# Case Studies of SCADA Firewall Configurations and the Implications for Best Practices

Dinesha Ranathunga, *Student Member, IEEE*, Matthew Roughan, *Senior Member, IEEE*, Hung Nguyen, *Member, IEEE*, Phil Kernick, and Nickolas Falkner

***Abstract*—Firewall configuration is an important activity for any modern day business. It is particularly a critical task for the supervisory control and data acquisition (SCADA) networks that control power stations, water distribution, factory automation, etc. Lack of automation tools to assist with this critical task has resulted in unoptimised, error prone configurations that expose these networks to cyber attacks. Automation can make designing firewall configurations more reliable and their deployment increasingly cost-effective. Best practices have been proposed by the industry for developing high-level security policy (*e.g.,* ANSI/ISA 62443-1-1). But these best practices lack specification in several key aspects needed to allow a firewall to be automatically configured. For instance, the standards are vague on how firewall management policies should be captured at a high-level using its specifications. In this paper, we uncover these missing pieces and propose extensions. We apply our extended best-practice specification to real-world firewall case studies to achieve multiple objectives: 1) to evaluate the usefulness of the refined best-practice in the automated specification of firewalls and 2) to illustrate that even in simple cases, SCADA networks are often insecure due to their misconfigured firewalls.**

***Index Terms*—SCADA network security, Zone-Conduit model, firewall autoconfiguration, security policy, SCADA best practices.**

## I. Introduction

SUPERVISORY Control And Data Acquisition (SCADA) networks control the distributed assets of many industrial systems. Power generation, water distribution and factory automation are just a few examples that illustrate the critical nature of these networks.

SCADA networks are not like corporate IT networks [34]. IT networks can accept a degree of reliability orders of magnitude lower than the network controlling a power station. A fault in the latter will cost serious money, if not lives.

SCADA devices are built for *reliability*, but often lack built-in security features to guard them from cyber attacks. Consequently, these devices depend on *firewalls* for protection [34]. Hence, firewalls are integral to the safe and reliable operation of SCADA networks.

Manuscript received January 28, 2016; revised May 7, 2016 and July 22, 2016; accepted July 23, 2016. Date of publication August 2, 2016; date of current version December 8, 2016. This project was supported by an Australian Postgraduate Award, Australian Research Council Linkage Grant LP100200493, and CQR Consulting. The associate editor coordinating the review of this paper and approving it for publication was M. Sloman.

D. Ranathunga, M. Roughan, H. Nguyen, and N. Falkner are with the University of Adelaide, Adelaide, SA 5005, Australia (e-mail: dinesha.ranathunga@adelaide.edu.au).

P. Kernick is with CQR Consulting, Unley 5061, Australia.

Digital Object Identifier 10.1109/TNSM.2016.2597245

Firewall configuration, in practice, is a complicated and repetitive manual task. It involves training in proprietary and device specific configuration languages and long and complex device configurations. Lack of automation tools to assist with such complexity has resulted in *unoptimised*, *error-prone* configurations that often deviate from the industry recommended network security guidelines [36], [37].

A cost-effective solution is to build network operations tools that automatically derive firewall configurations from high-level policy, *i.e.,* autoconfiguration. Bellovin and Bush [7] investigated automating security configuration. They identified requirements such as security, robustness and a database-driven approach as key, but left out high-level policies.

A high-level security policy description is the starting point for automation. Such a description would allow firewall policies to be specified by management-level policy makers.

The American National Standards Institute (ANSI)/ International Society for Automation (ISA) best practices introduce useful security concepts to mitigate threats in control systems [5]. They describe on abstract policy specification for SCADA networks. We analyse real SCADA firewall configurations using these concepts to identify the missing pieces in the standard required for automation and propose solutions.

This paper extends our previous work [27], and incorporates findings from *three* additional real-world SCADA case studies. The additions help re-enforce how the ISA security best practices, as they currently exist, are unable to cater for policy specification requirements found in practice. The solutions we propose increase the *precision* and *usefulness* of the best practice for the automated specification of firewalls. Our contributions are:

1) A series of case studies of real SCADA firewall installations. To the best of our knowledge, ours is the only existing case study that presents misconfigurations in real SCADA firewalls, in detail. These misconfigurations make the SCADA plants insecure and vulnerable to cyber attack.
2) We describe requirements for firewall autoconfiguration. The ANSI/ISA standard is too flexible to be able to cater for automation so, we refine the standard to make it precise and suitable for firewall autoconfiguration.
3) We extend our previous work in [27] by including VLAN processing in our Parser.

The new real-world case studies we present, further illustrate that Supervisory Control and Data Acquisition (SCADA) firewalls, even in simple cases are often misconfigured (see Sections IV-B and V), highlighting the need for better solutions. Particularly, we show how incorrect use of firewall security features such as Cisco *security levels* (described in detail in Section III) and *NAT-control* lead to a broad range of services to be enabled un-intentionally. Our case studies also uncover the use of outdated firewall software in production SCADA environments! Such software contains publicly-listed security vulnerabilities that make these firewalls (and the SCADA plants) *easy* targets for cyber attackers.

## II. Background and Related Work

Lack of internal network segmentation is a significant contributor to the rapid spread of security threats and attacks in SCADA networks [5], [9], [34]. The ANSI/ISA standards introduce the concepts of *zones* and *conduits* as a way of segmenting and isolating the various sub-systems in a control system [5]. The *Zone-Conduit model* is a very useful starting point for a high-level description of security policy, and so we describe it in detail here.

A *zone* is a logical or physical grouping of an organisation's systems with similar security requirements based on criticality and consequence [5]. By grouping systems in this manner, a *single security policy* can be defined for all members of a zone. For example, three security zones can be defined to accommodate low, medium and high-risk systems, with each device assigned to its respective zone based on their level of protection needed. A low-risk system can be accommodated within a high security zone without compromising security, but not vice versa.

A *single zone-policy* implies that selected subsystems in a zone (*e.g.,* a server) should not have their own separate policies (*i.e.,* no exceptions). Allowing *exceptions* would impart a false sense of security to those systems. These systems are only secure as the zone itself, in the absence of any firewalls enforcing a real separation [5].

A *conduit* provides the secure communication path between zones, enforcing the policy between them [5]. Security mitigation mechanisms (*e.g.,* firewalls) are implemented within a conduit. A conduit could consist of multiple links and firewalls, but logically is a single connector. For a specified policy (*i.e., what* is enforced), a conduit abstracts away the intricacies of *how* the policy is enforced.

Zone and conduit concepts are intended as a platform for high-level security policy description. Before using these concepts in policy specification for firewalls, it is best to evaluate their usefulness. Particularly how well they cater for security architectures used in practice in real networks (*e.g.,* back-to-back firewalls).

### A. Related Work

Previous works have conducted *bottom-up* analysis of firewall rulesets (*i.e.,* ACLs) to assist with *debugging* and *troubleshooting* [24], [35], [38]. Lumeta [35] and Fang [24] are such firewall analysis tools that allow users to run queries against ACL rules to check firewall configuration behaviour. Algosec Firewall Analyzer is a commercial closed-source tool that is based on Lumeta and Fang engines. It allows administration of a large number of firewalls [3]. Similarly, FireMon [16] and Skybox Firewall Assurance [32] are two commercial firewall maintenance tools that have audit capabilities. Skybox additionally automates firewall-rule lifecycle management, which helps keep firewall rulesets clean and optimised. Some non-commercial tools [1], [17] have also focused on maintaining concise firewall configurations by detecting and removing ACL implementation errors (*i.e.,* redundancies and conflicts).

Chen *et al.* [12] have proposed a method to automatically detect firewall misconfigurations. The solution first translates firewall policies into desired filtering actions for a set of test packets. These test packets are then used to detect an diagnose misconfigurations.

ACL analysis has also been used to determine policy change impact [22], [25]. To do so accurately, firewall topology needs to be accounted for. In some works [22], topology has to be incorporated manually; an impractical consideration when comparing policies of complex networks. Other (bottom-up) approaches overcome the problem by automatically incorporating topology to construct a network-wide ACL tree [1], [38].

Another approach [23] detects policy anomalies by grouping hosts whose packets are treated identically by the firewall, into equivalence classes. Doing so, suggests hosts in distinct equivalence classes have different policies, when in reality, they may not. For instance, hosts in the same zone, in actuality, have the same policy in the absence of firewalls separating them. We address this shortfall by identifying the disjoint zones in the network (*i.e.,* zones separated by firewalls). All hosts in a zone, then have the same security policy.

A tool has also been developed to help with policy design and automatically check correct implementation [8]. The access-control policies supported by the tool are built on the Role-Based Access Control (RBAC) abstraction. The implementation encodes details of the physical system and traffic control mechanisms. The tool concentrates on checking consistency between the policies and implementations and does not deal with autoconfiguration.

Rysavy *et al.* [29] have likewise proposed the verification of ACLs against a security policy using Satisfiability Modulo Theory (SMT). The policies are specified using the Security Policy Specification Language (SPSL) which operate at a low IP flow-level. The checking is performed on all possible flows, identified by source and destination IP addresses and port numbers. In contrast, our aim is high-level policy specification.

Salah *et al.* [30] has employed a queuing model to analyse the performance of rule-based firewalls including throughput, packet loss, delay and CPU utilisation. Our work mainly deals with firewall configuration not performance.

