Experiences Validating the Access Policy Tool in Industrial Settings

David M. Nicol, William H. Sanders, Mouna Seri, Sankalp Singh
Information Trust Institute
University of Illinois at Urbana-Champaign
Urbana, IL
{dmnicol,whs,seri,sankalp}@illinois.edu

***Abstract*—The Access Policy Tool (APT) analyzes the firewall configuration in an enterprise network for compliance with a global access policy, such as one that describes the industry's best practices. APT is the culmination of several years of academic research and development; in this last year, we have been working with industrial partners in the energy sector to validate the results of the tool. More importantly, through this interaction, we've learned of significant requirements for tools like APT that must be met for real industrial use, and have incorporated the changes needed in APT to meet those requirements. This paper describes our experience with validating APT, and documents its performance on systems of the scale on our industrial partners.**

***Index Terms*—access control; firewalls; security policy**

I. Introduction

Firewall configuration in an enterprise network is error-prone, as shown by Wool [31]. A small change in a firewall's rule-set can inadvertently create a large hole in the network defense. The problem is exacerbated in industrial control networks that employ different network partitions and demilitarized zones through the use of multiple firewalls. For such systems it is critical to ask what the intended access policy into and throughout the network ought to be, and whether the firewalls are configured to provide that access and only that access. The former question is addressed in part by best practices recommendations (e.g., NIST Special Report SP-800-82 [30]), emerging standards (e.g., NERC CIP access control [19]), and/or company policy. The latter question is one we are addressing with the Access Policy Tool (APT) [22]. APT supports specification of best practices and other global access policies in a machine-checkable form, and, based upon firewall rule-sets and other configuration information, discovers the underlying topology of the enterprise system and determines whether the security policy *implementation* conforms to the global security policy *specification*. APT provides a graphical interface that allows its users to investigate the network topology and the firewall rules. Following its analysis, APT provides a list of potential flows through the network that violate the global policy; through the graphical interface, a user can see the flow path, the firewall rules that admit that flow, and the global policy rules that are violated. As a single errant rule can lead to multiple global policy violations, APT analyzes the set of violations and rank-orders the rules involved, typically pin-pointing the problem rules quickly. The APT user is able to investigate the impact of changing firewall rules by using the graphical interface to add, delete, or modify rules and then re-run the analysis. APT does not change the rules in the devices themselves, but does document for its users what those changes are.

Our early work on APT focused on the analysis engine and the graphical interface. We developed sophisticated algorithms and data structures to make APT capable of analyzing large systems quickly. As APT became "useable" in our estimation, we were able to partner with two different companies in the energy sector to have them help us evaluate and improve APT for use in their environments. This experience was invaluable to us in that it provided us with configuration information from real systems on which to test APT's processing power. It also pointed out ways for us to augment APT in order to better meet these proto-customers' anticipated use scenarios. In particular, we extended APT to

- support firewall filtering that requires authentication,
- support object group definitions within firewall configurations,
- automate discovery of network topology from the firewall rules and other configuration information,
- incorporate configuration information that indicates that flows can pass between network "islands,"
- make it possible to run APT functions from the command line in such a way that they can be run using the Unix command "cron,"
- enhance the graphical user interface to automate the layout of a network, and to allow hierarchical graphical encapsulation of subnetworks as graphical nodes,
- enhance number of firewall models supported by APT,
- provide global policy templates for common best practices, and
- improve conflict detection/resolution in global policy specification.

We have also validated APT's analysis and network discovery on the configuration information our partners have shared with us.

The remainder of the paper is organized as follows. Section II describes access control in networks, III discusses related work, and §IV provides an overview of APT and how it functions. V provides more details about the enhancements we've made to APT as a result of our industrial interactions,

and VI identifies future work and presents our conclusions.

II. Access Control

Access control has long been the linchpin of system security; modern systems have multiple access control methodologies, different security models, and separate configurations for each methodology and each device. Together these all form the access control *implementation*. However, only very limited technology exists to answer crucial questions about precisely what security posture is produced, how the different access control policies interact, and whether the implementation is in compliance with an overall statement of global access control policy, among others.

