1 2 9 0

UNIVERSIDADE Ð
COIMBRA

Luis Miguel Batista Rosa

INTRUSION AND ANOMALY DETECTION IN INDUSTRIAL AUTOMATION AND CONTROL SYSTEMS

Tese no âmbito do Programa de Doutoramento em Ciências e Tecnologias da Informação, orientada pelo Professor Doutor Paulo Alexandre Ferreira Simões e pelo Professor Doutor Edmundo Heitor da Silva Monteiro, e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Dezembro de 2021

1 2 UNIVERSIDADE Ɖ COIMBRA 9 0

DEPARTMENT OF INFORMATICS ENGINEERING
FACULTY OF SCIENCES AND TECHNOLOGY
UNIVERSITY OF COIMBRA

INTRUSION AND ANOMALY DETECTION IN INDUSTRIAL AUTOMATION AND CONTROL SYSTEMS

Luis Miguel Batista Rosa

Doctoral Program in Information Science and Technology
PhD Thesis submitted to the University of Coimbra

Advised by Prof. Dr. Paulo Simões and Prof. Dr. Edmundo Monteiro

December, 2021

1 2 9 0

FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE Ð COIMBRA

Departamento de Engenharia Informática
Faculdade de Ciências e Tecnologia
Universidade de Coimbra

Intrusion and Anomaly Detection in Industrial Automation and Control Systems

Luis Miguel Batista Rosa

Programa de Doutoramento em Ciências e Tecnologias da Informação
Tese de Doutoramento apresentada à Universidade de Coimbra

Orientado pelo Prof. Dr. Paulo Simões e pelo Prof. Dr. Edmundo Monteiro

Dezembro, 2021

This work was partially supported by the following projects:

- CockpitCI (FP7-SEC-2011-1 Project, Grant Agreement ID 285647), funded by the European Union (EU).
- ATENA (H2020-DS-2015-1 Project, Grant Agreement ID 700581), funded by the European Union (EU).
- Mobiwise (reference POCI-01-0145-FEDER-016426), funded by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) of the Portugal 2020 framework, and by Fundação para a Ciência e Tecnologia (FCT).
- 5G – Componentes e Serviços para Redes 5G (POCI-01-0247-FEDER-024539), funded by the European Regional Development Fund (FEDER), through the Competitiveness and Internationalization Operational Programme (COMPETE 2020) and the Lisbon Regional Operational Programme (LISBOA 2020).

Co-funded by the
Horizon 2020
Framework Programme of the
European Union

UNIÃO EUROPEIA

Fundo Europeu
de Desenvolvimento Regional

Fundação
para a Ciência
e a Tecnologia

To my Wife.

Acknowledgements

I would like to take the freedom this section gives me to acknowledge my three supervisors. The completion of my dissertation would not have been possible without the valuable discussions and all the constructive criticism they provided me. In that sense, first, I would like to extend my deepest gratitude to my formal supervisors. Paulo Simões, with his experience and unique way to look at things, always encouraged me on this journey. Invariably, he gave the peace of mind I needed. Then, Edmundo Monteiro whose help and patience cannot be overestimated. In my long journey with ups and downs, whenever I requested, he had a word for me. And last but not least, I would like to express my sincere gratitude to Tiago Cruz. For the careful reader, Tiago does not appear in the first pages as my official supervisor. Nevertheless, after starting this dissertation, I was lucky enough to meet him. Since then, in the last years, I have been working closely with him and we became friends, he was always there to support me and listen to my endless complaints. More, he was kind enough to provide me his expertise and guidance as my supervisor and discuss all the technical details of this dissertation. For all of those reasons, I will always consider him one of my supervisors.

Next, without getting into details to avoid being unfair, I would like to thank my friends and colleagues who really worked with me for the past years and made this journey less lonely. We called ourselves the G54 team. This was a group of young and committed researchers who developed strong relationships way beyond the normal working environment. They consistently offered their help, technical support, and push me to complete my dissertation. I'm a proud member of the G54 spirit. Likewise, I would like to extend my sincere thanks to the LCT research group which I'm also proud to be part of. Their continuous support and all the pleasant moments we get together are truly priceless.

Finally, I'm deeply indebted to my family, and in particular to my wife, for the full support, motivation, love, and patience.

Abstract

In the domain of Industrial Automation and Control Systems (IACS), security was traditionally downplayed to a certain extent, as it was originally deemed as an exclusive concern of Information and Communications Technology (ICT) systems. But things were about to change.

Relying on air-gaping to ensure IACS security has proven to be unrealistic and, ultimately, irresponsible. The myth of the air-gap, as well as other preconceived notions about implicit IACS security, constituted dangerous fallacies that were debunked once successful attacks become known. Ultimately, the industry started shifting away from this dangerous mindset, discussing how to properly secure those systems.

In many ways, IACS security should not be treated differently from modern ICT security. For sure, IACS have distinct characteristics, assets, protocols and even priorities that should be considered – but security should never an optional concern. There is an established idea that, given the sensitive nature of such environments, extra care should be taken when adding security controls or changing anything else, for that purpose – the old *if it works don't touch it* mindset. Together with economic reasons, this has been one of the main causes for the proliferation of insecure, outdated and legacy systems in sensitive production environments.

With cyberattacks becoming increasingly commonplace, IACS often constitute desirable targets, for many reasons. Practices such as the continuous disregard for secure protocols in mission-critical systems, the systematic deferral of crucial updates or the delay in the implementation of proper security mechanisms are just making things worse. This calls for a suitable approach to IACS protection, encompassing: (1) the development of proper security assessment mechanisms/techniques; and (2), the means to mitigate and fix existing security issues. This dissertation is mainly focused on the first aspect, even though it also provides contributions to the second.

This dissertation proposes an holistic and data-driven framework capable of leveraging distinct techniques to increase the situational awareness and provide continuous and near real-time monitoring of Industrial Control Systems (ICS) infrastructures. For such purposes, this thesis proposes an evolution of the Security Information and Event Management (SIEM) concept, geared towards providing a unified security data monitoring solution by leveraging recent advances in the field of real-time Big Data analytics. This evolved SIEM relies on a wide range of open-source components which, despite being presumed to be a good match for the Supervisory Control And Data Acquisition (SCADA) security monitoring needs, are yet not widely explored in the literature. In the same way, the most recent machine-learning-based anomaly-detection techniques (which are becoming increasingly prominent in the cybersecurity field) were also analyzed and studied, in order to understand their benefits for developing and advancing IACS cyber-intrusion detection processes. The proposed mechanisms were developed and validated in the scope of the CockpitCI FP7 and ATENA H2020 projects.

Keywords: Industrial Automation and Control Systems; Cybersecurity; Intrusion Detection; Real-Time Big Data Analytics; SCADA Networks.

Resumo

No domínio dos Sistemas de Automação e Controle Industrial (IACS) a segurança foi tradicionalmente descurada, sendo considerada como uma preocupação exclusiva dos sistemas de Tecnologia da Informação e Comunicação (TIC). Mas as coisas estavam prestes a mudar.

Depender do *air-gap* para garantir a segurança de um IACS provou ser irrealista e, em última circunstância, irresponsável. Esse mito, bem como outras noções preconcebidas sobre segurança implícita, constituíram falácias perigosas que foram desmascaradas assim que ataques bem-sucedidos foram conhecidos. Por fim, a indústria começou a afastar-se dessa mentalidade perigosa, passando a empenhar-se em proteger efectivamente esses sistemas.

Na verdade, de muitas maneiras, a segurança dos IACS não deve ser tratada de forma diferente da segurança de um sistema de TIC moderno. De facto, os IACS têm características, componentes, protocolos e mesmo prioridades distintas que devem ser consideradas – mas a segurança nunca deve ser uma preocupação opcional. Há uma ideia pré-estabelecida de que, dada a natureza sensível de tais ambientes, devem ser tomados cuidados extras ao introduzir mecanismos de segurança ou alterar qualquer outra coisa – o chavão de *se funciona, não mexer*. No entanto, juntamente com razões económicas, essa tem sido uma das principais causas para a proliferação de sistemas inseguros, desatualizados e legados em ambientes de produção.

À medida que os ciberataques se têm tornado cada vez mais comuns, os IACS são frequentemente alvos desejáveis, por variadas razões. Práticas como o uso de protocolos inseguros em sistemas críticos, o adiamento sistemático atualizações críticas ou o protelamento da implementação de mecanismos de segurança, são práticas que agravam esta situação. É pois necessária uma abordagem adequada para a proteção dos IACS, abrangendo: (1) o desenvolvimento de mecanismos / técnicas de avaliação de segurança adequados; e (2), meios para mitigar e corrigir os problemas de segurança existentes. Esta dissertação está focada principalmente no primeiro aspeto, ainda que soluções, medidas de mitigação e medidas reativas sejam igualmente propostas ao longo do texto.

Esta dissertação propõe uma abordagem holística e orientada a dados capaz de alavancar diferentes técnicas por forma a reforçar a consciência situacional e fornecer uma monitorização contínua (e quase em tempo real) de um determinado IACS. Para tal, e tendo em conta os recentes avanços na área das ferramentas de análise de dados em tempo real e *Big Data*, é proposta uma evolução do conceito de *Security Information and Event Management (SIEM)*, focada no fornecimento de uma solução unificada de monitorização de dados de segurança. Este conceito de SIEM melhorado baseia-se numa ampla gama de componentes de código aberto que, apesar de presumivelmente formarem uma boa combinação para suprir as necessidades de monitorização da segurança dos IACS, ainda não são amplamente explorados na literatura. Da mesma forma, as mais recentes técnicas de deteção de anomalias baseadas em *Machine Learning (ML)* (cada vez mais proeminentes no campo da cibersegurança) são também analisadas e estudadas, a fim de compreender os seus benefícios para o desenvolvimento dos processos de deteção de intrusão em IACS. Os mecanismos propostos foram desenvolvidos e validados no âmbito dos projectos CockpitCI FP7 e ATENA H2020.

Palavras-Chave: Sistemas de Automação e Controle Industrial; Cibersegurança; Detecção de Intrusões; Análise de *Big Data* em tempo real, Redes SCADA

Foreword

The work described in this thesis was conducted at the Laboratory of Communication and Telematics (LCT) of the Centre for Informatics and Systems of the University of Coimbra (CISUC), in the context of two European projects:

- **FP7 CockpitCI,** Cybersecurity on SCADA: risk prediction, analysis and reaction tools for Critical Infrastructures, No 285647, FP7-SEC-2011-1.

- **H2020 ATENA,** Advanced Tools to assEss and mitigate the criticality of ICT compoNents and their dependencies over Critical InfrAstructures, No 7005813, H2020-DS-2015-1.

Among other goals, both projects had a strong emphasis on the research of new approaches for monitoring Industrial Automation and Control Systems (IACS), detecting cyber-physical anomalies in near real-time and, later, exposing that information as a decision-support mechanism for the system operator. Additional topics such as Critical Infrastructure (CI) interdependencies, process modeling and risk assessment were also addressed by these projects but are not directly related with the core of this dissertation. Within this research, we leveraged a hybrid testbed developed at the Israel Electric Corporation premisses for CockpitCI and further enhanced within ATENA, designated as Hybrid Environment for Development and Validation (HEDVa). Such testbed was pivotal, enabling the exploration and implementation of different Supervisory Control And Data Acquisition (SCADA) cyber-attack scenarios. It allowed, for instance, to explore SCADA protocols less known in the literature, such as PCOM.

The outcomes of this thesis contributed to the state-of-the-art analysis of both projects, to the conceptualization and definition of their detection layers, and with a Machine-Learning (ML)-based proof-of-concept mechanism to aggregate and classify network traffic, and different use cases in the form of cyber-attacks, which significantly contributed to the demonstration and evaluations activities of these projects. Part of this dissertation was also reflected in CockpitCI and ATENA deliverables.

The work conducted in the scope of this dissertation also led to the publications listed next.

Journal Papers:

- **Rosa, L.**, Cruz, T., Freitas, M., Quitério, P., Henriques, J., Caldeira, F., Monteiro, E., and Simões, P. (2021). Intrusion and anomaly detection for the next- generation of industrial automation and control systems. Future Generation Computer Systems, 119:50–67. DOI: 10.1016/j.future.2021.01.033

  Main Contributions: State-of-the-art review of intrusion and anomaly detection techniques in the context of IACS. Conceptualization and development of a proof-of-concept ML-based pipeline for network data classification. Discussion and evaluation of various event processing mechanisms and ML supervised techniques.

- Lima, A., **Rosa, L.**, Cruz, T., and Simões, P. (2020). A security monitoring framework for mobile devices. Electronics, 9(8):1197. DOI: 10.3390/electronics9081197.

  Main Contributions: Contribution to the conceptualization of the mobile monitoring framework proposed in the paper, as well as to the evaluation activities.

- **Rosa, L.**, Freitas, M., Mazo, S., Monteiro, E., Cruz, T., and Simões, P. (2019). A comprehensive security analysis of a scada protocol: From osint to mitigation. IEEE Access, 7:42156–42168. DOI: 10.1109/ACCESS.2019.2906926.

  Main Contributions: Conceptualization, investigation and development of various tools to perform a security analysis of the PCOM protocol. Collection and preparation of a PCOM dataset.

- Foglietta, C., Masucci, D., Palazzo, C., Santini, R., Panzieri, S., **Rosa, L.**, Cruz, T., and Lev, L. (2018). From detecting cyber-attacks to mitigating risk within a hybrid environment. IEEE Systems Journal, 13(1):424–435. DOI: 10.1109/JSYST.2018.2824252.

  Main Contributions: Participation in the conceptualization of the cyber attack detection subsystem and development of the validation use cases in the form of attack scenarios.

- Adamsky, F., Aubigny, M., Battisti, F., Carli, M., Cimorelli, F., Cruz, T., Di Giorgio, A., Foglietta, C., Galli, A., Giuseppi, A., Liberati, F., Neric, A., Panzieri, S., Pascucci, F., Proenca, J., Pucci, P., **Rosa, L.**, Soua, R. (2018). Integrated protection of industrial control systems from cyber-attacks: the ATENA approach. International Journal of Critical Infrastructure Protection, 21:72–82. DOI: 10.1016/j.ijcip.2018.04.004.

  Main Contributions: Design of the ATENA cyber-detection layer and participation on the specification of the overall ATENA architecture.

- Graveto, V., **Rosa, L.**, Cruz, T., and Simões, P. (2019). A stealth monitoring mechanism for cyber-physical systems. International Journal of Critical Infrastructure

Protection, 24:126–143. DOI: 10.1016/j.ijcip.2018.10.006

Main Contributions: Participation in the conceptualization of different evaluation use cases in the form of cyber-attacks, as part of the evaluation of the Shadow Security Unit (SSU) concept – a SCADA-specific probe.

- Stewart, B., **Rosa, L.**, Maglaras, L. A., Cruz, T. J., Ferrag, M. A., Simoes, P., and Janicke, H. (2017). A novel intrusion detection mechanism for scada systems which automatically adapts to network topology changes. EAI Endorsed Transactions on Industrial Networks and Intelligent Systems, 4(10). DOI: eai.1-2-2017.152155

  Main Contributions: Participation in the implementation and deployment of different use cases in the form of cyber-attacks, as part of the evaluation of an One Class Support Vector Machine (OCSVM) anomaly detection module for network traffic classification.

- Cruz, T., **Rosa, L.**, Proença, J., Maglaras, L., Aubigny, M., Lev, L., Jiang, J., and Simões, P. (2016). A Cybersecurity Detection Framework for Supervisory Control and Data Acquisition Systems. IEEE Transactions on Industrial Informatics, 12(6):2236–2246. DOI: 10.1109/TII.2016.2599841.

  Main Contributions: Participation in the specification of the CockpitCI Distributed Intrusion Detection System. Deployment and evaluation of various detection agents and event processing capabilities of the platform.

Conference papers:

- Borges de Freitas, M., Quitério, P., **Rosa, L.**, Cruz, T., and Simões, P. (2020). SDN-assisted containerized security and monitoring components. In NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium - Poster Session, pages 1–5. IEEE. DOI: 10.1109/NOMS47738.2020.9110291.

  Main Contributions: Participation in the conceptualization of SDN-assisted probes.

- Borges de Freitas, M., **Rosa, L.**, Cruz, T., and Simões, P. (2018). SDN-enabled virtual data diode. In Katsikas S. et al. (eds) Computer Security. SECPRE 2018, CyberICPS 2018. Lecture Notes in Computer Science, vol 11387., pages 102–118. Springer, Cham. DOI: 10.1007/978-3-030-12786-2_7

  Main Contributions: Participation in the conceptualization of a SDN-enabled data diode and in the definition of the evaluation scenario.

- **Rosa, L.**, Cruz, T., Simões, P., Monteiro, E., and Lev, L. (2017a). Attacking SCADA systems: A practical perspective. In 2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal. IEEE. DOI:

10.23919/INM.2017.7987369.

Main Contributions: Development and discussion of multiple practical attack use cases against SCADA systems.

- **Rosa, L.**, Proença, J., Henriques, J., Graveto, V., Cruz, T., Simões, P., Caldeira, F., and Monteiro, E. (2017b). An evolved security architecture for distributed industrial automation and control systems. In European Conference on Cyber Warfare and Security, pages 380–390. Academic Conferences International Limited. ISBN: 978-1-911218-43-2

  Main Contributions: Conceptualization and refinement of the main ideas and building blocks of the cyber detection layer of the ATENA.

- Stewart, B., **Rosa, L.**, Maglaras, L., Cruz, T. J., Simões, P., and Janicke, H. (2016). Effect of network architecture changes on OCSVM based intrusion detection system. In International Conference on Industrial Networks and Intelligent Systems, pages 90–100. Springer. DOI: 10.1007/978-3-319-52569-3_8

  Main Contributions: Participation in the implementation and deployment of a set of different use cases in the form of cyber-attacks, as part of the evaluation of an OCSVM anomaly detection module for network traffic classification.

- **Rosa, L.**, Alves, P., Cruz, T., Simões, P., and Monteiro, E. (2015). A comparative study of correlation engines for security event management. In ICCWS 2015-The Proceedings of the 10th International Conference on Cyber Warfare and Security, page 277. ISBN: 978-1-910309-98-8

  Main Contributions: State-of-the-art review and evaluation of a set of event correlator tools.

Book Chapters:

- **Rosa, L.**, Borges de Freitas, M., Henriques, J., Quitério, P., Caldeira, F., Cruz, T., and Simões, P. (2020). Evolving the security paradigm for industrial iot environments. In Cyber Security of Industrial Control Systems in the Future Internet Environment, pages 69–90. IGI Global.

  Main Contributions: Design and development of an Intrusion and Anomaly Detection System.

In addition to the aforementioned publications, the work hereby presented led to the following open-source contributions in the form of tools and datasets:

- **Wireshark PCOM dissector,** a Wireshark dissector for the PCOM protocol that constitutes an extra step to better understand the protocol while providing a starting point for further research. This dissector supports the different PCOM modes – PCOM/TCP, PCOM/ASCII and PCOM/Binary – allowing to easily dissect the structure of PCOM messages, both through graphical and command-line interfaces. The dissector was accepted into the upstream Wireshark repository.

Rosa, L. (2019e). pcomtcp: dissection of additional pcom/ascii fields.
`https://code.wireshark.org/review/#/c/31467/`
Rosa, L. (2019f). pcomtcp: new built-in dissector for pcom protocol.
`https://code.wireshark.org/review/#/c/30823/`
Rosa, L. (2019g). pcomtcp: Pcom/binary command to descriptions.
`https://code.wireshark.org/review/#/c/31858/`

- **NMAP CIP Scan** Leveraging a specific function code, this network scan that can be used by NMAP to collect all tag names and types for Allen-Bradley Logix 5000 equipment. The scan script was submitted to upstream NMAP code repository.

  Rosa, L. (2019h). Scada CIP enum scan.
  `https://github.com/nmap/nmap/pull/1539/`.

- **NMAP PCOM Scan** , that allows NMAP to collect device information for Unitronics PLCs via PCOM protocol. The scan script was submitted to upstream NMAP code repository.

  Rosa, L. (2019i). SCADA scan to collect information from Unitronics Plcs via PCOM protocol.
  `https://github.com/nmap/nmap/pull/1445/`

- **Two Metasploit modules,** used to implement several proof-of-concept attacks and to test PCOM Snort rules. They can also be used for security auditing procedures in future security assessments. Both modules were accepted into the upstream Metasploit code repository.

  Rosa, L. (2019a). New module pcomclient.
  `https://github.com/rapid7/metasploit-framework/pull/11219/`
  Rosa, L. (2019b). New pcom module to send admin commands.
  `https://github.com/rapid7/metasploit-framework/pull/11220/`

- **SCAPY PCOM Layer,** a SCAPY layer to easily decode, manipulate and craft PCOM network packets in an automated fashion. This layer was accepted into the upstream SCAPY code repository.

  Rosa, L. (2019j). Scapy - add a new pcom layer.
  `https://github.com/secdev/scapy/pull/1898/`

- **A Snort Ruleset** which adds PCOM signature-based detection capabilities to Snort. Such rules allow to easily match different PCOM function codes which can be used to detect and limit unwanted network communications for a given scenario. Those rules were integrated in the upstream Snort community ruleset.

  Rosa, L. (2019c). New snort rules for pcom protocol.
  `https://marc.info/?l=snort-sigs&m=154746968717558`

- **A Dataset of PCOM traffic,** generated to support PCOM protocol analysis efforts and tool development processes. For instance, such dataset can be used to evaluate the detection capabilities of a Network Intrusion Detection System (NIDS) in the context of a SCADA system.

  Rosa, L. (2019d). Pcom pcap captures.
  `https://github.com/lmrosa/pcom-misc/tree/master/pcaps`

Contents

List of Figures

List of Tables

List of Acronyms

ACL Access-control list.
AI Artificial Intelligence.
API Application programming interface.
APT Advanced Persistent Threat.
ARP Address Resolution Protocol.
AUC Area under ROC Curve.

BRMS Business Rules Management System.

CAN Controller Area Network.
CCA Canonical Correlation Analysis.
CEP Complex Event Processing.
CI Critical Infrastructure.
CIP Common Industrial Protocol.
CNN Convolutional Neural Network.
CPU Central Processing Unit.

DAG Directed Acyclic Graph.
DCS Distributed Control Systems.
DER Distributed Energy Resources.
DiD Defense-in-Depth.
DNP3 Distributed Network Protocol 3.
DNS Domain Name System.
DoS Denial of Service.
DPI Deep Packet Inspection.
DPIDS Dynamic Perimeter Intrusion Detection System.
DSL Domain Specific Language.

EE Elliptical Envelope.

EMS	Energy Management Systems.
EPL	Event Processing Language.
ES	Essential Services.
FNN	Feedforward Neural Network.
FPGA	Field-programmable gate array.
GBT	Gradient-boosted trees.
GPU	Graphics Processing Unit.
GRU	Gated Recurrent Unit.
HEDVa	Hybrid Environment for Development and Validation.
HIDS	Host Intrusion Detection System.
HITL	Human in the Loop.
HMI	Human-Machine Interface.
HNA-NN	Hierarchical Neuron Neuron Architecture based Neural Network.
HTTP	Hypertext Transfer Protocol.
HTTPS	HTTP Secure.
IaaS	Infrastructure as a Service.
IACS	Industrial Automation and Control Systems.
IADS	Intrusion and Anomaly Detection System.
ICA	Independent Component Analysis.
ICS	Industrial Control Systems.
ICT	Information and Communications Technology.
IDMEF	Intrusion Detection Message Exchange Format.
IDS	Intrusion Detection System.
IEC	International Electrotechnical Commission.
IF	Isolation Forests.
IIoT	Industrial Internet of Things.
IODEF	Incident Object Description Exchange Format.
IoT	Internet of Things.
IP	Internet Protocol.
IT	Information Technology.
IWP-CSP	Intrusion Weighted Particle based Cuckoo Search Optimization.
JSON	JavaScript Object Notation.
KNN	K-nearest Neighbor.

LOF Local Outlier Factor.

LSTM Long short-term memory.

MAC Medium Access Control.

MES Manufacturing Execution Systems.

MitM Man-in-the-middle.

ML Machine-Learning.

MLPC Multilayer Perceptron classifier.

MQTT Message Queuing Telemetry Transport.

NIDS Network Intrusion Detection System.

NIST National Institute of Standards and Technology.

NSE Nmap Scripting Engine.

OCSVM One Class Support Vector Machine.

OPC Open Platform Communications.

OS Operating System.

OS-ELM Online Sequential Extreme Learning Machine.

OSI Open Systems Interconnection.

OSINT Open-source intelligence.

OT Operational Technology.

PaaS Plataform as a Service.

PCA Principal Component Analysis.

PIT Packet Inter-arrival Time.

PLC Programmable Logic Controller.

POJO Plain Old Java Objects.

PTES Penetration Testing Execution Standard.

QDA Quadratic Discriminant Analysis.

RBM Restricted Boltzmann machine.

RFID Radio-frequency identification.

RNN Recurrent Neural Network.

RTC Real-time Clock.

RTU Remote Terminal Unit.

SaaS Software as a Service.

SASL Simple Authentication and Security Layer.

SCADA	Supervisory Control And Data Acquisition.
SEC	Simple Event Correlator.
SGAM	Smart Grid Architecture Model.
SHAP	SHapley Additive exPlanation.
SIEM	Security Information and Event Management.
SOAPA	Security Operations and Analytics Platform.
SOAR	Security Orchestration, Automation and Response.
SOD	Subspace Outlier Degree.
SPOF	Single Point of Failure.
SQL	Structured Query Language.
SSL	Secure Sockets Layer.
SSU	Shadow Security Unit.
SVM	Support Vector Machine.
TCP	Transmission Control Protocol.
TIC	Tecnologia da Informação e Comunicação.
TLS	Transport Layer Security.
UDT	User-defined Type.
URI	Uniform Resource Identifier.
UUID	Universal Unique Identifier.
VLAN	Virtual Local Area Network.
VPN	Virtual Private Network.
XDR	Extended Detection and Response.
XML	Extensible Markup Language.
YAML	YAML Ain't Markup Language.

1 Introduction

This thesis addresses the cybersecurity of Industrial Automation and Control Systems (IACS), a subset of Industrial Systems used to manage Essential Services (ES) and Critical Infrastructures (CIs) such as insdustrial plants and electricity, water, oil and gas production and distribution networks. More specifically, among other contributions, this thesis covers aspects such as the security analysis of Supervisory Control And Data Acquisition (SCADA) protocols and the conceptualization and design of advanced intrusion and anomaly detection solutions for such environments.

The motivation and problem statement are discussed next, followed by the presentation of the specific research question addressed by this work and its objectives. Next, the key contributions that resulted from this work are identified, followed by a description of the structure of the remainder of this thesis.

1.1 Motivation and Problem Statement

A successful cyber-attack against mission-critical IACS can lead to massive financial losses, damage of physical equipment or even human safety hazards. Often, these attacks are also used to conduct industrial cyber-espionage as a modern cyber-warfare weapon. This situation is continuously recalled by incidents such as the famous Stuxnet attack in 2010 [Langner, 2013] or the BlackEnergy attack which left hundreds of thousands of peoples without energy [E-ISAC, SANS, 2016]. The cybersecurity of IACS, sometimes overlooked in the past, is now paramount.

IACS traditionally relied on air-gapped SCADA systems using proprietary protocols and technologies, which provided a false sense of security through physical isolation and obscurity. However, the typical industrial system is no longer a siloed environment, confined to a physically or logically isolated domain.

The increased interconnection paths, the Operational Technology (OT) / Information Technology (IT) convergence and the gradual adoption of Ethernet- and TCP/IP-based networks in IACS faded the perimeter lines between what was assumed to be secure and the outside world. This introduced new attack vectors and amplified the security vulnerabilities from the past, exposing all the insecure protocols and components before hid behind physical barriers.

The majority of SCADA communication protocols still lack proper security enforcing mechanisms, despite the gradual introduction of some level of security support in new versions of some protocols and components. Moreover, many legacy systems without proper security support are still expected to operate for a long time before being replaced (due to economical and technical reasons). This problem is further aggravated because the current IACS / SCADA market fragmentation makes it difficult to assess all the security needs of each vendor's device and communication protocol – protocols found in the field are often vendor-specific and different domains (e.g. electricity distribution, railways) often use domain-specific solutions.

There is now extensive literature devoted to the security of SCADA communication protocols, but it clusters around a very small subset of the most used and well-known protocols. The security enforcing mechanisms of other protocols still lack research and validation, especially for closed or not properly documented protocols.

It should also be noted that secure protocols and components are just one of the building blocks of IACS security. Other key security components are required, such as monitoring tools and intrusion and anomaly detection mechanisms.

Classical tools such as rule-based Network Intrusion Detection System (NIDS) and Host Intrusion Detection System (HIDS) are useful to detect specific and known treats. Nevertheless, and contrary to what could be expected, their support for SCADA is limited. Even the most popular NIDS are often limited to *ad-hoc* rules for SCADA traffic, lacking the appropriate protocol decoding capabilities, richer signatures or stateful protocol decoding for the different protocols.

A large part of the literature addressing IACS security focuses on network anomaly detection, detecting anomalies such as network-based cyber-attacks or physical faults by looking either at physical process properties, SCADA network communications or a combination of both. Other potentially relevant features, such as diversified log sources and host-based events, are typically overlooked, therefore missing the opportunity to develop a more comprehensive approach against a broader spectrum of attacks.

Anomaly detection based on Machine-Learning (ML) techniques, increasingly prominent in other domains, are also expected to bring numerous benefits to IACS, including efficient classification of large amounts of heterogeneous data for spotting anomalies. Nevertheless, they are still presented in the literature as theoretical and isolated works, focused mostly on the applied algorithms.

