Flow whitelisting in SCADA networks

Barbosa, Rafael Ramos Regis; Sadre, Ramin; Pras, Aiko

Published in:
International Journal of Critical Infrastructure Protection

DOI (link to publication from Publisher):
10.1016/j.ijcip.2013.08.003

Publication date:
2013

Document Version
Early version, also known as pre-print

Chapter 1

FLOW WHITELISTING IN SCADA NETWORKS

Rafael Ramos Regis Barbosa, Ramin Sadre, and Aiko Pras

Abstract

Supervisory Control And Data Acquisition (SCADA) networks are commonly deployed to aid the operation of large industrial facilities. Modern SCADA networks are becoming more vulnerable to network attacks, due to the now common use of standard communication protocols and increased interconnection to corporate networks and the Internet. In this work, we propose an approach to improve the security of these networks based on flow whitelisting. A flow whitelist describes the legitimate traffic solely using four properties of network packets: the client address, the server address, the server-side port, and the transport protocol.

The proposed approach consists in learning a flow whitelist by capturing network traffic and aggregating it into flows for a given period of time. After this 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. We demonstrate the applicability of the approach using real-world traffic traces, captured in two water treatment plants and a gas and electric utility.

Keywords: SCADA, security, intrusion detection

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 are becoming increasingly interconnected and are based on commodity hardware, communicating through standard network protocols (such as TCP/IP). While this new scenario reduces costs and improves efficiency, the side effect is that these networks are now exposed to a much wider range of possible attacks.

In this paper, we propose a *flow whitelisting* approach to reduce the number of attack vectors in SCADA networks that use TCP and UDP as their primary transport protocol. We define a *flow* as a (bidirectional) sequence of packets with identical client address, server address, server-side port, and transport protocol. Flow whitelists represent all legitimate traffic solely based on the above four properties of network packets.

Flow whitelisting presents several advantages over deep packet inspection or host level intrusion detection systems [1–3]. By not depending on the packet payload, flow whitelisting should be able to handle proprietary protocols. Furthermore, it operates at the network level, that is, it is not necessary to modify the hosts. This should overcome a common resistance of SCADA operators in making changes in their environment. It should be noted that, although flow-level whitelists are not commonly used in traditional IP networks, as the number of legitimate connections is too large to be manageable, they have been proposed to some specific domains, such as reducing SPAM [6], avoiding phishing [7], guaranteeing access to important customers during DDoS attacks [8], and preventing various attacks in VoIP infrastructures [9].

The main motivation for the use of whitelists is that most of SCADA network traffic is generated by automated processes, like the periodic polling of field devices. Besides, these systems are closed with very limited external access, if any. Finally, changes should be rare, that is, hosts and services are not excepted to be frequently added to or removed from the network. In fact, the idea of whitelisting can be commonly found in recommendations for SCADA security. For instance, the Norwegian Oil and Gas Association suggests that "all access requests shall be denied unless explicitly granted" [10]. The American National Institute of Standards and Technology (NIST) recommends to "block all communications with the exception of specifically enabled communications" [11]. However, to the best of our knowledge, the viability of whitelists was never studied in real-world SCADA environments. In previous work [12],

we showed that the connection matrix is remarkably stable in SCADA networks, suggesting that whitelists might be feasible in these environments.

The goal of this paper is to present an approach for flow whitelisting in SCADA networks and to study its capability to assist the network administrator in detecting illegitimate network traffic. To be viable the whitelist must present two characteristics. First, its size must be manageable. A very large list with millions of entries, as it would occur in the Internet, would make the approach hard to implement and to manage. Second, the whitelist must be stable. If the list is unstable, i.e., it changes frequently, it either requires continuous updating by the network administrator or it generates a large number of false alerts. We demonstrate the feasibility of our approach using real-world traffic measurements captured in critical infrastructures: two water treatment facilities and one electric and gas utility.

The remainder of this paper is organized as follows. In Section 2 we describe our approach to flow whitelists. The experimental results obtained by applying the approach to real-world traffic traces are presented in Section 3. Section 4 discusses different aspects toward a real-world deployment of the approach. Finally, in Section 5 we present our conclusions.

