Managing the Transition from SNMP to NETCONF: Comparing Dual-Stack and Protocol Gateway Hybrid Approaches

Ronald J. Brash

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science at

Concordia University

Montréal, Québec, Canada

April 2017

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ronald J. Brash**

Entitled: **Managing the Transition from SNMP to NETCONF: Comparing Dual-Stack and Protocol Gateway Hybrid Approaches**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr.

_____ Examiner
Dr. Jeremy Clark

_____ Examiner
Dr. Lata Narayanan

_____ Supervisor
Dr. J.W. Atwood

Approved by _____
Dr. Volker Haarslev, Graduate Program Director
Department of Computer Science and Software Engineering

_____ 2017

_____
Dr. Amir Asif, Dean
Faculty of Engineering and Computer Science

Abstract

Managing the Transition from SNMP to NETCONF: Comparing Dual-Stack and Protocol Gateway Hybrid Approaches

Ronald J. Brash

As industries become increasingly automated and stressed to seek business advantages, they often have operational constraints that make modernization and security more challenging. Constraints exist such as low operating budgets, long operational lifetimes and infeasible network/device upgrade/modification paths. In order to bypass these constraints with minimal risk of disruption and perform "no harm", network administrators have come to rely on using dual-stack approaches, which allow legacy protocols to co-exist with modern ones. For example, if SNMP is required for managing legacy devices, and a newer protocol (NETCONF) is required for modern devices, then administrators simply modify firewall Access Control Lists (ACLs) to allow passage of both protocols. In today's networks, firewalls are ubiquitous, relatively inexpensive, and able to support multiple protocols (hence dual-stack) while providing network security.

While investigating securing legacy devices in heterogeneous networks, it was determined that dual-stack firewall approaches do not provide adequate protection beyond layer three filtering of the IP stack. Therefore, the NETCONF/SNMP Protocol Gateway hybrid (NSPG) was developed as an alternative in environments where security is necessary, but legacy devices are infeasible to upgrade, replace, and modify. The NSPG allows network administrators to utilize only a single modern protocol (NETCONF) instead of both NETCONF and SNMP, and enforce additional security controls without modifying existing deployments. It has been demonstrated that legacy devices can be securely managed in a protocol-agnostic manner using low-cost commodity hardware (e.g., the RaspberryPi platform) with administrator-derived XML-based configuration policies.

Acknowledgments

Firstly, I would like to express my sincere gratitude to my supervisor Professor J. William Atwood for his generous and continuous support throughout the process to complete this research. Without his assistance, mentoring and weekly meetings – the schedule may have gone astray or run down a rabbit hole.

I would like to thank my family and friends for their support. My experience in Montreal has been an adventure and without their encouragement, it would not have been possible to complete this adventure. Also without the years of mentoring by colleague and friend Eric Byres, my outlook and skills associated with technology would have been vastly different. Profound thanks to you all, mentioned or unmentioned.

Contents

List of Figures

List of Tables

List of Acronyms

AAA Authentication, Authorization and Accounting

ACL Access Control List

AH Authentication Header

ARP Address Resolution Protocol

ASN.1 Abstract Syntax Notation One

BitS Bump in the Stack

BitW Bump in the Wire

BEEP Blocks Extensible Exchange Protocol

CA Certificate Authority

CIA Confidentiality, Integrity and Availability

CLI Command Line Interface

DDoS Distributed Denial of Service

DNS Domain Name System

DoS Denial of Service

DPI Deep Packet Inspection

DTD Document Type Definition

EoL End of Life

ESP Encapsulating Security Payload

FS File System

FTP File Transfer Protocol

GID Group Identifier

GRPC Google RPC Protocol

HTML Hyper Text Markup Language

HTTP Hyper Text Transfer Protocol

IANA Internet Assigned Numbers Authority

ICS Industrial Control System

ICMP Internet Control Message Protocol

IETF Internet Engineering Task Force

IDS Intrusion Detection System

IKE Internet Key Exchange protocol

IPS Intrusion Prevention System

I/O Input/Output

IoT Internet of Things

IPsec Internet Protocol Security

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISAKMP Internet Security Association and Key Management Protocol

IT Information Technology

LAN Local Area Network

L2TP Layer Two Tunnelling Protocol

MAC Message Authentication Code

MD5 Message Digest version 5

MIB Management Information Base

MitM Man in the Middle

MPLS Multi-Protocol Label Switching

MS Microsoft

MTBF Mean-Time-Between-Failures

NAT Network Address Translation

NETCONF Network Configuration Protocol

NMS Network Management System

NSPG NETCONF/YANG Protocol Gateway

OID Object Identifier

OS Operating System

PLC Programmable Logic Controller

PPTP Point to Point Tunnelling Protocol

RBAC Role-based Access Control

RFC Request For Comments

RPC Remote Procedure Call

SA Security Association

SDN Software Defined Network

SIGMP Secure Internet Group Management Protocol

SLA Service Level Agreement

SNMP Simple Network Management Protocol

SNMPv1 Simple Network Management Protocol version 1

SNMPv2 Simple Network Management Protocol version 2

SNMPv2c Simple Network Management Protocol version 2c

SNMPv3 Simple Network Management Protocol version 3

SOAP Simple Object Access Protocol

SSL Secure Sockets Layer

SSH Secure Shell

TLS Transport Layer Security

TCP User Datagram Protocol

TCP-AO TCP Authentication Option

TCP-MD5 TCP MD5 Option

UDP Transmission Control Protocol

UID User Identifier

URL Uniform Resource Locator

UUID Unique Uniform Identifier

VLAN Virtual Local Area Network

VM Virtual Machine

VPN Virtual Private Network

XML eXtensible Modeling Language

XSLT eXtensible Stylesheet Language Transformations

YANG Yet Another Next Generation

YIN YANG Independent Notation

Chapter 1

Introduction

Existing computer networks are predominantly managed by network protocols such as Simple Network Management Protocol (SNMP) and human command line interfaces (CLIs). After many years of promoting SNMP to standardize network management, SNMP functionality is increasingly deployed on a multitude of network appliances as an approach to configuring and monitoring network devices. While SNMP has been heavily utilized to monitor devices, it is seen to be ineffective as a tool for device configuration, primarily because support has not been provided (by device manufacturers) for configuring the entirety of features supported by devices through the SNMP interface. Recently, there has been movement within the Internet Engineering Task Force (IETF) to migrate from SNMP to Network Configuration protocol (NETCONF) and Yet Another Next Generation (YANG) data models, due to a number of benefits such as defined transaction-safe configuration of devices, increased security, extensibility through standardized YANG modules and usage within Software Defined Networks (SDNs) (Cisco Systems Inc., 2014).

From a network operator's perspective, the above described benefits over currently employed methods are sufficient reason to use NETCONF in place of SNMP. As a result, there is considerable activity within the IETF to develop YANG modules for a wide range of network devices, and to coordinate development activity carefully to ensure consistency in the approach used. This also takes advantage of the inherent extensibility of YANG to permit controlled, manufacturer-specific extensions to match device- or manufacturer-specific features.

However, a number of issues surrounding NETCONF adoption remain unexplored. For example, how could end-users and network administrators deploy NETCONF solutions with minimal impact and/or additional risk or complexity? Alternatively, how can network device manufacturers migrate to NETCONF transparently while supporting both protocol stacks and customers with legacy SNMP managed networks? Is the security surface altered during the technology transition or are there ways to leverage the advantages of newer protocols in a way that is beneficial for legacy devices? The above questions are to be examined and a generic proposed solution is to be presented for NETCONF/SNMP, but the solution could also be utilized for HTTP and other legacy protocols.

Clearly, any new approach to network management must integrate with existing approaches, but it should not worsen security vulnerability surfaces or management complexity. Since the semantics of NETCONF/YANG are aligned more intimately to the needs of the operators, it is reasonable to assume that the existing (SNMP-based and CLI-based) approaches can be incorporated as "base elements" in a NETCONF/YANG framework. However, regardless of approach, support for NETCONF/YANG requires that NETCONF/YANG agents be hosted on a device, which requires vendor support, and/or facilities for third parties and customers to install their own software. If a vendor does not provide software updates or customers choose to not update deployed devices, transitional approaches should be carefully examined for security so that new attack vectors are not created and legacy ones forgotten.

There are many approaches to integrating legacy systems or software into more modern designs, but two common techniques are through the use of middleware as "glue" in heterogeneous environments or to use a "dual-stack" approach, which allows legacy protocol stacks to be used simultaneously without causing any disruptions for network operators. Using the "dual-stack" approach as a baseline for evaluating the proposed solution, a middleware solution will be developed for bi-directional translation between NETCONF/YANG and SNMP at the network layer instead of the application layer. In the past, a gateway device or protocol converter such as a serial Modbus terminal would have been used in a similar way, effectively converting the Modbus serial protocol onto a TCP/IP and Ethernet stack (Modbus Organization, 2015). However, the proposed solution for SNMP to NETCONF conversion is inherently more complex than simply adding or removing layers within a stack because conversion is within the same TCP/IP stack and between vastly different

protocol architectures.

The thesis is structured as follows:

- **Chapter 2 - Background Study**

  This chapter will provide the basic concepts to understand our proposed model and analysis.

- **Chapter 3 - Problem Statement**

  This chapter will describe and define the problem to be explored by this research.

- **Chapter 4 - Proposed Solution**

  This chapter will propose a solution that addresses the problem(s) identified within the previous chapter's problem statement.

- **Chapter 5 - Testing and Validation**

  This chapter will describe the use of the NETCONF/SNMP test-bed system to verify the proposed solution when compared to a traditional "dual-stack" approach and examine whether security can be moved closer to legacy end devices.

- **Chapter 6 - Conclusion and Future Work**

  This chapter will review and conclude research results and introduce possible avenues for future work.

Chapter 2

Background Study

In this chapter, the history around network management, network security, underlying protocols and related concepts including challenges, will be reviewed to provide context describing the environment surrounding the proposed solution. These basic concepts, software and architectures will be explored formally throughout this document to inform the reader about the following:

- Define network management and how it is achieved from general observation
- Define security from network management perspectives and realistic usability
- Define several tools used by operators, which may or may not fullfill operator needs

2.1 Network Management and Security

2.2 Network Management

Eloquently stated by Clark et al. “From the beginning, the shape and nature of network management was not clear in the early design of the Internet and whether human operators would or should be involved in network management and automated if possible” (Clark, 2009).

While the first Internet (the ARPANET) was a United States government sponsored project, ARPANET sites such as universities began to multiply with the result becoming a network of networks and not just research experiments (DARPA, 2008). Unexpectedly, the first major network

failure occurred on October 27, 1980. It caused the ARPANET to be unavailable for several hours. The unavailability was non-malicious, caused by a faulty network adapter, which created several malformed packets that exploited an unknown vulnerability or flaw that then consumed network resources until they were completely exhausted (Beranek & Rosen, 1981). As part of the process to resolve the crash, researchers and administrators of the ARPANET network had to work with minimal tools and diagnostics other than manually calling operators at ARPANET sites. Ultimately, the ARPANET community realized that better network management, disaster recovery and reconstruction of an event was highly desirable (Beranek & Rosen, 1981). This was also seconded by an adhoc group called the Internet Architecture Board (IAB), which recommended that Simple Gateway Management Protocol (SGMP) research be aimed at practical management tools to reduce support costs, improve remote management functionality and simplify management operations (*Simple Gateway Monitoring Protocol*, 1987). As time passed, hosts became more ubiquitous and SGMP eventually became the SNMP protocol, which was heralded as the de-facto configuration and management protocol (Cerf, 1988).

Fast forwarding to today, networks have become an integral part of communications, daily operations and business. Network maintenance and healthy network operation have become very important aspects of both research and business operations. For individuals managing and maintaining networks, a human is different from a machine because it has finite resources to expend during daily activities. This is unlike a computer, which can complete remedial tasks endlessly and potentially without error; these computer-performed tasks can be used to assist human network operators to improve efficiency in daily routines and improve monitoring network health (Van-Echtelt, Wynstra, & Van-Weele, 2007)(Wang, Wang, Yu, & Dong, 2010). Additionally, there are advantages for higher levels of management within organizations as it can also be used for an organization's business strategy and operational procedures.

Imagine the following scenario: Bob the network administrator manages a small network consisting of several office machines, switches and routers located at a single small office location. His tasks vary from handling user issues with Operating Systems (OSs), malware removal, network troubleshooting and administration. Sometimes he configures a router or machine upon failure or upgrade, but often his daily duties consist of user support or troubleshooting. If Bob can automate

any of his daily duties and improve his efficiency in monitoring network resources or solving issues more quickly, any improvement is likely desired personally and/or by management.

In continuation with the scenario, imagine Bob's employer has grown beyond a single central office and now has several satellite offices. There is a limited corporate budget, and only Bob accompanied by a couple of technicians manage tens of systems or devices. Now further imagine that Bob's manager suddenly inquires about poor network performance in one of the satellite offices, if Bob has deployed all of the security fixes for a vulnerability he read about in the media, or asks Bob to fix something where there is no technician in close proximity? These are common and reasonable requests any member of management may ask, but they are also valid reasons for efficient network management and security. The strategy Bob would likely use to help him deal with activities or be aware of conditions his manager may ask about is through using network management strategies and software designed to centralize management. This is often the central focal point where administrative tasks are performed and are communicated to managed end devices for purposes such as configuration, management and monitoring.

Beyond the practical scenarios of what network management consists of, formally it is defined by (Saydam & Magedanz, 2012) as:

"Network management includes the deployment, integration, and coordination of the hardware, software, and human elements to monitor, test, poll, configure, analyze, evaluate, and control the network and element resources to meet the real-time, operational performance, and Quality of Service requirements at a reasonable cost."

The above suggests that network management requires infrastructure for network management and management frameworks or protocols (Kurose & Ross, 2012). This section of background study relates to the protocols used for network management, specifically SNMP and NETCONF/YANG.

Additionally, we present how to manage and transition between legacy devices when used alongside more modern devices. Despite the absence of security in the definition of network management, security is an important aspect of network management and should be evaluated as an important factor when defining, choosing or implementing a transitional approach to network technologies. However, the term security is loosely defined. It needs further definition to clearly understand and articulate security impact on networks and technology.

2.2.1 Security

Security or computer security is generally defined by three properties often applied to a single computer, network or service. They are: Confidentiality, Integrity and Availability (CIA) (Leight & Hammer, 2006)(Gasser, 1988). Confidentiality often refers to keeping content within a stream of data secure by maintaining secrecy against attacks such as eavesdropping or "sniffing". Integrity implies that data transmitted or stored is exactly as intended without modification, tampering or technological error, which is useful against attacks or vulnerabilities where packets are intercepted and modified. Availability is the state when information or a system is reliably accessible as intended or otherwise made unavailable through Denial of Service (DoS) attacks or errors (Rescorla & Korver, 2003).

There are additional attacks categorized as peer entity authentication, non-repudiation, systems security, unauthorized usage and inappropriate usage. For this research, it is important to realize that the Internet environment should be considered to be insecure due to implementation and assumptions about malicious parties potentially having some level of control over communication channels (Perlman, 2004).

For example, Bob wants to send Sue a secret message over a network, malicious Malory is passively listening or "sniffing" over an insecure channel and intercepts Bob's message to Sue without either party knowing or manipulating the network. Upon capture of Bob's message Malory can read the contents directed to Sue in clear-text if not protected adequately through confidentiality via encryption. This insecure channel allows Malory to access Bob's message to Sue potentially through connecting to a spanning port on a network device, insecure network media, hardware taps, Ethernet hub or through compromised/insecure wireless networks.

Alternatively, Malory may use an active approach using a Man-in-the-Middle (MitM) attack and a technique called Address Resolution Protocol (ARP) poisoning, which tricks hosts on the local area network (LAN) to believe that Malory is the intended destination host. The tricked hosts then unknowingly transmit data to Malory who receives it, and Malory may optionally retransmit intercepted data to the original intended host as if nothing had happened. Hence Malory is designated

as a MitM who also has the ability to modify, delete or insert arbitrary messages if sufficient protections such as encryption, integrity checking and peer-verification are not present in the original connection from Bob.

Active attacks are enabled by security being an afterthought to IPv4 and lower level protocols such as Ethernet. The lack of protections allow malicious users such as Malory to "spoof" IP addresses when host verification is not present through cryptographic certificates/keys, read clear-text implementations without data or protocol security (IPv4, TCP and UDP) and alter data without cryptographic primitives such as hashes or digital signatures to verify data is as intended (Rescorla & Korver, 2003).

To improve the security of communications between Bob and Sue, there should be peer verification to prove that Bob or Sue are who they say they are and encryption to protect the content of Bob's message. This way Sue can verify that the message she receives is actually from Bob and contains exactly the content he intended for it to carry. For Malory to be unable to read the contents, the encryption mechanism must be computationally difficult for Malory to decrypt through "brute-force" by using a strong cryptographic primitives. The message should also contain a time-stamp to prevent Malory from resending the message through a "replay" attack or an integrity check to prevent Malory from modifying a valid message through "length extension" attacks (Rescorla & Korver, 2003).

In summary, network security implies that data or information is transmitted from one host to another without being tampered with, read only by authorized parties using user credentials or host authentication and reliably through an available channel.

2.2.2 Underlying Network Protocols

For the purposes of this section, a protocol is defined as a set of procedures and syntax that describe the data communication between two hosts similarly to human social communication for hello and good-bye (M. Rouse, 2012b). Additionally, a packet refers to the envelope and format of data being transmitted over a network either singularly or as a group of packets that create a segment of data (M. Rouse, 2012c). Collectively, these protocols and associated packets are used for management, security and other network tasks.

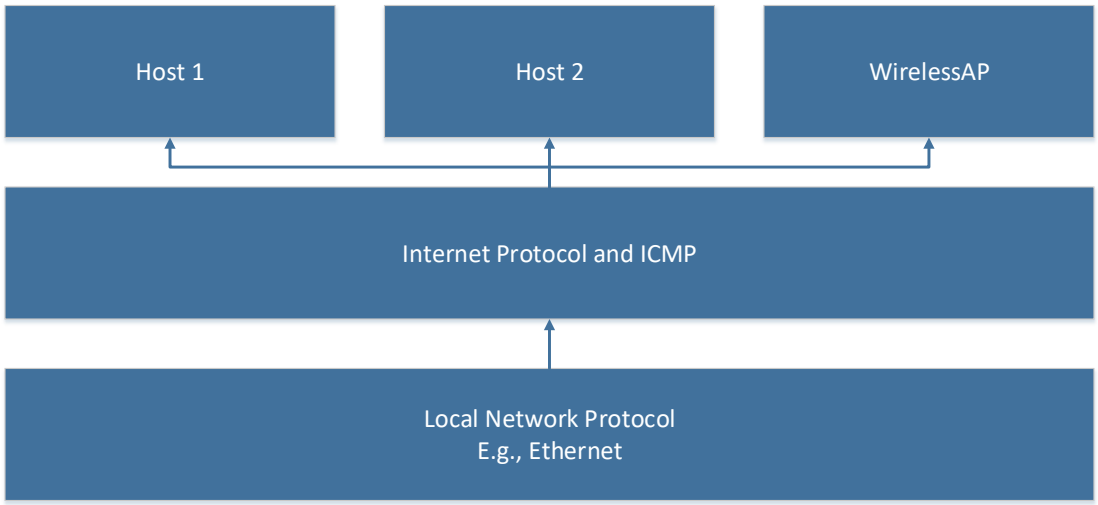

Figure 2.1: IP layer diagram

In addition, this section describes two highly used protocols and their parent, which belong to the Transmission Control Protocol (TCP)/ Internet Protocol (IP) suite. To reduce the scope of this research, sibling (Internet Communication Message Protocol, etc) and lower layer protocols (Ethernet, etc.) will be ignored for clarity, however, a brief introduction to the Internet Protocol will be provided in the following paragraphs. For information about ICMP and Ethernet see (*Internet Control Message Protocol*, 1981)(Intel Corporation and Xerox Corporation, 1980).

Internet Protocol

The Internet Protocol (IP) was designed to interconnect systems of packet-switched computer networks such that data could be transmitted to hosts identified by fixed length addresses (*DoD standard Internet Protocol*, 1980)(*Internet Protocol*, 1981). As a result, IP provides functionality such as fragmentation and reassembly of long data segments through a "best-effort" strategy with control, flow, and sequencing being provided by protocols layered upon the IP layer.

It is important to note that the IP is inherently vulnerable to ARP poisoning, eavesdropping or sniffing, MitM attacks, and IP address spoofing (Ferguson, 2000). This also means that protocols layered upon IP are also inherently vulnerable unless measures such as host authentication, encryption and integrity checking are available (Touch, 2007)(Eggert & Fairhurst, 2008).

User Datagram Protocol

UDP is a connection-less protocol, which provides lower protocol overhead and best-effort delivery for applications that internally manage data loss, duplication and transactions themselves. It is a bare-minimum protocol without congestion control, flow control, hand-shaking, or state information (*User Datagram Protocol*, 1980). Protocols and applications utilizing UDP often provide the aforementioned features (if needed) within the application's architecture similarly to the Domain Name System (DNS) protocol implementation (Patton, S.Bradner, Elz, & Bush, 1997)(Mockapetris, 1987).

UDP does not provide security mechanisms within its implementation or through an extension similar to TCP-MD5 Signature Option (TCP-MD5) and TCP Authentication Option (TCP-AO) (Heffernan, 1998)(Touch, Bonica, & Mankin, 2010). However, it has additional security concerns because it does not utilize congestion control and has limited state information for network firewalls (Eggert & Fairhurst, 2008). Security features such as encryption and integrity checking are absent and need to be added through through protocols using those identified in Section 2.2.4 or within application-level protocols.

Transmission Control Protocol

TCP is a connection-oriented protocol, which provides host-to-host communication at the Transport layer above the IP protocol. Through a three-way handshake, segmented data-streams, retransmission, error detection, congestion control and flow control, TCP can provide reliable delivery compared to User Datagram Protocol (UDP)(*Transmission Control Protocol*, 1981)(Hedrick, 1988). In particular, one feature of TCP is the use of the three-way handshake to determine a connection's state for both hosts directly involved in the connection and hosts or devices in which the connection passes through. This provides a small level of security allowing connection tracking facilities to discern through state if a connection is being spoofed to a certain degree, if the initial connection setup is non-complete, out-of-order, established and whether a connection is closed.

The initial specifications of TCP lacked encryption and integrity checking and as a result, TCP-MD5 and TCP-AO were added (Heffernan, 1998)(Heffernan, 1998). Initially, TCP-MD5 is used

to provide protection against spoofed segments being entered into a stream in use-cases such as those used by Border Gateway Protocol (BGP) (Heffernan, 1998). This was later extended with TCP-AO being used to provide integrity checking, replay protection and authentication (Touch et al., 2010). However, to overcome missing encryption confidentiality features, encryption protocols such as those identified in Section 2.2.4 are used to enhance TCP security.

2.2.3 Network Management Protocols

This section describes two types of network management protocols and a third category, which is utilized as Command Line Interfaces (CLIs) or shells on devices for manual configuration.

SNMP

Circa February 1988, a committee was formed to review a number of standards revolving around network management options for both the Internet and TCP/IP protocol suite. The review resulted in a number of suggestions, which were forwarded to the Internet Activities Board (IAB) including the recommendation for the Internet community to adopt the Simple Network Protocol (SNMP) (Cerf, 1988). However, the committee also recommended that the Internet Engineering Task Force (IETF) also be responsible for future SNMP efforts to reduce or eliminate vendor-specific bias or control over the protocol itself.

Eventually through many Request for Comments (RFCs) and three major protocol revisions, SNMP developed into the Internet-standard for managing networked devices (Mauro & Schmid, 2001). By itself, SNMP represents several components, which describe network management, an application layer network protocol, a data-base schema and a descriptive list of data objects through a Management Information Base (MIB)(Harrington, Wijnen, & Presuhn, 2002).

A MIB can be described as a virtual data-store that contains a number of objects defined in the format of Abstract Syntax Notation One (ASN.1). It is typically created by device vendors. This syntax, according to the Structure of Management Information (SMI), allows the object to be understood by explicitly stating object identifying names and syntax describing the data type and data encoding (IETF, 1990b). The resulting MIB then describes the SNMP controlled host as an SNMP entity that can be accessed by management software (McCloghrie, Schönwälder, Perkins, &

McCloghrie, 1999)(Presuhn, 2002).

As SNMP adoption continued, the protocol itself evolved to include a number of features that were not present in its initial conception. SNMPv1 provided basic features such as the ability to get or set objects and message traps, but suffered from weak security, lack of delivery guarantees such as notifications, and data-types of non-sufficient size (IETF, 1990a). It would be expected that the following edition of SNMP would have focused on the shortfalls of the previous version. However, SNMPv2 became a group of flavors with the final candidate having reached consensus without security despite security features being contained within sibling candidates. The final version of SNMPv2 contained enhancements for data types, efficient bulk operations, notification and error handling and tuning of the notification language (Mundy, Partain, & Stewart, 2002). Eventually, the SNMPv3 working group aimed to correct the inconsistencies across the SNMPv2 flavors in areas of authentication, privacy, authorization, access control and standards-based remote configuration.

NETCONF/YANG

Despite intentions of the earlier IETF's RFC1052 and wide-spread adoption of SNMP, device configuration was still dominated by proprietary protocols, which included command line interfaces (CLIs). As the trend of networked devices increased, issues of maintenance costs and reduced scalability of management also began to arise. To combat these effects, the IETF began to develop and standardize network configuration through the Network Configuration protocol (NETCONF). In 2006, the NETCONF protocol was published to standardize several features, which were considered missing from common network management protocols such as SNMP (Enns, 2006).

Effectively, NETCONF is a remote-procedure call (RPC) based protocol, which through extensive use of XML (eXtensible Markup Language) allows device configuration and programmability to be exposed as an application programming interface (API). NETCONF is then used to facilitate a logical connection-oriented Transmission Control Protocol (TCP) session between the client and server, where the client is either the "administrator" or "network management software" (NMS) and the server is the "device". A device must allow at least one NETCONF session and utilizes a four layer partition scheme to partition protocol operations and data: layer one is the transport mechanism, which can be based on Blocks Extensible Exchange Protocol (BEEP), Simple Object

Access Protocol (SOAP), Secure Shell (SSH), Secure Socket Layer (SSL) or CLI, layer two is the RPC layer or notification layer, layer three is for operations such as "get" or "edit config" and the remaining content layer contains configuration or notification data as noted in (Enns, 2006)(Wasserman & Goddard, 2006)(Goddard, 2006)(Lear & Crozier, 2006).

Through the multi-layer scheme, NETCONF provides facilities for network system operators and Software Defined Networks (SDNs), which are different from those offered by SNMP for configuration. Many of the features that differentiate NETCONF from SNMP are optionally implemented by vendors. These include capabilities exchange, writing while running, candidate configuration, confirmed commit, rollback on error, configuration validation, separate start-up and running configuration data stores, Uniform Resource Locators (URL) elements, support for XPath expressions, support for security and Internet Assigned Numbers Authority (IANA) support (Enns, Bjorklund, Bierman, & Schonwalder, 2011). It is through many of these features and improved security that NETCONF is desirable over SNMP (Santos, Esteves, & Granville, 2015).

On a NETCONF enabled device, the agent handles incoming requests and sessions according to YANG modules (Bjorklund, 2010). YANG is an extensible data modeling language, which is used by the agent to understand NETCONF-based operations, configuration data, RPCs and notifications, but can also describe data constraints to be enforced on data. YANG can be translated into an XML equivalent for manipulation and back again using YANG Independent Notation (YIN). This is useful for applications that utilize XML parsers and eXtensible Stylesheet Language Transformations (XSLT) to manipulate YANG models. In addition, SNMP SMIv2-based MIB modules can be translated into YANG for read-only access, but not in the reverse direction for YANG to SMIv2 (Bjorklund, 2010).

CLI-based

Despite protocols such as SNMP, many network devices implement CLI-based protocols for configuration such as the legacy, but insecure, Telnet or its modern sibling Secure Shell (SSH) protocol. Often CLIs consist of a user interface technique that utilizes manual text input being transmitted to the end host and characters being displayed on-screen when transmitted back. Without

```
bob@linuxbox:~$ sudo ifconfig wlan0 192.168.15.200/24
bob@linuxbox:~$ sudo ifconfig wlan0
wlan0     Link encap:Ethernet  HWaddr 3c:77:e6:92:3c:a7
          inet addr:192.168.15.200  Bcast:192.168.15.255
          Mask:255.255.255.0
          inet6 addr: fe80::3e77:e6ff:fe92:3ca7/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:69427 errors:0 dropped:0 overruns:0
          frame:4801625 TX packets:53559 errors:0 dropped:0
          overruns:0 carrier:0 collisions:0 txqueuelen:1000
          RX bytes:62410452 (62.4 MB)  TX bytes:10557015 (10.5 MB)
          Interrupt:17

bob@linuxbox:~$ route -n
Kernel IP routing table
Destination   Gateway Genmask        Flags Metric Ref  Use Iface
192.168.15.0  0.0.0.0 255.255.255.0  U     0      0      0 wlan0
```

Figure 2.2: Ifconfig command output

delving into the history of CLI or console protocols, they are advantageous in several ways for troubleshooting and configuration by human operators, because they are often operator task oriented, feature simple interface design, have context sensitive help, can be scripted, and often do not require custom software other than a terminal application. Unfortunately, CLI-based protocols have several downsides in order to achieve the flexibility of their advantages. Furthermore, in heterogeneous network environments featuring different devices and vendors, CLIs may cause unexpected problems when parsing proprietary syntax, non-standard data models, and manual input (Schonwalder, 2003).

To illustrate the issue of non-standardized network configuration protocols or interfaces, imagine Bob the network technician who on a Linux machine ran the configuration commands (in Figure 4.3) through an SSH session to change the IP address of a wireless interface named wlan0 to 192.168.15.200/24 and check the routing table.

The example, listed verbatim, demonstrates utility for human operators using a standard interface common across many Linux-based OSs. However, the reality of text-based interfaces is that when a single change occurs in a deployed environment, errors in any automation facilities such as scripts may cause failures. In addition, despite the common availability of CLIs, the human operator still retains human flaws such as limited working hours, susceptibility for typographic errors and

potentially lack of attention span.

The above accepts the disadvantages of CLI protocols in certain environments, but also acknowledges their utility for operators for specific tasks. It is the need for standardized network configuration protocols in enterprise or large deployments that necessitates protocols such as SNMP and NETCONF/YANG (Jones, 2004).

Miscellaneous Protocols

For completion of the topic, it is important to note that with the rise of Internet of Things (IoT), new protocols such as Google's RPC (GRPC) protocol have been open-sourced and are making appearances in IETF Internet drafts (Google, 2016). These types of protocols are aimed at providing functionality such as the configuration of devices, monitoring devices including sensor telemetry, high performance access, secure communications and service handling.

In low-cost resource constrained embedded systems, performance is often restricted, but actual features are not, which represents several drawbacks. It is from these drawbacks that often SNMP is not seen as a complete solution and that allowed protocols such as GRPC to be created (Google, 2016).

2.2.4 Security Protocols

In continuation with network protocols, this section explores three common protocols that are used to provide functionality such as encryption, frameworks for key exchange, integrity checking, host verification and protocol tunnelling or VPNs.

IP Security Protocol

The IP Security Protocol (IPsec) operates by providing network security at the network layer of the TCP/IP model and "is designed to provide inter-operable, high-quality, cryptographically-based security for IPv4 and IPv6" (Atkinson & Kent, 1998). This equates to security being provided at the IP layer which in turn, secures any higher layer protocol such as TCP, UDP, ICMP and Border Gateway Protocol (BGP).

Effectively, IPsec provides confidentiality, integrity checking and host verification through two modes employing vastly different strategies: tunnel mode and transport mode. The first completely encapsulates the original packets inside an IPsec packet, and the second encapsulates the data that follows the IP header with a few minor changes (Atkinson & Kent, 1998). Regardless of mode, IPsec can be configured to use either or both Encapsulating Security Payload (ESP) and Authentication Header (AH) security protocols (Kurose & Ross, 2012).

IPsec also provides key exchange protocols such as Internet Key Exchange (IKE) and Internet Security Association and Key Management Protocol (ISAKMP)/Oakley. IKE is used to generate cryptography "keys" for IPsec protocols, and can also be used for key negotiation for other protocols that require keys as well (Dhall, Dhall, Batra, & Rani, 2012). ISAKMP defines the mechanics and packet formats to communicate Security Associations (SA), which provide all of the necessary information for executing IPsec security protocols and SAs are also used by IKE (Kaufman, 2005).

Real-world IPsec implementations commonly use three main techniques: integration of IPsec into the native IP implementation or kernel in the case of Linux, middleware between the IP stack and network media drivers, and through specialized hardware (Atkinson & Kent, 1998). From exploring the literature, implementing IPsec everywhere is not feasible when complete ownership of devices is not possible such as within deployed legacy proprietary devices or sensitive critical infrastructure.

IPsec is included in this chapter as an acknowledgement of IPsec's role when securing network connections through VPNs; a form of security and networking used to secure insecure networks or provide additional security where needed.

SSL/TLS

Secure Sockets Layer (SSL) and Transport Layer Security (TLS) are two different protocols that provide secure communications by utilizing cryptography. SSL was an earlier and popular protocol by hosts connected to the Internet, created by Netscape in 1996 for encrypted communications, certificate identification and message integrity checking (Freier, Karlton, & Kocher, 2011). Built upon the transport layer through TCP or UDP, SSL can be used transparently regardless of the application layer protocol utilizing it and in many cases, SSL is used for encapsulation of these

higher layer protocols.

For example, HTTP utilizes TCP for its transport layer protocol, but HTTPS or secure HTTP utilizes TCP whose payload is HTTP encapsulated by SSL. SSL is also used to provide handshake capabilities for client/server authentication and negotiation of cryptographic properties such as algorithms and keys (Freier et al., 2011).

Unfortunately, version 3.0 of SSL suffered a number of vulnerabilities and released exploits against supported encryption routines and the key exchange and negotiation.

In 1999, the IETF created a similar offering called TLS to standardize SSL connections for the Internet community. Although SSL 3.0 was the current release at the time and designed to be extensible, it became the basis for TLS despite SSL being published in several Internet drafts (Freier et al., 2011). Similar to SSL, TLS is designed to provide security between two communicating host applications. This is performed through two layers of the TLS record protocol and the TLS handshake protocol (Dierks, 2008). The TLS record protocol provides the same level of data encapsulation as SSL for higher level applications and the TLS Handshake protocol provides seamless cryptographic parameter exchanges. Generally SSL-based applications may never realize that communications were performed over a TLS connection and between differently configured hosts.

In 2015, TLS was released and experts recommended that any version of TLS including version 1.0 be considered more secure than SSL and should be used in preference to SSL 3.0. It was also reported that the SSL record layer was broken, encryption ciphers suffered critical weaknesses, key-exchange was vulnerable to MitM attacks, weak hash functions and non-sufficient security capabilities (Barnes, Thomson, Pironti, & Langley, 2015).

TLS also provides security through host identity authentication, and secure shared secret negotiations to prevent disclosure of information while in transit. Additionally, any modifications to cryptographic parameter negotiations are detected by both parties within the authorized communication channel (Dierks, 2008). Without outlining specific differences, TLS v1.2 is recommended for secure communications and as such, it is incorporated in experimental implementations of NETCONF on top of TLS.

SSH

Secure Shell (SSH) is a network protocol that was designed originally for secure remote connections, secure shell operations and forwarding TCP/IP connections through a tunnel. It was created in roughly the same era as SSL, however, it is different from SSL/TLS in that there are IETF RFC's that outline three components: User Authentication protocol and a Connection protocol in addition to the Transport Layer protocol, which is present in SSL/TLS (Lonvick & Ylonen, 2006b)(Lonvick & Ylonen, 2006a)(Lonvick & Ylonen, 2006c).

SSH uses cryptographic primitives and provides confidentiality and peer verification, but has two different trust models. The first is where the client has a local database of keys that are correlated to names. The second is the use of a certified and trusted certification authority (CA), which reduces some of the overhead of PKI on the client (Lonvick & Ylonen, 2006a).

2.2.5 Network Tools

For network administrators and technicians, using tools to improve security, management, visibility and daily operations is critical to a healthy network. This section briefly describes several tools and/or techniques often used to provide and improve security for computer networks; each of these are vast topics and often supported by research, RFCs and best practices.

Virtual Local Area Networks

Virtual Local Area Networks (VLANs) or 802.1Q are an IEEE standardized technique used to segment LANs further than using purely IP sub-netting and routing local LANs. VLANs are implemented using VLAN capable switches and offer better utilization of broadcast domains, IP address allocations, port aggregation and flexibility at a per-switch-port level (Dykes & McPherson, 2001).

In essence, a VLAN operates by adding a "shim" into the Ethernet frame at layer 2 designating a VLAN frame and a VLAN identifier, which designates it being one of 4096 different VLANs. Reversibly, VLAN "shims" can be removed when the VLAN is not required or relevant. This allows additional granularity for administrators if proper configurations are implemented (IEEE, 2015).

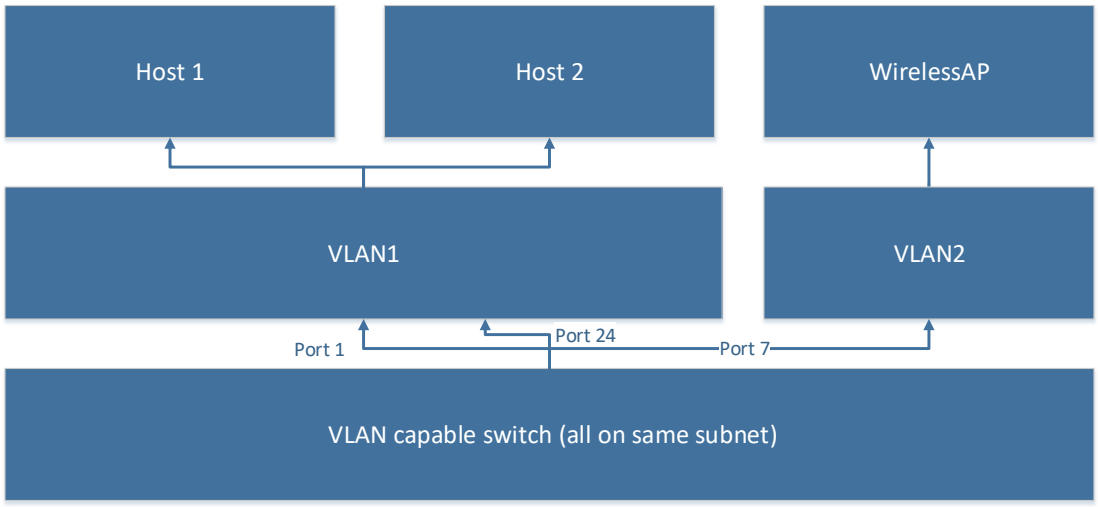

Figure 2.3: VLAN diagram

Configuration of VLANs can also provide improved security by preventing unicast connections between hosts by segmenting specific users away from each other and using ARP proxies (Foschiano & HomChaudhuri, 2010).

Firewalls

The term "firewall" refers to a technology or a device that has ingress and egress IP packet filtering capabilities, to be used at edge locations connected to the Internet (Smith, McCloghrie, Rijhsinghani, & Langille, 1999). Originally, firewalls used only "static packet filtering", which leveraged only the IP addresses of source and destination nodes combined with port and protocol pairings to be used as rules within Access Control Lists (ACLs). Through an administrative interface, network administrators can insert, update and delete ACL rules as needed for hosts or networks needing access.

Using only static packet filtering is insufficient. Attackers can "spoof" or inject packets appearing to be from different hosts to bypass ACLs being used by the firewalls protecting a network. As a result, new techniques such as state-tracking, SYN-cookies, timeouts, Deep Packet Inspection (DPI) applications and Intrusion Detection/Prevention Systems (IDS/IPS respectively) are used today as features often intermingled with firewall technology or devices (Bradner, 1991)(Smith et al.,

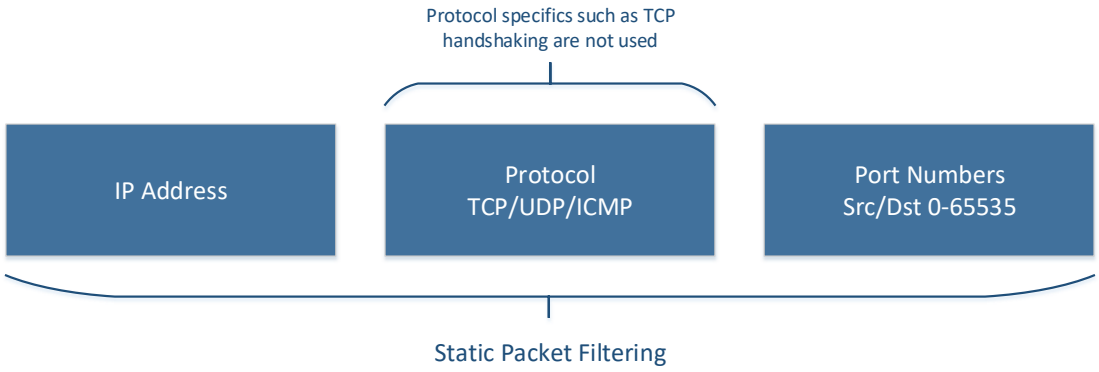

Figure 2.4: Static packet filtering diagram

1999)(Ferguson, 2000).

In addition to spoofing, to prevent Denial of Service (DoS) attacks, connections are tracked using state and connection monitoring tables in addition to other TCP SYN flooding mitigations (Eddy, 2007). With more intelligent tracking of TCP connections, firewall resource consumption when tracking connections is reduced and explicit firewall rules are minimized. Using an active FTP connection as an example, if a connection is initiated, it is now tracked with a SYN cookie. If the connection is established, the cookie is valid and if the client was connecting to TCP port 21 (FTP's control port), it should upon response to the client requesting a FTP connection open TCP port 20 to transfer data (*File Transfer Protocol*, 1985).

The FTP example describes a situation when firewall/filtering mechanisms are unaware of a second connection being established ahead of time or need technologies that are aware of the application protocol being transmitted in order to provide adequate security instead of creating intentional security risks through broad ACL rules. This is important when attempting to secure UDP-based connection ingress/egress points because the UDP protocol itself does not have state, but there may be state within the application level itself. The solution is to perform application layer filtering through protocol aware DPI, which is combined transparently with state-full packet filtering for increased security.

