the two *water* data sets, and for an even longer period for the *electric-gas* data set. As a result of this observation, we set the learning time to one day in the next two experiments.

3.4. Nature of alarms

This experiment focused on the sources of instability in the whitelists, specifically, the nature of the flows not observed during the learning phase. During our analysis we identified four main alarm classes:

Table 1 – Overview of data sets.

Name	Hosts	Duration (days)	Packets	Bytes	Connections
water1	45	13	591 M	96 GB	76 K
water2–control	14	10	26 M	4 GB	131 K
water2–field	31	10	67 M	24 GB	215 K
electric–gas	388	86	2 G	511 GB	179 M
loc6	93	7.5	53 M	53 GB	264 K
uni	22,685	15	161 G	126 TB	1 G

Table 2 – Whitelist size ratios.

Data set	Internal hosts	Host pairs	Whitelist size
water1	51	58 (1.1)	81 (1.6)
water2–control	22	40 (1.8)	542 (24.6)
water2–field	14	20 (1.4)	23 (1.6)
electric–gas	388	542 (1.4)	1188 (3.1)
loc6	93	23,322 (250.8)	26,759 (287.7)
uni	22,685	56,425,836 (2487.4)	141,744,206 (6248.4)

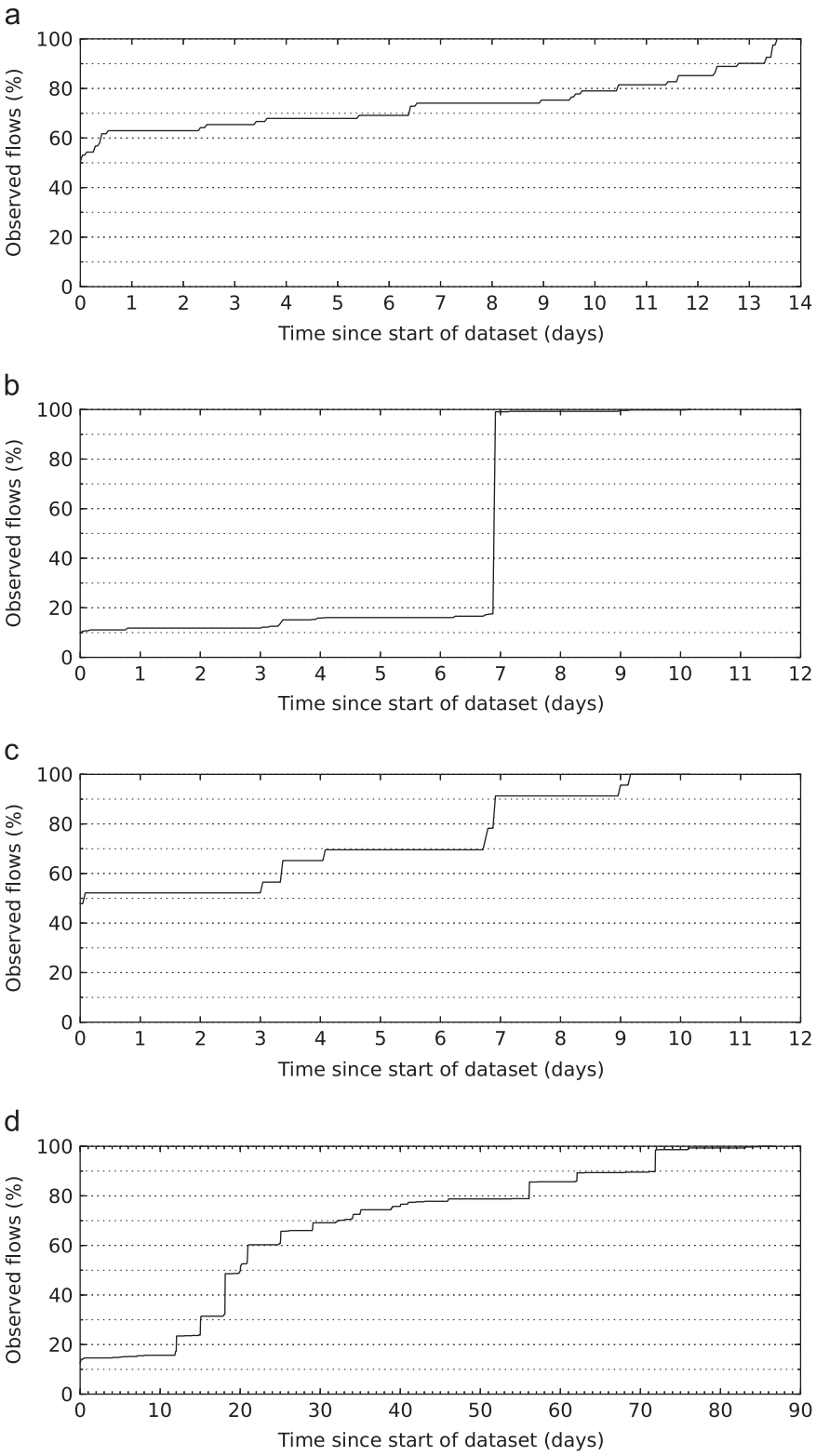


Fig. 2 – Number of learned flows over time. (a) Data set *water1*, (b) Data set *water2-control*, (c) Data set *water2-field*, (d) Data set *electric-gas*.

- *DPA Anomalies*: The service definition assumes a one-to-one mapping with transport ports, which is problematic in the case of DPA. We did not attempt to uncover all the services using dynamic ports, but we identified anomalies that were most likely triggered by DPA. The *water2-control* and *electric-gas* data sets had instances where several TCP connections were made by the same hosts in a short sequence, with monotonically increasing transport port numbers on the client and server sides. Table 3 shows an excerpt of such an instance.

Table 3 – Dynamic port allocation example.

Start time	Protocol	Source port	Dest. port	Packets	State
09:26:50.944328	TCP	3714	1178	16	FIN
09:26:50.960961	TCP	3715	1178	2	RST
09:26:50.976884	TCP	3716	1180	16	FIN
09:26:50.990740	TCP	3717	1180	2	RST
09:26:51.007886	TCP	3718	1183	16	FIN
09:26:51.021606	TCP	3719	1183	2	RST

- *Manual Activity*: This class consists of human-triggered flows. All the flows with the following services (identified by a protocol and port number) fall in this class: `telnet` (TCP-22), `ssh` (TCP-23), `http` (TCP-80), `https` (TCP-443), `shell` (TCP-514), `kshell` (TCP-544), `rdp` (TCP-3389), `vnc` (TCP-5800 and TCP-5900) and `x11` (TCP-6000 to TCP-6007). The *water1* and *water2* data sets had flows involving operator workstations. If the client side of a flow was an operator workstation, then the flow was also classified as manual.
- *New Host*: This class contains all flows for which at least one host (server or client) did not communicate during the learning period and, therefore, would not be in the whitelist.