2 The Flow Whitelisting Approach

Our approach to flow whitelisting is outlined in Figure 1. First, the traffic in the SCADA network is captured, aggregated to connections, and finally aggregated to flows. These steps are described in Section 2.1. In the *learning phase* (see Section 2.2), the flows observed in a certain period of time are used to create an initial flow whitelist. A flow whitelist is a list of entries of the form *(client IP address, server IP address, server port, IP protocol)*. Once the whitelist is generated, connections are analyzed in the *detection phase* (see Section 2.3). All network traffic matching a whitelist entry is considered legitimate. Every connection not matching an entry generates an alarm.

2.1 Connection and Flow Creation

Since our approach relies on information from the IP packet header, packet headers have to be captured in the (sub)networks to be monitored by the whitelist. In this paper, we consider only TCP and UDP packets. The *connection creation* consists in aggregating the captured packets to *connections*. We define a *connection* as all packets with the same source/destination IP address, source/destination port and IP protocol,

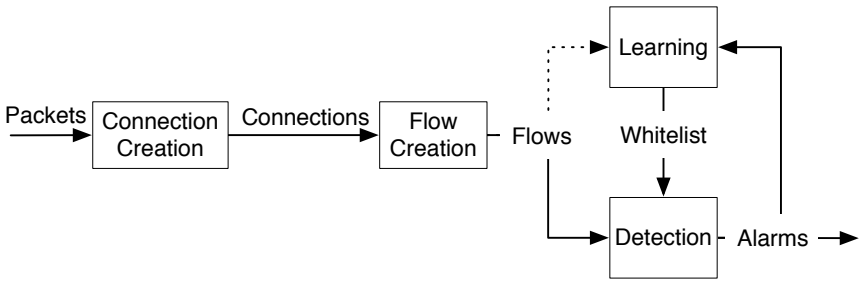


Figure 1: Outline of the flow whitelisting approach

regardless of the direction. The end of a connection is determined either using the TCP state machine or an inactivity timeout of 300s. In our experiments (see Section 3), we perform this task with the open source tool *argus*[1].

The *flow creation* step identifies the client and server sides of the connections and further aggregates the connections according to our 4-tuple flow definition given in Section 1. We sequentially apply four rules to identify the server side:

- Rule 1 applies to all TCP connections for which we observe the 3-way handshake. The server is set to be the host which received the SYN packet or sent the SYN/ACK packet.
- Rule 2 is applied if a well-known port (bellow 1024) is observed: the host using such port is set as the server[2].
- Rule 3 is a heuristic. If the same protocol and port are re-used by a host in multiple connections, this host is set as the server and we use this protocol/port combination to identify the service. We rely on the fact that client ports normally vary with each connection, and are less likely to be repeated. This rule makes it necessary to keep every connection not classified by rules 1 or 2 in memory until a second connection with a repeated host address, protocol and port is observed, potentially indefinitely delaying the analysis. For an online implementation, we would recommend the use of a timeout, after which the connection should be classified by rule 4. In this paper, we have implemented an offline analysis with an infinite timeout.
- Finally, for flows which do not match any of the previous rules, rule 4 sets the server to be the destination of the first packet observed in the connection.

Note that our rules implicitly assume that every transport port and IP protocol pair used by a server uniquely identifies a service in the SCADA network. This definition is particularly problematic with network services that use Dynamic Port Allocation (DPA), such Microsoft's Active Directory. In this service, *high ports* (above 1024) are dynamic allocated for Remote Procedure Calls [13]. We acknowledge this limitation, and discuss its effects when presenting our experimental results in Section 3.

2.2 The Learning Phase

Ideally, the network administrator knows all services deployed in the network. Therefore, the flow whitelist could be constructed from this knowledge. In practice, however, complete information is rarely available, partly due to the involved proprietary protocols.

