Towards Standardising Firewall Reporting

Dinesha Ranathunga[1], Matthew Roughan[1], Phil Kernick[3], and Nick Falkner[2]

[1] ARC Centre of Excellence for Mathematical and Statistical Frontiers, University of Adelaide, Australia
[2] School of Computer Science, University of Adelaide, Australia
[3] CQR Consulting, Australia

Abstract. Rubin and Greer stated that "The single most important factor of your firewall's security is how you configure it." [18]. However, firewall configuration is known to be difficult to get right. In particular domains, such as SCADA networks, while there are best practice standards that help, an overlooked component is the specification of firewall reporting policies. Our research tackles this question from first principles: we ask what are the uses of firewall reports, and we allow these to guide how reporting should be performed. We approach the problem by formalising the notion of *scope* and *granularity* of a report across several dimensions: time, network elements, policies, *etc.*

Keywords: SCADA security, firewall autoconfiguration, Zone-Conduit model, firewall reporting, report granularity, granularity dimension

1 Introduction

"The only thing worse than no coffee is bad coffee."

The quote is germane because the only thing worse than no security is bad security. The appearance of security permits behaviour that would be more carefully scrutinised in an insecure domain. For instance, we might allow unpatched systems behind a firewall, confident in the blanket of protection provided.

In the context of complex networks, it is terribly easy to either misconfigure firewalls or, for some part of the security setup to otherwise malfunction, either breaking the network or reducing its security.

There are many workable steps to guard against such failure, but one of the most important is to constantly examine the security mechanisms of the network. This can be performed externally, and by using reports from the devices involved.

However, even in domains such as Supervisory Control and Data Acquisition (SCADA) networks, which control the distributed assets of many critical systems such as power generation and water distribution, the standards for reporting and analysing firewall data are scant and vague. At the same time, SCADA networks often incorporate highly vulnerable devices. The Programmable Logic Controllers (PLCs) that control physical devices such as gas valves have highly

constrained memory and computational power. Today, they often include network functionality such as a TCP/IP stack, but exclude sophisticated security functionality. Thus, firewalls are a critical component of SCADA networks. As Rubin and Greer note [18] it is vital that these firewalls are configured correctly.

Most firewalls have the ability to generate reports in highly flexible and diverse ways (*e.g.,* logs, traps, alarms and alerts). But we lack clear direction in what they *should* report. This paper starts the discussion on how reporting policy should be specified by

i. Looking at the ways firewall reports can be used, and how that impacts reporting requirements.
ii. From this, formalising notions of the *scope* and *granularity* of reports.
iii. Considering which use cases are actually reasonable uses of firewall resources.

We find that reporting at the right granularity is key to saving valuable firewall and network resources while achieving the required use case outcomes. In some cases a low (*i.e.,* coarse) granularity such as the reporting of configuration changes at a firewall level is sufficient. In other cases a higher (*i.e.,* refined) reporting granularity in the likes of recording individual IP packets is required. The volume of firewall reports generated usually increases with granularity and can have a detrimental effect on a SCADA firewall's primary function- traffic filtering. Identifying reporting granularity requirements can help justify whether a potential use case is best served if it is conducted by a SCADA firewall or not.

We also find that reporting needs to be coupled with security policy specification to be of use. This important principle has been overlooked in the best practices [2, 21]. Coupling reporting and policy also provides security managers with a *single source of truth* for easy reference.

2 Background

Firewall configuration is a critical activity, yet hard to get right. It involves training in proprietary and device specific configuration languages, and the production of long and complex device configurations.

The problem of firewall configuration is well studied. Fang [11] and Lumeta [22] are interactive management and analysis tools that run queries on firewall rules to check for errors. Tools such as these have been used to analyse working firewalls configurations and have shown that critical errors are very common, even in quite simple networks [16, 23, 24]. We argue that this difficulty extends to firewall reporting, and that this component is also important to get right.

Firewall vendors have introduced many products and security management tools with varying levels of sophistication [3, 4, 6, 9]. However, what these tools have typically done is to increase the range of options. They provide more power, but we still lack guidelines on how to best utilise this power.

In the context of SCADA networks, critical systems are protected by firewalls, and so best practice guidelines have been prepared to help. They suggest a number of high-level policy abstractions for SCADA networks [1]. In particular,

we refer to the *Zone-Conduit abstraction* as a way of segmenting and isolating the various sub-systems. Firewall reporting is not considered in detail, however. We will demonstrate that it is useful to link firewall reporting to policy, so we include a brief description of this abstraction.

A *zone* is a logical or physical grouping of an organisation's systems with similar security requirements so that a single policy can be defined for all zone members. A *conduit* provides the secure communication path between two zones, enforcing the policy between them [1]. Security mitigation mechanisms (*e.g.,* firewalls) implemented within a conduit helps it to resist Denial of Service (DoS) attacks, and preserve the integrity and confidentiality of network traffic. A conduit could consist of multiple links and firewalls but, logically, is a single connector.