To better appreciate the issue, let's examine some of the access control components of a distributed system. Firewalls are critical assets in the protection of a network. A firewall is configured through its rules (collectively referred to as a *rule-set*), which it can use to mold and shape the traffic that crosses it. A firewall matches the incident network traffic against its rules, using traffic characteristics such as the source, intended destination, and communication protocol, and either forbids or allows the traffic to pass through depending on the action indicated by the matching rule. A typical setting usually contains a distributed firewall implementation, wherein traffic may need to pass through more than one firewall to transit the network. Firewalls can be used to divide the network into secure zones that limit user and application access between zones. For example, Figure 1 illustrates a prototypical control network that is partitioned by firewalls into several subnetworks, including a Corporate network, a Control System network, and some Demilitarized zones (DMZ). The firewalls are used to severely limit the connectivity between devices in the Control System, and everything else.

A system can also have host-based firewalls, implemented in either software or hardware. Software-based firewalls, such as iptables [20] in Linux and several commercial ones available for Windows, are implemented in the host operating system. Hardware-based firewalls, such as 3Com's Embedded Firewall (EFW) PCI card [1], are implemented on the network interface card (NIC) itself, and as a result, those firewalls are tamper-resistant to cyber-attackers who might gain control of the operating system on a host.

There are published guidelines (e.g., the National Infrastructure Security Coordination Center (NISCC) guide to good practices in firewall deployment [8]) to facilitate the development of unique rule-sets for the different firewalls at an operating site, but they are fairly generic and have to be customized by network system administrators to the specific needs of their sites.

However, access control is more complex than just firewall rules. In security-conscious settings, hardened versions of traditional operating systems have been adopted. These include SELinux [17] (developed by the NSA), which is a secure version of the Linux operating system. Similar functionality for Windows operating systems can be provided through third-party software, such as the Cisco Security Agent [9]. Such software can provide mandatory mechanisms such as role-based access control, type checking, and multilevel security models.

One can use a global policy to specify at a system level the overall objectives with respect to access control. It specifies both what connectivity between roles and devices is inadmissible, and what connectivity must be supported. The rules are stated in terms of sets of roles and devices, rather than individual ones. For example, "An account manager in the sales network zone must be able to use sftp to forward any file in the Monthly Sales directory to his Reports directory, on the sales server found in the management network zone." Clearly, multiple access control policies are at play here; it is also clear that the rules can be stated in a form that allows a program analyzing access control configurations to determine whether the rule is satisfied.

The critical problem this work addresses is that networked systems only check fine-grained local access policy rules at single devices, not global policy. Global access policy implementations may or may not comply with global policy requirements. Without a way to check that compliance, serious security vulnerabilities can and do exist in real implementations of critical systems, causing them to fail in potentially harmful ways when attacked.

III. Related Work

The complexity of implementing an access control policy through configuration of a large number of distributed devices and the risk of conflicts among these devices has spawned a significant amount of work. However, the majority of the work focuses on internal consistency among rules of a single firewall, as discussed in [2], [3], [7], [13], [14], [16]. A conflict detection tool for distributed firewall systems has been described in [4], but it only checks for certain kinds of syntactic errors, such as overlapping rules, and not for semantic errors with respect to a specification of the intent. Furthermore, none of the work takes into account the collective fine-grained access control provided by the network-based mechanisms (e.g., firewalls) and host-based mechanisms (e.g., SELinux policies). Also, none of the above work has been applied to and optimized for specific classes of networks.

Another vector of research effort focuses on automatic generation of consistent policy implementation (at devices) from formal description of global policy [5], [15]. Cisco provides CiscoWorks [10], a suite of products that enable top-down firewall policy management that includes configuration management, a policy manager that generates policies from high-level specifications, and log/audit analysis. Those are appropriate for new, vendor-homogeneous systems, but do not address existing implementations or heterogeneous systems. Furthermore, they require specification of detailed and exhaustive global security policy. For most current network system administrators, the ability to check existing configurations against policy specifications that indicate intended high-level behavior would likely be more useful.

