Intrusion Detection in a Private Network by Satisfying Constraints

Md Siam Hasan, Ali ElShakankiry, Thomas Dean, Mohammad Zulkernine
School of Computing
Queen's University, Kingston
Ontario, Canada K7L 3N6
{hasan,alishak,dean,mzulker}@cs.queensu.ca

***Abstract*— With the advancement of newer technologies, the frequency of malicious attacks is growing rapidly. Even private networks without external connections cannot hide from these attacks. Constant monitoring of the network is a vital element of an organization's security system. Among many monitoring techniques, network behavior analysis has become a common practice. Restricted private networks are characterized by a limited number of protocols. The normal traffic pattern of a network can be modeled as network constraints. Violation of any of these constraints indicates that an intrusion has occurred. This paper presents a novel framework to detect intrusions in a private network. We illustrate the framework with ten significant constraints on the real-time publish, subscribe and internet group management protocols. We present how the framework evaluates these constraints against traffic from an experimental network to prevent attacks.**

I. INTRODUCTION

Private networks or intranets are networks created by a single organization that controls its security policies and network management. This is a popular choice as it increases the security of the system by preventing external access. Critical infrastructures such as telecommunication, water, and power plant maintain this type of network to protect their industrial control operation from outside attacks. However, even a private network is not fully secured. If malware is injected into the network using vectors such as USB drives or installation discs then the security of the entire network can be compromised. The Stuxnet worm was used to attack the nuclear plant at Natanz in order to interfere with Iranian nuclear program [1]. The worm was distributed by USB flash drives and internal network connections. The main purpose of this attack was to modify the behavior of the programmable logic controllers of that power plant while replacing the monitoring libraries to mask the attack.

The above mentioned attacks show the necessity of developing more protection for private networks. Perimeter checking such as firewall rules and authentication policies are useful defensive techniques, but lack the abilities to monitor the ongoing network traffic. Intrusion detection systems (IDS) are needed. Network-based IDS can be categorized as signature and anomaly based [2]. Unlike signature-based IDS, zero-day attacks may be detected by anomaly-based IDS because attacks break the normal packet flow patterns [3]. Numerous anomaly-based detection techniques have been developed in the last decade [4].

A conventional network runs a wide range of protocols and services. It is difficult to characterize the behavior of such a network because of busy network traffic. However, some private networks, such as command and control networks or industrial control networks use fewer protocols. Recognizing abnormal patterns in this kind of traffic should be easier. The normal behavior of this network can be characterized as a set of constraints. While monitoring the network, if any constraint is violated, it should be a clear indication of an intrusion. We propose a framework to evaluate these constraints against network traffic in real time.

This paper is an initial exploration of the proposed framework using two protocols: Internet Group Management Protocol (IGMP) [5] and Real-Time Publish-Subscribe Protocol (RTPS) [6]. RTPS is implemented on Data Distribution Service (DDS), a protocol used in different application domains. Real-time DDS provides effective support for mission and business critical applications such as financial trading, air traffic control management, and complex supervisory and telemetry systems. The positive side of DDS is the availability, reliability, safety and integrity of services in a real-time environment. But the mechanism has numerous security holes. The proposed framework should be able to alert against several attacks. The framework is evaluated against captured PCAP[1] files on a closed network. For this exploration, we derive ten network constraints over the two protocols. Four of them are constraints on IGMP protocol and the remaining are RTPS constraints. These constraints are satisfied against normal traffic.

Paper Organization: The rest of the paper is organized as follows. Section II gives an overview of our framework. Section III describes the ten constraints and threats related to their violation. Section IV explains the implementation in detail. The evaluation environment and findings are presented in Section V. Section VI reviews the related research to this work. Finally, Section VII provides conclusion and states our future plans for this research.

II. FRAMEWORK

The proposed framework has three modules: Scanner, Parser, and Constraint Checker. Figure 1 shows a simplified version of interaction among these modules. The first module, Scanner, reads the packets from a PCAP file and extracts the IP payload. These packets are parsed in the

[1]Network traffic captured file which can be understood by an application capable of reading the format such as Tcpdump, Wireshark [7], [8].