The bottom-up approaches discussed above, have a common drawback in that ACLs contain network and vendor intricacies such as IP addresses. So, ACL content can change even when the policy intent remains unchanged. Regardless of how compact and accurate the ACLs are, they are unsuitable for describing policies in automation. Our main interest in automation is to describe policy intent not their implementations.

Autoconfiguration requires firewall policy to be described *top-down*, flexibly enough and in detail. To do so, a single network-wide policy must be maintained, allowing security administrators to easily determine who gets in and who doesn't [21]. These network-wide policies need to be free of network-centric minutiae, so changes to policy intent can be clearly distinguished from changes to the network. Network-wide firewall policy changes are also more useful to administrators than per-firewall changes when analysing change impact, because the latter can be made redundant by other firewalls policy through rule interactions.

Current top-down solutions allow creation of network-wide policies [4], [6], [15], [20], but these policies still contain network intricacies. For one, Firmato [6] employs the Role-Based Access Control (RBAC) paradigm in its network grouping language. But, the abstraction requires minutiae such as IP addresses to be provided in-policy, when implementing policy on a network instance. For another, Cisco has also introduced security policy management products (*e.g.,* VNMC for VSG policy management) to cater for complexities introduced in network virtualisation [15]. For scalability, the products allow operating systems (*i.e.,* VMs) to be allocated to zones and policies to be defined per zone. But, each VM still needs to be defined using low-level detail such as hostnames.

We build our (top-down) firewall policy description on the Zone-Conduit abstraction. The model decouples network topology and security policy, so, the reduced policy complexity allows these policies to be easily understood by humans. To the best of our knowledge, our work is the first to attempt to identify such a description suitable for autoconfiguring firewalls. We employ a bottom-up approach that parses firewall configurations to identify the missing pieces of the Zone-Conduit model. Doing so, allows to refine the model for use in the high-level specification of SCADA firewalls.

The related works described above commonly target non-SCADA domains. But, the observations made through parsing firewall configurations (*e.g.,* misconfigurations) in these domains also occur in SCADA environments. For instance, Wool [36], [37] analysed 74 real Corporate firewall configurations and found 4 out of 5 firewalls were badly configured. He found over 70% of the configurations allowed access to the firewall over insecure (*i.e.,* unencrypted and poorly authenticated) protocols (*e.g.,* Telnet). Over 60% of the configurations included rules that were inherently too broad and admitted far too many services than necessary (*e.g.,* a rule that permits all-TCP traffic between two hosts). In Sections IV and V we see that such misconfigurations also occur in real SCADA networks which is surprising given the critical nature of these networks!

## III. Methodology

Firewall configurations are long and complex. For example, one configuration we discuss has 432 lines. Existing tools such as Fang or Lumeta do not support Zone-Conduit based

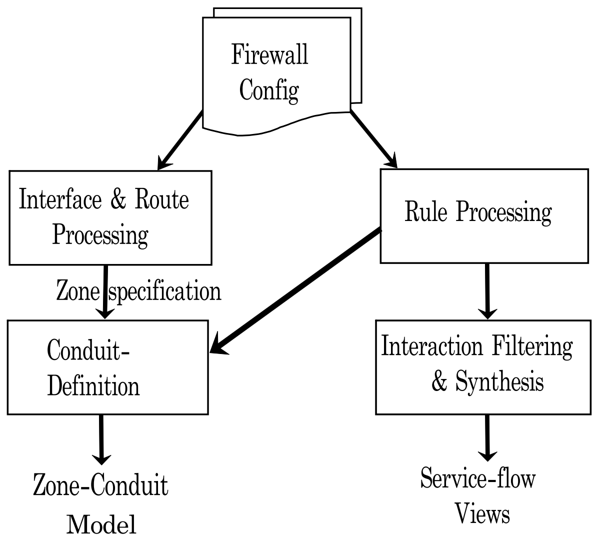


Fig. 1. Firewall configuration parsing process.

high-level policies. So we built an automated Parser to parse our configurations using Zone-Conduit concepts.

We describe the Parser in detail here because it explains the use of Zone-Conduit concepts in the analysis of practical networks. The Parser is depicted in Figure 1. The details are described below:

*Firewall Config:* The input firewall configuration text-file containing interface configurations, static routes and Access Control Lists (ACLs). Multiple files can be input in a network with more than one firewall.

*Interface and Route Processing*: The processing of firewall interface configurations and static routes. This extracts interface names, subnet IP addresses, security levels, additional network and gateway IP addresses.

*Rule Processing:* The processing of ACLs assigned to firewall interfaces and any implicit rules. Implicit rules enable services through the firewall over and above ACLs. More details are discussed in Section III-B.

*Conduit Definition:* The definition of conduits that inter-connect the security zones in the SCADA network. Details are covered in Section III-C.

*Zone-Conduit Model:* The zone and conduit topology output of the SCADA network.

*Interaction Filtering & Synthesis:* The filtering of ACL rule interactions and synthesis with implicit rules. Details of this stage are covered in Section III-F.

*Service-Flow Views:* The output traffic-flow views for the firewall. A service-flow view describes the zones that a given service can access.

Our current Parser can use the following firewall configurations as input: Cisco Adaptive Security Appliance (ASA), Cisco Private Internet eXchange (PIX), Cisco Internetwork Operating System (IOS) or Cisco Firewall Services Module (FWSM). It begins by processing the individual firewall interface configurations. It then processes any static route configurations to identify the location of additional networks and gateways. Rule processing partly involves parsing the

ACLs assigned to firewall interfaces. These indicate the traffic permitted to traverse each of the firewall interfaces. Rule processing also involves parsing *implicitly* enabled services. The Parser creates the ANSI/ISA Zone-Conduit model of the network and enabled service-flow views as output.

## A. Zone Construction

The Parser constructs zones by analysing the interfaces and subnets defined in the firewall configurations. It assumes initially that each firewall interface connects to a disjoint zone, and looks for indications that these potential zones should merge. If the assumed disjoint zones actually form a single zone, the traffic flow from these zones (*e.g.,* to a third zone) is equally controlled by the ACLs on the firewall interfaces connected to these zones. In the absence of any such evidence in the ACLs, we can deduce our original assumption holds and the zones must be disjoint. The original *Zone-Firewall model* is updated to reflect any merged zones identified.

The Parser also processes static routes; these contain IP address details of next-hop gateways and networks reachable via them. By identifying and including these additional networks and gateways, the Zone-Firewall model can be further extended.

## B. Implicit Rules

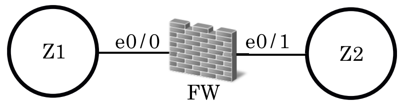
In a Cisco firewall, traffic flows can be enabled explicitly through ACLs or implicitly via several alternate methods. One available method in ASA, PIX and FWSM firewalls is to assign *security levels* to the firewall interfaces [14]. An interface security level is defined as a level of trust bestowed on the network connected to that firewall interface. In the absence of an ACL assigned to such an interface, certain traffic flows are permitted by default from an interface with a high security level to one with a lower security level [14].

Special configuration commands with higher precedence than interface-assigned ACLs can also enable services implicitly in Cisco firewalls. For example, these commands can enable SSH or HTTP firewall management traffic into the firewall interfaces [14]. Such commands are absent in Check Point firewalls [11]. We aim to develop a policy description that is firewall-vendor independent. However, due to Cisco's dominance [33] as a security appliance vendor in SCADA and corporate environments, the networks we had access to used Cisco firewalls. In Section III-D, we discuss in detail how zones are used to accomodate this firewall management traffic.

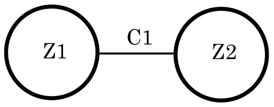
Implicit rules provide quick and easy alternatives to ACLs in enabling services through the firewall. They may not map to clear policies but are convenient. However, autoconfiguration relies on clear security policies to permit traffic through a firewall. Implicit rules may aim to provide this, but we will see that they actually confuse the situation.

## C. Zone-Conduit Model

As Section II discussed, a Zone-Conduit model describes the logical grouping of systems in a network. It provides a high-level view of the network-segregation strategy of an organisation.

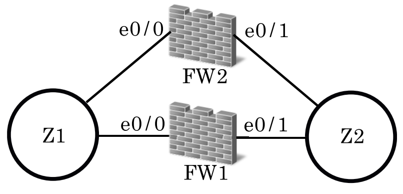


(a) Zone-Firewall model for 2 zones separated by a firewall.

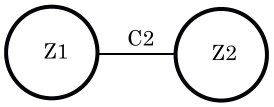


(b) Zone-Conduit model.

Fig. 2. Single-firewall Conduit definition.



(a) Zone-Firewall model for 2 zones separated by parallel firewalls.



(b) Zone-Conduit model.

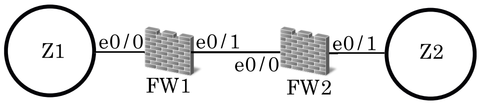
Fig. 3. Parallel-firewall Conduit definition.

The Parser builds the Zone-Conduit model using *firewall-only paths* to define conduits. An example conduit (C1) between 2 zones with a single-firewall path is shown in Figure 2. This case is almost trivial, but the question of how to map a network to zones and conduits has more complex cases.