The goal of the *learning phase* is to automatically create an initial whitelist from a given period of traffic, the *learning time*. This whitelist contains the entries for all flows observed during the learning period. We make two assumptions: (i) all flows in the learning period are legitimate and (ii) most legitimate flows can be observed in the learning period. We argue that the first assumption is valid, as anomalous or malicious events are much rarer in SCADA networks than in the Internet. In fact, no attacks were reported during the capture of our datasets. The second assumption is based on the expectation that most of the traffic in SCADA networks is automated, thus flows should be repeated fairly often. We discuss how to set the duration of the learning phase in Section 3.3.

Note that we do not expect to see *all* legitimate flows in the learning phase. For example, manual changes in the configuration of PLCs could, depending on the setup, only happen rarely, thus flows related to this activity will probably not be present in the whitelist. Hence, the whitelist can be extended by the network administrator during the detection phase.

2.3 The Detection Phase

The whitelist created by the learning phase is used in the detection phase to identify illegitimate flows. If the flow is whitelisted, then nothing happens, otherwise an alarm is raised. In a real-world deployment, an administrator would have either to add the flow that caused the alarm to the whitelist (treating the alarm as false positive) or to block it (true positive). Note that, differently to traditional IT networks, where hosts are commonly put in quarantine in case of malicious activities, an automatic blocking is *not* advised for SCADA environments, as blocking

legitimate traffic could have dire consequences, such as a blackout. This topic will be discussed further in Section 4.

3 Experimental Results

In this section, we present the four tests used in our analysis to evaluate the viability of flow whitelists in SCADA networks. In the first test, discussed in Section 3.2, we verify whether the size of the whitelist is manageable by comparing the size of the complete whitelist with the number of hosts and communicating pairs in a network.

Our discussion over the stability of the whitelists is divided in three parts. In Section 3.3, we determine the ideal *learning time* to be used in the learning phase of our approach. Then, in Section 3.4, we present the classification method used to discuss the nature of the alarms present in our datasets. Finally, in Section 3.5, we apply the classification method to provide an overview of the distribution of the number of alarms over the classes.

In our experiments, we need to simulate the administrator's intervention discussed in the detection phase (see Section 2.3). We do this by always adding the flow which caused the alarm back to the whitelist, and at the same time storing the alarm for post processing. This means that an alarm is never repeated, which allows us to focus our analysis on the nature of alarms, rather their absolute number.

3.1 Datasets

In this paper we use network packet traces collected at three different SCADA environments: two water treatment facilities and one electric and gas utility, referred to as *water1*, *water2* and *electric-gas*. At one of the water treatment facilities two data collections were performed simultaneously, one in the *field* subnetwork, consisting of programmable logic controllers (PLCs), Remote Terminal Units (RTUs) and field devices; and one in the *control* subnetwork, consisting of servers with different functions (e.g., polling of PLCs, keeping historical data and performing access control), and Human Machine Interfaces (HMI), i.e., operator workstations. In the other locations, a single collection was performed, containing all data of these two logical subnetworks. All SCADA datasets consist in full packet *tcpdump/libpcap*[3] traces and we treat each collection as a separate dataset.

For comparison, we use two additional traditional IT networks datasets. One is a publicly available *tcpdump/libpcap* trace captured at an educational organization: "Location 6" (referred to as *loc6*) from [14]. We use only a portion of the available data, approximately the first 7.5

days of the trace. The last dataset consists of 15 days of *NetFlow*[4] records collected at an internal router at a university campus, referred to as *uni*. An overview of the datasets is presented in Table 1.

Note that we cannot apply the *flow creation* step described in Section 2.1 to the *uni* dataset. NetFlow records do not contain enough information to identify which host initiated a TCP connection according to the 3-way handshake, which it is necessary for the applicability of rule 1. Instead, we use the techniques described in [5] to aggregate the NetFlow records to connections. Therefore, the client side (termed *originator* in that work), and, as a consequence, the server side and the service port of a connection are determined by these techniques.