This gap creates the opportunity to introduce evolved Security Information and Event Management (SIEM) systems – a concept further discussed in Chapter 2 – as a good match for monitoring and integrating a wide range of additional security mechanisms. More than a single Intrusion Detection System (IDS) or a single ML-based anomaly detection algorithm, SIEM systems are expected to bring a global, aggregated and valuable insight into the security state of the infrastructure – taking into consideration a wide number of different data-sources, which can feed multiple anomaly detection mechanisms. Nevertheless, the applicability of SIEMs in SCADA environments is still in its early stages. There are still several open challenges, such as the data formats, integration/interoperability between components or overall platform orchestration, which need to be addressed to achieve practical and complete solutions.

This scenario is further aggravated by the digital transformation we are witnessing, as we move towards the Industry 4.0 and the Industrial Internet of Things (IIoT) paradigms. There is an increased business demand for more distributed and collaborative models which can improve productivity, the decision-making process and, ultimately, maintenance costs. Such rapid expansion, observed in the last years, also represents new challenges and an evolved cyber threat landscape. The advances in technologies such as Artificial Intelligence (AI), edge/cloud computing and 5G foster remarkable technological and business opportunities for those environments, but also pose numerous security challenges. The next-generation IACS are expected to be highly distributed, capillary and, in some cases, multi-tenant, which will require an extremely elastic and flexible security platform that can cope with the increasing volume of data produced by heterogeneous data sources.

This calls for new security approaches, specifically tailored for IACS systems, that can not only help covering classic security flaws associated with IACS but also accommodate the new architectural challenges imposed by the ongoing IACS transformation.

1.2 Research Question and Objectives

Taking into consideration the problems identified in the previous section, this thesis addresses the following research question: **How to improve the security of next-generation IACS through a holistic data-driven framework ?**

The aforementioned research question was addressed by exploring and intersecting several topics, such as IACS cybersecurity, SIEM systems, anomaly detection mechanisms and Big Data analytics.

The work started with a literature review, focused mainly on anomaly-detection approaches and event-processing techniques that (1) address the detection of cyber-physical attacks in the context of IACS systems and (2) could later integrate the proposed framework. Afterwards, different SCADA protocols and cyber-attack scenarios were explored and assessed, to better understand the IACS security landscape. Finally, the SIEM concept was leveraged to provide a unified and distributed holistic, data-driven framework to monitor the cyber-security of IACS, based on state-of-the-art components tailored for SCADA systems. This framework includes the aggregation of multiple data sources, an efficient event-processing layer and anomaly detection capabilities based on AI, ML algorithms and open-source components.

This approach led to three key objectives:

- **Objective 1.** To review, identify and assess intrusion and anomaly detection algorithms and techniques potentially suitable for IACS, so that they could be later incorporated into the aforementioned holistic framework.

- **Objective 2.** To explore different cyber-security scenarios and IACS protocols, in order to better understand the dynamics of cyber-attacks and the vulnerabilities of SCADA tools and protocols.

- **Objective 3.** To devise a holistic data-driven framework, to build a proof-of-concept prototype for demonstration and evaluation purposes, and to validate it in typical IACS scenarios.

1.3 Contributions

This thesis lead to several contributions, of which we emphasize the following:

- **Contribution 1. Security analysis of SCADA protocols in the scope of practical attack scenarios.** Among other specific results, we point out the definition and exploration of practical attack scenarios, often from the attackers' perspective (less used in the literature) and based on testbeds representative of real IACS operated by energy utilities. Regarding the security analysis of SCADA equipment and protocols, besides some initial work focused on the well-known Modbus protocol, a detailed security analysis of PCOM protocol was conducted. This protocol was chosen because it was still not extensively discussed in the literature (from a security standpoint) and was one of the protocols used in the CockpitCI and ATENA testbeds, which made it possible to reproduce various representative use cases and to develop several open-source contributions to widely used tools (as mentioned in the Foreword section). This contribution is reflected mainly in the contents of Chapter 4.

- **Contribution 2. Conceptualization and design of holistic data-driven framework for intrusion and anomaly detection in IACS scenarios.** Leveraging the SIEM and lambda architecture concepts (cf. Chapter 2), this conceptualization and design work addresses challenges such as the integration of multiple, disperse and heterogeneous data-sources, platform elasticity and scalability (for being able to ingest large amounts of disperse data while keeping time-boundaries for data processing within required levels, for streaming and batch processing), and to flexibly accommodate and combine different anomaly detection mechanisms in a neutral fashion. The concepts of this holistic framework are introduced in Chapter 3, while Chapters 5 and 6 further detail the framework's approach to, respectively, streaming processing and anomaly detection.

- **Contribution 3. Integration and evaluation of different mechanisms for classifying network traffic.** As part of the demonstration and validation of the aforementioned framework, several network traffic classification mechanisms were integrated into the framework and evaluated considering various representative scenarios, both for near real-time detection (based on stream processing) and for batch processing. This contribution reflects mainly in the contents of Chapters 5 and 6.

1.4 Structure of the Thesis

The rest of this thesis is organised as follows (cf. Figure 1.1).

- **Chapter 2 Cyber-security in the Scope of IACS**
  Provides a review of the literature, including the current status of IACS cyber-security, IACS-specific ML-based anomaly detection approaches, stream processing techniques and practical framework implementations focused on IACS domains.

- **Chapter 3 A Holistic Intrusion and Anomaly Detection System for IACS**
  After describing the reference architectures adopted by the CockpitCI and the ATENA projects, this chapter introduces the main concepts and building blocks of the proposed holistic Intrusion and Anomaly Detection System (IADS) framework.

- **Chapter 4 Exploratory Analysis of the Security of SCADA Protocols**
  This chapter presents a practical exploratory analysis of different SCADA protocols, tools and attack scenarios, including the attacker's perspective and a more extensive

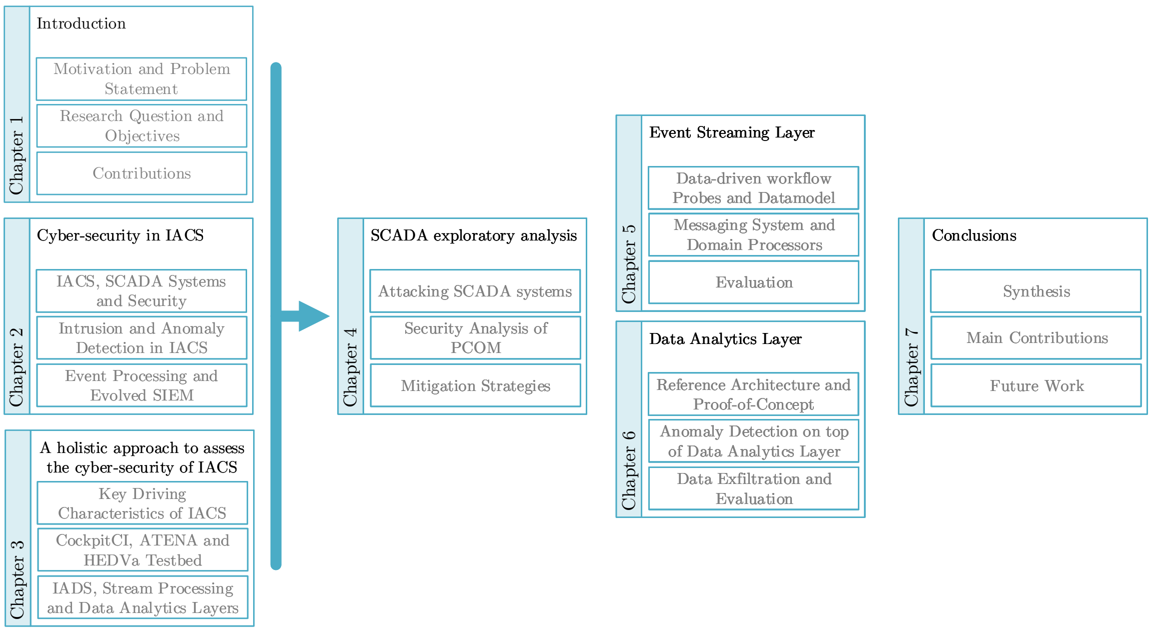

Chapter 1
Introduction
Motivation and Problem Statement
Research Question and Objectives
Contributions
Chapter 2
Cyber-security in IACS
IACS, SCADA Systems and Security
Intrusion and Anomaly Detection in IACS
Event Processing and Evolved SIEM
Chapter 3
A holistic approach to assess the cyber-security of IACS
Key Driving Characteristics of IACS
CockpitCI, ATENA and HEDVa Testbed
IADS, Stream Processing and Data Analytics Layers
Chapter 4
SCADA exploratory analysis
Attacking SCADA systems
Security Analysis of PCOM
Mitigation Strategies
Chapter 5
Event Streaming Layer
Data-driven workflow Probes and Datamodel
Messaging System and Domain Processors
Evaluation
Chapter 6
Data Analytics Layer
Reference Architecture and Proof-of-Concept
Anomaly Detection on top of Data Analytics Layer
Data Exfiltration and Evaluation
Chapter 7
Conclusions
Synthesis
Main Contributions
Future Work

Figure 1.1: Structure of this Thesis

analysis of the PCOM protocol.

- **Chapter 5 Event Streaming Layer**
  This chapter discusses the framework's event streaming layer that supports efficient intercommunication and enables per-domain processing capabilities. The way stream processing techniques were integrated into the platform and used in the overall network traffic classification mechanisms is also discussed in this chapter, as well as the validation work related with the streaming layer and those processing techniques.

- **Chapter 6 Data Analytics Layer**
  This chapter details the framework's analytics layer used to support different types of processing mechanisms and combine various types of supervised ML-based techniques. It also presents the validation work related with the data analytics layer and the evaluation of the mechanisms for network traffic classification.

- **Chapter 7 Conclusions**
  Concluding remarks and future research directions.

2 Cyber-security in the Scope of Industrial Automation and Control Systems

This chapter introduces the topic of cyber-security in the scope of Industrial Automation and Control Systems (IACS).

First, a brief overview of IACS and Supervisory Control And Data Acquisition (SCADA) systems is presented, intended mostly for readers not familiar with this domain (Section 2.1). Then, in Section 2.2, an enlarged discussion of the current status and challenges related with the cyber-security of such systems is provided, encompassing for instance usual threats and vulnerabilities, the typical anatomy of attacks targeting SCADA protocols, and the discussion of defense models, tools and mitigation strategies.

This introduction is followed by an analysis of intrusion and anomaly detection techniques for IACS, focused mostly on a survey of the more relevant advances in this field in the last years (Section 2.3).

Next, we move to the more general problem of correlating and processing events in the IACS cyber-security domain. First, classic correlation tools are analysed, followed by the introduction of more powerful approaches, such as those derived from Big Data event processing architectures (Section 2.4).

Finally, in Section 2.5, the concept of Security Information and Event Management (SIEM) is introduced, as a more unified and holistic approach to monitoring the security of IACS. Relevant related works are discussed and a reference taxonomy for SIEM systems is proposed – centered on the most relevant and envisioned features of the next generation SIEM systems.

It should be noted that this chapter includes content that has already been published ([Rosa et al., 2021] [Rosa et al., 2019] [Rosa et al., 2015]).

2.1 IACS and SCADA Systems

According to the International Electrotechnical Commission (IEC) 62443 standard [International Electrotechnical Commission, 2018], IACS can be defined as a *collection of personnel, hardware, and software that can affect or influence the safe, secure, and reliable operation of an industrial process.* Often, the term Industrial Control Systems (ICS) is also used interchangeably, which as stated by the National Institute of Standards and Technology [Stouffer et al., 2015], *encompasses several types of control systems, including supervisory control and data acquisition (SCADA) systems, Distributed Control Systems (DCS), and other control system configurations such as Programmable Logic Controller (PLC) often found in the industrial sectors and Critical Infrastructures (CIs).* Additional terms, such as CI and Essential Services (ES), are also used, usually to refer to a broader class of domains and services playing a vital role in our modern society, such as energy grids, water distribution systems or transportation systems.

IACS systems have undergone a long evolution since the 1st generation monolithic and isolated systems, progressively becoming more distributed (2nd generation) and networked (3rd generation) [Ujvarosi, 2016]. Today, we are witnessing their fourth revolution, with the adoption of the Industry 4.0 concept.

According to the IEC 62443 definition [ISA, 2007] (Figure 2.1), SCADA systems can be organised into five distinct levels: level 0, reserved for the equipment under control; level 1, containing the local control devices such as Programmable Logic Controller (PLC)s and Remote Terminal Units (RTUs); level 2, for the local supervisory control equipment such as Human-Machine Interface (HMI) devices; level 3, for the global operation management and control center; and level 4, for the remaining business-related systems. The IEC 62443 standard [International Electrotechnical Commission, 2018] also specifies the concept of *zones* as a physical or logical way of segregating the network and assets and the concept of *conduit* to enforce security mechanisms across zones. In distributed industries such as Smart Grids, they are structured as a central control center (Level 3) and multiple remote sites (Levels 0, 1 and 2) spanning across a large geographic area, as depicted in Figure 2.1.

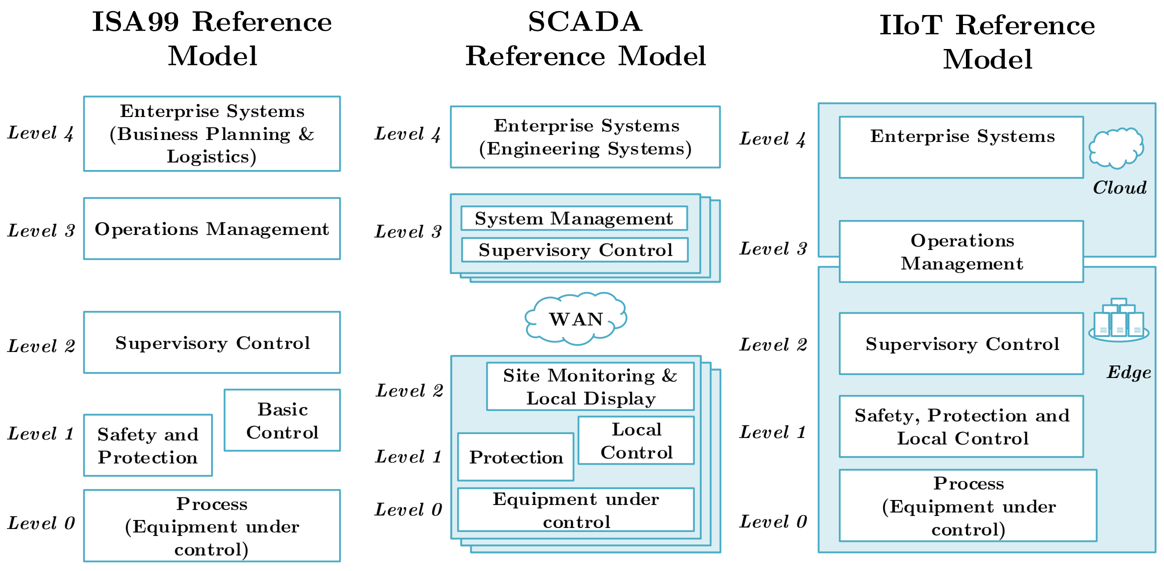


Figure 2.1: ISA99, SCADA and Industrial IoT reference models (adapted from [ISA, 2007] [McLaughlin and McAdam, 2016]).

The PLCs (level 1) are specialized industrial controllers and key components in modern automation, used to replace hard-wired electromechanical relay control systems. They offer a more flexible and cost-effective approach to implement the process and Input/Output logic (Figure 2.2). Contrary to the hard-wiring approach, a PLC can be reprogrammed. This way, a malicious actor can leverage that to easily modify the program logic, which can lead to destructive impact, including physical damage. The majority of SCADA research focuses on classical SCADA communication protocols (e.g. Modbus [Modbus Organization, Inc, 2006]) and does not extensively discuss the capability of remotely reprogramming a PLC (often through proprietary and closed protocols).

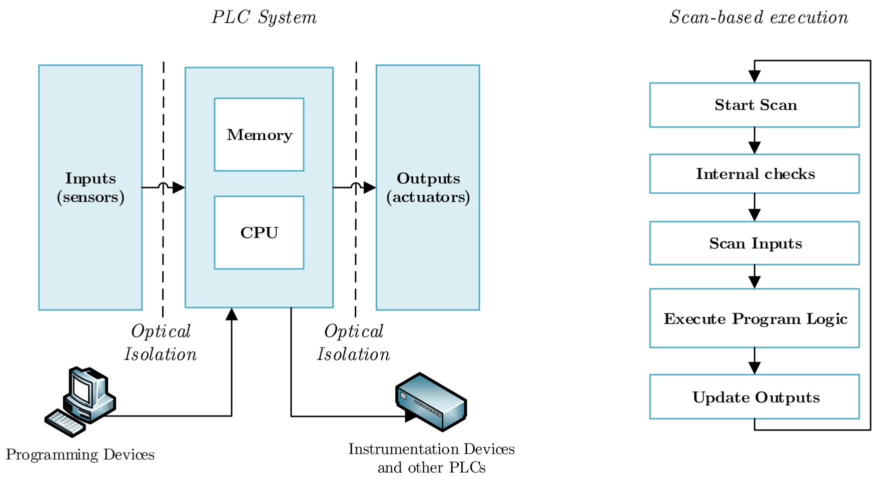


Figure 2.2: PLC architecture and execution overview (adapted from [Gonzalez, 2015]).

Critical Infrastructures (CI)s such as Smart Grids – which can be defined as an improved electricity network with two-way digital communications between suppliers and consumers, composed of an enhanced metering and monitoring infrastructure [Egozcue et al., 2012] – are a common example pointed out in the literature of how the latest technology advances and the Operational Technology (OT) / Information Technology (IT) convergence introduce new security challenges. For Smart Grids, it is also common to refer to the Smart Grid Architecture Model (SGAM), a joint work of CEN, CENELEC and ESTI [Smart Grid Coordination Group, 2012]. On the top of the zones (similar to the levels in the ISA definition), SGAM includes a three-dimensional view of the complete electrical energy conversion chain (from generation to customer premises) and a set of interoperability layers. This is a more complete domain-specific view, that helps to better understand how the different components and layers relate.

IACS systems differ from IT systems [Iturbe et al., 2017]. Their primary focus is on safety, availability and service continuity, rather than confidentiality – as in IT networks. Longer lifetime cycles (to avoid service interruptions or unexpected behaviors) expose them to known vulnerabilities. Industrial physical processes are typically more complex and heterogeneous. Contrary to current IT networks, IACS network communications, depending on the actual use case and components, might range from deterministic and periodic traffic to non-deterministic and aperiodic [5G-ACIA, 2019]. Such network traffic periodicity, when applicable, is often argued to be a useful feature in the context of anomaly detection and network traffic classification.

The next-generation of IACS is expected to be increasingly more distributed and capillary, given the growing number of interconnected devices, in line with the Industrial Internet of Things (IIoT) concept. It is also expected to start adopting cloud-based architectures, shifting the need to maintain a complex infrastructure on-premises to a edge/cloud mix architecture, motivated by cost, reliability and functionality.

According to a 2019 survey of the Capgemini Research Institute to more than 500 executives, intelligence automation (e.g. process automation and Artificial Intelligence (AI)-based technologies) in the energy and utilities market at scale can allow outsized benefits and cost savings from 237 to 813 billions of USD [Capgemini Research Institute, 2019a]. Similarly, Gartner rates *hyperautomation* (including AI and Machine-Learning (ML)-based automation) as the top 2020 technology trend that will transform industries and enterprises in the following years [Gartner, 2020]. As we rapidly advance in the technology, in the near future, what can be seen as Industry 5.0, we can also expect an unprecedented level of automation and AI-based processes, which further increases the complexity of security monitoring (e.g. AI-powered cyber-attacks and the emergence of adversarial attacks – cf. Section 2.2).

Moreover, this shift suggests that security monitoring of such distributed and complex infrastructures might evolve into a Big Data problem. While this may be debatable, since it is still not clear *how big is the data* (or how big it needs to be to become Big Data), using a data-centric and Big Data like approach helps to cope with (1) volume – the amount of information produced by all the interconnected devices; (2) velocity – how to handle real-time information from the physical processes; and (3) variety – how to handle all the heterogeneous information (e.g. sensors data, network traffic, logs).

2.2 Cyber-Security for IACS

In this section we overview the topic of cyber-security for IACS, including the discussion of most relevant threats and vulnerabilities, the analysis of the anatomy of attacks targeting SCADA protocols, and the identification of main defense models and mitigation strategies.

In the last years, IACS security has been extensively discussed [Leszczyna, 2019] [Nazir et al., 2017] [Humayed et al., 2017] [Antón et al., 2017] [Knapp and Langill, 2014] and security practitioners have been rather vocal about its numerous design flaws. The lack of authorization, authentication and encryption in popular SCADA protocols such as Modbus [Modbus Organization, Inc, 2006], Distributed Network Protocol 3 (DNP3) [DNP Users Group, 2005], EthernetIP/Common Industrial Protocol (CIP) [Schiffer, 2016] or IEC 61850 [International Electrotechnical Commission, 2020b] have been dominating this discussion. The literature has mentioned several distinct network-based attacks [Zhu et al., 2011], debating the intelligence-gathering process, the network reconnaissance, how to access and manipulate classified process parameters and, ultimately, how to disrupt the physical processes under control (with the inherent consequences).

Since most of the protocols are based on plain-text communications without any security enforcing mechanisms, as soon as the attacker is able to breach the security perimeter, it becomes a matter of using the right function codes and either directly connecting to field devices or hijacking Transmission Control Protocol (TCP)/Internet Protocol (IP) sessions on-the-fly. Fortunately, the industry is finally starting to move away from such completely insecure protocols. For instance, in October 2018 the Modbus Organization released a new Modbus/TCP Security protocol specification [Modbus Organization, Inc, 2018]. New Modbus conforming devices must use Transport Layer Security (TLS) 1.2 or better to achieve confidentiality and data integrity on the top of TCP sessions (thus avoiding replay and Man-in-the-middle (MitM) attacks), while the authentication and authorization of requested Modbus function codes rely on x.509v3 certificates and a combination of an *AuthZ function* and a *Roles-to-Rights Rules Database* (cf. [Modbus Organization, Inc, 2018]). Yet, both authorization and database implementation details were left outside of specification, leaving room for vendor-specific implementations.

From a standardization point-of-view, there are dozens of standards, guidelines and best practices recommendations [ENISA, 2011] [Ghosh and Sampalli, 2019]. This creates a fragmentation problem and a challenge to implement them consistently across the entire heterogeneous IACS ecosystem, from energy-related systems (e.g. Smart Grids) to Manufacturing Execution Systems (MES).

The attack-surface of IACS has also grown significantly over the past years, as a result of all the aforementioned paradigm changes. Several large incidents, from Stuxnet [Langner, 2013] to Lockergoga [Manuel and Salvio, 2019], keep showing their vulnerabilities, including the lack of security of SCADA communication protocols, the internal actors (even when unintentional) or the importance of adequate software/firmware security updates and maintenance.

Various incidents, malware examples and Advanced Persistent Threat (APT) campaigns targeting Critical Infrastructures (CI)s and IACS (summarized in Figure 2.3) are described in the literature [Anton et al., 2017] [Hemsley and Fisher, 2018] [Al-Hawawreh et al., 2019]. For instance, Win32/Industroyer, one of the major publicly-disclosed real-world malware [Cherepanov, 2017] targeting SCADA systems, provides a good example of how recent malware can use multiple protocols in a coordinated campaign. Win32/Industroyer was specifically conceived for targeting SCADA protocols and affecting electric power systems. It used a modular design to accommodate four different protocols: IEC 60870-5-101 for serial connections [International Electrotechnical Commission, 2020a]; IEC 60870-5-104 for TCP/IP connections; IEC 61850 [International Electrotechnical Commission, 2020b]; and Open Platform Communications (OPC) protocol [OPC Foundation, 2020]. Three SCADA-specific features should also be highlighted in the scope of that malware: the network device enumeration capability, by parsing device responses of different protocols; the capability of accessing field values leveraging SCADA protocols (which, again, do not enforce proper security mechanisms); and, finally, a Denial of Service (DoS) tool to send specifically-crafted malicious network packets to a specific family of Siemens devices.

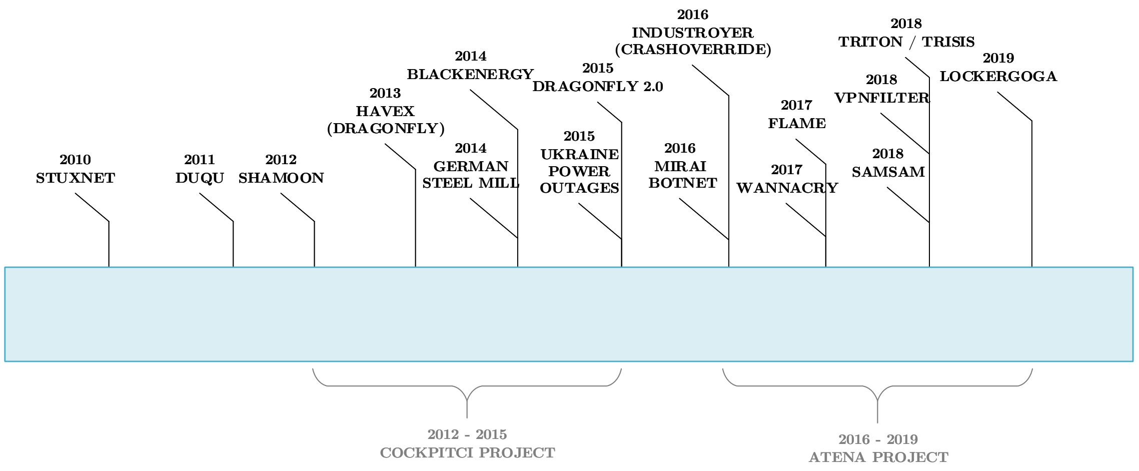


Figure 2.3: Timeline of the major Critical Infrastructures and IACS-related incidents between 2010 and 2019

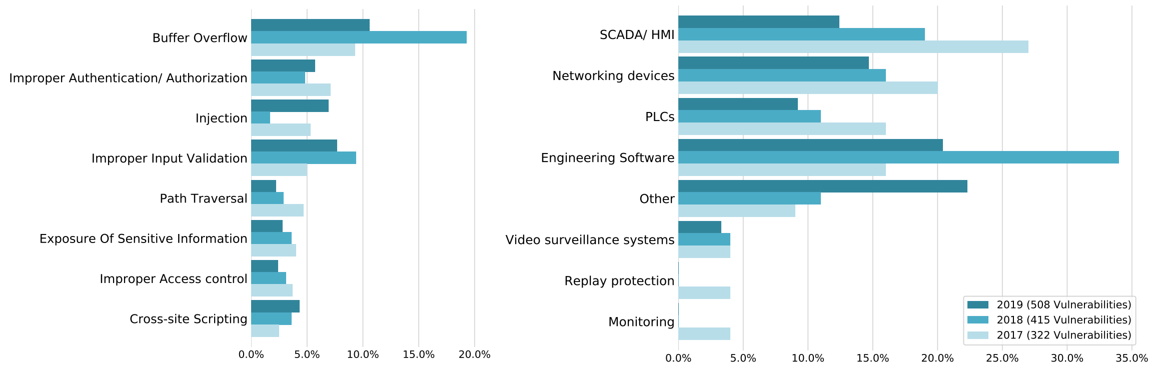


Figure 2.4: Major ICS Vulnerabilities between 2017 and 2019 by component and type (compiled from [Kasperky, 2017] [Kasperky, 2018] [Kasperky, 2019])

2.2.1 Threats and Vulnerabilities

Figure 2.4 provides a compilation of the latest disclosed vulnerabilities, per year, per component and per type, based on publicly available databases [Kasperky, 2017] [Kasperky, 2018] [Kasperky, 2019]. It not only shows increased interest in IACS security, but also exposes the naked reality of the number of vulnerabilities in different components. As expected, those numbers highlight the fact that security problems derive from more than just insecure network communication protocols. A high number of disclosed vulnerabilities refer to software or specific components, such as the HMIs. Worst, in the IACS these vulnerabilities take greater importance, since due to the update policies, assets might be exposed for longer periods of time – a device may be required to run uninterruptedly for years before being patched.

From a different perspective, Table 2.1 summarizes the outcomes of a 2019 survey to more than 300 IACS professionals [Filkins, 2019], providing a breakdown of the perceived risk and impact in OT/Control system components. According to this survey, outdated server assets present the highest risk, whereas compromised network connections were rated to have the greatest impact. This survey also highlights a rather focused IT monitoring strategy, opposed to more ambitious comprehensive OT/IT approaches. While the reason of such limited IT focus is not clear in the survey, this could be a consequence of the (lack of) OT technology maturity, complexity or even cost-benefit. It should also be noted that the notion of risk and perceived impact are tied to the complexity of the attacks and the assumption of different types of actors with different skills levels. Unfortunately, based on the major publicly reported incidents, even the most high-profile and less likely attacks are becoming more frequent and should not be disregarded.