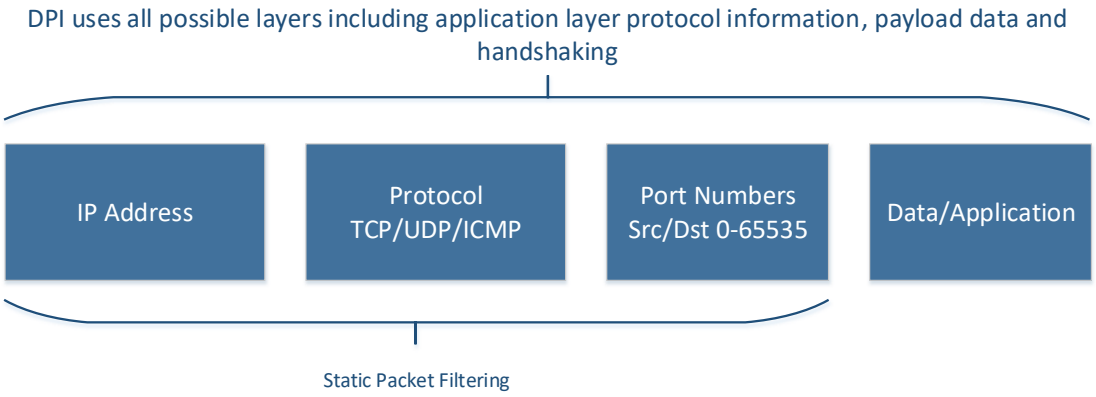

Figure 2.5: DPI vs static packet filtering diagram

Deep Packet Inspection

Deep Packet Inspection (DPI) is a technique used to inspect the contents at the application layer within a packet being transmitted. This has great advantages for network providers and administrators for management and enforcement activities, but can also negatively affect privacy (Office of the Privacy Commissioner of Canada, 2013).

From Figure 2.5, it can be observed that DPI is another layer of inspection, validation and verification being placed upon the earlier layers in the packet. This allows an application using DPI to determine information such as:

- Determining connection states that include requests and responses
- Examining data within payload fields
- Sanity-checking of the protocol as it is transmitted
- Detecting malicious data and unauthorized behaviors

While DPI is not limited to performing tasks similar to a "security guard", its effectiveness is limited by its knowledge of the protocol it is inspecting; given the level of protocol coverage including edge-cases implemented by the DPI developers. Using the example of a DPI monitoring HTTP 1.0 connection streams, this application could perform the following information:

- Determine if a response for HTTP 1.1 or HTTP 1.0?
- Whether to allow or deny commands or requests such as GET or PUT?
- Whether to block specific responses for the issued request?
- Whether to allow or deny in-line or pipelined data?
- Whether to deny clear-text information such as user credentials, and browser/web-server parameters?
- Whether to detect or block malicious exploits?
- Whether to enforce content and ACL restrictions?

For network administrators, DPI can provide security and functionality that may be difficult or unsupported for existing deployed infrastructure. *When DPI is used in an application context*, it can be used for detecting platform or protocol specific exploits, replacing user content, validating protocol fields, correcting information leaks within participants transmitting, maintaining quality of service, and adding more granular permissions (Smallwood & Vance, 2011)(Chaudhary & Sardana, 2011)(Li, Mao, & Rexford, 2012). For these reasons, DPI is explored in Section 2.2.5 as a component of Intrusion-Detection/Prevention Systems.

DPI does have some weaknesses and associated risks. Applications using DPI often require additional system resources to operate (when compared to a firewall), may introduce latency, may have inherent protocol problems such as overlapping segments and the potential for an attacker to attack the host system (Bellovin et al., 2008)(Chaudhary & Sardana, 2011). It also has to be deployed using a method that routes all packets to be visible by the DPI application and/or be in-line such that packets will travel through it in a Bump-in-the-Wire approach.

Intrusion Detection/Prevention Systems

Similar to firewalls, the term Intrusion Detection/Prevention System (IDS and IPS respectively) can refer to a software or to a device. Generally, there are two kinds, IDS and IPS. The first type, IDS are used passively to purely monitor and report events as they happen according to the detection

paradigm used. The second type, IPS can monitor network activity passively, but are differentiated by being able to actively perform actions upon detecting an event (Douligeris & Serpanos, 2007). Both utilize strategies to view network traffic as it passes across the network often at the application layer with specialized applications. However, IPS employ strategies to take active actions.

IDS detection paradigms often monitor network traffic through signature detection or traffic usage metrics, but can also use statistical techniques to detect anomaly events (Zhang, Li, & Zheng, 2004). The main objective of an IDS is to only monitor attacks, but it can generate misinformation for human operators and may have zero impact on an attack if alerts are not generated or acted upon. IDS can be deployed at the network level and integrated or reside on hosts such as firewalls (Zeng, Yao, & Chen, 2010)(Chaudhary & Sardana, 2011).

IPS are different from IDS in a way such that it can react to changes and they are often deployed in a way that is inline like a network tap, which sniffs traffic between two interfaces or switches with content awareness (Zhang et al., 2004). Ideally, IPS are to be in a location capable of interacting with network streams, operate invisibly, integrate into management systems and have support for future types of attacks unknown (Douligeris & Serpanos, 2007).

Virtual Private Networks

Virtual Private Networks (VPNs) are virtual networks that provide communication through existing public networks appearing as if they were the organization's own private network. Essentially through emulating the properties of a real network, network traffic is tunneled over different networks transparently, but VPNs can also add security through encryption, integrity checking, authentication and access control. Generally, VPNs are categorized into two categories: remote access and site-to-site. VPNs may use one of the following technologies: Point to Point Tunnelling Protocol (PPTP), Layer Two Tunnelling Protocol (L2TP), IPsec, SSL/TLS, and Multi-Protocol Label Switching (MPLS) (Heinanen, Gleeson, & Armitage, 2000)(Jaha, Shatwan, & Ashibani, 2008).

Network administrators and organizations often use VPNs to extend their networks to remote locations or offices securely and to enable remote access for employees in a cost-effective method instead of owning or leasing dedicated networks. VPNS are also used to provide an extra layer of

security over unsecure connections such as wireless networks and extend security to legacy clear-text protocols (Sharma, Ghani, & Fang, 2007). This is an attractive option to improve security when used within networks hosting legacy protocols, but does not address the concept of moving security closer to the end-nodes or using dual-stack approaches for legacy protocols.

2.3 Challenges with Network Management

Network management consists of five categories of management functions within the OSI standard: configuration, performance, fault, accounting, and security (Sharma et al., 2007). While the OSI standard competed against the TCP/IP model, each of these functional areas are fundamental, but largely related to technical details that do not necessarily reflect the needs of network management professionals or their employers. As networks evolve both in size and heterogeneously with technological advancements, knowledge transfer among employees, human efficiency, budgetary costs and network management tools must adapt (Wallin & Leijon, 2006).

From an industry study, several organizations were surveyed for feedback regarding challenges faced by telecom operators. The study also outlined six problem areas such as excessive alarm management, constant technological change, complex service interactions, customer interactions, operational costs/budget pressures and interfacing with a heterogeneous technology environment (Wallin & Leijon, 2009)(Wallin & Leijon, 2006). Each has technological interactions and is classified into four main categories for this section.

2.3.1 Managerial

The term "managerial" relates to functions pertaining to management or a manager (Merriam Webster, n.d.). For network management, managerial tasks that provide insight for upper organizational employees include monitoring operational costs, budgeting, ensuring business advantages and services, and continued revenue generation (Gorur, 2006)(Wallin & Leijon, 2009).

For example, an enterprise's upper management has decided to create a new service, which is to scale across to numerous customers using an existing network. All testing indicates an implementation that would scale linearly with growth over time, however, after several weeks, technical issues

begin to pile up and pressure for an explanation is beginning to build. Upper management wants to remedy the situation for the least cost and see progress right away, while lower level managers must perform the necessary actions to mitigate this scenario.

Challenges for enterprise or employees are having the necessary information available to make the correct decisions when dealing with current enterprise-wide issues and accurately forecasting or planning for the future. A manager cannot investigate several thousand issues, but rather will look at the trends indicated by the information presented (Gorur, 2006)(Mokhov, Paquet, & Debbabi, 2014). For example, an immediate a lack of IPv4 addresses may indicate a need to migrate to IPv6. However, it may not be seen as cost-effective if technologies such as NAT are used or other more urgent issues are present such as improving network uptime (Claffy, 2012).

2.3.2 Daily Operations

Organizational size can greatly affect challenges for network management. In a smaller organization or enterprise, network management tasks may range from higher-level decisions to lower-level decisions such as handling organization technology purchases to resolving specific hands-on technical issues. Daily challenges include dealing with intra-organizational politics, human capabilities such as work performed per hour, societal tendencies, security alerts, managing/interacting with current software or hardware deficiencies and insufficient training (Mokhov et al., 2014)(Assels et al., 2011). While some of the above could be classified as belonging to both lower and higher levels of management, it is important to distinguish that daily challenges are something often associated with a boots-on-the-ground position of operating networks (Aaltonen, Karvonen, Norros, & Fuentes, 2013).

Poor human communication and missing information can also be a challenge for network operators. An organization may have slow or non-existent communications to network operators, customers or users may make changes by themselves or communications can suffer human-related qualities (Aaltonen et al., 2013). Tools that help network operators automate or autonomic facilities may reduce psychological and organizational demands to improve the efficiencies and reduce time-spent on menial tasks such as polling and collecting performance statistics (Kouadri & Brézillon, 2006).

2.3.3 Interoperability

Managing networks can be a daunting task regardless of deployment age. They can be newly created, but also consist of several vendors, OSes, and many devices. For example, there may be several Apple devices, Microsoft workstations, Linux servers and various networking hardware. However, through the usage of the standardized TCP/IP protocol suite, most, if not all of the systems can communicate given the correct installed applications and connectivity or authorization. Through managerial and organizational constraints, it often is not feasible to replace or update entire systems and operators must manage an intermingled network with modern and legacy devices side-by-side (Aaltonen et al., 2013)(Claffy, 2012)(Kurose & Ross, 2012).

To achieve interoperability, network operators often need to support technology transitions through methods in Section 2.4. This can even affect operations such as configuring ACL rules, as extra rules may be required to support multiple protocols needed to configure and monitor legacy and new devices. For example, a legacy commercial printer may require TCP port 80 for HTTP configuration, UDP ports 161 and 162 for SNMP monitoring and configuration, but to support another new printer on the network, TCP port 143 for HTTPS was also allowed. Potentially, these firewall rules may not be mis-configured unintentionally or configured for operator convenience and represent a security risk by allowing network traffic that may be normally blocked (Formyduval, 2009)(Wool, 2004).

If a vendor does not provide updates and/or infrastructure cannot be modernized swiftly or feasibly, security attack surfaces may be affected short-term and long-term. It is important to note that transitioning between technologies affects the organization business risks and need to be mitigated (Manadhata, Wing, Flynn, & McQueen, 2006)(Sun & Sushil, 2014). Ideally, these periods of transitioning are accompanied by assessment procedures, where a product is evaluated and deemed whether or not there is an effect for users, business factors and other objectives (Annosi, Penta, & Tortora, 2012).

Sentiments for interoperability may echo “first, do no harm”, “leave it alone if it works” or only implement after thoroughly investigating the effect on operators completing tasks. This mind-set has potential for neglect because aspects such as security may appear non-relevant while completing

a technology migration (Edwards, Poole, & Stoll, 2008)(Dourish, Grinter, de la Flor, & Joseph, 2004). Implementing new technologies implies necessary management of new systems, their legacy counterparts and inherently, network security is entangled with network management.

2.3.4 Network Security

According to the SANS organization definition: "network security is the process of taking physical and software preventative measures to protect the underlying networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction, or improper disclosure, thereby creating a secure platform for computers, users and programs to perform their permitted critical functions within a secure environment" (*What is Network Security?*, n.d.).

Network security is not a task or concept limited to upper levels of management, but can also be among daily challenges for network operators and linked to the management of a healthy network. It represents the mitigation of risks of activities that can invalidate a network's integrity and negatively harm operation. These security related activities can range from a disgruntled employee to unintentional access, hackers and malicious software infections, and all of which if left unmanaged can affect the stability of a network. Unfortunately, network security can negatively affect network management by increasing costs when implementing security controls, securely managing legacy devices, and when there is a loss of productivity or intellectual property due to human error (Reid & Gilbert, 2007)(Whitman, 2003). Network security can affect business motivators on choosing which risks or vulnerabilities warrant an expenditure for mitigation or additional protections for various types of threats or attacks (Casado, Freedman, Pettit, & Luo, 2009). Determining the frequency and likelihood of network security events is also a factor in strong network management today when connected to a multitude of Internet connected devices.

For management and operators, discovery and enumeration of network threats can be a difficult process to undergo and complete. However, risk can be mitigated or managed often through in-house designs or models such as Microsoft's threat modeling technique (Swiderski & Snyder, 2004). Theoretically threats or vulnerabilities could be anything: "a vulnerability is a weakness that could allow an attacker to exploit or take advantage of that security vulnerability such that confidentiality, integrity or availability of a product is compromised through an attack" (Microsoft Corportation,

2016). Additionally, "an attack vector is a path or means by which a malicious person can gain access to a computer or network server in order to deliver a payload or malicious outcome" (M. Rouse, 2012a). Through the combination of vulnerabilities and threat vectors, it can be difficult or impossible to quantify and explore in sufficient depth all risks that could affect network security; defining realistic and adequate scope is a necessity (Swiderski & Snyder, 2004).

2.4 Transition Management

Transition management of modern day technology is best expressed using the automobile as an analogy. Early day automobiles featuring a combustion engine are very similar today in design. They have four wheels, a steering wheel, instrumentation for acceleration and deceleration and the concept of a road on which they drove. They were greatly slower than a modern car, but they could still drive alongside modern vehicles on most roads today. In essence, technology should transition gracefully and safely just like how current automobiles are superseded with electric and automated models today. Transitions should also be able to provide competitive advantages for system users and management while being aligned with the business needs, feasible and graceful (Young, 1993). Carefully managing technology transitions is an important activity for organizations to correctly execute. Strategies must be developed to cope with the development of new products, assimilation of legacy technology, global organization needs and needs within the organization itself (Cyert & Kumar, 1994)(Xu, Chen, & Guo, 1998)(Harmon & Laird, 2012).

When referring to software and hardware engineering, common interfaces and middleware describe two approaches when two different architectures are required to seamlessly interact with each other (Spivey & Flannery, 1991)(Spivey, Munson, Nelson, & Dietrich, 1997). For example, fiber optic network cabling and Ethernet copper cabling are different media, but requires software and hardware logic to translate effectively bidirectionally. Translation software has fewer difficulties to implement when compared to hardware translations because hardware may be near-impossible given deployment conditions. For example, newer IPv6 networks need to work alongside IPv4 "legacy" networks through tunnelling or translation while networks are upgraded and replaced. The same for legacy network devices operating adequately may offer limited incentives to upgrade, but

should eventually be migrated as technology changes through an assessment and testing phase (Annosi et al., 2012).

2.4.1 Middleware

In 1968, the term "middleware" was used to describe a layered approach when dealing with software to handle seamless changes to data from one source to another without causing a cascading effect especially if an underlying layer is altered (Naur & Randell, 1969). Today, middleware is ubiquitous whenever "glue" is required to interface different technologies together by creating a conversion or translation layer (Ibrahim, 2009).

Shared or common interfaces in the context of middleware can be deployed through Application Programming Interfaces (APIs) or application wrappers for legacy applications. Not only do common interfaces provide a mechanism for collaboration with different systems, they can be used to provide a layer for communication that would be normally be considered difficult. Distributed systems and Service-Oriented Architectures (SOAs) each often use their own approaches at the implementation level, but often expose or utilize interfaces such as SOAP or REST for interoperability (Millard et al., 2006).

(Venturi, Stagni, Gianoli, Ceccanti, & Ciaschini, 2007) suggest that there should be a set of standardized interfaces and resources should be shared even if there are different vendors or middleware involved. Through standardization of common interfaces, interoperability can be achieved and potentially security, but security may be assumed and not formally examined (Foley, Quillinan, O'Connor, Mulcahy, & Morrison, 2004)(Yin, Shi, Sui, & Wang, 2009).

This section describes common approaches to handling technology transitions and related works to SNMP translation or migration to NETCONF.

Gateway Devices

Computer networks, especially in highly proprietary environments, are often unable to work from one vendor or network medium to another. The heterogeneity of these types of networks has created problems with interoperability and portability, which can be solved using a conversion

solution or middleware for effective communication. This form of middleware handles data communication between two or more sources that use protocols specific to each. For example, translating industrial protocols IEC 60870-5-101 to/and from IEC 60870-5-104 protocols can be performed using a communication gateway device (von Gordon & Hancke, 2004).

Communication gateways can be described as multi-protocol gateways where an engineer can mix and match protocol translation from a selection of inputs and outputs. Alternatively, protocol converters typically convert between only two protocols and do not have the flexibility of communication gateways (Unsoy & Shanahan, 1981). Protocol converters are the simplest in construction and are the basis for either a one-to-one or one-to-many protocol relationship. Either gateways or converters can be used to handle translation between network media such as RS232 and wireless (Hong, Song, & Lin, 2015). More recently, IPv6 transition solutions are commonly being deployed through a dual-stack approach or tunneled one over the other when traveling through a heterogeneous Internet composed of both IPv6 and IPv4 (Aravind & Padmavathi, 2015). In February, 2016, a NETCONF SNMP gateway patent was granted specifically mentioning two interfaces (NETCONF and SNMP) translating between themselves using a "buffer" and outlines operations to be closely coupled protocol to protocol semantics (e.g., a NETCONF lock message directly translates to at least one SNMP GET operation) (VIEIRA, 2016).

Alternatively, a similar server-based approach called "proxies" can be used for translation of protocols or hardware. Proxies are typically used as an intermediary within the networking domain for content caching, content filtering, traffic tunneling and access control enforcement. For the purposes of this research, inline or invisible proxies that act as an interface for a service, but can also provide security features, will be considered an application proxy or a friendly MiTM (Shapiro, 1986).

SNMP to X Conversion

As technologies change, often management technologies must change as well to adapt. As listed inSection 2.2.3, SNMP utilizes an ASN.1 model for describing data, but the newer NETCONF/YANG uses XML. Before the invention of NETCONF, XML conversion for usage within

management software was being explored (Gao, Xing, Zhang, & Li, 2010). The difference in protocol data models is a key aspect when transforming SNMP to any other data model and user space or application level translation can be performed through technologies such as Simple API for XML (SAX) and PHP: Hypertext Preprocessor (PHP) (Neisse, Granville, Ballve, Almedia, & Tarouco, 2003)(Gao et al., 2010).

Effectively, transformation of SNMP to XML can be categorized into four layers (Klie & Straub, 2004):

- XML Data design - representation of the MIB as XML
- Separation of data model and data - a schema that is referenced separately from the data
- SMI MIBs are mapped to XML schema definitions - e.g., an unsigned integer maps to an XML representation of an unsigned integer
- Implementation - MIB to XML in operation and implementation

SNMP agents or SNMP clients are also integrated through transformation into components of XML-based management systems using HTML as the application layer network protocol. Despite some of the drawbacks of SNMP, SNMP is quite optimized for low-resource consumption on limited devices (Klie & Straub, 2004). SNMP data transfers such as bulk operations seem to not be supported quite as well (Yoon, Hong-Taek, & Hong, 2003).

In addition to SNMP being converted to XML, SNMP was explored for possible abstraction by converting SNMP MIBs to XML and using macro translation to bidirectionally translate protocol specific elements. This approach requires vendor input to determine all OID elements and mappings to XML, and NETCONF operation is 1:1 generically mapped to SNMP operations (e.g., NETCONF ¡get¿ ¡get-config¿ is equivalent to SNMP ¡get¿ (Wu & Chang, 2010).

MIBs to NETCONF/YANG (Read-only)

Within the IETF NETCONF/YANG community, there has been some work regarding converting SMIv2 MIB objects into YANG modules for read-only access when used with NETCONF (claise,

2014). The read-only access is largely due to the way NETCONF handles configuration state information when running and does not normally allow for alterations of running configurations. The NETCONF configuration limitation is not present within the SNMP protocol because SMIv2 objects can be either read-write or read-create. As a work around for this limitation, YANG modules may be written using a concept called "deviations". Deviations describe behavioural variations that allow a YANG module derived from the SMIv2 input to be considered valid (Schonwalder, 2012). This is used where SNMP and NETCONF functionality allows twisting of the semantics provided by edit-config or copy-config allowing such changes to remain as YANG data nodes.

Dual-stack Approach

Dual-stack currently describes IPv4 and IPv6 co-existing as two separate native stacks and is recommended by Cisco for most Enterprises (Cisco Systems Inc., 2010). However, dual-stack has roots during the period of time when the Internet would use either OSI or TCP/IP and explores an approach for their co-existance (Baeza-Yates & Manber, 2012)(Rose, 1990). Recently, dual-stack approaches to technology co-existance has been used for software SSL fallback when TLS is present. Given the usage of two simultaneous stacks, this research will refer to the concept of a dual-stack approach in situations where SNMP and NETCONF are deployed simultaneously using two native stacks (Freier et al., 2011).

Dual-stack approaches have advantages where all hosts support at least of one of the stacks, and can communicate should one stack be unavailable on the host they are attempting to communicate with. This also provides flexibility and zero performance degradation as packets will be transmitted natively without virtualization or translation. Unfortunately, administration complexity may be increased if hosts are deployed with different hardware and have multiple protocols in use (Rose, 1990)(Rose, 1989).

Using the above dual-stack approach is trivial to administer if there are few machines and limited ingress/egress points through firewalls. However, as this approach scales and more NMS stations are used in a distributed manner, the complexity of configurations for devices will increase with a dual-stack approach.

Chapter 3

Problem Statement

In this chapter, concerns related to the problem domain and the specific problem statement will be outlined. For ease of understanding, the problem has several areas of concern that need to be addressed in order to improve security of legacy devices:

- Device related
- Protocol related
- Dual-stack related
- Firewall, DPI and IDS/IPS related
- VLAN and VPN related
- Management and Operator related

After expanding each of the above problem concerns, the problem statement will be presented.

3.1 Device-related

Legacy devices represent differing challenges from those often seen in consumer electronics such as laptops or smart phones. Industrial applications usually have a longer deployment time and minimal plans to upgrade old, but still functional hardware. Installations may have an inventory

of spare components that unless depleted, delay upgrading legacy products to modern alternatives. These types of devices can be embedded systems that are resource constrained, use proprietary firmware, are bound to specific types of network media other than modern day Ethernet, or have safety or legislated restraints (Priller, Aldrian, & Ebner, 2014)(Tjoland, Brennan, & McPhee, 1999).

Devices that are past End-of-Life (EoL) or do not have support for newer firmware or technologies will severely impact mitigation measures. Specific upgrade paths such as those in the following sections may be unfeasible or impossible on existing devices.

3.2 Protocol-related

Earlier versions of SNMP had inadequate security mechanisms: clear-text or non-encrypted transmission, limited integrity verification and non-granular user permissions. Later versions had encryption and improved user permissions, which included role definitions, but are largely more complicated to implement. Both SNMPv1 and SNMPv2c rely on the same insecure authentication mechanisms (clear-text community/private strings), however, migrating or upgrading to later versions of SNMP may not be possible for the following two reasons:

(1) Existing deployed devices may not be upgradable to SNMPv3 (or even earlier versions)

(2) Operators may avoid migration to SNMPv3 due to effort and cost, particularly if another protocol is desired (e.g.. NETCONF) (Bierman, 2001)(Corrente & Tura, 2004)

Furthermore, when unable to migrate legacy SNMPv1/SNMPv2c to either SNMPv3 or a modern protocol such as NETCONF over SSH, improvement of legacy protocol security is limited to the following:

(1) Preventing unfiltered or unauthorized access to legacy devices

(2) Preventing write access to legacy devices through protocol controls

(3) Securing network traffic by tunnelling it through a VPN

(4) Replacing legacy devices with modern devices (hopefully more secure)

Item 1 can be satisfied using best-practices in secure network topologies by corralling legacy and sensitive devices into secure zones. Legacy or modern, this should be performed regardless of protocol (Flauzac, Nolot, Rabat, & Steffenel, 2009). However, security isolation does not satisfy items 2-4 in the list. Item 2 can potentially be implemented through basic SNMP private/community configuration strings, however, better protection through granular security controls in SNMPv3 or SNMP DPI can prevent "all-or-nothing" access upon discovery of the community strings. For example, using DPI, a policy could be derived to only allow access for the first several objects, but exclude later objects, which would normally be accessible unless DPI were used. Item 3 may be a preferable and a low-hanging fruit for security, however, it can increase overhead for operators, increase topology complexity and network latency through encapsulation. Item 4, depending on the organization, is often non-feasible for operators due to constraints listed in Section 2.3. However, it has potential for success if sufficient motivations technologically or monetarily exist to drive an effort to replace all legacy devices.

Items 1 and 2 are restricted when NETCONF over SSH or SNMPv3 with encryption are used. Encryption limits DPI because cryptographic mechanisms limit packet content visibility by providing confidentiality. For obvious reasons, granular DPI is not effective without access to unencrypted data streams, unless a middlebox for encryption/decryption operations is provided before DPI can be performed (Naylor, Schomp, & Varvello, 2015).

Additionally, the usage of proprietary SNMP MIBs has also restricted the flexibility and openness of data structures. This has prevented administrators from easily extending existing data structures for devices as well as negatively affecting large platforms by increasing computational resources when performing "discovery" operations (Google, 2016)(Bi, 2012).

3.3 Dual-stack

As alluded to in Section 2.4.1, both SNMP and NETCONF/YANG are often deployed side-by-side to reduce technology transition risk and improve interoperability. While NETCONF uses modern protocols for underlying secure communications, earlier versions of SNMP do not and connections are established insecurely. Each of SNMP and NETCONF generate different vectors

that can be utilized by attackers:

- Management software
- ACL rules for each protocol
- Specific vulnerabilities for each protocol
- The client management software
- Host-specific intricacies

Given the nature of interoperability while "doing no harm", the dual-stack approach will likely remain operational well into the future even when not protecting legacy devices. This may also have a compounding effect with the pace of technological progress; current dual-stack approaches will likely be modified with additional dual-stack approaches generating further legacy security loopholes. For example, today there is SNMP and NETCONF, but tomorrow there may be SNMP, NETCONF and NextNewFutureProtcol without migrating the first two or even SNMP which is the eldest. To limit the compounding effect, network operators must be aware of the consequences of enabling extra features when only having the goal of maintaining operations for legacy devices immediately, and the consequences of temporary transitional measures must be knowingly noted for future operators so they are not forgotten.

3.4 Firewall, DPI, IDS/IPS related

For each firewall device traversed (designated as *fd*), devices requiring access (*id*) and each protocol that should be allowed through (*n* equals 2 = NETCONF and SNMP), dual-stack firewall ACLs require at a minimum of *fd * (2 * id)* rules instead of *fd * (1 * id)* if a single protocol is used. This excludes additional details and device specific operations for controls against directionality and the underlying protocols for each SNMP (UDP) and NETCONF (TCP). Configuration of ACLs is a labour intensive operation when there are large numbers of firewall devices on the network, particularly when these devices are commonly accessed manually through CLIs.

For UDP SNMP traffic, connection state cannot be discerned without protocol knowledge. Therefore using purely ACL rules does not provide sufficient security controls for directionality or optimal protection against address and packet spoofing. This is due to the connectionless nature of the UDP protocol. Without DPI (Section 2.2.5), the firewall will be ignorant of the version of SNMP in use over the connection. If there had been a version of SNMP created using TCP, then security using connection state-tracking could be improved.

Alternatively, NETCONF utilizes TCP and therefore, firewall ACL rules can be combined with state information to track initialization, establishment and tear-down. If the firewall has protocol awareness using DPI or heuristics, the firewall can also inject TCP packet(s) with RESET flags that (ideally, if sent correctly in both directions) forces the connection to be re-established.

For the above reasons, firewalls do not typically improve interoperability without operator intervention to modify and verify configurations for correct operation. Ideally, once legacy devices are retired from the network, the operators must also be vigilant and remove the unnecessary ACL rules. These limitations imposed on the operators are necessary, but further exacerbate risk of human error in these potential situations:

(1) Operators should have an inventory of all active devices such that connections are not erroneously severed

(2) ACL rules may be forgotten by operators and leveraged by an attacker

(3) ACL rules may be misconfigured, lacking granularity or specifics for addressing and protocols

Item 2 can be mitigated by operators inventorying their network to prevent unused TCP/IP ports being enumerated by attackers performing reconnaissance or active attacks. Additionally, item 3 is further affected if firewalls are protocol unaware or lack DPI/IPS functionality. This limits dual-stack security to state-full ACLs, TCP/IP connection tracking and potentially allows attackers to inject custom (crafted) packets or exploit specific vulnerabilities without the extra scrutiny provided by DPI/IPS. It also places the burden of additional security on host applications, which may be insecure by nature, unpatched, operating past End-of-Life (EOL) or unable to be updated due to deployment environmental constraints.

For item 1, it is ideal in some environments if operators can utilize active protective measures such as interrupting network connections by dropping/stopping packets. Port scanning for open or unfiltered hosts should be used cautiously and may be inappropriate for highly sensitive critical environments (Stouffer, Falco, & Scarfone, 2011). If connections cannot be actively monitored and processed, adequate logging procedures and IDS should be used at a minimum for auditing purposes. Passively logging or monitoring connections will not suffice to positively affect items 2-3, but will hopefully provide information during an active attack or for post-attack auditing (Vaarandi & Pihelgas, 2014).

3.5 VLAN and VPN Related

Protecting and further segmenting network zones according to best practices requires using VLANs and IP subnetting (Narasimhan & Triantafillou, 2012). As noted in Section 2.2.5, VLANs do not provide direct security, but can assist in providing improved granular security controls at layer 2 (Ethernet layer). To reiterate, VLANs are:

(1) Unaware of higher level protocols above the VLAN layer

(2) Best utilized for fine-tuning IP subnetting and broadcast domains

(3) Used to provide physical port configuration into reconfigurable groups

Even though VPNs are able to provide confidentiality and integrity to tunneled network traffic, they are not a blanket solution providing end-to-end security. There is a number of factors including configuration that can affect the security of the VPN and traffic contained within. For example, an organization must be aware that security can be affected bidirectionally through end-points and by other hosts within the VPN network segment through malicious activity and poor management of credentials/certificates. If credentials are weak in strength (e.g., simple to guess or short), an attacker can potentially brute-force access into a system, and unsuspectingly, malware such as worms can infect seemingly unconnected hosts. Furthermore, if credentials are poorly protected, malicious users could enter a network deemed secure. VPNs may be implemented as part of a firewall solution including ACLs, but VPNs are generally protocol unaware and DPI/IPS is required. Using VPNs to

protect the confidentiality of network traffic when being transmitted over insecure and potentially insecure networks such as wireless, requires the following at a minimum:

(1) Support hardware or software VPN implementations

(2) Sufficient resources for adequate performance

(3) Extra configuration by operators

VPNs are also restricted by deployment/platform and whether devices such as network switches have support for a client embedded within. For example, it is unlikely that the software contained on switch devices will include a VPN client and be configured as a VPN end-point such that it can tunnel insecure legacy management protocols to itself. This may be a solution for some models of routers, but not on "simple" IoT devices such as temperature probes or Uninterruptable Power Supplies (UPSs).

3.6 Management and Operator Related

As noted in Section 2.3, network operators, upper levels of management and users are all concerned with adept network management. Whether it revolves around the stakeholders being the organization's chief technical officer (CTO) or an end-user purchasing a service, effective network management is crucial for network monitoring coverage, improving operating costs, preventing litigation, managing security and end user satisfaction. For operations such as implementing security and transitions, the following must be acknowledged:

(1) Maintaining Service Level Agreements (SLAs)

(2) Minimizing required resources and costs for network management

(3) Dedicating resources for implementing and managing network transitions

(4) Dedicating resources for continuous and ongoing network management/security

(5) Maintaining and/or improving organizational network management and security practices

(6) Minimizing or eliminating network disruptions or performance degradation

(7) Maintaining and/or improving the level of visibility of network entities

(8) Maintaining and/or improving the level of security within in the network

(9) Secure management of device passwords, configurations, cryptographic passkeys and certificates

Beginning with an end-user group with the least visibility into an organization's structure, end users will likely only be concerned with item 1. User needs will likely push technology transitions based on lowering costs, adding features and increasing performance, but if security is not an industry cause for motivation, effort for items 2-9 may be minimal (Kreutz, Ramos, & Verissimo, 2015) (Cyert & Kumar, 1994)(Xu et al., 1998)(Harmon & Laird, 2012)(Reid & Gilbert, 2007)(Whitman, 2003)(Casado et al., 2009). Item 6 can be linked to item 1 because SLAs will typically cover contractual aspects including network up-time and minimal down-time.

Items 2-9 also can be interpreted differently by the levels within the organization. Upper management may see these as procedural, reporting, compliance related or even contractual. Lower management (including operators) may see these as daily events, increasing human efficiency, password/security management, network monitoring for new or rogue hosts, maintaining network operation for performance and managing issue tickets or errors. Item 9 is listed for completeness and is a topic for attention, but will not be covered by this research.

Network management and security are limited by individuals, the organization, and is cost-prohibitive. It is also constrained by the tools and quality of resources operators may have access to. For example, if an organization or department has a limited budget, security and network improvements will be prioritized such that operators may have to manage with using inefficient tools in order to keep the business profitable or sustainable.

3.7 Related Requirements and Problem Statement

After describing several areas on concern in Sections 3.1 to 3.6, we derived four requirements that broadly summarize these concerns. These are introduced as necessary aspects to improve the

security and management of legacy devices:

(1) Integrate into modern network management paradigms

(2) Improve granular security controls where there may be none

(3) Improve security in general without adding complexity for legacy devices

(4) Integrate into heterogeneous network transparently

The first requirement is semi-addressed through the existence of SNMP-to-XML and NETCONF/YANG MIB modules, but an actual implementation on how to seamlessly translate between SNMP and NETCONF/YANG bidirectionally is absent from the literature (Bjorklund, 2010)(Gao et al., 2010). Regardless, if an organization has deployed or invested in NMS, minimal effort should be required to integrate any technology. Upstream NMS solutions such as Tail-f's ConfD integrate multiple protocols using a dual-stack approach and convert the desired protocol into a proprietary data-model without converting SNMP and NETCONF using a 1:1 method. Additionally, there is a knowledge gap on how an actual implementation could be used by researchers to explore and identify any potential advantages and disadvantages from a middleware approach to improve security and interoperability of legacy devices.

The second requirement asks to improve security controls where current controls may be either loosely defined, insufficient or non-existent. This is noticeable when legacy devices (including SNMP hosts) have no upgrade path to versions with adequate security, utilize insufficient SNMP security controls, or lack the ability to migrate to NETCONF. Modifying a device without open firmware or a commitment by the vendor for upgrades will not feasibly satisfy the second requirement. However, security controls can be improved by protecting access to the devices through more restrictive ACLs and utilizing a state-full transport protocol. By using DPI/IPS technologies with protocol aware components, device specific controls can be added, which may include pattern matching for vulnerability signatures or enforcing organization specific policies. However, any solution aiming to satisfy the second requirement requires protocol awareness and configurable policies for more granular access control.

The third requirement specifies that security for legacy devices must not add more complexity for operators when performing modifications to existing infrastructure. NETCONF/YANG migration will inherently require modifications to firewall ACLs, but state-less rules for SNMP can be removed if a dual-stack approach is not chosen. NETCONF increases management complexity by requiring resources for managing cryptographic certificates and credentials if such an infrastructure does not already exist. Generally, it is assumed that key/user management should already (to a certain extent) be in place for providing secure access to web and local resources including VPNs. Adding VPNs to provide protection for all traffic instead of specific vulnerable channels may increase security transporting insecure protocols, but does not protect traffic within the VPN and can increase network complexity. In a perfect world, legacy devices should be left untouched, magically integrating into modern technologies. Security solutions should be transparent to both legacy devices and NMS.

The final requirement realizes that operational networks are not homogeneous and have a blend of technology vendors, solutions (in-house or external) and eras of components. Legacy devices may be present among more modern alternatives being deployed. Alternatively, no modern replacements may exist and legacy technologies are kept in place indefinitely. Supporting both legacy devices or devices that have limited support by a more modern management platform represents a problem for network administrators. If product solutions do not provide adequate flexibility, in-house custom interfaces such as those used by Concordia University can be used (Assels et al., 2011). Dual-stack approaches do satisfy the forth requirement, but negatively affect requirements 2 and 3.

Additionally IDS/IPS technologies utilize similar concepts explored by the proposed solution, but are differentiated because they aim to provide generalized security while unable to convert one protocol to another. They also can have negative performance and require high levels of system resources for operation (Salah & Kahtani, 2009). IDS and IPS in passive modes will provide intrusiveness security without adding complexity, but to take active actions they are required to have protocol awareness (that may be too generalized or incomplete for critical infrastructure).

While it is not required to develop and test a solution that can operate using low-cost and resource constrained hardware, it is important to determine whether there are any technical barriers that may obstruct product development. If implementing security for legacy devices requires higher

performance devices then ubiquitous deployment may not be feasible or face organizational challenges when security is costly. For example, if the cost of security for wide-spread deployment is high, and the budget needs to be allotted to a new service, the new service will likely receive the funding allocation instead of a project securing legacy devices.

This research explores the issues of:

- Given that legacy devices may not have an upgrade path, can a system be designed that improves network security of these devices?
- Whether this new security can be implemented in a way that reduces the complexity of a solution, achieve the above requirements (1-4) and be compared to an existing transitional approach (e.g., dual-stack)?

3.8 Problem Evaluation Approach

In order to achieve improved security for legacy devices using a developed solution, legacy devices must be deployed and communicated with using both NETCONF and SNMP.

Firstly, a dual-stack approach must be demonstrated and evaluated for the purposes of a security surface evaluation. Devices hosting either NETCONF and SNMP agents will be queried as is typical of a network operator performing systematic information retrieval and setting of configuration values. For the purposes of simplicity, only GET and SET type operations will be performed.

Secondly, a developed solution that aims to secure legacy devices as an alternative to a dual-stack approach will be evaluated using the same administrator operations as the previous step. This will be performed using the following methods to create a comparative model of security for legacy hosts and the effectiveness of the developed solution:

- **Analyse packet flows** using traffic captures upstream and downstream of the developed solution and a firewall deployed in a dual-stack configuration
- **Generate threat models** describing attack surfaces, data boundaries and possible mitigations for both the developed solution and dual-stack approach

Finally, once a comparative model has been derived from the previous two steps, we can determine whether the developed solution is technically feasible, what advantages and disadvantages it has relative to a dual-stack approach and where it can be utilized with existing security/networking tools to provide increased security for legacy devices.

Chapter 4

Proposed Solution

In this chapter, we will outline a proposed solution called the "NETCONF-SNMP Protocol Gateway hybrid" (NSPG). Using a hybrid approach by combining protocol proxy and gateway functionality, the solution improves legacy system security as noted in the problem statement and achieves the requirements listed in Section 3.7.

In order to gain a comprehensive understanding of the proposed solution, the following topics need to be explored:

- **Objectives of the proposed solution** and the exact requirements to be satisfied through the completion of an objective
- **Detailed description** of the approach and reasoning behind the design decisions
- **Novelties** of the approach accomplished either directly or indirectly

4.1 Objectives

The proposed solution accomplishes the following objectives (re-iterated from the problem statement):

(1) Integrates into modern NMS paradigms

(2) Improves granular security controls where there may be none

(3) Improves security in general without adding complexity for legacy devices

(4) Integrates seamlessly into heterogeneous networks transparently

Objective 1 is accomplished by utilizing NETCONF/YANG/SSH as the primary upstream communication protocol to interact with an NMS. Secondly, objective 1 utilizes XML for managing configuration data and request/responses as is used currently by NMS. SNMP (versions 1, 2c, and 3) are supported, but they are indirectly used through the NSPG on the downstream interface. Thirdly, any software that can be integrated into modern NMS needs to support modern protocols. The NSPG uses NETCONF over SSH and acts as a mere NETCONF agent, but can be connected via an interface capable to communicate directly with an NMS or NETCONF capable CLI (e.g., CLI program that converts NETCONF/YANG XML operations and results into human-friendly interface primitives). Furthermore, objective 1 is met by the NSPG using YANG as a method to describe its capabilities. This allows any NETCONF capable client/software to be able to connect to the NSPG NETCONF agent and use these features IF it is in possession of the NSPG YANG model.

Objective 2 is observed and achieved by creating a set of XML policies that dictate NSPG operation, specify legacy devices for communication, specify the data to be polled and give the mapping of a 32 byte UUID to a user ID. Using a *NIX user paradigm, a user can be setup such that it can be assigned to various groups. These groups can be assigned, reassigned and removed, but their presence when assigned to a user signifies which policy can be used (specified by a UUID) should an incoming NETCONF connection query a protected device. Earlier versions of SNMP (1 and 2c) have a security paradigm limited to two clear-text authenticated groups: public and private community strings. If deployed SNMP devices use only a small selection of differentiated strings, a single user role may have access to more devices than necessary for role-based access control (RBAC) given the principal of least privilege. Using Linux OS user internals, a user (e.g., Jane) can be a member of group 123 and access device A, but not group 456 and cannot access device 2 despite the same private/public strings on configured devices. Furthermore, in future versions Jane does not need to know the actual SNMP strings used unless she has the permissions to view them.

Objective 3 states that security shall be improved without adding complexity for legacy devices

(and inherently their operators). The NSPG meets this goal by providing a simple to use XML interface (alluded to in objective 2), which maps a user to group identifiers used to identify associated policies. Any changes to how the NSPG operates or a user using the solution communicates with legacy devices are made only on the NSPG itself. For example, if a network technician is removed from his/her job, the administrator simply needs to revoke the user from the NSPG's host user facilities. In tandem with Objective 2, NSPG installation is no more difficult than plugging/unplugging one or two Ethernet cables, creating/modifying an XML policy file (for the NSPG) and executing NETCONF commands (either through an NMS or NETCONF capable CLI). At the network level, security is improved by the fact that NETCONF uses TCP to provide connection-state to upstream firewalls (whereas UDP does not have state without DPI) as well as redundancy on networks where packet loss and congestion may be present. The upstream connections use SSH to provide confidentiality of data and access controlled via the NSPG to never allow direct access by upstream hosts to downstream legacy devices. Otherwise, if upstream hosts had communicated with a clear-text implementation of SNMP, network traffic could be passively collected and the credentials maliciously used. The usage of the NSPG as a friendly MitM is also advantageous if downstream hosts are susceptible to malformed packets or deviations from the protocol specification; a single protocol implementation can be used, thus simplifying testing and improving stability.