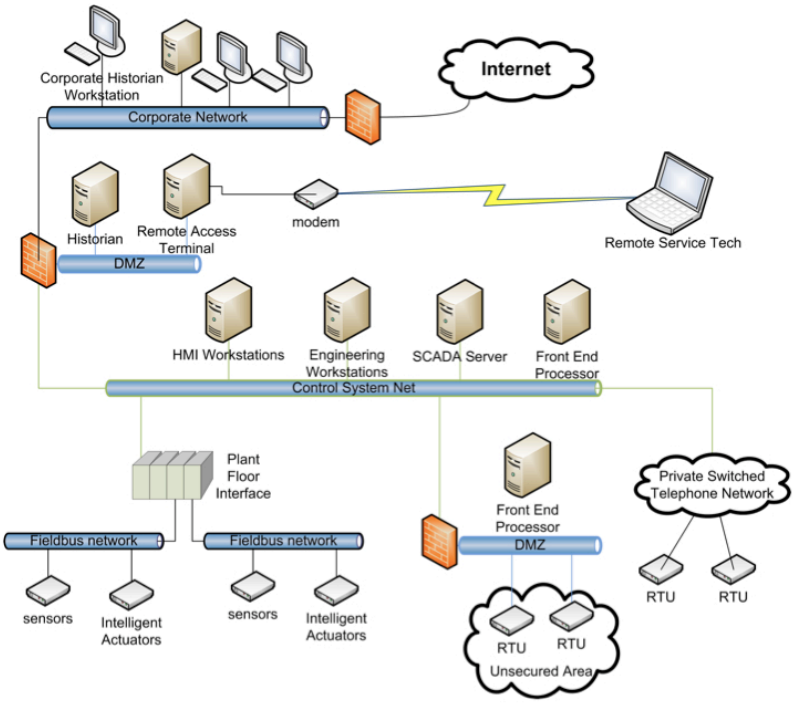

Fig. 1. Representative Network

Vendor-neutral tools have recently appeared that consider access control in a distributed system, including the Skybox firewall compliance auditor [29] and Red Seal Network Advisor [26]. Both tools use network topology and routing information to determine potential network flows. Red Seal SRM uses vulnerability databases and specification of software running on hosts to compute risk measures, while Skybox determines when firewall rules allow violation of a more abstractly stated global policy. In comparison, our approach incorporates sophisticated statistical analysis for highly improved scalability and also naturally admits integration of higher-level layers of access control policy (e.g., SELinux policies, and/or role-based access in trusted networks).

One approach to checking implementation against specification is known as "model checking" [11]. This line of research has a rich history in the context of proving the correctness of both hardware and software. The key abstraction is a finite-state machine, and the key problem is that the size of the state space explodes with the complexity of the system. As we consider integration of higher layers of access control mechanisms, the finite state machine view may be appropriate for them. However, while it is possible to map *network* access control into a classical model-checking framework, we aren't convinced of the value of doing so. Much of the attention in model-checking is paid to compact representation of the state space; we already have a representation of the problem (i.e., the rule-graph, to be described), which is a graph whose number of nodes is linear in the number of rules. Furthermore, our analysis takes advantage of the problem domain rather than a general state-space, with an approach that lends itself naturally to optimizations for scalability. The research on "attack graphs," largely an application of model checking, is of some relevance when considering the scalability issue, particularly when compared to the statistical analysis that we have developed. Ritchey and Ammann [27] use model checking to identify a *single* violating path in an attack graph. Sheyner et al. [28] provide a more comprehensive analysis, but suffer from the state-space explosion problem, since the entire attack graph needs to be analyzed to provide the relevant metrics, thereby severely limiting the scalability. Ou et al. [24], [25] have used a prolog-based approach, rather than the traditional model checking, for attack graph generation and analysis; however, their solution does not seem to be able to scale to large networks that are also deep (i.e., graphs with potentially long paths), or to calculate generic metrics that are functions of paths rather than edges in the graph.