- *Other*: This catchall class contains all flows that do not fall in any of the other classes.

Each flow was mapped to a single class and class membership was tested in the same order listed above. For instance, consider the case of an alarm for a flow where the client is not present in the whitelist and where the service is `ssh`. In this case, the flow is classified as "manual activity" because this class has precedence over the "new host" class.

When analyzing the *electric-gas* data set, we observed two events that deserve special consideration. In SCADA environments, it is very common for most network functions to be replicated (e.g., using duplicate servers) in order to increase reliability. The first event of interest involves a single redundant host taking over the tasks of one of the main servers in the network, specifically, the SCADA server responsible for polling field devices.

Just before the change occurred, we observed reboot commands issued over `telnet` connections to some of the PLCs. We did not observe `telnet` traffic to all the PLCs. However, because all the changes occurred in a relatively small time interval, we assumed that they were related.

In addition to the flows involving the PLCs, several other long-lived flows presented the same behavior, for example, some `ssh` flows were also "switched" to the redundant host. According to the SCADA operators, changes like these are routinely performed in order to verify if the redundant hosts work properly.

In our analysis, we adopted the following procedure to identify flows corresponding to this event. If one of the hosts in the flow was the SCADA server, we looked for another flow with a similar key, where only the SCADA server address was changed to its backup or vice versa.

The second event involved the relocation of many hosts in the network, mostly PLCs. At times, a continuous range of IP addresses had their address changed to (logically) separated sub-networks. For example, all the hosts in the IP address range `X.Y.Z.61` to `X.Y.Z.71` had their addresses changed to the range `X.Y.A.61` to `X.Y.A.71`, respectively. We observed `telnet` commands being issued to perform the address change, but not to all hosts. Again, the small time interval between the changes suggests that they are related. According to the operators, a large sub-network was split into several smaller ones. After the change, the logical address better represented the geographical location of a host.

We identified flows corresponding to this event simply by verifying if either host in the flow (client or server) was part of one of the newly created networks. It is important to note that we did not classify `telnet` access to these hosts leading to these events as part of the flows; this is because `telnet` connections are always classified as manual activity.