Table 1: Datasets overview

Name	**Hosts**	**Duration**	**Packets**	**Bytes**	**Conn.**
water1	45	13 days	591M	96GB	76K
water2-control	14	10 days	26M	4GB	131K
water2-field	31	10 days	67M	24GB	215K
electric-gas	388	86 days	2G	511GB	179M
loc6	93	7.5 days	53M	53GB	264K
uni	22685	15 days	161G	126TB	1G

3.2 Whitelist Size

The first characteristic we study is whether the whitelist size is manageable. In other words, we verify if the connection matrix is sparse, i.e., the number of acceptable flows should be small in comparison to the number of possible flows.

We test this characteristic by setting the *learning time* to the full duration of each trace and count the number of flows observed. This allows us to estimate the size of a trace's complete whitelist, assuming no attacks are present in the dataset. While no attacks were reported during the capture of our SCADA datasets, malicious activities such as network scans are so common in traditional IT networks that they are most probably present in the *loc6* and *uni* datasets. We attempt to reduce this bias by only considering flows for which traffic is observed in both directions. We argue that this greatly reduces the number of observed flows caused by network scans and other types of network anomalies.

Table 2 shows the results. The column *Internal Hosts* gives the number of observed hosts located inside the monitored networks. In the column *Host pairs*, the number of communicating host pairs and in the column

Whitelist, we show the size of the whitelist. In order to make the different traces comparable, we express these metrics both as absolute values and as a ratio to the number of internal hosts (in parenthesis).

For most cases, the whitelist size for the SCADA datasets is in the same order of magnitude as the number of internal hosts, suggesting that flow whitelisting might be feasible in these environments. In comparison, the traditional IT counterparts present a whitelist orders of magnitude larger than the number of internal hosts, illustrating why the approach does not scale in these environments. Due to the excessively large size of the whitelist for the traditional IT datasets, we do not consider them in the tests performed in the remainder of this section.

Another observation is that the difference between the host pair and the whitelist ratios is not very large, meaning that in average we have one or two services per server. This means that a whitelist without the service information, which is less restrictive and, thus, considerably less secure, would *not* greatly reduce the size of the whitelist.

The only exception to the results presented here is the dataset *water2-control*, in which the whitelist size is one order of magnitude greater than the number of communicating host pairs. However, we show in Section 3.4 that this difference is mostly caused by a traffic anomaly.

Table 2: Whitelist size ratios

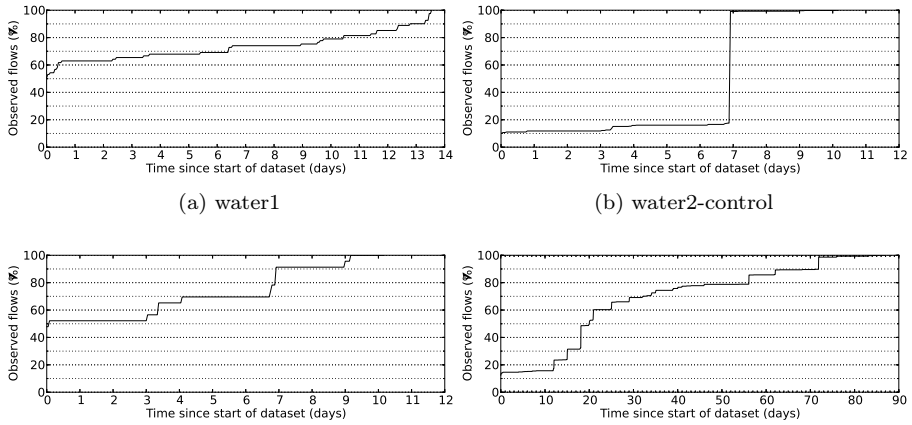
Dataset	**Internal Hosts**	**Host Pairs**	**Whitelist**
water1	51	58 (1.1)	81 (1.6)
water2-control	22	40 (1.8)	542 (24.6)
water2-field	14	20 (1.4)	23 (1.6)
electric-gas	388	542 (1.4)	1188 (3.1)
loc6	93	23 322 (250.8)	26 759 (287.7)
uni	22 685	56 425 836 (2487.4)	141 744 206 (6248.4)

3.3 Training Set Size

In the following we study the influence of the learning phase duration on the size of the learned whitelist. Figure 2 shows the size of the learned whitelist as the percentage of the total number of flows as function of the learning time. For the datasets *water1* and *water2-field*, over 50% of the flows are observed within the first hour of traffic, with a few other additions during the first day. This percentage is much lower for datasets *water2-control* and *electric-gas*, around 10% and 15%, respectively. The *water2-control* dataset shows a significant *jump* in the

list size after around 7 days. The whitelist for the *electric-gas* dataset grows steadily from day 10 to day 40. The reasons for this behavior are explained in the next section.