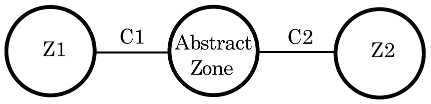
When two zones are connected by parallel links, the ANSI/ ISA standard allows them to be modelled as multiple conduits. But, multiple conduits imply multiple policies could exist between these zones, when only one is possible from the strict interpretation of a *zone*. So, we define a *single conduit* to implement the single policy relationship (*e.g.,* C2 in Figure 3). The resultant conduit policy $p_R = p_{Q \cup T}$, where $Q, T$ are the packet sets allowed by the policies (*i.e.,* ACLs) of the parallel firewalls $FW1$, $FW2$. Autoconfiguration is simplified by this single-conduit representation, since any policy between the two zones is enforced by the firewalls within conduit C2 only.

Firewall paths can also include firewalls in series (Figure 4(a)). This 'back-to-back' firewall architecture is one of the industry recommended security architectures [10] where defence-in-depth is achieved by using different vendors' devices. The ANSI/ISA guidelines lack clarity on how to define zones and conduits to precisely capture the distinct policy requirements of these serial firewalls (*e.g.,* FW2 may have logging enabled while FW1 may not).

A single conduit containing both firewalls exists, if we dismiss the inter-firewall link. But, for automation, a single conduit hinders precise configuration of the distinct firewalls.

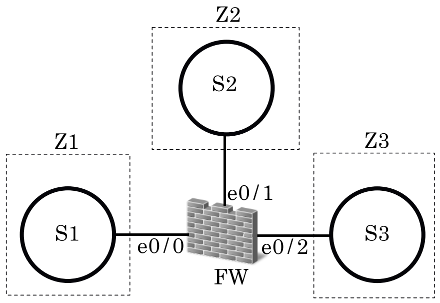


(a) Zone-Firewall model for 2 zones separated by serial firewalls.

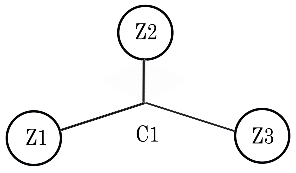


(b) Zone-Conduit model.

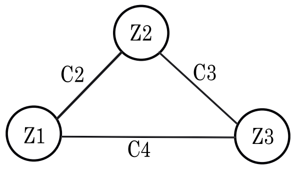
Fig. 4. Serial-firewall Conduit definitions.



(a) Zone-Firewall model for 3 zones separated by a firewall.



(b) Hyper-graph Zone-Conduit model.



(c) Simple-graph Zone-Conduit model.

Fig. 5. Conduit-definition alternatives.

We treat this connecting link as a separate zone, to overcome the specification shortfall. It is referred to as an *Abstract-Zone* in the absence of any real network devices within it (Figure 4(b)). The approach creates two separate conduits (C1 and C2), each containing one firewall. Autoconfiguration can now leverage the distinct conduits to unambiguously specify the individual firewall policy requirements.

A conduit may also inter-connect more than two security zones [5]. ANSI/ISA guidelines lacks clear specification on appropriate conduit definitions in such circumstance. Consequently, the example Zone-Firewall model depicted in Figure 5(a), could be modelled using a hyper-graph (Figure 5(b)). In this model, the firewall (FW) is located inside the hyper-edge conduit C1 which has *one-to-many zone-communication* paths. This complex conduit can implement multiple security policies; between Z1 and Z2, Z2 and Z3 and Z3 and Z1. Catering for this complexity requires the conduit to track the participating zones per policy. There is also no clear mapping of the ACL rules enforcing each policy to the firewall interfaces. Hence, the hyper-graph conduit model is difficult to use in firewall autoconfiguration.

To simplify the complexities of hyper-edge conduits, we propose to generate a Zone-Conduit model that consists of only *one-to-one zone-communication* paths (Figure 5(c)). Each conduit now implements a single security policy between two zones. The simple design also requires each conduit to only contain the firewall interfaces attached to its connecting zones (*e.g.,* C2 contains e0/0 and e0/1). A conduit path now reveals the exact firewall interfaces and their layout with respective to the connecting zones, enabling easy placement of required ACL rules. Consequently, the choice of simple-edge conduits, allows us to enforce a strict *1:1 mapping* between conduits and policies. This restriction yields a precise high-level specification, useful for firewall autoconfiguration.

Simple-edge conduits may seem to reduce the expressive power of the policy description. But, the power it buys over hyper-edge conduits is that it forces administrators to tie policy to a single edge (*i.e.,* two zones).

This logical method of conduit-definition can lead to multiple conduits sharing the same firewall in their mitigation offering (*e.g.,* C2, C3, C4 share FW in Figure 5(c)).

In summary, a single conduit need not always map to a single firewall. In fact one-to-many and many-to-one mappings between conduits and firewalls are more useful for high-level security specification. However, our recommendation for firewall autoconfiguration is that one conduit should always implement a single relationship between only two zones.

### D. Firewall Management Access Control

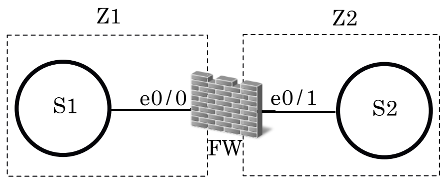
Firewalls also provide secure, authorised network management access to themselves, supplementary to offering mitigation capabilities to zones. ANSI/ISA standards lack clear direction on how zone and conduit concepts should be used to capture firewall management traffic requirements. This is a critical shortfall, because if management of the firewall is compromised, the entire system is compromised.

Our original study [27] discussed the alternatives available to address the issue in detail. Considering a firewall interface to be a part of a zone it is directly connected to (Figure 6(a)) prevents enforcing traffic restrictions from that zone to the firewall. Sharing the entire firewall (*i.e., all interfaces*) with all connecting zones (Figure 6(b)), does not overcome this shortfall. However, placing the firewall management in a new security zone on its own (Figure 6(c)), captures the real situation well and allows enforcing firewall management traffic restrictions from each zone. For instance, in Figure 6(c), we can block HTTP access from zone Z1 to the firewall by disallowing the service from Z1 to the *Firewall-Zone*.

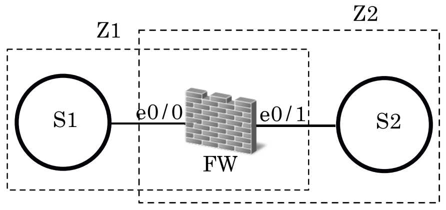
Our introduction of a dedicated Firewall-Zone simplifies management policy specification and hence, autoconfiguration. Firewall-management and non-management traffic can now be considered equally, but specified separately. Of course additional security mechanisms (*e.g.,* authentication) are required, but these are outside the scope of this analysis.

### E. Carrier Network Abstraction

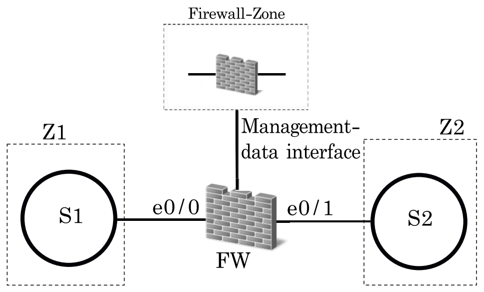
Real networks often utilise a Carrier Network (CN) provided by a telecommunication service provider to interconnect geographically dispersed sites. This is prevalent in SCADA networks which control distributed field-site equipment from



(a) Firewall interfaces in zones.



(b) Firewall shared by zones.



(c) Dedicated Firewall-Zone.

Fig. 6. Firewall-Zone alternatives for 2 subnets S1 & S2 separated by a firewall.

a centralised control centre over, for example, a leased line Wide Area Network (WAN).

The traffic relayed via the CN is controlled by the security policies between the zones within the two interconnecting sites (Figure 7). We model the interconnectivity provided by a CN by abstracting its underlying implementation details via a *Carrier-Zone* as shown in Figure 7.

### *F. Service-Flow Views*

A service-flow view is a directed-graph of zones that can initiate and/or accept that service protocol. The Parser generates these for the protocols; IP, TCP, UDP and ICMP *etc.*, broken down by port and zone as applicable. The output views are graphical representations based on GraphML [18], readily viewable using tools that support the format such as yEd [39].

We generate the *explicit service-flow views* by processing the firewall ACLs. In doing so, intra-ACL and inter-ACL interactions that stem from rule-overlaps need to be considered.

The outcome of a pair of interacting rules depends on several factors: the rule order; the level of overlap (*i.e.,* partial, full overlap or subset); and the rule actions. Depending on the extent of rule overlap, an intra-ACL rule interaction can be a *generalisation*, a *shadowed-rule*, a *partial-overlap* or a *conflict* [2]. Our original study [27] describes each rule-overlap type in detail. Algorithm 1 describes how we process the rule overlaps within an ACL to derive an intra-ACL interaction free version (*i.e.,* ACL_V1) of each ACL. The algorithm has time complexity $O(n^2)$ where $n$ is the number of ACL rules.

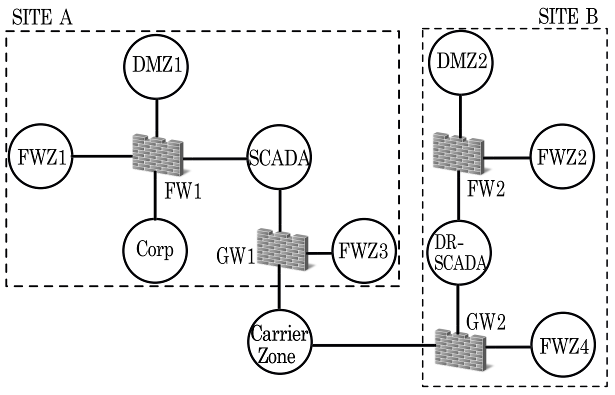


Fig. 7. Carrier-Zone interconnecting geographically dispersed sites.