Objective 4 states that neither the upstream nor downstream devices can know about each other directly and can be situated adjacent to modern and legacy devices alike. Effectively, the NSPG appears to be a typical NETCONF agent to the managing upstream NETCONF connection, but it is seamlessly performing SNMP operations and returning any corresponding data through the NETCONF paradigm. Alternatively, the downstream devices are unaware that they are eventually speaking to an upstream host that does not "speak" SNMP and continue to communicate SNMP in ignorance. The NSPG also has a configurable polling rate, which can block an upstream NETCONF user from intentional (possibly malicious) and unintentional polling legacy devices in a way that could cause a negative effect. Using a caching data structure within the NSPG also allows low-latency transactions for already polled data improving the appearance of a truly seamless user experience. Should modern devices be deployed adjacent or downstream of the NSPG, traffic needs only to be forwarded as if the NSPG were not even present.

4.2 Detailed Description

There are three possible deployment strategies that can be used with the NSPG (see Figure 4.1 for an illustrated comparison):

- "Bump-in-the-Wire" (BitW) effectively becoming a "friendly MitM" such that all NETCONF traffic and access to legacy devices must be routed through the NSPG
- "Bump-in-the-Stack" (BitS), which is an additional layer in the processing stack on legacy devices that can be modified
- NETCONF termination point deployed somewhere on the same network where access to legacy devices is required

Clearly, the latter two approaches have merit and advantages in certain deployments, but only the BitW approach will be used in this thesis for validation of the technical feasibility and security.

Using the BitW approach, the NSPG is deployed inline between upstream modern hosts such as an NMS and downstream legacy devices using SNMP for management and monitoring (see Figure 4.1). Assuming that downstream devices are not already compromised, the NSPG in BitW configuration provides network security without modifying legacy devices in any way. This enables the following functionality:

- All traffic can be forced to pass through or to be inspected by the NSPG host and any unfiltered direct-access to devices downstream can be blocked
- Non-policy abiding network traffic can be intercepted and existing policies can be enforced using NSPG policies
- Protocol sequences or messages that may be non-conforming to standards or "problematic" are not forwarded to legacy devices; only valid and known SNMP traffic can be generated by the NSPG
- Incoming upstream NETCONF network traffic can be directly forwarded if downstream hosts communicate using NETCONF or other protocols except for SNMP

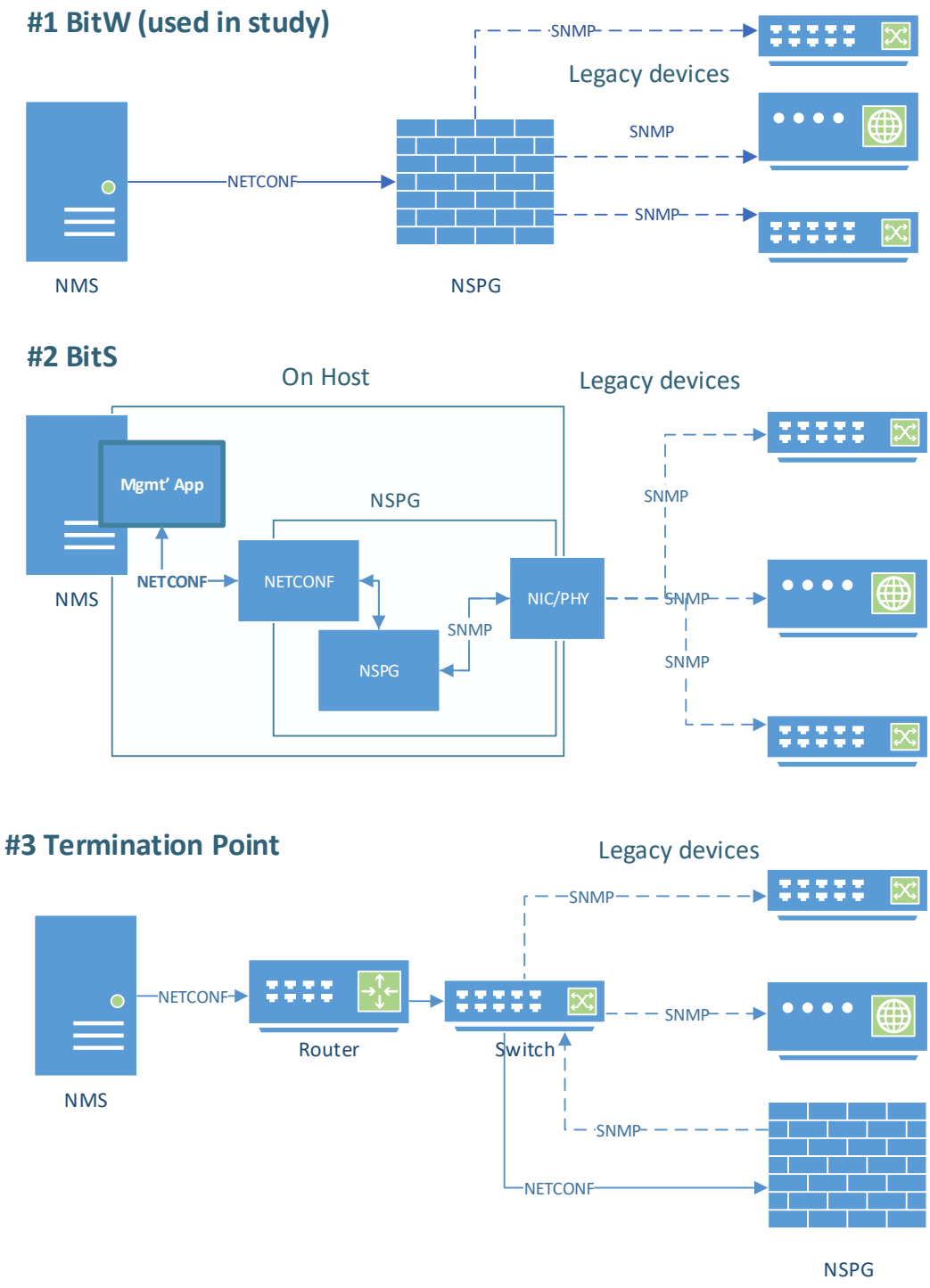

Figure 4.1: Possible proposed deployments for solution

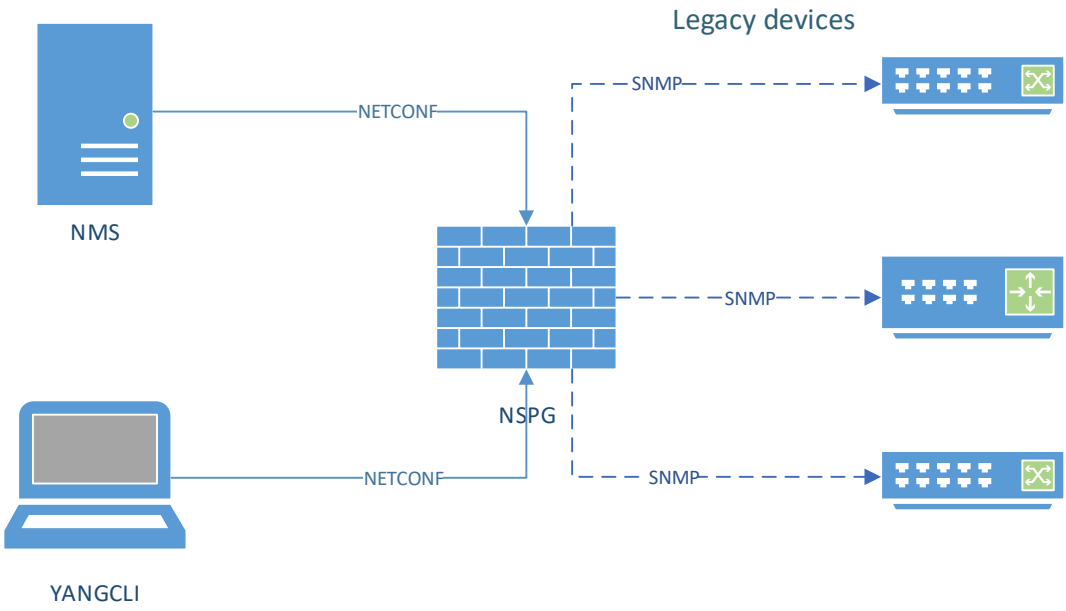

Figure 4.2: Logical BitW topology

In addition to being a BitW, the NSPG combines four major components to provide the transitional glue between NETCONF and SNMP (see Figure 4.3):

- NETCONF/YANG agent
- Policy and authentication management
- Cache/data store
- SNMP master

4.2.1 NETCONF/YANG Agent

For upstream devices to communicate to a NETCONF capable host, both the client and NETCONF agent are required to be possession of the same YANG model describing a particular set of functionality. In this case, for a client to interact with a NETCONF agent hosting NSPG-specific features, both the client and server need to have the NSPG YANG model installed/loaded (listed

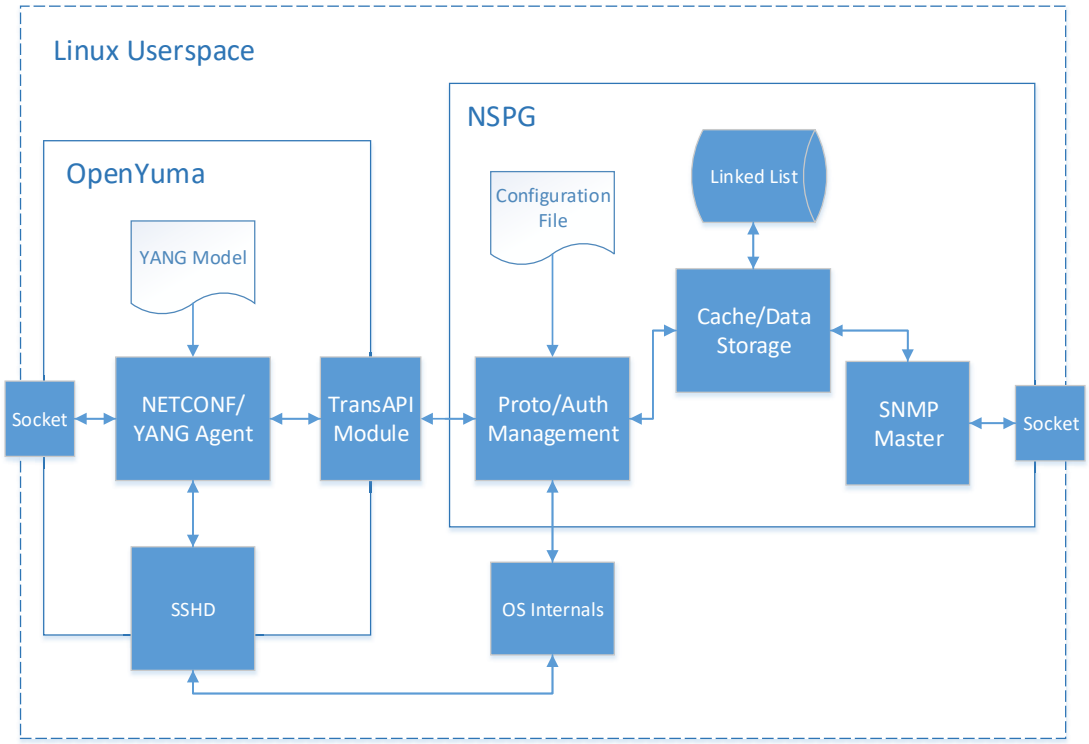

Figure 4.3: NSPG component block diagram (on the left is the upstream, right is downstream)

in full in Section A.5). Not only does the YANG model describe specific features available to the client, but lists commands available for execution (query authorized single policy or all policies available), input parameter validation requirements and output data formats. Without the NSPG YANG model, a client or a NMS could connect to the NSPG NETCONF agent, but would only be able to communicate using basic NETCONF defined in (Bjorklund, 2010).

The NETCONF/YANG agent component leverages a freely available open-source NETCONF agent called "OpenYuma" seen in Figure 4.3. OpenYuma exposes a NETCONF interface, integration into the Linux SSH daemon allowing access to the OS user authentication facilities and NETCONF TransAPI loadable module interface. The TransAPI modules are a very important piece of the design because they allow the OpenYuma agent to be extended with minimal effort through an API that allows the agent to load binary libraries at run-time (Shared Objects or Dynamic-Link Libraries)

While the NSPG YANG model does not include functionality to configure the actual NSPG

policies, it allows any upstream NETCONF management to perform queries that are validated against the NSPG host user access facilities. Effectively, OS user authentication that is provided by the SSH daemon allows a NETCONF connection to connect and be validated against a known list of users on the system. The list of the users available on the system is beneficial because a valid user can have system groups created and assigned. For example, the **user "Bob" and group "01be2dc8689d4e789926d0ceb7e542fe"** can be created. Bob can now be assigned to the group without affecting any system privileges, but can be used as a privilege identifier for other purposes such as NSPG policy authorization. This mapping of groups to users allows a list of group names (which are 32 bytes long) to be mapped 1:1 to the UUIDs (when scrubbed of the "-" delimiter) used in the policy management component listed in Section 4.2.2.

Figure 4.4 is an example exchange between the NETCONF master (using a user named "Bob") and the NSPG NETCONF/YANG component.

Figure 4.4 highlights a sub-component that interconnects the NETCONF TransAPI module and the remaining components: policy and authentication management, cache and data store, and the SNMP master. Using a UNIX domain socket as a form of Inter-Process Communication (IPC) and a simple request/response protocol, data can be sent bidirectionally for NETCONF queries (Linux Man-pages Project, 2015e). Alternatively, other methods such as shared-memory or local-host TCP/IP sockets could be used, but for security and simplicity UNIX domain sockets were chosen (Linux Man-pages Project, 2015c)(Linux Man-pages Project, 2015b).

This design choice improves the security of the NETCONF component by allowing Linux security mechanisms to be used and isolates the data stream when compared to using OS system() calls to execute commands (Linux Man-pages Project, 2015d)(CERT, 2016). If system() and similar calls are used to execute functionality such as local CLI commands (e.g., rm -rf *) then any number of vulnerabilities could be introduced. For example, an attacker could manipulate the buffer of a char array that contains the function calls input and have the system call execute the attacker's command (not the one the program intended to run).

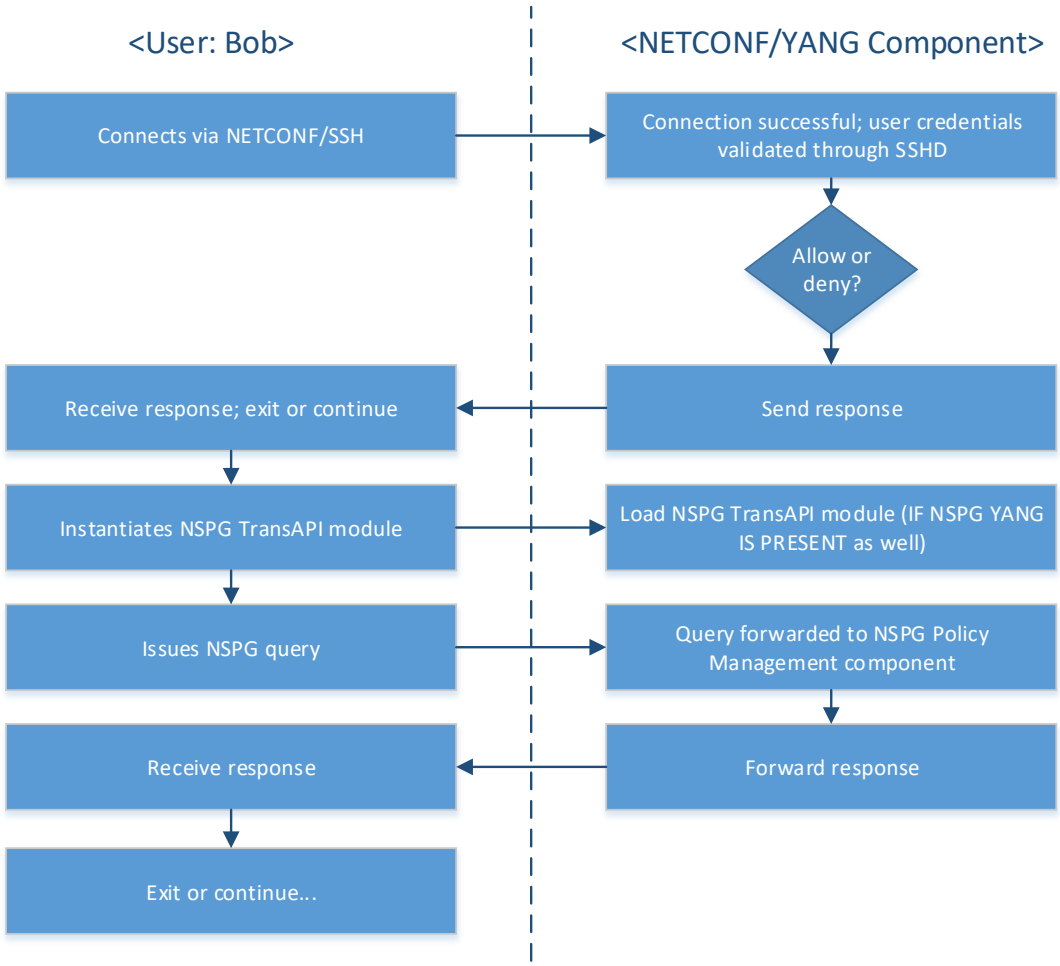

Figure 4.4: NSPG/YANG component swimlane diagram

4.2.2 Policy and Authentication Management

In order to map SNMP commands such as GET and SET to NETCONF commands, the NSPG requires at least one policy to be configured for communication with a legacy SNMP host. The NSPG upon initialization reads an aggregate XML configuration file containing one or more host policies and creates a dynamic data structure in memory (see Figure 4.5 for a skeleton example). The data structure is used to poll legacy devices given specific stored procedures and store any SNMP responses in the appropriate related element.

```
<host>
  <uuid>01be2dc8-689d-4e78-9926-d0ceb7e542fe</uuid>
  <address>127.0.0.1</address>
  <security>
    <version>2c</version>
    <user/>
    <community>public</community>
    <authpass/>
    <authmethod/>
    <privpass/>
    <privmethod/>
  </security>
  <task>
    <op>
      <type>get</type>
      <oid>1.3.6.1.2.1.1.5.0</oid>
      <format>string</format>
      <data/>
    </op>
  </task>
</host>
```

Figure 4.5: Minimal configuration depicting an NSPG policy XML element

Overall, the policy and authentication management component is not directly responsible for the initial authentication process (this is provided by the NETCONF/YANG component), but is augmented by further validating the username provided by queries generated by the NSPG NETCONF component TransAPI module. When a query is received, the policy and authentication management component then returns only the policies where UUIDs match OS known groups (scrubbed of the

"-" delimiter) assigned to the validated username provided by the original query.

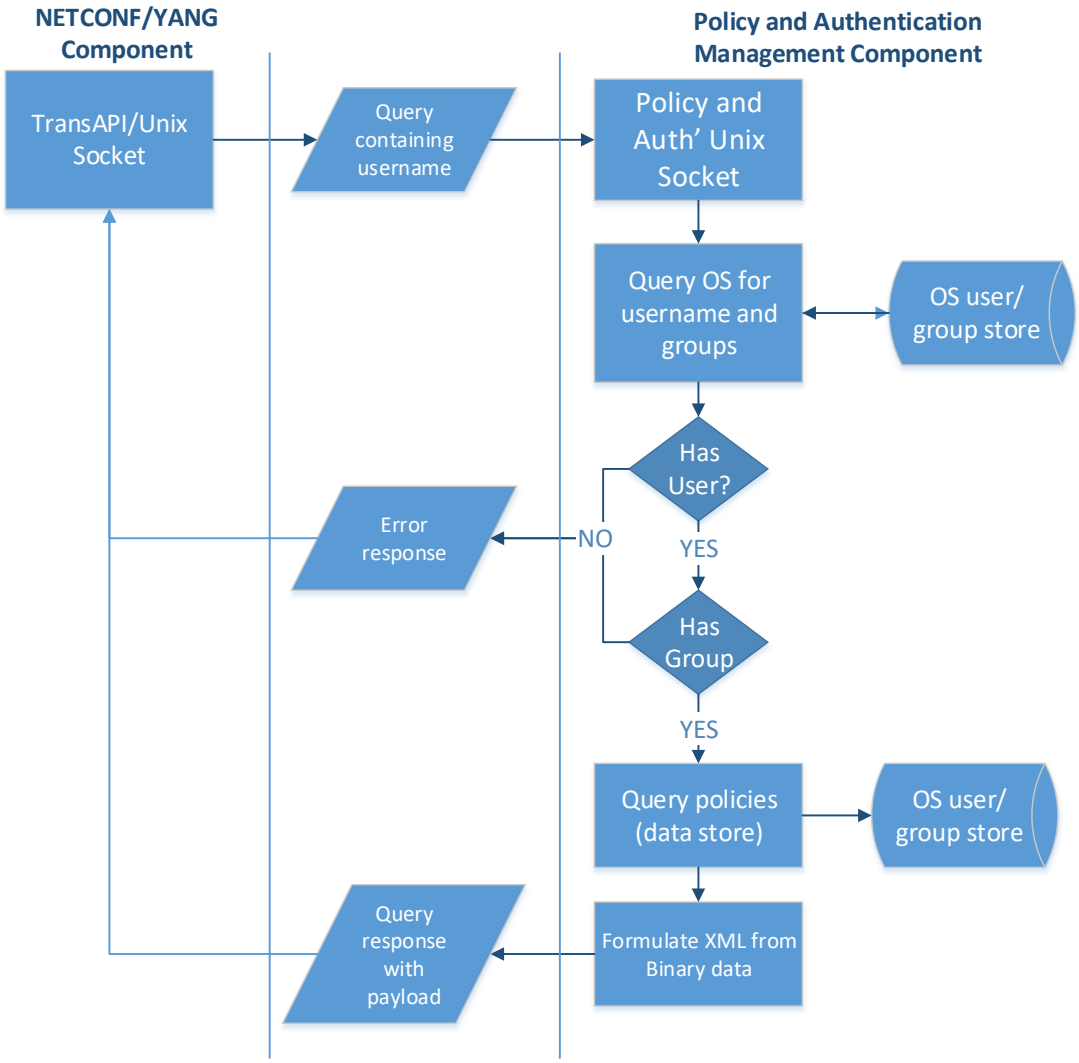

Figure 4.6: Policy and authentication management flow/mapping diagram

If the policy and authentication management component (or NSPG if we are referring to it as a whole) is provided a valid user name within a query and there IS a matching UUID policy to group name, a response such as that shown in Figure 4.7 will be returned:

```
yangcli bob@192.168.10.100> performSingleOP nspg-uuid= \
ff994a62-f6d5-453d-912c-104431f1c100

Warning: 'rpc-reply' has no child node 'data'. Using anyxml
RPC Data Reply 4 for session 1:

rpc-reply {
  data '<?xml version="1.0"?>
  <config>
  <host>
  <uuid>ff994a62f6d5453d912c104431f1c100</uuid>
  ...
  <task>
    <op>
      <type>get</type>
      <oid>1.3.6.1.2.1.1.5.0</oid>
      <format>string</format>
      <data>myoldname.localdomain</data>
    </op>
    <op>
      ...
    </op>
  </task>
  </host>
  </config>'
}
```

Figure 4.7: Response from the Policy and Authentication component containing only a single policy

If the policy and authentication management component (or NSPG if we are referring to it as a whole) is is provided a valid user name within a query (given that a successful connection has occurred) and a matching UUID policy is unable to be matched to a group name, an error message similar to the following will be returned:

```
yangcli bob@192.168.10.100> performSingleOP nspg-uuid= \
ff994a62-f6d5-453d-912c-104431f1cfff

Warning: 'rpc-reply' has no child node 'data'. Using anyxml
RPC Data Reply 3 for session 1:

rpc-reply {
  data '<?xml version="1.0"?>
    <config>
      <host>
        <error>No policy found</error>
      </host>
    </config>'
}
```

Figure 4.8: Error message from the Policy and Authentication component

4.2.3 Cache/Data Store

The cache/data store component manages several aspects that are controlled dynamically by the NSPG's configuration data. When the application is initialized, a single linked-list data structure is created in memory in byte form for improved efficiency and lower memory consumption over using purely ASCII XML. The cache functionality is guarded by the policy and authentication management component (see Section 4.2.2), but when authorized access occurs, a seamless transfer of SNMP to NETCONF occurs without the need to fetch data directly from legacy hosts; data is buffered in the data structure and forwarded upon user query.

```
typedef struct snmp_op_s {
  int type;
  char format;
  char oid[MAX_OID_LEN];
  char data[MAX_VAL];
} snmp_op_t;

typedef struct snmp_policy_s {
  char uuid[MAX_UUID_LEN];
  char ip_address[MAX_IP_LEN];
  uint8_t security_version;
  char user_name[MAX_VAL];
  char community_name[MAX_VAL];
  char auth_pass[MAX_VAL];
  char auth_method[SMAL_VAL];
  char priv_pass[MAX_VAL];
  char priv_method[SMAL_VAL];
  int num_of_tasks;
  snmp_op_t tasks[MAX_TASKS];
} snmp_policy_t;
```

Figure 4.9: C typedefs of policy structures contained in the data store

The data structure is used not only to determine the actions to be performed by the SNMP master component, but also to cache returned values from SNMP commands executed from the stored policies regardless of SNMP version. If functionality such as a SNMP community name is not necessary (as is the case for SNMPv3), then associated fields will be NULL (contain NULL bytes). In addition to simply storing information, there are also two other critical pieces of functionality in the cache/data store component:

- A polling timer used to update the data store by executing the SNMP master component on a routine basis
- A UNIX domain socket listener (server) to accept and process incoming connections from

the NETCONF component. This listener also handles access to the data store and forwards the appropriate data back to the requesting UNIX domain socket client within the NETCONF component.

Unfortunately, raw data in binary form contained in the data store is not optimal for parsing by third-party applications such as a NMS or human-readable output. Therefore stored data to be sent back as a response requires conversion back to XML upon query.

4.2.4 SNMP Master

For the data store and caching of data to be useful, it is routinely updated by a SNMP aware master at specific intervals signaled by a timer. Upon this signal, the master executes SNMP operations according to the directives contained within the data store and updates the shared result for each SNMP operation. This is synchronized such that concurrent access will not cause an error condition to be generated within the NSPG.

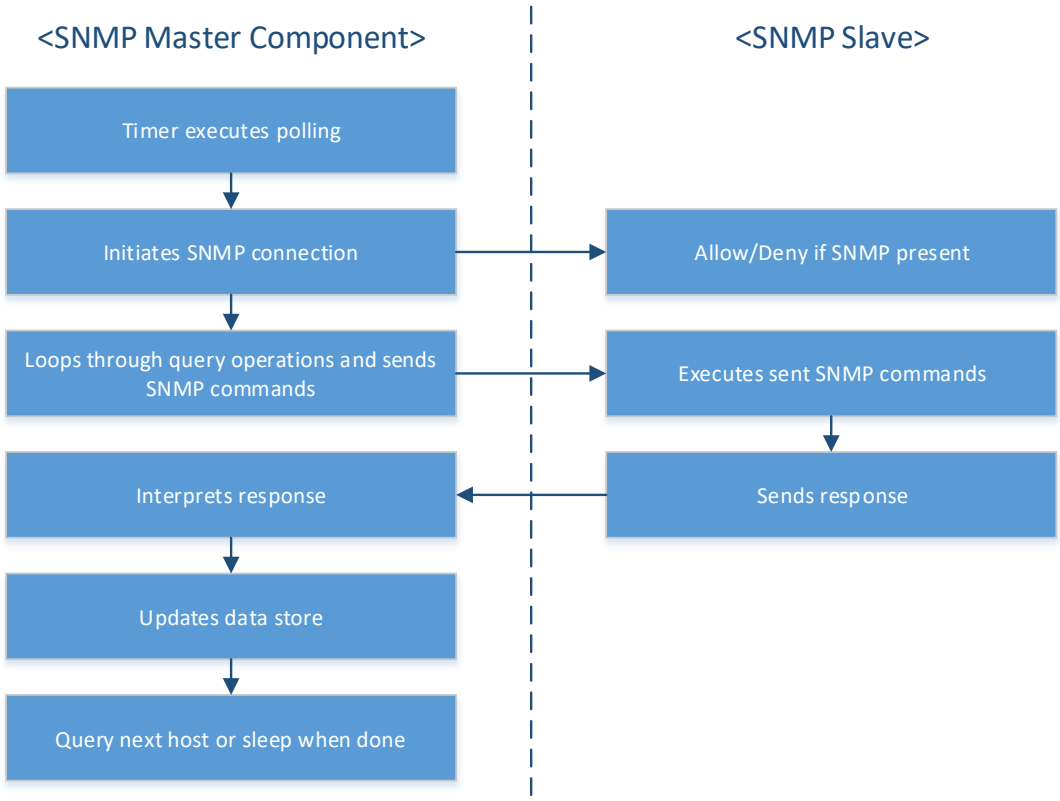

Figure 4.10: SNMP Master component swimlane diagram

Currently, the SNMP master component only supports SNMP GET and SET commands for versions: v1, v2c and v3. It supports the various SNMP authentication mechanisms including: community strings, SHA1/MD5 authentication hashes and AES/3DES encryption. It also prevents upstream hosts from communicating directly with downstream legacy hosts and only valid SNMP traffic can be generated assuming other hosts downstream are not communicating with a host other than the NSPG using SNMP.

4.3 Novelty

The NSPG is novel in the following ways:

- Supports both NETCONF/YANG and SNMP (all versions) (Enns, 2006)

- Isolates directional dependencies when translating protocols. Neither upstream or downstream "know" they are communicating indirectly to devices that do not "speak" the same protocol
- Secures upstream connections by using NETCONF over SSH while preventing direct access to legacy devices
- Isolates upstream communication method. Solution is agnostic to the actual method used by the upstream NETCONF connection: CLI or NMS can be used
- Translates NETCONF commands seamlessly to SNMP while isolated from the NETCONF component
- Glues together an existing agent using a transAPI module demonstrating that this approach could be applied to other agents supporting the transAPI (Bjorklund, 2010)
- Caches data and performs protocol gateway functionality. This is not normally performed in such a deployment and can be further extended
- Adds additional granularity to access controls when not possible due to device or existing legacy protocol constraints
- Uses Linux users and groups for access control, which can be easily extended by network administrators familiar with managing Linux hosts
- Implements policies that are mapped to the authenticating user using Linux users and groups using 32 byte UUIDs
- Supports legacy devices that cannot be upgraded and even those that could be upgraded to utilize SNMPv3 security controls
- Extensible approach can be applied to other protocols
- Offers flexible deployment strategies: termination end-point, BitW and as a BitS.

Chapter 5

Testing and Validation

As is described in the problem statement, we believe that securing legacy technologies in critical/sensitive industrial environments can be difficult and potentially insecure using the common dual-stack approach. The goal of our research is to examine the superiority of the proposed solution (NSPG) over a traditional dual-stack approach from a security perspective. Before expanding on the experimental methodology, environment and results, let us first explore a common network scenario that we believe accurately represents a common industrial network topology.

Imagine ABC Corp, a multinational company that has acquired several production facilities of various ages in deployment. One of these facilities, a pulp-mill, is running on very sensitive operating margins and any changes to infrastructure or operation can greatly affect both productivity and financial feasibility. It is a large facility spanning several subnets with many industrial networked IP devices all connected over a routed network with a single centralized place of monitoring and configuration. The networking infrastructure is a standard enterprise/industrial one that uses primarily stateful firewalls without DPI for fear of DPI affecting industrial control system (ICS) network communications.

Extending this example, pulp mills can be subject to numerous hazards, but one in particular is the combustion of sawdust materials stored in hill-sized piles. In order to be compliant with new fire regulations and improve plant safety, the plant management installed a new fire-suppression/monitoring system to work with their still working, but antiquated temperature-based equipment that utilizes several antiquated Programmable Logic Controllers (PLCs). This system

uses a new network configuration protocol, but needs to work alongside the existing deployed legacy devices in the same cabinet.

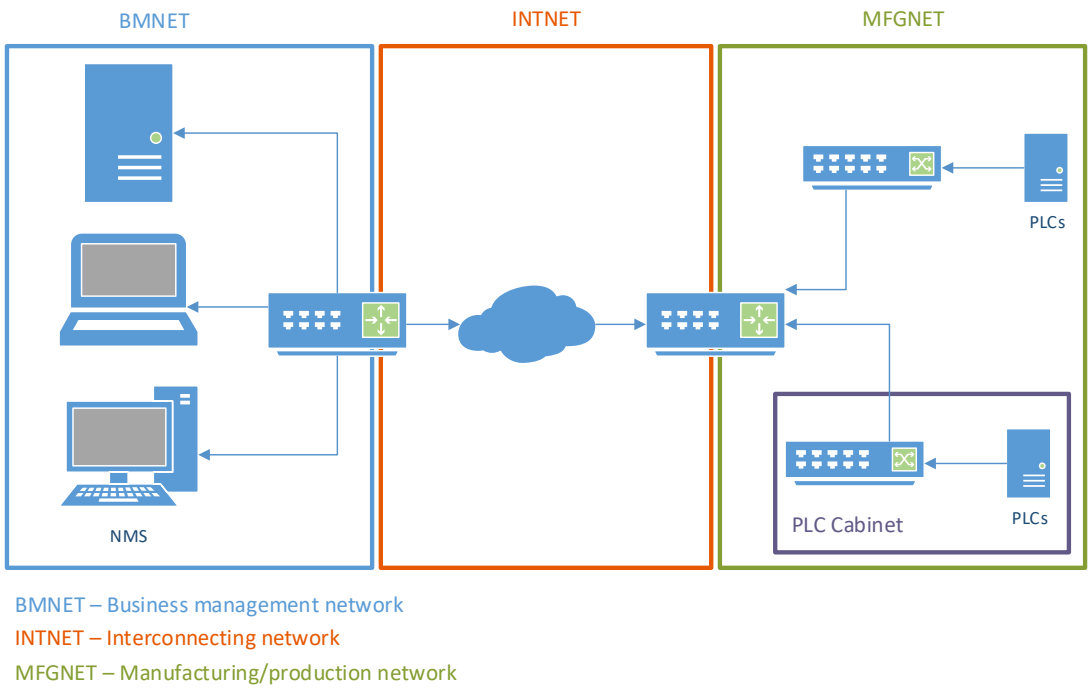

Figure 5.1: Overview of larger model industrial plant network

Given the limited resources at the facility (for economic reasons), Jane the main plant network administrator is charged with keeping things "merely running at the lowest cost/effort possible". To keep management happy by not adding any new costs or delay, she just aims to "make it work" by ensuring that access is possible from her central workstation to the end device cabinet. She does this simply by adding firewall ACL rules to each intermediate device such that access to the monitored devices over both protocols is possible.

Additionally, what if Jane had to traverse multiple networks such as an ICS network (where the industrial processes are networked and controlled) and the business management network (where daily operations are performed) without a secure mechanism to keep traffic safe from prying eyes or other malicious purposes? Jane could use VPN technologies to secure traffic, but is unlikely to use a VPN within her own network (although this can be a great idea) due to additional overhead in

maintaining a VPN infrastructure. If she doesn't use a VPN, she will continue to communicate over all networks (including the interconnecting network) using both protocols even if one is insecure. Alternatively, even if she did not communicate with the legacy devices using the legacy protocol (ACL rules installed), the firewall ACL rules she had created still exist with the potential to be bypassed given the protocol's underlying mechanisms (connection-oriented or connectionless) and a malicious user or malware. Furthermore, even if traffic is legitimate or spoofed using valid IP addresses, policy violating traffic and malformed/unsupported application layer data may not be intercepted to be enforced.

For our experimental work, the large-scale topology presented above in Figure 5.1 is too complex (and costly) to mimic in its entirety. We therefore chose a subset of the features and devices, which we believe is still representative of the issues we need to test. This model consists of an upstream NMS connecting to a smaller cabinet of legacy devices. Since we do not have access to industrial PLCs, we modelled our network as a small subnet consisting of three low-end microcomputers (which mimic computing resources in the real-world more or less) connected together by an unmanaged switch, protected by a single simple stateful firewall that is positioned between an operator on the uplink (upstream) and the small switch on the downlink (downstream).

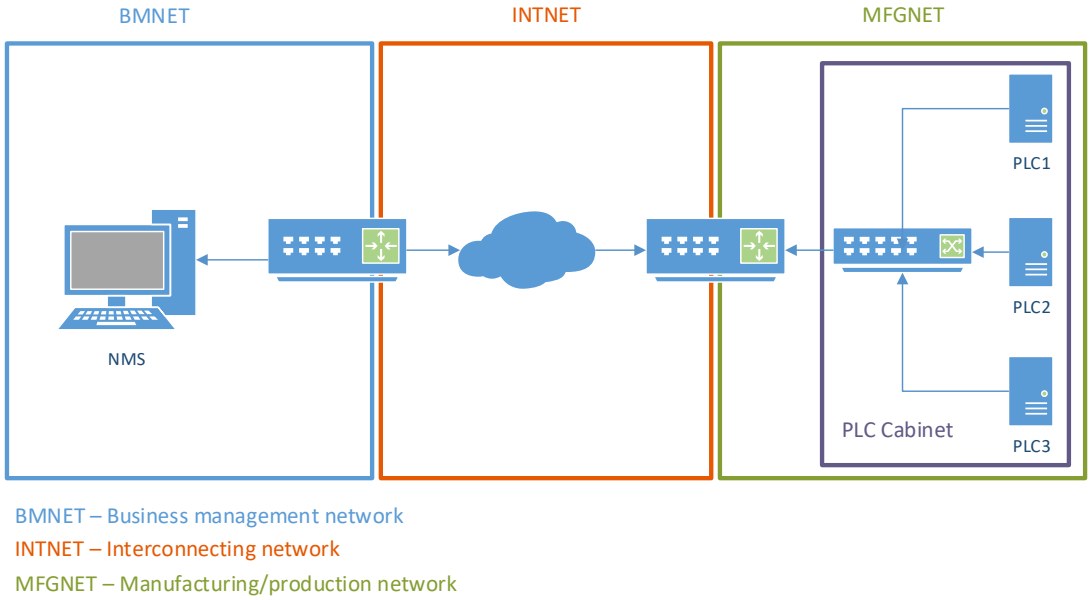

Figure 5.2: Simplifying the subset model network to demonstrate the path from NMS to legacy devices in a PLC cabinet

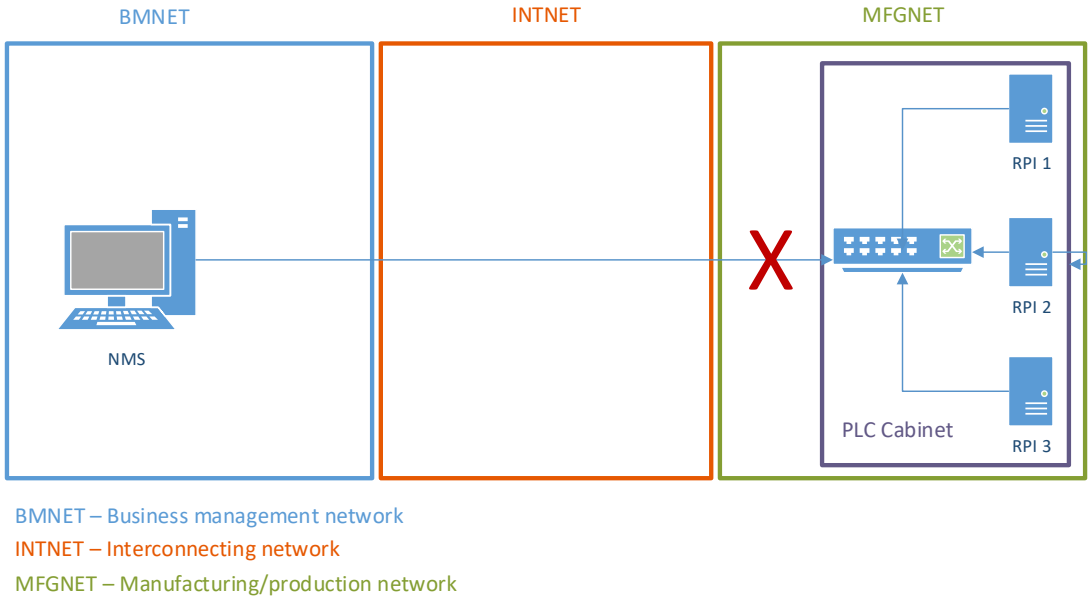

Figure 5.3: Sub-model network topology demonstrating security appliance location

Using a subset of the model network in Figure 5.1, the network in Figure 5.2 allows us to effectively limit the scope of any testing or validation to a single zone that would be seen in industrial networks. With a defined scope, this network acts as a baseline environment. In Figure 5.3, the midstream point (marked X) represents the location that Jane our administrator would place a network security appliance, which can be exchanged depending on the approach being tested; dual-stack or NSPG. This improves testing repeatability and reliability, and standardizes any test automation, network traffic generation or lower-level technical inspections (see Section 5.1.2 for more information). Additionally, the standardized upstream and downstream interfaces through the midstream point also allow us to examine any approaches (even though they may differ) in a controlled way for threat modelling (see Section 5.1.2 for more information).

Now that the network topology has been established at a high level (see Section 5.1.1 for detailed information on exact configurations), the generalized testing methodology to explore the proposed solution involves testing the dual-stack approach using our sub-model network to derive a base-line threat model supported by lower-level details such as performance and packet/ACL inspections. After the baseline has been determined, the same tests (where applicable given that it is a different approach: minimally filtered pass-through vs. Proxy-hybrid) were run to generate a threat model using the same approach used to create the base-line's threat model. Once both models have been derived, then we examine the results to compare and contrast both approaches to infer the advantages or disadvantages of the proposed solution base-line to make our recommendations.

The remainder of the chapter is organized as follows:

- **Testing Environment and Methodology** – describes how the network is configured and the testing methodologies used
- **Verification of Testing** – describes the models obtained with supporting details from secondary network tests
- **Testing Insights** – summarizes insights generated from testing

5.1 Testing Environment and Methodology

This section describes how the high-level methodology including the testing environment was configured and used. For ease of understanding the results of all testing, this section is dedicated to describing the sub-model mentioned in the previous section and how a testing methodology was applied to the two approaches used within it.

