Usable Global Network Access Policy for PCS

David M. Nicol
William H. Sanders
Sankalp Singh
Mouna Seri
Information Trust Institute, and Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

1 Introduction

With the nation's increasing awareness of the importance of securing the computer/communication infrastructure of process control systems (PCS), recommendations for their configuration have appeared, most notably NIST Special Publication SP-800-82 [1]. Some of these recommendations concern network connectivity, specifically describing near isolation of some subnetworks from the PCS. Security mechanisms that formalize these kinds of notions are known as network access control policy. However, while recommendations like those in NIST SP-800-82 refer to rules that ought to be enforced globally over an entire network, in practice the enforcement of network access control is distributed among firewalls that enforce some *local* policies, defined by their rule sets.

We are interested in two related problems. First, how can English language recommendations for global policy be expressed in a machine-checkable form that is *useable*, in the sense that typical administrators of PCS networks can easily formulate and understand it? Next, how can we determine whether the access control provided by the firewalls precisely meets the requirements of the machine-checkable global policy? Automation is crucial, for implementation of mechanisms to protect a PCS is very detailed, very complex, and easily admit to human error in its formulation and verification. This paper first outlines characteristics of common best practices recommendations, and discusses what is needed to formally express and automatically check those recommendations. We then describe a software tool—the Access Policy Tool—designed to express and check such policy.

Concern for protecting critical infrastructure is leading to exploration of many research paths. Network access control is but one aspect of a problem with many components. The work we discuss should be viewed as supporting development of technologies to be used in the context of more comprehensive visions for critical infrastructure protection, such as [2].

2 Global Network Access Policy

Access control in a PCS occurs at a host, and on the network. PCS systems tend to use only normal login procedures for access to hosts; more sophisticated mechanisms (e.g., role-based access control) can easily be imagined, but that lies in the future. By contrast, large installations routinely use firewalls to protect network access. Global access control policy languages are vehicles for expressing—formally, but at a high level—a system's security requirements with respect to access. There is an active literature on such languages e.g., [3, 4, 5, 6]. Problems considered include detecting inconsistencies in policy specification, and the automatic creation of correct rule sets to reside in the devices that enforce policy. Very little of that work has been used in real PCS contexts, in part because it is largely academic, in part because real systems have more heterogeneity of devices than is typically assumed, and in part because PCS system operators simply don't know about such tools.

Policy statements involving what computers or networks must or must not do are generally implemented using localized checks. Any attempt to make an access is compared with rules that govern access to decide whether to grant it or not. Access rules for data security or platform security may be configured in a computer's operating system; rules governing communication may be part of firewall systems. At this level the rule specifications are very specific, very detailed, and there tend to be many of them. Any individual rule is an implementation of part of a higher level policy.

Given the complexity of computer networks, guidelines for operators on securing PCS are necessary. The last few years have seen the development of such *best practice* recommendations by committees of experts. Best practices for PCS security are stated in ordinary written languages. If compliance with best practice recommendations is to be automatable, the recommendations must be transformed into rigorous machine-checkable global policy language, with some assurance of having captured the intended semantics correctly.

A focused effort on providing standards and best practices for PCS security has resulted in draft versions (the latest published in September 2007) of NIST special publication 800-82, "Guide to Industrial Control Systems (ICS) Security". This document is extensive, highlights aspects of ICS systems that require particular attention, and extensively refers to other documents that chronicle best practices in more general contexts, such as cryptography, network security, and firewall configuration. Idaho National Labs external report #INL/EXT-06-11478, "Control Systems Cyber-security : Defense in Depth Strategies" [7] covers some of the same ground, with a focus on network architecture for providing layers of protection. A key suggestion for protection in these reports (and others like them) is that the electronic connection between a PCS and the enterprise system that encompasses it be tightly constrained, and thoroughly understood.

English language description of best practice recommendations is ideal of course for conveying the concepts, even in some detail. But it is not possible with today's technology to have a computer program check the compliance of an installation against a list of best practice suggestions written in English. What we can hope for though is to create a formal best practices specification that is precise enough to be machine checkable, is expressive enough to capture these best practice recommendations, but still is *useable* by ordinary network administrators.