3 Firewall Reporting Use Cases

Firewalls can generate logs, alerts, traps, and provide other information for instance via SNMP (Simple Network Management Protocol) polling. We term all these informational content together as *reporting*.

Most firewalls allow a panoply of highly flexible and configurable reports. It is hard to even categorise all of the possibilities without some framework. It is useful here to consider the classical hierarchy of knowledge that $data \rightarrow information \rightarrow knowledge \rightarrow decisions$. If the data does not inform decisions its value is zero, so simply collecting data is insufficient motivation for the cost of collection. Hence we base our framework on *use cases*. That is, we frame our discussion of what to report based on how the data will be used.

We reviewed the literature on firewall reporting in both SCADA and Corporate domains [10, 19, 20] and classified the various uses of the reports. Out of necessity, we grouped certain activities together, and simplified the nomenclature. Our resulting classification is as follows:[4]

Accounting measures network usage, for instance, to monitor network bandwidth usage for network planning.

Network-based Intrusion Detection (ID) is the near real-time monitoring of network traffic to identify traffic from unauthorised sources [20]. Near real-time implies without a significant delay (allowing transmission and automated processing delays), usually up to a few minutes [20]. Network-based ID particularly helps administrators to secure a SCADA network by blocking the attack before it causes damage to critical systems.

For example, some firewalls can monitor traffic that passes key network locations, and generate alarms when known attack signatures are present. Other firewalls might send traffic data to an ID system for analysis.

Post-mortem analysis is the analysis of an *incident* after the fact. Analysis aims to identify root causes in order to prevent future occurrences. In a cyber-physical domain, analysis may be mandatory to meet regulatory compliance.

[4] At this point we classify existing activities and do not consider which of these cases is a sensible use of firewall resources.

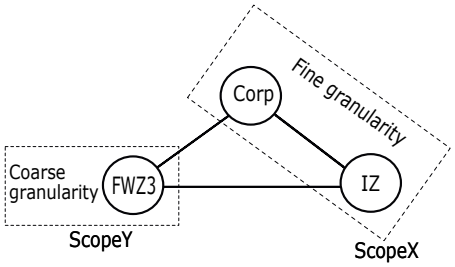


Fig. 1: Illustration of scope vs granularity.

Security policy verification checks a firewall rule base for invalid rules. Configuration inefficiencies in firewalls are largely due to obsolete rules (*e.g.,* rules pertaining to a decommissioned server) and incorrect rules (*e.g.,* rules with source and destination IP address in wrong order) [16, 23]. Efficient firewall configurations can be maintained by identifying such invalid rules.

Troubleshooting operational issues is the debugging of network errors associated with firewalls, for instance, to identify the root cause(s) of connection problems through a firewall.

Given the primary function of a firewall is traffic filtering, we need to carefully consider what secondary reporting functions should be conducted by a firewall. To do so, we need to identify the reporting granularities associated with each use case above. We discuss these requirements in detail next.

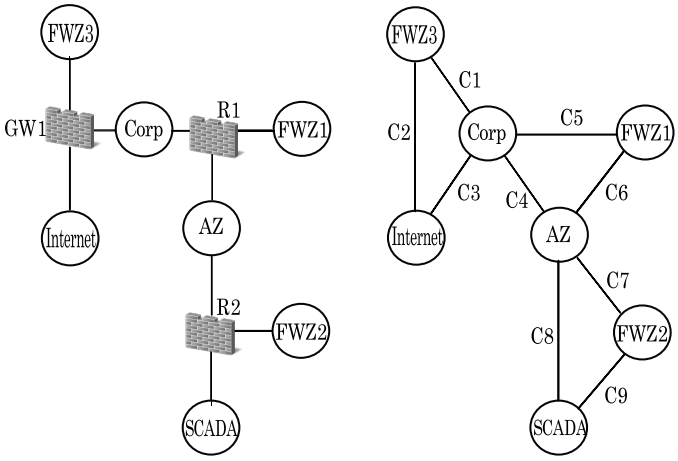
4 Report Granularity and Scope

Firewall reports can be thought of as a form of *lossy data compression*. A firewall will internally register myriad events and data. A report, whether it be an alarm or log-message or polled counter, is a compressed form of this information.

Ideally, we would lose as little information as possible in compressing the data. Even the most effective compression mechanism (*e.g.,* JPEG images) allow some loss, but try to ensure that it is not important information. In the context of firewalls, we aim to initiate a discussion of what is important through discussing the *resolution* or *granularity* of reports[5]. Granularity refers to the finest level of the discrimination we can make. For instance, in an image, resolution or granularity is the pixel size – we can't separate objects in an image that are smaller. The first concern when compressing an image would be how many pixels do we need, and so this is a first step in considering firewall reports.