Despite this difference, one characteristic is shared by all SCADA datasets: no additions are made to the whitelist in the second day of traffic. In fact, almost no addition is made up to the third day in the *water* datasets, and a even longer period in the *electric-gas* dataset. Due to this observation, we set the learning time to 1 day in the following experiments.



(a) water1 (b) water2-control

(c) water2-field (d) electric-gas

Figure 2: Number of learned flows over time

3.4 Nature of Alarms

In this section, we present our post-processing analysis of the alarms. Its goal is to determine the sources of instability in the whitelists, that is, the nature of the flows not observed during the learning phase.

During our analysis we identified four main alarm classes:

- *Dynamic Port Allocation Anomalies:* As discussed in Section 2.1, our service definition assumes an one-to-one mapping with transport ports, which is problematic in cases of DPA. We did not attempt to uncover all services using dynamic ports, but we identified anomalies which are most likely triggered by it. The datasets *water2-control* and *electric-gas* present moments in which several TCP connections are made by the same hosts in short sequence,

with transport port numbers monotonically increasing on both client and server side. Table 3 shows an excerpt of such moment.

- *Manual Activity:* This class consists in human triggered flows. All flows which used 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). In addition, for datasets *water1* and *water2*, we have a list of operator workstations. If the client side of a flow is on this list, such flow is also classified as manual.

- *New Host:* This class contains all flows for which at least one host (either server or client) did not communicate during the training period and, obviously, can not be present on the whitelist.

- *Other:* A *catchall* class for all flows that do not fit any of the other classes.

We map each flow to a single class, and membership to a class is tested in the same order presented here. For instance, consider 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*, as this class has precedence over the *new host* class.

When analysing the *electric-gas* dataset, we observed two events that deserve to be studied separately. In SCADA networks, it is very common for most functions in the network to be replicated, including duplicating servers, in order to increase reliability. The first event consists in a single *redundant* host taking over tasks of one of the main servers in the network, the SCADA server responsible for polling the field devices.

Just before the change occurs, we observe *telnet* traffic to some of the PLCs, issuing a reboot command. We do not observe telnet to all PLCs, however, as all changes occur in a relatively small time interval,

Table 3: Dynamic port example

StartTime	**Proto**	**Sport**	**Dport**	**Pkts**	**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

we presume they are related. Besides the flows involving the PLCs, several other long-lived flows present the same behavior, for example, some *ssh* flows are also "switched" to the *redundant* host. According to the operators, changes like this one are routinely performed in order to verify if the redundant hosts work properly.

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

The second event consists in the *relocation* of many hosts in the network, mostly PLCs. At times, a continuous range of IP addresses have their address changed to (logically) separated subnetworks. For example, all hosts in the IP address range X.Y.Z.61 to X.Y.Z.71 have their addresses changed to the range X.Y.A.61 to X.Y.A.71. We observe *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 subnetwork was split in several smaller ones. After the change, the logical address better represents the geographical location of the hosts.

We identify flows belonging to this event simply by verifying if either host in the flow (client or server) is part of one of the newly created networks. Its important to note that we do not classify the *telnet* access to these hosts leading to these events as part of them. *telnet* connections are always classified as *manual activity*.

3.5 Frequency of Alarms

We apply our classification method to all SCADA datasets to provide an overview of how frequent each class of alarm is. As discussed in Section 3.3, the *learning time* is set to be the first day of the dataset. The results of the classification are presented in Table 4. This table presents the absolute number of alarms and approximate percentages for each class. The results for the *electric-gas* dataset are broken down in the two events discussed in the previous section.