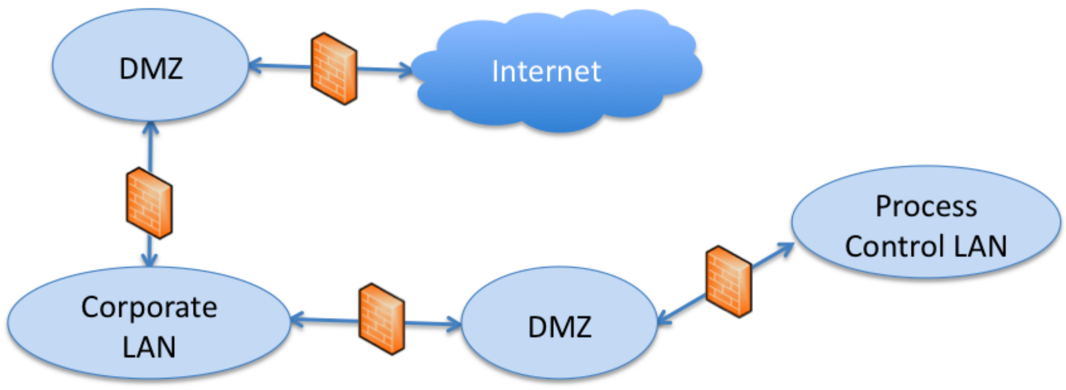


Figure 1: Recommended Architecture for Process Control Networks

3 Best Practices Recommendations

NIST special report SP-800-82 goes into significant detail on how Industrial Control Systems ought to be architected, and how their firewalls ought to be configured. Other best practices recommendations are quite similar. The key to the recommended strategy is separation. The corporate network ought to protect itself from the general Internet through a "Demilitarized Zone" (DMZ) where the corporation's electronic public face can be placed (e.g. web servers, mail servers). The DMZ is isolated from the Internet using a firewall, and is isolated from the Corporate LAN by another firewall. The corporate LAN has need of data provided by the process control LAN, but the two ought to be separated by another DMZ. Figure 1 gives a high-level illustration of the architecture.

Best practices recommendations go on to describe limitations on how communication between applications in the different LANs ought to constrained, on the kinds of rules that ought to be in the firewalls, and on exposure of different networks to the Internet. A representative sampling from NIST SP-800-82 follows:

(a) The base rule set should be deny all, permit none.

(b) All "permit rules should be both IP address and TCP/UDP port specific.

(c) Traffic should be prevented from transiting directly from the control network to the corporate network. All traffic should terminate in the DMZ.

(d) Outbound packets from the control network or DMZ should be allowed only if those packets have a correct source IP address that is assigned to the control network or DMZ devices.

(e) Any protocol allowed between the control network and DMZ should explicitly NOT be allowed between the DMZ and corporate networks (and vice-versa).

An important question asks whether it is possible to *automatically* determine whether an installation adheres to such practices. It is easy to imagine writing a program that analyzes firewall rule-sets to check recommendations (a) and (b). Recommendation (a) is checked by seeing whether the last rule in the set (the default) is "deny all"; checking recommendation (b) is obvious. However, there are subtleties to checking the other recommendations. Consider recommendations (c) and (d). Firewall rules can be used to enforce these restrictions, but if the system employs host-based access control and authentication to the firewalls, the host operating systems within the process control network may put these limits on outbound traffic, and through authentication prohibit a successful intruder from generating traffic from within the network that violates the policy. In principle these recommendations are machine-checkable, but the analysis may need to extend beyond just the firewall rule-sets. Recommendation (e) is even more problematic in that enforcement may *have to* occur at hosts. If the process control network is writing into the DMZ using Modbus carried on TCP/IP, an ordinary firewall won't "see" Modbus, only TCP. In the general case, unless a firewall does deeper inspection into packets, we cannot construct firewall rules to enforce this role. However, under operating systems such as SeLinux it *is* possible to enforce limitations on which applications (and consequently which protocols) are allowed access to the network. In principle then this rule is statically machine checkable—meaning that compliance can be determined by analysis of locally enforced rules—provided either that the firewalls have deep packet inspection, or that the hosts run local access control policies. If such mechanisms do not exists, it is conceivable that an on-line monitoring system that does deep packet inspection might look for violations of this rule. Run-time monitoring is an important topic, but is outside the scope of this note.