Parser module. It converts them into C data structures for the use of the Constraint Checker. For every single packet, the Scanner passes packet envelope to the Parser. The envelope contains:

- Source IP
- Destination IP
- Source port (for RTPS Packets)
- Destination port (for RTPS Packets)
- Arrival time

Fig. 1. Modules of Intrusion Detection Framework

The envelope is passed in turn along with the parsed packet to the Constraint Checker. If the Parser fails to parse the packet, including any context sensitive constraints, an alert is generated. Examples of failures are errors in lengths (i.e., attempted buffer overflows) and malformed data (i.e., nonsensical date or time values). If the packets can be successfully parsed, they are sent to the final module for constraint satisfaction. Examples of constraints are ensuring read/write packets conform to the initial open requests, or, in the examples we use in the paper, that all members of a multicast group are participants in a DDS domain.

The initial parser, inherited from previous penetration testing research [9] uses a general engine that is parameterized by a grammar graph. This graph is generated from a Syntax Constraint Language (SCL) [10] protocol description. SCL is a derivative of ASN.1 that contains XML markup to provide both context sensitive parsing constraints and general constraints. Grammar for IGMP and RTPS are validated against multiple sources of network data to ensure that the grammar is correct.

During evaluation period of this parser against packet data from an industrial partner, it appeared too slow for our intrusion detection prototype. The new Parser is a handwritten, top-down, context-sensitive parser. The new parser has been designed with the intention of automatically generating their source code in the future. That is, we have manually followed a rigorous approach to translating the SCL to produce a hand-coded parser.

The Constraint Checker implements the constraints described in the next section. It is also currently handwritten and designed as a template to be automatically generated in the future from protocols specified in SCL.

III. Network Constraints

Network constraints define the normal behavior of the network. The constraints that define the structure of each packet are handled by the Parser but constraints between multiple packets are handled by the Constraint Checker. Normally an IDS uses rules or events to represent flow patterns of a network traffic [11]. We introduce the concept of constraints in a network because it represents more concise actions than rules [12]. These constraints are developed based on the protocol documentation [5] [6] and some attack patterns on RTPS protocol. The main objective of this research is to make the network secure from attacks. The constraints we are proposing are much easier to understand and flexible to update.

A. IGMP Constraints

1) Frequency of join report (C1): Hosts do not wait for a query from the router so that it can send a membership report to join a specific multicast group. A host will transmit the join report whenever it decides to join the group. To cover the possibility of the initial report being lost or damaged, the host repeats the join report once or twice after a short delay. This is the foundation of our first constraint.

A host is only allowed to send at most two successive join reports to a specific group – C1.

Violation of these rules indicates a malformed packet may has been sent to the target. An attacker can issue a broadcast attack causing vulnerable hosts become unresponsive and eventually start denying the service to legitimate users leading to a denial of service (DoS) attack.

2) Frequency of membership report (C2): During the multicast event, the router sends a periodic general or group-specific query. It tries to determine the existence of hosts on a specific network segment. In response to that query, active hosts send back their membership reports. However, without receiving a query, a host should not send a membership report. This leads to our second constraint.

Between two queries, a host can send its membership report only once to its specific group address – C2.

The main point of using IGMP for multicast operation is to reduce the network traffic and achieve better performance. A failure of **C2** means an increase in the frequency of the membership reports. An attacker can start multicast flooding (symptom of a DoS attack) and eliminate the benefit of using a snooping switch.

3) Destination of membership report (C3): If a router transmits a general query, then all the members send membership reports to their specific group or to a new group. This simple pattern of behavior brings the third constraint.

After receiving a general query, a host will eventually send its membership report – C3.

For an attacker, there is a little motivation to forge a membership report message. However, being successful in doing that, an attacker can become a recipient of the broadcast.

4) Validity of membership report (C4): When a host wants to leave a group, it will send a leave message to its specific group address. If the host sends a general leave message to the all-router multicast group (224.0.0.2), then it wants to leave from all the multicast groups. The life cycle of a member helps us writing the fourth constraint.

After leaving a group, the host will not send any membership report to any multicast group unless it joins back again – C4.

Not satisfying this constraint should have similar consequences of **C3**. If a membership report is sent from

a non-member host, it should be considered as a forged membership report.

Denying services is not a direct effect of violating constraints **C3** and **C4**. However, they are needed as part of checking constraints of other protocols which use multicasting such as RTPS.