The Parser also accounts for the inter-ACL interactions that alter an ACL rule's intended behaviour. It analyses potential inter-ACL interactions of each ACL_V1 using Algorithm 2, to derive a second version (ACL_V2) that now reflects the *net-effect* of all rule interactions possible for a given network. The time complexity of Algorithm 2 is $O(pnm+pn^2)$ where $n$ is the average number of rules in an ACL, $m$ is the average number of valid paths between a zone pair in the Zone-Firewall model and $p$ is the number of ACLs in the network.

The net effect of each ACL (*i.e.,* ACL_V2) is then used by the Parser to generate explicit service-flow views that depict the overall services enabled by each ACL.

Qian *et al.* [26] have also proposed an algorithm to derive intra-ACL interaction free versions of ACLs but do not classify the interactions as shadows, generalisations, partial overlaps and conflicts. Also their proposed algorithms do not derive inter-ACL interaction free versions of the ACLs, but, rather determines the net-traffic enabled or disabled between a given source and destination.

The Parser also processes implicit rules based on interface security levels and builds an implicit service-flow view. This service-flow view depicts inherently broad (*i.e.,* generic) IP traffic enabled between the zones. Special Cisco configuration commands that permit firewall management traffic above ACLs, are also parsed and corresponding service-flow views are created.

Redefining existing policy at a high-level is seen as requiring more effort than simply analysing the deployed rule-sets [2]. The Parser makes this task easier by automatically deriving the high-level security policy implemented in the network using service-flow views. The derived high-level policy can then be used in an autoconfiguration system as input to generate network-device configurations. Some human intervention may be required to verify the generated configurations enforce the intended policy (as suggested by Guttman [19]). But, the automated capability allows to consistently generate anomaly-free, firewall configurations for a network as its composition of heterogenous firewalls changes with time.

### *G. VLAN Considerations*

VLANs operate at the Ethernet layer (*i.e.,* layer 2) and have no understanding of the traffic 'state'. VLAN tags are hence

**Algorithm 1** Process Intra-ACL Interactions

```
input: An ACL consisting of potentially interacting rules.
output: A rule-set containing intra-ACL interaction free rules.

Step
1. For each rule in input ACL, find all rules that precede and
     overlap it. Two ACL rules r1, r2 overlap if every field in r1
     forms either a subset, superset or is equal to the corresponding
     field in r2, i.e., $\forall i$; $r1[i] \otimes r2[i]$ where $\otimes \in \{\subset, \supset, =\}$ and
     $i \in \{source\_ip, source\_port, dest\_ip, dest\_port, protocol\}$.
2. Derive the net effect of each rule in ACL (i.e., r1), and its
     preceding and overlapping rules (e.g., r2) as:
     (i) NULL; if r1 is shadowed by r2 (i.e., $\forall i$; $r1[i] \subset r2[i]$).
     (ii) (r1-r2); if r1 is a generalisation of r2,
          (i.e., $\forall i$; $r1[i] \supset r2[i]$).
     (iii) r1 - (intersection of r1, r2); if rules partially overlap.
     (iv) any of the above; if rules conflict (i.e., rules overlap
          but their actions mismatch, so, update r2's action to r1's
          and use Step 2 recursively to determine net effect).
3. Build a new rule-set consisting of the net rules.
```

**Algorithm 2** Process Inter-ACL Interactions

```
input: An ACL that potentially interacts with other ACLs
          in the network, a list of all network ACLs and
          the Zone-Firewall model of the network.
output: A rule-set depicting the net-effect of the original ACL.

Step
1. Find all ACLs that interact with provided ACL by:
     (i) considering each ACL rule's source, destination zones
          and identifying all elementary paths from source to
          destination in the Zone-Firewall model.
          These paths should exclude Firewall-Zones.
     (ii) locating the network ACLs that lie in each
          elementary path found.
2. Compute the net effect of provided ACL and each interacting
     ACL by deriving rule-wise net effect.
     So, rule r1 of input ACL and r2 of interacting ACL
     will render r1's net effect as:
     (i) r1; if r1 is shadowed by r2; or r1, r2 do not overlap; or
          r1 is a generalisation of r2 and both are deny rules; or
          r1 and r2 partially overlap and both are deny rules.
     (ii) r1-r2; if r1 is a generalisation of r2 and both
          are not deny rules.
     (iii) r1-(intersection of r1 and r2); if r1 and r2 partially
          overlap and both are not deny rules.
     (iv) NULL; if r1 and r2 are both deny rules
          and r1 is shadowed by r2.
     (v) any of (i)-(iii) above; if r1 and r2 conflict
          (update r2's action to r1's and use Step 2 recursively
          to determine net effect).
3. Build a new rule-set consisting of the net rules.
```

easily spoofed and there are many hacking tools [28], [31] designed to bypass their security.

VLANs also logically segregate the same underlying physical network. A purely logical segregation between a SCADA and a corporate network is highly inadequate and should be avoided [10]. For example, a DoS attack within a VLAN-separated corporate network can render the SCADA network useless, since the (shared) physical network gets saturated with malicious corporate traffic. Hence, VLANs are a less reliable mechanism for enforcing security policy in a SCADA network, and we handle them in our Parser with warnings to make the user aware of the fact.

## IV. A Series of Case Studies

Obtaining real firewall configurations from working SCADA networks is very hard due to the sensitive nature of the data. We were able to obtain such configurations from seven SCADA networks. A high-level summary of the Systems Under Consideration (SUCs) is provided in Table I. Four are described in detail in [27]. We add three more here; one is discussed in detail.

Due to security concerns and non-disclosure agreements, a modified version of each real SCADA network analysed is presented for discussion. Effort has been taken to ensure that the security strategies and underlying issues uncovered remain intact. However, details such as IP addresses are anonymised.

### *A. Analysed Configuration Data*

We choose to describe SUC 2 in detail here because it employs a defence-in-depth security architecture and has a wide set of security features enabled. The SUC used a Cisco ASA firewall. It's configuration was extracted in August 2011. The configuration consisted of 432 lines with 12 ACLs averaging 16 conventional rules each. The SUC is shown in Figure 8 and employs a single Cisco ASA 5510 firewall to separate the corporate network from $\alpha$-`WAN`; the entry point to the SCADA network. The firewall also had a active/standby fail-over unit configured as backup.

The firewall has four physical interfaces operational, two of these were divided into sub-interfaces supporting five VLANs each with 802.1Q trunking enabled. These interfaces each pointed to the following subnets.

*The $\alpha$-WAN:* High security WAN responsible for providing access to the SCADA network (`SCADA`).

*The $\beta$-WAN:* Medium security WAN responsible for sending data updates to the $\beta$-DMZs.

*The $\gamma$-WAN:* Low security WAN responsible for sending data updates to the $\gamma$-DMZ.

*The Old-WAN:* High security WAN that hosts VPN clients for providing secure access to the $\alpha$-DMZs.

*The $\alpha$-Demilitarised-Zone1 ($\alpha$-DMZ1):* High security DMZ that receives updates from the SCADA network and $\alpha$-DMZ2. The DMZ facilitates inbound access from the corporate network and other DMZs. $\alpha$-DMZ1 hosts Web servers, a Network Monitoring Station (NMS) and a time server.

*The $\alpha$-Demilitarised-Zone2 ($\alpha$-DMZ2):* High security DMZ responsible for receiving updates from the SCADA network, which in-turn logs syslog messages and alerts to the $\alpha$-DMZ1. This DMZ hosts a data historian.

*The $\beta$-Demilitarised-Zone1 ($\beta$-DMZ1):* Medium security DMZ that periodically synchronises data with the NMS in $\alpha$-DMZ1. It also shares updates with the corporate network and $\beta$-DMZ2. This DMZ hosts Web, RDP and SNMP servers.

*The $\beta$-Demilitarised Zone2 ($\beta$-DMZ2):* Medium security DMZ that allows RDP clients in the corporate network to access its servers.

TABLE I
HIGH-LEVEL SUMMARY OF THE SUCs (* BACKUP FIREWALL, ** CONVENTIONAL FORMAT, LoC - LINES OF CODE)

| SUC | Configuration date | Firewall type | Firewalls | Gateways | Zones | Average LoC | ACLs | Average rules per ACL** |
|---|---|---|---|---|---|---|---|---|
| 1 | Sep 2011 | Cisco IOS | 2 | 1 | 2 | 1360 | 8 | 237 |
| 2 | Aug 2011 | Cisco ASA | 1(2)* | 5 | 11 | 432 | 12 | 16 |
| 3 | Oct 2011 | Cisco PIX | 2 | 2 | 5 | 125 | 8 | 6 |
| 4 | Mar 2011 | Cisco ASA | 1 | 2 | 4 | 819 | 3 | 80 |
| 5 | Apr 2015 | Cisco ASA | 1(2)* | 2 | 12 | 853 | 12 | 677 |
| 6 | Apr 2015 | Cisco ASA | 1(2)* | 3 | 13 | 900 | 8 | 1034 |
| 7 | Jul 2015 | Cisco ASA, Cisco FWSM | 2(4)* | 4 | 15 | 860 | 13 | 724 |

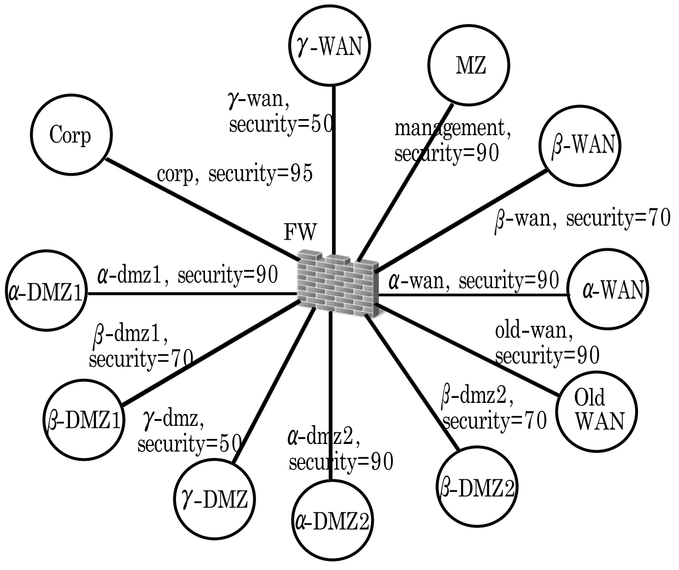


Fig. 8. Single firewall SCADA network studied in SUC 2.