Related to granularity is *scope*. In image terms, this is analogous to field-of-view, *i.e.,* how widely is the data collected. We can talk about both scope and granularity with the same terminology, though the details can vary across the network as shown in Figure 1.

[5] Although we view the two terms as close to synonymous in this context, resolution is overloaded with meaning and so we prefer the term granularity.



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

Fig. 2: Security models of a SCADA network adapted from a case study [16].

To help refine our discussions, Figure 2 depicts the *Zone-Firewall* and *Zone-Conduit* models of an example SCADA network from an actual case study [16].

Most vendor firewalls support different reporting *verbosity* levels, *e.g.,* debug, info, warning [3, 4, 9, 15]. However, the terminology (for instance *warning* level) is not universal. We aim here for *vendor-independent* notions of reporting, and so will avoid these terms.

The terms also obscure the multiple dimensions of firewall reporting: for instance, the level could refer to nature of events with respect to the network, policy definitions, or time. Hence, in the next section, we will tease these aspects of reporting detail by considering granularity and scope with respect to the multiple aspects of a network that firewalls observe.

4.1 Granularity Dimensions

Network granularity is the level of detail resolved in a network. Firewall reports may need to resolve network-specific detail, often at a network-wide, Zone-Conduit, interface, prefix or IP address level. We describe these levels of detail in order of increasing granularity next.

(i) ***Network-wide-level*** refers to an entire network. At this level we might only distinguish between events internal, and external to the network. This might be important for intrusion detection or post-mortem analysis, and for general accounting of network usage.

(ii) ***Zone-Conduit-level*** resolves information per zone. For instance, in intrusion detection, we might like to understand which components of the network (*e.g.,* SCADA or Corporate) were potentially compromised, and

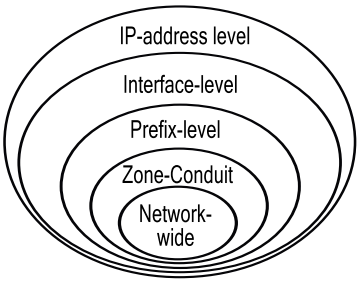


Fig. 3: Network granularity hierarchy: coarse granularities are subsets of the finer.

the zone concept requires that all network elements in a zone are treated equally (*i.e.,* even if only one IP address is *observed* to be attacked, we assume all might have been, at least through proxy attacks).

(iii) ***Prefix-level*** resolves information per block of IP addresses. Where zones are not used, we might naturally wish to group data for analysis by subnet.

(iv) ***Interface-level*** resolves information per firewall interface. Large firewall installations could have multiple interfaces serving a zone, and/or there is the question of where traffic triggers an event: when it enters a firewall, or leaves it. This level of resolution is needed when such fine discrimination is important, for instance, in troubleshooting firewall problems.

(v) ***IP-address-level*** resolves information per network host (*e.g.,* a server). This is the finest network-granularity usually possible, though sometimes virtual machines could be hosted by separate ports at a single IP address.

We seek to define granularity in terms of a hierarchy in which the information present in a coarser granularity can be derived from the finer. This hierarchy is shown in Figure 3, and note that it should recur in other granularity dimensions.

Policy granularity is the level of detail resolved in a security policy. Firewall reports may need to resolve policy detail, often at a global, zone, service, rule or sub-rule level. We describe these below.

(i) ***Global-level*** resolves rule-set wide information for a policy. *i.e.,* it separates events or traffic into *in-policy* and *out-of-policy* traffic. Out-of-policy traffic includes 'defective' traffic (*e.g.,* packets with ill-formed headers). Such data can help track the amount of attack traffic aimed at a network.

(ii) ***Zone-policy-level*** refines the global-level by resolving traffic that is in- and out-of-policy at the zone level, so that we can see, for instance the amount of attack traffic breaching the outer level of a defence-in-depth.

(iii) ***Service-policy-level*** level of detail resolves policy details per service either allowed or explicitly denied in a firewalls security policy.

(iv) ***Rule-level*** refines the Service-policy-level by noting that a "service" might be defined by several rules. One common example is a service such as WWW, which could include HTTP and HTTPS. Another is a service that requires requests to be allowed in one direction, and responses in the other.

Traffic measurement granularity is one of the common reports firewalls may produce concerns the traffic they observe, either generally, or because it matches some particular pattern (*e.g.,* it is invalid, or matches a particular type of attack) or translation (*e.g.,* IP header changes by NAT, or IP payload encryption by IPSec in a VPN tunnel). We describe here the granularity with which traffic can be reported.