In the *water1* dataset, the *new host* alarms consist in a few short *snmp* connections, likely due to testing; one *ntp* connection that seems to repeat once a week; and one real anomaly: several single packet TCP connections attempts at port 1010. The single *other* flow seems to be caused by DPA; a few moments before it starts, a flow involving the same hosts, but different server port, ends. Finally, a few *http(s)* and *x11* connections and a connection originated from a operator's workstation compose the *manual activity* alarm class.

Table 4: Alarm breakdown

Dataset	Dyn. Ports	Manual	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%)

The leading cause of alarms in the *water2-control* dataset is a DPA anomaly, being responsible for around 91% of the alarms. This anomaly is responsible for the majority of the flows which compose the *jump* present at Figure 2b. In fact, if we remove the flows generated by this anomaly, over 60% of the flows would be present in the whitelist (i.e., be observed in the training period), much like the other *water* datasets. In addition, the ratio between the size of the whitelist and the number of internal hosts would be considerably smaller, 4.7 instead of 24.6, thus in the same order of magnitude as the other SCADA datasets.

In the *water2-control* and *water2-field* datasets, most *new* and *other* alarms involve a server which, according to the network administrator, relates to user authentication and thus, probably are generated due to manual activity. An unexpected behavior is that some connections are made from this authentication server, which is in the control network, directly to PLCs, which are in the field network. According to the network administrator, this type of connection is not allowed. All connections from the control network to the field should be done through a specific server. The remaining alarms involve hosts foreign to the control and field networks where the data collection was performed. It is not clear if these connections should be allowed.

In comparison to the other datasets, the *electric-gas* dataset contains a considerably larger number of alarms: 1037 of the flows are not observed in the training period. Like *water2-control*, the largest class is DPA anomalies, accounting for 35% of the total. The redundant and relocation events are responsible for over half of the *other* and *new* alarms, respectively.

Figure 3, shows a time series for the alarms observed daily, broken down per alarm class. We show only the most active period. DPA connection bursts happen at 4 distinct times (one not shown), accounting for the largest peaks. The *redundant* event happens on day 15, and a

portion of the *manual* alarms present on the same day represent the *telnet* connections used to reboot the RTUs. Interestingly, a larger peak classified as *redundant* appears before, on day 11. All flows in this peak represent single packet connections, sent by the *redundant* host to several RTUs. This was probably a test or caused by a configuration mistake.

Peaks of *manual* activity happen on days 20, 25 and 29. On each of these days a large portion of the address space is accessed, for various reasons. For instance, connections are configuring hosts for the first time, some which appear later as *new*, around days 35 and 40.

Most of the hosts are relocated around day 55 and 61. Note that no peak of manual activity happens around theses days. The *telnet* connections used to change the hosts' addresses were previously accounted for in the peaks of manual activity.

These peaks account for the majority of alarms in the dataset. The remaining alarms consist mostly 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, which might be caused by DPA.

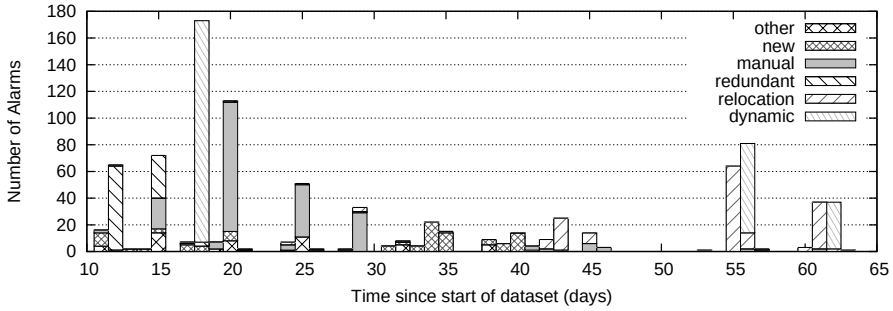


Figure 3: Time series for alarms in *electric-gas* dataset

4 Discussion

In this section we discuss some of the practical issues network administrators will face when implementing flow whitelists in real-world environments.