B. RTPS Constraints

1) Validity of participants (C5): After the announcement of an RTPS topic, there should be at least one publisher and one subscriber for that topic. Publishers and subscribers are dynamically discovered by Global Data Space (GDS) [6]. They can independently join or leave GDS. Judging from the fact that RTPS depends on multicast groups, we present our next constraint.

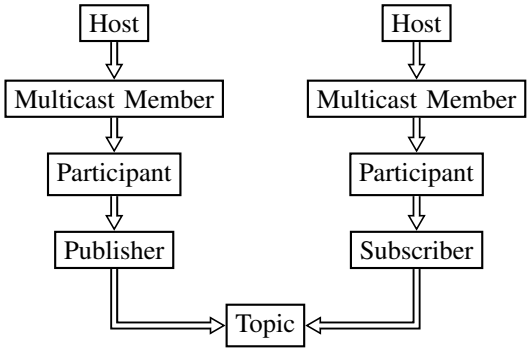

Fig. 2. Participants of an RTPS topic

All the subscribers and publishers must be valid members of at least one IGMP multicast group. These participants should send their membership reports to specific group addresses before showing their interests in a topic – C5.

C5 makes sure that all the subscribers and publishers are authorized participants. An attacker can eavesdrop [13] with malformed packets to get the privilege of being a reader or writer. The situation leads to unauthorized access to data published through infrastructure services. Sometimes, this kind of attack can be used to prevent the authentication of legitimate participants. Figure 2 explains the whole process of how a host becomes a publisher or subscriber of a particular topic.

2) Arrival of Participants (C6): After joining a group for either publishing or subscribing, the host announces its participation. This can be considered a liveliness property because eventually a participation packet should be sent from that host. The threshold time of arrival of a packet depends on the network settings. We developed one constraint based on this attribute. This time-dependent constraint [14] can be compared to forward checking as rather validating conditions from the past, it anticipates to satisfy conditions in near future.

After joining a multicast group, participation should be announced from that host in between a constant period of time – C6.

The purpose of this constraint is to prevent authentication hijacking attack of legitimate participants. An attacker can legit identity using the GUID of another participant. They can easily obtain the GUID by snooping traffic. The first effect of this attack is the blockage of valid participants.

It is important to note that these two RTPS constraints depend on the information gathered from the IGMP traffic on the network. **C5** and **C6** are defined as multi-packet constraints.

3) Participant's dual Role (C7): Publishers and subscribers are connected via the same topic. A participant is capable of playing the dual role of publisher and subscriber at the same time. It is also possible that one participant become publisher and subscriber of the same topic. This feature of RTPS can be manipulated by an attacker.

A Participant cannot be publisher and subscriber of the same topic – C7.

For compromised participant, it can record data for retransmission on a covert channel or it can alter data and replay it on the same domain. This can also increase the network traffic causing DoS attack.

4) Subscriber becoming Publisher (C8): Normally a group of subscribers listen to multiple instances of one publisher publishing a particular topic. For achieving high performance, multiple data writers can write to the same instance of a single topic and multiple data readers can receive updates from multiple instances of a single topic. By gaining the knowledge about the quality of service (QoS), a subscriber of the different domain can turn itself into a publisher of that topic. This is the main motivation behind the eighth constraint.

A topic key is only published from a specific set of hosts – C8.

Some Air Traffic Control (ATC) systems use the RTPS protocol. It has two radar systems, primary and secondary radar. The secondary radar publishes more information than primary radar which only provides basic information about the aircraft position. These radars typically publish using different RTPS topics. If suddenly a different machine starts to publish data with either radars RPTS topic then the integrity of information provided to the controller is lost. This causes a lost of confidence in the system by the controllers and pilots that rely on it and may even cause an air traffic incident.

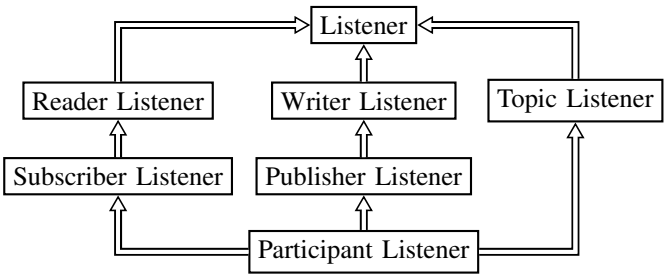

Fig. 3. Listeners of RTPS Protocol

5) Publishing frequency (C9): Participants are attached with three kind of listeners: subscriber, publisher, and topic listeners. Figure 3 describes different type of listeners call backs in RTPS protocol. RTPS uses either synchronous or asynchronous mode for communication. In synchronous mode, the listeners wait synchronously using a wait set with query conditions [6]. This creates a big security concern for publishers.