For readability, this chapter is organized into two sections:

- **Testing Environment** – describes how the environment is configured for both approaches
- **Testing Methodology** – describes how both approaches are tested within the testing environment to derive threat models supported with lower level tests/details

5.1.1 Testing Environment

Testing both approaches requires a network that reasonably mimics a sub-section of an industrial network. In continuation of using a small section of what we deem to be a real network, the network consists of four components regardless of implementation:

- **Upstream** – Where an operator or in this case, the researcher, would initiate any network operations directed at downstream devices
- **Midstream** – Where both approaches would be situated one at a time to be tested
- **Downstream** – Where legacy devices to be communicated with are located
- **Inspection points** – Where network traffic can be sniffed or inspected

Together, these components are used to explore the network topology of a pulp mill undergoing a safety refit (mentioned in Section 5) where a branch of an industrial network containing legacy devices is to be communicated with from an upstream connection through a traditional stateful firewall device.

In Figure 5.4, the three network segments (business, interconnecting and manufacturing) are still present, but the X (marked in red) indicates where a security appliance may be deployed to protect the devices contained within the PLC cabinet.

Given the placement of the security appliance, Figure 5.4 illustrates a simplified version of the network topology and marks two locations where network traffic needs to inspected to verify the effectiveness of the solution(s) being tested:

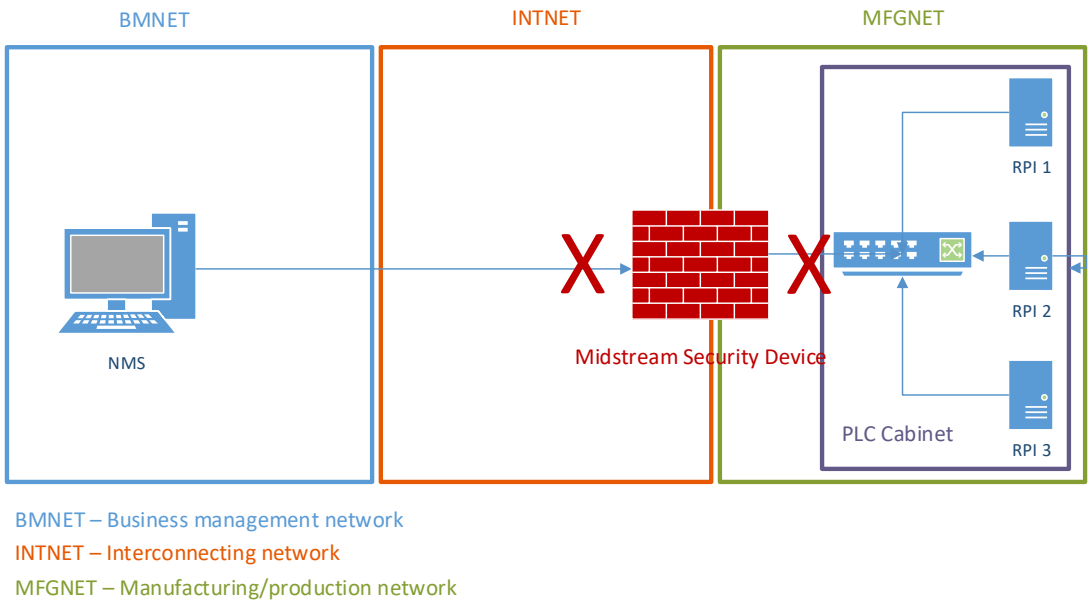

Figure 5.4: Inspection points on sub-model network topology

From the diagram, the network between the midstream point and downstream is considered secure. Conversely, the network on the upstream is considered insecure.

To maintain the industrial theme, it is important to note that industrial devices operate under different constraints from that of military, enterprise and consumer hardware. Industrial devices are often very proprietary, but are targeted for high-performance, low-latency input/output (I/O) operations and to operate continuously for years. They are highly tested for their deployment environments, but operators/engineers are often very reluctant to make changes to these devices and attached processes once deployed. Additionally, these devices are often very sensitive even if a vulnerability or fault is not present; this adds additional pressures when making changes to networks and securing these types of devices (Markovic-Petrovic & Stojanovic, 2013). For example, these devices often have less than modest hardware when compared to today's modern smart phone processors and if a legitimate port scan takes place, the device may suffer a DOS or crash, severely

interrupting the productivity and safety of an industrial process.

While industrial devices are often sub-300 MHz, have less than 128 MB of RAM and have 10/100 Mb Ethernet network connectivity, we chose to represent the downstream devices using the low-cost RaspberryPi B+ platform to attempt to mirror industrial legacy device hardware. (See Appendix A.4 for more information on hardware.) The RaspberryPiB+ v2 has a 700MHz CPU (which can be down clocked), 512 MB of RAM and 10/100 Ethernet (RaspberryPi Foundation, 2016a). The constraint of using devices that are 10/100 Mbit instead of 10/100/1000 Mbit (Gigabit) Ethernet is not problematic when modelling current industrial networks (Stubbs, 2013)(NXP, 2011). The RaspberryPi besides being slightly more powerful than some real-world devices, uses a fully-fledged Linux OS tailored to RaspberryPis called Raspbian (RaspberryPi Foundation, 2016b). This creates a flexible and modestly capable test environment able to host both SNMP and NETCONF services.

For the midstream component, the RaspberryPi B+ platform was again chosen to host a Linux OS and be used for both dual-stack and NSPG approaches. The reasoning behind this choice is three-fold: the RaspberryPi platform is inexpensive for prototyping, modest hardware resources should highlight any potential performance bottlenecks and finally, if security can be deployed ubiquitously, then low-cost hardware similar to a RaspberryPi could be used. On this host, both approaches (dual-stack and NSPG) can be applied independently of one another for all tests.

The upstream device used to communicate to the downstream devices will be a device/system without the performance restrictions as the mocked up legacy devices. It is assumed that Jane the administrator would often be executing tasks from a system with greater resources than a PLC using commodity enterprise hardware such as a mobile workstation or desktop/server running a full-fledged OS such as MS Windows or Linux. Given this assumption, the upstream host has above average performance, running Linux with a full suite of tools for packet capture, packet generation and utilities to perform both SNMP and NETCONF configuration on end devices (see Appendix A.4 for more information on upstream device hardware).

In order to passively collect and verify network traffic passing between upstream, midstream and downstream hosts, inspection points are used. The most basic methods to inspect traffic use either a specialized device called a network tap or a router/switch spanning port combined with software

that is capable capturing (sniffing) and decoding packets at a desired physical location. For the purpose of this research, we used a multi-homed sniffing configuration similar to a "network tap" or a "bridge" except that traffic is routed through USB Ethernet adapters and captured using packet sniffing tools for analysis as depicted in the high level Figure 5.4 at the marked inspection points. For further detail, Figure 5.5 illustrates the physical network and where the interfaces were added to create low cost sniffing bridges to actually inspect the traffic.

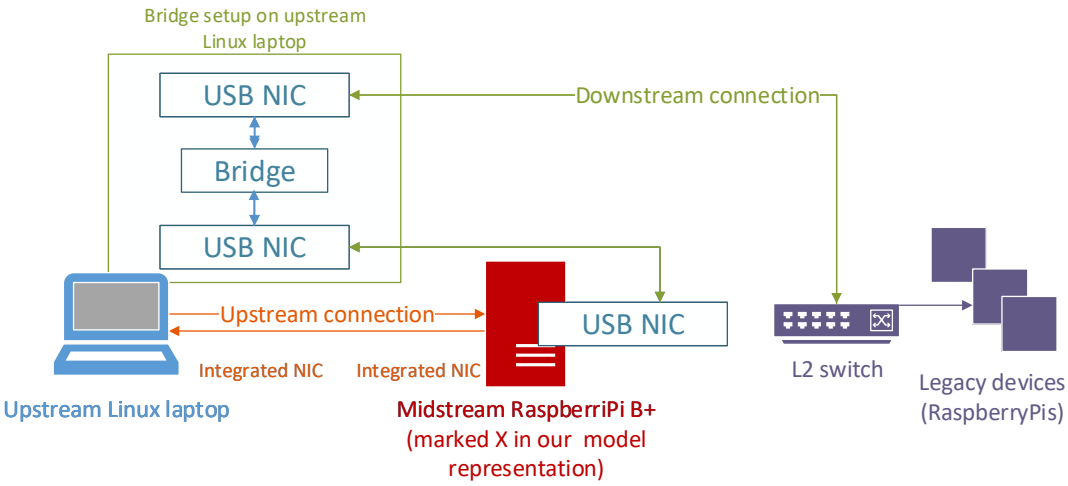

Figure 5.5: Test packet sniffing layout

It is important to note that when using non-hardware capture methods and USB Ethernet adapters, latency can be an issue for real-world process control and the effect should be considered when creating a real-world implementation. However, for the purposes of this research, real-world control systems were not used and latency does not affect tasks such as configuration or monitoring, beach such tasks have no real-time constraints.

Dual-stack Approach

The dual-stack approach is likely the first approach an engineer or network technician would deploy to secure or isolate a group of downstream legacy devices. Before assuming this to be fact, let us explore why this might be the case using a simple real-world example (albeit slightly contrived):

Jane the administrator has a rack of industrial devices she needs to isolate from a network or limit connections to said devices. Her first solution is probably going to utilize a mature technology called a stateful packet filtering device (a firewall) that can enforce ACLs. Enforcing strict ACLs may not be a primary concern for Jane, but avoiding minimal network disruptions while accomplishing the goal of segmenting these devices away from other networks might be the most important. Therefore, Jane is required to get management approval, buy an appropriate firewall and deploy the firewall during a maintenance period (likely after testing it extensively in a dedicated test facility). Jane installs and configures the firewall ACLs for specific industrial protocols, verifies that connections traverse the firewall successfully and deems that everything is considered satisfactory with little additional afterthought except periodic checks, which will fade away if no errors surface.

From this example, Jane implemented a dual-stack approach. It is a low-risk approach, offers a potential degree of additional security for the end devices, minimal work required for completion and *it worked*. For these reasons, this approach is considered a de-facto standard for securing networks and was chosen to be explored by this research as the base-line approach. Within the context of the test environment described in Section 5.1.1, the midstream component is configured to support a dual-stack approach such that two protocols (SNMP and NETCONF) are allowed to traverse the firewall from upstream to downstream. As the packets traverse the firewall, packets are captured on both sides of the midstream device independently so as to not affect performance results. The number of firewall rules and the resource consumption are also recorded.

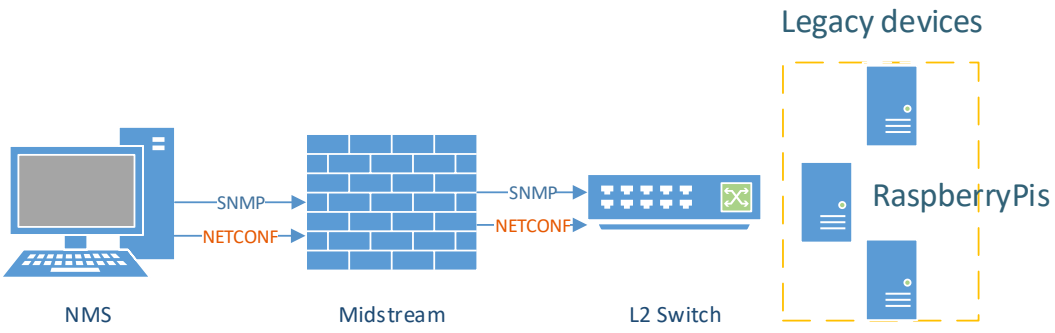

Figure 5.6: Dual-stack testing network topology

In the test environment, the dual-stack configuration is set up and run as follows (for a detailed technical description of the topology see Appendix A.4):

(1) Verify network physical network connectivity (e.g., is there a physical link established?)

(2) Flush all firewall ACLs, packet counters and ready packet capturing applications

(3) Install the necessary firewall rules and bridge configurations

(4) Verify network connectivity is possible through the firewall and flush packet counters once verified (e.g., was X protocol able to traverse correctly)

(5) Perform testing and save any recordings once complete

NSPG Approach

There is a common phrase that describes a task with many ways to achieve completion, this phrase is "there are many ways to skin a cat". Networking is one realm where this phase applies because there are often many techniques that can be used or mixed-and-matched to achieve the desired effect. Each approach has advantages and disadvantages individually or when combined with others. However, the effect may or may not reach an operator's expectations or goals. In continuation of Jane the administrator, let us explore a potential use case of the NSPG approach and how it is used in the test environment.

Jane has a mandate from higher levels of management to implement security for downstream legacy devices, but has been given more budget and time to consider all possible approaches to securely deploy her SNMP and NETCONF devices. She performs an evaluation of current technologies and literature to define a list of several approaches, but needs to match them to a more specific set of criteria other than simply isolating devices using a firewall and ACLs. Jane's requirements indicate that a firewall that only inspects traffic at the layers lower than the application layers and does not convert protocols from one to another may not be sufficient. Additionally, Jane knows that traditional information technology (IT) enterprise approaches support DPI and typical business needs, but often are not tailored to industrial network constraints such as specific industrial protocols or environmental operation characteristics (e.g., temperature and mean-time-between-failures

(MTBF)). Therefore Jane needs to select a vendor that has experience in industrial environments and an approach that fits her environment:

(1) She has devices that are unable to be upgraded/modified to support a newer technology

(2) She requires security controls/mechanisms that are not present in the legacy protocols used by the legacy devices

While we do not have access to industrial environment approved hardware for an NSPG platform, we can use a RaspberryPi and the NSPG to act as a proxy/gateway hybrid that is different from a pass-through styled approach. Neither the upstream nor the downstream configurations change. However, the upstream connectivity to downstream hosts is now interrupted by the NSPG. Once the traffic is interrupted, more fine-grained policies can be enforced by mapping requests to authenticated users and groups in a way that prevents the original requests from having direct access to the legacy devices. Secondly, legacy access via SNMP is now not possible at all because SNMP is disabled and all operator activities must use NETCONF and the NSPG solution without modifying the end devices. For Jane, she aims to leave the legacy devices intact (without modification), while providing additional access controls and improved security for the legacy devices. Given these criteria, the NSPG approach can at least be suggested for testing for validity in her application.

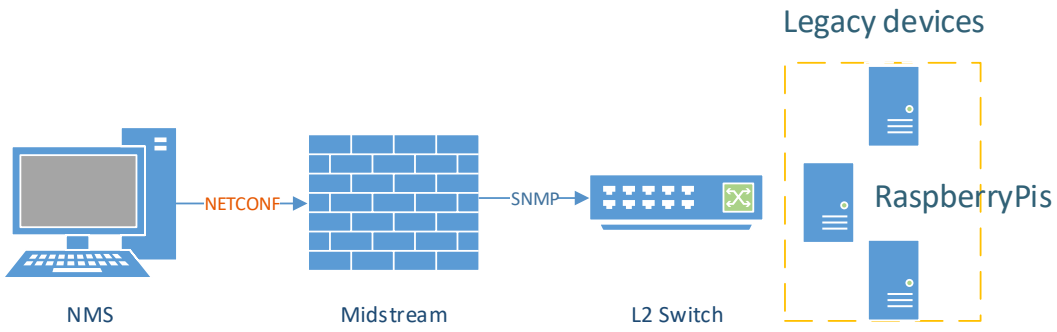

Figure 5.7: NSPG testing network topology

Using the same mid-stream component as the dual-stack approach, we replace the dual-stack

with the NSPG and utilize a derived configuration from elements that an administrator may want to routinely access. Starting with two primary actions (GET or SET), we can achieve most configuration/monitoring functionality except setting primitives such as event notifications. Using either GET or SET operations, we can determine a set of variables (SNMP OIDs in this case) specific to each end device to be implemented in a policy (where there could be more than one per device). These policies are then mapped to a user (an operator in this case) and are used to allow/deny queries according to the assigned policies. If policies do not exist or queries are made outside of the values carried inside the query, access is denied. The example in Figure 5.8 conveys the intent of the XML that needs to be written for a policy.

```
device X @ IP 192.168.1.10
> is SNMPv2
> uses private community string "thisIsTheKeyToKingdom"
> has policy 1234567890
   > policy 1234567890 allows a GET request for OID 987753
   > policy 1234567890 returns a value of string type
```

Figure 5.8: Example of a pseudo configuration for a legacy device

While the paragraph in Figure 5.8 may be considered quite technical for a non-engineer (assuming that an engineer should be able to determine OIDs and host information), once the NSPG is configured, the implementation is seamless for a user. For example, Jane the administrator wants to access device B's configuration to execute a specific property on that device. Jane connects to the NSPG via an exposed IP address and port through the YANGCLI application, queries a specific UUID for the policy associated with his desired device and is returned structured XML. Alternatively, another user if he/she manages to authenticate with the NSPG, can query the same UUID and be denied access. Either way, the upstream connection does not know that the legacy devices do not "speak NETCONF" and that it is eventually speaking to legacy devices. Conversely, the legacy devices do not "know" that the NSPG is bilingual as well.

The NSPG is set up and run as follows (for a detailed technical description of the topology see Appendix A.3):

(1) Verify network physical network connectivity (e.g., is there a physical link established?)

(2) Flush all firewall ACLs, packet counters and ready packet capturing applications

(3) Install the necessary firewall rules, the NSPG and necessary configuration

(4) Verify network connectivity is possible through the NSPG by using a NETCONF/YANG CLI from the upstream and flush packet counters once verified (e.g., was able to access X OID on device A)

(5) Test and save all recordings/logs/statistics once complete

5.1.2 Testing Methodology

This section describes the testing methodology for both approaches and how the test environment described in Section 5.1.1 is utilized. Now relating back to the problem statement and proposed solution, the goal of our testing methodology is to explore the security of the base-line solution (dual-stack) and compare it to the security of the proposed solution (NSPG) to see if recommendations can be made to improve the security of legacy devices. To derive the final recommendations, we will:

(1) Use the test environment for both approaches to observe network operations at the packet and host levels (for more information on Threat Models, see Section 5.1.2)

(2) Create a threat model for each approach and strengthen/adjust them using the observations obtained in the network tests

(3) Derive a table of advantages and disadvantages derived from the threat models in the previous steps to discern whether either approach truly is beneficial and in what situations if any, does this occur

The remainder of the section is dedicated to outlining the test methods used and examples of how they are used to create the final outcomes of this research:

- **Network and Resource Consumption Tests** - describe the generation and inspection of network traffic that has been created for the purposes of evaluating both approaches. It also includes monitoring resource consumption as supporting details to be used to refine the threat models

- **Threat Models** - describes a specific approach to evaluating a system or set of components such that a comprehensive model showing potential threats to a system can be enumerated. In addition to discovery of threats, boundaries and data flows, a report of threats and mitigations can be generated

Network and Resource Consumption Tests

Network and resource consumption tests can represent ambiguous terminology connoting a multitude of metrics to describe a solution's efficiency often for marketing or standards compliance. For example, network tests can be used simply as indicators of why X feature is great or when ABC product is better than XYZ product. In this research, network and resource consumption tests have the goal to support and refine claims generated by the threat models in Section 5.1.2:

(1) **Network layer** - creating/generating packet flows from upstream to downstream for network operations that could be generated by a human operator, injecting known malicious packets that could bypass an ACL if not tightly filtered and capturing/verifying/inspecting that packets are transmitted on both upstream and downstream sides of the midstream device

(2) **Resource consumption** - simply is the recording of the midstream hosts performance for both dual-stack and NSPG approaches. This includes CPU utilization, packets per second, and memory consumption

The first step in using the network tests was to perform a sanity check once the solution being tested was deployed. This involved verifying the correct network behaviours (depending on the solution deployed) such as whether packets are dropped appropriately, and verifying if they flowed from point A on the upstream to point B on the downstream through the midstream component (dual-stack or NSPG) as expected. This is corroborated by using two diagnostic tools:

- **Tcpdump** - is a CLI utility that is used from a CLI to passively sniff packets as they go across an interface for recording (to a file or CLI output) and decode them so they are human readable (e.g., parsing raw byte stream to show IP addresses, TCP ports and ASCII payloads) (*TCPDUMP*, 2016). It is a light-weight tool that can be used on any *NIX system. Figure 5.9 demonstrates an example capture showing local multicast traffic

- **Wireshark** - is a graphical tool that extends the basic functionality of tcpdump, but offers more advanced functionality oriented to decoding various protocols, following network streams and troubleshooting network-related issues such as performance bottlenecks or malformed packets (Riverbend, 2016). It is heavier-weight compared to tcpdump and requires a graphical window manager. Figure 5.10 demonstrates an example capture showing DNS packets

```
rbrash@ackbar:~$ sudo tcpdump -i wlan0
[sudo] password for rbrash:
tcpdump: verbose output suppressed, use -v or -vv for full \
 protocol decode listening on wlan0, link-type EN10MB \
 (Ethernet), capture size 262144 bytes
14:41:59 IP ackbar.lan.19942 > RouterPro.lan.domain: \
 43879+ PTR? 203.15.168.192.in-addr.arpa. (45)
14:41:59 IP RouterPro.lan.domain > ackbar.lan.19942: 43879* \
 1/0/0 PTR goliath.lan. (70)
14:41:59 IP ackbar.lan.36734 > RouterPro.lan.domain: 19072+ \
 PTR? 237.15.168.192.in-addr.arpa. (45)
14:41:59 IP RouterPro.lan.domain > ackbar.lan.36734: 19072* \
 1/0/0 PTR ackbar.lan. (69)
```

Figure 5.9: Example of tcpdump output

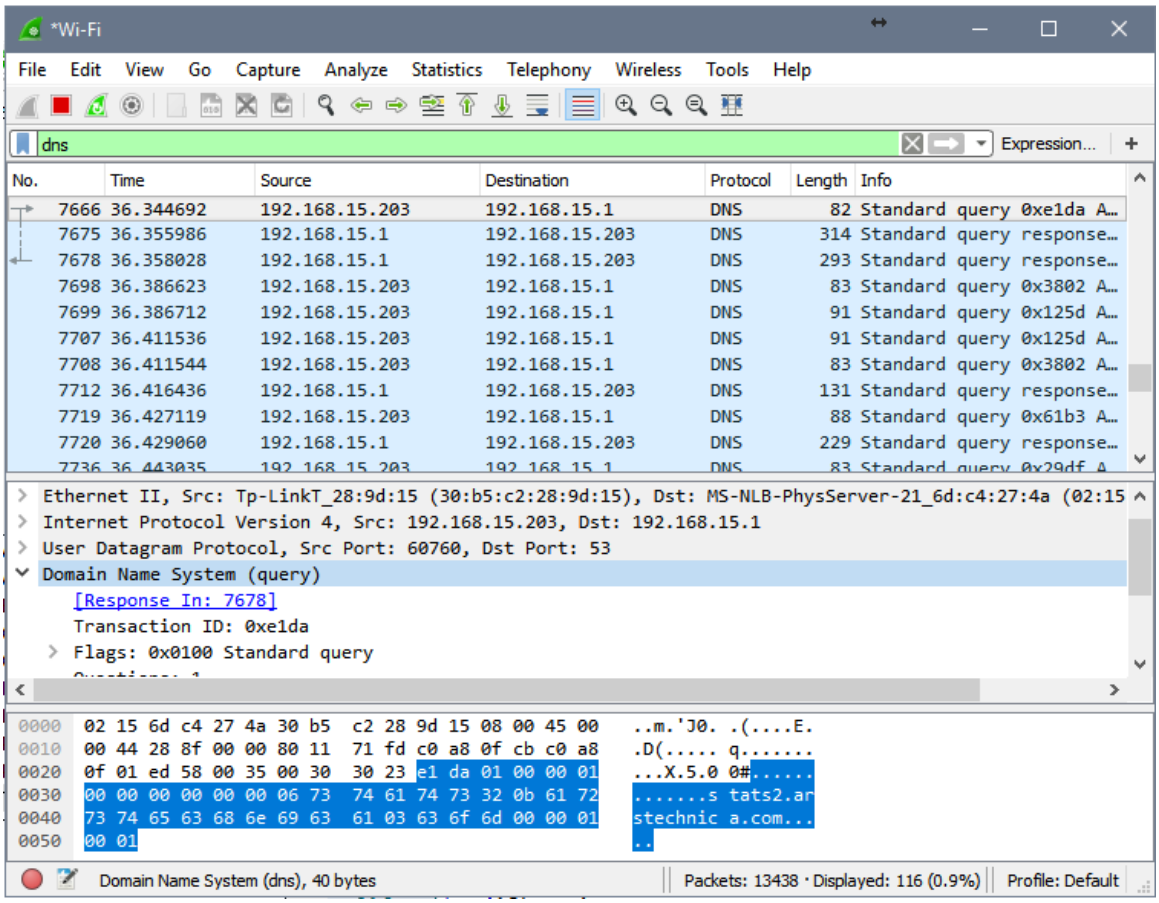

Figure 5.10: Example of wireshark output

To capture traffic before and after the midstream component, a relatively powerful Linux machine sniffed traffic on both sides as if we had access to a commercial "wiretap" device. A wiretap is a device that allows an administrator or potentially a malicious user to passively sniff network traffic inline without (hopefully) being noticeable in any way. To create our own wiretap, three USB NICs and an integrated NIC were used in a way that traffic was sniffed passively at the upstream using a single NIC, and actively using an inline software network bridge to forward traffic from one interface to another. The later technique involved using a set of Linux utilities called bridge-utils (brctl) and ebtables/iptables (Ethernet bridge tables and IPtables). The sniffing environment used the topology in Figure 5.5 and the packet capture software was placed on the upstream integrated NIC and bridge.

The next step was to generate the set of actions required to get a repeatable network test. An

action is an event such as Jane the administrator connecting to a device via SNMP. This causes a flurry of network traffic corresponding to Jane's action. The idea was to determine these actions in a programmable/scripted way such that they could be duplicated. For both SNMP and NETCONF, a GET/SET operation targeting the downstream hosts was performed from the upstream station. To do this, three utilities (one for generating SNMP operations, one for generating NETCONF queries and one to generate arbitrary packets) were used:

- **Snmp[walk, get, set]** - is a suite of CLI utilities that allows an administrator to execute SNMP operations without a graphical interface (Net-snmp, 2011). Results range from simple successes to formatted human-readable output (it is a binary protocol)

- **Yangcli** - is a CLI utility that converts human readable commands into NETCONF operations described by YANG models and the NETCONF protocol (OpenYuma, 2016). NETCONF's payloads are in ASCII/XML protocol, but are tunnelled through SSH; yangcli performs the series of procedures for establishing an SSH connection and executing NETCONF commands in the correct order/syntax

- **PackEth** - is a GUI and CLI packet generator tool for Ethernet networks (PackETH, 2015). Packets can be created and sent in almost any format or sequence on an Ethernet link. It is a useful tool to verify DPI, ACL depth and arbitrary packet generation/injection.

Using the yangcli, snmp[walk, get, set], and packeth CLI utilities, we manipulated the GET/SET operations for both SNMP and NETCONF using Linux scripting/automation languages: BASH and Expect (GNU Project, 2016)(Libes, 2009). The usage of scripting languages allowed us to reliably execute and parse results as if a human operator were present during testing. It is important to note that scripting the yangcli is more complex than "executing a send command, expect response" protocol. This means that NETCONF requires a series of operations to be performed in order to utilize a feature such as the NSPG TransAPI module.

Additionally, the packeth tests, despite their simplicity, had a slightly more involved logic than simply defining variables to be get/set or executing commands. Packeth required an extra use-case where a single generated packet needed to be completely protocol specific (SNMP in this case),

protocol syntactically valid and query a device specific OID. This was to demonstrate that without DPI, a SNMP packet could be spoofed (including the SNMP credentials) and bypass a firewall ACL to execute functionality on an end device. For this, we chose the OID used by CISCO to enable a configuration reset using a Trivial File Transfer Protocol (TFTP) server (CISCO, 2011). While the CISCO reset functionality/feature is not considered a security exploit, it is a vulnerability or opportunity for an attacker if left exposed or unprotected. This demonstrates that using only stateful firewall ACLs are ineffective when:

- Packets can be spoofed to appear legitimate. Using valid addressing information, spoofed packets can bypass firewall ACLs
- Packets do not have state information (e.g., UDP) and can be injected any time
- Packet contents need to be inspected for granular controls, but bypass the firewall without any verification

Depending on the approach being tested, scripting behaviours were adjusted such that packets/ connections were directed at either the NSPG or the downstream hosts when appropriate. This is because in dual-stack mode, the midstream component forwards traffic in a pass-through configuration. Alternatively, in NSPG mode, the midstream component intercepts NETCONF traffic and does not allow direct access to the downstream hosts.

For more information about the scripts, see Appendix A.7.

Given that injecting packets or causing packets to be generated using upstream does not illustrate a whole picture of the resources consumed regardless of approach used (dual-stack or NSPG), the midstream host required monitoring of system resources. As packets flowed through the midstream component, we captured system operation metrics such as CPU/memory utilization, and packet rates on both upstream and downstream interfaces. This serves to allow us to solidify the threat model where risk of unfettered packet rates is important for the defence of sensitive devices and where minimal hardware resources may be required to deploy a solution (e.g., cost of NSPG is higher than allowed due to high-performance constraints). Monitoring system resources was achieved solely using:

- **Top** - which is a Linux utility to monitor system resources in real-time (Kerrisk, 2016). Output was directed to a file and parsed for event and time frame synchronization. For example, X memory was consumed at 1.0010 seconds of the test

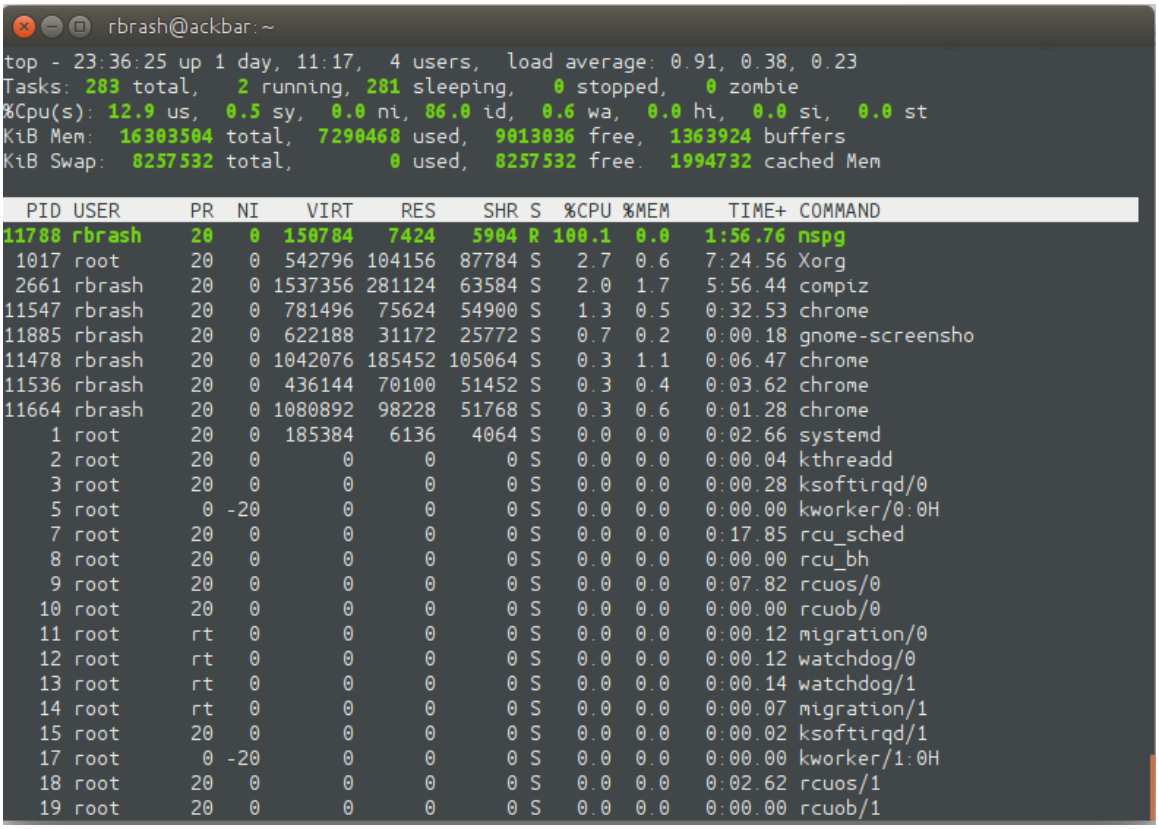

Figure 5.11: Top system monitoring example

The usage of user-space tools and a Linux kernel that was not compiled for near deterministic profiling is an important distinction when measuring performance/resource consumption of an application. Performance and near real-time execution can be near deterministic in the kernel itself, but application level performance is non-deterministic due to OS internal components such as schedulers and parameter tuning. This is different from performing analysis of a hardware circuit capable of pumping X bits per second to a peripheral chip over Z bus. Evaluating the resource consumption of a kernel component or a user space application is also different because specific kernel internals cannot be monitored, but overall system consumption such as total memory free is

available to be monitored. In some cases, depending on the utilization of the kernel, kernel scheduling can either negatively or positively affect performance in user space. Therefore, it is important to note that numbers discerned from system resource monitoring tools can be averages, statistically driven or even incorrect without dedicated instrumentation. We believe that these numbers represent reasonable performance indicators given the hardware used.

For both of the approaches (dual-stack and NSPG) using the processes outlined earlier in this section, the following procedures were performed in this order:

(1) Reset test environment including zeroing any logs, packet captures etc.

(2) Begin recording at network layer for packets

(3) Begin recording resource consumption on midstream host

(4) Execute testing scripts

(5) Upon finish, stop and save all recorded data

Final summary and analysis of all recorded data is in Section 5.2.1.

Threat Model

One approach to evaluating security is the use of threat models such as those created using the Microsoft (MS) methodology (Swiderski & Snyder, 2004). The goal of threat models is to choose a subject or potential target such as an application binary to discover or enumerate potential threats and determine whether any vulnerabilities exist without adequate threat mitigation. Alternatively, even if mitigations exist, this model can also highlight flaws that may have been missed during the initial processes such as data flowing over a program or devices boundaries. Through the modelling of both the dual-stack and NSPG approaches, it is hoped that the movement of security away from the legacy hosts and onto the intermediary host using the NSPG will be apparent with a direct side-by-side comparison. The remainder of the section is to serve as an introduction to MS threat model basics and an example on how a fictitious analyst would use a threat model to demonstrate the security of a solution; just as we aim to do this to compare dual-stack to the NSPG approach.

To establish common terminology in the context of MS-based thread models, we need to explore the following vocabulary:

- **Boundaries** - are virtual or physical boundaries that are used as interfaces that data can cross over when moving to another host or component. For example, a program able to save to a file system (FS) would have a boundary between itself and the FS
- **Data-flows** - are a flow of data that may move in a single direction or bidirectionally. They are not required to cross over a single boundary, however, they may cross over more than one as well. For example, two programs could converse over a network socket, which would represent two boundaries (one for each host)
- **Actors** - are external operators (human or otherwise) that initiate an action impacting the program or thing being evaluated. For example, a user enters input to be sent to the cloud via a HTML POST operation
- **Threats** - are potential vectors for an attacker to exploit (e.g., malicious Mallory could spoof an IP address to access X host) or operational conditions that could cause a DoS for example (e.g., a malformed packet could cause an application to crash if not handled correctly).
- **Mitigations** are processes or properties that have been deemed by the model's authors to adequately satisfy any potential threats. It should be noted that they may not be sufficient outside of the organization and mitigations can be bypassed

When considering potential threats to a system or application, it is useful to ask questions such as:

- How can an attacker access an unauthorized resource?
- What is the impact if an attacker can read application data in transit?
- What happens if malformed data are transmitted or entered through input fields?

Given these types of questions to ask about a particular system, threats can be grouped into categories to help you derive security-centric questions. For the MS threat model, STRIDE, an acronym for the following six threat categories is used:

- **Spoofing identity** - An example of identity spoofing is illegally accessing a resource by using an IP address that is designated for another host.
- **Tampering with data** - Data tampering involves the malicious modification of data. Examples of this include modifying an email without the author knowing, or modifying documents.
- **Repudiation** - Repudiation involves activities where a package or document could be received and the receiver denies having received it.
- **Information disclosure** - Information disclosure threats involve the confidentiality of information and the ability for others to read it. If privileged information can be read by an unauthorized user, this is considered an unauthorized information disclosure.
- **Denial of service** - Denial of service (DoS) threats revolve around system availability and reliability. A DoS may be intentional (malicious) or unintentional loss of access to a system due to an error.
- **Elevation of privilege** - Involves a specific user or system component that is executing under a specific set of restrictions such as a simple service, and somehow (unauthorized) becomes able to execute any task.

While the MS approach is not a definitive tool for discovering *all* threats of a target, the threat-model framework and tooling are free for use on recent versions of Windows using a graphical interface (Microsoft, 2016). To create a threat model, the following steps are performed:

(1) In canvas mode, create a diagram of major components used in the process being evaluated

(2) Determine and draw all boundaries within the components

(3) Using the boundaries as a point of intersection, draw all data flows between components while noting any directionality dependencies

(4) For each component, boundary and data flow, set the appropriate properties describing it. E.g., does the protocol have data integrity verification?

(5) In reporting mode, for each threat enumerated from the diagram, determine and describe the method of mitigation and validity of the threat if possible. E.g., is threat ABC valid? What did you do for X threat to mitigate it? Discovered threats will be listed as: “not started”, “not applicable”, “mitigation implemented” and “needs investigation”

(6) Finally, a comprehensive report can be generated once the evaluation is complete (e.g., all threats are reviewed and explored)

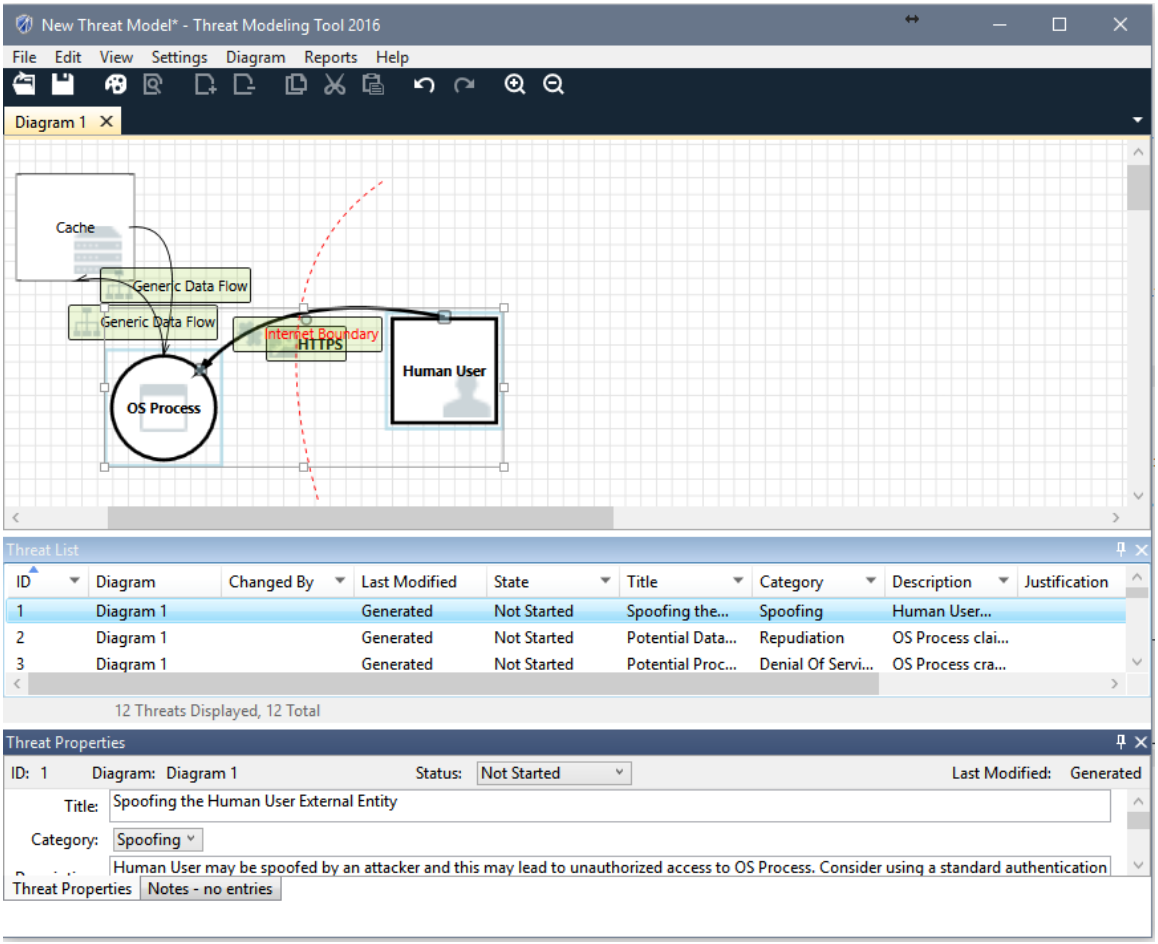

Figure 5.12: Threat modelling tool in detailed view

Once a threat model has been derived, then the author of the model can work through a detailed process of verifying whether or not any identified data flows have been mitigated against a selection of threats.