3.5. *Frequency of alarms*

The final experiment applied the classification method to all the SCADA data sets to examine the frequency of each type of alarm. As discussed above, the learning time was set to be the first day of the data set. Table 4 presents the results of the classification, including the absolute number of alarms and approximate percentages for each class. The results for the *electric-gas* data set are broken down to show the two events discussed in the previous section.

In the *water1* data set, the "new host" alarms involved a few short `snmp` connections, likely due to testing; one `ntp` connection that appeared to repeat once a week; and one real anomaly – several single-packet TCP connections attempts at port 1010. The one flow in the "other" alarm class appeared to be caused by DPA; a few moments before it started, a flow involving the same hosts but a different server port ended. Finally, the "manual activity" alarm class comprised a few `http(s)` and `x11` connections, and one connection that originated from an operator workstation.

The leading cause of alarms in the *water2-control* data set was a DPA anomaly, which was responsible for around 91% of the alarms. This anomaly was also responsible for the majority of the flows that contributed to the jump seen in Fig. 2(c). In fact, if the flows generated by this anomaly were to be removed, more than 60% of the flows would be present in the whitelist (i.e., be observed during the learning period), much like the other *water* data sets. In addition, the ratio of the whitelist size to the number of internal hosts would have been considerably smaller, 4.7 instead of 24.6, about the same order of magnitude as that for the other SCADA data sets.

Most of alarms in the "new host" and "other" alarm classes for the *water2-control* and *water2-field* data sets involved a server that, according to the network administrator, was related to user authentication. Thus, the alarms were likely generated by manual activity. The unexpected behavior involved connections made from the authentication server (located in the control network) directly to PLCs (located in the field network). According to the network administrator, such a connection is not allowed – all the connections from the control network to the field network must go through a designated server. The remaining alarms involved hosts foreign to the control and field networks where the data collection was performed. It is not clear if these connections should be allowed.

Compared with the other data sets, the *electric-gas* data set contained a significantly larger number of alarms: 1037 of the flows were not observed in the training period. As in the case of the *water2-control* data set, the largest alarm class was "DPA anomalies," accounting for 35% of the total number of alarms. Redundant and relocation events were responsible for more than half of the alarms in the "new host" and "other" alarm classes.

Fig. 3 shows the numbers of alarms over time, broken down by alarm class. Only the most active period is shown. DPA connection bursts occurred at four distinct times (one is not shown), accounting for the highest peaks. The redundant event occurred on day 15, and some of the manual alarms raised the same day correspond to the `telnet` connections that were used to reboot the RTUs. Interestingly, the larger peak classified as redundant appeared earlier on day 11. All the flows in this peak correspond to single-packet connections that were sent by the redundant host to several RTUs. This was probably the result of a test or a configuration mistake.

Peaks corresponding to the "manual activity" alarms occurred on days 20, 25 and 29. On each of these days, a large portion of the address space was accessed, for a variety of reasons. For instance, connections were created to configure hosts for the first time, some of these appeared later as "new host" alarms, around days 35 and 40.

Most of the hosts were relocated around day 55 and 61. Note that no "manual activity" peak occurred around these days. The `telnet` connections that were used to change host addresses were previously accounted for in the peaks for the "manual activity" alarm class.

These peaks account for the majority of alarms in the data set. The remaining alarms consisted mainly of some manual `ssh`, `x11` and `http` flows; a few Samba-related ports (e.g., TCP/UDP 137–139 and 445); and several high port flows that may have been caused by DPA.

4. Discussion

This section discusses some of the practical issues that network administrators face when implementing flow whitelists in real-world SCADA environments.

4.1. *Dynamic port allocation*

By far, the majority of the alarms identified in our analysis were due to DPA anomalies, and we only identified a portion of the port and protocol pairs used by these services. In the data sets, there were many more connections using DPA than we considered in the analysis. For a flow-level whitelisting approach to work with this type of service, it is necessary to whitelist the entire range of transport ports that might be

Table 4 – Alarm breakdown.