In synchronous mode, a publisher has to publish in a fixed frequency rate – C9.

By changing query condition of the wait set, a publisher can be blocked from publishing by putting it in sleep mode forever. This is a classic style to run a DoS attack.

6) Quality of Service (C10): Before starting communication between publisher and subscriber of the same topic, it is ensured that the QoS is compatible between them. They should have a fixed request data rate and delivery level during this full period of time.

The QoS policy between a subscriber and publisher cannot be altered during run-time of the application – C10.

Since the communication will break down if the QoS is mismatched, any type of alteration to service policy leads to DoS attacks.

IV. Implementation

This section presents a brief description of the framework's implementation. This framework has been implemented in C++ in Linux environment (Ubuntu 14.04). It only requires a gcc compiler and an open source C/C++ network packet capture library named libpcap to run.

In this framework, a separate grammar is used for each of the Internet Protocol classes of the protocol. There is a grammar for the RTPS messages, which will eventually include all of the UDP protocols [15] on the network. For IGMP protocol, a different grammar is used. The leftmost token analysis of the grammar allows us to identify that the field *type* is the initial field of all IGMP messages, and can be factored out while parsing. This allows the constraint to be hard-coded into the parser.

Returning the structure as a return value of the Parser would require a union type or the use of an inheritance class hierarchy. The Constraint Checker will duplicate part of the Parser's functionality to determine which structure had been returned. We instead implement multiple entry points into the Checker, one for each packet type. As mentioned before, this part of the Checker will be generated automatically from the protocol description and the constraints written in SCL.

The facts needed for the constraints are implemented as a set of fixed length Member Pools, one for each group address/net address pair. Each element of the queue contains a list of the reports for an associated IGMP query. The pool also contains a separate list of the members who joined before the traffic experienced any kind of query. Both of the pools are constantly updated whenever host leaves a group or does not respond to two consecutive queries.

For RTPS packets, each time a topic is announced, the authenticity of subscriber or publisher is verified by checking these pools. A publisher generally associates itself with a data writer and sends DATA(w) packets where subscriber through data reader gets DATA(r) packets [6]. Token keys of reader and writer objects associated with a participant are constantly checked to ensure that they are not the same key. A separate pool mapping hosts and tokens are managed to validate the identity of the token publishers.

For two timer constraints **C6** and **C9**, the threshold time is learned from the learning mode. In learning mode, the traffic flow is scrutinized to learn settings of that network. A certain type of packets such as HEARTBEAT and DATA(p) [6] are expected from specific hosts in between these marked up timers, otherwise, these constraints are considered to be violated.

V. Evaluation

To evaluate our approach we need a source of RTPS and IGMP traffic. The Canadian Automated Air Traffic System (CAATS) [16] uses the RTPS protocol to share data between devices, controllers and ATC centers [17]. This is used as an inspiration to adapt the Euroscope Simulator and ATC Display [18] to use the RTPS protocol to communicate between the Flight Simulator server and the virtual ATC display [19]. Euroscope is an ATC control console popular with virtual aviation community that consists of pilots flying for virtual airlines using flight simulators and air traffic controllers using Euroscope to control the virtual airspace. The use of Euroscope as a basis for the simulator provides an arms length source of network data to use for our experimentation. During this evaluation, we test the performance and reliability of the proposed framework.

A. Environment

Figure 4 shows the simulator as deployed at Queen's University. The network architecture is built based on the proposal from Lemay et al. [20]. Two machines, FS1 and FS2, are connected by three networks. One network, labeled *control*, connects to both machines and the router and is used to control the experiment. Two networks, *Exp1* and *Exp2* carry the experimental data. Both of the machines, the router, and the networks are deployed in the KVM (Kernel-based Virtual Machine) environment [21].