Other best practices recommendations concern limitations on well-known protocols such as DNS, TFTP, SMTP, and HTTP. These are typically carried on TCP/IP, but also typically use well-known port numbers in the destination address. A recommendation that no SMTP bearing traffic be admitted to the PCS network might be checked by looking at the port numbers on admitted traffic. Likewise, the recommendation that the process control network not post DNS queries can be implemented by a firewall denying egress traffic with the DNS port number in the destination. These checks are not rock-solid though, for they cannot detect if a device that intrudes in the network (or a compromised device in the network) uses these protocols, but through alternative ports.

Just from this sampling of recommendations we see that compliance with some recommendations may be checkable just by looking at individual firewalls' rule set, some may require a joint analysis of multiple firewall's rule sets, and others may require existence of host-based mechanisms that constrain access to the network and the analysis of their rules as well. Given that automated compliance checking is possible, the next question to consider is a mechanism for expressing global access policy. The mechanism needs to be *useable*, in the sense that the transformation of best practices recommendations into formal form needs to be readily apparent to a PCS system administrator, needs to be automatically checkable, and needs to enable computationally efficient verification. We turn next to how we have addressed this problem.

4 Access Policy Tool

We have built *the Access Policy Tool* (APT) to address specification of global access policy, and verification that the implementation adheres to that specification. In APT one describes a network, describes global access policy, and verifies whether the implementation (as expressed in firewalls distributed throughout the network) implements the global access policy, exactly. APT can analyze networks with heterogeneous mixtures of fireware brands and models, and deals with sophisticated firewall features such as authentication. We have also begun to integrate rule-sets from host-based access control mechanisms such as SeLinux, and the Java Security Manager.

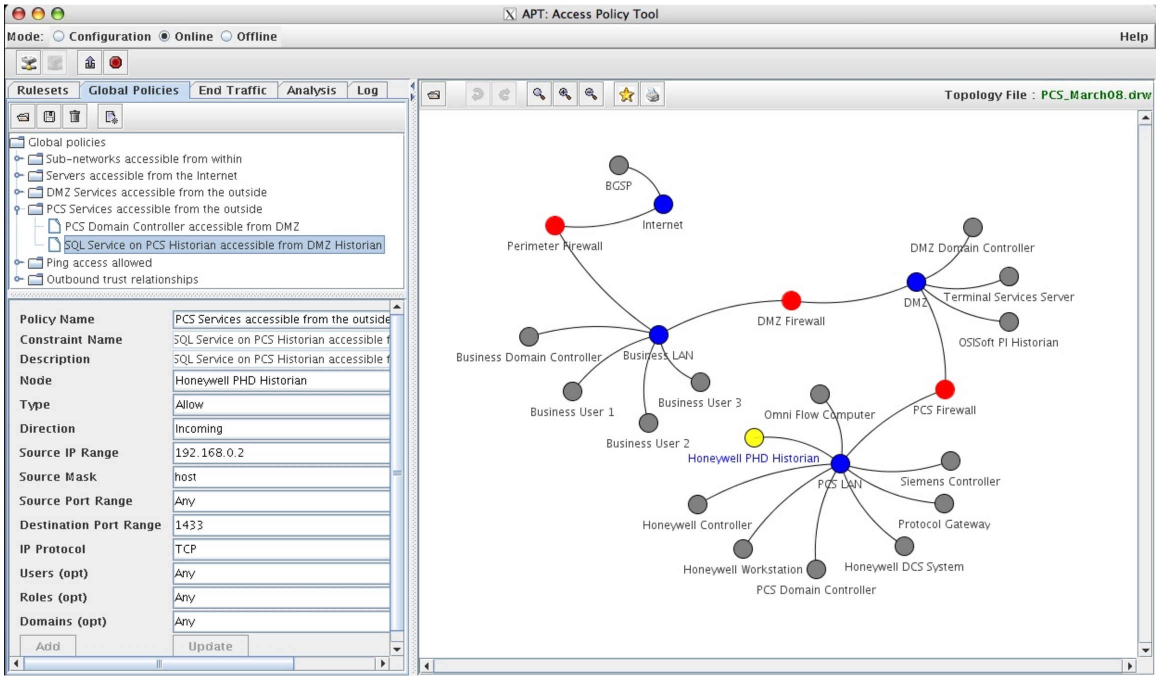
APT has a graphical interface which is used to browse the firewall rules, run the verifier, and display the results. Expression of global access policy in APT is tied closely to APT's graphical view, and expectation that its user is familiar with firewall rules and networking.