(i) ***Counter-level*** resolves traffic measurement details into certain bins, and then counts the amount of traffic into these bins. The granularity of the bins is defined by the previous two dimensions of network- and policy-level. For instance, in policy verification, the key interest is in whether a policy rule has hits (*i.e.,* if IP packet matches have been recorded for the corresponding ACL rules).
The nature of the information recorded in each bin might vary: commonly we might measure packets, bytes, or connections (or all three). However, we lose all of the meaning within the packets.

(ii) ***Connection-level*** records traffic per connection. It is very similar to flow-level, which we discuss in more detail below.

(iii) ***Flow-level*** resolves traffic measurements per IP flow: *i.e.,* by grouping a series of connected packets, typically those with the same IP protocol, IP source and destination addresses, and TCP source and destination ports.
For instance, in accounting for network usage, it may be necessary to resolve the source IP from the destination IP address of a flow. This would resolve the upload and download traffic of a host or subnet. Such flows are almost analogous to connections, but are easier to collect.

(iv) ***Packet-header-level*** records each packets' headers. It includes information such as the *flags* and *fragment-offset* fields in the IP header, which indicate whether IP fragments can produce a complete datagram.

(v) ***Packet-level*** requires us to store whole packets, or at least a substantial part of each packet. This allows us to reconstruct, for instance, the details of a particular attack.

The traffic granularity discussion illustrates one important issue. Traffic measurements are collected using many different *mechanisms*, and the mechanism is often related to the type of measurement. For instance, SNMP is often used to collect counter-level data, and NetFlow to collect flow-level data.

However, this is just how it is done now. We aim to define universal concepts, and leave the mechanisms of their implementation to the engineers building devices. The goal of device and vendor independence requires this approach of decoupling what we want to measure, from how it is measured.

Performance measurement granularity is another of the common reports firewalls may produce concerns performance metrics. Of the many measurements attributable to a firewall or its interface performance, the memory and CPU utilisation are most significant. Both metrics can help identify ongoing attacks (*e.g.,* DoS) or other problems. Similarly, packet queue-length and packet loss

measurements can indicate attacks through overloaded queues resulting from high traffic, *e.g.,* as might result from a DoS attack.

We describe here the granularity with which performance can be reported.

(i) ***Firewall-level*** records performance measurements per individual firewall. For instance, post-mortem analysis might use firewall CPU and memory utilisation to help resolve the root cause of a network problem.
(ii) ***Process-level*** records performance measurements per firewall software process. For instance, when a new policy is pushed to a firewall, the policy processing module may fail to load the policy, if the memory required for the policy exceeds the module's allowance [5]. Module's CPU and memory utilisation reports can help troubleshoot firewall problems in this scenario.
(iii) ***Interface-level*** records performance metrics specific to a firewall interface. For instance, in troubleshooting firewall errors, it may be necessary to resolve reverse DNS lookup errors per firewall interface.

Temporal granularity is the level of detail (*e.g.,* measurements or counts) resolved per set of time instances. In this context, granularity is not exactly the right concept, but we will explain below.

(i) ***Per-T*** records detail per time interval T. The measurements could be counter measurements of traffic, configurations changes and so on. Common intervals T vary from
  – daily;
  – hourly; and
  – minutes;

  and potentially finer intervals. However, the mechanisms used to support granularities down to minutes are often not suitable (*e.g.,* SNMP polling) for measurements at per-second granularity and finer, so we qualitatively separate these into the following category.
(ii) ***Near-real-time*** means reporting data as soon as possible given the limitations of processing and network speed. Delays of up to seconds are reasonable, but not minutes. While the previous granularity can be reported through *pull* mechanisms, nearly all near-real-time support is provided by *push* mechanisms, *e.g.,* traps or notifications or alarms.

Note that it is non-trivial to set up accurate distributed clocks, but doing so should be seen as vital for any level of temporal granularity in order that any data collected is meaningful.

Operational measurements is another of the common reports firewalls can produce concerns the events they observe. Granularity does not seem to be a useful concept in this domain because there is no reason we would ever store data at a coarser granularity than its origin. The standard means to describe the level of detail in event logs are vague terms such as *debug*, *error*, *warning*,

informational. They specify the "level" of events to be reported, not the level of detail of the actual reports, but this notion of level is too context dependent to be universally agreed. A warning in one domain is an error in another.

Errors can also be ambiguous to interpret. Some report abnormal behaviour that require no response action (*e.g.,* traffic with a broadcast destination address dropped). Others relate to a significant breakdown in operation that needs urgent attention. We increase clarity by classifying the latter type of errors as *failures*.

We studied corporate firewall logs and identified some common events that occur in real firewall deployments. Logs from five such firewalls were analysed. These aggressively reported on traffic denials at the external firewall interfaces.

Of over 4.5 million log messages in a month, 97.58% were traffic denial events (both in- and out-of-policy); 1.18% were power-state changes; 0.78% were VPN-state changes; 0.42% were *failures*; and 0.03% related to firewall-user activity.