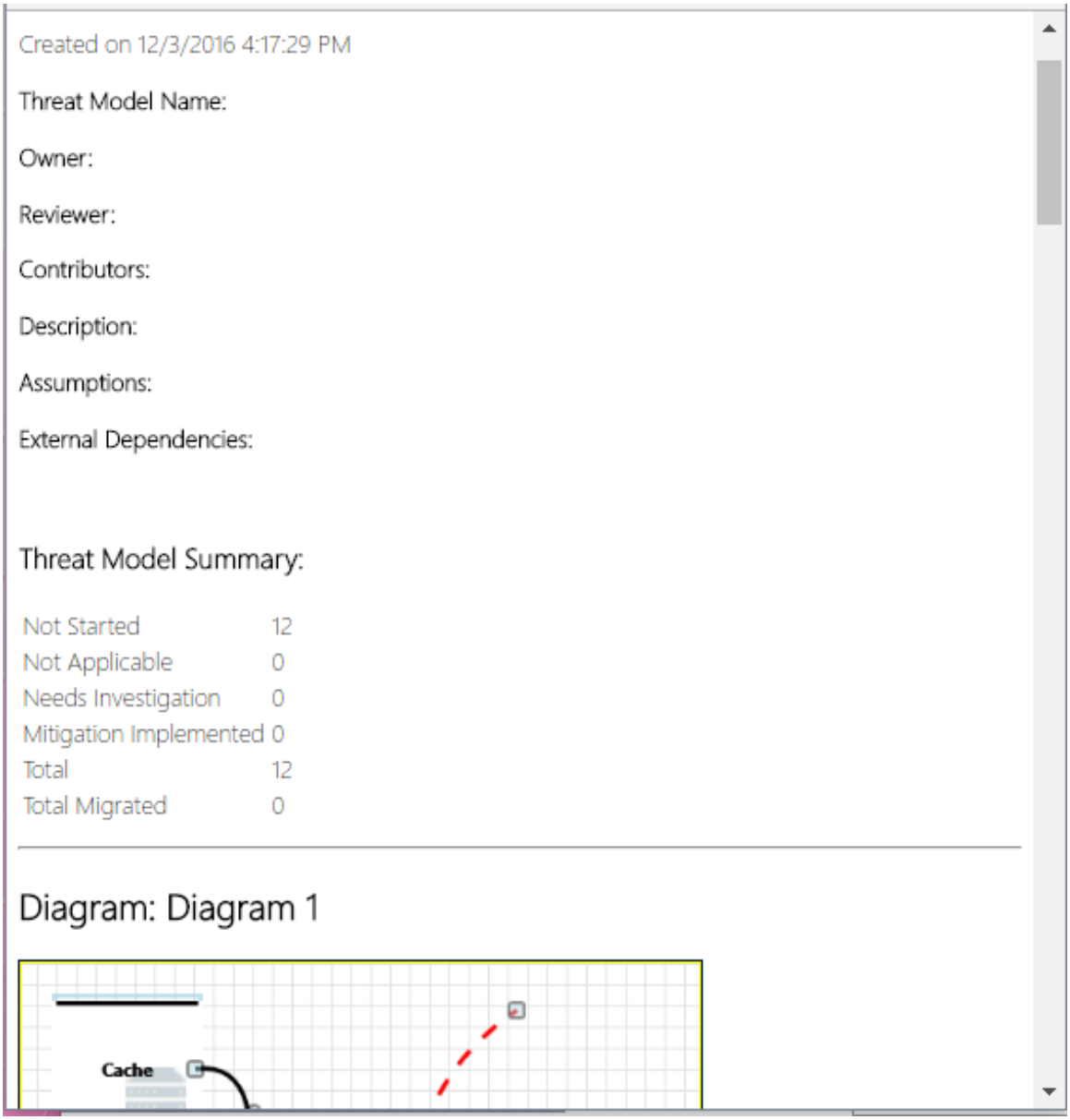

Figure 5.13: Threat modelling report example

Using the previous information about MS threat models and a generalized overview of the process, we can follow Jane the administrator using a threat model as an example. Now let us assume that Jane the administrator wants to create for herself a diagram to explore a dual-stack approach to another technology. She could use the MS threat model tool and online resources to quickly create a threat model *to the best of her ability*. Jane has a few choices, but she would likely create a threat

model using the steps listed earlier in the section. Depending on whether she has an in-depth exposure to network security, she may not be able to create as complete a threat model as she may like (unknowingly). Security is not a trivial endeavor and threats may be undetected despite herculean efforts to eradicate them. In the effort to discover as many flaws as possible, this is why we validate and supplement the threat models using the network tests in Section 5.2.2. Combined with the network testing results, the models are more accurate than without them to paint a more complete picture contrasting dual-stack and NSPG approaches.

After creating the threat models for both the dual-stack and NSPG (with refinement using network and consumption test results), we contrasted the models to accomplish our goals of verifying the superiority of the NSPG approach. Furthermore, the results, advantages and disadvantages of each approach will be discussed in Section 5.2.

5.2 Verification of Testing

In this section, the results of the testing and threat models for both the dual-stack and NSPG approaches are reviewed to discern a set of recommendations for the security of legacy devices. Using our fictitious example character, network administrator Jane, we examine the results to conclude a set of recommendations as if she were to come to the same conclusions from testing the methodologies in a network similar to the one modeled in Figure 5.2. To do this, we deployed both solutions depicted in Section 5.1.1, tested using the processes in Section 5.1.2, and explored the results collected post-testing. Following this, we built initial threat models for both approaches and supplemented them with the lower-level details the network and resource consumption tests.

The remainder of this section consists of two subsections:

- **Network and Resource Consumption Test Results** - Reports the testing results of network and resource consumption of both the baseline and proposed solution using the procedure outlined in Section 5.1.2
- **Threat Model Results** - Provides the threat models for both approaches using the MS method of threat model creation in Section 5.1.2

5.2.1 Network and Resource Consumption Test Results

Using the procedures Section 5.1.2, this section explores the lower-level (network) tests of both approaches: dual-stack and NSPG. The approaches were deployed on sub-model network in Section 5.1.1 and used automated tests to generate the results in the following sections.

Dual-stack Deployment

Before stepping into the key observations and results of testing the dual-stack, a key principle to remember is that the dual-stack approach is based on a different paradigm to control network traffic than the NSPG. Traditional stateful packet filtering firewalls merely filter packets while allowing them to pass through according to ACLs and the direction of establishment for a network stream; this is akin to installing a chest-high chain-link fence to protect a chicken coup. On one layer, a fence can be a deterrent from feeble attempts, but more enthusiastic attackers (such as the clever fox) can traverse the fence by going over or under to reach the vulnerable chickens. The legacy devices in our model are the vulnerable chickens (and are easy to disrupt), but were afforded additional protection to a certain degree by the dual-stack approach. Similarly to a process that may be executed by Jane our network administrator, we configured the firewall in a dual-stack approach to support SNMP and NETCONF. The remainder of this section contains the results for the dual-stack approach.

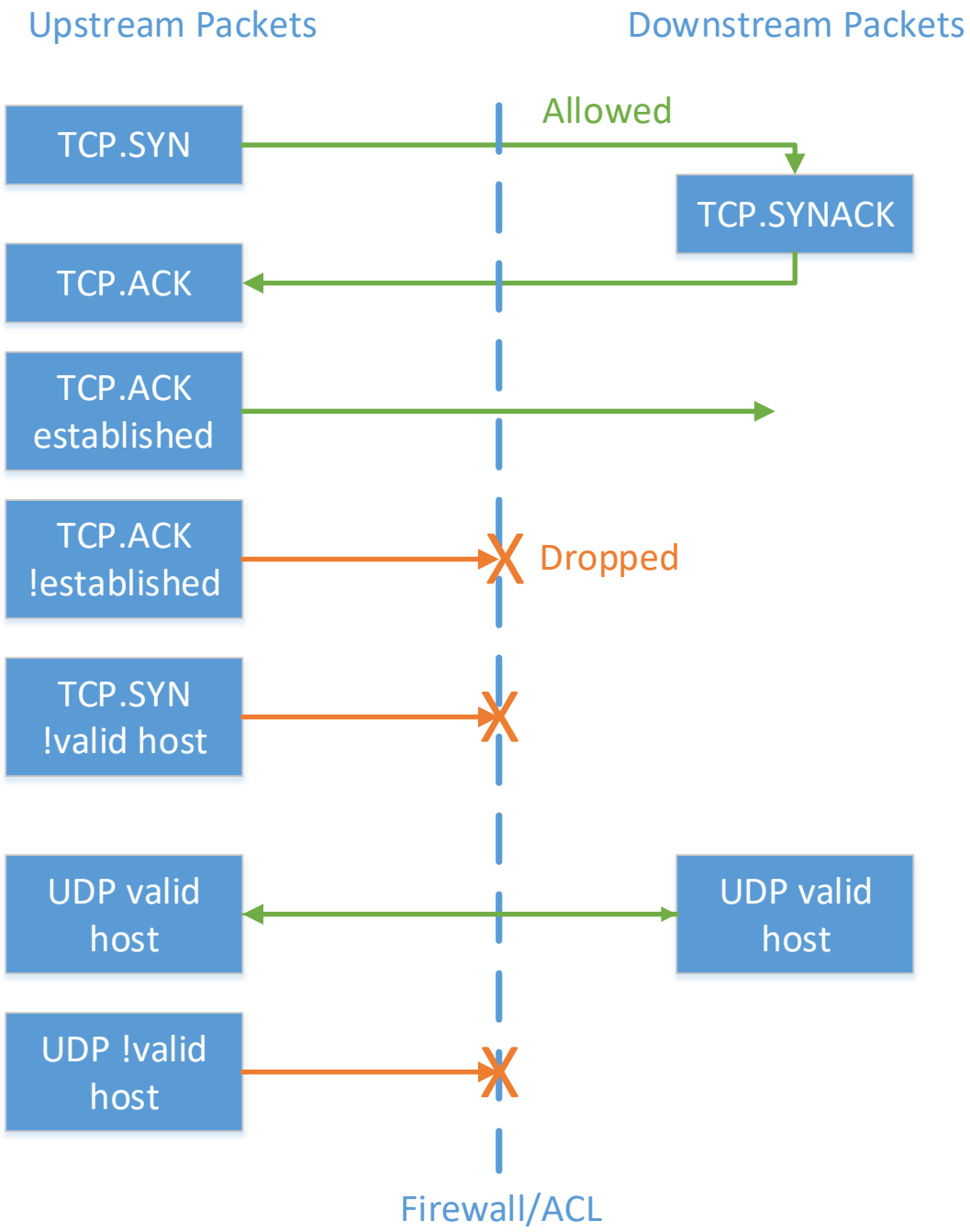

Figure 5.14: Infographic demonstrating normal packet ACL access/deny

Firstly, we verified that pass-through functionality allowed packets to traverse through the firewall and be forwarded from the upstream host to all downstream hosts. This was performed by

using upstream CLI applications to generate valid successful network connections for both SNMP and NETCONF (see Figure 5.14 for a brief graphical comparison). We also tested attempting to perform an SSH connection and ICMP ping to the legacy downstream hosts from the upstream host. The packets were dropped on the latter two connections; the firewall ACLs did filter and provide some level of protection for the legacy devices, but the question remains regarding how much protection was actually offered.

Next, in order to determine the level of protection afforded by even "locked down" ACLs (meaning only allowing connections from specific hosts), we connected to the legacy devices using SNMP and NETCONF to execute a series of commands. While SNMPv3 encryption and mandatory encryption when using NETCONF did provide confidentiality for packets traversing the firewall, the firewall was not troubled by this when matching packets against the ACLs. Unfortunately, the small distinction of encrypted/clear-text protocols has zero effect of traditional firewall operation because the firewall only inspects a packet for source and destination IP addresses, whether the packet is TCP, UDP and ICMP, if it has the allowed ports, and if it matches any state tracking rules (if applicable). This means that *any functionality at the application layer (SNMP or NETCONF) is not parsed* and can be "slipped" through the firewall intentionally or otherwise (see Figure 5.15 for examples). Otherwise, if packets do not match the ACLs, they are simply "dropped".

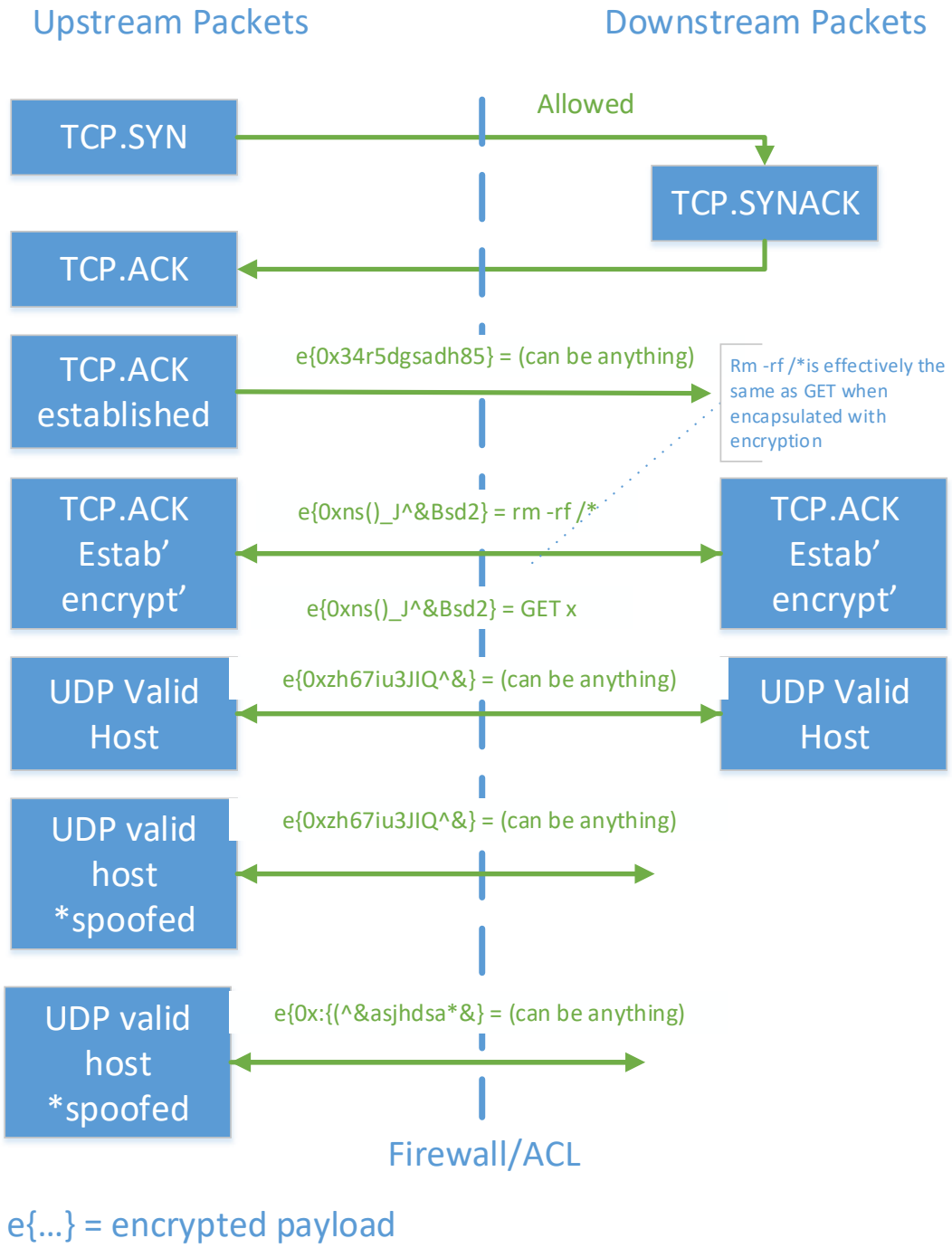

Figure 5.15: Infographic demonstrating indifference to packet payloads

Jane the network administrator may argue that an attacker would have to know a number of

things in order to successfully execute such an attack and she would be correct. An attacker would need to learn/guess the network addressing scheme, upstream and downstream host IP addresses, and any firewall ACLs. In this case, "security by obscurity" is not enough deter an attacker should he or she have obtained insider knowledge, performed passive sniffing of network traffic, and determined ACL rules through active probing using a tool such as NMAP (nmap.org, 2016). Therefore, Jane's assumption that the firewall provides adequate security is effectively "security by obscurity". The slight increase in difficulty for an attacker is a false sense of improved security and a small reduction of the potential attack surface for the legacy devices. (See Figure 5.16 for an example scenario where an anomaly in probing a firewall provides information about the ACL and network). Depending on the hosts and types of networking policies involved, a successful probe may result in a SYNACK (in the case of TCP) or a TCP RESET for example. This would give the attacker probing the network/ACL further information to refine the reconnaissance process.

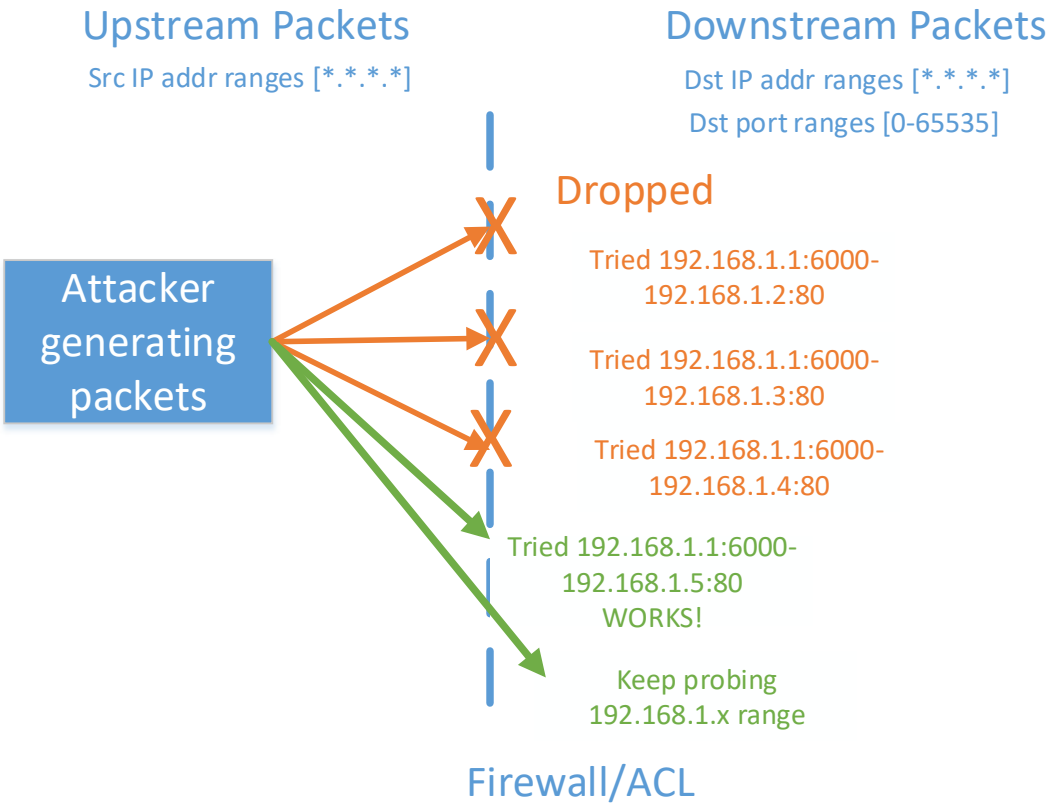

Figure 5.16: A simplified version of the ACL probing process used by attackers

Assuming that packets can be "slipped" through a traditional firewall, we examine which packets could traverse through a firewall if an attacker had determined the ACLs. These assumptions generate three cases that need to be explored:

(1) **The effect of a stateless protocol (UDP) and the firewall** - Whether or not any packet can be simply be spoofed if the application layer does not have the correct controls for verification and validation of network traffic? Can an attacker simply spoof a packet with the correct addressing information and reach a legacy device?

(2) **The effect of a stateful protocol (TCP) and the firewall** - Whether or not any packet can traverse the connection tracking mechanism of the firewall and what can an attacker do in regards to injecting a packet into an established stream?

(3) **The effect of an unfiltered payload reaching an end-device** - Whether or not the payload of a packet's application layer has any effect on the verdict (drop or allow decision) of a firewall? Alternatively, is there an effect should unfiltered network traffic reach a legacy device?

In the first case, we assume an attacker has determined the ACLs of the firewall. We injected a UDP packet using a MAC address and IP address of a valid host (from a non-valid host) on the upstream of the firewall targeting a legacy device on the downstream. The MAC addresses are not absolutely necessary as we could perform an active MitM attack by redirecting all traffic from valid hosts to be routed through an upstream attack host via ARP poisoning. Instead, it was to demonstrate that MAC addresses can be spoofed and that MAC address filters can be bypassed easily as well. The packet contained an arbitrary ephemeral source port, the destination port of SNMP, and a valid SNMP GET payload. Using the inspection points (upstream and downstream), we saw packets traverse the firewall in Figure 5.17.

```
13:27:04.675780 IP (tos 0x0, ttl 64, id 20486, offset 0, flags [DF],
  proto UDP (17), length 71)

192.168.1.2.57442 > 192.168.1.20.snmp:
  { SNMPv2c { GetRequest(28) R=2042202393  system.sysUpTime.0 } }
  E..GP.@.@.............b...3.F0).....public....y.........0.0...
  +.........

13:27:04.676035 IP (tos 0x0, ttl 64, id 20487, offset 0, flags [DF],
  proto UDP (17), length 74)

192.168.1.20.snmp > 192.168.1.2.57442:
  { SNMPv2c { GetResponse(31) R=2042202393
  system.sysUpTime.0=1694752 } }
  E..JP.@.@..............b.6.I0,.....public....y.........0.0...
  +.......C..
```

Figure 5.17: ASCII packet trace showing spoofed SNMP packet downstream bypassing the firewall ACL

Next, we examined injecting TCP packets through a firewall that has state tracking functionality. This is slightly less trivial for an attacker to circumvent and requires in-depth knowledge of TCP/IP to perform a successful injection compared to UDP injection. Successful TCP injection and session hijacking attacks require an active approach because of the TCP three way handshake during connection establishment, and packet sequence/acknowledgement numbering. If we generalize TCP sequence numbering and assume a successful handshake, imagine this example: host 1 sends packet A (SEQ 1, ACK 10, and size 10) to host 2. Host 2 responds with packet B (SEQ 1, ACK 20 (ACK 10 + size of packet A), and size 10) to host 1. For an attacker to insert a packet into the stream, the crafted packet would have to have the appropriate SEQ and ACK such that matches the receiver's expectations or it will be discarded. Alternatively, if an attacker sniffs packet A and packet B, the attacker could close the session host 1, and send a packet to the host 2 (SEQ 11, ACK 30, size 10) spoofing the MAC and IP addresses of host 1. In this scenario, host 1 would be unaware of the connection hijacking and would likely attempt to reestablish the connection.

To bypass TCP sequence/acknowledgement barriers and receive any packets post-injection, an attacker needs to be in a network location that grants knowledge of established valid connections. This can be performed by listening on a spanning port or performing an active MitM attack. Similar to an attacker, we performed the following actions in order to bypass the Linux firewall's stateful connection tracking:

(1) Established a NETCONF connection to a downstream host and kept it alive without terminating the NETCONF session

(2) Sniffed the network - We are on the upstream host passively sniffing the network in a location where all traffic must pass. Therefore, all communications including established streams are visible for monitoring

(3) Monitored and disrupted an active network stream - Choosing an established and monitored NETCONF connection, we can determine the next sequence numbers that have not been yet sent. This allows us to inject a TCP reset packet to the upstream host to close the connection, and perform a modified attack that resembles TCP session hijacking

(4) Injected non-legitimate (crafted) packets and verified that packets bypassed the firewall by sniffing on the midstream's downstream network interface

```
16:41:01.108171 IP (tos 0x0, ttl 64, id 51112, offset 0, flags [DF],
  proto TCP (6), length 60)192.168.0.2.60488 > 192.168.0.20.830: Flags [S],
  cksum 0x1128 (correct), seq 1013986434, win 29200, options
  [mss 1460,sackOK,TS val 7074602 ecr 0,nop,wscale 7], length 0

..... (Allow connection to be established through firewall)

..... (Inject RESET on bridge to legitimate host)

16:41:01.149474 IP (tos 0x0, ttl 47, id 51114, offset 0, flags [none],
  proto TCP (6), length 40)
  192.168.0.20.830 > 192.168.0.2.60488 : Flags [R], cksum 0x98cb (correct),
  seq 1, ack 1, win 0, length 0

.... (Inject crafted packet on bridge)

16:41:01.149490 IP (tos 0x0, ttl 47, id 51115, offset 0, flags [none],
  proto TCP (6), length 72)
  192.168.0.2.60488 > 192.168.0.20.830: Flags [P], cksum 0x8512 (correct),
  seq 1, ack 1, win 318, length 10 ThisIsFake

.... (Legitimate RESET packet returned)

16:41:01.149503 IP (tos 0x0, ttl 47, id 36070, offset 0, flags [none],
  proto TCP (6), length 40)
  192.168.0.2.60488 > 192.168.0.20.830 : Flags [R], cksum 0xa5e6 (correct),
  seq 73, ack 215, win 0, length 0
```

Figure 5.18: Simplified ACL bypass and TCP session hijack with crafted packet

In Figure 5.18, we observed the injected packet traversing the firewall, but also that a response packet was sent back. If we had not been monitoring upstream in an omniscient position (with

a very fast packet capture/processing/crafting application) or actively sniffing/MitM, the returning response packet would have been unknown. It is important to note that an upstream application would have re-instantiated the closed session (given the attacker sending a TCP RESET packet) and our hijacked session would continue existence until an inactivity time out. Alternatively, a well-designed application may have executed logic to end a duplicate session. Jane the network administrator may argue that using TCP alone increases security over UDP-based protocols because of the extra effort required by an attacker to inject into an established TCP stream. However, it is not completely secure as we have demonstrated nor is it feasibly possible to "rebase" widely established protocols that use UDP to use TCP instead (if we ignore any caveats of TCP vs. UDP as well).

For the last case, sending unfiltered payload data is a scenario that is agnostic to the underlying network layer protocols. If a packet can traverse firewall ACLs, then any payload at the application layer traverses as well. For legacy devices, a particular string of bytes crafted in a certain way could have disastrous results or could cause specific functionality to be engaged such as a reconfiguration event (see TFTP/SNMP reconfiguration for CISCO devices (CISCO, 2011)).

In the first two cases (TCP and UDP packets can be injected), we can see that valid, but not desired (potentially policy breaching) commands can be sent to legacy devices. Secondly, we demonstrated that arbitrary nonsense/malformed traffic can be injected into valid streams or directed at legacy hosts without being inspected (see Figure 5.18).

Moving forward from the analysis of the dual-stack approach at the packet level, we examined the resource consumption when running a dual-stack approach during testing. In real-world network deployments, packet rates fluctuate largely around tasks such as configuring devices or monitoring device statistics/events. During our simulated network get/set administration operations, the packet rates were quite low. The polling rates used for device information can vary from relatively lenient (once every five minutes) to quite rapid (once every thirty seconds) and are dependent on a number of factors to determine the actual frequency. For example, a network with copious hardware resources and low traffic load may allow frequent polling rates when compared to resource-constrained legacy devices with few spare computing cycles/resources. For these reasons, we chose four polling rates: 0 ms (fast as possible), 1000 ms (1 second), 30,000 ms (thirty seconds) and 60,000 ms (60 seconds).

	Idle	0ms	1000ms	30,000ms	60,000ms
Memory Utilization	21.19	40.87	40.42	38.12	36.01
CPU Utilization	0.07	100.01	87.42	16.73	9.96
Packets	0	200	182	6	7

Table 5.1: Performance and resource consumption of dual-stack on RasberryPi

After reviewing Table 5.1, we can conclude that short intervals, or when pauses between intervals are near non-existent, create high levels of resource consumption/utilization. For an experienced developer or system administrator, this would be an understandable consequence for frequent polling rates. However, besides resource consumption, there are indirect consequences that inexperienced personnel may not consciously be aware of. For example, there might be a higher potential to drop and re-transmit network packets, and increased latency for packet processing or hosted services. If polling rates are less frequent, resource utilization and consumption begin to stabilize and have only a slight effect on real-world performance. The latter observation is critical in the world of "dollars and cents" devices where manufacturers aim to increase Return-On-Investment (ROI) by developing devices with the bare-minimum of hardware necessary to generate the highest revenue per device sold. It is likely that performance and resource consumption would be vastly improved in devices that are specialized for network operations and that have performance optimized software.

NSPG Deployment

To test the NSPG, we used a strategy similar to the one used in the testing of the dual-stack approach, but with some alterations due differences in architecture paradigms. The NSPG's key architecture difference is that traffic does not pass through the device (e.g., NETCONF and SNMP), but the device acts as a proxy making connections on behalf of authenticated requests. Connections to legacy devices are only possible with an upstream host/application connecting to the NSPG and the NSPG acting as a hybrid proxy/protocol gateway. Unless upstream NETCONF connections are forwarded to downstream NETCONF capable hosts, no incoming traffic can reach the legacy downstream devices in its original form forwarded or tunnelled (see Figure 5.19).

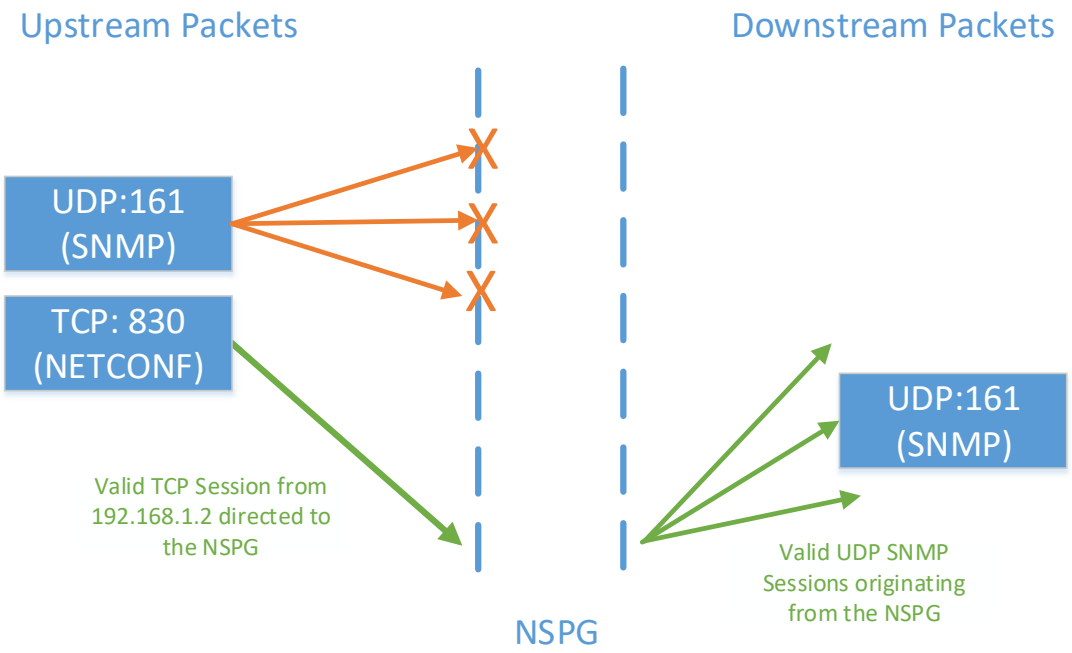

Figure 5.19: Simplified infographic of the NSPG and host dropping traffic, while only the NSPG can only generate SNMP traffic (on the downstream interface) based on policy configuration

Similar to the initial tests for the dual-stack approach, we began with sanity checking network connectivity against what we suspected to be true: the NSPG would drop all traffic except NETCONF connections destined for itself. This was performed using tcpdump at the inspection points, yangcli tool for NETCONF, and snmp-get to verify that NETCONF connections aimed at the NSPG would succeed and SNMP connections would fail.

Further comprehensive testing on the upstream suggested that connections were either successful or unsuccessful. In order to determine if the NSPG was indeed actually intercepting traffic correctly, we needed to further explore the protection provided by the NSPG at the network layer by injecting and attempting connections to the downstream legacy hosts using a variety of protocols/ports (detailed results can be found in Appendix A.8). From tcpdump captures on both the upstream and downstream, we observed that the upstream NETCONF connections utilize TCP (and therefore have state given the TCP three way handshake) and the downstream SNMP connections utilize UDP in all flavors of SNMP (1, 2c and 3). The upstream NETCONF traffic is encrypted due

to the use of SSH as the underlying application layer protocol. Depending on the version of SNMP, it is either clear text or encrypted (SNMPv3 uses encryption in this test), and connections are not forwarded downstream (as expected).

On the downstream side of the NSPG, SNMP connections originating from the NSPG were observed. These connections were destined for the downstream legacy hosts. This could be a cause for concern if the same SNMP traffic was also present on the upstream and the NSPG was a "black-box" solution that could be forwarding tunneled SNMP traffic in a way similar to a VPN or had an ACL "hole". Fortunately, neither is true given details from the NSPG design in Section 4.1 (detailed results can be found in Appendix A.9).

Next, we tested the NSPG's access controls by verifying that legacy device information can be accessed by specific users and whether they can access the appropriate resources. We installed four different users on the NSPG and assigned them various policies (see Figure 5.20). Using these four users and an additional non-existent one (for testing simple NETCONF authentication mechanisms), we queried the policies in an attempt to access information that would be beyond the specified controls.

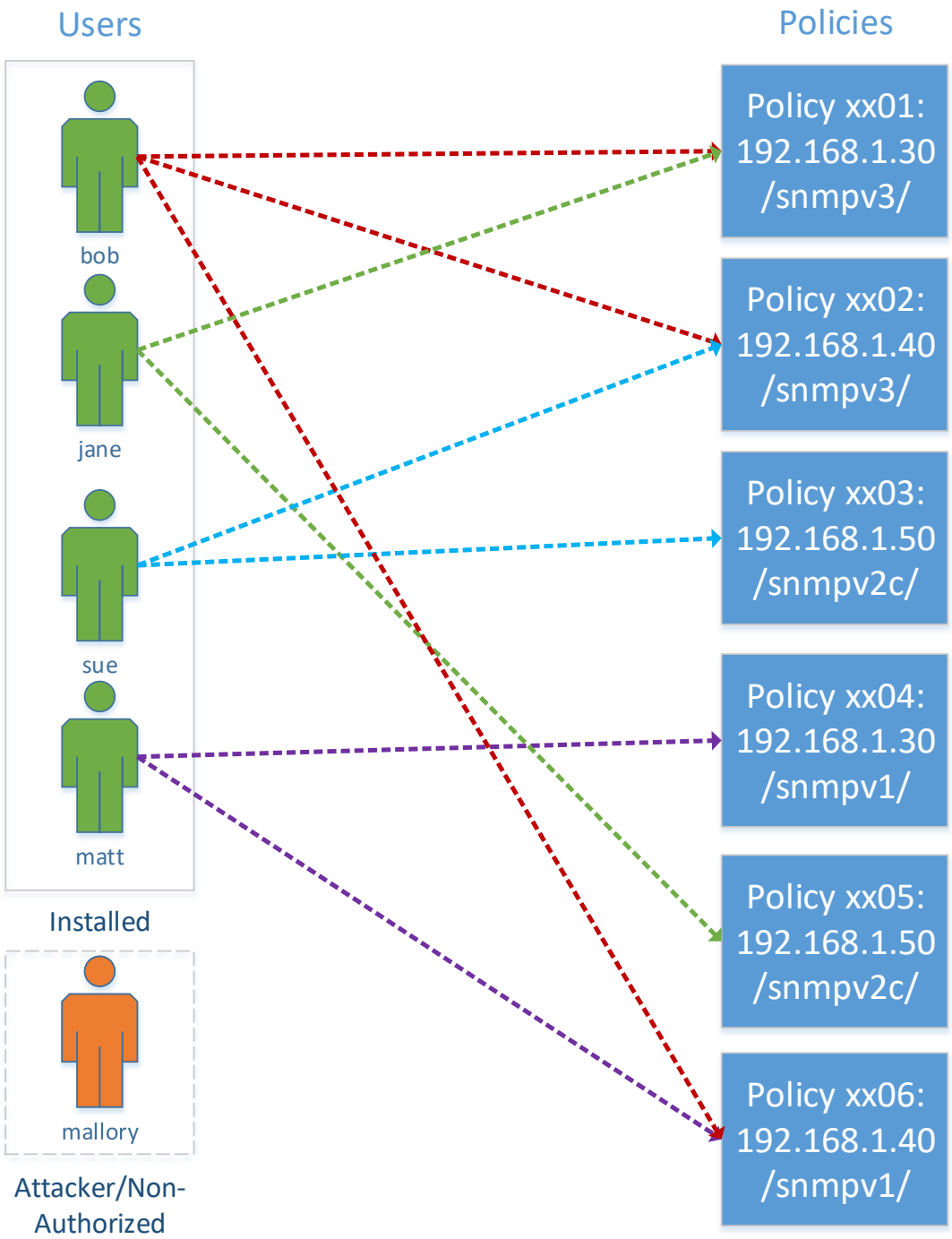

Figure 5.20: High-level diagram showing NSPG policy to user assignments

Using the user and group assignments in Figure 5.20, we began testing the NSPG's policy ACL

management component using a non-installed user to verify that connections to the NSPG cannot be instantiated if an invalid user is used. Then we verified that the three installed users can access the appropriate policies by connecting to the NSPG using the yangcli utility. For further verification, we attempted with each of the users to access policies assigned to other users and were denied when appropriate. For a comprehensive view, Figure 5.21 outlines the access of policies. For detailed information about the policies themselves and whom they are assigned to, see Appendix A.5.

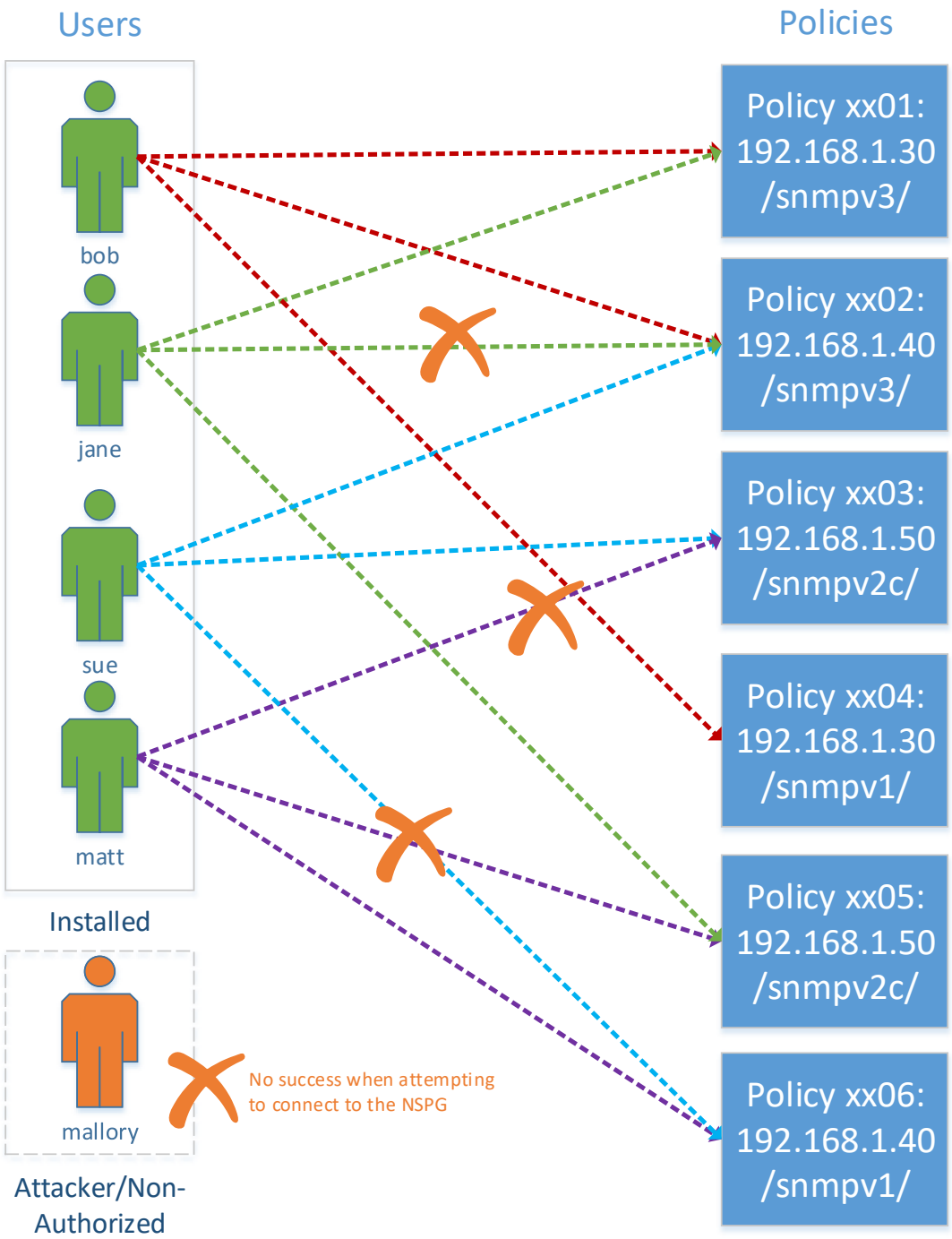

Figure 5.21: High-level diagram showing policy access successes and failures

Moving forward from the analysis of the NSPG approach at the packet and policy ACL levels,

	Idle	0ms	1000ms	30,000ms	60,000ms
Memory Utilization	37.25	52.87	48.42	45.12	42.01
CPU Utilization	25.41	100.00	95.41	75.67	55.49
Packets	0	218	201	53	28

Table 5.2: Performance and resource consumption of NSPG on RasberryPi

we examined the resources required by an operating NSPG. The yangcli can send requests faster than a human being typing commands, and the NSPG can also more-or-less parse commands and return a response as fast as the application can be executed. The testing automation scripts execute commands through the yangcli and the NSPG responds with the data requested (if allowed, a response containing the information, or an error if not). Unfortunately, due to host resource scheduling, we could not account for the NSPG process "scheduling misses" if the NSPG application misses execution "time-slices" due to voluntary preemption. Having said that, we felt that millisecond resolution results for NSPG operation should suffice for determining measurements for the processes load for the NSPG for a single connection, for multiple connections and for several polling rates: 0 ms (fast as possible), 1000 ms, 30,000 ms (thirty seconds) and 60,000 ms (one minute). The later polling rates are the same values as used in the dual-stack approach to simulate the similar load that would be applied to the legacy devices should a upstream host be polling them with SNMP. However, the difference is that the NSPG is generating SNMP traffic instead of the upstream host. The upstream polling rate had a negligible effect on resource consumption and in real-world deployments, we believe that upstream polling may be less frequent than the polling of the legacy devices so we simulated upstream polling using a userspace timer. This is because in the future, we suspect that an administrator may perform periodic polling for statistics, but focus on using near real-time notifications or alerts when abnormal values (or the absence of values) are present.

Using Table 5.2 as a reference, we averaged the recorded low and high values of resource consumption for a period of one second, and used an appropriate divider to derive an approximation. Frequent polling rates cause an increase of resource consumption and conversely, as polling rates become more reasonable, the NSPG resource utilization is lessened. The values within Table 5.2 provide a measure to determine the bare minimum of resources required to protect a single end device.