Two modules, implemented in python, provide an interface between the Euroscope simulation server, the Euroscope ATC display and the experimental networks, respectively. These modules take the existing Euroscope protocol and repackage it as RTPS messages using the OpenDDS [22] framework. By placing the two components on separate network segments the router generates the appropriate IGMP traffic. With the help of Wireshark on the router, we capture the traffic from one of the experimental networks.

B. Detection Results

For properly assessing the effectiveness of the framework, several malware are needed to perform DoS attacks on

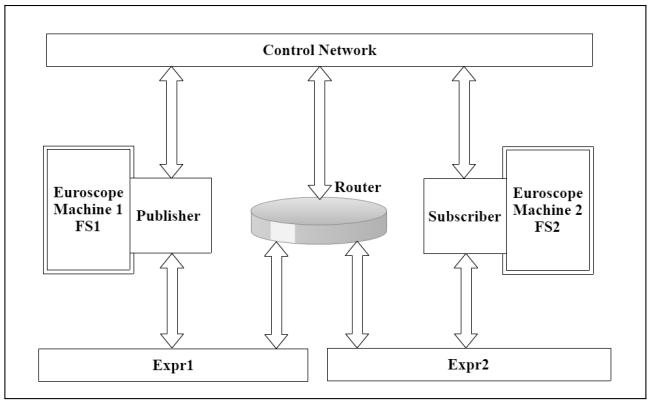

Fig. 4. Network Architecture of the ATC Simulator

the experimental network. Instead of following that path, numerous attack patterns are crafted in the captured PCAP file that resembles some DoS attack to RTPS protocol. For editing the PCAPs, WireEdit [23] tool is used. We craft some malicious hosts and non-responsive known hosts on the captured stream. Specific bytes of packet are altered to plot the attack of run time QoS setting's modification.

Table 1 shows the full success we have over alerting constraint failures on ten attacks. We do believe that two timer constraints **C6** and **C9**, and QoS setting constraint **C10** should produce false alarms [24] in a real network. However, with current environment this rate cannot be determined.

To reduce false alarms, constraints need to be validated. During experimental evaluation, all the constraints are checked and we manage to find one invalid constraint. According to IGMP manual, the first version of **C3** was:

After receiving a general query, only one host from that network will eventually send its membership report.

With this version, the framework constantly fires **C3** as all the members start to send their membership reports periodically. After careful inspection of host response module, we conclude that this is the normal behavior and update **C3**.

Constraints	Number of DoS attacks	Detection
C1, C2, C3, C4	3	yes
C5, C6	4	yes
C7, C8, C9	2	yes
C10	1	yes

TABLE I: Detection of Constraint Failures

C. Performance

We have captured the network traffic while running the flight simulator for approximately one hour. The captures consist of RTPS packets, with 0.48% IGMP packets. The PCAP files are grown using the mergecap tool resulting in three test files 667, 1331, and 2662 MB in size. Running as a single thread on a 3rd generation core i5-3337U Mobile 2.7 GHz laptop with 7.7 GB of RAM, the files take between 3.9 and 15.6 seconds to process, for an average throughput of 168.6 MB/s. In all cases, all the constraints are satisfied over the simulated air traffic data only containing non-attack packets. Better processing rate can easily be achieved by throwing multiple threads to the process on a more powerful machine. It shows that this framework can capture, parse and satisfy constraints at the same level of accuracy on a real-time basis.

VI. Related Work

Intrusion detection is a well-established research area. Countless mechanisms do exist for recognizing intrusions from anomalous network behavior. Despite all these facts, developing a full-fledged IDS based on constraint checking is considered a novel approach. Constraint checking has been used to solve problems in many fields of research area [25]. On the contrary, examples of researches going on constraints in network application are not that common.

Few approaches can only be found similar to this framework. One approach is to develop constraints for both hosts and the private network ensuring proper authentication [26]. Another method is shown in [27] which develops critical rules for SCADA system in a private network. The common approach in these techniques is developing and evaluating simple rules after careful examination of the full network architecture and host specification.