IV. APT Overview

We developed the Access Policy Tool (APT) [22] as a result of experience in a project for which the system had a large number of firewalls, but no way for us to determine just exactly what access was permitted. APT development occurred in the context of an I3P [12] project providing support for security in industrial control systems, and as part of that effort, we eventually partnered with two large companies in the energy sector to validate APT.

APT takes the set of rule-sets and other configuration information from a system's set of firewalls, determines the network topology being referenced by that configuration, and

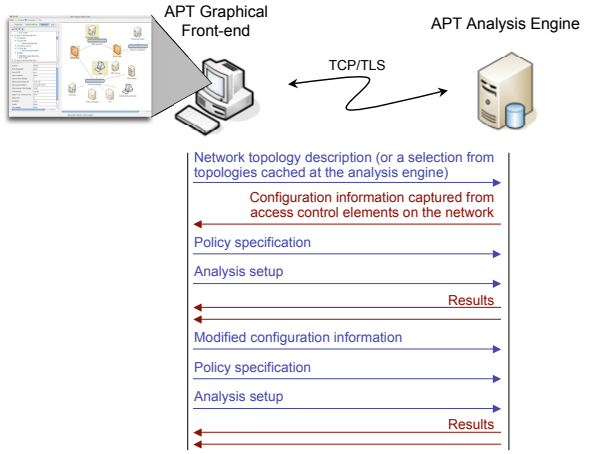

Fig. 2. APT Architecture

then determines whether the access policy implementation expressed in the rules exactly expresses the constraints expressed at a much higher view by the global access policy.

As shown in Figure 2, APT has two independent components:

- a graphical front-end (or management console), written in Java Swing, that system administrators can run from their workstations and use to provide information about the basic network topology, to enter specifications of the global access policy (or subsets thereof), and to set up analyses; and
- an analysis engine, written in C++, which captures the system state and analyzes that information with respect to policy specifications.

As indicated in Figure 2, the two components of the tool communicate securely over the network using TCP/TLS protocols. This segregation of functionality makes it possible to use the analysis engine as an appliance that can be plugged into a suitable spot in the network, from which it can establish secure connections to the access control elements present on the network and capture the relevant configuration information. The actual interface between the tool and the user (the front-end) can then be freely placed where it is convenient for the user.

As shown in the figure, the network topology is shown at the graphical front-end. The topology can be drawn explicitly with the easy-to-use drawing tools included with the front-end, or via text files that can be read by the front-end. The text files may be the result of automatically discovering the topology from the firewall rule-set. The network includes all the relevant access control elements (firewalls, proxy servers, wireless access points); details for each element, such as its IP address(es) and parameters that the analysis engine can use to establish secure connections to the element to capture its configuration; and all the network interactions between the listed elements. Entities can be grouped together to indicate subnetworks and LANs, and properties can be specified for such groupings. The front-end converts the visual topology representation provided by the user to an internal XML representation. We use XML as our underlying language for internal representation for most of the information in the tool, which provides the added benefit that we may interface third-party tools and applications simply by providing wrappers that output the XML conforming to our schemas. It is also possible to browse and select from the list of topology descriptions previously cached remotely at the analysis engine. In either case, once the topology is decided, the information is sent over the network to the analysis engine. The analysis engine uses that information to securely obtain the snapshot of the access control policy implementation, in the form of configuration files from the various devices indicated in the topology description. APT can itself fetch the configuration files from those devices, provided that the network connectivity for this exists, and APT is given a script that logs into the device and downloads the configurations. Our users more typically use APT's ability to read the configurations from files, rather than from devices. APT also indexes and caches the topology description and captured configuration information (both of which are encrypted before storage for added security). Automated checks verify that the topology (either entered through the user interface, or determined automatically) and the firewall configurations are logically consistent with each other. The captured information from all the different sources is then converted to XML (one unified schema for firewall rules, and another for OS-based mechanisms), and the resultant XML is sent back to the front-end, where the user can then highlight various elements in the topology diagram and look at their current configurations (e.g., rule-sets for a firewall).