4.1 Global Policy in APT

Figure 2 provides a screenshot of APT applied to a representative PCS. Elements of the network are represented graphically on the right. Here gray nodes represent host devices, red nodes represent firewalls, and blue nodes represent subnetworks of devices. A node highlighted in yellow is one for which information is provided on the left. On the left we find a pane that describes global policy rules. Other selectable panes display information on network connectivity, the results of an APT verification, and rulesets within the firewalls. The global policy pane presents global access rules, organized by descriptive labels provided by the policy designer. In the example we have elected to look at one global access rule, in the grouping of those related to PCS services that are to be made available to the DMZ. The rule selected and illustrated describes a "constraint" on the PHD Historian within the PCS LAN. The constraint has a notion of entity to which it is bound (here, the PHD Historian); it has a notion of source addresses—origins of traffic to be affected by this constraint, a destination (here the host with which the constraint is associated), a direction of traffic flow (here "incoming", which means the host is a destination for the contrained flow), protocols (those observable in the IP packet header) and whether the constraint is to allow or deny the flow. Other specifics concerning the constrained flow include port numbers on source and destination, protocol used to carry the traffic, and information needed for authentication, if used. The illustrated rule specifies that source 192.168.02 (which is mapped to the PI Historian in the DMZ) must be able to connect (through any of its source ports) to port 1433 (SQL) on the PHD Historian.

This representation of the rule describes the source and destination as IP addresses; the mapping from rule specification to rule implementation (with IP addresses) is automatic. At the time the rule is created the source and destinations are selected (by pull-down menu) as nodes in the graphical display. One selects the node, or the logical complement of the node (the reason for which will become apparent shortly); the set of allowed (or disallowed) protocols is also selected. A constraint looks very much like a firewall rule, with the exception that we will allow a greater degree of freedom in specifying the source and destination sets, and the protocols. The system administrator using APT should have no problem understanding the structure or meaning of a constraint. Indeed, it is not far from the truth to think of the global policy expression in APT as being a set of rules in a sort of global virtual firewall situated between all sources and all destinations.

Now consider how APT constraints can capture the best practices recommendations enumerated earlier. Recommendation (a)—default deny—is covered by APT's default assumption of "deny"—any flow that is not explicitly allowed is assumed by the global policy to be denied. Recommendation (b)—specific addresses on permit rules—can be implemented by constructing an individual constraint for each permitted connection; here the source and destination sets would be singleton sets. However, since this constraint concerns more the form of a firewall rule than its content, re-expression of firewall rules as APT constraints is unnecessary. It is better to run a separate script that finds any "permit" rules that are not address-specific. Recommendation (c)—traffic



Mode: Configuration [✓ Online] Offline

Global policies
- Sub-networks accessible from within
- Servers accessible from the Internet
- DMZ Services accessible from the outside
- PCS Services accessible from the outside
  - PCS Domain Controller accessible from DMZ
  - SQL Service on PCS Historian accessible from DMZ Historian
- Ping access allowed
- Outbound trust relationships