Dynamic port allocation (DPA): By far, the largest class of alarms identified in our analysis is due to DPA anomalies, and we only identified a portion of the port and protocol pairs used by these services. In practice, there are more connections using DPA in our datasets. For a flow level whitelisting approach to work with this type of service, it

is necessary to whitelist the whole range of transport ports that might be used by these service. This is not an ideal solution, as it makes the whitelist more permissive.

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

Dealing with real-world attack scenarios: Our datasets contained no attack data, so we could not test the efficacy of whitelisting against realistic attack scenarios. We use a list of real-world attack types presented in a previous work [15] to motivate how these attacks could be observed. We consider four types of attacks.

The first type is formed by *information gathering attacks*, such as network scans. These are normally performed by injecting several requests into the network, with the objective of discovering with services and/or hosts are available. At the flow level, these attacks not much different from the DPA anomalies identified in this work. Therefore, they should be easily identified by our approach, as non-whitelisted connections are likely to be made. The other three types of attacks are: *denial of service attacks*, which prevent a legitimate user to access a service or reduce its performance; *network attacks* used to manipulate the network protocols; and *buffer overflow attacks*, which attempt to gain control over a process or crash it by overflowing its buffer. These would only be observed if attempted from a host which is not allowed to access a given server/service or if they targeted a server/service not existent in the network.

In general, an attack will only remain undetected in two situations. Either the whitelist is incorrectly constructed, i.e., it contains entries representing illegitimate traffic, or the attacker misuses whitelisted traffic, e.g., an operator machine, normally used to access a PLC, sets an invalid parameter. In the later 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 attempting to masquerade a legitimate flow. Protection mechanisms against such attacks are discussed in [4].

Blocking or Flagging: In a traditional IT environment is a common practice to take a host offline in case it is suspected to be under attack. This is done to limit the impact of the attack, and prevent a possible spread. However, taking a SCADA host offline might have dire consequences, as a critical process might depend on it.

The same reasoning can be applied to blocking traffic, the cost of false positives might be too great. Whitelists, as any other systems can suffer from configuration problems. In our analysis, we observed a number of alarms due to rare activities, such as manual access to PLCs and hosts switching to backup servers (or being accessed by backup clients), which might be overlooked while building an whitelist. We recommend that, when whitelists are first deployed in a real world scenario, connections that are not whitelisted should only be flagged (raise an alarm). The decision of blocking the traffic or add it to the whitelist is left to the network administrators. Only after they are confident that the configuration mistakes are solved, they should consider using the whitelist to automatically block traffic.

Limitation of the learning step: Many alarms are the effect of a limitation of the technique used for learning the initial whitelist. These represent connections that do not happen regularly, and it would be impractical to extend the *learning time* in order to include them. The larger the *learning time*, the larger is the chance of including an anomalous flow to the whitelist.

In addition, some alarms are caused by the presence of new hosts, not observed in the learning step. Although changes in the topology are not common, they should be taken into consideration when deploying our 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 would require a similar update after a change in the network, as the "normal" behavior has changed.

The limitation of the learning step shows that the network administrator's (and/or SCADA vendor's) knowledge is necessary to build a complete flow whitelist. However, relying only on this knowledge can also be dangerous, as mistakes are likely to happen. For instance, the addition of flows representing backup servers connections or infrequent *ssh* connections might be overlooked. Presenting a list of flows learned from network measurements as proposed in the *learning* phase, could help administrators in identifying acceptable flows that might otherwise be missed.

5 Conclusions

In this work, we present an approach for flow whitelisting in SCADA networks. Our study shows that it is a practical solution to reduce the

number of attacks that a SCADA network is exposed to. The size of the whitelist is manageable, considering the number of internal hosts. Besides, the whitelists are fairly stable. In most cases, over 50% of the acceptable flows can observed within one day of measurement.