Table 2.1: Perceived risk and impact by component, according to [Filkins, 2019]

OT/ICS Control System Capabilities	Risk	Impact
Connections to the field control networks	36.10%	34.10%
Embedded controllers or components	22.90%	33.20%
Server assets running commercial OS	57.60%	32.70%
Connections to other internal systems	42.00%	31.20%
Network devices	30.20%	30.20%
Engineering workstations	38.00%	29.30%
Operator workstations	33.20%	28.80%
Control system communication protocols	23.90%	20.50%
Process control application	16.10%	20.00%
Field devices	19.50%	19.00%
Remote access appliances	25.40%	18.50%
Physical access systems	22.40%	16.60%
Wireless communication devices and protocols	27.80%	13.20%
Plant historian	14.60%	13.20%
Mobile devices	36.10%	12.20%
Analog modems	12.20%	6.30%
Other	5.90%	2.00%

Figure 2.5 pinpoints the numerous threats that might be present within a complex and distributed Smart Grid [Suleiman et al., 2015]. As depicted in the figure, there are many types of specific devices and communications, from smart home devices and grid-specific elements (e.g. substations assets) up to the control center. Most of the mentioned threats leverage insecure communications to disrupt, intercept, change or spoof all the different process values and configuration settings.

2.2.2 Anatomy of Attacks Taking Advantage of SCADA Protocols

The ICS Cyber Kill Chain [Assante and Lee, 2015] was defined as a two-stage ICS attack model, extending the original concept of Cyber Kill Chain [Martin, 2014] – a framework for identification and prevention of cyber intrusion proposed by Lockheed. A first stage refers to the initial process of gaining access to the ICS Infrastructure and components (e.g. via internet-facing devices), and a second stage refers to when the attacker leverages the outcomes from first stage to actively explore the OT domain. Such a detailed view is useful to understand, structure and address the specific challenges of each step (cf. Figure 2.6).

One of the first steps performed during a typical IACS cyber-attack is *intelligence gathering* – collecting as much detail as possible for each asset in the target system. For networks in general, and more specifically for IACS systems, this means discovering and enumerating involved devices and collecting their specific characteristics (manufacturer, model, firmware

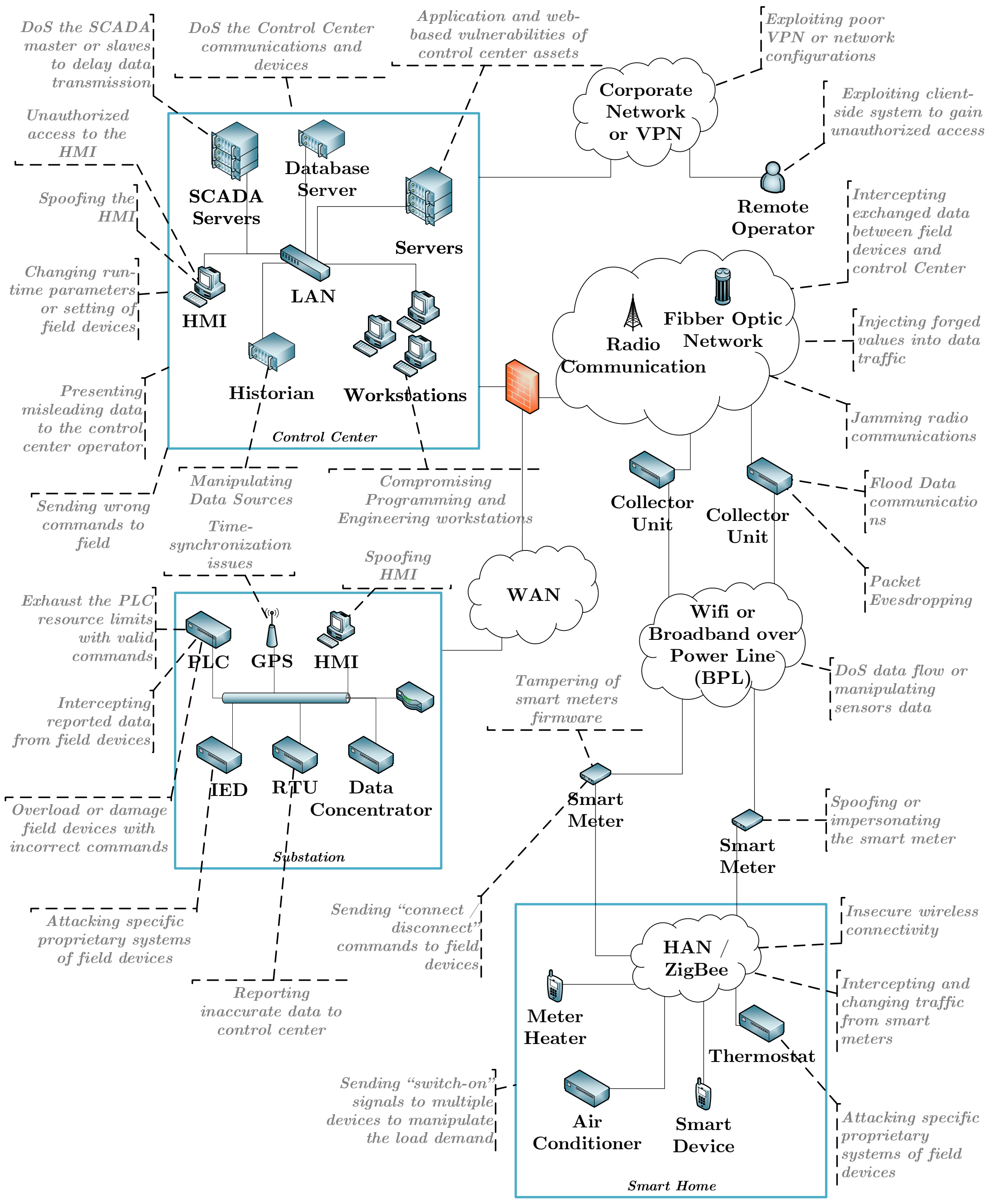

DoS the SCADA master or slaves to delay data transmission
DoS the Control Center communications and devices
Application and web-based vulnerabilities of control center assets
Exploiting poor VPN or network configurations
Corporate Network or VPN
Exploiting client-side system to gain unauthorized access
Unauthorized access to the HMI
SCADA Servers
Database Server
Servers
Remote Operator
Spoofing the HMI
LAN
HMI
Changing run-time parameters or setting of field devices
Intercepting exchanged data between field devices and control Center
Radio Communication
Fibber Optic Network
Injecting forged values into data traffic
Historian
Workstations
Presenting misleading data to the control center operator
Control Center
Jamming radio communications
Collector Unit
Collector Unit
Flood Data communicatio ns
Sending wrong commands to field
Manipulating Data Sources
Compromising Programming and Engineering workstations
Time-synchronization issues
Spoofing HMI
WAN
Packet Evesdropping
Exhaust the PLC resource limits with valid commands
PLC
GPS
HMI
Wifi or Broadband over Power Line (BPL)
DoS data flow or manipulating sensors data
Intercepting reported data from field devices
Tampering of smart meters firmware
IED
RTU
Data Concentrator
Smart Meter
Overload or damage field devices with incorrect commands
Spoofing or impersonating the smart meter
Smart Meter
Substation
Attacking specific proprietary systems of field devices
Sending "connect / disconnect" commands to field devices
HAN / ZigBee
Insecure wireless connectivity
Reporting inaccurate data to control center
Meter Heater
Intercepting and changing traffic from smart meters
Thermostat
Sending "switch-on" signals to multiple devices to manipulate the load demand
Air Conditioner
Smart Device
Attacking specific proprietary systems of field devices
Smart Home

Figure 2.5: The many threats of a Smart Grid (adapted from [Suleiman et al., 2015])

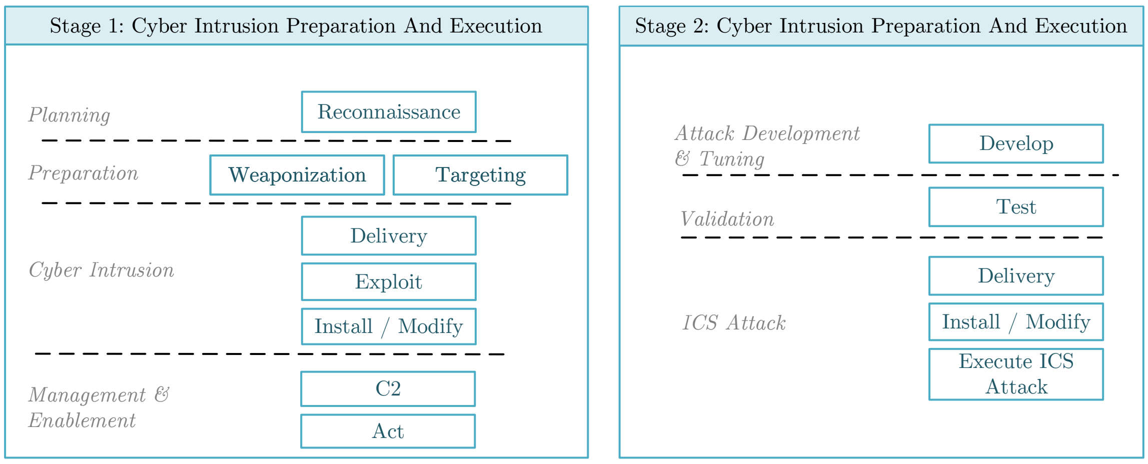


Figure 2.6: SANS Kill Chain ICS Attack stages (adapted from [Assante and Lee, 2015])

version, etc.), to look for known vulnerabilities. A classic network scan is useful to find responding IPs and open ports, but collecting details about PLC models and versions requires additional investigation. In SCADA systems, a PLC might be configured to bridge serial segments which may hide additional PLCs that will not be disclosed, for instance, by common TCP SYN Scans.

Each SCADA protocol typically requires a different approach. In [Rudakov, 2010], a Nmap NSE script is used to identify and enumerate Modbus devices, including Modbus slaves (please note that multiple Modbus slaves may be behind a single IP address). For the EtherNet/IP protocol, another Nmap NSE script, from [Hilt, 2014a], explores the lack of authentication and, by sending a Request Identification packet, is capable of retrieving multiple information from a device, such as the model, firmware, OS and hardware versions and serial numbers. Other examples, such as those found in [Deneut, 2017] and [Hilt, 2014b], may be used to find and identify specific details for various Siemens models.

After the enumeration phase, since many SCADA protocols still use unencrypted TCP connections, it becomes possible to hijack TCP sessions or to simply establish new connections to PLCs, in order to *access or modify sensitive data.* For instance, a metasploit module available in [EsMnemon et al., 2018] allows reading and writing different types of registers using standard Modbus functions.

In addition to reading and writing registers, a PLC may sometimes be remotely shut down, either by sending a valid command (from a malicious actor) or by exploring vulnerabilities in the PLC input validation. A metasploit module for Modbus [Wightman, 2018b], for instance, allows remotely starting and stopping a PLC using Modbus requests. For Ethernet/IP CIP, there are also several modules [Santamarta et al., 2012] [Monteiro et al., 2019] that explore application layer issues in the packet handling, eventually leading to *Denial-of-Service (DoS)* conditions.

Finally, reprogramming the entire logic of a PLC is also possible, as demonstrated by [Wightman, 2018a], a metasploit module that allows downloading and uploading ladder logic code from/to Schneider Modicon PLCs.

Based on the observed pieces of evidence and publicly disclosed information, Figure 2.7 shows the main steps of the real-world Industroyer attack mapped into the ICS Cyber Kill Chain and IACS levels. Again, this stresses the notion that a complete IACS attack or campaign is more than just exploiting a SCADA protocol. Most of the time, other vulnerabilities, not necessarily SCADA-specific (e.g. a phishing campaign or a compromised supply-chain), are the pivot for accessing the SCADA components. Similarly, protocols such as Domain Name System (DNS) or HTTP Secure (HTTPS) are commonly used to exfiltrate sensitive data or to *Command & Control (C2)* a compromised device. On the other side (often overlooked), several incidents described in the literature as targeting SCADA environments did not end up reaching low level controls, but only the IT network – nevertheless, a breach in the IT or higher OT network levels, for example, in a large electrical system company, is still a significant event.

The variety of different SCADA communication protocols demands a huge effort, not just to somehow accommodate them all, but also to design unified solutions able to fit such an heterogeneous SCADA ecosystem. For instance, PCOM – a SCADA protocol that enables applications to communicate with devices – suffers from the same security issues previously discussed. Nevertheless, perhaps due to its relatively smaller market penetration, when compared with the couple of more popular protocols, it is not widely discussed in available literature. Since in this dissertation PCOM is extensively used to discuss a set of scenarios and tools developed as part of the SCADA exploratory analysis (cf. Chapter 4), we introduce it in Section 4.2.1.

2.2.3 Defense Models and Mitigation Strategies

The Defense-in-Depth (DiD) strategies from the past are no longer enough to address the new physical and logical boundaries of IACS. The attack threats are highly heterogeneous and complex, which demands more collaborative and orchestrated anomaly detection techniques.

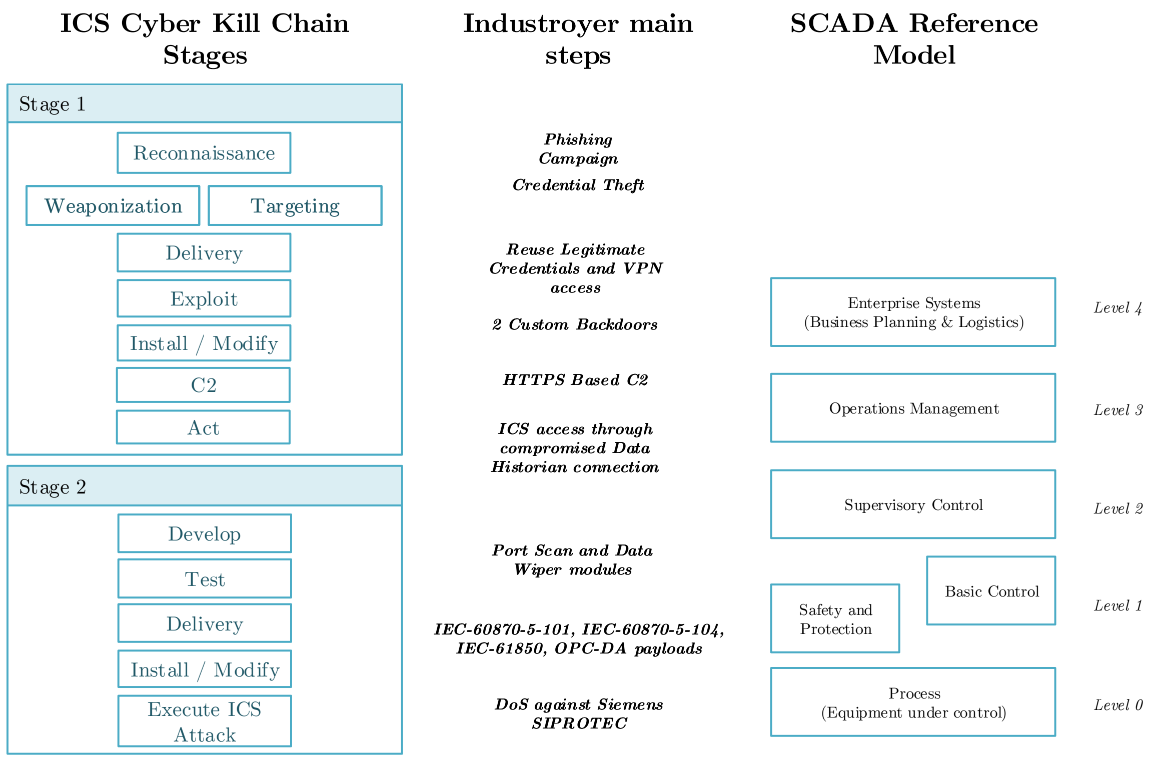


Figure 2.7: Mapping of Industroyer attack steps with ICS Cyber Kill Chain and SCADA levels (based on [SANS, 2017] [Cherepanov, 2017] [Slowik, 2018]).

Recognizing those specificities, as well as the tremendous impact they can have on SCADA-based IACS, there is a strong investment in research towards enhancing the security of (both legacy and more recent) SCADA systems.

Network Intrusion Detection System (NIDS) are among the most effective tools when it comes to dealing with the security issues of legacy or insecure protocols (as Modbus or PCOM). The role of a NIDS is to distinguish legitimate traffic patterns from malicious ones, offering a complementary approach to traditional firewalls.

Firewalls are helpful to shield the trusted perimeter and block the traffic based on TCP/IP header fields, such as an IP address or a TCP port, mostly playing a preventive role. Nevertheless, a fine control based on protocol features (e.g. determine which SCADA functions should be blocked and which ones are allowed) would require Deep Packet Inspection (DPI) and custom SCADA parsing. However, even when supporting DPI, classic firewalls often fall short when it comes to offering comprehensive protection against more sophisticated attacks. For instance, because protocols such as Modbus and PCOM are vulnerable to spoofing attacks, the single reliance on classic firewalls for protection is insufficient, requiring complementary solutions (e.g. ARP monitoring solutions such as Snort's ARP preprocessor, to detect ARP poisoning attacks [Diogo, 2018]). Moreover, because of the specific nature of the SCADA ecosystem, where availability has been traditionally favored over confidentially and integrity, the decision to specify traffic blocking or throttling reactions for specific rules is often questioned, due to the potentially negative impact in terms of operational and safety levels, often leading instead to simple event logging.

A NIDS such as Snort might be used to detect and report unwanted communications, but to be able to effectively block them it must be deployed in the middle of the communications path, providing intrusion prevention capabilities – this is referred to as the *inline mode* in Snort documentation. The inline mode has its own fair share of issues, which makes it unpopular among IACS operators – besides introducing a single point of failure in the communications path (by placing the Snort host in a traffic mediation point), there is also the possibility that a knowledgeable attacker may try to abuse it to deliberately drop legitimate traffic. On the other hand, if the NIDS deployment is only passive (offering pure detection capabilities, by just reporting alarms), specific attacks that only require a single SCADA packet to go through (like the remote shutdown of a PLC) might be successful, despite being detected and reported. In the end, it all comes down to a trade-off between prioritizing availability (passive mode) or having effective reaction capabilities (inline mode).

Besides the traditional strategies recommending the introduction of segregated network domains with disallow-by-default traffic control policies, or the use of bump-in-the-wire Virtual Private Networks to provide encryption with role-based access-control, there are other alternatives for which this research may also be relevant, namely the implementation of SCADA-aware data diodes and honeypots.

Data diodes [Borges de Freitas et al., 2018], also known as unidirectional gateways, allow to enforce strict one-way communication between different components. For instance, they allow to limit the network traffic from a restricted domain to a less secure network segment (but not in the opposite direction). Such gateways require the use of additional software components in each side of the unidirectional link in order to support the conversion of TCP/IP SCADA protocol requests into unidirectional data streams – since protocols such as Modbus/TCP or DNP3 were conceived for bidirectional operation, relying on a three-way handshake and continuous acknowledgments between peers.

SCADA honeypots such as conpot [Rist et al., 2013] or the one proposed by [Simões et al., 2015] are used to replicate the behavior of a given protocol or asset. While such a honeypot would hardly constitute an attack deterrent, it could play a relevant role in disclosing attackers at early phases, providing a means not only to detect them but also to profile their strategy.

Based on a collaborative effort involving industry, academia, and government, the National Institute of Standards and Technology (NIST) proposed a reference framework for improving critical infrastructure cyber-security (Figure 2.8). This framework includes a description of core security-related functions (i.e. Identify, Protect, Detect, Respond, and Recover) and maps to additional informative references. With particular relevance to this work (and sharing some of the goals of this research), the *detection function* (Table 2.2) details several recommended activities, such as continuous monitoring, event collection and aggregation from different data sources (e.g. network, connections, devices, personnel and software).

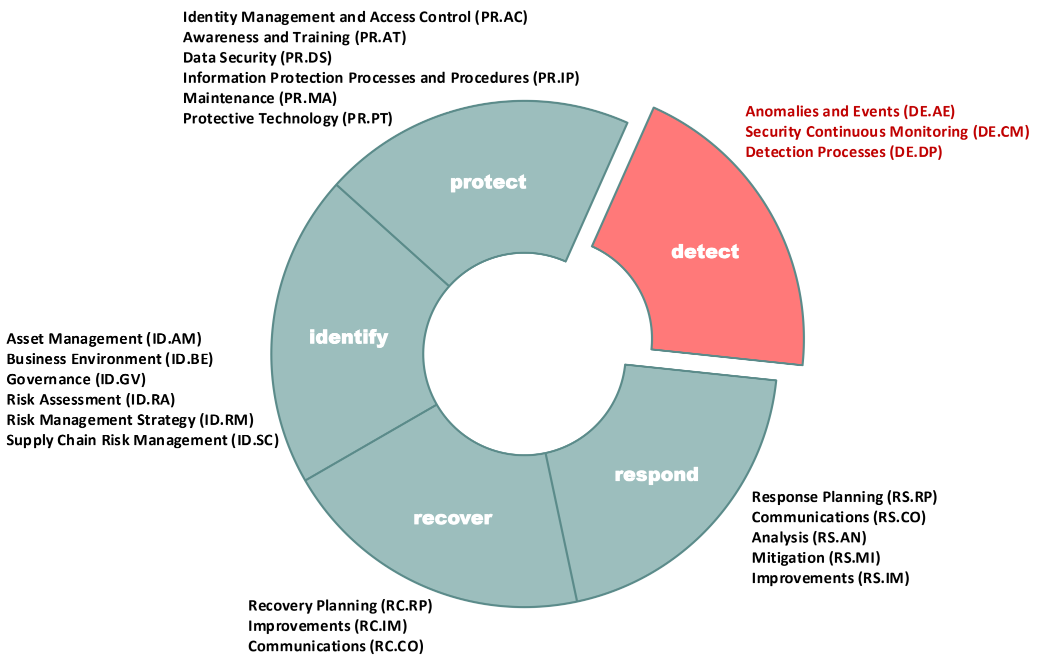

Identity Management and Access Control (PR.AC)
Awareness and Training (PR.AT)
Data Security (PR.DS)
Information Protection Processes and Procedures (PR.IP)
Maintenance (PR.MA)
Protective Technology (PR.PT)
Anomalies and Events (DE.AE)
Security Continuous Monitoring (DE.CM)
Detection Processes (DE.DP)
protect
detect
identify
respond
recover
Asset Management (ID.AM)
Business Environment (ID.BE)
Governance (ID.GV)
Risk Assessment (ID.RA)
Risk Management Strategy (ID.RM)
Supply Chain Risk Management (ID.SC)
Response Planning (RS.RP)
Communications (RS.CO)
Analysis (RS.AN)
Mitigation (RS.MI)
Improvements (RS.IM)
Recovery Planning (RC.RP)
Improvements (RC.IM)
Communications (RC.CO)

Figure 2.8: NIST Critical Infrastructure Cybersecurity Framework Version 1.1 (adapted from [NIST, 2018])

AI-enhanced security mechanisms are increasingly playing a key role in the cyber-security of both IT/OT environments, being used to support different types of analytics. A survey to more than 800 senior executives from IT Information Security, Cyber-security and IT Operations, covering the IT, Internet of Things (IoT) and OT domains [Capgemini Research Institute, 2019b], highlights the importance and the need of AI capabilities in cyber-security

Table 2.2: Overview of *Detect* function of NIST Cybersecurity Framework.

Category	Sub-Category
Anomalies and Events (DE.AE): Anomalous activity is detected	DE.AE-1: A baseline of network operations and expected data flows for users and systems is established and
	DE.AE-2: Detected events are analyzed to understand attack targets and methods
	DE.AE-3: Event data are collected and correlated from multiple sources and
	DE.AE-4: Impact of events is determined
	DE.AE-5: Incident alert thresholds are established
Monitoring (DE.CM): The information system and assets are monitored to identify cybersecurity events and verify	DE.CM-1: The network is monitored to detect potential cybersecurity events
	DE.CM-2: The physical environment is monitored to detect potential cyber security
	DE.CM-3: Personnel activity is monitored to detect potential cybersecurity events
	DE.CM-4: Malicious code is detected
	DE.CM-S: Unauthorized mobile code is detected
	DE.CM-6: External service provider activity is monitored to detect potential cybersecurity events
	DE.CM-7: Monitoring for unauthorized personnel, connections, devices, and software is performed
	DE.CM-7: Vulnerability Scans are performed
Detection Processes (DE.DP): Detection processes and procedures are maintained and tested to ensure awareness of anomalous events.	DE.DP-1: Roles and responsibilities for detection are well defined to ensure accountability
	DE.DP-2: Detection activities comply with all applicable requirements
	DE.DP-3: Detection processes are tested
	DE.DP-4: Event detection information is communicated
	DE.DP-5: Detection processes are continuously improved

– with more than half of the respondents saying they make extensive use of AI for cyber-detection. The importance of correctly identifying data sources and creating data platforms to operationalize AI as one of the first steps in the road-map to implement AI in cyber-security is also one of the key points from this survey. This is aligned with the strategy of supporting multiple anomaly detection techniques based on various ML algorithms, pursued in this thesis.

Since the aforementioned attack steps require access to the SCADA process control network, it makes sense to consider the protection of the communications infrastructure as part of a mitigation strategy. One of the possible solutions for detecting, alerting or blocking such attacks is to use NIDS or firewalls. Nevertheless, in order to have a perspective of what is going on in the network, such solutions must be able to perform Deep Packet Inspection (DPI). For instance, read operations might be allowed but write and administrative operations might be blocked and reported. Repositories such as [Hilt, 2015] contain collections of Snort rules for a few SCADA protocols (Modbus, DNP3 and S7, among others) that might be used for detecting several types of operations, including network scans and Modicon PLC reprogramming attempts.

For more complex protocols or interactions, it is recommended to have specific preprocessors able to decode specific protocol values that may happen to be not always at the same offset. For instance, Jordan [Jordan, 2012] details how specific preprocessors for Modbus and DNP3 protocols allow creating syntax-richer rules by providing an extended list of keywords. Furthermore, a set of Ethernet/IP rules to be used with Suricata Intrusion Detection System (IDS) [Open Information Security Foundation, 2020] is described in [Digital Bond, Inc. and N-Dimension Solutions, Solana Networks, 2011].

Another key tool in this quest for secure IACS is the availability of good quality datasets of realistic SCADA traffic. These datasets are useful both for training AI-based systems and for assessing the effectiveness of security tools in general. Table 2.3 provides a non-exhaustive list of the most relevant publicly available SCADA datasets. This table shows that, despite a few datasets that appeared in recent years, there is still a lack of more and better datasets, covering different scenarios, protocols and specific equipment. The majority of existing datasets refer to small to medium testbeds or initiatives, which might be explained by the complexity, time and monetary resources needed to reassemble a larger scale testbed. Richer and more realistic environments containing more use cases would definitely help the community researching for better detection mechanisms.

Table 2.3: Publicly available IACS and SCADA datasets

Description	Reference
Datasets from various tesbeds of power systems, gas pipelines and water storage tanks	[Morris and Gao, 2014]
Miscellaneous ICS lab with s, RTUs, servers, industrial network equipment (switches, firewalls, etc) (4SICS)	[NETRESEC, 2015]
Scaled-down testbed of a water treatment plant (SWaT)	[Goh et al., 2016]
Water distribution tesbed (WADI)	[Ahmed et al., 2017]
Power grid testbed including Generation, Transmission, Micro-grid, and Smart Home (EPIC)	[Adepu et al., 2018]
Water storage tank's control emulated testbed	[Teixeira et al., 2018]
A real-world, medium-sized water distribution system (BATADAL)	[Taormina et al., 2018]
Small simulated electric network testbed	[Lemay and Fernandez, 2016]
Miscellaneous SCADA network packet captures	[NETRESEC, 2020]
Collection of different SCADA network packet captures	[Hink, 2020]

2.3 Intrusion and Anomaly Detection Techniques for IACS

Intrusion and anomaly detection is one of the key capabilities in the cyber-security monitoring field. Intrusion detection is traditionally defined as the process of monitoring a given system to detect potential malicious behaviors, whereas anomaly detection can be defined as the process of searching for deviant patterns over complex datasets – often relying on AI and ML algorithms. In the last years, both have attracted considerable attention. In the IACS domain, such techniques make it possible to spot deviant patterns and identify anomalies (e.g. created by cyber-attacks or physical process anomalies) based on the processing of large amounts of data.

Two main approaches can be found in the literature: signature-based Network Intrusion Detection Systems focused on mainstream SCADA communication protocols (e.g. Modbus, DNP3, CIP) and anomaly detection based on machine learning algorithms.

Signature-based approaches are able to detect abnormal communications patterns based on known packet signatures. They are often not aware of the physical process and are likely to fail against unknown vulnerabilities. On the other hand, anomaly detection based on supervised ML models require previous training and are usually tuned for a single scenario, either for a specific process or for a single communication protocol. Finally, unsupervised approaches are typically less accurate, which might lead to huge amount of false positives that overwhelm the operator.

As already mentioned, multiple open-source NIDS, like Snort, Suricata or Bro, have been the focus of several attempts to improve their support for SCADA protocols, either by dedicated preprocessors or by specific rules [Wong et al., 2017] [Udd et al., 2016] [Vávra and Hromada, 2016] [Lin et al., 2013] [Rosa et al., 2020]. Such signature-based approaches represent a simple but somehow limited solution for detecting and enforcing communication policies at the network level.

The remaining of this section presents a literature review focused on anomaly-based detection techniques for IACS proposed in the literature in the last years, thus complementing earlier surveys in the topics of SCADA-anomaly detection [Ding et al., 2018] [Nazir et al., 2017] [Iturbe et al., 2017] and cyber-security scenarios [Handa et al., 2019] [Nisioti et al., 2018] [Terzi et al., 2017] [Dewa and Maglaras, 2016] [Agrawal and Agrawal, 2015].