*The γ-Demilitarised-Zone (γ-DMZ):* Low security DMZ that periodically synchronises data with the NMS in α-DMZ1. It also shares updates with the corporate network. The DMZ hosts Web, RDP and SNMP servers.

*The corporate network (Corp):* Provides access to business applications and the Internet. Corp hosts NTP servers, a TFTP server, an Email server and several RDP clients.

*IT Management Network (MZ):* Hosts multiple workstations that allow network-device management via Telnet.

All subnets were configured to accomodate up to 254 hosts. There are 25 individually access-controlled hosts through the firewall, across the 11 subnets above.

Varying security levels are assigned to each firewall interface (from 0 to 100), based on the security policies of the network connected to that interface. The corp interface has a security level of 95 (Figure 8). The α-wan, management, old-wan, α-dmz1 and α-dmz2 firewall interfaces have a security level of 90. The β-wan, β-dmz1 and β-dmz2 interfaces are assigned a level of 70. γ-wan and γ-dmz interfaces are assigned a security level of 50.

With the highest security-level assigned, Corp is classified as the highest-trust zone. Classifying so, prevents the zone from being accessed by a lower-trust zone by default (*i.e.,* without the use of ACLs). Corp is also allowed to access all other zones by default.

Old-WAN, Management, α-DMZ1, α-DMZ2 and α-WAN are classified as second highest-trust zones (assigned security level=90). Traffic can flow between these five zones by default. This is because same security-level traffic is permitted between their firewall interfaces, via the Cisco CLI command: 'same-security-traffic permit inter-interface'.

β-WAN, β-DMZ1 and β-DMZ2 are classified as third highest-trust zones (assigned security level=70). All zones with security level >70 have access to these six zones by default. These zones can also access each other by default.

γ-DMZ and γ-WAN zones are classified as least-trust zones in the group (assigned security level=50). All zones with security level >50 have access to these by default. Additionally these zones can also access each other by default.

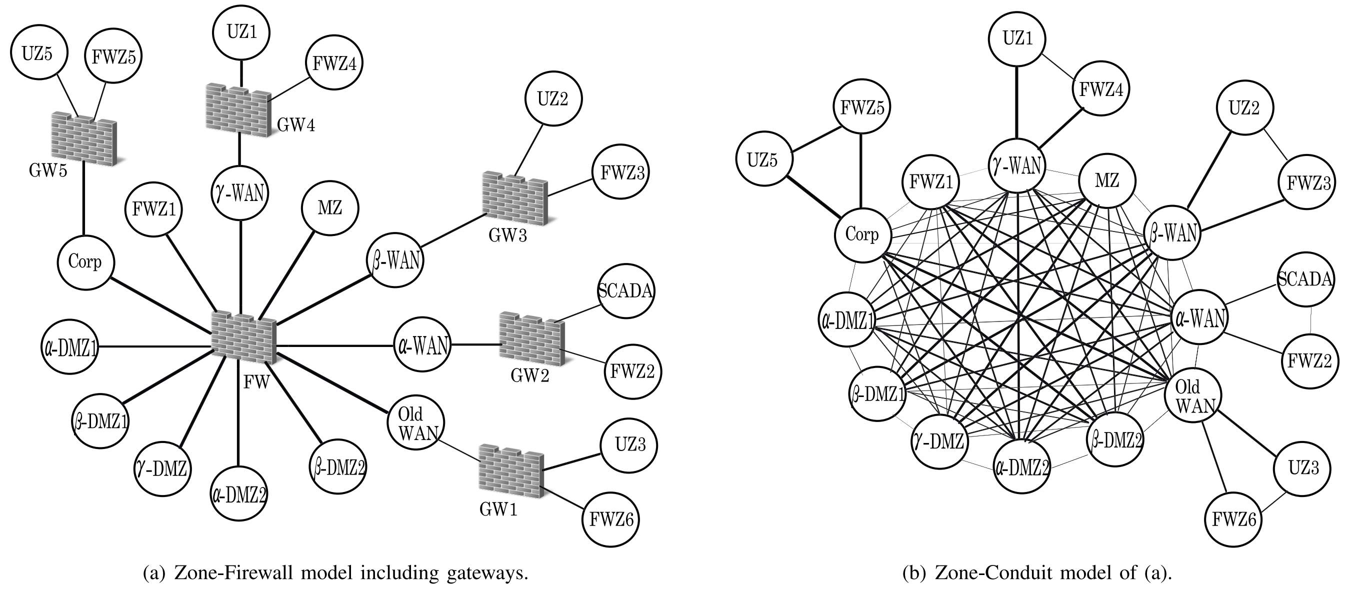
Cisco extended ACLs are used with conventional rules. Of the 12 ACLs defined in the configuration, only 10 were in use. Eight of these were assigned inbound to the following firewall interfaces: *corp*, *α_dmz1*, *β_dmz1*, *γ_dmz*, *α_dmz2*, *β_dmz2*, *α_wan* and *management*. Two ACLs were assigned outbound on *α_dmz1* and *β_dmz2* interfaces. Once assigned to firewall interfaces, the ACLs render the security levels assigned to the interfaces obsolete. This is because the default security-level based filtering behaviour is overridden by the ACLs [14].

Corp has an extended ACL; acl_corp_in assigned inbound on the corp firewall interface. This ACL is used to restrict traffic flow from Corp to the other zones. Without this ACL, *all-IP* traffic flow is enabled by default through the security-level assigned to corp interface. Particularly, the ACL restricts access to SCADA, α-DMZ1 and the medium and low security DMZs: β-DMZ1 and γ-DMZ.

α-WAN has an ACL; acl_α_wan_in assigned inbound on its firewall interface. Similar to in acl_corp_in, this ACL also restricts traffic flow from α-WAN to the other zones, which otherwise enables *all-IP* traffic through security-levels. Particularly, the ACL enables restricted access to α-DMZ1, α-DMZ2 and Old-WAN based VPN clients.

Similar ACLs are assigned on the high, medium and low security DMZ interfaces to restrict access to and from their shared servers.

The SCADA network studied enforced Network Address Translation (NAT) on all traffic traversing from high-security (*i.e.,* inside) interfaces to low-security (*i.e.,* outside) interfaces by default. This meant unless the traffic matched a specified NAT rule, it was blocked by the firewall [14]. NAT exemptions (provided via ACLs) were used to omit certain traffic having to undergo NAT translation. Traffic that matched the ACL rules did not have their inside addresses translated when traversing



(a) Zone-Firewall model including gateways.

(b) Zone-Conduit model of (a).

Fig. 9. Security models of the network.

outbound. A NAT-exemption also allows both translated and remote hosts to initiate connections [14].

Interfaces at the same security level as well as traffic traversing from outside to inside interfaces were not required to use NAT to communicate, since dynamic-NAT or Port Address Translation (PAT) was not configured on any of the interfaces. The *stateful* nature of the Cisco ASA firewall permitted all legitimate return traffic.

The firewall also had an active/standby fail-over configuration enabled via a dedicated Ethernet link. So, an identical standby firewall could take over the functionality of the primary unit on failure [14]. The primary unit automatically replicates its configuration to the standby unit once special configuration commands are issued [14]. Hence, the auxiliary unit mirrors the primary unit's configuration. The standby unit is also accessed only via the primary unit, so both are managed as one using a single Firewall-Zone.

The firewall configuration indicated the use of a remote Syslog server located off `Corp` with *info-level* traps enabled. SNMP messages were also sent to a NMS on authentication, link-up and link-down events. Basic threat detection was also enabled on the firewall. This captured ACL statistics, recording packet denial rates and connection limit exceeds.

The firewall configuration also revealed that only static routes were in use.

### B. Results and Industry Best Practice Implications

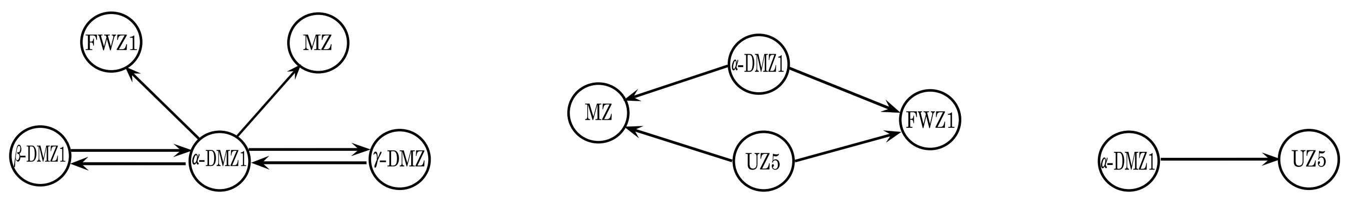
Figure 9(a) shows the Zone-Firewall model of the SUC generated by parsing the configuration data using the Zone-Conduit extensions in Section III. In generating the model, we uncovered five gateways (GW1-GW5), and we assume here the conservative security option: each gateway enforces a security policy and hence behaves as a firewall (their configurations were unavailable).