Motivated by our findings, and avoiding bland ambiguous terms in favour of precision, we classify events by their nature [6]:

(i) ***State transitions:*** reports of state changes of the firewall and its components (*e.g.,* interface power-up, software process startup- such as for a HTTP or VPN server).
(ii) ***User activity:*** reports of user login activity, commands executed by a user logged into firewall including actions to change its configuration, and their consequences, including any errors or warnings.
(iii) ***Failures:*** reports of breakdown of normal operation (*e.g.,* VPN tunnel failure), that require immediate action. These are not regular state changes and hence are excluded from *state transitions*.
(iv) ***Diagnostics:*** reports of self-tests deployed on the firewall and their outcomes (*e.g.,* test failover interface).
(v) ***Table dumps:*** tables of comprehensible information (*e.g.,* active NAT translations) or potential events, internal to a firewall.

As an illustration of the concepts, Figure 4 is an example summary of the firewall reporting requirements against network-granularity for the use cases discussed. It shows that in each use case a *Zone-Conduit level* network-granularity is required.

4.2 Relation between Reporting Granularity Dimensions

Reporting granularity dimensions may not be entirely orthogonal. We describe the relation that exist between some dimensions below.

Traffic measurement and Time: Packets, flows and connections can arrive at variable rates in time to a firewall. So, the traffic granularity required for each use case can restrict the temporal granularity achievable (and vice versa). For instance, if a use case requires reporting every IP packet, that typically implies

[6] Note that some types of events are already implicitly included in traffic or performance measurements, for instance, denied packet counts.

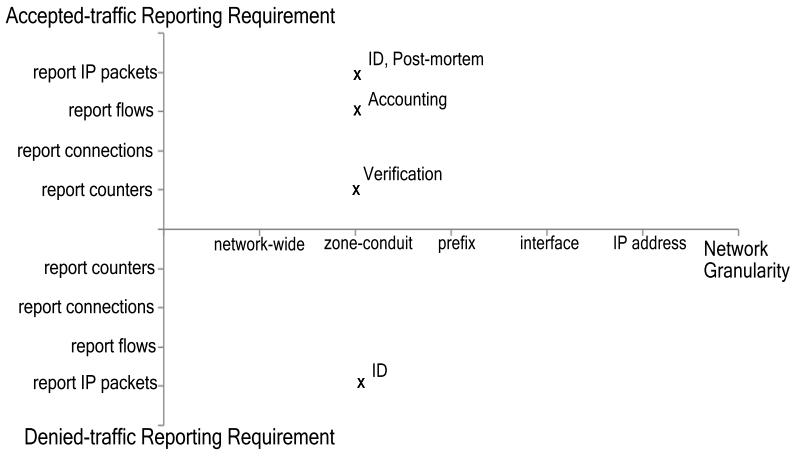


Fig. 4: Traffic reporting requirement vs network granularity: depicts the *zone-conduit level* network-granularity required for each reporting use case.

reporting at a per millisecond temporal granularity. Conversely, if a firewall reports per millisecond, a packet or flow level of granularity can be achieved.

Traffic measurement and Policy: Traffic granularity requirements can also restrict the policy granularity achievable (and vice versa). For instance, reporting every IP packet implies a rule-level policy granularity. Conversely, if reporting at a rule-level, a packet or flow or connection-level traffic granularity is achievable.

Traffic management and Network: Traffic granularity requirements can also restrict the network granularity achievable (and vice versa). For instance, reporting every connection implies an address-level network granularity. Conversely, if reporting at an address-level, a connection or flow or packet-level traffic granularity is achievable.

For all reporting use cases, the network granularity required is *Zone-Conduit level*. But the relation between traffic and network granularity implies that the actual network granularity achieved is dependent on the traffic granularity of a use case. For instance, intrusion detection requires reporting every IP packet. Adhering to this requirement, a finer *address-level* network-granularity is achieved. A similar network-granularity is obtained for all other use cases.

5 Reporting Cost

Report generation costs CPU, memory and network resources. The cost can potentially impact the performance of a firewall, and compromise its primary function- traffic filtering, so it is an important factor to consider.

The cost depends on several factors: the granularity and scope; whether the reports are distributed or centralised; and the retention period. We have already discussed granularity and scope in detail: finer granularity measurements cost more. We discuss the other two issues below.

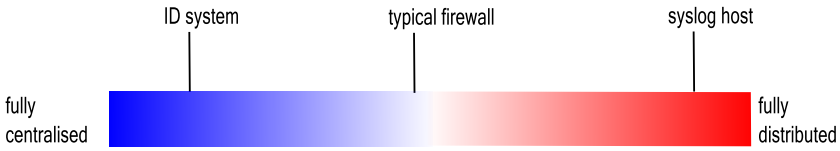


Fig. 5: Spectrum of centralised vs distributed report collection.