Phillips et al. [Phillips et al., 2020] presented four different anomaly detection approaches – Support Vector Machine (SVM), Decision trees, K-nearest Neighbor (KNN), and k-means – to classify SCADA network traffic, having obtained better results with the supervised methods. Similarly to others, their evaluation was based on a public dataset from a gas pipeline [Morris and Gao, 2014]. Nevertheless, no details were provided regarding the algorithms implementations nor how such approach could be deployed on real systems to detect attacks on real-time.

McKinnon et al. [McKinnon et al., 2020] provided a comparison of three algorithms – One Class Support Vector Machine (OCSVM), Isolation Forests (IF) and Elliptical Envelope (EE) – for wind turbine fault diagnosis, based on real historical turbines data in Europe. They have obtained the best results (82% of accuracy) with the first two methods. Nevertheless, again, no details were provided regarding the algorithm implementations or the used tools. Moreover, no dataset was available to reproduce the results.

Gao et al. [Gao et al., 2019] presented an ensemble approach of Feedforward Neural Network (FNN) and Long short-term memory (LSTM), having obtained better results to detect temporally uncorrelated attacks with FNN and temporally correlated attacks with LSTM. Among the list of used features, the authors included several Modbus-specific fields, thus limiting their approach to scenarios based on this protocol. Moreover, the authors conducted their experiments within a simulated SCADA environment and no datasets are available.

Similarly to [Gao et al., 2019], in [Yang et al., 2019] the features are extracted from network packets into a 25-tuple for DNP3 [DNP Users Group, 2005] communications. The authors used a Convolutional Neural Network (CNN) to classify different types of network-based anomalies into several classes. This has the advantage of not only reporting the anomalies but also classifying them into more specific type of attacks. Even though some of the classes were misclassified, they obtained an overall accuracy of 99.38% and a low number of false positives related to the normal class. The authors assessed their model against two different testbeds, but none of them is available.

A comparison between SVM and Random Forests based on two datasets – Modbus and OPC UA [OPC Foundation, 2020] communications respectively – is presented in [Anton et al., 2019]. The authors identified the most relevant features in the first dataset as a combination of process specific values (e.g. pressure) and packet related features (e.g. packet length). Random forests consistently outperformed the SVM approach. Similarly to most of the SCADA datasets, they started with an unbalanced dataset (i.e. the number of normal values is significantly superior to the anomalies) and a large percentage of missing values. To overcome this, the authors used Principal Component Analysis (PCA) as a preprocessing means for Random Forests and a zero mean scaling for the SVM case.

The performance of two Recurrent Neural Network (RNN) architectures (Gated Recurrent

Unit (GRU) and LSTM) is analysed in [Sokolov et al., 2019], arguing their benefits when predicting unseen anomalies, compared to traditional supervised classifiers more suited to detect known anomalies. Similarly to other research works, the authors used a public gas pipeline dataset [Morris and Gao, 2014] and a combination of packet and process features. While the authors suggest there is room for improvement regarding the obtained results, the accuracy around 90% for both methods is smaller than other classification approaches. Although more computing-intensive, they obtained slightly better results as the number of epochs increases using LSTM, suggesting it is more suitable than GRU for larger datasets and unlimited computation resources.

Another study [Ramotsoela et al., 2019], follows a different approach, adopting an ensemble technique using a Quadratic Discriminant Analysis (QDA) to combine two density-based estimation algorithms (Local Outlier Factor (LOF) and Subspace Outlier Degree (SOD)). Such combination avoids the traditional classification path and does not make an assumption about the distribution of the data (as opposed to other parametric methods). Combining both methods, their approach was able to cope with both local and global outlier detection, leading to an overall good performance without penalty for high dimensional data. Nevertheless, this is a computationally expensive method and might not be as accurate as other parametric approaches if the data distribution is known. The authors run their model against the BATADAL dataset [Taormina et al., 2018], allowing to perform comparisons with publicly available results using the same data. The ensemble approach does not always outperform their counterparts and, therefore, one cannot conclude it would always be a better option. Nevertheless, it would be interesting to see how such a technique performs with other datasets and IACS domains.

In another study [Khan et al., 2019], the authors used a two-level approach by first applying a Blooming Filter and, afterwards, a KNN classifier for network anomaly detection. Each packet needs to pass both algorithms to be considered as normal. They used a public SCADA dataset [Morris and Gao, 2014] referring to Modbus communications within a gas pipeline facility. The authors handle the unbalanced dataset problem by under-sampling using an AllKNN algorithm. The authors also highlight the importance of a feature preprocessing step, having used three different algorithms for that purpose: PCA, Canonical Correlation Analysis (CCA), and Independent Component Analysis (ICA). Nevertheless, and despite their interesting 97% of accuracy, they obtained significantly different results (from 68% to 100%) depending on the class of anomalies. Moreover, since they depart from Modbus-related features, it would be interesting to understand how they perform on different protocols and feature spaces.

Another interesting approach with 98% of accuracy combined Online Sequential Extreme Learning Machine (OS-ELM) with a set of Restricted Boltzmann machine (RBM) to classify network flows into several classes of attacks [Demertzis et al., 2019]. It also used the public dataset of a Gas pipeline [Morris and Gao, 2014] but does not specify whether the entire feature space is used or not, only mentioning that the data is grouped into sliding windows of 100 samples with overlapping of 400 instances. OS-ELM is used as a first step to classify the flow as normal or in one of the known classes. Whenever an anomaly is detected, it is forwarded to one of the RBMs. Then, each RBM, trained for a single class of anomalies using a unary classification method, is responsible for deciding whether the anomaly matches its class or not. If there is no match it will sequentially test the remaining RBMs. Such extra step might be valuable to improve the correct class classification since its RBMs can be highly specialized. Nevertheless, it is important to acknowledge that the OS-ELMs are susceptible to produce false negatives. In that case, the flow is marked as normal and never reaches any of the RBMs. Moreover, this approach depends on the previous knowledge of each class and might not be suitable for unknown types of attacks.

A three-stacked LSTM approach to predict anomalies in time series windows against different types of datasets, including one power demand dataset, is proposed in [Nguyen et al., 2018]. Despite claiming 92% of precision, no details are provided about the used features or the used datasets (which are not publicly available).

An approach to predict cyber-attacks by analysing different combinations of CNNs is presented in [Kravchik and Shabtai, 2018]. It successfully detected 32 out of 36 attacks in a public industrial water treatment dataset [Goh et al., 2016], processing physical process parameters in time windows of 200 seconds. Nevertheless, using only physical process parameters means this approach will not detect layer 2/3 attacks without direct impact on the physical process (e.g. network scans).

Intrusion Weighted Particle based Cuckoo Search Optimization (IWP-CSP) and Hierarchical Neuron Neuron Architecture based Neural Network (HNA-NN) are used to classify SCADA network data into nine different classes, in [Shitharth and Prince Winston, 2017]. The authors mention an experimental evaluation using a simulated environment, with 100 nodes and two process level features (humidity and temperature). However, there is no information on how such a small number of features are related with their feature optimization layer, how they relate with all the categories of attacks, or whether an additional set of features were used but not referred.

The authors of [Keliris et al., 2017] present an anomaly detection solution capable of detecting attacks against managed physical processes based on: 1) a first stage using a SVM model to detect whether there is an anomaly and 2) and a second set of SVMs specifically trained to classify the anomaly into a known category. They evaluate their model against a Human in the Loop (HITL) testbed, using a hybrid combination of a simulated process (Tennessee Eastman chemical process) and a real (Wago 750-881). Time windows are used to minimize data noise and to reduce the number of false positives. The model was able to detect the two injected payloads. Nevertheless, it requires training and it will likely fail to detect other categories of attacks that do not interfere with process values.

Table 2.4 presents a summary of the surveyed literature. Since each research work might contain multiple experiment variants, the presented performance indicators refer to the best results (scenarios) found for each work. Similarly, since multiple distinct datasets are often used during the evaluation step, the presented results prioritize SCADA-based validations. As already mentioned, additional surveys in the topic of SCADA-anomaly detection, covering earlier works, can be found in [Ding et al., 2018] [Nazir et al., 2017] [Iturbe et al., 2017]. Additional anomaly detection literature focused on more general cyber-security scenarios can be found in [Handa et al., 2019] [Nisioti et al., 2018] [Terzi et al., 2017] [Dewa and Maglaras, 2016] [Agrawal and Agrawal, 2015].

Table 2.4: Summary of the latest literature contributions of SCADA Anomaly Detection Algorithms

Focus	Method	Dataset	Features	Anomalies	Indicators	Reference
Gas Pipeline	SVM, Decision trees, KNN, and k-means	Public Dataset	Process and network related	L7 Attacks	ACC=99.99	[Phillips et al., 2020]
Wind Turbine Systems	OCSVM, IF and EE	Custom Dataset	19 Turbine features	Historical turbine faults	ACC=82	[McKinnon et al., 2020]
Industrial Control network data (Modbus related)	FNN and LSTM	Custom Modbus Related	19 Network Packet Features	Layer 2/3/7 attacks	P=99.76 R=99.57 F1=99.68	[Gao et al., 2019]
Industrial Control network data (DNP3 related)	CNN	2 private testbeds	25 Packet Based	Layer 2/3/7 attacks	ACC=99.38	[Yang et al., 2019]

Precision (P), Recall (R), F-Score (F), Accuracy (ACC), Not Available (N.A.)

Table 2.4: Summary of the latest literature contributions of SCADA Anomaly Detection Algorithms (continued)

Focus	Method	Dataset	Features	Anomalies	Indicators	Reference
Industrial Control network data	AMPSO-SVM-K-means++ / GSA-AFSA-ELM	N.A.	4 Network packet related Features	N.A.	DR=95 FA=0.02	[Chen et al., 2019]
Gas pipeline (Modbus and OPC-UA related)	SVM and Random Forests	2 Public data sets	Network packet and application based	Layer 2/3/7 attacks	ACC=99.98	[Anton et al., 2019]
Gas Pipeline	RNN (LSTM and GRU)	Public Dataset	N.A.	35 application level attacks	P=0.92 R=0.92 ACC=0.92	[Sokolov et al., 2019]
Electric Power Systems	CNN	Simulated IEEE-Bus systems	N.A.	Power related faults and Data injection	ACC=98.67	[Basumallik et al., 2019]
Water distribution	SOD / LOF / QDA	BATADAL	N.A.	Application Level Anomalies and Replay attacks	P=0.88 R=0.94 F=0.91	[Ramotsoela et al., 2019]
Gas Pipeline	Bloom Filter and KNN	Public Dataset	20 Network Packet Features	Layer 3/7 Attacks	ACC=0.97 P=0.98 R=0.92 F=0.95	[Khan et al., 2019]
Gas Pipeline	OS-ELM and RBM	Private dataset	26 attributes	Layer 3/7 attacks	P=0.99 R=0.99 F=0.99	[Demertzis et al., 2019]
Water treatment system	CNN and LSTM	SWaT	51 attributes	36 application level attacks	P=1 R=0.85 F1=0.92	[Kravchik and Shabtai, 2018]
Power Demand	LSTM	N.A.	N.A.	Power Demand faults	P=0.92 R=0.14 F1=0.87	[Nguyen et al., 2018]
Electric Power Systems	SVM / ANN	Transmission & Distribution datasets	5 PMU correlation Features	Replay Attacks	ACC=98.47 P=99.54 F1=0.92	[Jiang et al., 2017]

Precision (P), Recall (R), F-Score (F), Accuracy (ACC), Not Available (N.A.)

Table 2.4: Summary of the latest literature contributions of SCADA Anomaly Detection Algorithms (continued)

Focus	Method	Dataset	Features	Anomalies	Indicators	Reference
Sensors Data	HNA-NN	100 nodes simulated in NS-2	2 Features (humidity and temperature)	DoS and Spoofing attacks	P=72 R=100 ACC=95	[Shitharth and Prince Winston, 2017]
TE Chemical Process	SVM	Simulated TE process	12 sensors measurements	Ladder Logic Injection	Graphic only	[Keliris et al., 2017]

Precision (P), Recall (R), F-Score (F), Accuracy (ACC), Not Available (N.A.)

2.4 Event Processing and Correlation

The previous section overviews intrusion and anomaly detection techniques for IACS. However, it is not enough to look at the algorithms as isolated components, since it is necessary to have a broader framework for efficiently handling large amounts of events collected from multiple sources and different heterogeneous domains, able to feed multiple detection algorithms.

This section is focused on event processing and correlation frameworks. First, classic event correlation tools will be introduced and analysed. Next, based on the conclusion that those classic tools are no longer able to handle the requirements of modern IACS, event processing architectures originated in the Big Data domain will be discussed.

The term *event* is used within this thesis to refer to a generic occurrence collected within the system (i.e. IACS infrastructure). Thus, it can represent almost any kind of information (e.g a particular security incident alert, log messages, network trace data, field process values, a group of other events). This way, event correlation can be generically defined as a way of analyzing events to identify potential relationships. Rule-based correlation employs a set of rules to represent all the event processing logic. Each event, which might undergo a previous normalization stage, is compared against a ruleset and, in case there is a match, a predefined action is triggered (such as an alarm or the activation of an automatic reaction).

2.4.1 Classic Event Correlation Tools

This subsection introduces five rule-based event correlations tools (Esper, SEC, Drools, Nodebrain and Prelude), selected based on our empirical experience and available literature. This analysis was undertaken in an early stage of this thesis work, in the scope of the design of the CockpitCI detection layer. A more detailed comparison work, including performance tests, can be found in [Rosa et al., 2015].

Esper is a correlation framework capable of analyzing streams of events, constituting a Complex Event Processing (CEP) engine [EsperTech Inc., 2014]. There is an open-source version with a limited number of components and two full-featured commercial versions: Esper Enterprise Edition and EsperHA. In the open-source category, two distinct distributions are available: a Java version, evaluated in the scope of this study, and a .NET based version designated as Nesper. The Java-based variant is available as a set of Java packages that can be integrated into a full solution using their Java APIs.

In the literature, there are references to the usage of Esper to detect inter-domain stealthy port scans [Aniello et al., 2011], analyzing the establishment of TCP connections and applying a rank-based algorithm to classify the scans. Another usage of Esper [Dunkel et al., 2011] refers to event processing of road traffic information produced by a group of sensors (such as the average speed, occupancy or number of vehicles) to support decisions on the traffic management system. Esper uses a Structured Query Language (SQL)-like approach for rule description, designated as Event Processing Language (EPL), that provides pattern-matching mechanisms via state machines. Esper loads the EPL query set and performs matching within the incoming stream of events, implicitly incorporating a time-based correlation logic that accounts for ordering and sequencing, within a time-window. Events can be represented in Java structures like Plain Old Java Objects (POJO) or in a more abstract way using Extensible Markup Language (XML). The insertion and handling operations need to be developed on top of the Esper library and input/output adapter APIs.

Simple Event Correlator (SEC) is a lightweight event correlation engine [Vaarandi, 2014]. It is written in Perl and distributed under an open-source license with ports for multiple operating systems, being also easy to integrate as a component of a full-featured correlation or SIEM framework. SEC has been used for event reduction within distributed scenarios, helping to reduce the amount of information transmitted between log-producing/generating agents and the central servers [Myers et al., 2011], and also implementing different operation modes to define the level and type of information that is passed between the logger systems. SEC has also been integrated into a generic intrusion detection architecture [Ficco and Romano, 2011], where it was used to diagnose whether a previously detected attack was successful or did not lead to an intrusion.

SEC rules follow a text-based approach, with matching conditions being defined as string occurrences, regular expressions or Perl subroutines. Input data can be read from text files, named pipes, or using the standard output descriptor. Output actions can be defined as the execution of a shell command or the creation of a context to use as a matching condition in other rules (therefore allowing second-order correlation). SEC sequentially tests all the rules found inside a text configuration file but can perform parallel analysis if the rules spread over multiple files.

Drools is constituted by a suite of tools classified as a Business Rules Management System (BRMS) [Red-hat, 2014]. The correlation rule engine-referred in this survey is part of the complete suite, being designated as Drools Fusion (simply referred to as Drools in this document). It is deployed as a set of Java libraries that can run on multiple operating system environments.

There is a documented use case explaining how to integrate Drools in a platform to filter and aggregate Radio-frequency identification (RFID) information in a "smart" hospital environment [Yao et al., 2011], using RFID sensors distributed all over the hospital to collect information about doctors, patients and objects, which is correlated to detect time-critical emergencies, based on a set of policies. Another literature reference to Drools describes the use of its complex event processing capabilities to track and trace an object on a large-scale environment, once again using RFID information from a large number of customers spread along different physical locations to trace and follow logistics operations [Wu et al., 2012].

Drools has two operation modes: cloud mode, without a time notion; and stream mode, where events are processed using time notion as they are inserted in the correlation engine. Events are defined as Java classes. Rules are defined in a custom expression-based format that may contain Java code to express the actions to perform. For rule processing, Drools uses an adapted version of the Rete [Liu et al., 2010] algorithm, which was designed to sacrifice memory for increased speed.

Nodebrain is described by its authors as a lightweight event monitoring agent [NodeBrain.org, 2014]. It has the capability of applying a set of rules to a stream of events in order to correlate them. Nodebrain is available under an open-source license, also including a set of scripts and a C-based API designed to extend its features.

A documented use case describes the use of Nodebrain to control the policy rules of a military ad-hoc network in a lightweight management system [Jormakka et al., 2007]. To configure a remote network element, each element can trigger an action/command in another one, based on a set of policies (Nodebrain rules). Nodebrain events are based on text inputs, while the rules use a custom declarative format. It supports several operation modes, including: (i) a daemon mode to continuously execute in the background; (ii) a batch mode to process and terminate execution; and (iii) an interactive mode for debugging purposes. Input and output options vary from simple named pipes, secure TCP/IP communications between multiple nodes or custom and specific modules like remote Syslog communication. Custom shell commands can be defined as actions.

The **Prelude** tool provides additional features besides the rule engine component [Lang, 2014]. Two versions are available: a full-featured commercial version and a more limited open-source version, based on Python and with substantially lower performance and less features.

There are several use cases for Prelude, mostly related to its use in event normalization and correlation for intrusion detection. For instance, [Ficco et al., 2013] proposes a Prelude-based distributed architecture for Cloud Computing with multiple instances of correlation engines collecting information at several levels, together with the use of Bayesian Networks to detect sequences of unauthorized activities. Another scenario [Petri et al., 2013] describes the use of Prelude for reduction, normalization and aggregation of events produced by different NIDS, deployed in a university computer network.

The design of Prelude is based on a modular approach, encompassing components such as the correlation engine (Prelude-Correlator) or the log analysis (Prelude-LML) tool. Prelude provides native support to handle and create Intrusion Detection Message Exchange Format (IDMEF) [Feinstein et al., 2007] events. The rules are defined using Python code, also being able to handle text-based logs using regular expressions. The Prelude-Manager module provides input via Unix sockets or TCP/IP communications and outputs via different adaptors like a database or text-based adaptors.

Esper and Drools are both released as platform-independent Java-based libraries, already providing several internal components and APIs such as input/output adaptors. On the other hand, Nodebrain and SEC are standalone applications developed for GNU/Linux environments, which rely on a scripting-oriented approach to execute actions, handle inter-process communication and integration with other components. The simplistic approach of SEC may be adequate for small environments, or in situations where the correlation overhead must be minimal (e.g., in embedded systems).

2.4.2 Event Processing Architectures for Big Data

Classic correlation tools such as those aforementioned are no longer able to cope with the requirements of current IACS, especially in the highly distributed and capillary scenarios of modern industrial systems, and in terms of capacity for flexibly processing very large volumes of data. On the other hand, in the generic field of *Big Data* several architectures and tools have been recently proposed to address such limitations.

Since the "three V's" (Volume, Velocity and Variety) discussion in 2001 [Laney, 2001], the concept of *Big Data* has been successively refined and revised, raising increasing interest in the past few years. It is often linked to a multitude of dedicated and highly efficient tools addressing the problem of acquiring, processing and presenting data in their different shapes [Saggi and Jain, 2018] [Mohamed et al., 2020].

Notably, in 2014, the Lambda Architecture concept was formalized, encompassing two distinct data processing schemes into a unified framework: batch and stream processing [Marz and Warren, 2015]. Later, the Kappa architecture was introduced, containing only the stream processing path [Kreps, 2014b] (Figure 2.9).

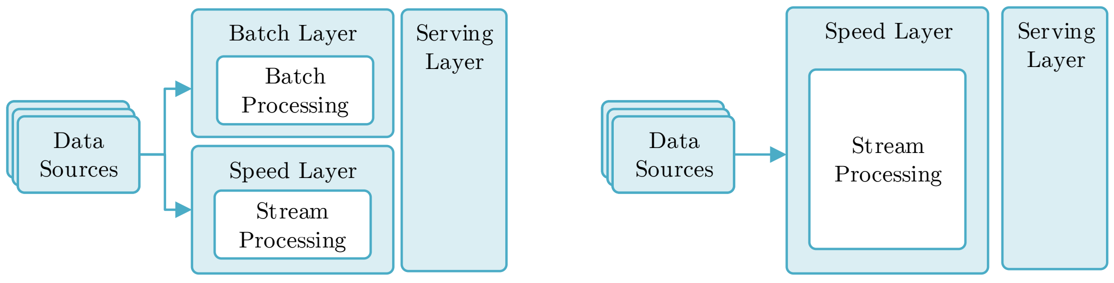


Figure 2.9: Lambda Architecture (left), Kappa Architecture (right) (adapted from [Forgeat, 2015])

Both approaches have advantages and disadvantages [Forgeat, 2015] [Feick et al., 2018]. The two processing paths considered by the lambda architecture are often considered as useful to cope with heterogeneous processing needs (heavy processing algorithms vs. low-latency computations). This way, both paths can be further and independently optimized, leveraging specialized tools for each processing scheme.

The batch layer addresses *slow* algorithms (which do not produce immediate results), being also considered useful in the re-computation of historical data – opposed to the focus on real-time within the stream processing. On the other hand, the need to support two separate pipelines, often using distinct tools, is argued to make the batch layer more complex. Currently, such a constraint is less relevant. For instance, unified APIs such as the Apache Spark dataset Application programming interface (API) can already be used to abstract and express an either bounded (batch) or unbounded (stream) dataset [Apache Software

Foundation, 2020d]. Hence, the processing logic becomes simplified.

In the Kappa architecture, all the data is considered as an append-only log (the data stream). The processing logic uses an incremental strategy, where each streaming job can theoretically provide faster outputs by processing each new record at a time – as opposed to waiting for the batch interval. The re-computation of historical data, a selling point in the lambda concept, leverages the data immutability concept to reconstruct the original stream using independent jobs. Note, however, that such re-construction is constrained by the retention period of each stream, whereas in batch mode data storage is typically used to persist historical data.

More recently, the Delta architecture [Databricks, 2019] was proposed to truly merge batch and stream processing paths into a single pipeline (opposed to one architecture supporting two different pipelines). The Delta architecture (and Delta Lake [Linux Foundation, 2019] – a Delta architecture implementation) takes a different direction. Instead of considering data immutability, it uses a multi-hop strategy where data tables are successively computed, refined and updated.

All these architectural concepts have been applied into a wide range of domains, such as the cyber-security field. Figure 2.10 provides a list of open-source tools and libraries that emerged in the last years, to support the development of this concept of real-time Big Data processing. This is not intended to be an exhaustive list, just a high-level timeline of several tools relevant and further explored in the context of this thesis. A more detailed analysis about the evolution of such tools can be found in [Casado and Younas, 2015] [Saggi and Jain, 2018].

2.5 The Concept of SIEM as a Unified and Holistic Approach

The last sections discussed anomaly detection algorithms and event processing and correlations tools. This section introduces the concept of SIEM, as a more complete and holistic approach to security monitoring.

The SIEM term was originally coined by Gartner [Williams and Nicolett, 2005], to name systems able to aggregate different types of events from multiple sources. While SIEMs were originally associated to log data, currently they cover a much broader set of application domains and focus not only on simple event correlation, but on the entire incident detection and response process. Within this document, the term SIEM will refer to the concept of a full-featured SIEM, sometimes referred to as the next-generation SIEM or evolved SIEM, that

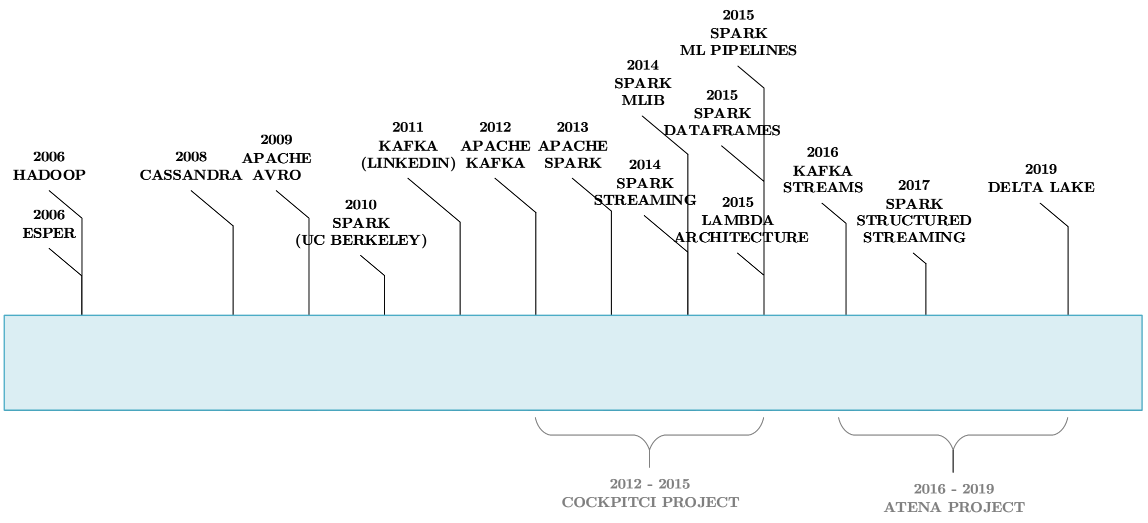


Figure 2.10: Timeline of Open-source Real-time Big Data and event processing tools

allows fulfilling the continually changing needs of modern security monitoring. Related and somehow competing concepts have also emerged, such as Security Operations and Analytics Platform (SOAPA) (which, on top of log aggregation, comprises an analytics layer and AI capabilities), Security Orchestration, Automation and Response (SOAR) (focused on security orchestration and response) and Extended Detection and Response (XDR) (aggregating both detection and response).

In IACS, similarly to other domains, the myriad of process-related data sources can easily overwhelm the most experienced security operator. From a security standpoint, a full-featured SIEM is a step towards the so-called IT/OT convergence, providing a global outlook of each IACS domain. Another key aspect, often overlooked in the literature, is how to characterize events in such a way that they can be understood by all the actors (either the different software components, security practitioners or third-party entities). Table 2.5 provides a compilation of different event formats and tools sorted by categories that can be used to represent security events.

When it comes to security management of infrastructures and services, a SIEM system has quickly become a mandatory component, as demonstrated for instance by: the Sarbanes-Oxley Act of 2002, Sec 103 (Auditing, Quality Control, and Independence Standards and Rules), which regulates the use of log collection, processing and retention for any traded company in the USA [Sarbanes, 2002]; the Payment Card Industry (PCI) council requisites for Data Security Standards (as stated by Requirement 10 "Track and monitor all access to

Table 2.5: Event formats and tools (compiled from [Pawlinski et al., 2014] [Roth and Patzke, 2019] [The MITRE Corporation, 2020])

Formats for low-level data	Actionable observables	Enumerations	Scoring and measurement frameworks	Reporting Formats
NetFlow	cybox	CAPEC	CCSS	ARE
IPFIX	MAEC	CPE	CVSS	CVRF
PCAP	MM DEF	CVE	CWSS	IODEF
PcapNG	OpenIOC	CWE	XCCDF	IODEF-SCI
CEF	Snort rules	ISI		IDMEF
	YARA rules	OSVD3		MARF
		SWID Tags		OVAL
		TLP		STIX 2
		WASC TC		VERIS
				X-ARF
				ATT&CK
				Sigma

network resources and cardholder data") [Payment Card Industry, 2002]; the North American Electric Reliability Corporation (NERC) CIP-009-2 Critical Infrastructure Protection [North American Electric Reliability Corporation, 2009]; or the Information Technology Infrastructure Library (ITIL) framework [Steinberg et al., 2011], which encompasses components for incident response in line with the role of the SIEM concept.

Following such trends, leading commercial SIEM solutions such as IBM QRadar, Splunk, Exabeam, Rapid7, LogRhythm, Securonix or Dell RSA [Kavanagh et al., 2020] are no longer solely focused on event aggregation but also support a wide range of features, from detection to incident response. Nevertheless, commercial SIEM systems are typically provided as generic appliances, software or managed services, lacking specific support for specialized application domains such as IACS.

2.5.1 SIEM-inspired Frameworks for IACS

This subsection surveys SIEM-inspired frameworks proposed in the literature for the domain of IACS cyber-security. These frameworks are generally characterized by three key functionalities: collection of information from several sources in real-time; pre-processing and aggregation of those events; and processing of those events in order to determine if they correspond to anomalies.

A tri-modular platform to support existing cyber-security tools at Smart Grid Control Centers, composed of a data module, a classification module and an action module, is presented in [Sundararajan et al., 2018]. This approach explores the advantages of having several types of data sources and a data layer that implements several types of data preprocessing and data ingestion. Nevertheless, the analytic component is limited by a single classification approach (i.e. LSTM), which might be not sufficient to cope with all types of anomalies. Moreover, details about the internal parts of the framework are not provided. This raises multiple questions, such as: How are the outputs of several heterogeneous sources jointly processed? Which data format is used? Which feature space is used in the classification module?