The user then specifies the global access constraints, which are a subset of the possible comprehensive access policy. They indicate some intended behavior against which the user wishes to check the policy implementation for compliance. The user uses a graphical front-end to specify constraints for the various elements or groupings of elements, and the tool considers their conjunction. The individual constraints can express positive as well as negative assertions about the nature of traffic, roles, user classes, or applications that can access a particular set of resources (hosts, specific files or applications, and so forth). In other words, we can express things that should never happen as well as things that must be allowed. Once specified, the constraints are converted into an internal XML representation. This further allows the freedom to use third-party technologies, such as a variety of modal logics, to specify the global access constraints.

The next step for the user is to set up the analysis. The user can select either an exhaustive analysis or a statistical analysis, following which the front-end sends the policy specification and setup information for the analysis to the analysis engine. The analysis engine sends back the results as it obtains them, giving the user the option to abort the analysis at any stage and do post-analysis on the partial results. The user can manipulate, filter, or navigate through the results, if any violations are found, to visualize the problem and diagnose the key misconfigurations behind the violations. A variety of post-analysis techniques are available in the front-end to help the user identify the likely root causes of the violations. Once the user has some possibilities in mind, hypothetical changes can

be analyzed using the front-end; the user can make proposed changes to the configuration of various elements (e.g., modify, delete, or reorder certain rules), and the new information will be sent to the analysis engine, which then performs a quick re-analysis to see if the (hypothetically) modified configuration information now conforms to the specification of the access policy. This feature can also be used to check planned changes in configuration for compliance quickly before they are actually rolled out.

V. Interactions with Industry

Our partnerships with industry have their basis in other projects. Both of our APT partners had representatives serving on industrial advisory boards of other projects we're involved in. Through those representatives, we were able to make contact with people in their organizations who their representatives knew would be interested in APT. Our collaboration evolved through teleconferences (approximately quarterly) and visits. The structure was informal; we tried to be very respectful of their time, which had the effect of making our infrequent meetings quite focused on the most important needs of the project at those times.

Interactions with our industrial partners were invaluable in illustrating to us some characteristics of real firewall systems in use, real characteristics of industrial control networks, and real expectations of potential users of APT. We didn't know what to expect of real topologies, and were surprised when our first partner gave us configuration information for a system with over 70 firewalls and over 1000 devices. Our second partner's system is much smaller in terms of firewalls—fewer than 10—but reflects a sophisticated architecture involving multiple control networks, each with a hot backup system in case of failure. We now describe areas where these interactions led us to make enhancements to APT.

a) **Authentication, Authorization, and Accounting (AAA).:** The first set of firewall rules provided to us contained access control lists (ACL) that applied to authentication and authorization. Of course these options exist in the CISCO PIX firewall rule specifications, but in our early development of APT we saw AAA as an area that we would address on an as-needed basis. The rules in those ACLs described the traffic that needs to be authenticated before normal filtering rules are applied. We also found commands setting up the authentication mechanism (such as RADIUS, in which case the location of the RADIUS server was specified) and the incoming interface to the firewall to which the AAA-related ACL applied, among others. We consequently modified APT to support

- encoding of the AAA-related information into its universal XML schema for firewall rule-sets,
- indication of global policy constraints that apply only to authenticated traffic, and
- consideration of the above two aspects (rule-sets characterizing authenticated traffic and constraints applicable to such traffic) during the analysis.