Policy Name	PCS Services accessible from the outside
Constraint Name	SQL Service on PCS Historian accessible f
Description	SQL Service on PCS Historian accessible f
Node	Honeywell PHD Historian
Type	Allow
Direction	Incoming
Source IP Range	192.168.0.2
Source Mask	host
Source Port Range	Any
Destination Port Range	1433
IP Protocol	TCP
Users (opt)	Any
Roles (opt)	Any
Domains (opt)	Any

Figure 2: Screenshot of Access Policy Tool

terminates in DMZ—is expressed with a constraint on firewalls that border a DMZ, with the direction specified as being into the DMZ. That constraint denies all flows whose source is within the originating network, and whose destination is the *complement* of the set of destinations in the DMZ. Recommendation (d)—no source address spoofing—is expressed by a deny constraint bound to the control network, with the direction being control network to DMZ. The source set is the *complement* of the set of all addresses within the control network, the destination is the set of all addresses. The binding of the constraint to the control network is important here, because the constraint needs to be able to differentiate between the network, and the source identities of outbound traffic. Recommendation (e)—different protocols are used to target the DMZ—does not yet have expression in APT constraints. We are addressing this and related problems as we research how best to integrate diverse host-based local access control rules into the APT analysis engine, and APT expression of global policy.

It is possible for a flow to have multiple rules apply to it, and have conflicts in the rule outcomes. Coupled with a good conflict resolution mechanism, this is actually quite a useful tool for compactly specifying desired policy. For example, imagine that an installation wants to follow the best practice recommendation which has traffic from the control network and from the corporate network terminate in the DMZ, *except* for one specific application where it is desired that a host *control-database* in the control network be able to upload data to a host *corporate-database* in the corporate network. This is accomplished by creating two constraints. The first implements the recommendation as we have already described—this is a "deny" rule. The second is a "permit" rule that names *control-database* and the specific port used as the source set, names *corporate-database* and the specific port used as the destination set, binds the constraint to *control-database*, makes the direction "outgoing", and names the protocol (e.g., TCP) used to carry the connection. The union of these two rules does what we want, so long as the conflict resolution mechanisms applies the second rule to the specific flow of interest, and the first rule to all others coming out of the control network.

APT resolves rule conflicts using the *principle of greatest specificity.* This is a principle known to system administrators who manage routers, where if there are multiple forwarding rules that apply to the destination of an incoming packet, the rule with most significant prefix bits ("longest prefix") is the one chosen. Applied to our context, a rule that in all ways is more narrowly applicable to sources, destinations, and protocols than another ought to take precedence over it, because it is more specific in all of its descriptive details. Clearly this is exactly the case in the example. A small bit of notation helps to make this idea more precise for the general case.

We associate with every global policy rule a set of network sources $\mathcal{S}_s$, and a set of network destinations $\mathcal{S}_d$. Each element of these sets is a host-port pair, i.e., semantically equivalent to the source address of an IP packet. There is a constraint type from $\{$*deny,permit*$\}$, and a set of protocols $\mathcal{Q}$. Elements of $\mathcal{Q}$ are those protocols defined for inclusion in IP packet headers.

For the example under discussion, let $\mathcal{C}$ be the set of all possible sources in the corporate network, $\mathcal{D}$ be the set of all destinations in the DMZ, and $\mathcal{P}$ be the set of all possible sources in the process control network. We state that traffic out of the process control network ought to terminate in the DMZ by defining $\mathcal{S}_s = \mathcal{P}$, $\mathcal{S}_d = \overline{\mathcal{D}}$, set the constraint type to *deny*, and set $\mathcal{Q}$ to the set of all protocols. A rule for the special case flow identifies $\mathcal{S}_s$ with *control-database* and the port it uses for the flow, identifies $\mathcal{S}_d$ with *corporate-database* and the port it uses, sets $\mathcal{Q}$ to the allowed protocol and defines the constraint type to be *permit.* This second rule is more specific than the first because its source set is a subset of the source of the first rule, its destination set is a subset of the destination set of the first rule, and its protocol set is a subset of that of the first rule. It is straightforward to detect this, and apply the more specific rule for conflict resolution.

The principle of greatest specificity is stated as follows.