5.1 Distributed vs Centralised Analysis and Retention

Firewalls can be configured to report to several destinations including internal storage, NAS (Network-Attached Storage), a report server or historian, a Telnet or SSH session, or an email account [3,4,9].

Distributed collection and retention – by which we mean that the reports are collected and stored at the firewall – has a low network communications cost, but the trade-off is that it requires local storage. Firewalls rarely have very large internal storage, so it must be frequently overwritten or cleared [4].

Moreover, the data lacks use in this form. For example, accurate ID and troubleshooting require correlating reports from multiple sources (firewalls, routers, servers, *etc.*) and to manually extract these from each device is cumbersome.

At the other end of the spectrum lies centralised collection. In this strategy network devices perform minimal analysis of data: they just collect it and pass it to a single repository. Reports can then be analysed together. However, centralised collection creates potentially large volumes of network traffic.

Centralised collection can also introduce security vulnerabilities in a network. The centralised server, by its nature, must collect data across zones with different levels of security. However, many reporting mechanism (for instance syslog) are based on UDP, so a central syslog server would inherit any UDP vulnerabilities. Therefore, a central syslog server should not be located in a high-security zone (*e.g.,* in a SCADA zone) [2]. Another vulnerability stems from unsafe storage of sensitive information (*e.g.,* firewall-admin passwords) in the reports.

However, these vulnerabilities can be minimised with a carefully constructed collection strategy. For example, a syslog server can be placed in a DMZ and polled from the internal networks. Reports can also be sanitised to ensure that an attacker cannot obtain sensitive information. Doing so would allow each use case to reap the benefits of centralised collection, and reduce associated risks. However, this does require a careful and considered report architecture.

In reality, the two polarised extremes of decentralised and centralised are rare. There is actually a spectrum (Figure 5) where some processing and analysis is distributed but eventually summary data is brought to one or small number of data historians. For example, firewalls typically perform significant processing of network traffic prior to sending back some notifications to a central historian.

5.2 Report Retention

Reported information needs to be retained for some time determined by the requirements of the use case. There are cyber-security industry standards [12,13]

for data retention. The rationale for the lengths suggested are, however, not clear, and the standards also make a distinction between *reportable events* and other, without clearly specifying what is reportable.

Given the progressively declining electronic storage costs, and the original data collecting cost, we argue that, from a cyber-security industry point of view, it is more cost-effective today to retain firewall reports (almost) perpetually.

If storage really becomes a problem, then progressive reduction in granularity can be a useful tool. For instance see RRDtool [14].

However, we do consider minimum retention periods for each use case here.

Accounting for network usage reports are useful until the end of an usage evaluation period. Once processed and usage information is extracted, these reports can be purged. They should be retained for a minimum of 90 days [13].

Intrusion detection: Some attacks (*e.g.,* DoS, Port scans) can last longer than others. So, at a minimum, reports must be retained until the attack has passed. Post-attack retention also helps identify the types of attacks common in a network, so better defence mechanisms can be formulated for the future. Industry recommends retaining these reports for a minimum of 90 days [13].

Policy verification reports are useful until a verification period ends. Post report-processing, invalid rules are located and rectified, so reports can be purged. Industry recommends retaining these reports for a minimum of 90 days [13].

Post-mortem analysis reports should allow tracing back of, for instance, the origins of a SCADA network attack that circumvented intrusion detection systems. Doing so, may require processing of historical records that date back several years. These reports should be retained for a minimum of 3 years [12].

Troubleshooting reports are useful to monitor firewall configuration errors in near-real time. Once the error is rectified, they may be archived for future reference. Industry recommends retaining these reports for a minimum of 90 days [13].

6 What *Should* Firewalls Report?

The previous sections discussed what a firewall *could* report, and the use of that data. However, the ability to generate data does not mean we should. There is a cost to data collection, and some functions are better supported elsewhere.

For instance, firewalls have a limited perspective of a network so effective ID requires additional sensors (for instance at hosts and on wireless networks [20]). However, given such sensors why use the firewall at all? Collecting data at the firewall could compromise its main function through the cost of data collection, which may be amplified by a DoS attack. Moreover, a firewall, by its nature, is a visible target. It is far better to use passive (invisible) sensors for ID.

The conclusion is that a firewall is a poor source of data for ID.

Post-mortem analysis has fine granularity requirements, similar to ID, but lacks the near-real-time requirement. Hence, this use case is a marginally more acceptable use of firewall resources, but should not be implemented lightly.

Network usage accounting also requires traffic reporting but only at *flow-level* traffic-granularity. During a DoS attack, many flows can amplify the effect of the attack. But, if accounting focuses on accepted traffic, this effect is mitigated.