Services using dynamic port allocation are the main cause of alarms. These alarms can be eliminated by adding to the whitelist the complete range of ports that can be allocated by these services, or by removing them from critical segments of the network. Most of the remaining alarms are caused by a limitation the approach used to construct the whitelists, which, in real-world implementations, would be overcome by employing the knowledge of network administrators and SCADA vendors when creating them.

In future work, we plan to develop an user interface to aid the creation of the whitelists and facilitate its manipulation by providing more detailed information about alarms. In addition, we want to investigate how to improve security by identifying intrusion attempts that (mis)use whitelisted flows.

Notes

1. http://www.qosient.com/argus/
2. In the case of Active FTP, where the originator of a data connection is the server, we set the source port (20) as a service port. In the case of protocols which use the same (well-known) port on both hosts, e.g. NTP, we either use rule 1 or 4 for classification.
3. www.tcpdump.org
4. www.cisco.com/go/netflow

References

[1] S. Cheung, K. Skinner, B. Dutertre, M. Fong, U. Lindqvist, and A. Valdes, *Using Model-Based Intrusion Detection for SCADA Networks in Proceedings of the SCADA Security Scientific Symposium*, 2007, pp. 1–12.

[2] Digital Bond, Quickdraw SCADA IDS [Online]. Available: `http://www.digitalbond.com/tools/quickdraw/`

[3] D. Hadžiosmanović, D. Bolzoni, and P. H. Hartel, *A Log Mining Approach for Process Monitoring in SCADA*, International Journal of Information Security, vol. 11(4), pp.231–251, 2012.

[4] C. L. Abad and R. I. Bonilla, *An Analysis on the Schemes for Detecting and Preventing ARP Cache Poisoning Attacks*, in 27th International Conference on Distributed Computing Systems Workshops (ICDCSW'07), IEEE, 2007, pp. 60–68.

[5] R. Sommer and A. Feldmann, *NetFlow: Information Loss or Win?* Universität des Saarlandes, Saarbrücken, Germany, Tech. Rep., 2002.

[6] Y. Cao, W. Han, and Y. Le, *Anti-Phishing Based on Automated Individual White-List*, Proceedings of the 4th ACM workshop on Digital identity management - DIM '08, pp. 51–60, 2008.

[7] D. Erickson, M. Casado, and N. McKeown, *The Effectiveness of Whitelisting: a User-Study*, in The Fifth Conference on Email and Anti-Spam - CEAS '08, Mountain View, California, USA, 2008.

[8] M. Yoon,*Using Whitelisting to Mitigate DDoS Attacks*, on Critical Internet Sites Communications Magazine, IEEE, vol.48(7), pp. 110–115, 2010.

[9] E. Y. Chen and M. Itoh, *A Whitelist Approach to Protect SIP Servers from Flooding Attacks*, 2010 IEEE International Workshop Technical Committee on Communications Quality and Reliability - CQR '10, pp. 1–6, Jun. 2010.

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

[11] K. A. Stouffer, J. A. Falco, and K. A. Scarfone, *NIST SP 800-82. Guide to Industrial Control Systems (ICS) Security: Supervisory Control and Data Acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controllers (PLC) Gaithersburg*, NIST Special Publication, United States, 2011.

[12] R. R. R. Barbosa, R. Sadre, and A. Pras, *Difficulties in Modeling SCADA Traffic: a Comparative Analysis*, in Proceedings of the 13th international conference on Passive and Active Measurement - PAM '12, pp. 126–135, 2012.

[13] Microsoft, *Service Overview and Network Port Requirements for Windows*, [Online]. Available: `http://support.microsoft.com/kb/832017`

[14] R. R. R. Barbosa, R. Sadre, A. Pras, and R. Meent, *Simpleweb/University of Twente Traffic Traces Data Repository*, University of Twente, Netherlands, 2010. [Online]. Available: `http://doc.utwente.nl/71273/`

[15] R. R. R. Barbosa, R. Sadre, and A. Pras, *Towards Periodicity Based Anomaly Detection in SCADA Networks*, in IEEE 17th Conference on Emerging Technologies & Factory Automation - ETFA 2012, Kraków, Poland, 2012.