Consequently, our assumption yields two additional zones attached to a gateway: one is the Firewall-Zone that represents the control plane of the gateway, the other zone encompasses all subnets reachable via the gateway (as per the static routes). We group these subnets to a single zone because without actual gateway configurations we cannot accurately identify the disjoint zones the subnets may reside in.

We can directly compare this network segregation model against the industry recommended SCADA architectures in [10] and [34]. Accordingly, the model falls into the category of a 'single firewall employing demilitarised zones'. The model complies with the critical requirement that the SCADA-Zone should not be directly connected to the Internet (reachable only via gateway GW2). Figure 9(b) shows the corresponding Zone-Conduit model including Firewall-Zones (FWZs).

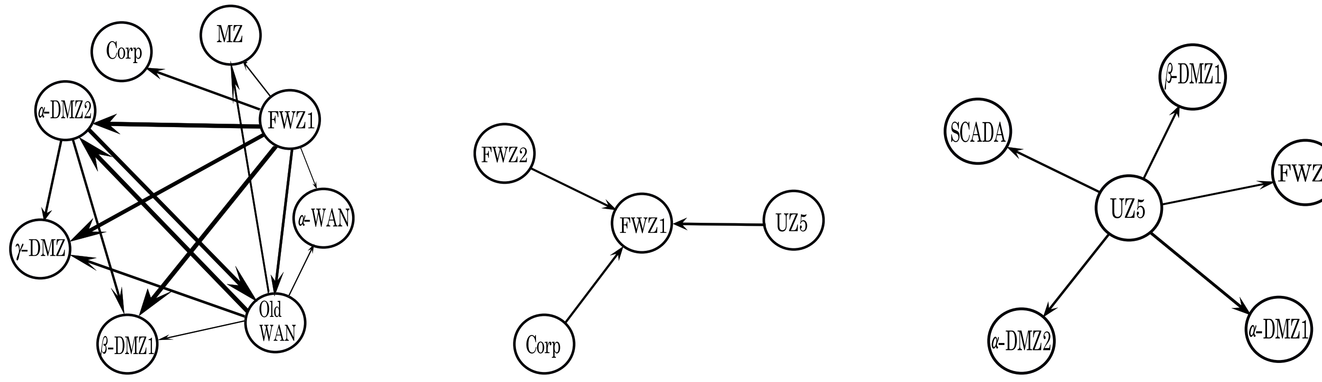
*Security Violations:* We compared the explicit service-flow views generated by our Parser (Figure 10, Figure 11(c)) against best practices in [10]. Doing so, revealed several significant violations caused by incorrectly (explicitly) enabled services through the firewall. For one, *all-TCP* traffic was explicitly enabled inbound to SCADA from UZ5 (Figure 11(c)). This allowed inherently unsafe protocols such as FTP, HTTP and Telnet to enter the SCADA-Zone. HTTP for instance, is known to transport worms and attacks, so allowing it inbound significantly increases the vulnerability of the SCADA-Zone to cyber attack. For another, *all-IP* traffic was also (explicitly) enabled inbound to the SCADA-Zone from $\alpha$`-DMZ1`, with similar adverse effects.

Figure 11(a) depicts the intended service flows of the security-levels assigned on the firewall interfaces. It shows that *all-IP* traffic flow from `Corp` to $\alpha$`-wan` is intended. $\alpha$`-wan` provides access to the SCADA-Zone, so, the intent could have made the SCADA-Zone highly vulnerable to cyber attack. But, the ACLs assigned on the firewall interfaces actually override the security-levels and mitigate this oversight. However, incorrect security-levels lead to a confusing issue of precedence of rules. So, security levels must be assigned consistently to comply with industry best practices: *i.e.,* `Corp`



(a) HTTP. (b) Telnet. (c) SMTP.

Fig. 10. Explicit Service-flow Views of HTTP, Telnet and SMTP traffic.



(a) Generic IP (implicit). (b) Telnet (implicit). (c) Generic TCP (explicit).

Fig. 11. Service-flow Views of generic IP, Telnet and generic TCP traffic.

should be assigned a lower trust level than α-wan and not vice versa. By assigning security levels correctly, we can additionally enforce defence-in-depth. For instance, if the ACLs on α-wan and corp interfaces were accidentally removed, correct security levels would continue to protect the SCADA-Zone by only allowing IP traffic flow to be initiated from the more trusted α-wan to Corp and not vice versa.

The SUC employed less-secure VLANs to separate Corporate traffic from SCADA traffic. Industry recommends against such virtual segregation of traffic for obvious reasons.

Implicit rules enabled firewall management traffic in the SUC. As shown in Figure 11(b) Telnet was utilised for firewall management, but, Industry recommends against using the protocol for its lack of support for encryption and robust authentication.

HTTPS was also enabled between the Management-Zone, Firewall-Zone and α-DMZ1, but *excluded* the SCADA-Zone. This safe protocol is encouraged by [10] to carry Web traffic to and from the SCADA-Zone, but was not used.

NAT-control was also incorrectly used. The feature is meant to be turned on when NAT translations are required, and not for simple traffic-flow control. There were no explicit NAT translations (*i.e.,* through dynamic NAT or PAT) but only a few NAT-exemptions; a clear misuse of the feature. ACLs should be used to control traffic flow instead of NAT [14].

*Configuration Inefficiencies:* Our Parser identified ACL entries with incorrect source and/or destination IP addresses. In some cases the order of the addresses were wrong while in others they were simply invalid. There were 14 such occurrences that wasted configuration space. There were also two unused ACLs in the configuration file that wasted 27 lines.

TABLE II
SUMMARY OF CPU TIMES TAKEN BY OUR ALGORITHMS TO PROCESS INTRA-ACL INTERACTIONS AND INTER-ACL INTERACTIONS IN THE SUCS. THE ALGORITHMS WERE RUN ON A STANDARD LAPTOP COMPUTER; *e.g.,* INTEL CORE CPU 2.8-GHZ COMPUTER WITH 8 GB OF RAM RUNNING UBUNTU LINUX 14.04.3. (*CONVENTIONAL RULE FORMAT)

| SUC | ACLs ($p$) | Average rules per ACL ($n$)* | intra-ACL interaction processing CPU time (seconds) | inter-ACL interaction processing CPU time (seconds) |
|---|---|---|---|---|
| 1 | 8 | 237 | 386.7 | 231.7 |
| 2 | 12 | 16 | 34.5 | 112.9 |
| 3 | 8 | 6 | 5.8 | 4.3 |
| 4 | 3 | 80 | 176.3 | 3.9 |
| 5 | 12 | 677 | 511.9 | 126.7 |
| 6 | 8 | 1034 | 608.1 | 712.4 |
| 7 | 17 | 724 | 527.4 | 480.3 |

TABLE III
INTER-ACL INTERACTIONS SUMMARY

| ACL1 | ACL2 | Interaction | Count |
|---|---|---|---|
| acl_β_dmz1_in | acl_α_dmz1_out | shadow | 1 |
| acl_α_wan_in | acl_α_dmz1_out | generalisation | 12 |
| acl_α_wan_in | acl_α_dmz1_out | shadow | 4 |
| acl_γ_dmz_in | acl_α_dmz1_out | shadow | 1 |

The Parser also found two intra-ACL redundancies within acl_α_ DMZ1_out through rule processing. These were generalisations that consisted of a specific narrow rule in the ACL with an overlapping, broad rule further down the list.

We described in Section III, how the intra-ACL and inter-ACL interaction processing algorithms in our Parser each had time complexity $O(n^2)$ and $O(pnm + np^2)$ respectively.

TABLE IV
SUMMARY OF SECURITY BEST-PRACTICE VIOLATIONS FOUND IN THE SUCS (✗ INDICATES A VIOLATION)

| Best practice | SUC1 | SUC2 | SUC3 | SUC4 | SUC5 | SUC6 | SUC7 |
|---|---|---|---|---|---|---|---|
| Physical separation of Corporate and SCADA traffic | ✓ | ✗ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Backup firewalls available | ✗ | ✓ | ✗ | ✗ | ✓ | ✓ | ✓ |
| Secure remote access of SCADA network | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Secure firewall management | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Prohibit Internet-SCADA direct traffic transition | ✗ | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ |
| Prohibit Corporate-SCADA direct traffic transition | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Prohibit inbound insecure traffic (*e.g.,* DNS, HTTP) to SCADA | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ | ✗ |
| Clear documentation of ACL rules | ✗ | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Firewall firmware is current | ✓ | ✓ | ✓ | ✗ | ✗ | ✗ | ✗ |
| Firewalls have active vendor support | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |

Table II shows the CPU times for running these algorithms for the SUCs, on a standard laptop computer (*e.g.,* Intel Core CPU 2.8-GHz computer with 8GB of RAM running Ubuntu Linux 14.04.3). The data shows that the computational cost of parsing the firewall configurations using our algorithms is pragmatic.