For any type of IDS, validating the packet structure and extraction of information should be the first essential step to complete. A number of research methods show different kinds of packet parsing techniques [28], [29]. Accuracy and speed of parsing are the key factors as they change the detection speed of an IDS. Automatic generation of protocol parser receives some attention from the research community. Gibb et al. [30] discusses a programmable parser based on parse graphs. This approach is similar to our original parser that is too slow to handle the traffic of our industrial partner. Zhang et al. [31] shows the use of lex and yacc in a parallel environment to achieve speed in packet decoding. Bangart and Zeldovich [32] also investigate the automatic generation of a packet parser from a context sensitive grammar. Their approach is tuned for application network stacks, while this framework is designed to encode multiple protocols in a single grammar, including constraints between multiple protocols. Significant research are conducted in the custom code to implement a parser in Domain Specific Languages. One of the most successful parser generators is the ANTLR [33]. We use a similar approach but include context sensitive parsing constraints.

Several open source detection tools are currently available such as Snort, Suricata, and Bro [34], [35]. These tools offer highly effective features and gigantic rule sets. With the help of customized plugins, they are also capable of processing IGMP and RTPS protocol. However, it requires building complicated rules to prevent any serious attack. These rules are a lower level of abstraction where our framework aims at a higher level. Network constraints are built from the idea of detection rules. Snort has a feature called dynamic/activate rule [11]. This type of rule triggers a set of actions to monitor one specific event. The role writer needs to make

the proper assumption of the timing of the sequences rather than anticipating the next packet of interest for some period. This is where the proposed network constraints are powerful with their liveliness properties.

VII. Conclusions And Future Work

The importance of keeping a private network safe and secure is rising with growing cases of newer kinds of attacks. In this paper, we propose a basic framework to recognize the valid network constraints of some protocols. An initial evaluation of the framework is provided to show that constraints can be used to express normal traffic of two protocols. Tests are conducted on the framework against captured and altered data from an experimental private network. The results show that the constraints can be evaluated against normal traffic. The next goal is to determine the rate of attack success in case of any constraint failure and capability of recognizing unfamiliar attacks. We will improve the threat model for RTPS and use those threats to examine the set of constraints implemented and introduce anomalous traffic into our simulation. We will also introduce several more protocols into the simulation, including Network File System (NFS) and Network Time Protocol (NTP).

ACKNOWLEDGMENT

This research is funded in part by the National Science and Engineering Research Council of Canada (NSERC), the Department of National Defense (DND), Canada, and the Ontario Research Fund (ORF), Canada.

References

[1] R. P. Andrew Clark, Quanyan Zhu and T. Basarauthor, "An impact-aware defense against stuxnet american control conference, ACC, washington, dc, usa," 2013.

[2] D. E. Denning, "An intrusion-detection model." IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, 1987.

[3] V. Jyothsna, S. V. Engineering, A. R. Tirupati, V. V. R. Prasad, S. V. Engineering, A. R. Tirupati, K. M. Prasad, and S. V. Engineering, "A review of anomaly based intrusion detection systems."

[4] P. Garcia-Teodoro, J. E. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," 2009.

[5] B. Fenner, H. He, B. Haberman, and H. Sandick, "Internet group management protocol (igmp) multicast listener discovery (mld)-based multicast forwarding (igmp/mld proxying)." https://tools.ietf.org/html/rfc4605, 2006.

[6] H. S. snf A. Rao and R. Lanphier, "Real time streaming protocol (rtsp)." https://www.ietf.org/rfc/rfc2326.txt, 1998.

[7] "Tcpdump." http://www.tcpdump.org/cgi-bin/cvsweb/cvs/tcpdump/README?rev=1.66.

[8] "Wireshark." https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html.

[9] S. Marquis, T. R. Dean, and S. Knight, "Packet decoding using context sensitive parsing," in *Proceedings of the 16th IBM Centre for Advanced Studies Conference(SCASCON 06)*, pp. 263–274, 2006.

[10] S. Marquis, T. R. Dean, and S. Knight, "Scl: A language for security testing of network applications," in *Proceedings of the 2005 conference of the Centre for Advanced Studies on Collaborative Research (CASCON 05)*, pp. 16–19, 2005.

[11] "Snort dynamic rules." http://archive.oreilly.com/pub/h/1393.

[12] D. Odden, "Rules v. constraints," 2007.

[13] C. Valli, A. Woodward, C. Carpene, P. Hannay, M. Br, R. Karvinen, and C. Holme, "Eavesdropping on the smart grid."