Policy verification is key to maintaining robust and efficient firewall configurations. The granularity requirements for this use case are quite coarse, and other devices cannot provide accurate information on packets that are matched by a firewall's policy, so this type of use case should definitely be supported.

Similarly, troubleshooting requires coarse information that can only be collected by the firewall, *e.g.,* an accurate change history can only be obtained from the firewall itself. Hence, firewall reporting should be enabled for this use case.

7 Implementation

We make the ideas discussed concrete, by presenting our implementation next. One of our goals is to extend *ForestFirewalls*: a high-level security policy specification system built using the Zone-Conduit model [17]. It aims to improve the efficiency and reliability of the SCADA firewall configuration process. This paper concentrates on reporting, so we describe the mechanism for the two use cases that require firewall reporting: (i) *policy verification* and (ii) *troubleshooting.*

Current firewall configuration platforms present "policy" and reporting as separate functions. Decoupling can be useful in some contexts to separate *structure* from *function*, *e.g.,* to separate security policy from the underlying network.

However, reporting and security policy are *inter-dependent* network functions. Scope and granularity are intimately related to the use of data, which, in turn, is related to the corresponding policies. Decoupling the two allows bad decisions to be made: for instance, addition of policies that aren't verified.

The Object Oriented Programming [7] paradigm makes it clear that *encapsulation* of related concepts and code together is vital for reliable and maintainable systems. A useful policy specification platform should encapsulate related specifications together. This implies that security policy and reporting specifications should be encapsulated together. Doing so, gives SCADA security managers a *single source of truth* to see 'who gets in and who doesn't' along with audit trails required to check the configuration. We show how this works in the following.

7.1 Reporting Policy for Verification

The granularity and scope requirements for a typical policy verification scenario are given in Table 1. We then list, in Table 2, the resulting reporting *attributes* grouped by granularity dimension. These are specified in the system as follows:

```
rule_group security_policy_rules {<rule1>, <rule2>,..., <ruleN>}

reporting_rule verify_a_rule { use_case=verification;
  granularity.policy={rule_or_group={<rule1>}};
  granularity.traffic={measurement={counter}; counter_type={connection};};
  granularity.temporal={per_hour};}

policy <policy-name>  { security_policy_rules; verify_a_rule; }
```

Table 1: Summary of reporting granularity requirements for policy verification.

Granularity dimension	Required granularity	Scope
Network	zone-conduit	network-wide
Policy	rule	global
Traffic-measurement	connection-count	counter
Temporal	per-hour	per-day
Performance	process	firewall

Table 2: Reporting policy attributes derived for policy verification.

Reporting attribute	Example value
use_case	verification;
granularity.network	{zone_or_group={SCADA}; traffic_direction={inbound};}
granularity.policy	{rule_or_group={rule1}; policy_action={permit};}
granularity.traffic	{measurement={counter}; counter_type={connection};}
granularity.temporal	{per_hour}
granularity.performance	{process}

A `rule_group` contains a set of security policy rules, and a `reporting_rule` object defines a set of reporting requirements. The `policy` object encapsulates the security policy rules and the reporting rule together. This facilitates reuse both of policies and reporting rules (which could come from a library), but encapsulates them together in the final specification.

7.2 Reporting Policy for Troubleshooting

A summary of the granularity requirements for this use case is given in Table 3. The list of specification attributes is given in Table 4. We use these attributes to specify troubleshoot reporting for one or more firewalls as:

```
rule_group security_policy_rules {<rule1>, <rule2>,..., <ruleN>}

zone_group FIREWALL_ZONES {<zone1>, <zone2>,..., <zoneM>}

reporting_rule debug_firewalls { use_case=troubleshoot;
    granularity.network={zone_or_group={FIREWALL_ZONES};
                         traffic_direction={inbound, outbound}};
    granularity.policy={rule_or_group={security_policy_rules}};
    granularity.performance={measurement={interface};
       performance_type={memory, CPU, packet_loss, queue_length}};
    granularity.temporal={near_realtime};}

policy <policy-name>  { security_policy_rules;  debug_firewalls; }
```

As before, the `reporting_rule` object for the use case includes the attributes, *e.g., interface-level* performance statistics and *near real-time* temporal-granularity. The reporting rule is encapsulated in the `policy` statement.

Table 3: Reporting granularity requirements for troubleshooting firewall errors.

Granularity Dimension	Required granularity	Scope
Network	Zone-Conduit	network-wide
Policy	rule	global
Temporal	near real-time	per-day
Performance	interface	firewall

Table 4: Reporting policy attributes for troubleshooting firewall errors.