Later experience showed us that these features are apparently very common in industrial control environments and in corporate networks that support mid-to-high levels of security.

b) **Object Groups:** Firewall configurations may support user definition of groups of objects, e.g., groups of networks, services, protocols, or users. The group definitions may then appear in a firewall rule, e.g., a network group might be identified as the set of destinations to which the rule applies. Our industrial partners use these extensively, as they are invaluable aids in describing large networks. We modified APT to parse object group definitions (even recursive ones) and to provide a graphical interface to those definitions. We also enhanced APT to allow a user to define APT object groups to be used in the specification of global policy rules. For example, in Figure 1, one object group can be used to encapsulate all the devices in the Corporate network, and another to encapsulate all the devices in the Control network; global policy rules that govern connections between the two networks can use the object group definitions in their specifications.

c) **Topology Discovery.:** In our initial design of APT we assumed that the user could specify his system's network topology. However, we discovered that our partners did not keep explicit topology information in a form that makes it convenient to communicate topological information to APT. Furthermore, we were anxious to minimize our demands on partners' engineers' time in support of our efforts. We learned of the ANTFARM tool [6], communicated with its developers, and obtained it shortly thereafter when it was released as open-source software. ANTFARM uses a relational database to coordinate disparate pieces of connectivity information. When a new connection is discovered, it is entered into the database, and an ANTFARM script is executed to forge any new connectivity relationships that are a consequence of the prior state of connectivity knowledge and the new knowledge. Working with ANTFARM and our partners, we discovered that our partners needed to annotate some configuration information to help differentiate between networks, and that ANTFARM needed to parse that annotation. We have since made extensive customizations to the ANTFARM logic, again tailoring it to the technical requirements of topology discovery for our partners' networks. Issues related to topology discovery, verification, and validation have consumed a great deal of our attention this last year. However, it has been worthwhile. In one notable session with a partner's network engineer, we found that APT had discovered a path through the network that the engineer did not believe should be present. With APT we were able to point out to him the configuration that allowed that path, and he agreed that the path must be present (and presumably made the configuration change to close the hole).

d) **Network Islands:** One of our partners' configurations contained "switch" statements. These indicate that traffic leaving through that firewall is routed to another network, which is not necessarily adjacent to the firewall. Global policy statements may involve such flows, so we augmented APT's topology discovery mechanisms to include the connectivity information specified by switch statements.

e) **Command Line Processing. :** One uses APT much less interactively than we envisioned. They keep track of changes made to firewall configurations using a tool called "rancid" (Really Awesome New Cisco config diff) [21]. This tool maintains a CVS repository of the changes (diffs) made to the native firewall configurations and can be used to obtain the latest copy of the rule-sets for any firewall in the system. APT uses the set of such rule-set files (in their native format, which uses the PIX/ASA command-set, all collected in a directory) as input for inferring topology and conversion to XML conforming to APT's universal rule-set schema. Their vision for APT is as an off-line checker that would be run automatically every night against the current rule-set. Correspondingly, we augmented APT so that it could be run entirely from a command line with explicitly named input and output files, and thus be run automatically through a "cron" script.

f) **Enhanced Graphical Interface:** APT's graphical display is written using Java libraries. Our initial display was written using the JGraph library [18], but we found that the size of the networks we were trying to display required use of a different tool. The APT graphical engine was re-factored to use the Java Universal Network/Graph Framework (JUNG) [23] and the Model-View-Controller architectural pattern. The engine was augmented with features such as node clustering for easier viewing of large topologies, and zooming. It was also extended to provide multiple automatic layout schemes (easily accessible from a drop-down menu) for quickly drawing uncluttered representations of topology files. The users can then further arrange various elements to their satisfaction and save the topology back to the disk, such that the tool remembers all the changes made to the drawing. To further reduce clutter, the user can collapse various networks or collections of networks into a single element in the topology diagram, which can then be dragged around. Expanding such a collapsed node causes the constituent nodes to show up again, preserving their relative locations before the collapse.