Similarly, the authors of [Khodabakhsh et al., 2017] proposed a framework for fault detection, targeting oil and gas industries. This framework uses a lambda architecture based on widely used open-source tools (i.e. Apache Kafka, Spark and Cassandra). Nevertheless, few details about the implementation and obtained results are provided.

The authors of [Yang et al., 2017] focused on time-series analysis within an industrial field environment, presenting a Big Data framework that encompasses several layers: acquisition, transmission, data processing and visualisation. All components were based on widely used open-source tools such as Message Queuing Telemetry Transport (MQTT), Apache Kafka, Spark, InfluxDB and Grafana. A custom JSON-based message format was used for seamless communication between the components. Despite the potential value of such a platform, there are no implementation details or more practical evaluation results.

A SCADA honeypot environment, together with a SIEM system for cyber-attack profiling, is presented in [Lee et al., 2016]. Although the platform was not developed for detecting cyber-attacks within a real SCADA environment, it provides some related functionalities, such as log processing capabilities, a real-time processing approach and a dedicated visualization interface. Similarly to others, the authors designed the platform based on open-source software such as an Elasticsearch stack [Elasticsearch B.V., 2020] and Suricata [Open Information Security Foundation, 2020]. Nevertheless, no details were provided about the used visualization techniques, the evaluated processing algorithms or the adopted message formats.

In [Kang et al., 2016], the authors explored the Suricata detection plugin functionality to perform an additional packet inspection to recreate a stateful analysis of the IEC 61850 [International Electrotechnical Commission, 2020b] protocol family. This is an interesting approach to complement single packet signature matching, especially for complex protocols.

The challenges and the desired features of a policy-based SIEM are analysed in [Gao et al., 2016]. This is a different but interesting approach, focused on the correlation of general business policies rather than on physical process values or network traffic. The authors describe the classic example of a logon into a system event, detected after a leaving event. Individually, each of the events could be regarded as normal, but when considering their sequential order they reveal an anomalous pattern.

A framework focused on EtherNet/IP and CIP [Schiffer, 2016] protocols is presented in [Ghaeini and Tippenhauer, 2016]. The authors extended the functionality of Bro, an open-source NIDS, and proposed a hierarchical deployment of several instances (based on IEC 62443 SCADA reference architecture levels [International Electrotechnical Commission, 2018]). All the Bro logs were aggregated using ElasticSearch and visualized using a dedicated web interface. While this approach can successfully detect several types of attacks, few implementation details were provided regarding the framework itself.

Table 2.6 presents a summary of the surveyed frameworks. Most of them were designed to address a single specific problem or a single protocol. The ability to combine multiple sources, techniques and protocols into a comprehensive framework for supporting SCADA operators' decisions seems to remain a challenge.

Table 2.6: Summary of the latest literature contributions of SIEM-related Frameworks

Focus	Data Sources	Data Format	Messaging	Analytics	References
Smart Grid	Logs and Sensor data	N.A.	Kafka	Spark, R and TensorFlow	[Sundararajan et al., 2018]
Oil and Gas Industry	Logs and Sensor data	N.A.	Kafka	Fault detection using Esper and Spark	[Khodabakhsh et al., 2017]
Industry 4.0	OPC / Network Data	Custom JSON based	MQTT / Kafka	Storm	[Yang et al., 2017]
SCADA network traffic	Public exposed SCADA Honeypot	Suricata alarms	N.A.	Profiling analysis using ELK stack	[Lee et al., 2016]
Smart Grid	IEC 61850 Network Traffic	Suricata alarms	N.A.	IEC 61850 Stateful protocol analysis based on Suricata Plugins	[Kang et al., 2016]
Policy Monitoring	Security Control Logs	Custom Structured Logs messages	N.A	Log Correlation	[Gao et al., 2016]
Water Treatment	CIP/EthernetIP traffic	Bro alarms	N.A.	Log aggregation from multiple domains with ELK	[Ghaeini and Tippenhauer, 2016]

2.5.2 A Taxonomy for Evolved SIEMs

In the course of the aforementioned survey, no specific SIEM taxonomies were found. In order to fill this gap, this subsection proposes a taxonomy that captures and organizes the main concepts and features of evolved SIEMs, from key capabilities to processing models (Figure 2.11). This taxonomy considers ten different characteristics, as detailed below. Apart from the specific data feeds, most of those features are not restricted to the IACS domain, since they applicable to many other domains. Next, each of those features is individually discussed.

- **Key capabilities.** Refers to the supported security-related capabilities, including classic event correlation techniques such as event suppression, compression, generalization and root cause analysis [Jakobson and Weissman, 1995]. Event correlation, one of the original key capabilities associated with a SIEM, can be further organized according to the specific clustering approach: tuple-based, feature-based, or time-based. For instance, an event might be filtered according to a predefined threshold, event characteristic or timestamp. Time-based aggregation is typically divided into hopping windows or tumbling windows [Lal and Suman, 2020]. Incident detection capability is also one of the key SIEM features, and be further split into categories such as signature and ML-based anomaly detection techniques. A more detailed view of specific types of anomaly detection methods can be found in [Hindy et al., 2018]. Moreover, the taxonomy also includes the latest security capabilities envisioned to be part of a full-featured SIEM: incident response, security orchestration, security automation, user-behavior, analytics and regulatory compliance.

- **Data feeds and Threat Indicators.** Different types of data sources and other threats indicators are paramount for a comprehensive monitoring strategy. Among others, in the SCADA domain it is critical to monitor all the network traffic in all segments (both OT-specific and remote connections) and the network infrastructure itself (e.g. routers, switches, gateways), all the different types of logs (e.g. HMIs, engineering stations, historians), access controls (e.g physical controls to enter in a substation, logon attempts), the output of other security-specific endpoints (e.g. firewalls, NIDS, Host Intrusion Detection System (HIDS), honeypots, data diodes) and the physical process values through the usage of specialized physical probes. Other data sources, such as business policies or external threat intelligence feeds (e.g. vulnerability alerts correlated with existing infrastructure assets), might also be relevant.

- **Key Layers.** A complete SIEM should encompass at least the following key layers: data ingestion, referring how the events are ingested in the overall SIEM; data streaming, which includes how the inter component communication is performed (critical for

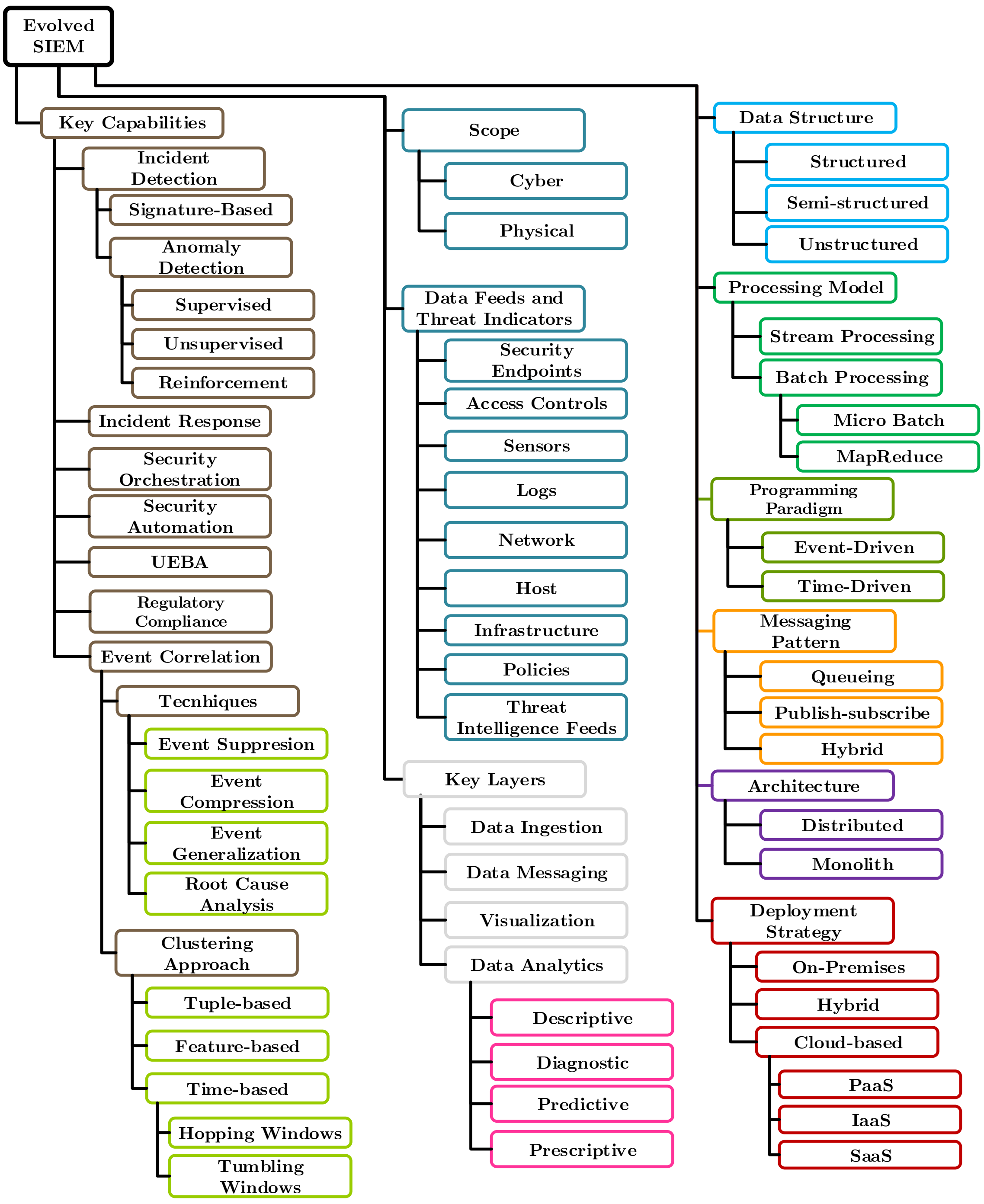


Figure 2.11: Evolved SIEM proposed taxonomy

distributed systems); and data analytics, where different types of analytics (with different levels of autonomy and complexity) can be included (e.g. descriptive, diagnostic, predictive and prescriptive [Pathak et al., 2018]).

- **Scope.** IACS rely on a wide range of interconnected physical assets. This way, a SIEM can be used to handle physical, cyber or both domains (e.g. compare network-based traffic with physical access controls to spot anomalous patterns). This might be useful for instance to correlate maintenance operations with anomalous traffic patterns or security access controls.

- **Data Structure.** Three major types of data structure were identified: structured (all the fields of each event follow a rigid semantic structure); semi-structured (containing a hybrid between a rigid structure and flexible fields); and unstructured (not structured at all). Whereas in theory more structure means more fine control about the event contents, in practice this can also bring an increased complexity to map and describe all the inner details of each potential event.

- **Processing Model.** Two major types of event processing are commonly referred. Event Stream Processing, where the processing logic is applied in near real-time to a unbounded stream of data (e.g. events generated from field sensors), and batch processing, where the processing is applied to bounded batch of events. In batch processing, for smaller batch sizes (i.e. processing less events but at a more frequent rate) we might also refer to micro-batch processing.

- **Programming Paradigm.** Batch processing is typically time-driven (e.g. count every 2 minutes) or event-driven (e.g. average every 10 records). Stream Processing is often tied to an event-driven strategy since the processing is applied on-the-fly to a stream of data (i.e. events). Regardless, in the Stream Processing, the windowing can be both time-driven or event-driven (i.e. using a global time-scheduler or relying on the event time).

- **Messaging Pattern.** The messaging pattern can range from a queue where each message is consumed by a single consumer, a publish-subscribe model were each message can be consumed by more than one consumer, or a hybrid approach such as the one used by Apache Kafka, were each message can be consumed multiple times by different consumer groups and, at the same time, load-balanced across each consumer member within the same group.

- **Architecture.** The evolved SIEM architecture can range from a centralized and monolithic application to a distributed and micro-service based architecture. Whereas monolithic applications are typically simpler, a distributed architecture can be leveraged to load-balance for instance the messaging and analytics layers or to reassemble fail-over strategies.

- **Deployment Strategy.** Last, but not least, the deployment strategy might vary from the classical on-premises deployment up to cloud-based deployment in different levels (i.e. Software as a Service (SaaS), Plataform as a Service (PaaS) and Infrastructure as a Service (IaaS)). Whereas the security and privacy aspects should be taken into consideration, the multiple cloud-based strategies can represent a cost-effective solutions with shared responsibilities. For instance, an Essential Service provider might require a large computation infrastructure but it might not want to own or manage it.

2.6 Summary

Anomaly detection based on ML techniques is increasingly used in different fields, including IACS, and is expected to bring valuable improvements to support (albeit not replace) the overall intrusion detection process. Nevertheless, applied research towards practical IACS solutions is still in its early stages.

The majority of the reviewed literature focused on detecting anomalies such as network-based cyber-attacks or physical faults by looking either at physical process properties, network SCADA communications or a combination of both. Other potentially relevant features, such as diversified log sources and host-based events, are commonly ignored, therefore missing an important opportunity to develop a more comprehensive approach covering a broader spectrum of attacks.

For instance, a model that only takes process features into consideration will simply fail to detect an important range of attacks that do not have a direct impact on the process, such as network scans or brute force login attempts. If we look at the example of recent APT campaigns targeting IACS [Kasperky, 2019], they typically start by collecting information from the environment over long periods of time – a latent stage during which the threat may go undetected for months if one just looks at physical process values. A NIDS (with support for SCADA protocols) is still a valuable source for detecting such kind of unauthorized communications – one of the most referred SCADA security issues. Nevertheless, since a NIDS is limited to network traffic, it is clearly insufficient to address all the security needs. This suggests that including more data sources is a better strategy.

On the other hand, the complexity of a SCADA environment is one of the biggest challenges for anomaly detection based on ML approaches. The amount of different types of potential features can severely affect the performance of the model, and should be carefully selected. Another frequently observed problem is the lack of datasets. Given the cost and the complexity of recreating an industrial control system, most surveyed works focused either on small scenarios or in the few publicly available datasets. Moreover, some of them lack a diversity of anomalies, resulting on imbalanced datasets that might lead to inaccurate or biased classification issues. Intuition also suggests that a better approach is to take into consideration both supervised (more apt at detecting known issues) and unsupervised techniques (more suitable to detect unknown anomalies).

Similarly, by combining several different approaches, it is possible to infer both local and global anomalies (e.g. a single component fault on a given domain might not have immediate effects on the overall process). Additionally, other techniques such as stream processing (especially windowing) might help reducing the number of false positives, since the analysis is performed on the top of a group of events, rather than on a single event. Finally, as we are moving towards a Big Data problem, in order to avoid scaling issues it is critical that the chosen approaches can fit into scalable, distributed and parallel computation environments.

Taking all of that into consideration, in the next chapter, we propose a comprehensive and holistic data-driven framework for intrusion and anomaly detection in IACS environments. Such framework follows an evolved SIEM-like approach leveraging various types of data sources and ML-based techniques to detect potential intrusions and anomalies.

3
A Holistic Intrusion and Anomaly Detection System for IACS

This chapter introduces the proposed holistic framework for intrusion and anomaly detection in Industrial Automation and Control Systems (IACS) scenarios. This framework was built upon the idea of having an evolved Big Data-like Security Information and Event Management (SIEM) system capable of detecting in (near) real-time the occurrence of cyber-physical attacks in such complex environments. By contrast to other works surveyed in the previous chapter, this research focused on the concept of having a holistic framework capable of supporting different scenarios and techniques. Thus, such a Big Data-like SIEM was designed to be able to integrate different Machine-Learning (ML)-based mechanisms, which in turn can be used to detect different types of anomalies on top of large amounts of heterogeneous data produced by all devices, sensors and actuators. Moreover, the proposed framework was also inspired upon the idea of a Lambda Architecture (cf. Section 2.4.2), taking into consideration both Batch and Stream Processing techniques.

This chapter starts by enumerating and discussing the key driving characteristics of IACS that motivated and influenced the design of the proposed holistic intrusion and detection framework (Section 3.1).

Next, for sake of context, the main ideas behind the CockpitCI and ATENA projects and the related Hybrid Environment for Development and Validation (HEDVa) testbed are presented. As already mentioned, this work was performed in the scope of those two projects – it was initially conceived within the CockpitCI project and, later on, extensively redesigned in the ATENA project.

In the CockpitCI project (Section 3.2), the author of this thesis contributed to the state-of-the-art analysis of intrusion and anomaly detection techniques, the research and development of the event correlation components and overall intrusion detection architecture.

In the scope of CockpitCI and ATENA, Israel Electric Corporation designed and implemented the HEDVa testbed, which was essential for the work conducted in this thesis, both for early exploratory work and for defining and evaluating the validation scenarios of the framework components. This testbed is briefly presented in Section 3.3.

Later, in the ATENA project (Section 3.4), the author of this thesis was one of the key contributors to the overall detection architecture and was directly responsible for the conception and development of the streaming processing and data analytics layers (cf. Chapters 5 and 6). Furthermore, in ATENA, the author of this thesis was responsible for the research and development of a proof-of-concept ML-based mechanism to aggregate and classify network traffic. Finally, in both projects, the author of this thesis was also the main contributor to the development of different validation scenarios in the form of practical attack scenarios to support the functional validation of the different components (cf. Chapter 4).

The proposed overall framework for the holistic Intrusion and Anomaly Detection System (IADS) is presented in Section 3.4, since in practice it was adopted as reference architecture for the ATENA IADS. Section 3.5 discusses the stream processing and data analytics layers, which are key contributions from this thesis and will be further detailed in the next chapters.

3.1 Key Driving Characteristics of IACS

This section overviews the key characteristics of IACS that directly influenced the design of the proposed framework. Those characteristics were identified based on the literature reviewed in the previous chapter, the work conducted in the scope of the CockpitCI and ATENA projects, and on what we envision for the next generation of IACS (e.g. the Smart Grids of the future).

First of all, it is important to acknowledge that, contrary to common Information Technology (IT) networks, an incident on IACS can have an immediate catastrophic economic impact and, ultimately, threaten human-lives (e.g. environmental disaster in a nuclear power plant, energy blackout on an electrical grid, or water poisoning in a water-distribution system). As an example, according to [Maynard and Beecroft, 2015], an unlikely but plausible coordinated attack against the USA power grid, hypothetically damaging between 50 and 100 power generators across 15 States, was estimated to cause a total of \$243bn up to \$1trn of direct and indirect costs. Such values were considered based on an average power outage from 3 days up to weeks of rolling blackouts having an impact on 93 million people, which already gives a clear indication of how serious this can escalate. Despite of all the countless benefits provided by the recent advances in industrial automation (e.g. more intelligent and connected processes, the Operational Technology (OT)/IT convergence), there is the side effect of increasing dependency on the technology, which in the case of a cyber attack can turn into huge losses. Thus, IACS must be considered as **mission-critical** and their **cyber-security is of utmost importance**.

Similarly, IACS and Supervisory Control And Data Acquisition (SCADA) systems traditionally relied on data collected from field (e.g. sensors values) and components such as data historians to remotely supervise and control different parts of the system (e.g. open/close a circuit breaker in a electrical grid based on field measurements). As we move into more intelligent and advanced systems, data assumes an even more relevant role. For instance, in Smart Grids, Energy Management Systems (EMS) strongly rely on the valuable data collected across the different domains to understand, model and forecast energy consumption patterns, monitor the grid stability or implement advanced self-healing mechanisms. This way, data is already a central piece in modern automation and should be considered as one of main pillars of next-generation IACS – expected to be increasingly **data-centric**. From a security standpoint, such data is also valuable to monitor and timely detect potential anomalies (e.g. to detect a cyber-attack based on anomalous data patterns) – thus the emphasis on the data-driven nature of the proposed approach.

The next-generation IACS are also expected to be highly distributed. In fact, in numerous domains, IACS already span across larger geographical areas. For instance, a Smart Grid encompasses a distributed smart metering infrastructure, the centralized energy production facilities, the high/medium/low voltage transmission/distribution infrastructure, and the Distributed Energy Resources (DER) up to the customer premisses. Moreover, as postulated before, the leaning towards hybrid edge/cloud architectures (where non-critical components can be, for instance, deployed on a third-party cloud facility) and the exponential growth of connected assets (with their ramifications) are blurring and eroding the physical boundaries of IACS environment. This way, even more **distributed and capillary IACS architectures** are expected in a foreseeable future. Therefore monitoring framework solutions should also be be able to collect and cover such distributed environments.

Finally, the intrinsic characteristics of the IACS physical processes are substantially different from one domain to another (e.g. the underlying processes within a Smart Grid are quite different than those within a nuclear power plant). Worst, the wide range of assets, technologies and protocols used in such distinct systems demands an extra effort to accommodate them all. This is not expected to change in the near future. Moreover, the ongoing usage of undocumented or proprietary protocols from the past poses additional barriers to the cyber-security monitoring process. Likewise, network communications, depending on if we are referring to time-sensitive operations such as real-time remote I/O or non-critical telemetry pulls, are quite different (e.g. different levels of network periodicity, determinism or priority). As an example, in the Common Industrial Protocol (CIP) protocol, there are two messaging modes: *explicit* for non-time-critical messages and *implicit* for real-time I/O data). This way, IACS are highly **heterogeneous environments**, where different components and protocols require different monitoring approaches and security solutions.

3.2 CockpitCI - Towards Cyber-Security Awareness

In 2010, Stuxnet was considered by many as the tipping point from which IACS security, frequently neglected in the past, started to get the deserved attention. Thenceforth, seemly far-fetched threats started to get widely discussed and researched.

The CockpitCI project (2012-2014) focused, among other objectives, on developing a framework to monitor and evaluate the cyber-security of a Critical Infrastructure (CI). The CockpitCI detection subsystem was designed as a Dynamic Perimeter Intrusion Detection System (DPIDS) to continuously assess the electronic security perimeter of a SCADA system.

This DPIDS was designed to collect information from the environment in (near) real-time and, afterwards, analyse the probability of the occurrence of a cyber-attack. The DPIDS, built upon the idea that *the whole is greater than the sum of the parts* [Cruz et al., 2016], encompassed a combination of diversified and innovative security probes, together with hybrid and multi-level correlation capabilities. Its design already considered the benefits of having a unified framework across multiple domains, along with the benefits of having distinct intrusion detection techniques. These two ideas were the initial inspiration and the main precursor of the remaining work developed within this thesis.

Figure 3.1 provides a high-level view of the DPIDS architecture, which was composed of the following main components [Cruz et al., 2016]:

- **Event Correlators.** A two-level approach was used to filter and aggregate the security events originated from the different segments. Based on an early comparative analysis [Rosa et al., 2015], Esper was used for both local and global correlation.

- **Network Intrusion Detection System (NIDS).** Three distinct NIDS instances were used to cover each network segment, each configured to monitor all the local network traffic of each domain. Snort [Cisco, 2020] was used for this purpose.

- **Host IDS.** Host Intrusion Detection System (HIDS)s were deployed in the SCADA stations and servers to provide host-level anomalous behaviour detection capabilities. OSSEC [OSSEC, 2016] was adopted for this purpose.

- **Honeypots.** Three types of honeypots were used to cover each segment. Each acted as a decoy by simulating different types of SCADA devices (e.g. a Modbus-based Programmable Logic Controller (PLC)), detecting suspicious activity [Simões et al., 2015].

- **Shadow Security Unit.** A security and safety device attached in parallel with a PLC or Remote Terminal Unit (RTU), to continuously evaluate and compare the SCADA communications content with the actual physical state of I/O [Cruz et al., 2015].

- **Exec Checker.** A component capable of reconstructing and analysing suspicious binary files from network communications.
- **Configuration Checker.** A component used to check for unauthorized modifications of configuration files.
- **OCSVM.** An One Class Support Vector Machine (OCSVM)-based mechanism capable of performing outlier detection with high accuracy and low overhead, on the top of unlabeled network traffic data [Stewart et al., 2017].

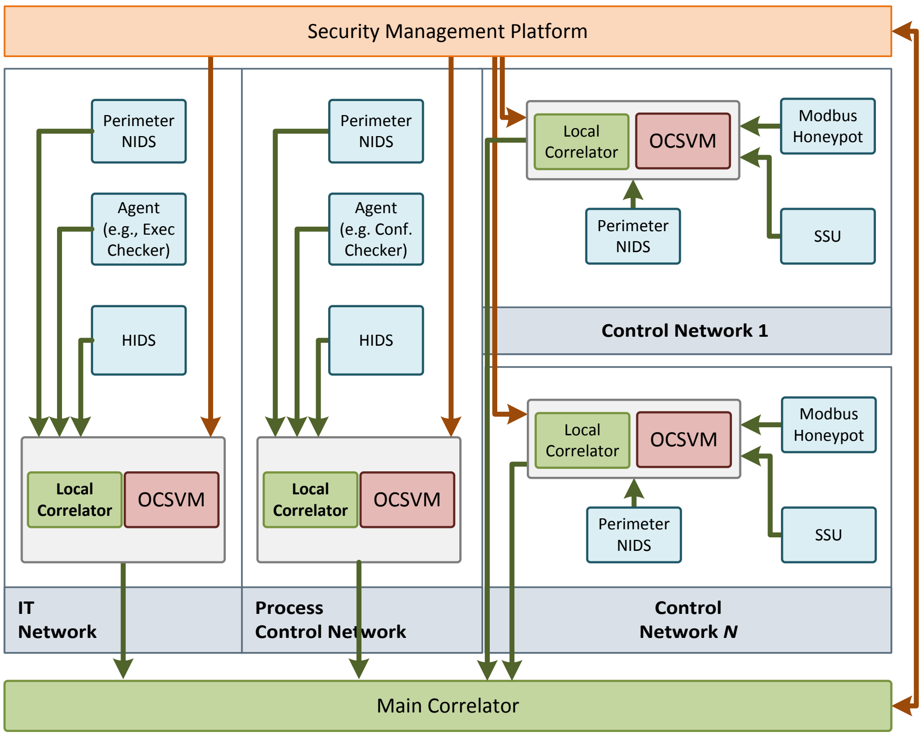


Figure 3.1: High-level view of the CockpitCI PIDS architecture ([Cruz et al., 2016]).

In a nutshell, distinct and domain-specific security probes were used to collect and report telemetry data and security-related events using the Intrusion Detection Message Exchange Format (IDMEF) format [Feinstein et al., 2007] – a vendor and component-neutral format. IDMEF allowed all the heterogeneous components to communicate the incidents in a more structured and standardized manner. Moreover, the security probes were deployed across three domains (control, process control and IT), corresponding respectively to: the field domain (SCADA levels 0, 1 and 2); the remote supervisory control (level 3); and the remaining enterprise IT domain (level 4).

In a first stage, those events were pushed to a local event correlator and to a OCSVM component used for initial filtering, aggregation and outlier detection. After this first pre-processing, events were routed to a global correlator which, based on a set of pre-defined rules, was able to classify them as potential cyber-physical incidents. Both correlator components used a rule-based approach to match and correlate events.

It should be noted that the CockpitCI detection layer was conceived by all the University of Coimbra members that participated in the CockpitCI project and not only by the author of this thesis. As mentioned before, the author of this thesis contributed to the state-of-the-art analysis of intrusion and anomaly detection techniques and to the overall intrusion detection architecture. Moreover, the author of this thesis was also the key contributor to the research and development of the event correlation components and to the development of different validation scenarios in the form of practical attack scenarios to support the functional validation of the different components (cf. Chapter 4).

3.3 The HEDVa Testbed

The HEDVa testbed, designed and implemented by the Israeli Electric Corporation, emerged in the CockpitCI project and was extended later in the ATENA project. It supported the research, development and validation of both projects. HEDVa followed a hybrid approach combining both real and physical SCADA components into an electrical process simulation (natively implemented in the ladder logic of additional PLC devices). For this work, such an highly complex environment allowed to recreate different attack scenarios, and to assess the cyber-security of different physical devices. Last but not least, it allowed to conduct all the PCOM-related security research (cf. Chapter 4).

Figure 3.2 shows the networking view of the HEDVa testbed, highlighting its different domains, organized into different Virtual Local Area Network (VLAN)s along with a central gateway, simultaneously connected to all the network segments and providing remote access through a Virtual Private Network (VPN) connection. The testbed was physically deployed in Israel and the different project members needed to remotely access it. Moreover, as shown in the figure, all the CockpitCI components were setup in an additional and separate VLAN, to avoid any interference with the process-related network segments while still allowing to remotely manage them. Likewise, the NIDS instances were setup with an extra link for the sole purpose of mirroring all the network traffic for Deep Packet Inspection (DPI) and further analysis.