Data set	Dynamic ports	Manual activity	New host	Other
water1	0	14 (47%)	15 (50%)	1 (3%)
water2–control	437 (91%)	16 (3%)	6 (1%)	19 (4%)
water2–field	0	5 (45%)	6 (55%)	0
electric–gas	358 (35%)	269 (26%)	274 (26%)	136 (13%)
Redundant	0	13 (5%)	16 (6%)	75 (56%)
Relocation	0	14 (5%)	148 (54%)	1 (0%)
Remaining	358 (100%)	242 (90%)	110 (40%)	60 (44%)

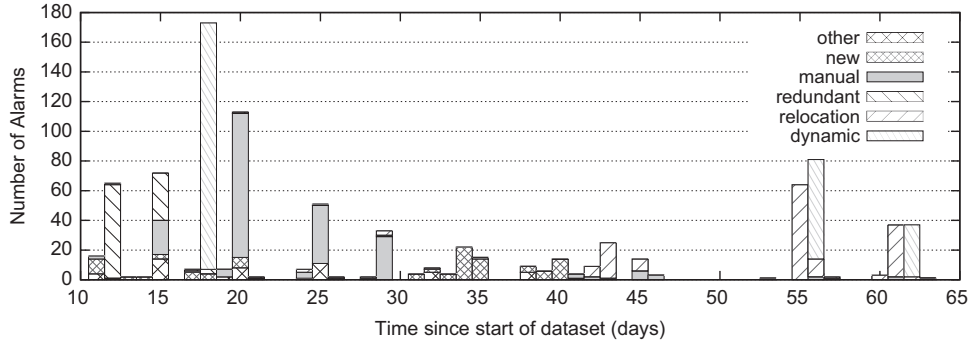


Fig. 3 – Alarms observed for the *electric-gas* data set.

used by the service. This is not an ideal solution because it makes the whitelisting approach more permissive.

One of the main advantages that security experts have when protecting SCADA environments is that traffic patterns are rather predictable compared with traditional IT environments. DPA reduces this predictability. We argue that SCADA systems should be designed without services that use DPA or, at least, the services should be restricted to non-critical segments of the network.

4.2. *Real-world attack scenarios*

The data sets that were used contained no attack data, so it was not possible to test the efficacy of whitelisting against realistic attack scenarios. We use the real-world attack types specified in [3] to motivate how these attacks could be observed.

We consider four types of attacks. The first type comprises information gathering attacks, such as network scans. These are normally performed by injecting several requests into a network, with the objective of discovering the available services and hosts. At the flow level, these attacks are not much different from the DPA anomalies identified in this work. Therefore, they would be easily identified by our approach because non-whitelisted connections are likely to be made.

The three other attack types are as follows: (i) denial-of-service attacks, which prevent a legitimate user from accessing a service or degrade network performance; (ii) network attacks, which manipulate network protocols; and (iii) buffer overflow attacks, which attempt to gain control over a process or crash it by overflowing its buffer. These attacks would only be observed if they were launched from a host that is not allowed to access a given server or service, or if they targeted a non-existent server or service.

In general, an attack is undetected in two situations. Either the whitelist is incorrectly constructed (i.e., it contains entries representing illegitimate traffic) or the attack misuses whitelisted traffic (e.g., an operator machine that is normally used to access a PLC and set an invalid parameter). In the latter case, the connection itself is legitimate, but its contents are not. Note that our approach does not prevent an attacker from spoofing an IP address and masquerading as a legitimate flow. Interested readers are referred to [1] for a discussion of mechanisms that can protect against such attacks.

4.3. *Blocking and flagging*

In a traditional IT environment, it is common practice to take a host offline when it is suspected to be under attack. This is done to limit the impact of the attack and control its spread. However, taking a SCADA host offline could have dire consequences, especially if critical processes depend on its continued operation.

The same reasoning may be applied to blocking traffic – the cost of false positives could be too high. Whitelists, like other systems, can suffer from configuration problems. In our analysis, we observed a number of alarms caused by rare activities such as manual access to PLCs and hosts switching to backup servers (or being accessed by backup clients); these could be overlooked when constructing a whitelist. Therefore, we recommend that, when whitelists are first deployed in a real-world scenario, connections that are not whitelisted should only be flagged (i.e., raise alarms). The decision to block traffic or add it to the whitelist is left to network administrators. Only after they are confident that configuration mistakes are addressed, should they consider using the whitelist to automatically block traffic.

4.4. *Learning limitations*

Many of the alarms were the result of limitations of the technique used to learn the initial whitelist. The alarms mostly corresponded to connections that did not occur frequently and it was impractical to extend the learning time to accommodate them. In general, the longer the learning time, the greater the chance of including an anomalous flow in the whitelist.