There were also 18 inter-ACL interactions involving four ACLs, as summarised in Table III. These consisted of 12 generalisations and six shadowed-rules. These inter- and intra-ACL interactions consisted of wasteful rules that prevented from maintaining a concise firewall configuration.

## V. CASE STUDY VARIATIONS

We discussed SUC 2 in detail in the previous section, we now summarise and incorporate findings from the other cases. Table IV presents a summary of the security best-practice violations found across the SUCs.

In all cases except in SUC 7 and SUC 2, the SCADA-Zone was physically separate from the Corporate-Zone. Industry recommends this physical separation of SCADA traffic from insecure corporate traffic for obvious reasons. SUC 7 and SUC 2 violated this recommendation and employed less secure VLANs to virtually separate the two zones.

SUCs 4-7 were running firewall software that were three years out of date or older at the time their configurations were extracted. Many security vulnerabilities were identified in these versions since their release. For instance, the firewall software version in SUC 5 and SUC 6 (ASA Version 8.2) had a memory-leak vulnerability that allowed remotely authenticated users to carry-out DoS attacks [13]. Some of these vulnerabilities were not applicable to the firewall's configuration context. But, enabling a restricted set of firewall functionality is a poor excuse for not upgrading/patching firewall software regularly. Firewall configurations evolve continuously, hence vulnerabilities that were initially inapplicable, could eventually expose a network to cyber threats.

All firewall hardware except one in SUC 7 had active vendor support. The decade-old Cisco FWSM in SUC 7 was completely out of vendor support. Yet, it was used in a production SCADA environment! Regular patching and upgrading of systems is less practical in mission critical SCADA networks, but this is a poor excuse for allowing critical components like firewalls to go un-maintained for years!

SUC 5 and SUC 6 also had active/standby fail-over configurations enabled through dedicated Ethernet links. Alarmingly, this redundancy of critical network infrastructure components was lacking in three real-world SCADA plants: SUC 1, 3 and 4. In SUC 7, the Cisco FWSM was embedded within a Catalyst 6500 switch that had built-in redundancy.

SUC 5 and SUC 6 allowed remote access to the SCADA-Zone from the Corporate-Zone. VPNs and multi-factor authentication were not used as per recommended industry practices in enabling this access. Bypassing secure remote-access practices exposed the SCADA-Zone to threats and vulnerabilities in the Corporate-Zone. Some remote-access protocols enabled in the SUCs also violated industry best practices (*e.g.,* RPC).

SUCs 4-7 also had varying security levels assigned to each firewall interface (from 0 to 100), based on the security policies of the network connected to that interface. The firewall configurations in these case studies also used *object-groups* to classify devices, protocols and ports into groups. These groups were then applied to ACLs in a single rule.

Object-groups can be very useful in practice since the size of a conventional ACL can be large. For example, the ACLs studied in SUC 6 contained on average 1034 conventional rules each, as opposed to 24 object-group based rules. With frequently changing rules, managing lengthy conventional ACLs is a difficult task. Using object-group based ACLs makes ACLs smaller, more readable and easier to configure and manage. This and *security-level* are nascent attempts to provide higher-level security policy description, but they don't map clearly to the Zone-Conduit model.

In all SUCs, ACL-rule comments were present. However, in some cases (*e.g.,* SUC 7) these comments failed to provide clarity on the purpose or requirement of the rules. In others (*e.g.,* SUCs 5-6), the comments were comprehensible but included incorrect or obsolete rule descriptions (*e.g.,* the intended traffic flow described mismatched that implemented by the rule).

Each SUC included static routes, additionally SUC 2 utilised EIGRP while SUC 3 utilised OSPF.

SUC 1 consisted of a pair of serially-connected firewalls and hence included an Abstract-Zone in its Zone-Conduit model. SUC 3 included a Carrier-Zone that interconnected two geographically dispersed SCADA networks.

Intra-ACL interactions were present in all SUCs analysed except for SUC 3. These interactions were predominantly shadowed-rules and generalisations.

Inter-ACL interactions were also present in all SUCs except for SUC 5 and SUC 6. These were mostly shadowed-rules caused by identical rules in distinct ACLs, collectively enabling traffic flow between zones.

In all the SUCs studied, traffic restrictions were enforced between the Corporate-Zone and the SCADA-Zone through the presence of one or more conduits in the Zone-Conduit model. SUC 1 and SUC 2 allowed direct communication between the SCADA-Zone and the Internet-Zone, violating industry best practices. SUCs 3-7 met the industry recommendations disallowing such direct communication.

In most cases, insecure protocols (*e.g.,* HTTP, Telnet) were explicitly disallowed inbound to the SCADA-Zone. But in all cases, such traffic was then implicitly allowed to reach the same zone, breaching industry recommendations.

We also found explicitly allowed NTP, DNS and FTP traffic transiting directly between SCADA and Corporate Zones, in most cases. This setup easily exposed the SCADA-Zone to threats or attacks in the Corporate-Zone.

Implicit rules were used across all case studies to enable firewall management traffic. SSH was utilised in SUC 1 and SUCs 5-7 while Telnet was used in SUC 1 and SUC 2. HTTP was utilised in SUCs 3-7 for firewall management.

## VI. Discussion

Our first case study [27] found many flaws in firewall configuration that need to be fixed and that current mechanisms do not work. These case studies allowed us to identify the requirements for autoconfiguration of firewalls. Most prominent is a good set of high-level abstractions.

Implicit rules are nascent attempts by firewall vendors to provide high-level abstractions, but are too restrictive in that you cannot write flexible rules. For example, Cisco security levels allow quick and easy access between internal and external firewall interfaces, but lack the flexibility to specify detailed traffic restrictions. Hence the large ACLs supplementing these levels.

Likewise, the ANSI/ISA Zone-Conduit abstraction proved too flexible, allowing alternate ways of defining zones and conduits to cater for business models. The abstraction is good when used by humans, but for automation we need precision.

A good abstraction is therefore, a tussle between these two approaches. It should provide clear mapping between policies and networks, with some restrictions, but also the required amount of flexibility.

For one, the standards allow 1:n or n:1 mapping between conduits, firewalls and policy. We argue that maintaining a 1:1 mapping between policies and conduits leads to a simple, understandable and useful abstraction for high-level policy specification. Otherwise the ambiguity might lead to specification of policies that breach the restrictions implied by a zone, *i.e.,* a single policy within a zone.

For another, when firewalls are in series, the best practice is vague on how to define zones and conduits. We argue that there needs to be an Abstract-Zone to capture the distinct policies that could be reasonably applied to the two firewalls.

ANSI/ISA best practices also lacked specification on how to precisely capture firewall management traffic. Adding a Firewall-Zone addressed the problem.

On average, a firewall configuration in our case studies had 764 lines. Hence, it is easy to accidentally leave-in lapsed ACL rules, when the composition of network devices changes with time. These rules can lead to potentially dangerous latent errors and keep firewall configurations from being concise and up-to-date. An autoconfiguration process should therefore, allow detection and removal of obsolete rules.

ACLs and implicit rules can have complex interactions. For example, a rule within an ACL can overlap and conflict with other preceding rules in the same ACL, potentially altering or even reversing its intended effect. Smaller rule-bases have been observed to contain fewer errors [36], [37]. But, with lengthy ACLs, maintaining interaction free rule-sets manually is nearly impossible and requires automation to achieve it.

Implicit rules can override ACLs, rendering the effort tendered to the careful design and deployment of ACLs obsolete. For example, an ACL can be configured on a Cisco ASA firewall interface to block inbound HTTP traffic to the firewall. If an implicit management policy on the device (*e.g.,* defined using *http* command) allows the service, the management policy overrides the ACL [14].

We assert that the use of implicit rules should be avoided where possible, and replaced with explicit ACL based access control instead. This will be the difference in being able to automatically generate clear, simple and effective firewall configurations from confusing, complex and ineffective ones.

Our case studies did not comprise large, complex networks. This simplicity implies that the task of configuring the network firewalls should be relatively easy. Additionally, due to the critical nature of the industrial control equipment protected by these firewalls, one expects them to be correctly configured. As we found, this is far from reality. Even in the simplest of cases, SCADA firewalls are still badly configured! Needless to say, what chances do we have of correctly configuring firewalls in a large, complex network?

We have taken a significant step towards making firewall autoconfiguration a reality. By refining the ANSI/ISA Zone-Conduit abstraction we make it precise and complete. Firewall configuration is complex and difficult as re-asserted by our case studies. The refined Zone-Conduit model, provides a precise, simple yet rich high-level abstraction for firewall policy description, that is suitable for automation.

## VII. Conclusion

ANSI/ISA best practices provide a Zone-Conduit model for firewall policy specification, but, the model lacks key aspects for automation of firewall configuration. We propose several extensions to address these missing pieces.

Several additional requirements of auto-configuration were also identified through this research. Namely, eliminating ACL interactions, avoiding implicit rules, removal of lapsed rules and platform or device specific compilation of high-level policy. We hope to consolidate these requirements further to formulate a feasible firewall auto-configuration prototype.

## Acknowledgment

The authors would like to thank Dr. Morris Sloman and anonymous reviewers for their insightful feedback.

## References

[1] E. S. Al-Shaer and H. H. Hamed, "Discovery of policy anomalies in distributed firewalls," in *Proc. IEEE INFOCOM*, vol. 4. Hong Kong, 2004, pp. 2605–2616.