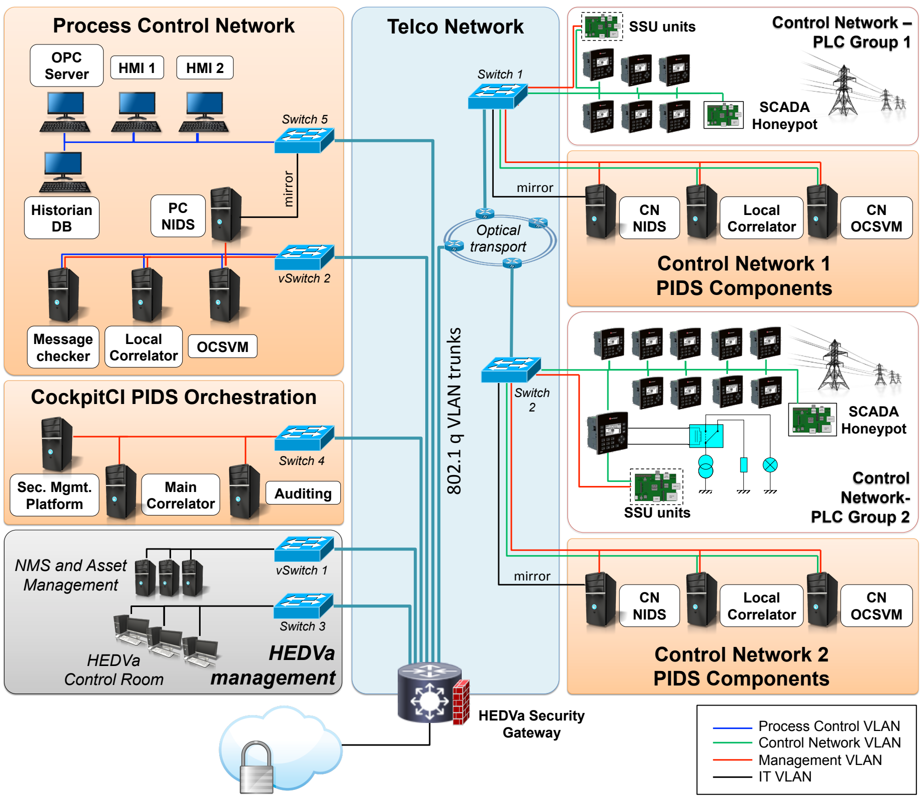


Figure 3.2: HEDVa networking architecture ([Cruz et al., 2016])

Eleven Unitronics PLCs were used in CockpitCI to emulate the behavior of a Medium/Low voltage energy distribution scenario with two redundant energy supply links. Different "points" of the electrical grid were emulated by those PLCs, including, as later detailed, the current, voltage and the state of electric circuit breakers (Figure 3.3). Part of the overall project evaluation, the HEDVa testbed also allowed to simulate different grid failure scenarios and to execute the respective recovering procedures. Thus, among others, one of the goals of the practical scenarios explored in this thesis (cf. Chapter 4) was to prevent the SCADA operator from noticing a failure in the grid and/or to interfere with recovery procedures.

Fast-forward to ATENA, the HEDVa testbed was significantly enriched with high-fidelity simulations such as energy production systems, energy transmission links, smart homes and different energy consumption profile simulations – representing the different domains of a Smart Grid. Moreover, it also included the deployment of additional SCADA-specific assets

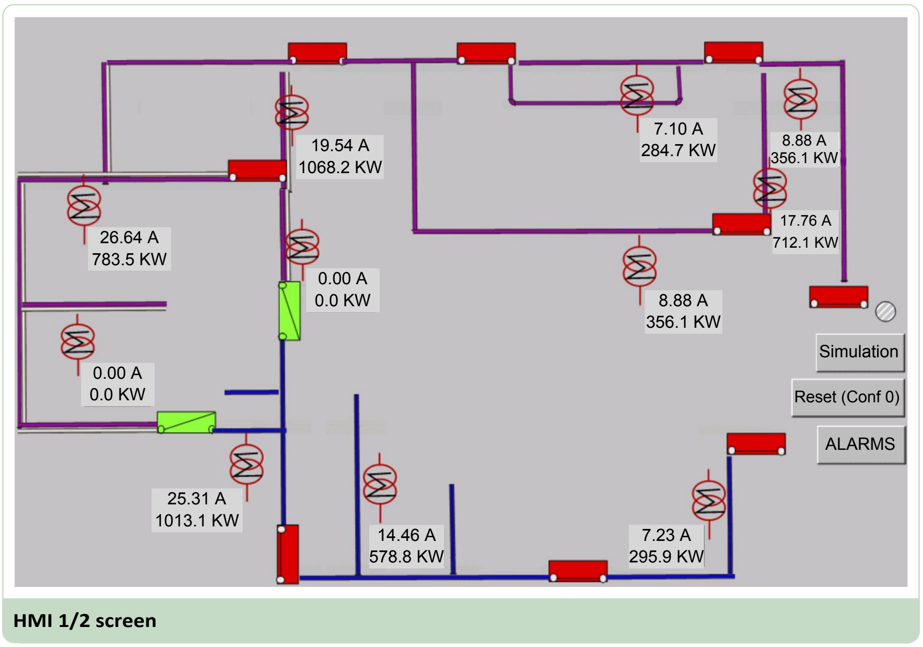


Figure 3.3: HEDVa grid scenario, with circuit breakers and two energy paths ([Cruz et al., 2016])

and protocols. Figure 3.4 shows the mapping between the Smart Grid domains and the PLC models used in each domain. Figure 3.5 provides various examples of Human-Machine Interface (HMI) interfaces developed by the Israel Electric Corporation to remotely control the environment, such as the diesel generators simulation interfaces, additional substations and the High/Medium/Low grid infrastructure.,

HEDVa and the different domains it emulates already hint at how complex and heterogeneous an IACS can be. This further reinforces the importance of having a holistic approach to collect evidences and monitor the entire infrastructure. For this thesis, such additional domains and the increased assets coverage were also decisive to explore additional practical scenarios.

3.4 ATENA – a More Scalable Architecture

Instead of replicating the recipe of CockpitCI, in the ATENA project (2016-2019) the entire detection layer was redesigned, taking into consideration the aforementioned key characteristics of IACS.

While CockpitCI was based on the underlying idea of monitoring a single infrastructure,

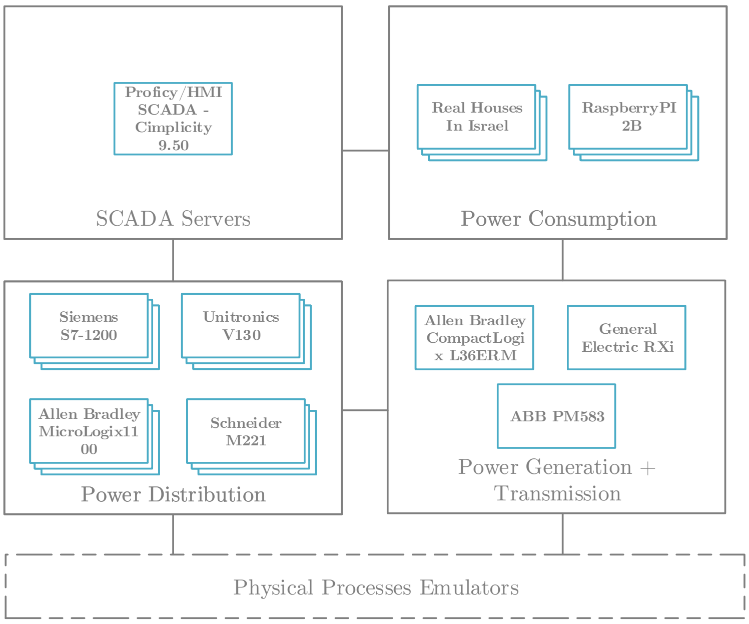


Figure 3.4: Smart Grid domains emulation in HEDVa / ATENA

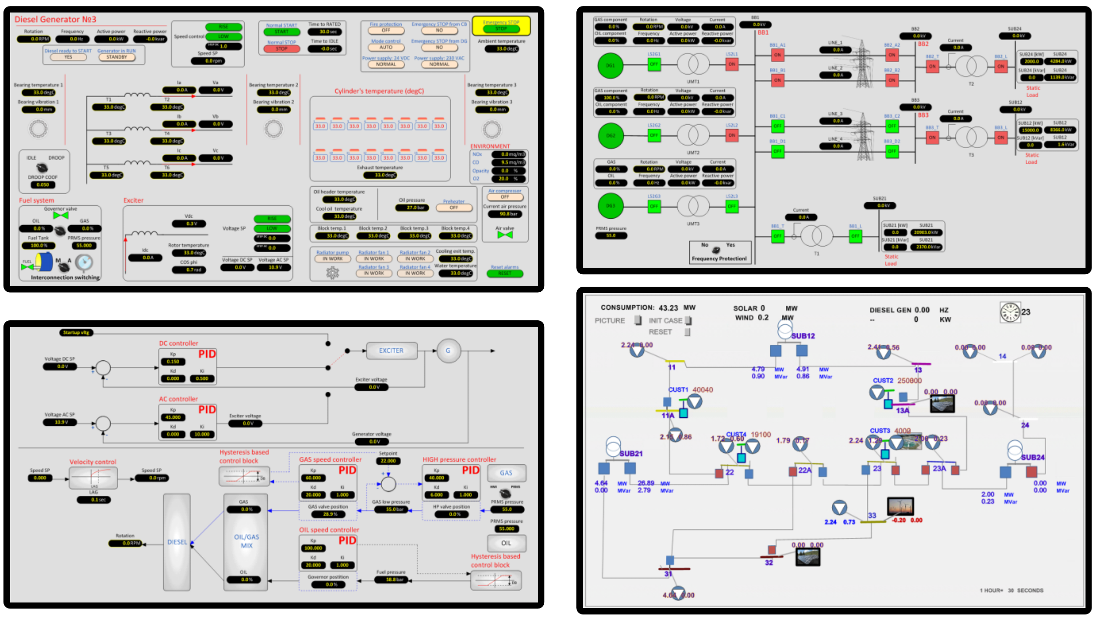


Figure 3.5: The many HEDVa scenarios in ATENA

ATENA focused on highly distributed, capillary and heterogeneous IACS. Moreover, ATENA leveraged on the integration of new open-source tools and on a major leap in Big Data-oriented approaches. Eventually, this led to the design of the ATENA detection layer (designated as IADS) as a Big Data-like architecture, truly data-driven, distributed and scalable.

The ATENA IADS decoupled the evidence-gathering, event transport and processing components by using a multi-layer approach (Figure 3.6) composed of the following components [Rosa et al., 2020]:

- **Management.** The management module was designed as a multi-tenant graphical interface to easily visualize all the security events and remotely reconfigure the IADS components, as well as to monitor and visualize the health state of the platform.

- **Probes.** Representing the *eyes* of the platform, a set of domain-specific probes were developed to collect all kinds of data (e.g. process values, network traffic data, security events), from classical NIDS up to SCADA specific probes.

- **Messaging system.** A dedicated event bus, following a hybrid approach between queuing and publish-subscribe mechanisms to support all the inter-component communication.

- **Domain Processors.** Such components, ideally deployed near the collection points, were used as mechanism to pre-process the events coming from the probes. More than just filtering and routing, such components were also used as a way of aggregating different events and extracting additional statistic features (e.g. average and standard deviation of different features per time windows).

- **Big Data-like SIEM.** Designed as a scalable and distributed computation framework, it was used to implement the ML-based anomaly detection mechanisms of the IADS. Those mechanisms used a set of features extracted from the stream of events to classify and to detect cyber-attacks, as detailed in Chapter 6.

- **Forensics and Compliance Auditing.** A semi-automatic feature identification and extraction component designed to support the forensics and *post-mortem* analysis of different types of data sources (e.g. Authentication, Authorization and Accounting (AAA) sessions, physical access control systems), as well as for policy conformity checks.

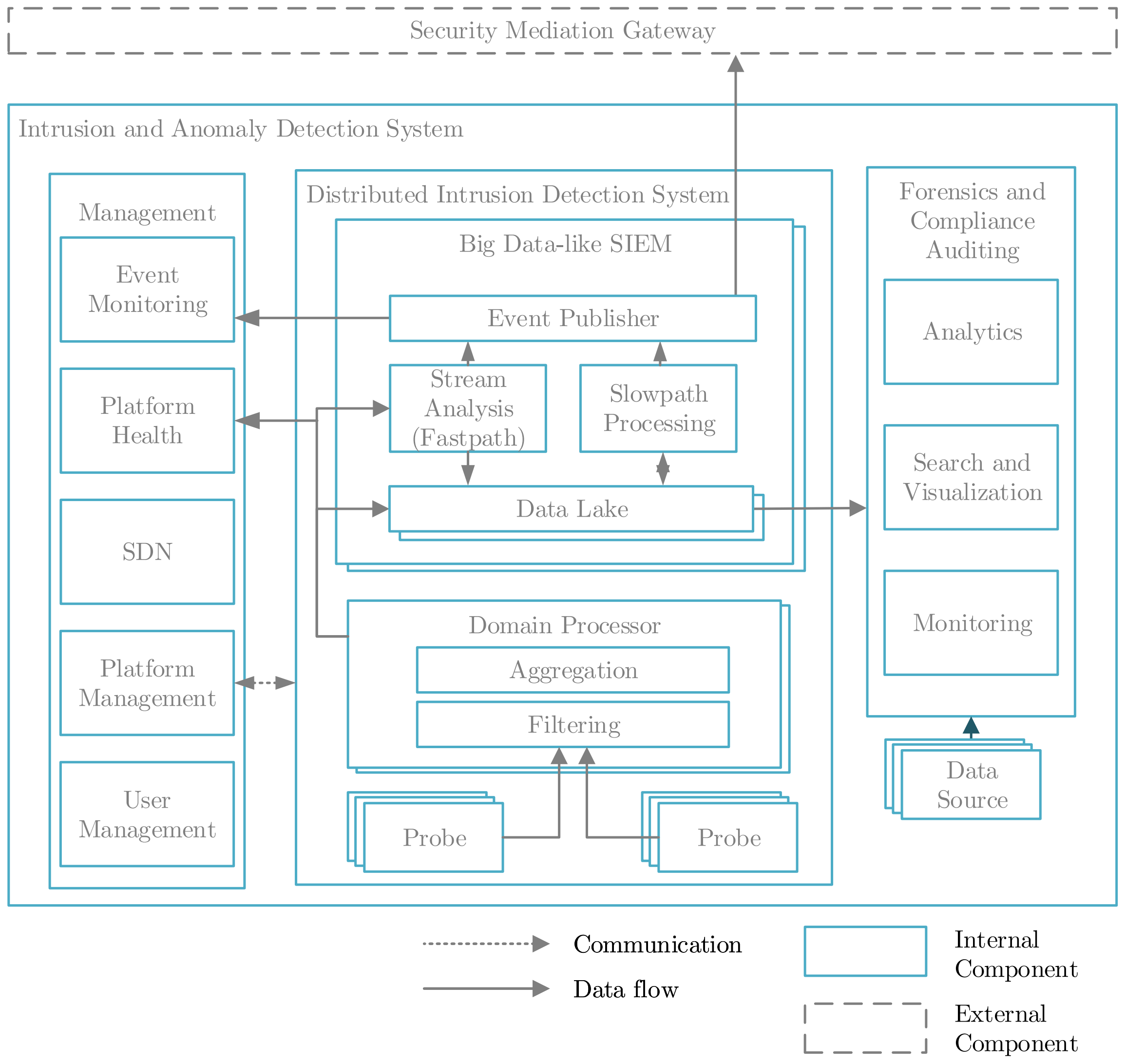


Figure 3.6: Intrusion and Anomaly Detection System architecture

As in CockpitCI, multiple probes were developed to collect evidence from the field. Moreover, as detailed in Chapter 5, in ATENA those probes used a custom and predefined format to report the events. The local event correlators and event bus from CockpitCI were replaced by a full-featured event streaming layer, which comprises the messaging system and a set of domain processors. The global event correlator was replaced by a Big Data-like SIEM (depicted in Figure 3.6) which was no longer limited to a single anomaly detection mechanism or rule-based event correlation. Instead, it incorporated the concept of a Lambda Pattern (cf. Section 2.4.2) with two main types of processing: Stream and Batch Processing. A distributed computation framework was used to support the simultaneous deployment of different ML-based mechanisms capable of running in parallel and on top of a horizontally scalable infrastructure.

As in CockpitCI, the overall IADS architecture was conceived by the University of Coimbra team participating in ATENA project, with a key contribution from the author of this thesis. The author of this thesis was also responsible for the streaming processing and data analytics layers (cf. Chapters 5 and 6), as well as for the research and development of a proof-of-concept ML-based mechanism to aggregate and classify network traffic. Finally, the author of this thesis also the main contributor to the development of different validation scenarios in the form of practical attack scenarios to support the functional validation of the different components (cf. Chapter 4).

3.5 Stream Processing and Data Analytics Layers

The previous section provided a first look at the proposed holistic IADS, in the scope of the ATENA project. This section specifically introduces the two key layers addressed in this thesis: the Streaming Processing layer (i.e. the messaging system and the domain processors) and the Data analytics layer (i.e. Big Data-like SIEM), which will be further detailed in Chapters 5 and 6.

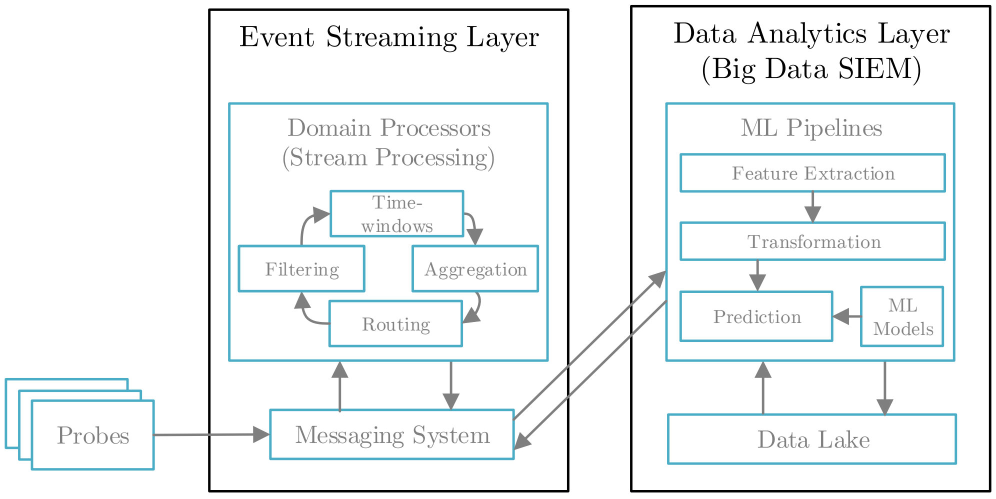


Figure 3.7: Stream processing and Data Analytics layers – the proposed approach.

The streaming layer, opposed to the somehow limited event correlation approach of CockpitCI, is composed of multiple domain processors per domain. Each one, designed as a lightweight event-driven streaming processing task, has its own topology (i.e. a chain of small processing steps). Multiple domain processors and individual tasks such as routing, filtering, time-window processing and aggregation can also be composed into complex and per domain processing schemes (e.g. a chain of small domain processors where the output of each one is processed by the next) – thus enabling virtually unlimited event processing capabilities. This way, new domain processing tasks can be seamless incorporated, on-demand, to accommodate future changes.

The messaging system decouples the communication between the different components. Spread across the multiple domains, different clusters of event brokers are used to persist and serve the events (as detailed in Chapter 5). This allows to have multiple and distributed producers (e.g. probes) reporting events to distinct topics of interest and, later, various consumers cooperatively consuming them in parallel (e.g. processing tasks). Likewise, different anomaly detection mechanisms, which can be seen as both consumers and producers, can also report their outputs in parallel.

The global correlator concept from CockpitCI was also replaced by a full-featured data analytics layer (Figure 3.7), designed as a Big Data-like SIEM which, as mentioned before, was used to run different anomaly detection mechanisms (i.e. ML-based pipelines) on top of all the data collected by probes and domain processors. As detailed in Chapter 6, each pipeline is composed of a set of steps, from feature extraction, transformation, up to the actual detection of the probability of the occurrence of a cyber-attack based on previously trained ML-models. Such a generic computation framework fed by multiple domains, together with the possibility of running different ML-based mechanisms, is expected to increase overall efficacy in the intrusion detection process as well as to support the handling of different types of cyber-security issues (e.g. computing-intensive algorithms, global deviant patterns, cross-domain incidents, network-based attacks, physical-based attacks hidden from the network).

The proposed approach was designed taking into consideration the following principles:

- **Fault-Tolerance.** IACS are designed from the ground up to be fault-tolerant and resilient to different kinds of adverse scenarios. The proposed solution must go along with such a design. First, it is imperative to ensure that none of the components harms the availability of the IACS (e.g. by avoiding active security mechanisms in the loop, which might fail and/or become themselves a target). A Single Point of

Failure (SPOF), which, in the case of compromise might affect the entire intrusion and anomaly detection process, should be avoided. While ideally no SPOF should ever exist, in practice it depends on the specific deployment scenario. For instance, by considering cost-benefit ratios, one might accept a higher downtime of a non-critical telemetry component but require strong fault-tolerant guarantees in a core component.

- **Effectiveness and Efficiency.** The proposed approach should be able to effectively detect intrusions and anomalies as well as report them within a reasonable time-frame, ideally in near real-time. No objective metrics were predetermined on purpose – it is almost impossible or unrealistic to devise a single number for all scenarios. Instead, the focus and all the design decisions privileged the state-of-the-art techniques, algorithms and tools, in the attempt to obtain the most performant and adequate way of detecting anomalies.

- **Scalability.** By acknowledging the inherent heterogeneity and the different needs of data processing, another pursued requirement refers to scalability. The proposed approach should scale to different scenarios, from very small to very large deployments. A small country like Portugal already has more than 6.5 million electricity metering points [European Commission, 2014] – which does not account for all the other data sources or security-related probes. This means we can easily scale from a few events up to millions of events per second in different scenarios (e.g. small electricity substation monitoring up to a national level distribution network control center).

- **Flexibility.** The proposed approach should not only accommodate different scenario dimensions, but also support different types of processing and techniques. It is also important to be capable of keeping up with new vulnerabilities and models.

- **Security.** Deploying such a complex monitoring framework introduces new elements within an already complex and mission-critical environment. In fact, vulnerable and insecure monitoring components might represent new potential attack vectors and be the target of cyber-attacks. The idea of compromising a SCADA system through them is a valid concern. Therefore, it should be possible to ensure the security of new components and all their communications, for instance by means of authentication, authorization and accounting.

- **Distributed computation, messaging and storage.** The proposed distributed architecture is designed from the beginning to horizontally scale in its various layers. Performance-wise, the computation layer is used to efficiently distribute the analytics workload for each node in each scenario, the resilient messaging layer to abstract and support all the inter-component communication, and the distributed storage to ensure high scalability and availability without sacrificing performance.

- **Node redundancy and strong processing semantics.** Likewise, such a distributed architecture can also be used to implement fail-over mechanisms with node redundancy, to ultimately ensure an uninterrupted security monitoring process. Moreover, strong processing semantics (e.g. exactly-once processing guarantees) should be privileged, to properly handle different types of failures.

- **Multiple detection and analytics techniques.** From the surveyed literature, it is clear that each detection technique has each own merits. This way, multiple detection and analytics techniques should be considered (e.g. local and smart probes deployed on the field as well as global analytics combining different techniques and capable of correlation inputs from multiple domains).

- **Stream and Batch Processing.** Finally, in line with the previous topic, the proposed framework adopts a combination of both Batch and Stream Processing. The main idea is not to limit the nature of processing techniques but rather to embrace a holistic approach where different types of processing techniques could be deployed on-demand on different scenarios, without being constrained by the underlying computation framework.

3.6 Summary

By contrast to the majority of the surveyed research on IACS security, mainly narrowed to theoretical mechanisms or specific algorithms, the proposed approach widens the discussion towards effective strategies and approaches to materialize and combine different components, mechanisms and tools, in the form of a highly flexible and scalable framework.

This chapter presented the main building blocks of the proposed approach, emerged in the context of the CockpitCI and ATENA projects. After presenting the overall IADS architecture proposed by the Coimbra team working in those projects, the two main layers addressed in this thesis (Stream Processing Layer and Data Analytics Layer) were presented.

Before a more detailed discussion of those two layers (Chapters 5 and 6), the next chapter presents the practical analysis of different SCADA protocols, tools and attack scenarios (including the attacker's perspective and a more extensive analysis of the PCOM protocol) that were performed as exploratory work.

4

Exploratory Analysis of the Security of SCADA Protocols

The previous chapter introduced the main building-blocks of the proposed approach, along with the discussion of related context – the CockpitCI and ATENA projects and the HEDVa testbed.

This chapter presents an exploratory analysis of the security of Supervisory Control And Data Acquisition (SCADA) protocols. This work was originally conducted on the Hybrid Environment for Development and Validation (HEDVa) testbed, as an essential part of the evaluation activities of CockpitCI and ATENA, and encompassed a large range of attack scenarios and different types of SCADA equipment and applications. Nevertheless, for sake of readability and conciseness, the content provided in this chapter is mostly focused on the electricity distribution grid (as detailed next) and on the Modbus and PCOM protocols.

In addition to the contributions to various project deliverables and internal reports, this work directly led to two research papers where the author of this thesis is the first and main contributor [Rosa et al., 2019] [Rosa et al., 2017] and to several open-source contributions, as enumerated in the Foreword of this thesis.

When it comes to the intrusion and anomaly detection process, *data* can be seen as the most significant component – thus, the importance of the *data-driven* focus within the proposed approach. For Industrial Automation and Control Systems (IACS), network communication protocols are one the most valuable data sources. SCADA protocols are used, for instance, to: actively control components from a local / remote site; continuously poll values of an autonomous process (to monitor its correct behavior); and connect different controllers and assets (e.g. PLCs and RTUs).

In that sense, this chapter focuses on network-based scenarios. More specifically, leveraging the HEDVa testbed, it starts by exploring the Modbus protocol – one of the SCADA protocols most commonly referred in the literature. Rather than exclusively describing its widely known vulnerabilities, in this thesis Modbus is analysed from a different perspective: how SCADA systems can be effectively exploited from a practical standpoint, to create incidents on electrical grid scenarios.

The second part of this chapter is devoted to the security of the PCOM protocol – as an example of a SCADA protocol found in many IACS but almost not covered in the literature from a security point of view. The choice of PCOM is opportunistic: it was used in the HEDVa testbed for some ancillary functions and accidentally discovered in network traffic captures, in the course of the exploratory work conducted from the attacker's perspective (originally focused on Modbus). After discovering PCOM traffic, the author of this thesis concluded that there were almost no previous analyses about the security of this SCADA protocol. Furthermore, only a couple of tools supported PCOM, making it difficult to understand what was transmitted over the network (Wireshark, for instance, did not support PCOM at all).

This motivated the author of this thesis to conduct a detailed assessment of PCOM, from a security standpoint, eventually also leading to several contributions to open source tools such as Wireshark, Snort, Metasploit, Nmap and Scapy. This assessment holds value by itself, but also as an example of how a motivated attacker can take advantage of less known SCADA protocols to conduct attacks on IACS.

Last but not least, this chapter extends the discussion of possible mitigation strategies for cyber-attacks taking advantage of SCADA protocols such as Modbus or PCOM, already started in Section 2.2, with a more detailed description of how to create protocol-specific rulesets for Network Intrusion Detection System (NIDS).

4.1 Attacking SCADA Systems: a Practical Perspective

The hereby presented attack experiments were based on a grey box penetration testing approach (i.e. the attacker knows a few, but not all the details of the target environment). With no surprise, due to the natural developments of CockpitCI and ATENA projects, the author of this thesis had some insights about the environment and the emulated processes, but was not aware of all the details (e.g. the usage of PCOM). Such grey box testing approach had the benefits of narrowing the scope of the experiments to the existing SCADA assets and vulnerabilities, rather than blindly explore other types of attack vectors (e.g. social engineering techniques, phishing campaigns, vulnerable internet-faced services and devices). Additionally, the pursued approach was also based on the Penetration Testing Execution Standard (PTES) methodology [Penetration Testing Execution Standard Group, 2019], which describes the penetration assessing activities as a set of incremental stages such as intelligence gathering, vulnerability analysis or the exploitation itself.

In this line, the attacker was granted access to the process control network (e.g. as a result of a compromised host), intentionally omitting the initial attack steps – since such initial steps, as discussed in Chapter 2, might result from all sorts of vulnerabilities and not necessarily a SCADA-specific issue (e.g. it might result from a thumb drive containing malware being inserted in the wrong device or from a compromised supply-chain [FireEye, 2020]). For practical purposes, and to represent the attacker (i.e. the compromised component), a dedicated host was deployed on HEDVa.

A three-stage attack strategy was devised, with the following phases: monitor the process values (to gain knowledge about the nature and characteristics of the controlled process), change them without being noticed in the SCADA Human-Machine Interface (HMI) consoles and, finally, induce service disruption on the electrical grid. This strategy resulted from the project objectives of showcasing how different types of detection mechanisms could be integrated within the energy grid use case. Moreover, these three stages represent the main steps an attacker can perform at process level (i.e. acquire field information, manipulate it and finally disrupt the service). Such strategy, primarily focused on network communications, also covers a large subset of SCADA specific cyber-attack scenarios – including, amongst others, TCP/IP network scans, Modbus specific scans, different variants of Denial of Service (DoS) attacks, and a SCADA-specific Man-in-the-middle (MitM) attack specifically customized for this process environment.

The remaining of this section is organized as follows. First, the HEDVa scenario is described in some detail. Next, the network reconnaissance experiments (i.e. discover assets and services) are discussed. Finally, an Address Resolution Protocol (ARP)-based MitM attack that fulfills the aforementioned goals of gaining knowledge about the process and inducing service disruption by means of intercepting and diverting the normal communication between SCADA servers and PLC devices is presented.

4.1.1 The HEDVa Use Case Scenario for Attack Implementation

The HEDVa distribution grid scenario, depicted in Figure 4.1, represented a medium/low voltage energy distribution scenario. Two energy feeders (reproducing the behavior and role of two redundant energy substations) and different points of the energy grid were emulated by a set of real Unitronics V130 PLCs. Each PLC corresponded to a given link/point of the grid (as illustrated in Figure 4.1) and was used to reproduce the real behavior of the electrical links (e.g. in the event of a failure in a given energy line, all the next PLCs in the same line would run out of power).