	Idle	0ms	1000ms	30,000ms	60,000ms
Memory (1)	21.62	40.03	34.47	32.84	29.02
CPU (1)	0.11	100.00	87.42	16.73	9.96
Memory (2)	NA	41.23	36.12	34.03	31.79
CPU (2)	NA	100.00	94.31	25.09	17.32
Memory (3)	NA	44.15	34.47	36.65	29.02
CPU (3)	NA	100.00	99.76	31.72	24.56

Table 5.3: Performance and resource consumption of NSPG for 1, 2, and 3 policies

However, in the real world, we reasonably assume that there would be a single upstream connection polling multiple downstream devices (e.g., Jane the administrator polling the devices from an NMS). Table 5.3 illustrates the effect of polling rates using the following values: 0 ms (fast as possible), 1000 ms, 30,000 ms (thirty seconds) and 60,000 ms (one minute) for three policies (one for each user installed on the NSPG host). The NSPG utilizes (21.62/0.11) (RAM/CPU respectively) resources as a baseline point with a single connection from the upstream the NSPG, and for a single connection with a single policy polling as fast as possible (44.15/100.00), and (29.02/9.96) once a second. With two connections (41.23/100) quick as possible, and once a second (31.79/17.32). Finally, three connections with only a single host connected use (44.15/100.00) fast as possible and (29.02/24.56) with 1 second polling rate. Overall, resource consumption increases at a rate consistent with adding more policies and polling rates.

Similar to the dual-stack approach, higher polling rates results in higher levels of resource consumption, but the NSPG results in even higher resource consumption. This is likely due to the usage of having to perform operations inside of userspace: IPC between the NETCONF agent (OpenYuma) and the remaining NSPG components, and storing/converting data from binary to XML (and vice versa). Performance and resource consumption would be greatly improved in devices specialized for network operations and optimized software, but the NSPG will always have greater resource requirements than a "simple" dedicated firewall. It is important to note that rate-limiting on the upstream connection and resource quotas within the NSPG could be applied to constrain resource consumption should specific technical performance requirements be necessary.

5.2.2 Threat Model Results

Using the procedures in Section 5.1.2, this section explores the high-level threat models of both solutions: dual-stack and NSPG. Following both threat models, a comparison of both will be performed to demonstrate the movement of boundaries being focused on the midstream component and not the end legacy components themselves. For each approach, we briefly explore the overall threat model, data flows, associated threats (with their status) and a summary conclusion.

It is important to note that the exact threat names are not absolute nomenclature to describe a problem, but rather serve as markers such that a person doing threat modelling can apply hypothesizes to derive security conclusions/recommendations.

Dual-stack Model

Using the lower-level tests in Section 5.2.2 and understanding the dual-stack approach in Section 2.4.1, we generated the following threat model in Figure 5.22.

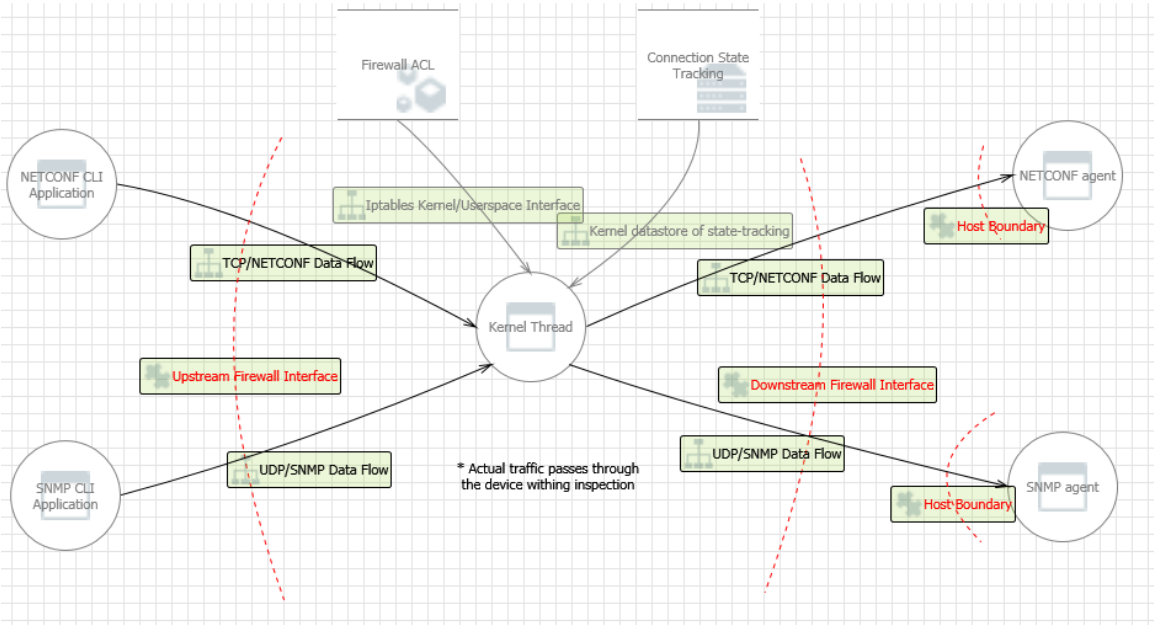

Figure 5.22: High-level threat model of the dual-stack approach

From the high level threat model, four boundaries surrounding the midstream component are

present. Starting from the left, the “upstream firewall interface” is the upstream side of the midstream device. Next, the “downstream firewall interface” is on the secure side of the device. It shares two data flows between the downstream “host boundaries”. To illustrate the general flow, packets move from the upstream applications through the midstream device to the downstream components. The legacy devices, whose security concerns us, are on the bottom right.

The faint grey data flows and lighter shaded icons are out of scope. The more prominent darker ones are “in scope“ and there are four that can be simplified down to only two: the TCP/NETCONF data flow, and the UDP/SNMP data flow. This research makes a critical assertion that the Linux midstream device is trusted to be secure. Any agent/CLI applications are not being reviewed for security outside of any interactions on determining the superiority of the NSPG.

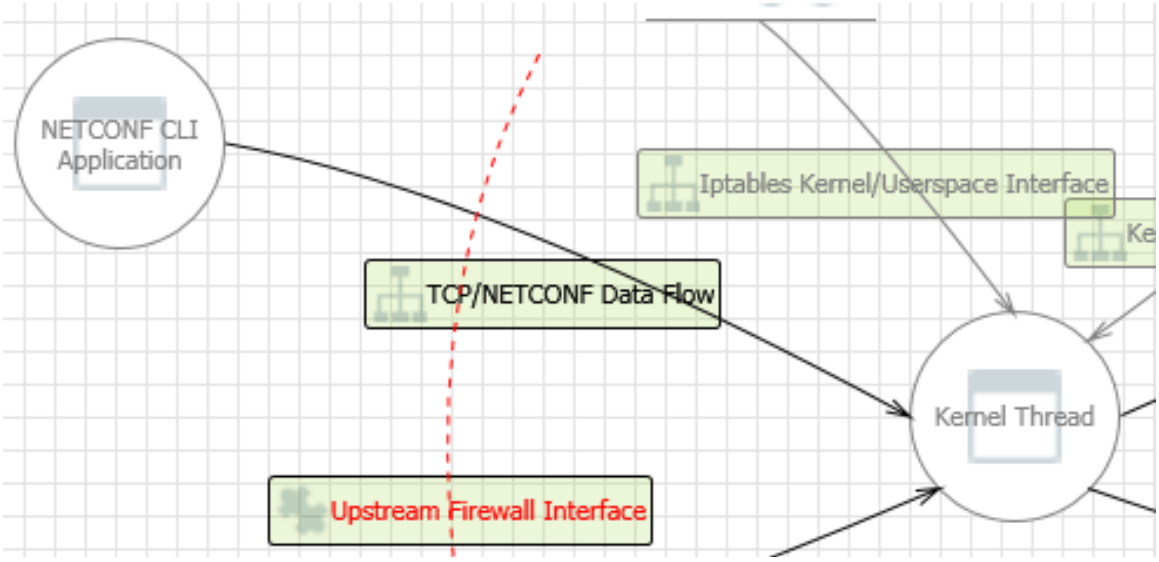

Figure 5.23: Dual-stack TCP/NETCONF data flow figure (upstream)

In the upstream TCP/NETCONF data flow, there is a number of potential threats identified by the model: 9 total for this data flow, 1 needs investigation, 2 are considered mitigation implemented and 6 not applicable.

- **XML DTD and XSLT Processing** - We know that XML is used by NETCONF within the encapsulated tunnel and is being parsed by the NMS and downstream agents. However, this

threat is considered out of scope for this flow because the actual NETCONF protocol implementation is out of scope for comparing the two transitional approaches. It is considered **not applicable**

- **Spoofing the Kernel Thread Process** - Involves whether an attacker could spoof the NSPG's existence. The host itself can be spoofed at the network level, but upstream clients could utilize certificate verification (NETCONF over SSH utilizes cryptographic certificates) to mitigate the possibility of the NSPG being spoofed. Considered as **mitigation implemented** because SSH keys are already used by the NSPG and the responsibility of this verification process is placed upon the upstream NETCONF applications

- **NETCONF CLI Application Process Memory Tampered** - The NETCONF CLI/NMS application could theoretically manipulate the Linux kernel. However, the Linux kernel and network subsystems are not under security scrutiny for this threat model. Therefore, this threat is considered **not applicable**

- **Potential Process Crash or Stop for Kernel Thread** - The Linux kernel could receive a packet and be silently dropped. However, most (if not all) packets received will be processed by a driver and Linux NETFILTER (iptables). If a packet never reached the kernel or was never forwarded to a valid host, there may be a duplicate IP address or another issue (such as a malicious person) on the upstream of the midstream host. Iptables offers logging functionality to monitor packet streams. Therefore, we consider this threat to have a **mitigation implemented** for most network scenarios

- **Data Flow Generic Data Flow Is Potentially Interrupted** - Involves an external agent/actor interrupting data flowing across the boundary in both directions. If ARP spoofing, route poisoning or physical network modifications are present, there are few preventative measures at the midstream level except for tables of static addressing entries. This threat is a known issue with TCP/IP IPv4 networks; it should be considered **needs investigation** for high availability/critical applications

- **Elevation Using Impersonation** - Potentially malicious crafted traffic may create conditions

where a vulnerability could be exploited in the Linux kernel and/or NETFILTER/iptables. This "is out of scope" because we are not investigating the strength of the Linux's security. This threat is considered **not applicable**

- **Kernel Thread May be Subject to Elevation of Privilege Using Remote Code Execution** - Potentially malicious crafted traffic could cause a vulnerability in the Linux kernel and/or NETFILTER/iptables as these subsystems run within the kernel itself (monolithic and with root permissions), but that is out of scope; we are not investigating the security of Linux as an OS. This threat is considered **not applicable**.

- **Elevation by Changing the Execution Flow in Kernel Thread** - Potentially malicious crafted traffic may create conditions where a vulnerability could be exploited in the Linux kernel and/or NETFILTER/iptables. Linux subsystems run within the kernel itself (monolithic and with root permissions), but they are out of scope for this model; we are not investigating the security of Linux as an OS. This threat is considered **not applicable**

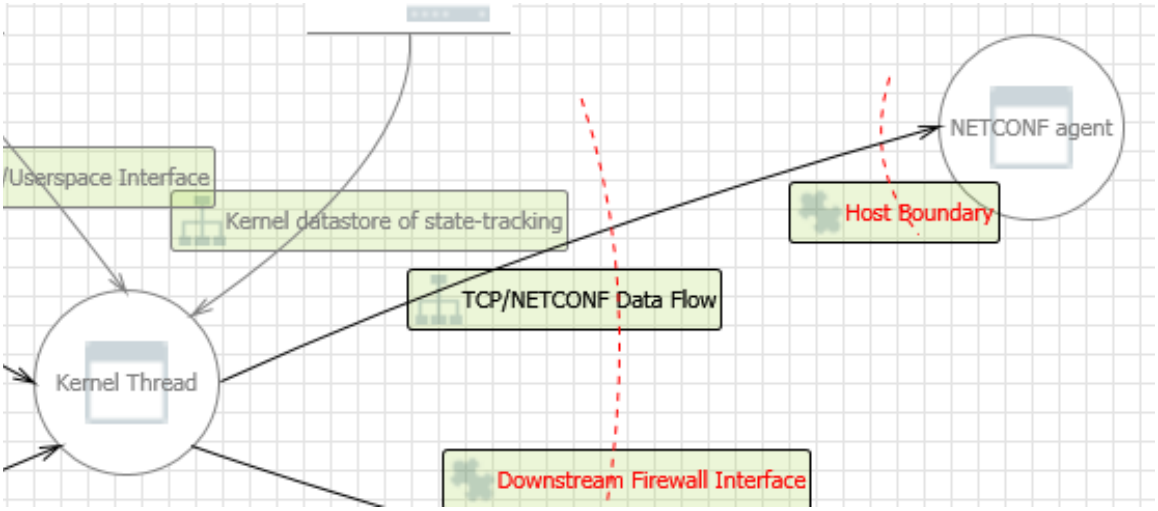

Figure 5.24: Dual-stack TCP/NETCONF data flow figure (downstream)

In the downstream TCP/NETCONF data flow, there are a number of potential threats identified by the model: 9 total for this data flow, 5 considered needs investigation, 4 are not applicable.

- **Elevation by Changing the Execution Flow in OS Process** - Should a user be able to authenticate and establish a session on the end host, invalid NETCONF/XML/input data could be injected. This is not a direct threat to the midstream host, but rather to the end point itself. If there are any security vulnerabilities on legacy hosts and a session is established, protection provided by ACLs is sub-par to the NSPG. The transference of security (or lack of) provided by the midstream host to the legacy end hosts demonstrates that a dual-stack approach may be ineffective in securing legacy end hosts (even if they use TCP and cryptography for communication). This threat is considered as **needs investigation**

- **OS Process May be Subject to Elevation of Privilege Using Remote Code Execution** - The kernel should not be able to remotely execute code causing an escalation on the end point as it is unaware of the content contained within the encrypted TCP stream. This is considered as **not applicable** because the Linux kernel is out of scope

- **Kernel Thread Process Memory Tampered** - The midstream host is not the same host as a downstream one and does not share or access the same memory. This is considered as **not applicable**

- **Elevation Using Impersonation** - Any downstream hosts should not be able to impersonate the midstream component as the midstream component is the gateway to the upstream network. The midstream component does not generate NETCONF traffic in this approach and is considered as **not applicable**

- **Data Flow TCP/NETCONF Data Flow Is Potentially Interrupted** - NETCONF communications should not appear to be originating from the midstream component (although they may if masquerading/NAT is present). In a simple forwarding configuration, this is not possible and considered **not applicable**

- **Potential Process Crash or Stop for OS Process (Linux)** - If the NETCONF agent crashes on a downstream host, it will not crash the dual-stack midstream component. However, when the downstream agent crashes, this would violate availability of a downstream host. The robustness/reliability of the end host is out of scope, but should be investigated in specific

environments. In this scenario, security provided by the midstream host was transferred to the end host. This is considered as **needs investigation**

- **Potential Data Repudiation by OS Process (Linux)** - The midstream component can log addressing and time information of when a connection may occur, but does not know whom (the user name as a parameter of the request) accessed the downstream host. This is an area that the midstream approach is noticeably deficient and projects the security boundary onto the downstream host. This threat is considered as **needs investigation**

- **XML DTD and XSLT Processing** - The dual-stack midstream approach merely forwards traffic and does not have knowledge of the data within the encrypted stream. Data within the stream may contain XML-related vulnerabilities that pass undetected through the midstream component and cause faults/vulnerabilities on the downstream component. The dual-stack approach does not provide any mitigation capabilities in this scenario and projects the security boundary onto the downstream host. This threat is considered as **needs investigation**

- **Spoofing the OS Process Process** - While various techniques of TCP spoofing may be used, they are different from those for UDP because of TCP handshaking, sequence/acknowledgement numbers and state tracking. The risk of "slipping" packets past the dual-stack approach should be investigated, but the risk of a fault on the downstream host is unknown. It should be considered as **needs investigation** because the responsibility of security is projected onto the end host instead of the midstream host.

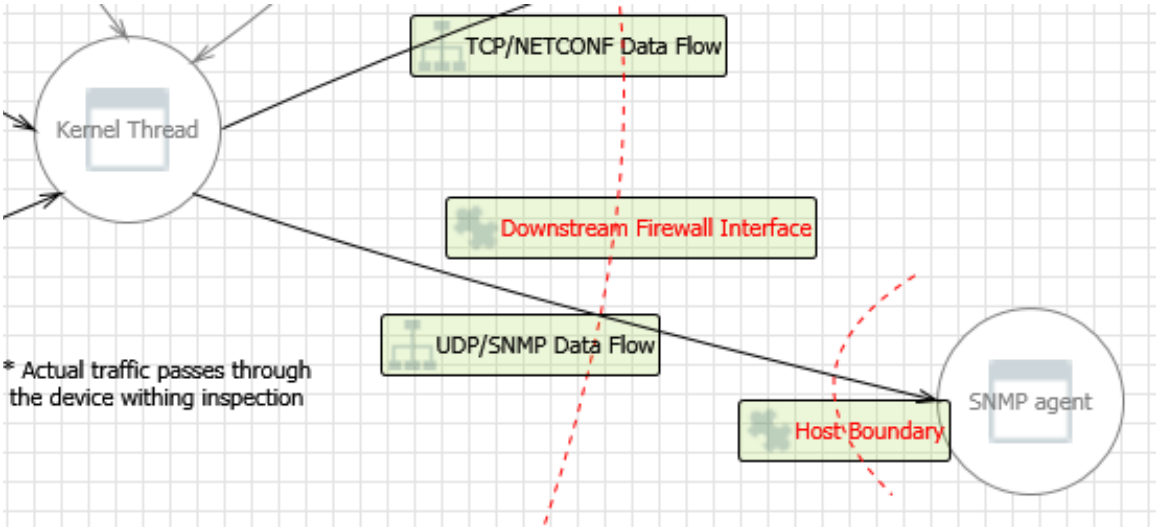

Figure 5.25: Dual-stack UDP/SNMP data flow figure (upstream)

In the upstream UDP/SNMP data flow, there are a number of potential threats identified by the model: 11 for this data flow, 9 need investigation and 2 are considered as not applicable.

- **Potential Data Repudiation by OS Process** - The midstream component does not generate traffic without forwarding traffic from the upstream interface to the downstream interface nor is SNMP a state-tracked protocol (it is UDP and connectionless). Upstream traffic may potentially be dropped and the midstream component may appear to be responsible; the problem may be upstream. Depending on the deployment environment, this should be considered as **needs investigation**

- **Potential Lack of Input Validation for OS Process** - Due to the nature of UDP and lack of cryptography in SNMPv[1, 2c], an attacker could the tamper with data in transit. Despite assuming that the downstream is secure, it is a false hope to consider the security of the downstream network/hosts immutable. Downstream hosts and networks can be compromised. Similarly, upstream networks are insecure and the midstream device may be unaware of such manipulation. SNMPv3 does provide security mechanisms to prevent tampering if used, however, the threat of unauthenticated traffic being forwarded without validation should be considered as **needs investigation**

- **Spoofing the OS Process Process** - Assuming that the downstream network is secure, the downstream host should not be able to be spoofed on the downstream side of the device. However, upstream networks are insecure and the midstream device may be unaware that it is the subject of manipulation; it could be unknowingly be spoofed. Therefore, we considered the threat as **needs investigation**

- **Spoofing the Kernel Thread Process** - Upstream hosts do not have direct access to the midstream hosts memory unless a packet exploits a specific vulnerability in the Linux kernel or C libraries. It is considered **not applicable**

- **Kernel Thread Process Memory Tampered** - The memory of both the midstream component and the upstream component are independent of each other by several layers including network interfaces unless a packet exploits a specific vulnerability in the Linux kernel or C libraries. This is considered **not applicable**

- **Elevation by Changing the Execution Flow in OS Process** - An attacker may be able to inject data into the SNMP agent downstream to alter flow program execution. This threat **needs investigation** as the dual-stack approach offers minimal security in this scenario

- **OS Process May be Subject to Elevation of Privilege Using Remote Code Execution** - The actual security of the end host is out of scope, but an attacker could inject packets that may be able to exploit weaknesses in the Linux kernel. This threat **needs investigation**

- **Elevation Using Impersonation** - If the upstream network is not adequately secured, a compromised or malicious host on the upstream network could poison routing and impersonate the midstream component. Therefore, this should be considered as **needs investigation**

- **Data Flow UDP/SNMP Data Flow Is Potentially Interrupted** - UDP or any network operation can be interrupted if the downstream host is taken offline for any particular reason. However, this should be considered out of scope and **not applicable** because the downstream network is considered secure.

- **Potential Process Crash or Stop for OS Process** - Depending on the network environment, a crash of the midstream host will halt forwarding traffic. This does not affect the security

for downstream devices, but it may interfere with critical network operations. It should be considered as **needs investigation**

- **Data Flow Sniffing** - Unless SNMPv3 traffic is used (with security mechanisms), then traffic can be sniffed using a number of methods (especially on the upstream network). This threat **needs investigation**

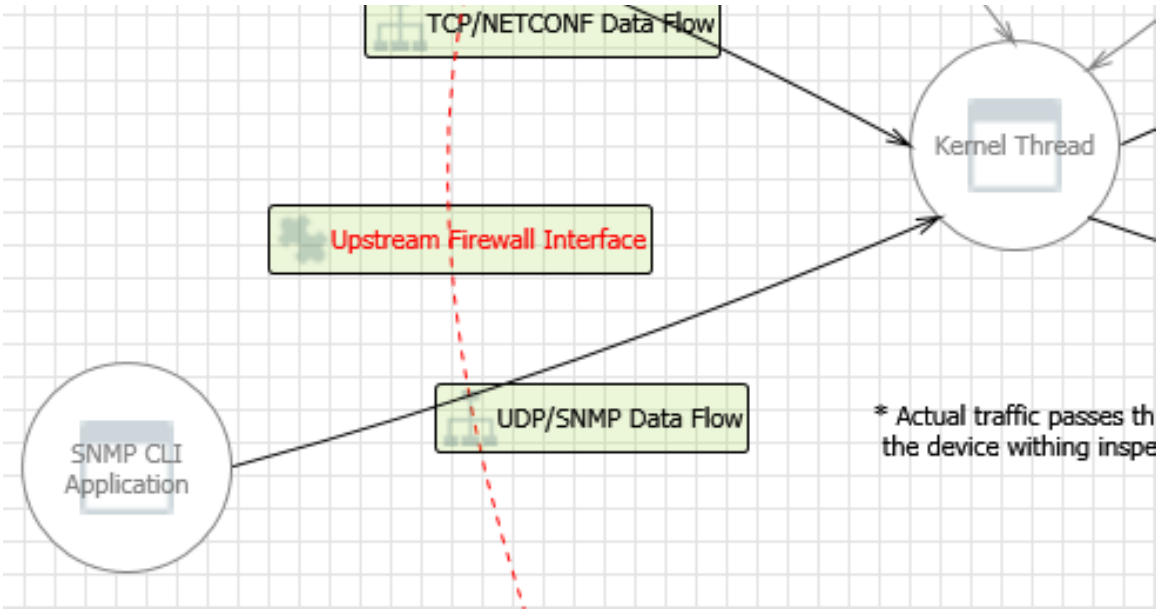

Figure 5.26: Dual-stack UDP/SNMP data flow figure (downstream)

In the downstream UDP/SNMP data flow, we can see that this is the UDP SNMP flow. There are a number of potential threats identified by the model: 11 for this data flow, 7 need investigation, and 4 are not applicable.

- **Elevation by Changing the Execution Flow in Kernel Thread** - The security of the Linux kernel and iptables/NETFILTER is out of scope. It simply tracks the existence of a connection and forwards valid traffic according to the configured ACLs. This is considered as **not applicable**

- **Kernel Thread May be Subject to Elevation of Privilege Using Remote Code Execution** - The security of the Linux kernel and iptables/NETFILTER is out of scope. The kernel

simply tracks the existence of a connection and only forwards traffic. This is considered as **not applicable**

- **Elevation Using Impersonation** - Impersonation of the upstream hosts may be mitigated by the use of strong mutual authentication mechanisms such as certificate verification on downstream hosts. UDP data can be spoofed including source and destination addresses, and slipped through a firewall. If credentials of downstream hosts are known, it is easy to escalate an attack to cause more damage or effect. This should be considered **needs investigation**

- **Data Flow UDP Is Potentially Interrupted** - Devices will still be protected if the midstream component is disrupted or becomes unavailable. However, the consequences of a disruption for ongoing communications may cause a number of issues in industrial or real-time networks. This should be considered as **needs investigation** depending on the use case

- **Potential Process Crash or Stop for Kernel Thread** - Devices will still be protected if the midstream component is disrupted or unavailable. However, the consequences on ongoing communications may cause issues in industrial or real-time networks. This should be considered as **needs investigation** depending on the use case or environment

- **Data Flow Sniffing** - SNMPv[1, 2c] is inherently secure and utilizes clear-text transmission of data; the downstream network is considered secure, but for packets to reach the downstream, they must be forwarded from the upstream. Unless SNMPv3 is used with encryption or another technique such as a VPN is employed, this is considered as **needs investigation**. On legacy hosts, VPN software may not be available and the existence of VPN software should not be relied upon especially in legacy industrial devices

- **Potential Data Repudiation by Kernel Thread** - We believe that any network traffic being filtered (including valid connections) should be monitored and logged. This is standard best practices in networking and should be considered as **needs investigation**

- **SNMP CLI Application Process Memory Tampered** - The security of the Linux kernel and iptables/NETFILTER are out of scope. They should simply track the existence of a connection and only forward traffic. This is considered as **not applicable**

- **Potential Lack of Input Validation for Kernel Thread** - The midstream host in this scenario is unable to inspect traffic outbound and forwards allowed traffic from the upstream interface. It is possible an attacker could manipulate downstream hosts to generate traffic that could cause an error in the Linux host through a calculated attack on seemingly secure hosts. It should be considered as **needs investigation**

- **Spoofing the Kernel Thread Process** - Although the midstream host could be spoofed, the downstream network is considered secure. This threat is considered as **not applicable**

- **Spoofing the SNMP CLI Application Process** - while the midstream component cannot be accessed by a upstream application, it forwards the packets generated by it. Similarly, downstream hosts are considered secure and could generate seemingly valid data. The midstream component could be misled and forward invalid connections in either direction. This should be considered as **needs investigation**, especially if sensitive information may be ex-filtrated from an internal production network

From the determined threats, a common theme emerged with varying severity depending on the trust of a system or technology. If a component (the midstream firewall) can be trusted, then a certain amount of security is implied based on the principle that merely filtering network connections according to an ACL provides a certain level of security (whatever that level may be in reality). Unfortunately, the issues arise from the transference of security onto the end hosts instead of the firewall technology when using state tracking and ACLs falls short of expectations. If ACLs can be bypassed with minimal effort (demonstrated in Section 5.2.1), then the security of an ACL is minimal when legacy hosts have to "fend" for themselves. Using firewalls to secure network perimeters is a best security practice as they are easy to implement and a solid first layer of security, but they should not be solely relied upon to adequately protect legacy devices.

NSPG Model

Using the lower-level tests in Section 5.2.1 and understanding of the NSPG approach in Section 4, we generated the following threat model in Figure 5.27.

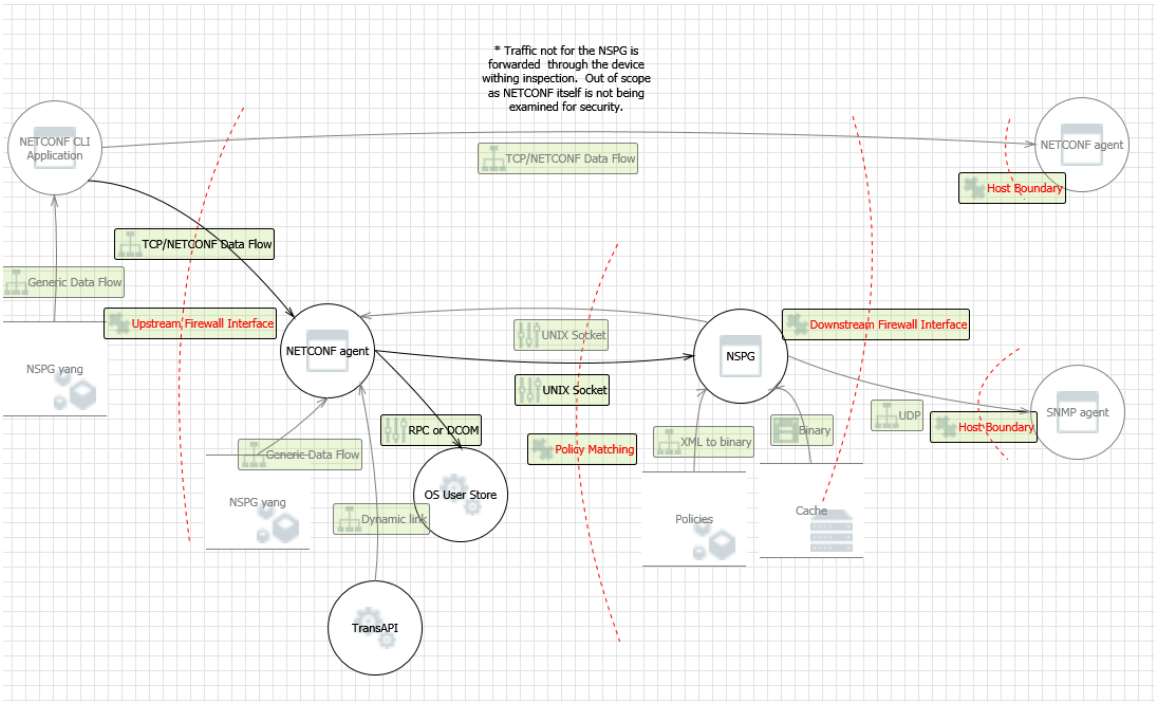

Figure 5.27: High-level threat model of the NSPG approach

From the high-level threat model, three boundaries surrounding the midstream component are present. Starting from the left, the “upstream firewall interface” is the upstream side of the midstream device. In the middle, there is the “policy matching interface” where requests are forwarded from the NETCONF agent component to the remaining NSPG components. Further to the right are the “downstream firewall” and “host boundary” interfaces. These interfaces are present such that a threat model comparable to the one generated for the dual-stack approach can be derived. To illustrate the general flow, packets move from the upstream applications through the midstream device (the NSPG) to the downstream components that are “out of scope” as the downstream side is considered secure. The “in scope” data flow involves data originating from an upstream NETCONF capable host, connecting to the NSPG, and the NSPG acting as a “traffic enforcer” allowing access to specific policies and the NSPG itself performing SNMP operations. Similarly to the dual-stack approach, the faint grey data flows and lighter shaded icons are “out of scope”.

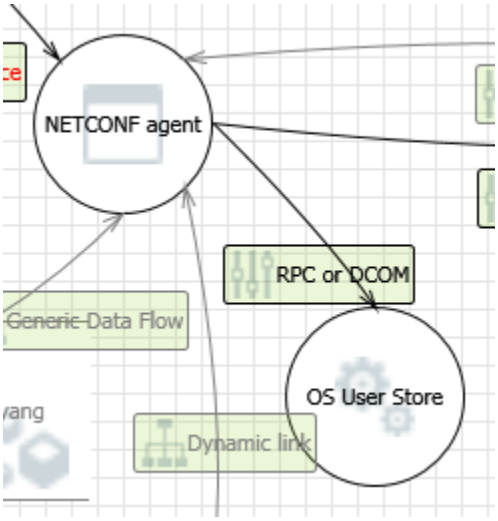

Figure 5.28: NSPG RPC/DCOM data flow figure

In the RPC/DCOM data flow in Figure 5.28, we can see that data traverses from the NETCONF agent to the OS credential data store. There are several potential threats identified by the model: 5 total for this data flow, 2 needs investigation, 1 mitigation implemented, and 2 not applicable. This boundary implies a layer of trust in any application being executed on a system or with the permissions to access credential stores. For example, an application must be given a certain level of trust if it is able to query a user's assigned groups or existence, otherwise, this functionality would not be possible.

- **Elevation Using Impersonation** - This threat assumes that the OS and credential stores are similar in function than that of another application. Fortunately, they are not, and it is largely impossible to hijack them unless a series of active attacks such as replacing general C system libraries or Linux kernel modules (e.g., rootkit attack). For these reasons, we consider this as **not applicable** because the security of the OS is out of scope

- **NETCONF Agent Process Memory Tampered** - The NETCONF agent uses a variety of OS system calls such as socket() or accept(). It also uses pointers and dynamic memory (e.g., malloc'd memory). If improperly sanitized inputs are parsed by vulnerable functions (e.g., strcpy vs. strncpy) or pointer arithmetic is incorrect, then an application may be vulnerable to having runtime memory altered and exploited. This is considered **needs investigation** as

would be normal for any non-managed C application

- **Replay Attacks** - In the NSPG, sequencing and time stamping are not used when querying OS credentials (this is due to the design of the Linux system call API). For this reason, and on the premise that the NETCONF agent is "trusted", we considered this **not applicable**

- **Collision Attacks** - Collision attacks are a type of data tampering where data sent can be overlapped using offsets and "stitching". The concept of collision is not possible for credential API calls as they are sequenced requests (system calls execute in the order received). This is considered **not applicable** for the lack of potential collisions when accessing credentials

- **Weak Authentication Scheme** - The NETCONF agent uses the OS's authentication scheme via the SSH system daemon. It is also very dependent on the password policies employed on the host to ensure the security and strength of a password-based credentials. This is also a problem for downstream hosts if poor credential management is being performed. Regardless, this threat should be considered as **needs investigation**

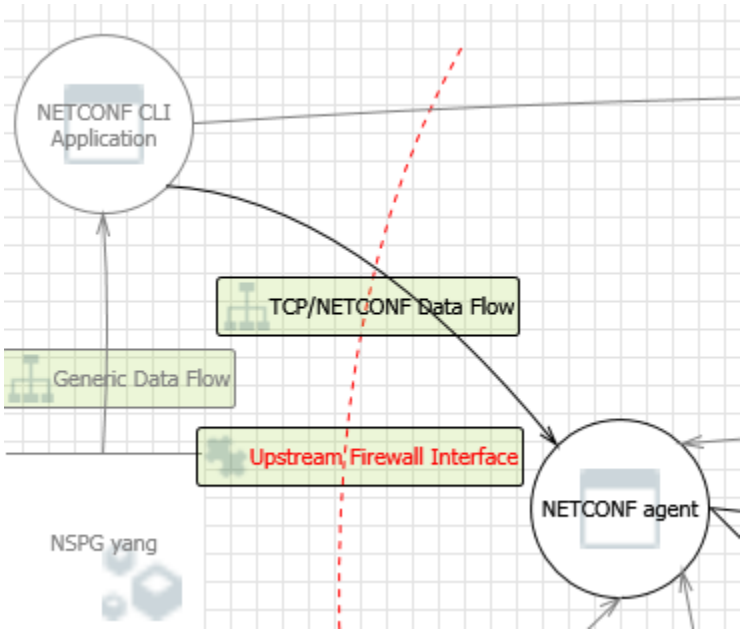

Figure 5.29: NSPG TCP/NETCONF data flow figure

In the TCP/NETCONF data flow in Figure 5.29, data traverses the upstream firewall interface

from the NETCONF CLI to the NETCONF agent (OpenYuma) There are several potential threats identified by the model: 9 total for this data flow, 7 need investigation, and 2 not applicable.

- **Spoofing the the NETCONF agent Process** - As with any host on a network or person in reality, we can spoof an identity unless we provide details only known to them or assumed to be in their possession. For example, Jane calls her credit card company, they also validate her full name, age, address and other details to prevent a malicious person from performing identity theft. The same is true for computers and networked hosts. Having an IP address, does not equate to that IP address equaling that exact host. The NETCONF CLI is responsible to verify that the NETCONF agent is exactly as identified. This is considered as **needs investigation**

- **XML DTD and XSLT Processing** - The data flow contains XML, but the XML is contained within an encrypted SSH connection and unable to be directly tampered with while in transit. We are relying on the third party open-source XML libraries for XML parsing, but vulnerabilities may exist within them. Therefore, we recommend that this threat **needs investigation**

- **Potential Data Repudiation by OS Process** - If the NETCONF agent claims that it did not receive data, logging at the input firewall level could at least record if a TCP handshake had occurred or a connection attempt had been made. This should be considered as **needs investigation**

- **Potential Process Crash or Stop for OS Process** - A crash or disabling of the NSPG due to resource over-utilization can be prevented partially through the usage of packet rate limiting, and monitoring system resources. Flaws in NSPG program logic may be harder to prevent even if a number of auditing measures are performed. This should be considered as **needs investigation**

- **Data Flow TCP/NETCONF Data Flow Is Potentially Interrupted** - This attack is absolutely possible by interrupting a physical network connection or actively interfering with a network stream. Monitoring network activity and securing networks physically should be

performed at a minimum to protect legacy devices. This should be considered as **needs investigation**

- **Elevation Using Impersonation** - From the perspective of the CLI, theoretically the NSPG could exploit an XML vulnerability by returning malicious or erroneous data to any queries from the NETCONF CLI. However, the NETCONF CLI is out of scope and considered **not applicable**

- **OS Process May be Subject to Elevation of Privilege Using Remote Code Execution** - Any remote host connecting to a networked application can "fuzz" or attack any application interface. This attack vector is currently unexplored and the NETCONF agent code has not been thoroughly audited for security and is not contained within a managed environment (e.g., containers). This should be considered as **needs investigation**

- **Elevation by Changing the Execution Flow in OS Process** - Again, any remote host connecting to a networked application can "fuzz", probe and/or attack any application interface. This attack vector is currently unexplored and the NETCONF agent code has not been thoroughly audited for security and is not contained in a managed environment (e.g., containers). If the NETCONF agent is executed as a root level user, then any successful execution of arbitrary code inherits root permissions. This should be considered as **needs investigation**

- **NETCONF CLI Application Process Memory Tampered** - The NETCONF CLI upstream host is written in C and does have a network interface. However, the upstream NETCONF CLI is out of scope and is thoroughly separated via a network interface from the NETCONF agent. This should be considered as **not applicable**

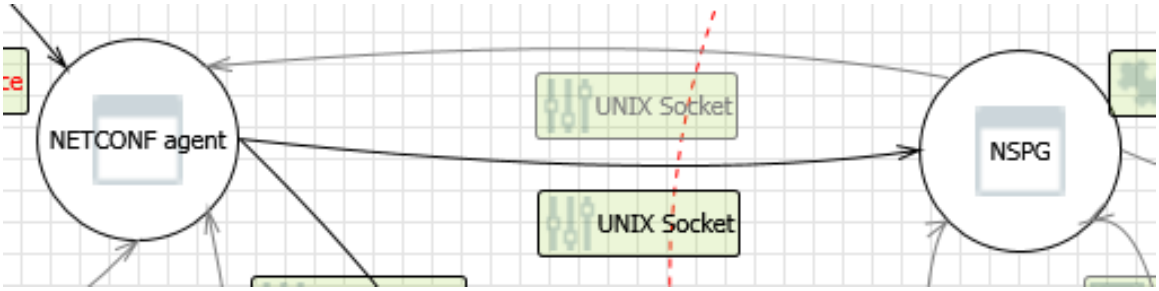

Figure 5.30: NSPG UNIX socket data flow figure

In the UNIX socket data flow in Figure 5.29, data traverses the upstream firewall interface from the NETCONF agent (transAPI module) to the remaining NSPG components (the data store, authentication and SNMP agent). There is a number of potential threats identified by the model: 13 total for this data flow, 7 needs investigation, 2 mitigation implemented and 4 not applicable.

- **Potential Data Repudiation by OS Process** - Due to the design of the NSPG components and usage of IPC, this is not possible. However, logs could be generated and linked to requests originating from any TransAPI module. This should be considered as **needs investigation**
- **Potential Lack of Input Validation for OS Process** - Traffic flowing across the IPC interface is not encrypted due to trust of the security provided by the parent system. We consider it low risk, but this threat **needs investigation**
- **Elevation Using Impersonation** - While possible, it makes little sense for the remaining NSPG components to impersonate the NETCONF agent when residing on the same host. However, an attacker's motives may be to attack the upstream NETCONF CLI or NMS. We consider this as **needs investigation**
- **Spoofing the OS Process Process** - Again, while possible, it makes little sense for the remaining NSPG components to impersonate any other component or the host itself. We consider this as **needs investigation** as the security of the midstream host is important should the midstream host be compromised then used as a platform to downstream legacy hosts.
- **Elevation by Changing the Execution Flow in OS Process** - If incoming input data are parsed for validity and sanitized, then a parsing-related vulnerability should have minimal

risk. The input data are parsed for sanity and we consider this as **not applicable**

- **OS Process May be Subject to Elevation of Privilege Using Remote Code Execution** - If incoming input data is parsed for validity and sanitized, then a parsing related vulnerability should have minimal risk. Parsing and locking down permissions can always be improved or the application could be deployed in a "jail". The input data are already parsed for sanity against the NSPG YANG module and we consider this as **not applicable**.

- **Data Flow UNIX Socket Is Potentially Interrupted** - If data flowing across the UNIX socket is interrupted, there is a major possibility that a system error has occurred or the system has been compromised ("owned"). Therefore we consider this as **not applicable** as a breach in the security and trust of the host system is required as a precursor to this threat

- **Potential Process Crash or Stop for OS Process** - It is unwise to assume a process can continuously operate for years (although possible) without faults. For critical systems, it could be considered a best practice to implement a watchdog or similar facility to monitor the status of the NSPG. As a result, we consider this as **not applicable**

- **Data Flow Sniffing** - If data flowing across the UNIX socket is sniffed, the system is compromised ("owned"). There is also little chance to detect this behavior as well as the host systems integrity has been breached. Therefore, we consider this as **not applicable** as a breach in the security and trust of the host system is required as a precursor to this threat

- **OpenYuma Process Memory Tampered** - Transferring packets across a UNIX socket requires memory copies to be made internally. No pointers are exchanged to memory-mapped locations as well. Therefore, we consider this threat as **not applicable**

- **Replay Attacks** - In the NSPG, no form of sequencing or time stamping is used in the policy queries. UNIX sockets are effectively a memory mapped message queue/pipe hybrid that is limited to "trusted" localhost traffic and OS file permissions (e.g., everything in Linux is a file). For these reasons, and on the premise that the NETCONF agent is "trusted" as well, we considered this **mitigation implemented**

- **Collision Attacks** - Collision attacks are a type of data tampering where data sent can be overlapped using offsets and "stitching" to bypass controls or exploit a vulnerability. In the NSPG, a single query is fit into a single request packet and inserted into a synchronized queue via the internals of a UNIX socket. The concept of collision is not possible, and considered as **not applicable**

- **Weak Authentication Scheme** - The NETCONF agent (OpenYuma) uses the host's native OS authentication scheme. Therefore, the NSPG is closely dependent on the password policies employed on the host to ensure the security and strength of a password-based credentials. This is also a problem for downstream hosts if poor credential management is being performed within the organization overall. The reliance on system password policy strength renders this threat as **needs investigation**

Similarly to the dual-stack threat model, reliance on the trust of a system or technology is required for the NSPG to provide a level of improved security. However, the main difference between the dual-stack and NSPG threat model is that the dual-stack transfers security risk to legacy hosts when firewall ACL limitations are exceeded. This is exacerbated when packets bypass the ACLs and are forwarded (without inspection) onto the legacy hosts. Using the NSPG approach, the NSPG funnels all incoming NETCONF queries to itself, and the authenticated user (the one who successfully initiated a NETCONF session) can then query the NSPG for information/policies assigned to his/her user. The NSPG initiates all downstream SNMP communications and upstream SNMP traffic is never allowed to either directly or indirectly access downstream legacy hosts (validated in the low-level tests in Section 5.2.1).