Definition 1 *Let* $R_1 = (\mathcal{S}_s, \mathcal{S}_d, \mathcal{Q}, a)$ *and* $R_2 = (\mathcal{S}'_s, \mathcal{S}'_d, \mathcal{Q}', a')$ *be two global policy rules, where* $a \in \{deny, permit\}$. R_2 *is* **more specific** *than* R_1 *if* $\mathcal{S}'_s \subseteq \mathcal{S}_s$, $\mathcal{S}'_d \subseteq \mathcal{S}_d$, $\mathcal{Q}' \subseteq \mathcal{Q}$, *and at least one of these inclusions is strict. Then for any flow to which* R_2 *applies, action* a' *is taken.*

The principle of greatest specificity implies that if there is a flow for which multiple rules apply, and among these rules there is one (say R) which is more specific in all particulars than every other rule in that set, then the conflicts are resolved and rule R is applied. The principle of greatest specificity will not resolve all conflicts; if an APT global policy has unresolved conflicts then the user needs to know what those conflicts are, and the characteristics of traffic flows that are governed by conflicting rules. APT performs this consistency check, allowing one to define a consistent set of rules prior to performing an analysis that checks the implementation against the policy.

The proposed formulation addresses the problem of useability by being simple—much simpler than more general global policy languages—yet reasonably expressive for access control. Importantly, we believe that its simplicity allows for a fairly straightforward translation of best practices descriptions written in English into a machine checkable form. This simple step closes what appears to us to be a critical gap in bringing expert's recommendations to the field for use.

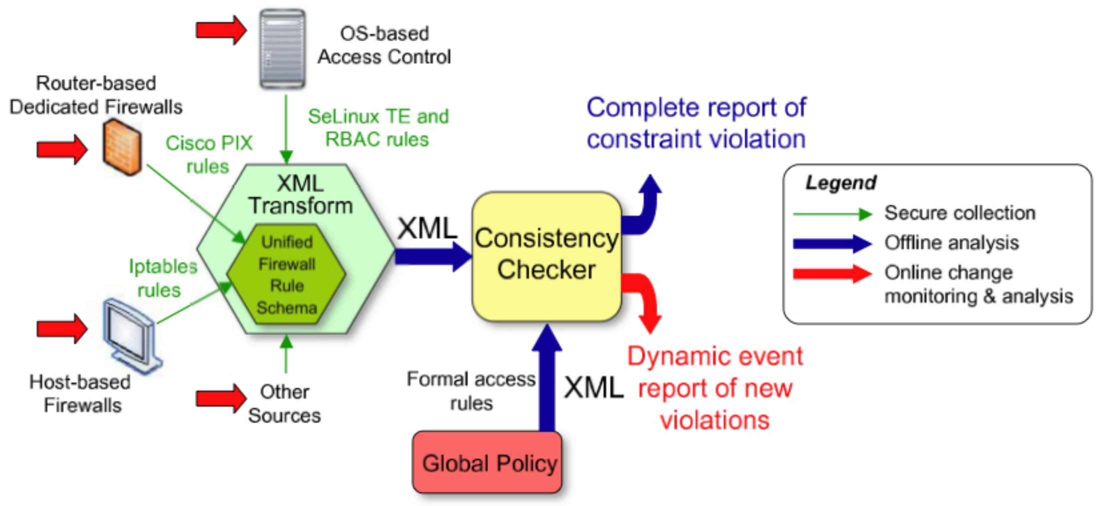

OS-based Access Control
Router-based Dedicated Firewalls
SeLinux TE and RBAC rules
Cisco PIX rules
XML Transform
Unified Firewall Rule Schema
Iptables rules
Host-based Firewalls
Other Sources
XML
Consistency Checker
Complete report of constraint violation
Legend
Secure collection
Offline analysis
Online change monitoring & analysis
Dynamic event report of new violations
Formal access rules
XML
Global Policy

Figure 3: Access Policy Tool Engine

4.2 Policy Checking in APT

APT analyzes a network for compliance with a global policy. Figure 3 illustrates the design. Rule-sets are gathered from a heterogeneous mixture of devices, and operating system-based rules; the network topology is likewise gathered. The firewall rules are transformed into a unified XML schema, and the network topology is expressed in an XML schema. These are feed to a consistency checker which also has as input XML expression of global policy. The checker performs an analysis, and reports on the violations observed.