Moreover, each PLC offered the possibility of controlling the state of a circuit-breaker and exposed different emulated values, such as the current and voltage as measured in a real scenario. Those circuit breakers were used to connect/disconnect a given path from the remaining grid (e.g. due to maintenance operations or to restore the power supply from a backup supply line). Finally, through the usage of an HMI, the SCADA operators were able to monitor all the grid state, the values exposed by the PLCs and perform remote operations such as changing the state of the circuit-breakers.

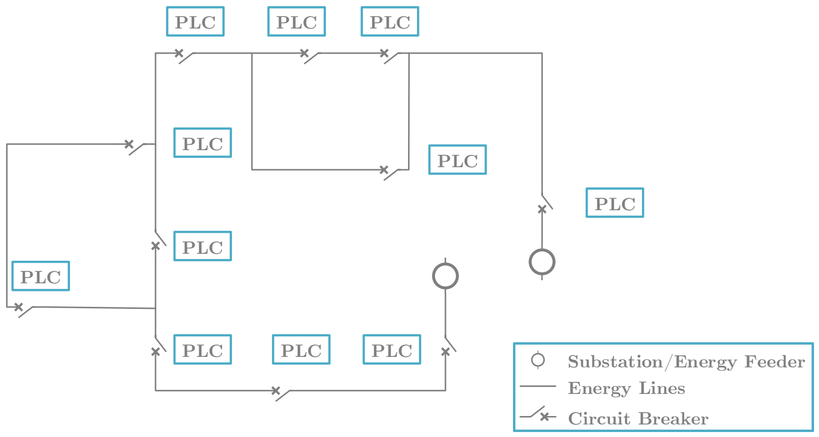


Figure 4.1: Conceptual view of the electrical distribution grid scenario.

Several HEDVa assets, including services, equipment (e.g. network switches, PLCs), servers (both physical and virtualized) and networks were also part of this use case. The PLCs and the remaining elements of the SCADA infrastructure in charge of the emulated grid were connected using an Ethernet LAN infrastructure (using VLAN segmentation for domain separation). Nevertheless, the attacks hereby described focus on a subset of HEDVa (cf. Figure 4.2) which includes two HMI hosts used for controlling and supervising the PLCs, an OPC SCADA server, a dedicated database for past events and offline analysis, and a NIDS offered means to detect the implemented attacks and offered the opportunity to analyse all the involved network traffic.

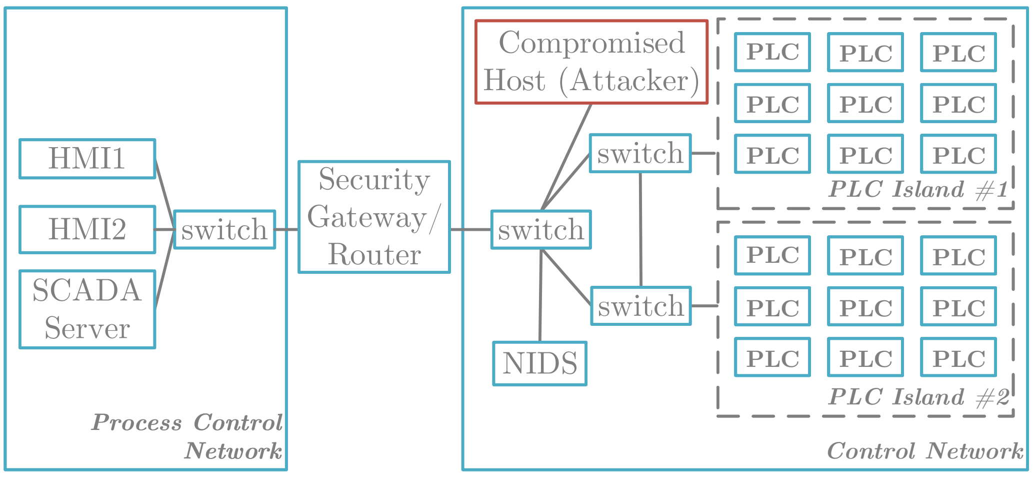


Figure 4.2: Network view of the components of the electrical distribution grid scenario.

4.1.2 Network Reconnaissance

Network scouting is one of the first steps of an attack, meant to gather information about all the components of the target environment, and to discover and identify topologies, hosts and services. For instance, traditional network components such as HMIs are identified by Internet Protocol (IP) and Medium Access Control (MAC) addresses, operating system versions and a set of services. In a first phase (cf. Figure 4.3, stage A), traditional network scanning techniques (e.g. FIN scans) are useful to identify specific components or software implementations and provide specific service footprints, together with TCP fingerprinting data.

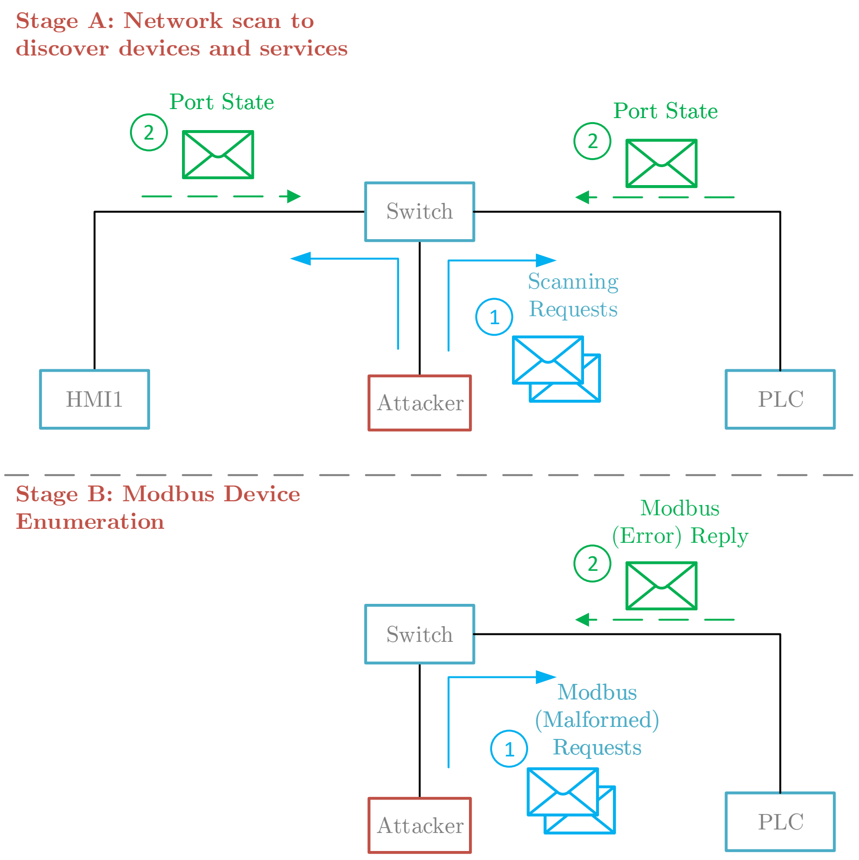

Stage A: Network scan to discover devices and services
2
Port State
2
Port State
Switch
Scanning Requests
1
HMI1
Attacker
PLC
Stage B: Modbus Device Enumeration
Modbus (Error) Reply
2
Switch
Modbus (Malformed) Requests
1
Attacker
PLC

Figure 4.3: Conceptual view of Network/Modbus scan with device enumeration.

For Modbus-based scenarios, each PLC is also identifiable and addressable by the `unitID` field, part of the Modbus frame (cf. Figure 4.3, stage B). For simple scenarios where one IP address corresponds to a single PLC, the `unitID` can be set to a fixed known value (typically "1") or may be "ignored" by the Modbus implementation. Nevertheless, a Modbus gateway, behind a single IP address, may hide several PLCs with different `unitIDs`. As part of an attack, a Modbus request with a wrong `unitID`, blindly used by an attacker, may be discarded or easily flagged with proper security mechanisms. Thus, for Modbus over TCP, it is critical to perform a Modbus enumeration on top of the traditional TCP/IP scans. Such scan consists of sending malformed Modbus packets for each open Modbus port, varying the value of the `unitID`. According to the standard behavior, this operation should get a return error message for each valid `unitID`, indicating the sent request was invalid. Note that, similar to the fingerprint methods, even if we don't have additional Modbus devices, this might be used to distinguish specific Modbus implementations that do not follow the standard. As other scanning techniques, Modbus scans need to be performed slowly, otherwise they can be easily flagged by any mechanism which spots abrupt deviations from the normal traffic flows.

Both types of scans are relevant and were used not only to discover devices and types of services, but also to perform fingerprinting and discovery of PLCs behind gateways. Figure 4.4 details the sequence diagram of both stages that were used in the HEDVa testbed for network reconnaissance, taking into consideration a FIN-based network scan and an additional Modbus enumeration stage.

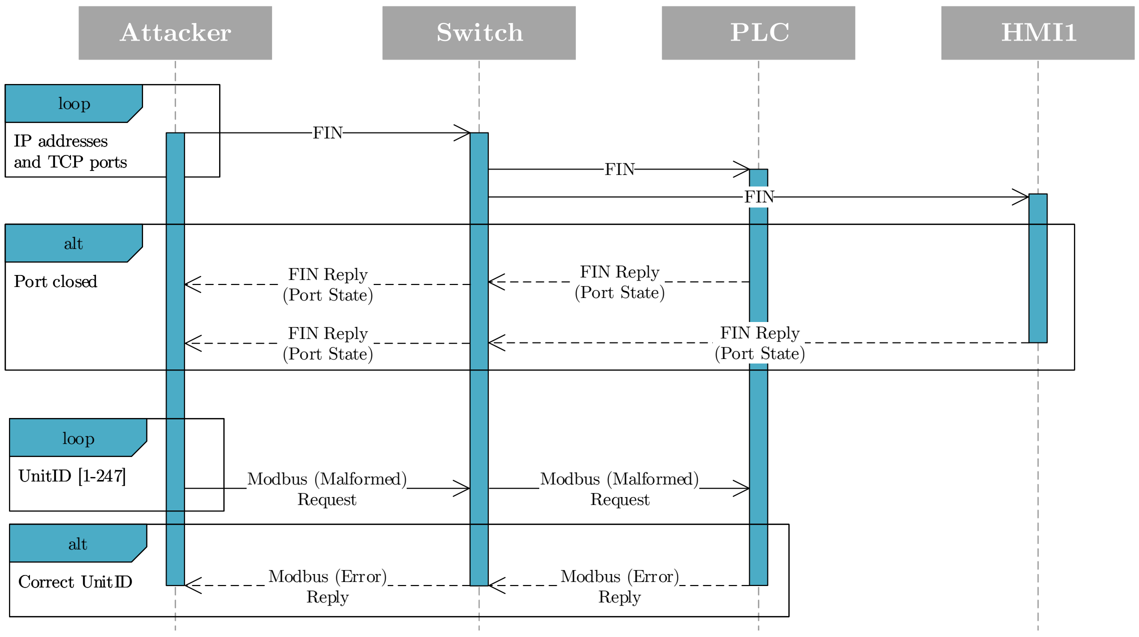


Figure 4.4: Sequence diagram of complete Network/Modbus scan with device enumeration

Network scouting provides a perspective on the target infrastructure from the network point-of-view, corresponding to the layers 2-4 of the Open Systems Interconnection (OSI) model. Despite its usefulness as a tool to identify and enumerate devices and services, it doesn't provide the process-level information required to implement process-level attacks. The next subsection presents one of the techniques that were used to obtain such information.

4.1.3 Using ARP poisoning to implement a MitM attack

The concept of a ARP poisoning MitM attack usually comprises two parts: ARP spoofing and communication hijacking. In the first stage, the goal is to spoof the ARP cache of both target devices, belonging to the same link, by sending malicious and unsolicited ARP "is-at" messages to the network to force both devices to send the packets through the attacker MAC address (cf. Figure 4.5, Stage A). This requires the attacker to know at least the IP and MAC addresses of the victims and the link they are connected to. As soon as the ARP cache of each victim is spoofed, the traffic gets redirected through the attacker.

In the second phase, when the traffic is already being redirected, the attacker can just read the messages and forward them (to acquire further process-level knowledge), or actively change them (cf. Figure 4.5, Stage B). Depending on the type of TCP connection, their payload and the actual data the attacker is interested in, the process may get complex. For persistent TCP connections, as opposed to one TCP connection per data request (Modbus can be implemented using both models), the attacker will need to keep the TCP fields consistent (e.g. sequence and acknowledgment numbers) and the connection open (e.g. TCP keep-alive packets). Moreover, in the case of Modbus, the requested values typically change in real-time and some of them are directly changed by the SCADA operator (e.g. Modbus writes). This means the attacker needs to not only keep track of all the interactions, but also compute and reproduce the effects in the physical process (e.g closing a circuit breaker in the electric path may change physical values such as current and voltage in other parts of the circuit). The complexity of this increases as the number of elements, relations and interdependencies increases.

For the implemented MitM scenario experiment, the objective of the attacker can be summarized as such: hijack the entire grid in such a way that the main HMI (HMI1) has no clue about the ongoing attack. Moreover, the attack objective should be accomplished by the attacker while going unnoticed. The sequence diagram of all involved steps is depicted in Figure 4.6. One of the first challenges faced by the attacker has to do with understanding the network topology and communication flows. For instance, the HMI1 host (one of the victims) is not part of the same network link as the PLCs, requiring the attacker to implement an

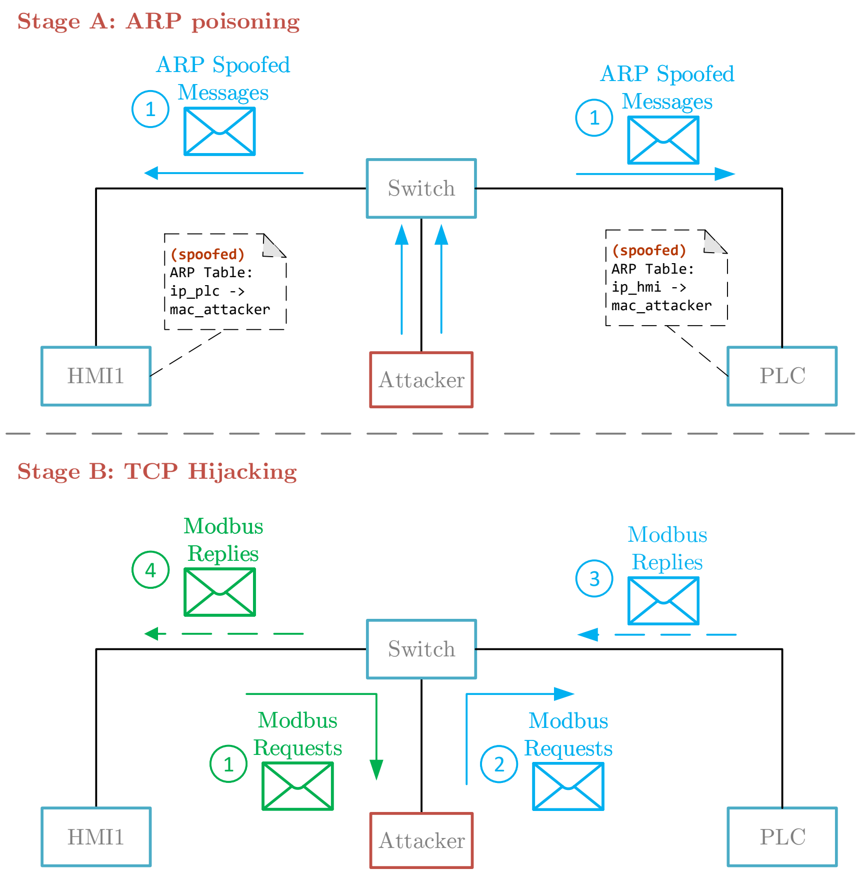


Figure 4.5: Conceptual view of a Modbus-based MitM using ARP poisoning

ARP spoof targeting the gateway interface of the network link where the attacker is placed, instead of the HMI1 (cf. Figure 4.6).

HMI1 uses TCP persistent connections to control several PLCs (11, to be more precise). Thus, the attacker needs to know how to handle or forward any spoofed packets in real-time, while avoiding TCP connection drops, to prevent any suspicious behaviour on the HMI console that could unveil his presence (cf. Figure 4.6). Packet drops automatically raise an alarm and change the view of the HMI for the corresponding PLC after a couple of seconds, indicating a potential issue. A TCP connection lost or a lack of a Modbus reply from the PLC is also visible from the HMI console. The second HMI did not use persistent connections. Later, during the trials, it was discovered that each PLC only supported a maximum of two simultaneous TCP connections. This may limit the way TCP connections are handled and redirected by the attacker.

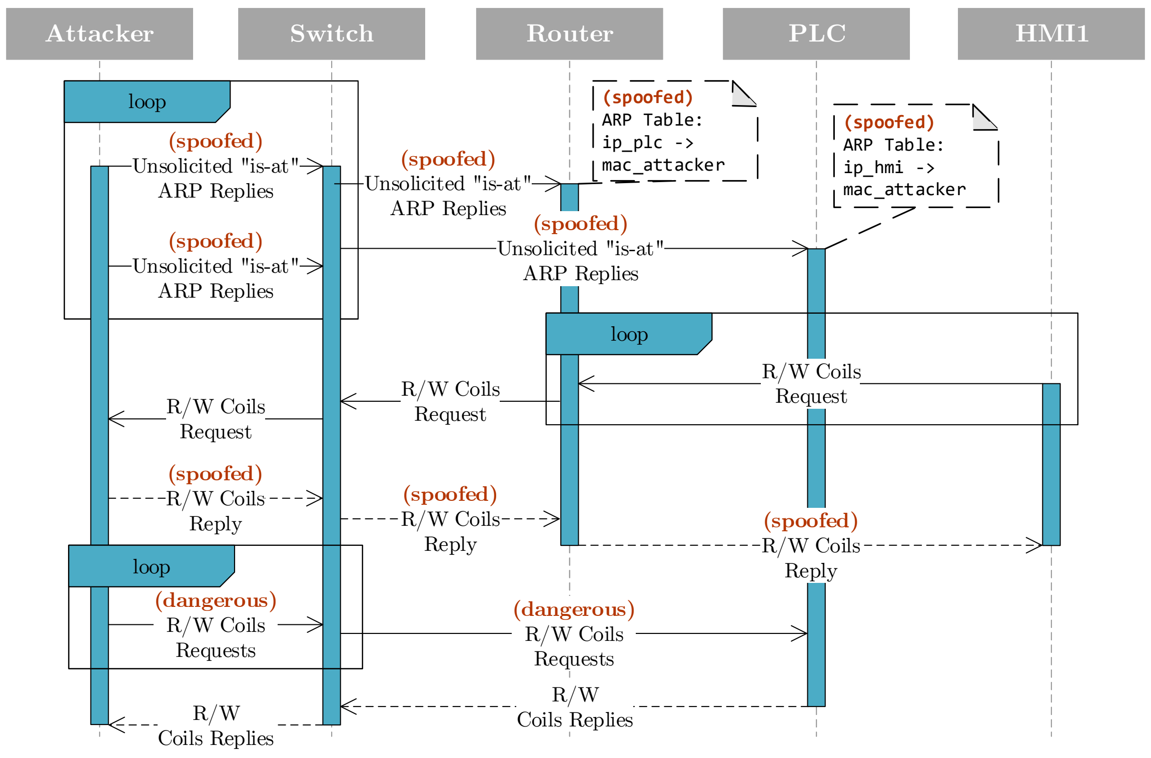

Attacker
Switch
Router
PLC
HMI1
loop
(spoofed) Unsolicited "is-at" ARP Replies
(spoofed) ARP Table: ip_plc -> mac_attacker
(spoofed) ARP Table: ip_hmi -> mac_attacker
R/W Coils Request
(spoofed) R/W Coils Reply
(dangerous) R/W Coils Requests
R/W Coils Replies

Figure 4.6: Sequence diagram of a Modbus-based MitM using ARP poisoning.

At first, the main concern was to place the attacker in the middle of the communication between the HMI1 and the PLCs to capture and analyze relevant process information. This allowed the attacker to gather more detailed information about the communications and the controlled process, learning how each Modbus register value affected the others (e.g. circuit breakers, current and voltage ranges).

Once the attacker was able to figure out the basic behavior of the controlled process, it was time to step up the challenge and hijack the entire process. This required forging the entire grid state in such a way that any HMI interaction produces a realistic state update, while decoupling HMI-PLC interactions. For this purpose, the attacker needs to reply to the Modbus requests in real-time. Moreover, TCP session hijacking requires the attacker to maintain the integrity of the TCP connection (such as TCP sequence numbers) to avoid connection drops.

The next task was crafting the Modbus frames and creating a fake view of the entire scenario in real-time. An application was developed for this purpose, on the top of Scapy framework [Biondi, 2020] – since common open-source tools normally used for this sort of attacks were not SCADA/Modbus-aware and did not fulfill the project needs, either not offering an integrated solution for all the steps or lacking the flexibility to adjust settings to the HEDVa scenario. After the ARP spoofing, the attacker first starts by capturing the current state of the grid. This is achieved by dumping and decoding one complete interaction cycle (i.e. the set of Modbus request-reply transactions) between the HMI1 and all PLCs. This represents the initial state of the simulated view and allows to restore the previous grid state after stopping the attack (in case the attacker wants to do so). The attacker must also perform deep inspection of each packet and selectively intercept all the TCP connections from HMI1 to the PLCs, while forwarding the others.

When requests from HMI1 are received, the attacker must compute the responses (based on its own replica of the model, obtained during the process analysis stage). This effectively decouples the HMI1 from the PLCs, creating two distinct communication flows: one between the HMI1 and the attacker and the other between the attacker and each PLC. This allows not only to hijack the data exchanged between them, but also to trigger any kind of service disruption against the PLCs compromising the physical process behind them.

Since the true state of the PLCs is hidden from HMI1, the attacker is free to do whatever he wants without the knowledge of the SCADA operator. Moreover, all the changes performed by the SCADA operator (such as opening or closing a breaker) are properly intercepted and handled by the attacker. Finally, whenever the attacker decides to stop the attack, he only needs to perform the inverse of the first steps, dumping the values of the simulated HMI1 view to the PLCs and restoring the ARP caches by sending additional unsolicited ARP replies with the correct associations between MAC and IP addresses.

4.2 A Security Analysis of the PCOM Protocol

The previous section described attack scenarios focused on the Modbus protocol. This section provides a more detailed analysis of the PCOM protocol. As already mentioned, PCOM was chosen both due to opportunistic reasons (it was available in the HEDVa testbed) and because it provides a good example of the security vulnerabilities of SCADA protocols used in many IACS but still not well known from a security point of view.

This is relevant because there are many real-world IACS using protocols like PCOM (for the whole process management or just for ancillary functions), not well known and not supported by security tools such as NIDS but still relatively easy to take advantage of to conduct attacks, as shown in the section.

The remaining of this section is organized as follows. First, it provides a brief description of the PCOM protocol (Subsection 4.2.1). Then, it discusses how to build a dissector for PCOM messages, an indispensable step for interpreting the PCOM messages captured from the network (Subsection 4.2.2). Next, the scenario used to conduct the PCOM experiments is presented (Subsection 4.2.3), followed by the discussion of various PCOM-based attacks – network scouting, accessing sensitive data, DoS attacks, rogue PLC reprogramming and protocol fuzzing (Subsections 4.2.4-8). Finally, the generated PCOM datasets are presented (Subsection 4.2.9).

4.2.1 PCOM Primer

There are few applications that support the PCOM protocol, with the two most notable exceptions being Visilogic [Unitronics, Inc, 2019b] and Crimson [Red Lion, 2019], both for Microsoft Windows. Crimson only allows reading/writing registers from/to PCOM-enabled devices. Visilogic allows additional administrative remote operations such as starting/stopping PLCs, but is not suitable for discovering devices on the network and cannot be used for detecting or blocking PCOM traffic in the network. A .Net driver that implements some additional internal functions, which are not part of the original PCOM specification [Unitronics, Inc, 2014], is also available [Unitronics, Inc, 2019a]. However, it lacks functions such as downloading/uploading ladder logic to/from PLCs or device enumeration. More recently, a basic Python implementation of PCOM for supporting the development of PCOM-enabled applications has been released [Theriault, 2019].

PCOM is based on requests and responses using command codes. Such codes identify the type of operation (e.g. reading a Memory Integer (`MI`) operand). This way, it can be used to continuously poll (or change) the values of a given set of PLC registers (e.g. process monitoring values), as well as for implementing other remote administrative interfaces.

PCOM communications may take place on top of different physical layers and field buses, including Controller Area Network (CAN) bus, RS-485 or Ethernet. It is also possible to have inter-PLC communication in master-slave schemes, where the master PLC acts as a bridge, forwarding all the requests and replies to/from the slave PLCs. A Unit ID field is used to uniquely identify a device on a network. For Ethernet networks, a special zero Unit ID value indicates a direct connection. Moreover, in such Ethernet networks, PCOM works on top of Transmission Control Protocol (TCP) sessions by adding an extra 6 bytes header between the TCP header and the original PCOM messages (PCOM/TCP). The protocol also supports two different modes, *ASCII* and *Binary*, hereafter referred to as PCOM/ASCII or PCOM/Binary. Figure 4.7 illustrates the structure of both modes. In PCOM/TCP, the communication socket in the PLC side defaults to TCP port 20256.

PCOM/ASCII allows reading and writing not just memory addresses (which are usually mapped from inputs and outputs), but also other types of operands and reserved values, such as System Bits (`SB`), System Integers (`SI`) and System Longs (`SL`). PCOM/Binary allows composed requests such as querying more than one type of operand in the same packet, using multiple data request blocks – opposed to only one type of operand per request in PCOM/ASCII. It also allows reading and writing PLC Data Tables – an operation not supported in the PCOM/ASCII mode.

PCOM capabilities go beyond reading and writing values. Remote administrative operations via specific command codes and parameters (e.g. reset a PLC, set the Real-time Clock (RTC) value) are also possible and can even be used to reprogram the entire ladder logic of a PLC. Nevertheless, not all PLC models support PCOM/Binary or even PCOM/TCP. Specialized applications such as Visilogic use PCOM to access and manage field devices such as PLCs.

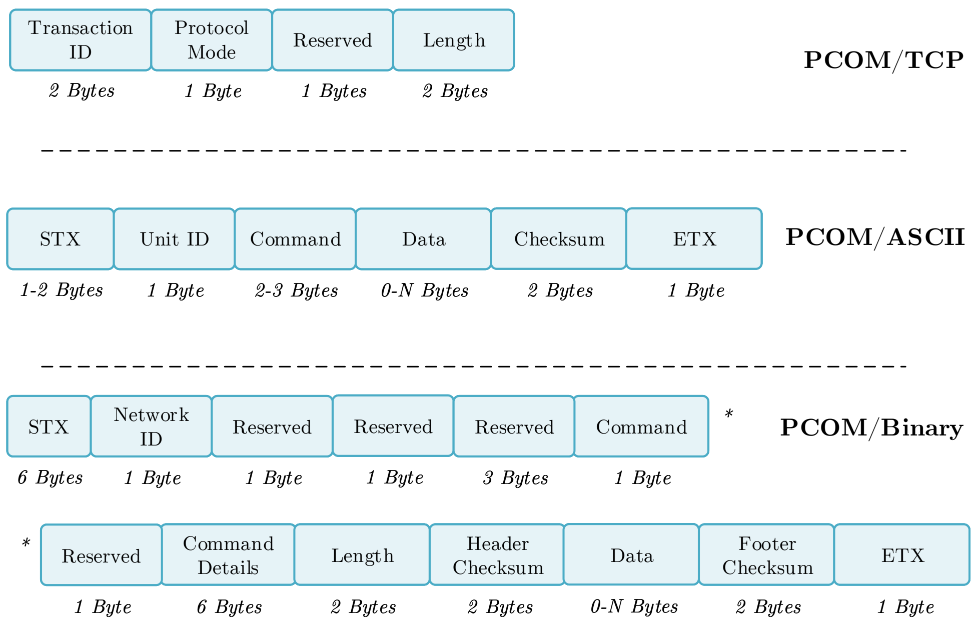


Figure 4.7: PCOM/TCP, PCOM/ASCII and PCOM/Binary protocol structure (based on [Unitronics, Inc, 2014])

4.2.2 Building a Dissector for PCOM Messages

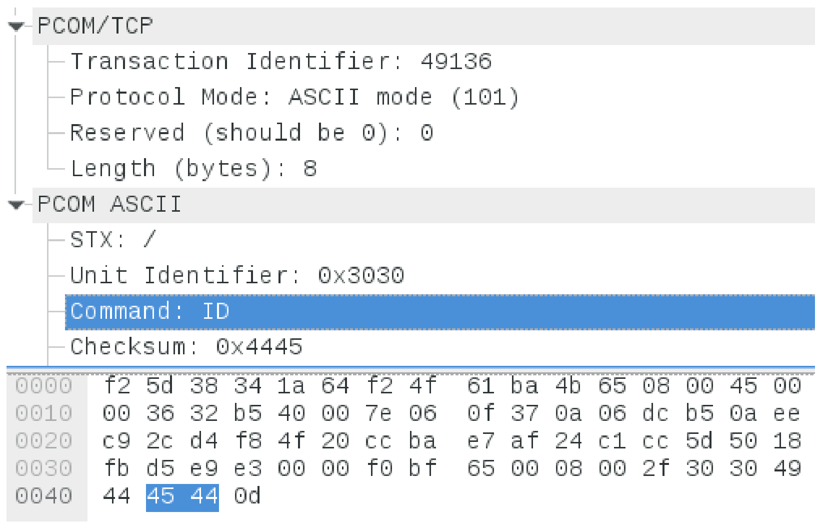
In order to develop an application, troubleshoot an erroneous behavior or even reverse-engineer the PCOM protocol to better understand the underlying communication, it is necessary to have a way of quickly parsing and analyzing PCOM messages. This dissection is typically done using a network packet analyser such as Wireshark. Since there were no publicly available tools able to perform the dissection of PCOM messages, in the scope of this thesis a Wireshark built-in PCOM/TCP dissector was developed – and later used to support the development and validation steps by helping to flag malformed packets and revealing undocumented features. Next, some details of the implementation of this PCOM dissector are discussed.