5.3 Testing Insights

This section is dedicated to describing and summarizing any observations made when examining firewalls in a dual-stack configuration versus the NSPG. Given the volume of technical information and presented in the previous result sections, we derived four insights regarding the security of the NSPG solution.

5.3.1 Insight 1 – Stateful Packet Filtering is Insufficient By itself

With the advent of networks, network professionals quickly realized that devices or a technology would be needed to limit access to hosts and networks. Originally, these devices provided access control by simply matching host IP addresses and ports for allowing and dropping connections. These lists were called ACLs and used static packet filtering. However, these ACLs were easily bypassed and stateful packet filtering was developed to extend filtering to include protocol handshaking and connection tracking. Unfortunately, packet filtering even with connection awareness is not effective for blocking spoofed packets. As seen in Section 5.2.2, an attacker can inject packets into a stream and bypass ACLs even with state tracking. Therefore, protocol awareness through DPI and use of technologies such as the NSPG is required.

5.3.2 Insight 2 – Dual-stack Approaches Reduce Interruption Risk, but with Limitations

If we imagine Jane the administrator once again, Jane (if doing her job correctly) should not needlessly interrupt or create interruptions when performing network maintenance and modifications. Assuming a mandate of "doing no harm", Jane may use technologies that are simpler to configure at the expense of security or cost IF the technology has a lower probability of creating additional work or inhibiting her daily duties. If Jane has to support two different eras of devices with two different protocols, then legacy (and insecure protocols such as SNMP) will remain active despite any known or unknown security vulnerabilities. In this case, Jane may resort to using firewalls with a dual-stack approach. Seemingly firewalls in dual-stack approaches can improve security with a low risk of interruption in the following scenarios:

- Can be used to limit ingress/egress traffic using ACLs
- Reduce attack surfaces
- Used to hide hosts behind the firewall on the "secure" side (masquerade/NAT)

Unfortunately, attackers can probe firewalls and determine ACLs by learning information about the network over time or through techniques such as social engineering. ACLs provide security by

obscurity, however, when is only an inconvenience for an attacker if he or she is determined. Then attackers can spoof packets (Section 5.3.1 and Section 5.2.1) and the stateful ACLs can be bypassed; particularly when legacy protocols are operating in a dual-stack configuration.

Therefore, additional technologies such as DPI can be used to match traffic according to policies, but this is often too complex for engineers or administrators (and has a high-risk of process disruption). In fact, using dual-stack approaches increase the number of firewall ACL rules, but could be minimized by consolidating legacy protocols using gateways or proxies.

Alternatively, insecure protocols traversing the firewall are at risk of being intercepted or having clear-text credentials discovered while in transit. The usage of firewalls does not mitigate the risk of interception and requires VPNs on the upstream (at a minimum) to provide confidentiality when none exists (e.g., SNMPv2).

Therefore, we conclude using the results in Section 5.2.2, that dual-stack approaches are insufficient in heterogeneous environments requiring improved security controls and seamless operation. However, we acknowledge that they have a low risk of interrupting network operation (and minimal costs) if deployed correctly when combined with other technologies such as VPNs and DPI.

5.3.3 Insight 3 – Firewalls Indirectly Rely on the Robustness of Legacy Hosts

From insights 1 and 2, firewalls are insecure when used as the primary or only form of security. Jane the administrator and her employers trust that firewalls imply an adequate level of security for the networks/devices being protected. This could not be farther from the truth because firewalls are like any other software or device and may have any number of flaws. In consumer or even enterprise environments, devices can up updated regularly or have shorter life cycles. Alternatively, in industrial or critical infrastructure, devices are often engineered for environmental resilience, long life cycles (excess of 5-10 year deployments), and performance constraints. In many situations, devices in this domain are developed with security/robustness as an afterthought to business drivers (customers are not willing to pay or upgrade) and overall security may be sufficiently lower than what is expected from modern commodity electronics/software.

An optimistic version of Jane may rightfully conclude that these “weaker” industrial devices should be replaced or upgraded with more modern software and hardware. However, business and

technological motivations may be insufficient and upgrade paths for providing a secure transition may be unfeasible to deploy.

Let us imagine again Jane's network. Jane is working in the pulp and paper industry, which today in Canada is not a thriving commodity as it once was. It is a competitive market with low profit margins, and ever looming threats of permanent facility shut downs. For management, as long as profits are being generated, pulp and paper facilities remain in operation. As a result, replacing hardware or modernizing networks/process control are sensitive topics and outweighs the cost of security, which further delays modernization or security improvements indefinitely unless a sufficient business motivator is found.

Many organizations such as the one that employs Jane, assume that firewalls are trusted for security by being merely present. The implicit trust of firewalls distributes the responsibility of security and robustness onto the legacy hosts that the firewall was supposed to be originally protecting. This transference of trust implies that firewalls are not a complete replacement for secure devices, correct configuration and secure technology transitions. Nor do firewalls in a dual-stack configuration have any effect on the risk of an attacker exploiting legacy hosts with vulnerabilities once bypassed; even if the firewall itself is not compromised as seen in Section 5.2.2. Again, to provide adequate protection, DPI and proxy/protocol gateways are required to provide adequate protection against invalid or policy violating traffic. VPNs may improve confidentiality, but do not protect already deployed devices if upstream networks/trusted hosts are compromised.

5.3.4 Insight 4 – The NSPG as an Interface Barrier is Superior to Dual-stack Approaches

As an alternative solution to dual-stack firewall approaches, we demonstrated that the NSPG when used at gateway locations provides a barrier interrupting network traffic and transparently provides policy matching, thereby providing superior security than that of a dual-stack firewall approach when protecting legacy devices.

Beyond insights 1-3, Table 5.4 and Table 5.5 highlight a number of differences between the two approaches (dual-stack and NSPG) threat models. Overall, the NSPG fares better in terms of total number of threats, number of valid threats, and has one fewer interfaces when compared to the

dual-stack approach.

Dual-Stack Interfaces	Needs Investigation	Mitigated	N/A	Total threats	Valid threats
TCP/NETCONF (up)	1	2	6	9	1
TCP/NETCONF (down)	5	0	4	9	5
UDP/SNMP (up)	9	0	2	11	9
UDP/SNMP (down)	7	0	4	11	7
Total				**40**	**22**

Table 5.4: Dual-stack threat model summary

NSPG Interfaces	Needs Investigation	Mitigated	N/A	Total threats	Valid threats
RPC/DCOM	2	1	2	5	2
TCP/NETCONF (up)	7	0	2	9	7
UNIX socket	7	2	4	13	7
Total				**27**	**16**

Table 5.5: NSPG threat model summary

Although the architectural paradigm of the NSPG is different from that of a dual-stack approach, the NSPG prevents unauthenticated requests from flowing through the NSPG and validates all sessions. Once a session is established using NETCONF/SSH, a user such as Jane can query policies without ever directly accessing the legacy devices. The NSPG parses the incoming queries, validates them against the NSPG YANG model, and queries its own data store using another messaging barrier (UNIX sockets). According to the policies known to the NSPG, the data-store is populated by the NSPG generating valid SNMP connections to the legacy devices, and storing the responses.

This separation of legacy device configurations and queries can be used to hide security credentials or to only execute specific commands WITHOUT modifying legacy devices. This way, an upstream connection can NEVER directly connect to a legacy device without the query being filtered or having to bypass three different interfaces. In addition, legacy devices NEVER require modification. If there are firewalls upstream of the NSPG, an administrator would only be required to allow NETCONF and not both SNMP and NETCONF. This would also simplify firewall ACLs and reduce potential future bypasses when/if firewall ACLs are forgotten over time.

Additionally, the use of the NSPG does not require a significant increase in degree of hardware resources when compared to those of a dual-stack approach. In many cases, the functionality can be integrated into an existing firewall or router. In situations where separate hardware is required,

security requirements can be satisfied with very modest equipment (e.g., a RaspberryPi or a small single-board embedded computer).

Hardware costs should be feasible for ubiquitous deployments once developed to a professional product.

In Table 5.6, we sorted all of the threats associated with the threat model interfaces. Firstly, there is a shift of threats from downstream hosts to the NSPG in the form of software-related threats. This is to be expected as packets are not forwarded through the firewall and are instead sent to the NSPG if a connection is possible. Secondly, when directly comparing the dual-stack approach to the NSPG approach, we can clearly see a reduction of direct interfaces; the NSPG only shares the TCP/NETCONF upstream interface and the other network interfaces are removed (TCP/NETCONF) downstream, UDP/SNMP upstream, UDP/SNMP downstream).

Threat	DS TCP NETCONF Up	NSPG TCP/NETCONF Up	DS TCP NETCONF Down	DS UDP SNMP Up	DS UDP SNMP Down	NSPG RPC/DCOM	NSPG UNIX socket
XML DTD and XSLT Processing	NA		NI			NI	NI
Spoofing the OS Process	NI	NI		NI		NI	
Potential Process Crash or Stop for Kernel	MT						
Data Flow Generic Data Flow Is Potentially Interrupted	NI	NA	NA				
Elevation Using Impersonation	NA	NA	NA	NI	NA	NA	
Kernel - Elevation of Privilege Using Remote Code Exec'	NA	NA	NA		NI	NA	NA
Elevation by Changing the Execution Flow in Kernel	NA	NA	MT				
Elevation by Changing the Execution Flow in OS Process	NA	NA		NA			
Process - Elevation of Privilege Using Remote Code Exec'	NA		NI	MT			NI
Kernel Process Memory Tampered		NA	NA	MT			
Data Flow TCP/NETCONF Is Potentially Interrupted		NA	NA	NA			
Potential Process Crash or Stop for OS Process		NA	NA			NI	NI
Potential Data Repudiation by OS Process		NA	NI	NI	NI	NI	NI
Potential Lack of Input Validation for OS Process		NA	NI	NI		NI	NI
Spoofing the Kernel Thread or Process	MT	NA	NA	NA			
Data Flow UDP/SNMP Is Potentially Interrupted	NA		NA	NI			
Data Flow Sniffing		NA	NA	NI			
Potential Data Repudiation by Kernel		NA	NI	NI			
Spoofing Process		NA		NI			
Process Memory Tampered		NA		NI			
Potential Lack of Input Validation for Kernel		NA		NA	NI	NA	
Replay Attacks		NA		NA			
Collision Attacks	NA			NA			
Weak Authentication Scheme	NA				NA		
Data Flow UNIX Socket Is Potentially Interrupted		NA			NI		

NA = Not Applicable, MT = Mitigation Implemented, NI = Investigation Needed

Grey column highlighted to demonstrate overlap

Table 5.6: Summary of threats per interface

Chapter 6

Conclusion and Future Work

Ultimately, we determined that the dual-stack approach is insufficient in many areas when being used as the primary source defence for security. In any approach, there is an implied trust in the host and technology. In the dual-stack approach (even if the firewall is not compromised), an intermediate level attacker can bypass ACLs easily to access legacy downstream devices, placing the responsibility for their security back onto them. Therefore the dual stack/firewall approach gives a false sense of security when relying on firewalls for the sole security for legacy devices.

We demonstrated that adding the NSPG to existing networks only increases complexity minimally; administrators only need to configure their upstream NMS to use NETCONF and configure NSPG policies. Upstream hosts connect to the NSPG via NETCONF over SSH, which provides security and secondly, authenticated users via NETCONF sessions can only access information authorized to them via policies. This allows administrators to mix-and-match policies on a per-role basis, improving security controls where they may be insufficient, and integrating into modern management paradigms (e.g., NETCONF-enabled NMS and SDNs).

Additionally, we demonstrated that the NSPG can be used transparently in heterogeneous networks without modifying existing legacy endpoint devices. Existing firewalls only need to support a single ACL exception versus two in a dual-stack approach (e.g., NETCONF instead of NETCONF and SNMP for each host/subnet).

Unlike the dual-stack approach, insecure SNMP does not traverse the NSPG in either direction nor is it tunneled through a VPN. Upstream connections leverage NETCONF over SSH, which

prevents sniffing of clear-text traffic on insecure networks. On the downstream side of the NSPG, only valid SNMP traffic can be generated (given the configured policies) when communicating to legacy devices on the secure side of the device. The NSPG also provides caching functionality that could prevent DoS on vulnerable legacy devices similarly to rate-limiting on firewalls using ACL rules.

Overall, we feel that the NSPG is superior in many ways over ACLs and no major technical hurdles exist to prevent deployment on low-cost devices. Having said that, future work is required for real-world testing, and a thorough security analysis of all software (and host OS) to ensure appropriate operation in ICS networks. It is suspected that further work is required to further strengthen the use case of the NSPG such as limiting application-level privileges (CERT, 2015). Alternatively, using sandboxes or Docker-esque containers to protect the NSPG may be a solution to reduce any threat attack surfaces (Linux Man-pages Project, 2015a) (Docker, 2016). Additionally, the NSPG may be of interest to IETF working groups such as Secure Inter-Domain Routing (sidr) and Interface to the Routing System (i2rs), and organizations concerned about legacy technology transitions and security (*Secure Inter-Domain Routing (sidr)*, 2015) (*Interface to the Routing System (i2rs) Charter*, 2015).

Appendix A

Appendix

A.1 SNMPD Host installation

As part of the processing of exploring some infrastructure for my thesis - I noticed that the SNMPD documentation needed some love despite the many eyes of the Internet. Here is how I set snmpd up in Ubuntu 15.10 for localhost access to IP-related information contained within the IP MIB.

Install the following:

```
sudo apt-get install snmp snmp-mibs-downloader snmpd libsnmp-dev
```

Edit the following configuration - this allows you to use 'proprietary' MIBs which may or may not abide by GPL etc..

```
sudo vi /etc/snmp/snmp.conf
```

Comment out the last line (mibs :) so that it looks as below:

```
# As the snmp packages come without MIB files due to license reasons, loading
# of MIBs is disabled by default. If you added the MIBs you can reenable
# loading them by commenting out the following line.
#mibs :
Edit the following snmpd daemon configuration - by noting: "## ^^ EDIT HERE":

sudo vi /etc/snmp/snmpd.conf

###############################################################################
#
# EXAMPLE.conf:
#   An example configuration file for configuring the Net-SNMP agent ('snmpd')
#   See the 'snmpd.conf(5)' man page for details
```

```
#
#  Some entries are deliberately commented out, and will need to be explicitly activated
#
###############################################################################
#
#  AGENT BEHAVIOUR
#
#  Listen for connections from the local system only
agentAddress  udp:127.0.0.1:161

## ^^ EDIT HERE

#  Listen for connections on all interfaces (both IPv4 *and* IPv6)
#agentAddress udp:161,udp6:[::1]:161

###############################################################################
#
#  SNMPv3 AUTHENTICATION
#
#  Note that these particular settings don't actually belong here.
#  They should be copied to the file /var/lib/snmp/snmpd.conf
#     and the passwords changed, before being uncommented in that file *only*.
#  Then restart the agent

#  createUser authOnlyUser  MD5 "remember to change this password"
#  createUser authPrivUser  SHA "remember to change this one too"  DES
#  createUser internalUser  MD5 "this is only ever used internally, but still change the password"

#  If you also change the usernames (which might be sensible),
#  then remember to update the other occurances in this example config file to match.

###############################################################################
#
#  ACCESS CONTROL
#

                                        #  system + hrSystem groups only
view    systemonly  included   .1.3.6.1.2.1.1
view    systemonly  included   .1.3.6.1.2.1.25.1

                                        #  Full access from the local host
 rocommunity public  localhost

## ^^ EDIT HERE
                                        #  Default access to basic system info
 rocommunity public  default    -V systemonly
                                        #  rocommunity6 is for IPv6
 rocommunity6 public  default   -V systemonly

                                        #  Full access from an example network
                                        #     Adjust this network address to match your local
                                        #     settings, change the community string,
                                        #     and check the 'agentAddress' setting above
#rocommunity secret  10.0.0.0/16
```

```
                                        #  Full read-only access for SNMPv3
 rouser    authOnlyUser
                                        #  Full write access for encrypted requests
                                        #     Remember to activate the 'createUser' lines above
#rwuser    authPrivUser    priv

#  It's no longer typically necessary to use the full 'com2sec/group/access' configuration
#  r[ow]user and r[ow]community, together with suitable views, should cover most requirements

###############################################################################
#
#  SYSTEM INFORMATION
#

#  Note that setting these values here, results in the corresponding MIB objects being 'read-only'
#  See snmpd.conf(5) for more details
sysLocation     Sitting on the Dock of the Bay
sysContact      Me <<a href="mailto:me@example.org">me@example.org</a>>
                                        # Application + End-to-End layers
sysServices     72

#
#  Process Monitoring
#
                                # At least one  'mountd' process
proc  mountd
                                # No more than 4 'ntalkd' processes - 0 is OK
proc  ntalkd    4
                                # At least one 'sendmail' process, but no more than 10
proc  sendmail 10 1

#  Walk the UCD-SNMP-MIB::prTable to see the resulting output
#  Note that this table will be empty if there are no "proc" entries in the snmpd.conf file

#
#  Disk Monitoring
#
                                # 10MBs required on root disk, 5% free on /var, 10% free on all other disks
disk        /       10000
disk        /var    5%
includeAllDisks  10%

#  Walk the UCD-SNMP-MIB::dskTable to see the resulting output
#  Note that this table will be empty if there are no "disk" entries in the snmpd.conf file

#
#  System Load
#
                                # Unacceptable 1-, 5-, and 15-minute load averages
load   12 10 5

#  Walk the UCD-SNMP-MIB::laTable to see the resulting output
#  Note that this table *will* be populated, even without a "load" entry in the snmpd.conf file

###############################################################################
```

```
#
#  ACTIVE MONITORING
#

                                    #   send SNMPv1  traps
 trapsink     localhost public
                                    #   send SNMPv2c traps
#trap2sink    localhost public
                                    #   send SNMPv2c INFORMs
#informsink   localhost public

#  Note that you typically only want *one* of these three lines
#  Uncommenting two (or all three) will result in multiple copies of each notification.

#
#  Event MIB - automatically generate alerts
#
                                   # Remember to activate the 'createUser' lines above
iquerySecName   internalUser
rouser          internalUser
                                   # generate traps on UCD error conditions
defaultMonitors          yes
                                   # generate traps on linkUp/Down
linkUpDownNotifications  yes

##############################################################################
#
#  EXTENDING THE AGENT
#

#
#  Arbitrary extension commands
#
 extend    test1   /bin/echo  Hello, world!
 extend-sh test2   echo Hello, world! ; echo Hi there ; exit 35
#extend-sh test3   /bin/sh /tmp/shtest

#  Note that this last entry requires the script '/tmp/shtest' to be created first,
#    containing the same three shell commands, before the line is uncommented

#  Walk the NET-SNMP-EXTEND-MIB tables (nsExtendConfigTable, nsExtendOutput1Table
#     and nsExtendOutput2Table) to see the resulting output

#  Note that the "extend" directive supercedes the previous "exec" and "sh" directives
#  However, walking the UCD-SNMP-MIB::extTable should still returns the same output,
#     as well as the fuller results in the above tables.

#
#  "Pass-through" MIB extension command
#
#pass .1.3.6.1.4.1.8072.2.255  /bin/sh       PREFIX/local/passtest
#pass .1.3.6.1.4.1.8072.2.255  /usr/bin/perl PREFIX/local/passtest.pl

# Note that this requires one of the two 'passtest' scripts to be installed first,
#    before the appropriate line is uncommented.
```

```
# These scripts can be found in the 'local' directory of the source distribution,
#     and are not installed automatically.

#  Walk the NET-SNMP-PASS-MIB::netSnmpPassExamples subtree to see the resulting output

#
#  AgentX Sub-agents
#
                                        #  Run as an AgentX master agent
 master           agentx
                                        #  Listen for network connections (from localhost)
                                        #    rather than the default named socket /var/agentx/master
#agentXSocket     tcp:localhost:705

## ADD THE FOLLOWING TWO LINES:

view    systemonly  included   .1.3.6.1.2.1.4.20
view    systemonly  include    .1.3.6.1.2.1.4.31

##
```

If the last two above lines above are not included, and including the line: rocommunity public localhost, you will receive messages about time-outs from the localhost and/or the OID not existing. Next download the collection of MIBs in addition to the ones packaged by the upstream maintainers:

```
sudo download-mibs
```

Then restart the daemon

```
sudo service snmpd restart
```

To test the output, use snmpwalk to output the information provided by the system

```
snmpwalk -v 1 -c public localhost ipadd
```

It should output information like this:

```
IP-MIB::ipAdEntAddr.10.240.200.59 = IpAddress: 10.240.200.59
IP-MIB::ipAdEntAddr.127.0.0.1 = IpAddress: 127.0.0.1
IP-MIB::ipAdEntIfIndex.10.240.200.59 = INTEGER: 3
IP-MIB::ipAdEntIfIndex.127.0.0.1 = INTEGER: 1
IP-MIB::ipAdEntNetMask.10.240.200.59 = IpAddress: 255.255.252.0
IP-MIB::ipAdEntNetMask.127.0.0.1 = IpAddress: 255.0.0.0
IP-MIB::ipAdEntBcastAddr.10.240.200.59 = INTEGER: 1
IP-MIB::ipAdEntBcastAddr.127.0.0.1 = INTEGER: 0
```

A.1.1 SNMPv3 Setup and Test

Run the following commands:

```
sudo service snmpd stop
sudo net-snmp-config --create-snmpv3-user -a "my_password" MD5User
sudo net-snmp-create-v3-user -A "my_password" -x "my_password" -a SHA -x AES SHA1User
sudo vi /etc/snmp/snmpd.conf
```

Edit the contents to have the following present and uncommented:

```
createUser authOnlyUser MD5
rocommunity public demopublic
sudo service snmpd start
```

Then make the src in the snmpv3 test app folder

```
cd TODO
make
./snmpdemoapp
```

Which should output something like:

```
SNMPv2-MIB::sysDescr.0 = STRING: Linux ackbar 4.2.0-35-generic #40-Ubuntu SMP Tue Mar 15 22:15:45 UTC 2016 x86_64
value #1 is a string: Linux ackbar 4.2.0-35-generic #40-Ubuntu SMP Tue Mar 15 22:15:45 UTC 2016 x86_64
```

```
rbrash@ackbar:~/playground_thesis/nspg$ snmpset -c private -v 2c localhost 1.3.6.1.2.1.1.5.0 s "nogoat"
SNMPv2-MIB::sysName.0 = STRING: nogoat
rbrash@ackbar:~/playground_thesis/nspg$ snmpget -c private -v 2c localhost 1.3.6.1.2.1.1.6.0
SNMPv2-MIB::sysLocation.0 = STRING: Sitting on the Dock of the Bay
rbrash@ackbar:~/playground_thesis/nspg$ snmpget -c private -v 2c localhost 1.3.6.1.2.1.1.5.0
SNMPv2-MIB::sysName.0 = STRING: nogoat
```

A.2 OpenYuma Host Installation

Here is my guide for a quickstart instead of reading their documentation:

Install Prerequisites

```
# sudo apt-get update
# sudo apt-get install -y git autoconf gcc libtool libxml2-dev libssl-dev make libncurses5-dev libssh2-1-dev openssh-server build-essentials
```

Reconfigure SSHD

Edit sshdconfig and under the line Port 22, add the following two lines

```
# sudo vi /etc/ssh/sshd_config

# What ports, IPs and protocols we listen for
Port 22
Port 830
Subsystem netconf /usr/sbin/netconf-subsystem --ncxserver-sockname=830@/tmp/ncxserver.sock
```

Restart SSH with the following command

```
sudo service ssh restart
```

Download, build and install OpenYuma

GIT clone, make and install the following - netconfd will run with your current user

```
# git clone <a href="https://github.com/OpenClovis/OpenYuma.git">https://github.com/OpenClovis/OpenYuma.git</a> openYuma
# cd openYuma
# make
# sudo make install
# sudo cp /etc/yuma/netconfd-sample.conf /etc/yuma/netconfd.conf
# /usr/sbin/netconfd --superuser=`whoami`
```

Connecting with yangcli

Open a second terminal and run the command:

```
# yangcli
Something like the following should appear - note the
 YOURUSERNAME and YOURPASSWORD variables which need to be
  replaced with something more appropriate

yangcli> connect server=localhost user=YOURUSERNAME password=YOURPASSWORD
val->res is NO_ERR.
yangcli: Starting NETCONF session for rbrash on localhost

NETCONF session established for rbrash on localhost

Client Session Id: 2
Server Session Id: 1

Server Protocol Capabilities
   base:1.0
   candidate:1.0
   confirmed-commit:1.0
   rollback-on-error:1.0
   validate:1.0
   url:1.0
   xpath:1.0
   notification:1.0
   interleave:1.0
   partial-lock:1.0
   with-defaults:1.0
   base:1.1
   validate:1.1
   confirmed-commit:1.1

Server Module Capabilities
   iana-crypt-hash@2014-04-04
      Features:
         crypt-hash-md5
         crypt-hash-sha-256
         crypt-hash-sha-512
   ietf-inet-types@2013-07-15
```

```
    ietf-netconf-acm@2012-02-22
    ietf-netconf-monitoring@2010-10-04
    ietf-netconf-partial-lock@2009-10-19
    ietf-netconf-with-defaults@2011-06-01
    ietf-system@2014-08-06
        Features:
            radius
            authentication
            local-users
            radius-authentication
            ntp
            ntp-udp-port
            timezone-name
            dns-udp-tcp-port
    ietf-yang-types@2013-07-15
    nc-notifications@2008-07-14
    notifications@2008-07-14
    yuma-app-common@2012-08-16
    yuma-arp@2012-01-13
    yuma-mysession@2010-05-10
    yuma-ncx@2012-01-13
    yuma-proc@2012-10-10
    yuma-system@2014-11-27
    yuma-time-filter@2011-08-13
    yuma-types@2012-06-01

Server Enterprise Capabilities
    None

Protocol version set to: RFC 6241 (base:1.1)
Default target set to: <candidate>
Save operation mapped to: commit
Default with-defaults behavior: explicit
Additional with-defaults behavior: trim,report-all,report-all-tagged

Checking Server Modules...
```

A.3 NSPG Host Installation

The NSPG requires no additional steps except for installing the NSPG yang model, NSPG transapi module and the appropriate firewall rules.

```
sudo iptables -I INPUT -p udp --dst-port 161 -j ACCEPT
sudo iptables -I OUTPUT -j ACCEPT
sudo iptables -I INPUT -p tcp --dst-port 830 -j ACCEPT
```

A.4 Test Environment Specifications

The RaspberryPi is a B+ v2 - this is used by the end-hosts and midstream host It operates at 700Mhz and has 512 MB RAM. It is advertised as 10/100, but actual line-rate speeds are limited by the USB controller.

The remaining host is an Dell Precision M4700, Intel Core i7-3740QM 2.70GHz Processor, 16GB of RAM, 512 GB SSD. It uses a dedicated PCI-e Ethernet port for 10/100/1000 MB. It ran vanilla 15.10 Ubuntu x84-64.

The bridge USB Ethernet ports were: TRENDnet USB 3.0 to Gigabit Ethernet LAN Wired Network Adapter for Windows, Mac, Chromebook, Linux, and Specific Android Tablets, ASIX AX88179 Chipset, TU3-ETG

A.5 Complete NSPG Policy Listing

```
<?xml version="1.0"?>
<config>
<host>
        <uuid>eeeea62-f6d5-453d-912c-104431f1c101</uuid>
        <address>192.168.1.10</address>
        <security>
                <version>3</version>
                <user>SHA1User</user>
                <authpass>my_password</authpass>
                <authmethod>SHA</authmethod>
                <privpass>my_password</privpass>
                <privmethod>AES</privmethod>
                <community></community>
        </security>
        <task>
                <op>
                        <type>set</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data>goat</data>
                </op>
                <op>
                        <type>get</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data></data>
                </op>
                <op>
                        <type>set</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
```

```
                        <format>string </format>
                        <data>not-goat </data>
                </op>
                <op>
                        <type>get </type>
                        <oid >1.3.6.1.2.1.1.5.0 </oid>
                        <format>string </format>
                        <data ></data>
                </op>
        </task>
</host>
<host>
        <uuid>eeeea62-f6d5 -453d-912c -104431f1c102 </uuid>
        <address >192.168.1.40 </address>
        <security>
                <version >3</version>
                <user>SHA1User </user>
                <authpass >my_password </authpass>
                <authmethod >SHA</authmethod>
                <privpass >my_password </privpass>
                <privmethod >AES</privmethod>
                <community ></community>
        </security>
        <task>
                <op>
                        <type>get </type>
                        <oid >1.3.6.1.2.1.1.5.0 </oid>
                        <format>string </format>
                        <data ></data>
                </op>
        </task>
</host>
<host>
        <uuid>eeeea62-f6d5 -453d-912c -104431f1c103 </uuid>
        <address >192.168.1.30 </address>
        <security>
                <version >2c</version>
                <user ></user>
                <community >private </community>
                <authpass >my_password </authpass>
                <authmethod ></authmethod>
                <privpass ></privpass>
                <privmethod ></privmethod>
        </security>
        <task>
                <op>
                        <type>set </type>
                        <oid >1.3.6.1.2.1.1.5.0 </oid>
                        <format>string </format>
                        <data>goat </data>
                </op>
                <op>
                        <type>get </type>
                        <oid >1.3.6.1.2.1.1.5.0 </oid>
                        <format>string </format>
```

```
                <data></data>
            </op>
            <op>
                <type>set</type>
                <oid>1.3.6.1.2.1.1.5.0</oid>
                <format>string</format>
                <data>not-goat</data>
            </op>
            <op>
                <type>get</type>
                <oid>1.3.6.1.2.1.1.5.0</oid>
                <format>string</format>
                <data></data>
            </op>
        </task>
</host>
<host>
        <uuid>eeeea62-f6d5-453d-912c-104431f1c104</uuid>
        <address>192.168.1.30</address>
        <security>
            <version>1</version>
            <user></user>
            <community>public</community>
            <authpass></authpass>
            <authmethod></authmethod>
            <privpass></privpass>
            <privmethod></privmethod>
        </security>
        <task>
            <op>
                <type>get</type>
                <oid>1.3.6.1.2.1.1.5.0</oid>
                <format>string</format>
                <data></data>
            </op>
            <op>
                <type>get</type>
                <oid>1.3.6.1.2.1.1.5.0</oid>
                <format>string</format>
                <data></data>
            </op>
        </task>
</host>
<host>
        <uuid>eeeea62-f6d5-453d-912c-104431f1c105</uuid>
        <address>192.168.1.50</address>
        <security>
            <version>1</version>
            <user></user>
            <community>private</community>
            <authpass>my_password</authpass>
            <authmethod></authmethod>
            <privpass></privpass>
            <privmethod></privmethod>
        </security>
```

```
        <task>
                <op>
                        <type>get</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data></data>
                </op>
        </task>
</host>
<host>
        <uuid>eeeea62-f6d5-453d-912c-104431f1c106</uuid>
        <address>192.168.1.40</address>
        <security>
                <version>1</version>
                <user></user>
                <community>private</community>
                <authpass>my_password</authpass>
                <authmethod></authmethod>
                <privpass></privpass>
                <privmethod></privmethod>
        </security>
        <task>
                <op>
                        <type>set</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data>goat</data>
                </op>
                <op>
                        <type>get</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data></data>
                </op>
                <op>
                        <type>set</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data>not-goat</data>
                </op>
                <op>
                        <type>get</type>
                        <oid>1.3.6.1.2.1.1.5.0</oid>
                        <format>string</format>
                        <data></data>
                </op>
        </task>
</host>
</config>
```

A.6 Complete NSPG YANG

```
// ------------------------------------------------------------------
//  module nspg
// ------------------------------------------------------------------
module nspg {

    yang-version 1;

    namespace
      "http://concordia.ca/nspg";

    prefix nspg;

    organization "Concordia University";

    contact
      "Ron Brash <ron.brash@gmail.com>";

    description
      "YANG model for the NSPG project

     Copyright (c) 2016 Ron Brash and the persons identified as
     authors of the code.  All rights reserved.

     ";  // Description

    revision "2016-06-10" {
      description
        "NSPG module in progress.";
    }

        // ------------------------------------------------------------------
        //  NSPG operation types
        // ------------------------------------------------------------------
    identity nspgOpEvent-type {
      description
        "Base for all NSPG event types.";
    }
    identity single-req {
      base nspgOpEvent-type;
      description "Single req event.";
    }
    identity perform-all {
      base nspgOpEvent-type;
      description "Perform all - read and write";
    }
    typedef uuid {
       type string {
         pattern '[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-'
               + '[0-9a-fA-F]{4}-[0-9a-fA-F]{12}';
       }
       description
        "A Universally Unique IDentifier in the string representation
         defined in RFC 4122.  The canonical representation uses
         lowercase characters.
```

```
          The following is an example of a UUID in string representation:
          f81d4fae-7dec-11d0-a765-00a0c91e6bf6
          ";
        reference
          "RFC 4122: A Universally Unique IDentifier (UUID) URN
                     Namespace";
      }
      typedef DisplayString {
        type string {
          length "0 .. 65535";
        }
        description
          "YANG version of the The string Built-In Type.";
        reference
          "RFC 6020, section 9.4";

      }

          // ------------------------------------------------------------------
          //  container nspg
          // ------------------------------------------------------------------
      container nspg {
        presence
          "Indicates the nspg service is available";
        description
          "Top-level container for all nspg database objects.";

        leaf nspgManufacturer {
          type DisplayString;
          config false;
          mandatory true;
          description
            "The name of the nspg's manufacturer. For instance,
                      Concordia nspg.";
        }

        leaf nspgModelNumber {
          type DisplayString;
          config false;
          mandatory true;
          description
            "The name of the nspg's model. For instance,
                      Radiant Automatic.";
        }

        leaf nspgVersionNumber {
          type DisplayString;
          config false;
          mandatory true;
          description
            "The version number of the NSPG";
        }

         leaf nspgUUID {
          type DisplayString;
```

```
        config false;
        mandatory true;
        description
          "The UUID of the NSPG";
      }

      leaf nspgStatus {
        type enumeration {
        enum "enabled" {
            value 1;
            description
              "The nspg is enabled.";
          }
          enum "disabled" {
            value 2;
            description
              "The nspg is disabled.";
          }
        }
        config false;
        mandatory true;
        description
          "This variable indicates the current state of
                the nspg.";
      }
    } // container nspg

        // ------------------------------------------------------------------
        //  rpc performALLOPs
        // ------------------------------------------------------------------
    rpc performALLOPs {
      description
        "Perform all SNMP operations and output results.
          The snmpOpDone notification will be sent when
          the operation is finished.
          An 'in-use' error will be returned if toast
          is already being made.
          A 'resource-denied' error will be returned
          if the nspg service is disabled.";
      output {
          leaf nspgResult {
              type string;
          }
      }
    } // rpc performALLOPs

        // ------------------------------------------------------------------
        //  rpc performSingleOP
        // ------------------------------------------------------------------
    rpc performSingleOP {
      description
        "Perform single SNMP operations and output results.
          The snmpOpDone notification will be sent when
          the operation is finished.
          An 'in-use' error will be returned if toast
```

```
            is already being made.
            A 'resource-denied' error will be returned
            if the nspg service is disabled.";
      input {
            leaf nspg-uuid {
                type uuid;
                default "00000000-0000-0000-0000-000000000000";
                description
              "This variable informs the nspg of the uuid to
            be queried.";
            }

        }
        output {
          leaf nspgResult {
              type string;
          }
        }
      }// rpc performSingleOP

    notification snmpOpDone {
      description
        "Indicates that the SNMP operation in progress or has completed.";
      leaf operationStatus {
        type enumeration {
          enum "done" {
            value 0;
            description "The SNMP operation is completed.";
          }
          enum "cancelled" {
            value 1;
            description
              "The SNMP operation was cancelled.";
          }
          enum "error" {
            value 2;
            description
              "The nspg service was disabled or
                        the nspg is broken.";
          }
        }
        description
          "Indicates the final SNMP operation status";
      }
    }  // notification snmpOpDone
  }  // module nspg
```

A.7 Testing Processes

Resources were monitored using a modified version of this script to monitor system resources.

```
#!/bin/sh
```

```
topp() (
  $* &>/dev/null &
  pid="$!"
  trap ':' INT
  echo 'CPU  MEM'
  while sleep 1; do ps --no-headers -o '%cpu,%mem' -p "$pid"; done
  kill "$pid"
)

topp ./myprog arg1 arg2
```

Bridge script

```
Create the bridge interface.
root@coruscant:~ # brctl addbr mybridge

Add interfaces to the bridge.
root@coruscant:~ # brctl addif mybridge eth1
root@coruscant:~ # brctl addif mybridge eth2

Zero IP the interfaces.
root@coruscant:~ # ifconfig eth1 0.0.0.0
root@coruscant:~ # ifconfig eth2 0.0.0.0

Put up the bridge.
root@coruscant:~ # ifconfig mybridge up
```

Packets were captured using tcpdump

```
sudo tcpdump -i mybridge -w file.pcap
sudo tcpdump -i eth0 -w file.pcap
```

Packets were generated using a custom tool. To be provided upon special request.