[2] E. Al-Shaer, H. Hamed, R. Boutaba, and M. Hasan, "Conflict classification and analysis of distributed firewall policies," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 10, pp. 2069–2084, Oct. 2005.

[3] *The Practitioner's Guide to Deploying, Optimizing and Managing Next Generation Firewalls*. AlgoSec, Inc., Boston, MA, USA, 2012.

[4] C. J. Anderson *et al.*, "NetKAT: Semantic foundations for networks," *ACM SIGPLAN Notices*, vol. 49, no. 1, pp. 113–126, 2014.

[5] *International Society of Automation, Security for Industrial Automation and Control Systems, Part 1: Terminology, Concepts and Models, ANSI/ISA-62443-1-1 (99.01.01)-2007*, Res. Triangle Park, Durham, NC, USA, 2007.

[6] Y. Bartal, A. Mayer, K. Nissim, and A. Wool, "Firmato: A novel firewall management toolkit," *ACM Trans. Comput. Syst.*, vol. 22, no. 4, pp. 381–420, 2004.

[7] S. M. Bellovin and R. Bush, "Configuration management and security," *IEEE J. Sel. Areas Commun.*, vol. 27, no. 3, pp. 268–274, Apr. 2009.

[8] I. C. Bertolotti, L. Durante, L. Seno, and A. Valenzano, "A twofold model for the analysis of access control policies in industrial networked systems," *Comput. Stand. Interfaces*, vol. 42, pp. 171–181, Nov. 2015.

[9] E. Byres, "Using ANSI/ISA-99 standards to improve control system security," White paper, Tofino Security, May 2012.

[10] E. Byres, J. Karsch, and J. Carter, *NISCC Good Practice Guide on Firewall Deployment for SCADA and Process Control Networks*, Northern Ireland Social Care Council, Belfast, U.K., 2005.

[11] *NGX R65 CC Evaluated Configuration User Guide*, CheckPoint Softw. Technol. Ltd., San Carlos, CA, USA, 2008.

[12] F. Chen, A. X. Liu, J. Hwang, and T. Xie, "First step towards automatic correction of firewall policy faults," *ACM Trans. Auton. Adap. Syst.*, vol. 7, no. 2, p. 27, 2012.

[13] *Cisco Systems*. Accessed on Jan. 5, 2016. [Online]. Available: http://www.cvedetails.com/vulnerability-list/

[14] *Cisco ASA Series CLI Configuration Guide, 9.0*, Cisco Systems, Inc., San Jose, CA, USA, 2013.

[15] *Cisco ASA 5585-X Adaptive Security Appliance Architecture*, White paper, Cisco Systems, San Jose, CA, USA, May 2014.

[16] *FireMon*. Accessed on Jul. 15, 2016. [Online]. Available: https://www.firemon.com

[17] M. G. Gouda and A. X. Liu, "Structured firewall design," *Comput. Netw.*, vol. 51, no. 4, pp. 1106–1120, 2007.

[18] Graph Steering Committee, GraphML, 2003.

[19] J. D. Guttman, "Filtering postures: Local enforcement for global policies," in *Proc. IEEE Symp. Security Privacy*, Oakland, CA, USA, 1997, pp. 120–129.

[20] J. D. Guttman and A. L. Herzog, "Rigorous automated network security management," *Int. J. Inf. Security*, vol. 4, nos. 1–2, pp. 29–48, 2005.

[21] C. D. Howe, *What's Beyond Firewalls?* Forrester Res., Cambridge, MA, USA, 1996.

[22] A. X. Liu, "Formal verification of firewall policies," in *Proc. Int. Conf. Commun.*, Beijing, China, 2008, pp. 1494–1498.

[23] R. M. Marmorstein and P. Kearns, "Firewall analysis with policy-based host classification," in *Proc. Large Installation Syst. Admin. Conf.*, Washington, DC, USA, 2006, p. 4.

[24] A. Mayer, A. Wool, and E. Ziskind, "Fang: A firewall analysis engine," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, 2000, pp. 177–187.

[25] T. Nelson, C. Barratt, D. J. Dougherty, K. Fisler, and S. Krishnamurthi, "The Margrave tool for firewall analysis," in *Proc. Large Installation Syst. Admin. Conf. (LISA)*, San Jose, CA, USA, 2010, pp. 41–58.

[26] J. Qian, S. Hinrichs, and K. Nahrstedt, "ACLA: A framework for access control list (ACL) analysis and optimization," in *Communications and Multimedia Security Issues of the New Century*. New York, NY, USA: Springer, 2001, pp. 197–211.

[27] D. Ranathunga, M. Roughan, P. Kernick, N. Falkner, and H. Nguyen, "Identifying the missing aspects of the ANSI/ISA best practices for security policy," in *Proc. 1st ACM Workshop Cyber Phys. Syst. Security (CPSS)*, Singapore, 2015, pp. 37–48.

[28] S. A. Rouiller. (2003). *Virtual LAN Security: Weaknesses and Countermeasures*. [Online]. Available: http://uploads.askapache.com/2006/12/vlan-security-3.pdf

[29] O. Rysavy, J. Rab, and M. Sveda, "Improving security in SCADA systems through firewall policy analysis," in *Proc. Federated Conf. Comput. Sci. Inf. Syst. (FedCSIS)*, Kraków, Poland, 2013, pp. 1435–1440.

[30] K. Salah, K. Elbadawi, and R. Boutaba, "Performance modeling and analysis of network firewalls," *IEEE Trans. Netw. Service Manag.*, vol. 9, no. 1 pp. 12–21, Mar. 2012.

[31] SANS. *Intrusion Detection FAQ: Are There Vulnerabilites in VLAN Implementations? Vlan Security Test Report*. Accessed on Jul. 15, 2016. [Online]. Available: https://www.sans.org/security-resources/idfaq/vlan.php

[32] *Skybox Security*. Accessed on Jul. 15, 2016. [Online]. Available: https://www.skyboxsecurity.com/products/skybox-firewall-assurance

[33] Statista. *Global Market Share Held by Security Appliance Vendors From 1st Quarter 2012 to 4th Quarter 2015*. Accessed on Jan. 5, 2016. [Online]. Available: http://www.statista.com/statistics/235347/global-security-appliance-revenue-market-share-by-vendors/

[34] K. Stouffer, J. Falco, and K. Scarfone, *Guide to Industrial Control Systems (ICS) Security* (NIST Special Publication, 800–882). Gaithersburg, MD, USA: Nat. Inst. Stand. Technol., 2008, p. 16.

[35] A. Wool, "Architecting the Lumeta firewall analyzer," in *Proc. USENIX Security Symp.*, Berkeley, CA, USA, 2001, pp. 85–97.

[36] A. Wool, "A quantitative study of firewall configuration errors," *Computer*, vol. 37, no. 6, pp. 62–67, Jun. 2004.

[37] A. Wool, "Trends in firewall configuration errors: Measuring the holes in Swiss cheese," *IEEE Internet Comput.*, vol. 14, no. 4, pp. 58–65, Jul./Aug. 2010.

[38] L. Yuan *et al.*, "FIREMAN: A toolkit for firewall modeling and analysis," in *Proc. IEEE Symp. Security Privacy*, Berkeley, CA, USA, 2006, pp. 199–213.

[39] yWorks. *yEd Graph Editor Manual*. Accessed on Jul. 15, 2016. [Online]. Available: http://yed.yworks.com/support/manual/index.html

**Dinesha Ranathunga** received the bachelor's degree in computer systems engineering from the University of Adelaide, Australia, in 2003, where he is currently pursuing the Ph.D. degree in applied mathematics with the School of Mathematical Sciences. His research interests include SCADA network security, firewall autoconfiguration, policy-based network management, and software defined networking.

**Matthew Roughan** received the Ph.D. degree in applied mathematics from the University of Adelaide in 1994. He has worked for the Co-Operative Research Centre for Sensor Signal and Information Processing, in conjunction with DSTO, at the Software Engineering Research Centre, RMIT, and the University of Melbourne, in conjunction with Ericsson, and at AT&T Shannon Research Labs, USA. He is with the School of Mathematical Sciences, University of Adelaide, SA. His research interests range from stochastic modeling to measurement and management of networks like the Internet. He has authored over a 100 refereed publications, half a dozen patents, and has managed over a million dollars worth of projects. He and his coauthors were a recipient of the 2013 Sigmetrics Test of Time Award, and his work has been featured in *New Scientist* and other popular press.

**Hung Nguyen** received the Ph.D. degree in computer and communication sciences from the Swiss Federal Institute of Technology, Lausanne, Switzerland. He joined the Teletraffic Research Centre, University of Adelaide in 2012. His research interests include software defined networking, 5G, network measurements, tomography, and privacy preserving techniques. He has published over 40 refereed papers on the above areas.

**Phil Kernick** received the B.Eng. (Elec.) degree from the University of South Australia in 1988 and the B.Sc. (Hons.) degree from Flinders University in 1996. He is the Co-Founder of CQR Consulting, the largest independent information security consultancy in Australia. He has presented at all major SCADA security conferences in Australia. His research interest is in SCADA security.

**Nickolas Falkner** received the Ph.D. degree in discovery and classification of information in large systems from the University of Adelaide, where he is a Senior Lecturer with the School of Computer Science. His research interests include automated network configuration, applications of cryptography, and data stream management. He is also active in educational research, with a focus on increasing student participation, retention, and enthusiasm.