Reporting attribute	Example value
use_case	troubleshoot;
granularity.network	{zone_or_group={SCADA}; traffic_direction={inbound};}
granularity.policy	{rule_or_group={rule1}; policy_action={permit};}
granularity.temporal	{near_realtime}
granularity.performance	{measurement={interface};}

8 Lessons Learned

There are several takeaways from our study:

1. A SCADA firewall should not cater for every use case. For some use cases, it is better to employ additional dedicated infrastructure to meet requirements, and allow firewalls to focus on their primary function: *traffic filtering*.
2. Firewall reporting should be configured at the right granularity for its use. Data that is collected but not used is just wasting resources.
3. Reporting and policy need to be coupled. Both are inter-dependent network functions and there is little sense in deploying one without the other.
4. Firewall vendors need to support standard firewall features to consistently map high-level reporting policy to firewall capabilities. These include: performance, operational and traffic measurements, and policy actions.

9 Conclusion and Future Work

The standards and best practices for reporting and analysis of firewall data lack clarity in what firewalls *should* report.

Our research utilises the use cases of firewall reports and specifies reporting in terms of scope and granularity. From this we identify reporting requirements with respect to several dimensions: time, network elements, policies, *etc*, and evaluate costs. We provide clarity on what a SCADA firewall *should* report, and demonstrate our high-level reporting implementation.

Acknowledgements. This project was supported by the Australian Government through an Australian Postgraduate Award, Australian Research Council Linkage Grant LP100200493, and CQR Consulting.

References

1. ANSI/ISA-62443-1-1. Security for industrial automation and control systems part 1-1: Terminology, concepts, and models, 2007.
2. Byres, E., Karsch, J., and Carter, J. NISCC good practice guide on firewall deployment for SCADA and process control networks. *National Infrastructure Security Co-ordination Centre*, 2005.
3. Check Point. *NGX R65 CC Evaluated Configuration User Guide*. Check Point, software technologies Ltd., USA, 2008.
4. Cisco Systems. *Cisco ASA 5500 Series Configuration Guide using the CLI*. Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706, USA, 2010.
5. Cisco Systems. ASA 8.3 and later: Monitor and troubleshoot performance issues. White paper, Cisco Systems, March 2014.
6. Cisco Systems. Cisco ASA 5585-X adaptive security appliance architecture. White paper, Cisco Systems, May 2014.
7. De Champeaux, D., Lea, D., and Faure, P. *Object-oriented system development*. Addison Wesley, Reading, MA, 1993.
8. Johnson, A., Dempsey, K., Ross, R., Gupta, S., and Bailey, D. Guide for security-focused configuration management of Information Systems. *NIST Special Publication*, 800(128):16–16, 2011.
9. Juniper Networks. *Firewall Filter and Policer Configuration Guide*. Juniper Networks, Inc., 1194 North Mathilda Avenue, Sunnyvale, California 94089, USA, 2011.
10. Kent, K. and Souppaya, M. Guide to computer security log management. *NIST Special Publication*, 800(92):16–16, 2006.
11. Mayer, A., Wool, A., and Ziskind, E. Fang: A firewall analysis engine. In *IEEE Symposium on Security and Privacy*, pages 177–187, 2000.
12. NERC. Cyber security- Incident reporting and response planning. *Critical Infrastructure Protection Standards*, 008(3), 2009.
13. NERC. Cyber security- Systems security management. *Critical Infrastructure Protection Standards*, 007(3a), 2013.
14. Oetiker, T. RRDtool. http://oss.oetiker.ch/rrdtool/.
15. Purdy, G. N. *Linux iptables pocket reference*. O'Reilly Media, Inc., 2004.
16. Ranathunga, D., Roughan, M., Kernick, P., Falkner, N., and Nguyen, H. Identifying the missing aspects of the ANSI/ISA best practices for security policy. In *Proceedings of CPSS*, pages 37–48. ACM, 2015.
17. Ranathunga, D., Roughan, M., Kernick, P., Falkner, N., and Tune, P. ForestFirewalls: Getting firewall configuration right in critical networks, http://tinyurl.com/pzqtzkm.
18. Rubin, A. and Geer, D. A survey of Web security. *Computer*, 31(9):34–41, 1998.
19. Scarfone, K. and Hoffman, P. Guidelines on firewalls and firewall policy. *NIST Special Publication*, 800(41), 2009.
20. Scarfone, K. and Mell, P. Guide to Intrusion Detection and Prevention Systems (IDPS). *NIST Special Publication*, 800(94):16–16, 2007.
21. Stouffer, K., Falco, J., and Scarfone, K. Guide to Industrial Control Systems (ICS) security. *NIST Special Publication*, 800(82):16–16, 2008.
22. Wool, A. Architecting the Lumeta firewall analyzer. In *USENIX Security Symposium*, pages 85–97, 2001.
23. Wool, A. A quantitative study of firewall configuration errors. *Computer, IEEE*, 37(6):62–67, 2004.
24. Wool, A. Trends in firewall configuration errors: Measuring the holes in Swiss cheese. *Internet Computing, IEEE*, 14(4):58–65, 2010.