A.8 NSPG Upstream Test

The annotated test results for the upstream

```
... SNMP
09:48:09.122375 IP (tos 0x0, ttl 64, id 45676, offset 0, flags [DF], proto UDP (17), length 71)
    localhost.58844 > localhost.snmp:  { SNMPv1 { GetRequest(28) R=545421570
    system.sysUpTime.0 } }
    E..G..@.@.................3.F0).....public....r.........0.0...+.........
... Several connection attempts
09:48:18.409520 IP (tos 0x0, ttl 64, id 47134, offset 0, flags [DF], proto UDP (17), length 71)
    localhost.32912 > localhost.snmp:  { SNMPv2c { GetRequest(28) R=1913493247
    system.sysUpTime.0 } }
    E..G..@.@.................3.F0).....public....r.........0.0...+.........
... Several connection attempts
09:48:26.381308 IP (tos 0x0, ttl 64, id 48186, offset 0, flags [DF], proto UDP (17), length 96)
    localhost.33422 > localhost.snmp:  { SNMPv3 { F=r } { USM B=0 T=0 U= } { ScopedPDU E=
    C=none { GetRequest(14) R=841961061  } } }
```

```
    E..‘.−@.@..]..............L._0B...0...^.  ..............0..............0.....none....2/Ne......0.
... Several connection attempts

..... NETCONF

09:49.31.458191 IP (tos 0x0, ttl 64, id 26198, offset 0, flags [DF], proto
  TCP (6), length 60)192.168.0.2.53091 > 192.168.0.20.830: Flags [S],
  cksum 0x1128 (correct), seq 2309306514, win 29200, options
  [mss 1460,sackOK,TS val 7074602 ecr 0,nop,wscale 7], length 0

09:49.31.458203 IP (tos 0x0, ttl 59, id 32198, offset 0, flags [none], proto
  TCP (6), length 60)
  192.168.0.20.830 > 192.168.0.2.53091: Flags [S.], cksum 0xd36d (correct),
  seq 4930176503, ack 2309306514, win 42540, options [mss 1430,sackOK,
  TS val 1018180827 ecr 7074602,nop,wscale 7], length 0

09:49.31.458212 (tos 0x0, ttl 64, id 26199, offset 0, flags [.], proto
  TCP (6), length 52)
  192.168.0.2.53091 > 192.168.0.20.830: Flags [.], cksum 0xa759 (correct),
  ack 1, win 229, options [nop,nop,TS val 7074612 ecr 1018180827], length 0

... Successful connection

... Several packets

09:49.31.687815 IP (tos 0x8, ttl 64, id 7467, offset 0, flags [DF], proto TCP (6), length 1832)
    192.168.0.20.830 > 192.168.0.20.53091: Flags [P.], cksum 0x051d (incorrect -> 0xd99f),
    seq 5206:6986, ack 2060, win 382, options [nop,nop,TS val 8926738 ecr 8926737], length 1780
    E..(.+@.@.............>....=.........~.......
    ..6...6.1.N..F3.D>2)!....~.Oj4c7...4.Z.....x..x..*g..q.@c.7u.....’.*h..I..<....?..7>I.zp.
    +.WGz...T.V..j8..5...+/S.F.Ng‘..~..  ...‘d.Rc.j.......(.R...4..F....+...}.k.mU...I.NV.......
    T.......f....BFiI.G.hl......{.  .K7Z;&..]e
    1]..:.......S...Iyt..T.i.....
```

A.9 NSPG Downstream Test

The annotated test results for the downstream

```
$ sudo tcpdump -v -A -i eth1 udp
[sudo] password for rbrash:
tcpdump: listening on lo, link-type EN10MB (Ethernet), capture size 262144 bytes

14:12:19.759077 IP (tos 0x0, ttl 64, id 44901, offset 0, flags [DF], proto UDP (17), length 71)
    localhost.36790 > 192.168.1.20.snmp:  { SNMPv1 { GetRequest(28) R=1051242895  system.sysName.0 } }
E..G.e@.@.. >.............3.F0).....public.... >.........0.0...+.........
14:12:19.759679 IP (tos 0x0, ttl 64, id 44902, offset 0, flags [DF], proto UDP (17), length 77)
    192.168.1.20.snmp > localhost.36790:  { SNMPv1 { GetResponse(34) R=1051242895  system.sysName.0="ackbar" } }
E..M.f@.@..7.............9.L0/.....public."..>.........0.0...+.........ackbar
14:12:26.760781 IP (tos 0x0, ttl 64, id 45567, offset 0, flags [DF], proto UDP (17), length 71)
    localhost.41808 > 192.168.1.20.snmp:  { SNMPv2c { GetRequest(28) R=1831996728  system.sysName.0 } }
E..G..@.@.............P...3.F0).....public....m2.8......0.0...+.........
14:12:26.761149 IP (tos 0x0, ttl 64, id 45568, offset 0, flags [DF], proto UDP (17), length 77)
```

```
␣␣␣␣192.168.1.20.snmp␣>␣localhost.41808:␣␣{␣SNMPv2c␣{␣GetResponse(34)␣R=1831996728␣␣system.sysName.0="ackbar"␣}␣}
E..M..@.@.............P.9.L0/.....public."..m2.8......0.0...+.........ackbar
14:14:42.262365 IP (tos 0x0, ttl 64, id 64984, offset 0, flags [DF], proto UDP (17), length 92)
    localhost.36429 > 192.168.1.20.snmp:  { SNMPv3 { F=r } { USM B=0 T=0 U= } { ScopedPDU E=
     C= { GetRequest(14) R=929622139  } } }
E..\..@.@. >.........M...H.[0 >...0...=..0.............0.............0.........7h.{......0.
14:14:42.262565 IP (tos 0x0, ttl 64, id 64985, offset 0, flags [DF], proto UDP (17), length 144)
    192.168.1.20.snmp > localhost.36429:  { SNMPv3 { F= } { USM B=65 T=844 U= } { ScopedPDU E=
    0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00 C= { Report(31) R=929622139
    S:snmpUsmMIB.usmMIBObjects.usmStats.usmStatsUnknownEngineIDs.0=1 } } }
E.....@.@. >............M.|..0r...0...=..0............."0␣........m...␣␣␣␣.W......A...L......06........
m...␣␣␣␣.W..........7h.{......0.0..
 +.........A..
14:14:42.262623␣IP␣(tos␣0x0,␣ttl␣64,␣id␣64986,␣offset␣0,␣flags␣[DF],␣proto␣UDP␣(17),␣length␣148)
␣␣␣␣localhost.36429␣>␣192.168.1.20.snmp:␣␣{␣SNMPv3␣{␣F=r␣}␣{␣USM␣B=65␣T=844␣U=MD5user␣}␣{␣ScopedPDU␣E=
␣␣␣␣␣0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00␣C=␣{␣GetRequest(28)
␣␣␣␣␣R=929622138␣␣system.sysName.0␣}␣}␣}
E.....@.@. >|.........M......0v...0...=../............)0 '........m...␣␣␣␣.W......A...L..MD5user....
03........m...␣␣.W..........7h.z......0.0...+.........
14:14:42.262758␣IP␣(tos␣0x0,␣ttl␣64,␣id␣64987,␣offset␣0,␣flags␣[DF],␣proto␣UDP␣(17),␣length␣151)
␣␣␣␣192.168.1.20.snmp␣>␣localhost.36429:␣␣{␣SNMPv3␣{␣F=␣}␣{␣USM␣B=65␣T=844␣U=MD5user␣}␣{␣ScopedPDU␣E=
␣␣␣␣0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00␣C=␣{␣Report(31)␣R=929622138
␣␣␣␣S:snmpUsmMIB.usmMIBObjects.usmStats.usmStatsUnknownUserNames.0=1␣}␣}␣}
E.....@.@>x............M....0y...0...=../............)0 '........m...␣␣␣␣.W......A...L..MD5user....06.
.......m...␣␣␣␣␣.W..........7h.z......0.0..
 +.........A..
14:14:47.442497␣IP␣(tos␣0x0,␣ttl␣64,␣id␣684,␣offset␣0,␣flags␣[DF],␣proto␣UDP␣(17),␣length␣92)
␣␣␣␣192.168.1.4.47548␣>␣192.168.1.20.snmp:␣␣{␣SNMPv3␣{␣F=r␣}␣{␣USM␣B=0␣T=0␣U=␣}␣{␣ScopedPDU␣E=␣␣C=
␣␣␣␣{␣GetRequest(14)␣R=1535716849␣␣}␣}␣}
E..\..@.@.9..............H.[0 >...0...,.]V.............0.............0.........[.−.......0.
14:14:47.442910␣IP␣(tos␣0x0,␣ttl␣64,␣id␣685,␣offset␣0,␣flags␣[DF],␣proto␣UDP␣(17),␣length␣144)
␣␣␣␣192.168.1.20.snmp␣>␣192.168.1.4.47548:␣␣{␣SNMPv3␣{␣F=␣}␣{␣USM␣B=65␣T=849␣U=␣}␣{␣ScopedPDU␣E=
␣␣␣␣0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00␣C=␣{␣Report(31)␣R=1535716849
␣␣␣␣␣␣S:snmpUsmMIB.usmMIBObjects.usmStats.usmStatsUnknownEngineIDs.0=2␣}␣}␣}
E.....@.@.9..............|..0r...0...,.]V............."0 ........m...     .W......A...Q......06.......
.m...     .W..........[.−.......0.0..
 +.........A..
14:14:47.443080 IP (tos 0x0, ttl 64, id 686, offset 0, flags [DF], proto UDP (17), length 148)
    192.168.1.4.47548 > 192.168.1.20.snmp:  { SNMPv3 { F=r } { USM B=65 T=849 U=MD5User } { ScopedPDU
    E= 0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00 C= { GetRequest(28)
     R=1535716848  system.sysName.0 } } }
E.....@.@.9.................0v...0...,.]U............)0 '........m...     .W......A...Q..MD5User...
.03........m...  .W..........[.−.......0.0...+.........
14:14:47.443489 IP (tos 0x0, ttl 64, id 687, offset 0, flags [DF], proto UDP (17), length 148)
    192.168.1.20.snmp > 192.168.1.4.47548:  { SNMPv3 { F= } { USM B=65 T=849 U=MD5User } { ScopedPDU
     E= 0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00 C= { GetResponse(28)
     R=1535716848  authorizationError[errorIndex==0] system.sysName.0= } } }
E.....@.@.9.................0v...0...,.]U............)0 '........m...     .W......A...Q..MD5User....
03........m...  .W..........[.−.......0.0...+.........
14:15:06.951843 IP (tos 0x0, ttl 64, id 4853, offset 0, flags [DF], proto UDP (17), length 92)
    localhost.45942 > 192.168.1.20.snmp:  { SNMPv3 { F=r } { USM B=0 T=0 U= } { ScopedPDU E=
    C= { GetRequest(14) R=1548061890  } } }
E..\..@.@.).........v...H.[0 >...0....>).............0.............0.........\E........0.
14:15:06.952257 IP (tos 0x0, ttl 64, id 4854, offset 0, flags [DF], proto UDP (17), length 144)
    192.168.1.20.snmp > localhost.45942:  { SNMPv3 { F= } { USM B=65 T=869 U= } { ScopedPDU E=
```

```
    0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00 C= { Report(31) R=1548061890
    S:snmpUsmMIB.usmMIBObjects.usmStats.usmStatsUnknownEngineIDs.0=3 } } }
E.....@.@.)e..........v.|..0r...0.... >).............."0␣........m...␣␣␣␣.W......A...e......06......
..m...␣␣.W..........\E........0.0..
+.........A..
14:15:06.952478␣IP␣(tos␣0x0,␣ttl␣64,␣id␣4855,␣offset␣0,␣flags␣[DF],␣proto␣UDP␣(17),␣length␣161)
␣␣␣␣localhost.45942␣>␣192.168.1.20.snmp:␣␣{␣SNMPv3␣{␣F=ar␣}␣{␣USM␣B=65␣T=869␣U=MD5User␣}␣{␣ScopedPDU
␣␣␣␣E=␣0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00␣C=␣{␣GetRequest(28)
␣␣␣␣R=1548061889␣␣system.sysName.0␣}␣}␣}
E.....@.@.)S..........v.......0.....0.... >).............503........m...␣␣␣.W......A...e..MD5User...
.......t.fB..03........m...␣␣␣␣␣.W..........\E........0.0...+.........
14:15:06.952871␣IP␣(tos␣0x0,␣ttl␣64,␣id␣4856,␣offset␣0,␣flags␣[DF],␣proto␣UDP␣(17),␣length␣167)
␣␣␣␣192.168.1.20.snmp␣>␣localhost.45942:␣␣{␣SNMPv3␣{␣F=a␣}␣{␣USM␣B=65␣T=869␣U=MD5User␣}␣{␣ScopedPDU
␣␣␣␣E=␣0x800x000x1F0x880x800xED0x6D0xC00x0C0x0B0x090xFD0x570x000x000x000x00␣C=␣{␣GetResponse(34)
␣␣␣␣␣R=1548061889␣␣system.sysName.0="ackbar"␣}␣}␣}
E.....@.@.)L..........v....0.....0.... >).............503........m...␣␣␣.W......A...e..MD5User....E.S..
.G....09........m...␣␣␣␣.W........"..\E........0.0...+.........ackbar
```

References

Aaltonen, M. L. I., Karvonen, H., Norros, L., & Fuentes, B. (2013). Coping with the demands of network management by autonomic functionalities and training. In *Proceedings of the 31st european conference on cognitive ergonomics* (pp. 10:1–10:6). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/2501907.2501941 doi: 10.1145/2501907.2501941

Annosi, M. C., Penta, M. D., & Tortora, G. (2012, June). Managing and assessing the risk of component upgrades. In *Product line approaches in software engineering (please), 2012 3rd international workshop on* (p. 9-12). doi: 10.1109/PLEASE.2012.6229776

Aravind, S., & Padmavathi, G. (2015, May). Migration to ipv6 from ipv4 by dual stack and tunneling techniques. In *Smart technologies and management for computing, communication, controls, energy and materials (icstm), 2015 international conference on* (p. 107-111). doi: 10.1109/ICSTM.2015.7225398

Assels, M., Echtner, D., Spanner, M., Mokhov, S., Carrière, F., & Taveroff, M. (2011). Multifaceted faculty network design and management: Practice and experience report. *CoRR*, *abs/1103.5433*. Retrieved from http://arxiv.org/abs/1103.5433

Atkinson, R., & Kent, S. (1998, November). *Security Architecture for the Internet Protocol* (No. 2401). RFC 2401. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc2401.txt doi: 10.17487/rfc2401

Baeza-Yates, R., & Manber, U. (2012). *Computer science: Research and applications*. Springer US. Retrieved from https://books.google.ca/books?id=hBrlBwAAQBAJ

Barnes, R., Thomson, M., Pironti, A., & Langley, A. (2015, June). *Deprecating Secure Sockets*

Layer Version 3.0 (No. 7568). RFC 7568. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc7568.txt doi: 10.17487/rfc7568

Bellovin, S. M., Blaze, M., Diffie, W., Landau, S., Neumann, P. G., & Rexford, J. (2008, Jan). Risking communications security: Potential hazards of the protect america act. *IEEE Security Privacy*, *6*(1), 24-33. doi: 10.1109/MSP.2008.17

Beranek, B., & Rosen, E. (1981, July). *Vulnerabilities of network control protocols: An example* (No. 789). RFC 789. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc789.txt doi: 10.17487/rfc789

Bi, J. (2012). *The challenges of sdn/openflow in an operational and large-scale network.* Retrieved 2015-08-15, from http://opennetsummit.org/archives/apr12/bi-tue-challenges.pdf

Bierman, A. (2001, December). *Snmp set: Can it be saved?* IETF. Retrieved 2016-10-21, from http://www.simple-times.org/pub/simple-times/issues/9-1.html

Bjorklund, M. (2010, October). *YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)* (No. 6020). RFC 6020. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc6020.txt doi: 10.17487/rfc6020

Bradner, S. (1991, July). *Benchmarking Terminology for Network Interconnection Devices* (No. 1242). RFC 1242. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc1242.txt doi: 10.17487/rfc1242

Casado, M., Freedman, M., Pettit, J., & Luo, J. (2009, August). Rethinking enterprise network control. *IEEE/ACM Trans. Netw.*, *17*(4), 1270–1283. Retrieved from http://0-dx.doi.org.mercury.concordia.ca/10.1109/TNET.2009.2026415 doi: 10.1109/TNET.2009.2026415

Cerf, V. (1988, April 1). *IAB recommendations for the development of Internet network management standards* (No. 1052). RFC 1052. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc1052.txt doi: 10.17487/rfc1052

CERT. (2015). *Sei cert c coding standard.* Retrieved 2015-08-15, from https://www.securecoding.cert.org/confluence/display/c/SEI+CERT+C+Coding+Standard

CERT. (2016). *Env33-c. do not call system().* Retrieved 2016-08-15, from https://www.securecoding.cert.org/confluence/pages/viewpage.action?pageId=2130132

Chaudhary, A., & Sardana, A. (2011, April). Software based implementation methodologies for deep packet inspection. In *2011 international conference on information science and applications* (p. 1-10). doi: 10.1109/ICISA.2011.5772430

CISCO. (2011). *Snmp technology configuration example.* Retrieved 2016-10-21, from http://www.cisco.com/en/US/partner/tech/tk648/tk362/technologies_configuration_example09186a0080094aa6.shtml

Cisco Systems Inc. (2010). *Dual stack network.* Retrieved 2015-10-21, from http://www.cisco.com/c/dam/en_us/solutions/industries/docs/gov/IPV6at_a_glance_c45-625859.pdf

Cisco Systems Inc. (2014). *Cisco open sdn controller 1.0 southbound interface guide* (Vol. 1.0). Retrieved 2015-10-21, from https://developer.cisco.com/site/openSDN/documents/southbound/

Claffy, K. (2012, March). Workshop on internet economics (wie2011) report. *SIGCOMM Comput. Commun. Rev.*, *42*(2), 110–114. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/2185376.2185394 doi: 10.1145/2185376.2185394

claise, B. (2014, March). *Writable mib module.* Retrieved 2015-08-15, from http://www.ietf.org/iesg/statement/writable-mib-module.html

Clark, D. (2009). *Toward the design of a future internet* (Vol. 7.0). Retrieved 2015-08-15, from https://groups.csail.mit.edu/ana/People/DDC/Future%20Internet.pdf

Corrente, A., & Tura, L. (2004, April). Security performance analysis of snmpv3 with respect to snmpv2c. In *Network operations and management symposium, 2004. noms 2004. ieee/ifip* (Vol. 1, p. 729-742 Vol.1). doi: 10.1109/NOMS.2004.1317760

Cyert, R. M., & Kumar, P. (1994, Nov). Technology management and the future. *IEEE Transactions on Engineering Management*, *41*(4), 333-334. doi: 10.1109/17.364554

DARPA. (2008). *Darpa: 50 years of bridging the gap*. Faircount LLC. Retrieved from https://

books.google.ca/books?id=A7ImSwAACAAJ

Dhall, H., Dhall, D., Batra, S., & Rani, P. (2012, Jan). Implementation of ipsec protocol. In *Advanced computing communication technologies (acct), 2012 second international conference on* (p. 176-181). doi: 10.1109/ACCT.2012.64

Dierks, T. (2008, August). *The Transport Layer Security (TLS) Protocol Version 1.2* (No. 5246). RFC 5246. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc5246.txt doi: 10.17487/rfc5246

Docker. (2016). *Docker use cases.* Retrieved 2015-08-15, from https://www.docker.com/use-cases

DoD standard Internet Protocol (No. 760). (1980, January). RFC 760. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc760.txt doi: 10.17487/rfc760

Douligeris, C., & Serpanos, D. N. (2007). Intrusion detection versus intrusion protection. In *Network security:current status and future directions* (p. 99-115). Wiley-IEEE Press. Retrieved from http://0-ieeexplore.ieee.org.mercury.concordia.ca/xpl/articleDetails.jsp?arnumber=5237781 doi: 10.1002/9780470099742.ch7

Dourish, P., Grinter, E., de la Flor, J. D., & Joseph, M. (2004, November). Security in the wild: User strategies for managing security as an everyday, practical problem. *Personal Ubiquitous Comput.*, *8*(6), 391–401. Retrieved from http://0-dx.doi.org.mercury.concordia.ca/10.1007/s00779-004-0308-5 doi: 10.1007/s00779-004-0308-5

Dykes, B., & McPherson, D. (2001, February). *VLAN Aggregation for Efficient IP Address Allocation* (No. 3069). RFC 3069. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc3069.txt doi: 10.17487/rfc3069

Eddy, W. (2007, August). *TCP SYN Flooding Attacks and Common Mitigations* (No. 4987). RFC 4987. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4987.txt doi: 10.17487/rfc4987

Edwards, K., Poole, E., & Stoll, J. (2008). Security automation considered harmful? In *Proceedings of the 2007 workshop on new security paradigms* (pp. 33–42). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/

10.1145/1600176.1600182 doi: 10.1145/1600176.1600182

Eggert, L., & Fairhurst, G. (2008, November). *Unicast UDP Usage Guidelines for Application Designers* (No. 5405). RFC 5405. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc5405.txt doi: 10.17487/rfc5405

Enns, R. (2006, December). *NETCONF Configuration Protocol* (No. 4741). RFC 4741. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4741.txt doi: 10.17487/rfc4741

Enns, R., Bjorklund, M., Bierman, A., & Schonwalder, J. (2011, June). *Network Configuration Protocol (NETCONF)* (No. 6241). RFC 6241. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc6241.txt doi: 10.17487/rfc6241

Ferguson, P. (2000, May). *Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing* (No. 2827). RFC 2827. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc2827.txt doi: 10.17487/rfc2827

File Transfer Protocol (No. 959). (1985, October). RFC 959. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc959.txt doi: 10.17487/rfc959

Flauzac, O., Nolot, F., Rabat, C., & Steffenel, L. (2009, Oct). Grid of security: A new approach of the network security. In *Network and system security, 2009. nss '09. third international conference on* (p. 67-72). doi: 10.1109/NSS.2009.53

Foley, S., Quillinan, T., O'Connor, M., Mulcahy, P., & Morrison, J. (2004, April). A framework for heterogeneous middleware security. In *Parallel and distributed processing symposium, 2004. proceedings. 18th international* (p. 108). doi: 10.1109/IPDPS.2004.1303059

Formyduval, W. (2009). Integrating static analysis and testing for firewall policies. In *Proceedings of the 24th acm sigplan conference companion on object oriented programming systems languages and applications* (pp. 749–750). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/1639950.1639996 doi: 10.1145/1639950.1639996

Foschiano, M., & HomChaudhuri, S. (2010, February). *Cisco Systems - Private VLANs: Scalable Security in a Multi-Client Environment* (No. 5517). RFC 5517. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc5517.txt doi: 10.17487/rfc5517

Freier, A., Karlton, P., & Kocher, P. (2011, August). *The Secure Sockets Layer (SSL) Protocol Version 3.0* (No. 6101). RFC 6101. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc6101.txt doi: 10.17487/rfc6101

Gao, L., Xing, B., Zhang, J., & Li, H. (2010, Oct). Developing efficient xml-snmp model: An xml-template based approach. In *2010 international conference on computer application and system modeling (iccasm 2010)* (Vol. 4, p. V4-731-V4-734). doi: 10.1109/ICCASM.2010.5619276

Gasser, M. (1988). *Building a secure computer system.* New York, NY, USA: Van Nostrand Reinhold Co.

GNU Project. (2016, Dec). *Wireshark.* Free Software Foundation. Retrieved 2016-10-21, from https://www.gnu.org/software/bash/

Goddard, T. (2006, December). *Using NETCONF over the Simple Object Access Protocol (SOAP)* (No. 4743). RFC 4743. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4743.txt doi: 10.17487/rfc4743

Google. (2016). *Use cases for grpc in network management.* Retrieved 2015-08-15, from https://tools.ietf.org/html/draft-talwar-rtgwg-grpc-use-cases-00

Gorur, P. Y. (2006, March). Converged network management: challenges and solutions. In *2006 optical fiber communication conference and the national fiber optic engineers conference.* doi: 10.1109/OFC.2006.215676

Harmon, R. R., & Laird, G. L. (2012, July). Roadmapping the service transition: Insights for technology organizations. In *2012 proceedings of picmet '12: Technology management for emerging technologies* (p. 3121-3130).

Harrington, D., Wijnen, B., & Presuhn, R. (2002, December). *An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks* (No. 3411). RFC 3411. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc3411.txt doi: 10.17487/rfc3411

Hedrick, C. (1988, November). *Telnet remote flow control option* (No. 1080). RFC 1080. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc1080.txt doi: 10

.17487/rfc1080

Heffernan, A. (1998, August). *Protection of BGP Sessions via the TCP MD5 Signature Option* (No. 2385). RFC 2385. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc2385.txt doi: 10.17487/rfc2385

Heinanen, J., Gleeson, B., & Armitage, G. (2000, February). *A framework for ip based virtual private networks* (No. 2764). RFC 2764. RFC Editor. Retrieved from https://www.ietf.org/rfc/rfc2764.txt doi: 10.17487/rfc2764

Hong, S. I., Song, S. Y., & Lin, C. H. (2015, July). An efficient multi-protocol gateway system design on the zigbee. In *2015 17th international conference on advanced communication technology (icact)* (p. 525-528). doi: 10.1109/ICACT.2015.7224850

Ibrahim, N. (2009, Oct). Orthogonal classification of middleware technologies. In *Mobile ubiquitous computing, systems, services and technologies, 2009. ubicomm '09. third international conference on* (p. 46-51). doi: 10.1109/UBICOMM.2009.24

IEEE. (2015). *802.1q-2014 - bridges and bridged networks* (Vol. 1.0). Retrieved 2015-10-21, from http://www.ieee802.org/1/pages/802.1Q-2014.html

IETF. (1990a). *A simple network management protocol (snmp)* (Vol. RFC 1157). RFC Editor. Retrieved 2015-08-15, from https://tools.ietf.org/html/rfc1157

IETF. (1990b). *Structure and identification of management information for tcp/ip-based internets* (Vol. RFC 1155). RFC Editor. Retrieved 2015-08-15, from https://tools.ietf.org/html/RFC1155

Intel Corporation and Xerox Corporation. (1980). *The ethernet - a local area network.*

Interface to the routing system (i2rs) charter. (2015). Retrieved 2015-08-15, from https://datatracker.ietf.org/wg/ir2s/charter/

Internet Control Message Protocol (No. 792). (1981, September). RFC 792. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc792.txt doi: 10.17487/rfc792

Internet Protocol (No. 791). (1981, September). RFC 791. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc791.txt doi: 10.17487/rfc791

Jaha, A. A., Shatwan, F. B., & Ashibani, M. (2008, Sept). Proper virtual private network (vpn) solution. In *2008 the second international conference on next generation mobile applications,*

services, and technologies (p. 309-314). doi: 10.1109/NGMAST.2008.18

Jones, G. (2004, September). *Operational Security Requirements for Large Internet Service Provider (ISP) IP Network Infrastructure* (No. 3871). RFC 3871. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc3871.txt doi: 10.17487/rfc3871

Kaufman, C. (2005, December). *Internet Key Exchange (IKEv2) Protocol* (No. 4306). RFC 4306. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4306.txt doi: 10.17487/rfc4306

Kerrisk, M. (2016, July). *Linux top.* man7.org. Retrieved 2016-10-21, from http://man7.org/linux/man-pages/man1/top.1.html

Klie, T., & Straub, F. (2004, July). Integrating snmp agents with xml-based management systems. *IEEE Communications Magazine*, *42*(7), 76-83. doi: 10.1109/MCOM.2004.1316537

Kouadri, G., & Brézillon, P. (2006). Human-centric network security management: A comprehensive helper. In *Proceedings of the 4th international workshop on wireless mobile applications and services on wlan hotspots* (pp. 47–52). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/1161023.1161031 doi: 10.1145/1161023.1161031

Kreutz, D., Ramos, M., & Verissimo, P. (2015, Jan). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, *103*(1), 14-76. doi: 10.1109/JPROC.2014.2371999

Kurose, J., & Ross, K. (2012). *Computer networking: A top-down approach* (6th ed.). Pearson.

Lear, E., & Crozier, K. (2006, December). *Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP)* (No. 4744). RFC 4744. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4744.txt doi: 10.17487/rfc4744

Leight, P., & Hammer, R. (2006). *What is defense in depth?* SANS. Retrieved 2015-08-15, from https://www.sans.edu/student-files/projects/200608_001.doc

Li, L. E., Mao, Z. M., & Rexford, J. (2012, Oct). Toward software-defined cellular networks. In *2012 european workshop on software defined networking* (p. 7-12). doi: 10.1109/EWSDN.2012.28

Libes, D. (2009, Aug). *Wireshark.* Retrieved 2016-10-21, from http://expect.sourceforge.net/

Linux Man-pages Project. (2015a). *Capabilities - overview of linux capabilities.* Retrieved 2015-08-15, from http://man7.org/linux/man-pages/man7/capabilities.7.html

Linux Man-pages Project. (2015b). *Shm overview overview of posix shared memory.* Retrieved 2015-08-15, from http://man7.org/linux/man-pages/man7/shm_overview.7.html

Linux Man-pages Project. (2015c). *Socket create an endpoint for communication* (Vol. 1.0). Retrieved 2015-08-15, from http://man7.org/linux/man-pages/man2/socket.2.html

Linux Man-pages Project. (2015d). *System execute a shell command.* Retrieved 2015-08-15, from http://man7.org/linux/man-pages/man3/system.3.html

Linux Man-pages Project. (2015e). *Unix sockets for local interprocess communication.* Retrieved 2015-08-15, from http://man7.org/linux/man-pages/man7/unix.7.html

Lonvick, C., & Ylonen, T. (2006a, January). *The Secure Shell (SSH) Authentication Protocol* (No. 4252). RFC 4252. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4252.txt doi: 10.17487/rfc4252

Lonvick, C., & Ylonen, T. (2006b, January). *The Secure Shell (SSH) Protocol Architecture* (No. 4251). RFC 4251. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4251.txt doi: 10.17487/rfc4251

Lonvick, C., & Ylonen, T. (2006c, January). *The Secure Shell (SSH) Transport Layer Protocol* (No. 4253). RFC 4253. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4253.txt doi: 10.17487/rfc4253

M. Rouse. (2012a). *Definition of an attack vector.* Retrieved 2015-08-15, from http://searchsecurity.techtarget.com/definition/attack-vector

M. Rouse. (2012b). *Definition of a network protocol.* Retrieved 2015-08-15, from http://searchnetworking.techtarget.com/definition/protocol

M. Rouse. (2012c). *Definition of a packet.* Retrieved 2015-08-15, from http://searchnetworking.techtarget.com/definition/packet

Manadhata, P., Wing, J., Flynn, M., & McQueen, M. (2006). Measuring the attack surfaces of

two ftp daemons. In *Proceedings of the 2nd acm workshop on quality of protection* (pp. 3–10). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/1179494.1179497 doi: 10.1145/1179494.1179497

Markovic-Petrovic, J., & Stojanovic, D. (2013, Oct). Analysis of scada system vulnerabilities to ddos attacks. In *2013 11th international conference on telecommunications in modern satellite, cable and broadcasting services (telsiks)* (Vol. 02, p. 591-594). doi: 10.1109/TELSKS.2013.6704448

Mauro, D., & Schmid, K. (2001). *Essential snmp*. O'Reilly. Retrieved from https://books.google.ca/books?id=sT4PTu1TwHgC

McCloghrie, K., Schönwälder, J., Perkins, D., & McCloghrie, K. (1999, April 1). *Structure of Management Information Version 2 (SMIv2)* (No. 2578). RFC 2578. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc2578.txt doi: 10.17487/rfc2578

Merriam Webster. (n.d.). *What does managerial mean?* Merriam Webster. Retrieved 2016-10-21, from http://www.merriam-webster.com/dictionary/managerial

Microsoft. (2016). *Sdl threat modeling tool.* Retrieved 2016-08-15, from https://www.microsoft.com/en-us/sdl/adopt/threatmodeling.aspx

Microsoft Corportation. (2016). *Definition of a security vulnerability.* Retrieved 2015-08-15, from https://msdn.microsoft.com/en-us/library/cc751383.aspx

Millard, D., Howard, Y., Davis, H. C., Jam, E., Gilbert, L., & Wills, G. B. (2006, Dec). Design patterns for wrapping similar legacy systems with common service interfaces. In *2006 european conference on web services (ecows'06)* (p. 191-200). doi: 10.1109/ECOWS.2006.14

Mockapetris, P. (1987, November). *Domain names - concepts and facilities* (No. 1034). RFC 1034. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc1034.txt doi: 10.17487/rfc1034

Modbus Organization. (2015). *Modbus.org faq* (Vol. 1.0). Retrieved 2016-10-21, from http://www.modbus.org/faq.php

Mokhov, S., Paquet, J., & Debbabi, M. (2014). Toward automated mac spoofer investigations. In *Proceedings of the 2014 international c* conference on computer science & software engineering* (pp. 27:1–27:6). New York, NY, USA: ACM. Retrieved from http://

0-doi.acm.org.mercury.concordia.ca/10.1145/2641483.2641540 doi: 10.1145/2641483.2641540

Mundy, R., Partain, D., & Stewart, B. (2002, December). *Introduction and Applicability Statements for Internet-Standard Management Framework* (No. 3410). RFC 3410. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc3410.txt doi: 10.17487/rfc3410

Narasimhan, P., & Triantafillou, P. (2012). *Middleware 2012: Acm/ifip/usenix 13th international middleware conference, montreal, canada, december 3-7, 2012. proceedings*. Springer Berlin Heidelberg. Retrieved from https://books.google.ca/books?id=KZC6BQAAQBAJ

Naur, P., & Randell, B. (Eds.). (1969). *Software engineering: Report of a conference sponsored by the nato science committee, garmisch, germany, 7-11 oct. 1968, brussels, scientific affairs division, nato*.

Naylor, D., Schomp, K., & Varvello, M. (2015). Multi-context tls (mctls): Enabling secure in-network functionality in tls. In *Proceedings of the 2015 acm conference on special interest group on data communication* (pp. 199–212). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/2785956.2787482 doi: 10.1145/2785956.2787482

Neisse, R., Granville, L., Ballve, D., Almedia, M., & Tarouco, L. (2003, March). A dynamic snmp to xml proxy solution. In *Integrated network management, 2003. ifip/ieee eighth international symposium on* (p. 481-484). doi: 10.1109/INM.2003.1194204

Net-snmp. (2011, May). *Net-snmp manuals.* Retrieved 2016-10-21, from http://www.net-snmp.org/docs/man/

nmap.org. (2016, Dec). *Linux nmap.* Author. Retrieved 2016-10-21, from https://nmap.org/

NXP. (2011). *Mpc8314e powerquicc ii pro processor hardware specifications.* Retrieved 2016-08-15, from http://www.nxp.com/assets/documents/data/en/data-sheets/MPC8314EEC.pdf

Office of the Privacy Commissioner of Canada. (2013, March). *What is deep packet inspection?* (Vol. 1.0). Retrieved 2015-10-21, from https://www.priv.gc.ca/information/

research-recherche/dpi_intro_e.asp

OpenYuma. (2016, Aug). *Yuma yangcli manual.* Retrieved 2016-10-21, from http://yuma123.org/wiki/index.php/Yuma_yangcli_Manual

PackETH. (2015, April). *Packeth tool.* Retrieved 2016-10-21, from http://packeth.sourceforge.net/packeth/Home.html

Patton, M., S.Bradner, Elz, R., & Bush, R. (1997, July). *Selection and Operation of Secondary DNS Servers* (No. 2182). RFC 2182. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc2182.txt doi: 10.17487/rfc2182

Perlman, R. (2004, March). *Network security protocols: A tutorial.* IETF. Retrieved 2016-10-21, from https://www.ietf.org/proceedings/61/slides/sectut-0/editorstrain.ppt

Presuhn, R. (2002, December). *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)* (No. 3418). RFC 3418. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc3418.txt doi: 10.17487/rfc3418

Priller, P., Aldrian, A., & Ebner, T. (2014, Sept). Case study: From legacy to connectivity migrating industrial devices into the world of smart services. In *Proceedings of the 2014 ieee emerging technology and factory automation (etfa)* (p. 1-8). doi: 10.1109/ETFA.2014.7005136

RaspberryPi Foundation. (2016a). *Raspberrypi hardware documentation.* Retrieved 2016-08-15, from https://www.raspberrypi.org/documentation/hardware/raspberrypi/

RaspberryPi Foundation. (2016b). *Raspbian os.* Retrieved 2016-08-15, from https://www.raspberrypi.org/downloads/raspbian/

Reid, R., & Gilbert, A. (2007). Managing security from the perspective of the business executive. In *Proceedings of the 4th annual conference on information security curriculum development* (pp. 15:1–15:5). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/1409908.1409925 doi: 10.1145/1409908.1409925

Rescorla, E., & Korver, B. (2003, July). *Guidelines for Writing RFC Text on Security Considerations* (No. 3552). RFC 3552. RFC Editor. Retrieved from https://rfc-editor.org/rfc/

rfc3552.txt doi: 10.17487/rfc3552

Riverbend. (2016, Dec). *Wireshark.* Author. Retrieved 2016-10-21, from https://www.wireshark.org/

Rose, M. T. (1989). *The open book: A practical perspective on open systems interconnection*. Englewood Cliffs, New Jersey, USA: Prentice Hall.

Rose, M. T. (1990, Jan). Transition and coexistence strategies for tcp/ip to osi. *IEEE Journal on Selected Areas in Communications*, *8*(1), 57-66. doi: 10.1109/49.46846

Salah, K., & Kahtani, A. (2009, December). Improving snort performance under linux. *IET Communications*, *3*(12), 1883-1895. doi: 10.1049/iet-com.2009.0114

Santos, P., Esteves, R., & Granville, L. (2015, May). Evaluating snmp, netconf, and restful web services for router virtualization management. In *Integrated network management (im), 2015 ifip/ieee international symposium on* (p. 122-130). doi: 10.1109/INM.2015.7140284

Saydam, T., & Magedanz, T. (2012). From networks and network management into service and service management. *Journal of Network and Systems Management*, *4*(4), 345–348. Retrieved from http://dx.doi.org/10.1007/BF02283158 doi: 10.1007/BF02283158

Schonwalder, J. (2003, May). *Overview of the 2002 IAB Network Management Workshop* (No. 3535). RFC 3535. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc3535.txt doi: 10.17487/rfc3535

Schonwalder, J. (2012, July). *Translation of Structure of Management Information Version 2 (SMIv2) MIB Modules to YANG Modules* (No. 6643). RFC 6643. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc6643.txt doi: 10.17487/rfc6643

Secure inter-domain routing (sidr). (2015). Retrieved 2015-08-15, from https://datatracker.ietf.org/wg/sidr/charter/

Shapiro, M. (1986). Structure and encapsulation in distributed systems: the proxy principle. In *Proc. 6th ieee int. conf. on distributed computing systems* (pp. 198–204).

Sharma, V., Ghani, N., & Fang, L. (2007, April). Virtual private networks [guest editorial]. *IEEE Communications Magazine*, *45*(4), 24-25. doi: 10.1109/MCOM.2007.343607

Simple Gateway Monitoring Protocol (No. 1028). (1987, November). RFC 1028. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc1028.txt doi: 10.17487/

rfc1028

Smallwood, D., & Vance, A. (2011, Dec). Intrusion analysis with deep packet inspection: Increasing efficiency of packet based investigations. In *Cloud and service computing (csc), 2011 international conference on* (p. 342-347). doi: 10.1109/CSC.2011.6138545

Smith, A., McCloghrie, K., Rijhsinghani, A., & Langille, P. (1999, August). *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions* (No. 2674). RFC 2674. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc2674.txt doi: 10.17487/rfc2674

Spivey, W. A., & Flannery, W. T. (1991, Oct). Technology transition: implications from studying a dod laboratory. In *Technology management : the new international language* (p. 521-524). doi: 10.1109/PICMET.1991.183705

Spivey, W. A., Munson, J. M., Nelson, M. A., & Dietrich, G. B. (1997, Nov). Coordinating the technology transfer and transition of information technology: a phenomenological perspective. *IEEE Transactions on Engineering Management*, *44*(4), 359-366. doi: 10.1109/17.649866

Stouffer, K., Falco, J., & Scarfone, K. (2011, June). *Guide to industrial control systems security.* NIST. Retrieved 2016-10-21, from http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf

Stubbs, J. (2013). *Can gigabit ethernet improve industrial control network performance?* Retrieved 2016-08-15, from http://www.controleng.com/single-article/can-gigabit-ethernet-improve-industrial-control-network-performance/786b27233b7328e426ac540d969e297e.html

Sun, K., & Sushil, J. (2014). Protecting enterprise networks through attack surface expansion. In *Proceedings of the 2014 workshop on cyber security analytics, intelligence and automation* (pp. 29–32). New York, NY, USA: ACM. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/2665936.2665939 doi: 10.1145/2665936.2665939

Swiderski, F., & Snyder, W. (2004). *Threat modeling*. Redmond, WA, USA: Microsoft Press.

Tcpdump. (2016, Oct). tcpdump.org. Retrieved 2016-10-21, from www.tcpdump.org/tcpdump_man.htm

Tjoland, W., Brennan, A., & McPhee, C. (1999). Reducing legacy ate system sustainment costs through modern system engineering architecture concepts. In *Autotestcon '99. ieee systems readiness technology conference, 1999. ieee* (p. 133-135). doi: 10.1109/AUTEST.1999.800369

Touch, J. (2007, July). *Defending TCP Against Spoofing Attacks* (No. 4953). RFC 4953. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4953.txt doi: 10.17487/rfc4953

Touch, J., Bonica, R., & Mankin, A. (2010, June). *The TCP Authentication Option* (No. 5925). RFC 5925. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc5925.txt doi: 10.17487/rfc5925

Transmission Control Protocol (No. 793). (1981, September). RFC 793. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc793.txt doi: 10.17487/rfc793

Unsoy, M. S., & Shanahan, T. A. (1981). X.75 internetworking of datapac and telenet. In *Proceedings of the seventh symposium on data communications* (pp. 232–239). New York, NY, USA: ACM. Retrieved from http://doi.acm.org/10.1145/800081.802679 doi: 10.1145/800081.802679

User Datagram Protocol (No. 768). (1980, August). RFC 768. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc768.txt doi: 10.17487/rfc768

Vaarandi, R., & Pihelgas, M. (2014, Oct). Using security logs for collecting and reporting technical security metrics. In *2014 ieee military communications conference* (p. 294-299). doi: 10.1109/MILCOM.2014.53

Van-Echtelt, F. E. A., Wynstra, F., & Van-Weele, A. (2007, Nov). Strategic and operational management of supplier involvement in new product development: A contingency perspective. *IEEE Transactions on Engineering Management*, *54*(4), 644-661. doi: 10.1109/TEM.2007.906858

Venturi, V., Stagni, F., Gianoli, A., Ceccanti, A., & Ciaschini, V. (2007, Dec). Virtual organization management across middleware boundaries. In *e-science and grid computing, ieee international conference on* (p. 545-552). doi: 10.1109/E-SCIENCE.2007.84

VIEIRA, E. (2016, February 9). *Netconf snmp gateway.* Google Patents. Retrieved from

https://www.google.com/patents/US9258132 (US Patent 9,258,132)

von Gordon, A. F. J., & Hancke, G. P. (2004, Sept). Protocol conversion for real-time energy management systems. In *Factory communication systems, 2004. proceedings. 2004 ieee international workshop on* (p. 319-322). doi: 10.1109/WFCS.2004.1377736

Wallin, S., & Leijon, V. (2006, Nov). Rethinking network management solutions. *IT Professional*, *8*(6), 19-23. doi: 10.1109/MITP.2006.144

Wallin, S., & Leijon, V. (2009). Wired-wireless multimedia networks and services management. In (pp. 15–26). Berlin, Heidelberg: Springer Berlin Heidelberg. Retrieved from http://dx.doi.org/10.1007/978-3-642-04994-1_2 doi: 10.1007/978-3-642-04994-1_2

Wang, X., Wang, L., Yu, B., & Dong, G. (2010, March). Studies on network management system framework of campus network. In *Informatics in control, automation and robotics (car), 2010 2nd international asia conference on* (Vol. 2, p. 285-289). doi: 10.1109/CAR.2010.5456547

Wasserman, M., & Goddard, T. (2006, December). *Using the NETCONF Configuration Protocol over Secure SHell (SSH)* (No. 4742). RFC 4742. RFC Editor. Retrieved from https://rfc-editor.org/rfc/rfc4742.txt doi: 10.17487/rfc4742

What is network security? (n.d.). SANS. Retrieved 2015-08-15, from https://www.sans.org/network-security/

Whitman, M. (2003, August). Enemy at the gate: Threats to information security. *Commun. ACM*, *46*(8), 91–95. Retrieved from http://0-doi.acm.org.mercury.concordia.ca/10.1145/859670.859675 doi: 10.1145/859670.859675

Wool, A. (2004, June). A quantitative study of firewall configuration errors. *Computer*, *37*(6), 62-67. doi: 10.1109/MC.2004.2

Wu, B., & Chang, Y. (2010, July). Integrating snmp agents and cli with netconf-based network management systems. In *2010 3rd international conference on computer science and information technology* (Vol. 1, p. 81-84). doi: 10.1109/ICCSIT.2010.5563913

Xu, Q., Chen, J., & Guo, B. (1998, Nov). Perspective of technological innovation and technology management in china. *IEEE Transactions on Engineering Management*, *45*(4), 381-387. doi: 10.1109/17.728579

Yin, G., Shi, D., Sui, P., & Wang, H. (2009, Aug). Towards general access control management for

middleware security. In *Inc, ims and idc, 2009. ncm '09. fifth international joint conference on* (p. 444-448). doi: 10.1109/NCM.2009.273

Yoon, J., Hong-Taek, J., & Hong, J. (2003, jul). Development of snmp-xml translator and gateway for xml-based integrated network management. *Int. J. Netw. Manag.*, *13*(4), 259–276. Retrieved from http://dx.doi.org/10.1002/nem.478 doi: 10.1002/nem.478

Young, E. A. (1993, May). Guiding technology development and transition into products responsive to end-user needs. In *Aerospace and electronics conference, 1993. naecon 1993., proceedings of the ieee 1993 national* (p. 762-768 vol.2). doi: 10.1109/NAECON.1993.290845

Zeng, B., Yao, L., & Chen, Z. (2010, Oct). A network intrusion detection system with the snooping agents. In *2010 international conference on computer application and system modeling (iccasm 2010)* (Vol. 3, p. V3-232-V3-236). doi: 10.1109/ICCASM.2010.5620022

Zhang, X., Li, C., & Zheng, W. (2004, Sept). Intrusion prevention system design. In *Computer and information technology, 2004. cit '04. the fourth international conference on* (p. 386-390). doi: 10.1109/CIT.2004.1357226