[14] P. A. Melgarejo, P. Laborie, and C. Solnon, "A time-dependent no-overlap constraint: Application to urban delivery problems." Integration of AI and OR Techniques in Constraint Programming - 12th International Conference, CPAIOR 2015, Barcelona, Spain, May 18-22, 2015, Proceedings.

[15] G. Gibb, G. Varghese, M. Horowitz, and N. McKeown, "Design principles for packet parsers," 2013.

[16] N. CANADA, "Av canada: Media - the canadian automated air traffic system." http://www.navcanada.ca/EN/media/Pages/publicationscorporate-direct-route-story-1.aspx.

[17] N. CANADA, "Av canada: Media - nav canada enhances atm technology platform with rti connext." http://www.navcanada.ca/EN/media/Pages/news-releases-2013-nr25.aspx, 2013.

[18] G. Csernak, "Euroscope - power of control." http://www.euroscope.hu.

[19] L.-P. Morel, J. M. Fernandez, S. Guigui, and T. R. Dean, "Adapting a Virtual Flight Simulator to DDS," tech. rep., Ecole Polytechnique, 09 2016.

[20] A. Lemay, J. M. Fernandez, and S. Knight, "An isolated virtual cluster for SCADA network security research," in *1st International Symposium for ICS & SCADA Cyber Security Research 2013, ICS-CSR 2013, 16-17 September 2013, Leicester, UK*, 2013.

[21] R. Shea and J. Liu, "Network interface virtualization: challenges and solutions," *IEEE Network*, vol. 26, no. 5, pp. 28–34, 2012.

[22] OCI, "Opendds." http://opendds.org.

[23] "Wireedit." https://wireedit.com/.

[24] S. Patton, W. Yurcik, and D. Doss, "An achilles' heel in signature-based ids: Squealing false positives in snort," 2001.

[25] H. Simonis, "Building industrial applications with constraint programming." Constraints in Computational Logics: Theory and Applications, International Summer School, CCL'99 Gif-sur-Yvette, France, September 5-8, 1999, Revised Lectures.

[26] C. Ko, P. Brutch, J. Rowe, G. Tsafnat, and K. N. Levitt, "System health and intrusion monitoring using a hierarchy of constraints," pp. 190–204, Recent Advances in Intrusion Detection, 4th International Symposium, RAID 2001 Davis, CA, USA, October 10-12, 2001, Proceedings, 2001.

[27] A. Carcano, I. N. Fovino, M. Masera, and A. Trombetta, "State-based network intrusion detection systems for SCADA protocols: A proof of concept," pp. 138–150, Critical Information Infrastructures Security, 4th International Workshop, CRITIS 2009, Bonn, Germany, September 30 - October 2, 2009. Revised Papers, 2009.

[28] G. Gibb, G. Varghese, M. Horowitz, and N. McKeown, "Design principles for packet parsers." Symposium on Architecture for Networking and Communications Systems, ANCS '13, San Jose, CA, USA, October 21-22, 2013.

[29] K. Zhang, J. Wang, B. Hua, and X. Tang, "Building high-performance application protocol parsers on multi-core architectures." 17th IEEE International Conference on Parallel and Distributed Systems, ICPADS 2011, Tainan, Taiwan, December 7-9, 2011.

[30] G. Gibb, G. Varghese, M. Horowitz, and N. McKeown, "Design principles for packet parsers," in *Architectures for Networking and Communications Systems (ANCS), 2013 ACM/IEEE Symposium on*, pp. 13–24, Oct 2013.

[31] K. Zhang, J. Wang, B. Hua, and X. Tang, "Building high-performance application protocol parsers on multi-core architectures," in *Parallel and Distributed Systems (ICPADS), 2011 IEEE 17th International Conference on*, pp. 188–195, Dec 2011.

[32] J. Bangert and N. Zeldovich, "Nail: A practical tool for parsing and generating data formats," in *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, (Broomfield, CO), pp. 615–628, USENIX Association, Oct. 2014.

[33] T. Parr, *The Definitive ANTLR Reference Building Domain-Specific Languages*, vol. 4. The pragmatic bookshelf, 4 ed., 05 2007.

[34] O. Eldow, P. Chauhan, P. Lalwani, M. B. Potdar, and G. P. School, "Computer network security ids tools and techniques (snort/suricata)."

[35] V. Paxson, "Bro: A system for detecting network intruders in real-time," 1999.