The checker's computation builds a data structure that is based on network's topology, the placement of firewalls in that topology, and the rulesets in those firewalls. It explicitly allows for the possibility of host-based firewalls, such as those embedded on a network interface card. Conceptually, APT uses this data structure to observe how a flow is treated by the implementation. It computes the flow's path from its source, through the firewalls and routers, towards its destination. Either the flow is inhibited at some firewall, or the flow is delivered to its destination. In either case the implementation has delivered a policy judgement, to allow or to deny the flow. Assuming that the global policy is consistent, if multiple policy rules apply to the flow then one of them is more specific than all the others. APT identifies that rule and compares its policy statement (e.g., require, or deny) to the decision rendered by the implementation. Policy violation occurs when the policy statement conflicts with the implementation's judgement.

APT is highly optimized; rather than consider every possible flow individually (as might be inferred from the description above), it rather analyzes every possible flow *equivalence class*, where all flows in the class follow the same path through the network, and are judged identically by the implementation. Sophisticated data structures and algorithms lay behind this approach.

At the end of an analysis, APT reports a list of all flows where the implementation violates global policy. For each such violation it identifies the firewall rules involved in handling that flow, and does an aggregate analysis of the rules involved in policy violations to produce a list of rules ordered by the number of times they were involved in a violation. This list has proven to be a valuable aid in debugging an implementation, by focusing the analyst's attention on the rules most likely to directly cause the violations.

APT is presently in beta test, driven by real rule sets from industrial partners' installations. Current work on the tool is focused on improving the user interface, and dealing with issues of scale. More information about APT is available at our web-site [8].

5 Conclusions

Sophisticated technology for system analysis and verification simply will not be willingly used in the PCS context if it is not easy to use. Easy to use means that to a person who is expert in PCS administration (but not in formal security policy languages) there is a straightforward mapping from best practices recommendations to global policy specification, and from global policy specification to what protection is provided. We have shown how best practices recommendations might be mapped in a straightforward fashion to global policy rules in our software tool APT. We've shown that the mapping is useable in the sense of appealing to domain experience one may expect from the system administrator of a PCS. APT then analyzes the network configuration for compliance with the global policy it is given, characterizes any flows that violate the policy, and identifies local access control rules most involved in the violations. APT is presently in beta test in PCS installations; our intention is to make it available for broader use shortly.

References

[1] K. Stouffer, J. Falco, and K. Scarfone. Guide to industrial control systems (ics) security. Technical Report SP-800-82 (Second Public Draft), NIST, September 2007.

[2] Miguel Correia Paulo Veríssimo, Nuno Ferreira Neves. Crutial: The blueprint of a reference critical information infrastructure architecture. In *Proceedings of the 1st International Workshop on Critical Information Infrastructures @ ISC06*, August 2006.

[3] S. Barker and P.J. Stuckey. Flexible access control policy specification with constraint logic programming. *ACM Trans. Inf. Syst. Secur.*, 6(4):501–546, 2003.

[4] G. Zanin and L. Mancini. Towards a formal model for security policies specification and validation in the selinux system. In *SACMAT '04: Proceedings of the ninth ACM symposium on Access control models and technologies*, pages 136–145, New York, NY, USA, 2004. ACM Press.

[5] P. Griffin. Introduction to xacml. http://dev2dev.bea.com/pub/a/2004/02/xacml.html.

[6] N. Dulay, N. Damianou, E. Lupu, and M. Sloman. A policy language for the management of distributed agents. In *AOSE*, pages 84–100, 2001.

[7] D. Kulpers and M. Fabro. Control systems cyber security : Defense in depth strategies. Technical Report INL/EXT-06-11478, Idaho National Laboratory, May 2006.

[8] D. Nicol, W. Sanders, S. Singh, and M. Seri. Access policy tool, 2008. http://www.perform.csl.uiuc.edu/apt/.