Some of the alarms were caused by the presence of new hosts that were not observed during the learning phase. Although changes in SCADA network topology are uncommon, they should be considered when using the whitelisting approach. For every change in the network, it is necessary to update the whitelist accordingly, either manually or by triggering a new learning step. Note, however, that this problem is not exclusive to our approach. Most, if not all, anomaly-based intrusion detection systems require updates after network changes because the "normal" behavior has changed.

The learning step limitation implies that input from network administrators and SCADA vendors is necessary to build a complete flow whitelist. However, relying on this expertise exclusively may be dangerous, as mistakes can occur. For example, the addition of flows representing backup server connections or infrequent `ssh` connections could be overlooked. Presenting a list of flows learned from network measurements, as proposed in the learning phase, could help administrators identify acceptable flows that would otherwise be missed.

5. Conclusions

Flow whitelisting is an appealing and practical solution for combating SCADA network attacks. Unlike traditional IT networks, the size of a whitelist for a SCADA network is manageable, largely due to the number of internal hosts. Another important property is that a SCADA network whitelist is fairly stable. Experiments reveal that, in most cases, more than 50% of the acceptable network flows can be observed within one day of measurement.

Services that use dynamic port allocation are the main cause of alarms in SCADA networks. These alarms can be eliminated by adding to the whitelist the complete range of ports that are allocated by these services, or by removing them from critical network segments. Most of the remaining alarms are the result of limitations in the whitelist construction approach, which, in real-world implementations, can be overcome by incorporating network administrator and SCADA vendor input to refine the whitelist.

Our future work will focus on the development of a user interface that assists in the creation of whitelists and facilitates their manipulation by providing more detailed information about alarms. In addition, we will attempt to enhance SCADA network security by identifying intrusion attempts that (mis)use whitelisted flows.

REFERENCES

[1] C. Abad, R. Bonilla, An analysis of the schemes for detecting and preventing ARP cache poisoning attacks, in: Proceedings of the Twenty-Seventh International Conference on Distributed Computing Systems Workshops, 2007, pp. 60–68.

[2] R. Barbosa, R. Sadre, A. Pras, Difficulties in modeling SCADA traffic: A comparative analysis, in: Proceedings of the 13th International Conference on Passive and Active Measurement, 2012, pp. 126–135.

[3] R. Barbosa, R. Sadre, A. Pras, Towards periodicity based anomaly detection in SCADA networks, in: Proceedings of the 17th IEEE Conference on Emerging Technologies and Factory Automation, 2012.

[4] R. Barbosa, R. Sadre, A. Pras, R. Meent, Simpleweb/University of Twente Traffic Traces Data Repository, Technical Report TR-CTIT-10-19, Center for Telematics and Information Technology, University of Twente, Enschede, The Netherlands (doc.utwente.nl/71273), 2010.

[5] Y. Cao, W. Han, Y. Le, Anti-phishing based on automated individual white-list, in: Proceedings of the 4th ACM Workshop on Digital Identity Management, 2008, pp. 51–60.

[6] E. Chen, M. Itoh, A whitelist approach to protect SIP servers from flooding attacks, in: Proceedings of the IEEE International Workshop on Communications Quality and Reliability, 2010.

[7] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, A. Valdes, Using model-based intrusion detection for SCADA networks, in: Proceedings of the SCADA Security Scientific Symposium, 2007.

[8] Digital Bond, Quickdraw SCADA IDS, Sunrise, Florida ⟨http://www.digitalbond.com/tools/quickdraw⟩.

[9] D. Erickson, M. Casado, N. McKeown, The effectiveness of whitelisting: A user study, in: Proceedings of the 5th Conference on Email and Anti-Spam, 2008.

[10] D. Hadziosmanovic, D. Bolzoni, P. Hartel, A log mining approach for process monitoring in SCADA, International Journal of Information Security 11 (4) (2012) 231–251.

[11] Microsoft Corporation, Service Overview and Network Port Requirements for Windows, Redmond, Washington (support.microsoft.com/kb/832017).

[12] Norwegian Oil and Gas Association, 104 – Norwegian Oil and Gas Recommended Guidelines for Information Security Baseline Requirements for Process Control, Safety and Support ICT Systems, Sandnes, Norway, 2009.

[13] R. Sommer, A. Feldmann, NetFlow: Information loss or win?, in: Proceedings of the 3rd ACM SIGCOMM Workshop on Internet Measurement, 2012, pp. 173–174.

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

[15] M. Yoon, Using whitelisting to mitigate DDoS attacks on critical Internet sites, IEEE Communications 48 (7) (2010) 110–115.