g) **Extended Range of Firewalls Supported:** APT transforms firewall rules from specific firewall vendors and models of firewalls into a unified form, expressed in XML. The analysis engine works on this unified representation. Whenever a new firewall syntax is to be integrated into APT, a "native-to-unified" translator needs to be augmented to support inclusion of the new device. Our partners had firewall models for which this extension was performed.

h) **Global Policy Templates:** Our partners desire assistance in the development of global access policies. To facilitate that we have built into APT a library of global policy templates that express constraints based on best practices recommendations such as those in NIST Special Report SP-800-82 [30]. The templates require a user to define object groups that identify sets of hosts and/or networks, and substitute those group identifiers into the templates.

i) **Global Policy Conflict Detection and Resolution.:** A global policy is a collection of statements on flows that should be allowed and/or should not be allowed. The expression formalism is broad enough that conflicting rules can be concurrently specified. This is not necessarily a bad thing; certain forms of conflict can be interpreted so as to give a clear and intuitive preference of one rule over another. Other conflicts cannot be so resolved, and indicate a need for the user to clarify the intent. We extended APT to identify conflicts in global policies, and to point out the rules that could be resolved and those that need a user's attention.

j) **Change Management and Auditing:** Change management and auditing is a use of APT that one of our partners is expressly interested in. We had foreseen this use, and at the time of our partnering, APT supported these functions.

VI. Conclusions

When we conceived of the Access Policy Tool, we were captivated by the research issues involved in automating the checking of firewall implementations against global access policies on very large systems, very efficiently. We identified and solved a number of crucial problems, and then turned to the challenge of making our work useable in industrial contexts. We were very fortunate to find two willing partners who (after we signed non-disclosure agreements) generously shared their time and information about their networks, to help us see how APT could best be used in their contexts.

We found that real-world systems have complexities that are convenient to overlook when pursuing purely academic research. The challenges of dealing with those complexities are not fundamental in a scientific way, but are certainly essential if the foundational work that academics naturally gravitate towards is to see practical use.

The enhancements made to APT as a result of our interactions with industry have shaped it into a tool that more useable by that industry. We are looking forward to seeing APT being used more widely soon.

Acknowledgments

This material is based in part upon work supported by the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001, under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security, the I3P, or Dartmouth College.

This material is also based in part upon work supported by the National Science Foundation under Grant No. CNS-0524695. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

We also thank Bryan Richardson and Michael McDonald for the graphic in Figure 1, and Bryan for all his help with ANTFARM.

REFERENCES

[1] 3Com Corporation. 3Com Embedded Firewall Solution. http://www.3com.com/other/pdfs/products/en_US/400741.pdf, Apr 2008.

[2] E. Al-Shaer and H. Hamed. Firewall Policy Advisor for Anomaly Detection and Rule Editing. In *Proc of IEEE/IFIP 8th Intl Symp on Integrated Network Mgmt (IM 2003)*, pages 17–30, Colorado Springs, CO, Mar 2003.

[3] E. Al-Shaer and H. Hamed. Management and Translation of Filtering Security Policies. In *Proc of the 38th IEEE Intl Conf on Communications (ICC 2003)*, pages 256–260, Anchorage, AK, May 2003.

[4] E. Al-Shaer and H. Hamed. Discovery of Policy Anomalies in Distributed Firewalls. In *Proc of IEEE INFOCOM (Vol. 4)*, pages 2605–2616, Hong Kong, Mar 2004.

[5] Y. Bartal, A. Mayer, K. Nissim, and A. Wool. Firmato: A Novel Firewall Management Toolkit. *ACM Trans on Computer Sys*, 22(4):381–420, 2004.

[6] National SCADA Test Bed. Advanced Network Toolkit for Assessments and Remote Mapping (antfarm). http://www.oe.energy.gov/DocumentsandMedia/12-ANTFARM.pdf, August 2008.

[7] F. Boboescu and G. Varghese. Fast and Scalable Conflict Detection for Packet Classifiers. *Computer Networks*, 42(6):717–735, 2003.

[8] Centre for the Protection of National Infrastructure. NISCC Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks. http://www.cpni.gov.uk/Docs/re-20050223-00157.pdf, Apr 2008.

[9] Cisco Systems. Cisco Security Agent. http://www.cisco.com/en/US/products/sw/secursw/ps5057/index.html, Apr 2008.

[10] Cisco Systems. CiscoWorks Management Center for Firewalls. http://www.cisco.com/en/US/products/sw/cscowork/ps3992/index.html, Apr 2008.

[11] E. Clark, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, Cambridge, MA, 2000.

[12] Dartmouth. The Institute for Information Infrastructure Protection. http://www.thei3p.org.

[13] D. Eppstein and S. Muthukrishnan. Internet Packet Filter Management and Rectangle Geometry. In *Proc of ACM-SIAM Symp on Discrete Algorithms (SODA 2001)*, pages 827–835. Society for Industrial and Applied Mathematics, 2001.

[14] M. G. Gouda and A. X. Liu. Firewall Design: Consistency, Completeness, and Compactness. In *Proc of DSN 2004*, pages 320–327, Tokyo, Japan, Mar 2004.

[15] J. D. Guttman. Filtering Postures: Local Enforcement for Global Policies. In *Proc of IEEE Symp on Security and Privacy*, pages 120–129, Oakland, CA, May 1997.

[16] A. Hari, S. Suri, and G. M. Parulkar. Detecting and Resolving Packet Filter Conflicts. In *Proc of IEEE INFOCOM (Vol. 3)*, pages 1203–1212, Tel Aviv, Israel, Mar 2000.

[17] Bill McCarty. *SELinux:NSA's Open Source Security Enhanced Linux*. O'Reilly, Sebastopol, CA, 2004.

[18] mxGraph. Jgraph - the Java Open Source Graph Drawing Component. http://www.jgraph.com/jgraph.html, 2001.

[19] NERC. Critical infrastructure protection standards. i\\http://www.nerc.com/page.php?cid=2%7C20.

[20] Netfilter.org. The Netfilter.org "iptables" Project. http://www.netfilter.org/projects/iptables/index.html, Apr 2008.

[21] Shrubbery Networks. RANCID-Really Awesome New Cisco config Differ. http://www.shrubbery.net/rancid/, 2006.

[22] D. M. Nicol, W. H. Sanders, S. Singh, and M. Seri. Usable Global Network Access Policy for Process Control Systems. *IEEE Security & Privacy*, 6(6):30–36, Nov.-Dec. 2008.

[23] Joshua O'Madadhain, Danyel Fisher, and Tom Nelson. Java Universal Network/Graph Framework. http://jung.sourceforge.net/, 2003.

[24] X. Ou, W. F. Boyer, and M. A. McQueen. A Scalable Approach to Attack Graph Generation. In *Proc of 13th ACM Conf on Computer and Communications Security (CCS 2006)*, pages 336–345, Alexandria, VA, 2006.

[25] X. Ou, S. Govindavajhala, and A. W. Appel. MulVAL: A Logic-Based Network Security Analyzer. In *Proc of 14th USENIX Security Symp*, pages 113–128, Baltimore, MD, 2005.

[26] RedSeal Systems. RedSeal Network Advisor . http://www.redseal.net, Apr 2008.

[27] R. W. Ritchey and P. Ammann. Using Model Checking to Analyze Network Vulnerabilities. In *Proc of IEEE Symp on Security and Privacy*, pages 156–165, Oakland, CA, 2000.

[28] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J. M. Wing. Automated Generation and Analysis of Attack Graphs. In *Proc of IEEE Symp on Security and Privacy*, pages 273–284, Oakland, CA, 2002.

[29] Skybox Security. Automated Firewall Management Software. http://www.skyboxsecurity.com/, Apr 2008.

[30] 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.

[31] A. Wool. A Quantitative Study of Firewall Configuration Errors. *Computer*, 37(6):62–67, Jun 2004.