As already mentioned, PCOM supports two modes: ASCII and Binary. These modes are distinguished based on the value of the third byte of the PCOM message. Distinguishing between PCOM/ASCII and PCOM/Binary is necessary to automatically handle and decode different field formats and endianness (e.g. the same Unit ID value is specified differently in PCOM/ASCII and PCOM/Binary) or to compute and validate checksums (since requests containing a bad checksum are automatically discarded by the PLC). Another example: in PCOM/ASCII the command code of PCOM responses is truncated to the first two chars, whereas in PCOM/Binary the same operation uses a different code, depending on whether it is a request or a response – the `\x80` value is added to the command code value in replies.

A PCOM dissector may also help to detect hidden features. For instance, based on the analysis of captured PCOM messages, it was observed that the 9$^{\text{th}}$ to 11$^{\text{th}}$ bytes of PCOM/Binary messages, which according to the specification [Unitronics, Inc, 2014] should always be zero, are in fact not always zero. In PCOM/Binary messages with command code 41 (a request to a PLC, part of a multi-message operation, cf. Section 4.2.3) those bytes start from zero but are incremented by one at each message. Conducted observations also showed that other reserved fields from the PCOM/Binary structure are sometimes used to specify data and not always contain the values described in the specification.

The PCOM dissector can interpret the PCOM/TCP header, as well as the header structure for both PCOM/ASCII and PCOM/Binary modes. It can also translate over 25 PCOM command codes into meaningful descriptions and dissect the details of PCOM/ASCII read and write contents of different operands (i.e. the PLC registers). Some dissector fields were also defined as *expert fields* [Wireshark Foundation, 2019], in order to easily spot protocol violations, based on the same concept Wireshark uses for flagging TCP anomalies by analysing sequence and acknowledgment numbers. The code of this dissector has been integrated into the upstream Wireshark repository [Rosa, 2019f] [Rosa, 2019e] [Rosa, 2019g].

Developing this built-in dissector for Wireshark provided an interface to visualize the flows of PCOM messages, their structure and their content. Figure 4.8 provides a snapshot of a PCOM packet dissection based on this tool.



```
PCOM/TCP
  Transaction Identifier: 49136
  Protocol Mode: ASCII mode (101)
  Reserved (should be 0): 0
  Length (bytes): 8
PCOM ASCII
  STX: /
  Unit Identifier: 0x3030
  Command: ID
  Checksum: 0x4445
0000  f2 5d 38 34 1a 64 f2 4f  61 ba 4b 65 08 00 45 00
0010  00 36 32 b5 40 00 7e 06  0f 37 0a 06 dc b5 0a ee
0020  c9 2c d4 f8 4f 20 cc ba  e7 af 24 c1 cc 5d 50 18
0030  fb d5 e9 e3 00 00 f0 bf  65 00 08 00 2f 30 30 49
0040  44 45 44 0d
```

Figure 4.8: Snapshot of a PCOM command packet using the Wireshark

From a security perspective, a protocol dissector helps understanding policy violations such as abnormal and unauthorized messages (e.g. reflecting malicious network scouting or even PLC reprogramming attempts). It also becomes possible to use specific PCOM filters, such as `pcomascii.command == "RC"`, to search and narrow the flow of PCOM communications.

In the field of forensics, such a dissector helps gathering, filtering and reconstructing evidence from network attacks. For instance, when programming a PLC, a set of specific PCOM commands (as described with detail in Section 4.2.3) is used to transfer files between an application and the PLC. The PCOM dissector enables quick detection by searching for file signatures, using Wireshark filters (e.g. `pcombinary.data contains "\x50\x4b\x03\x04"` for a PKZIP file format) and then reconstructing the file by looking at the content between the signature and the end of file. Generic tools to recover files from network traces, using TCP or Hypertext Transfer Protocol (HTTP) payloads, would fail, since the final file results from a concatenation of the bytes of a specific PCOM field of a specific PCOM operation across multiple packets rather the entire TCP or HTTP payload.

4.2.3 Reference Scenario Used for PCOM-based Experiments

Real-world attacks often depend on specific details of the target system. For the sake of simplicity, this section adopts a reference validation scenario which is simple enough to be quickly understood, while still being fairly representative of larger and more complex scenarios.

Rather than focusing on a particular SCADA process, as the Modbus scenario, the scenario adopted for discussing PCOM security intends to demonstrate specific attack use cases which take advantage of PCOM's characteristics, including for instance device scanning techniques, the possibility of accessing all the PLC data using unauthorized requests, how to disrupt the physical process under control via remote commands, or how to reprogram the PLC to introduce subtle changes that might only be perceived in the long term.

Figure 4.9 illustrates the reference scenario used to analyse attacks, to develop the ancillary tools, and to collect the respective datasets. Two different physical layers are used: an Ethernet segment and a CAN [Bosch et al., 1991] bus. The scenario is composed of two Unitronics V130 PLCs, one of them connected only in the CAN bus and the other deployed in both segments, acting as a bridge. An HMI with Visilogic 9.8.65 plays the role of an authorized device/operator. Finally, it is assumed that the attacker has access to a compromised device in the same network of the HMI and the PLC Master – for instance, an operator workstation, a historian database or a network printer able to reach the field network segment.

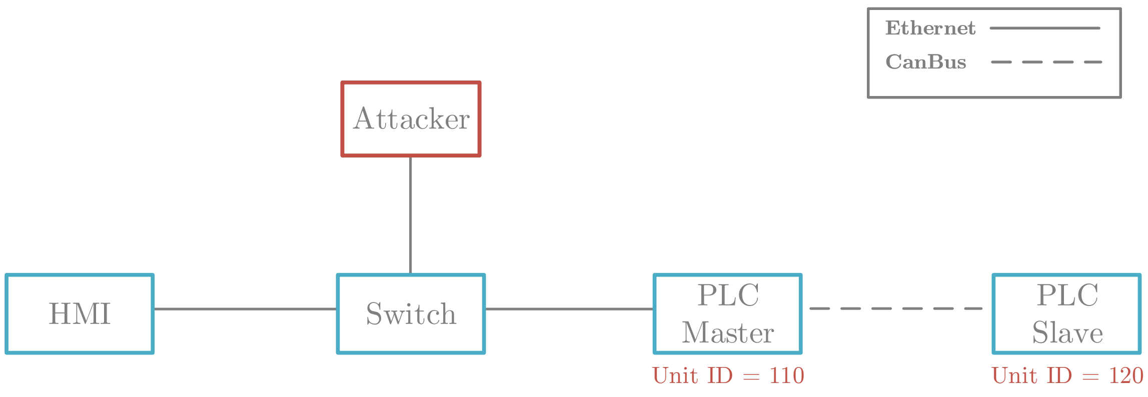


Figure 4.9: PCOM reference scenario for the validation tests.

This scenario is focused only on PCOM network communications between HMIs and PLCs (i.e. corresponding to levels 0, 1 and 2, according to IEC 62443 standard), and no other protocol or service vulnerabilities are addressed. Therefore, no assumptions are made on how the attacker compromised that device. Incident records show that getting control of such devices is fairly common as an intermediate step of successful attacks.

The Master PLC runs a minimal working ladder logic example where two sockets were initialized, one for PCOM/TCP and the other for Modbus. A single ladder `OR` element (cf. Figure 4.10), representing the process logic, was used to compute the input of two `MIs` (`MI1` and `MI2`), into a third one `MI3`, representing the field data.

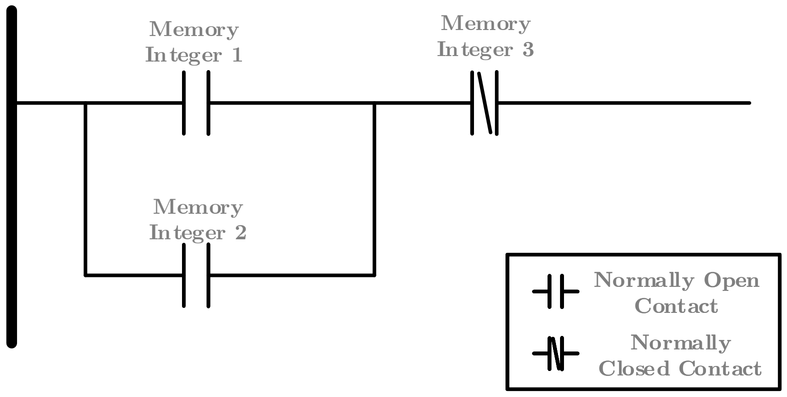


Figure 4.10: Ladder Logic encoding/representation of the logical process for the PCOM scenario.

In a real SCADA system, such PLC registers may have different interpretations, from circuit switch states on an electrical distribution grid to reservoir levels in water treatment systems. Even though in real SCADA environments the number of used registers is larger and significantly more complex functional logic is expected to be found, this reference scenario is representative enough to verify whether we can access and/or change register values or change process logic.

Given the lack of authentication, authorization and encryption, PCOM-based systems are vulnerable at least to two main classes of attacks: direct connection and session hijacking attacks. The following subsections discuss how each of the security issues of PCOM might be explored by a malicious attacker in the context of a SCADA system.

4.2.4 Network Scouting

As mentioned before, information gathering is one of the first steps for understanding the environment. Structured cyber-attacks usually start by collecting as much information as possible about target systems. Typical network scans such as SYN Scans are good at identifying hosts on a network, but they fail to identify specific host details. Therefore, they must be complemented with fingerprinting checks and additional scripts to collect specific host characteristics.

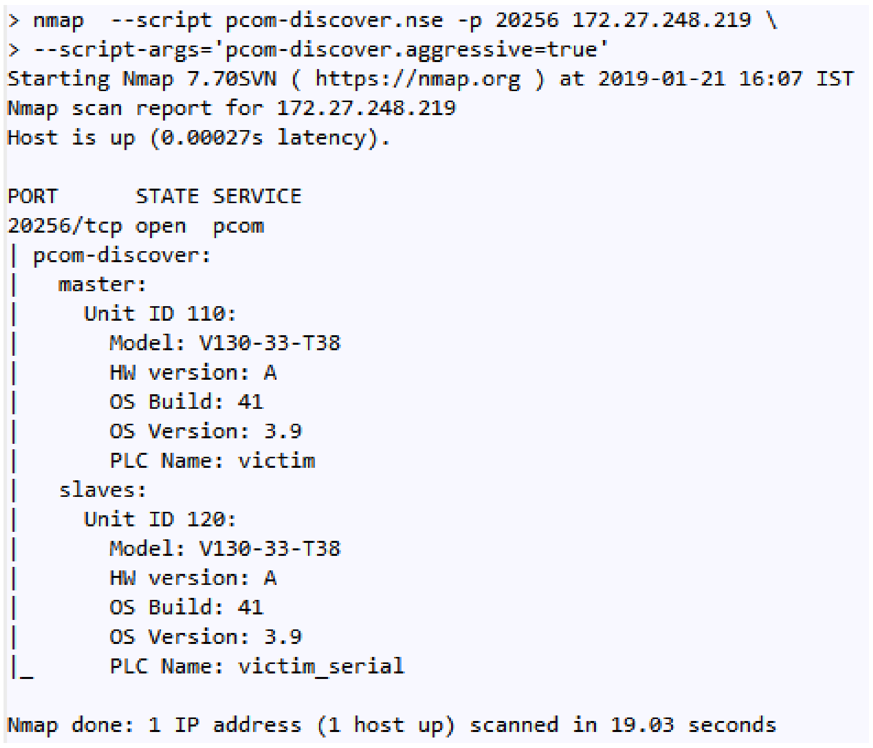
The PCOM protocol allows unauthenticated queries to PLCs that can be used to retrieve, among others, the PLC model, the hardware version, the Operating System (OS) build and OS version, the PLC name and the UnitID value. In the scope of this research, a Nmap Scripting Engine (NSE) script to collect such detailed information using PCOM/ASCII and PCOM/Binary commands was developed [Rosa, 2019i]. Using a Nmap script eliminates the need for developing and packaging an entire application from scratch. Moreover, using Nmap brings convenient added features such as portability, a powerful command-line interface and the possibility of integration with other tool-chains, on a single, well maintained open-source tool that can be used to assess multiple SCADA devices from different vendors, regardless of the communication protocol.

Algorithm 1 PCOM/TCP Network Scan

```
1: $unitId \leftarrow 0$
2: $master_t \leftarrow$ getUnitID($unitId$)                ▷ PCOM/ASCII
3: $master_t \leftarrow$ getID($unitId$)                    ▷ PCOM/ASCII
4: $master_t \leftarrow$ getPlcName($unitId$)               ▷ PCOM/Binary
5: if $aggressiveMode$ then
6:     for $unitId \leftarrow 1, 127$ do
7:         if $unitID \neq master_{id}$ then
8:             $slaves_t \leftarrow$ getID($unitId$)
9:             $slaves_t \leftarrow$ getPlcName($unitId$)
10:        end if
11:    end for
12: end if
```

Figure 4.11 shows the output of a scan using the aforementioned PCOM Nmap NSE script on the reference scenario. The script sends PCOM requests with command code `ID` and the UnitID field set to `00`, exploring two weaknesses: (1) no authentication is required to communicate with a PLC and, (2) the use of UnitID `00` will make any connected PLC respond, no matter with which UnitID it was configured. The PCOM/ASCII command code `UG` is then used to retrieve the actual PLC Unit ID, and PCOM/Binary command value `0x0C` is used to retrieve the PLC name. Algorithm 1 shows the simplified pseudo-code behind the aforementioned scan script.

As a technical remark, it should be noted that, at each step of the loop, the script needs



```
> nmap  --script pcom-discover.nse -p 20256 172.27.248.219 \
> --script-args='pcom-discover.aggressive=true'
Starting Nmap 7.70SVN ( https://nmap.org ) at 2019-01-21 16:07 IST
Nmap scan report for 172.27.248.219
Host is up (0.00027s latency).

PORT      STATE SERVICE
20256/tcp open  pcom
| pcom-discover:
|   master:
|     Unit ID 110:
|       Model: V130-33-T38
|       HW version: A
|       OS Build: 41
|       OS Version: 3.9
|       PLC Name: victim
|   slaves:
|     Unit ID 120:
|       Model: V130-33-T38
|       HW version: A
|       OS Build: 41
|       OS Version: 3.9
|_      PLC Name: victim_serial

Nmap done: 1 IP address (1 host up) scanned in 19.03 seconds
```

Figure 4.11: Snapshot of the Nmap output using the PCOM Nmap NSE script

to adjust to the fact that even though the PLC replies with a TCP acknowledgment to all received requests (including those for non-existent UnitID's), it does not send any PCOM reply if it doesn't hold the matching UnitID. It is therefore necessary to iterate across the whole range of possible UnitID values, which is fairly short anyway (1 to 127). Moreover, if several requests are sent in a burst they might be rejected, therefore it is convenient to adjust the rate of this interaction.

The information gathered this way allows an attacker to lookup for potential vulnerabilities and exploits applicable only to certain models or firmware versions. It may be used, for instance, to identify which commands a specific PLC may accept – according to the PCOM specification [Unitronics, Inc, 2014] not all models support all PCOM modes or operands.

The approach for developing the aforementioned script can be replicated for other protocols. For instance, following a similar approach, a Nmap NSE script [Rosa, 2019h] was developed to collect information from Allen-Bradley Logix 5000 controllers (part of the HEDVa testbed – cf. Chapter 3). Such script relies on the Common Industrial Protocol (CIP) protocol and a known CIP service code (`0x55`), part of the official documentation [Allen-Bradley, 2018], to retrieve a list of service tags stored in the PLC. Opposed to other protocols, where the data is queried through a memory address (e.g. Modbus), using these service tags (which can be seen as a list of variables names) can provide additional hints about the nature of the PLC process. This scan script was also submitted to the upstream Nmap code repository.

4.2.5 Accessing Sensitive Data

PCOM is vulnerable to all sorts of layer 2 and layer 3 attacks, and such vulnerabilities can be used to access all the sensitive data held by a PLC. Figure 4.12 illustrates two possible classes of attacks against PCOM, discussed next.

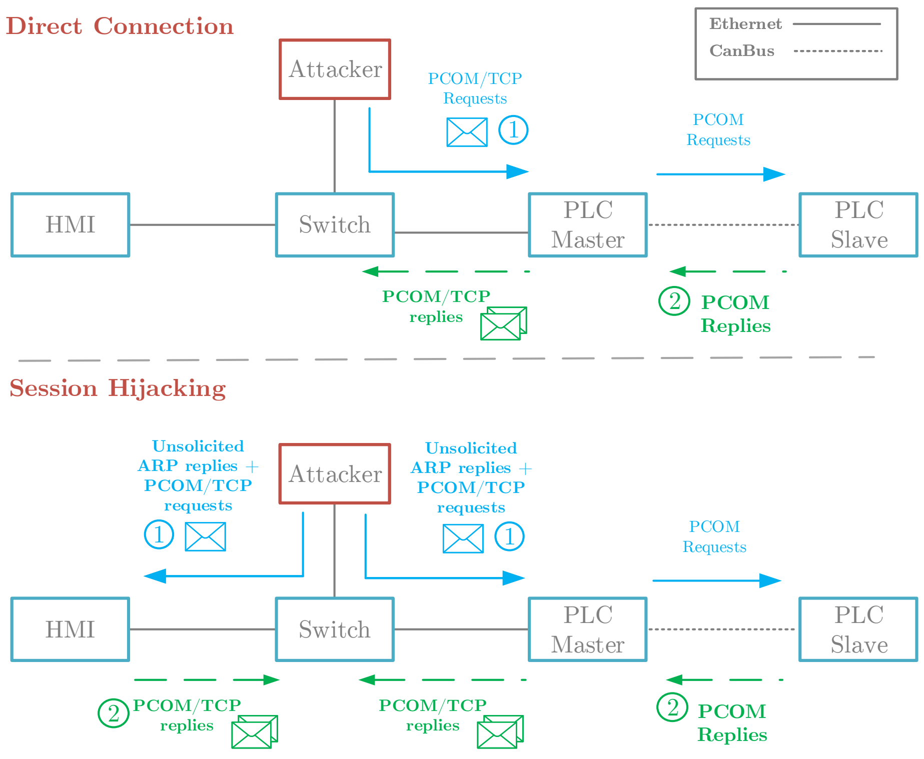


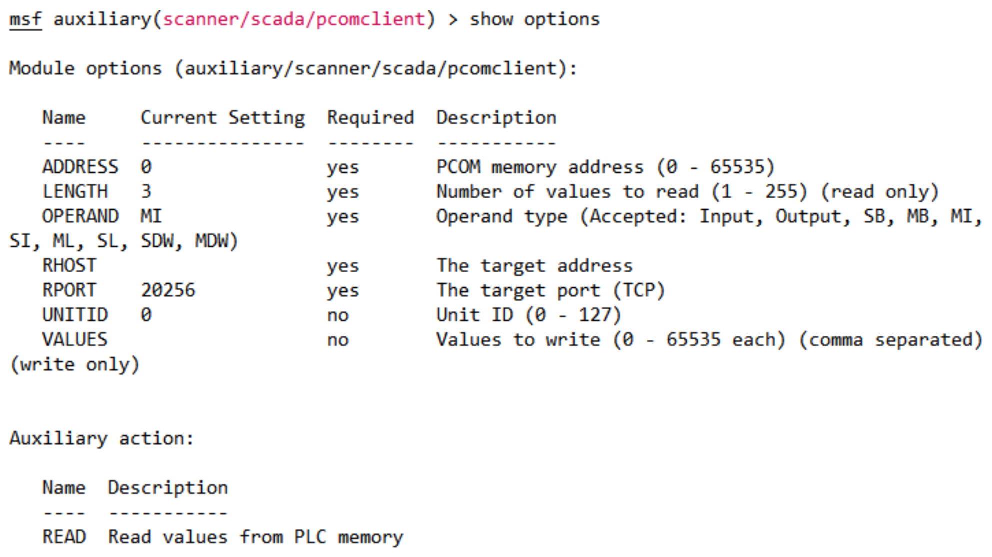
Figure 4.12: Direct TCP Connection and Session Hijacking Attacks

Direct TCP connections can be used to query not only process-related values but also all types of operands, without restriction, including inputs, outputs, System Bits (SIs), Memory Integers (MIs), etc. This allows to access values that are related to the process under control, to change configuration parameters (such as the network settings, in `SI[101-148]`, or the info mode password, in `SI[253]`), or even to disrupt the PLC operation by setting system registers (e.g. `SB[314]` which can be used to block the communications between legitimate nodes and the PLC.

Complex main-in-the-middle (MITM) scenarios where a PCOM/TCP session is intercepted and redirected via an attacker-compromised device are also possible, for instance, by means of an ARP poisoning attack against the communication endpoints. Such attacks are not specifically related to PCOM, but can be used against PCOM communications due to their lack of confidentially and integrity protection.

Moreover, and similarly to what was discussed in the network scouting section, an attacker positioned in an Ethernet segment can also reach the PLC in the CAN bus via the master PLC, simply by specifying its Unit ID in the PCOM/TCP requests.

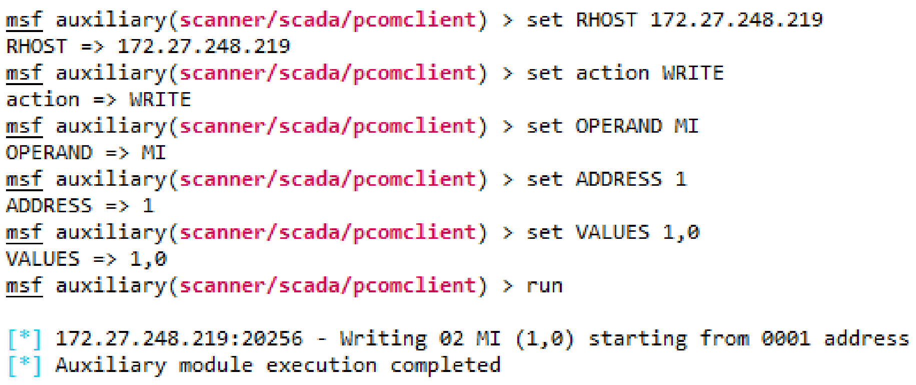
In the course of this work, a Metasploit auxiliary module was developed for reading and writing PLC registers by selecting the Unit ID, the operand type, the address and the number of values to read (or by providing the values to write) [Rosa, 2019a]. This module supports the operands `inputs`, `outputs`, `SBs`, `MBs`, `MIs`, `SIs`, `MDW`, `SDW`, `MLs` and `SLs`. Other commands are also supported by providing the raw hexadecimal of the PCOM/ASCII data payload, with the other fields being automatically computed – including the PCOM/ASCII checksum and the PCOM/TCP header. Figure 4.13 shows the main options of the Metasploit PCOM module.



```
msf auxiliary(scanner/scada/pcomclient) > show options

Module options (auxiliary/scanner/scada/pcomclient):

   Name     Current Setting  Required  Description
   ----     ---------------  --------  -----------
   ADDRESS  0                yes       PCOM memory address (0 - 65535)
   LENGTH   3                yes       Number of values to read (1 - 255) (read only)
   OPERAND  MI               yes       Operand type (Accepted: Input, Output, SB, MB, MI,
SI, ML, SL, SDW, MDW)
   RHOST                     yes       The target address
   RPORT    20256            yes       The target port (TCP)
   UNITID   0                no        Unit ID (0 - 127)
   VALUES                    no        Values to write (0 - 65535 each) (comma separated)
(write only)


Auxiliary action:

   Name  Description
   ----  -----------
   READ  Read values from PLC memory
```

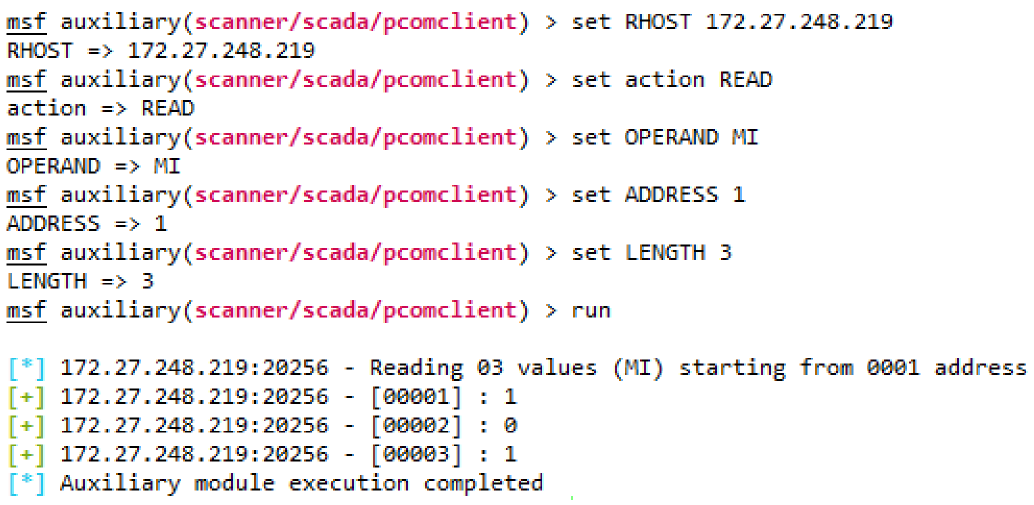
Figure 4.13: Snapshot of metasploit auxiliary pcom client options

Figure 4.14 and Figure 4.15 illustrate the usage of the module to write (and, later, to read) the process values of the reference scenario via PCOM/ASCII requests. In the write operation it is necessary to specify the operand type, the starting address and the values for the first two registers (the third one is always rewritten according to the `OR` operation). In the read mode it is necessary to specify the number of registers to read from the starting address.



```
msf auxiliary(scanner/scada/pcomclient) > set RHOST 172.27.248.219
RHOST => 172.27.248.219
msf auxiliary(scanner/scada/pcomclient) > set action WRITE
action => WRITE
msf auxiliary(scanner/scada/pcomclient) > set OPERAND MI
OPERAND => MI
msf auxiliary(scanner/scada/pcomclient) > set ADDRESS 1
ADDRESS => 1
msf auxiliary(scanner/scada/pcomclient) > set VALUES 1,0
VALUES => 1,0
msf auxiliary(scanner/scada/pcomclient) > run

[*] 172.27.248.219:20256 - Writing 02 MI (1,0) starting from 0001 address
[*] Auxiliary module execution completed
```

Figure 4.14: Writing registers of the reference scenario PLC using Metasploit PCOM client



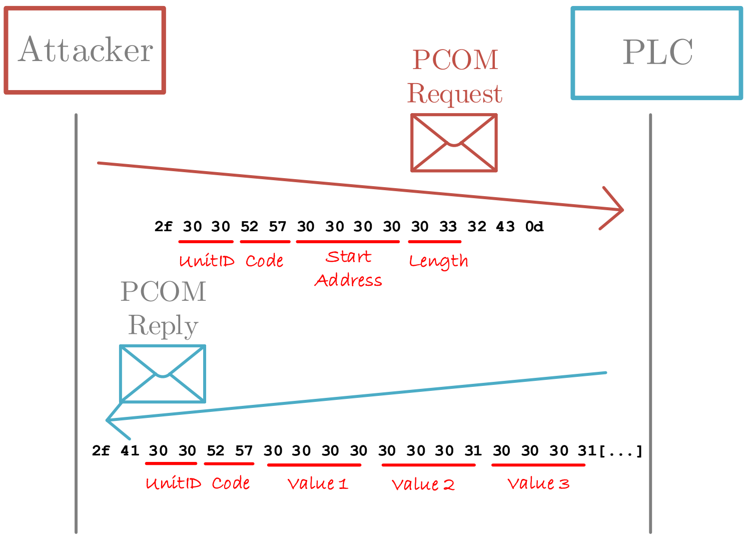
```
msf auxiliary(scanner/scada/pcomclient) > set RHOST 172.27.248.219
RHOST => 172.27.248.219
msf auxiliary(scanner/scada/pcomclient) > set action READ
action => READ
msf auxiliary(scanner/scada/pcomclient) > set OPERAND MI
OPERAND => MI
msf auxiliary(scanner/scada/pcomclient) > set ADDRESS 1
ADDRESS => 1
msf auxiliary(scanner/scada/pcomclient) > set LENGTH 3
LENGTH => 3
msf auxiliary(scanner/scada/pcomclient) > run

[*] 172.27.248.219:20256 - Reading 03 values (MI) starting from 0001 address
[+] 172.27.248.219:20256 - [00001] : 1
[+] 172.27.248.219:20256 - [00002] : 0
[+] 172.27.248.219:20256 - [00003] : 1
[*] Auxiliary module execution completed
```

Figure 4.15: Reading registers of the reference scenario PLC using Metasploit PCOM client

Figure 4.16 depicts the PCOM/ASCII request and reply messages used to read the three `MI` PLC operands of the reference scenario.

4.2.6 Denial of Service (DoS) attacks

Responsible pentesting procedures for IACS [Stouffer et al., 2015] often recommend special caution when handling PLCs. This is due to the fact that it is not uncommon to find devices which are vulnerable against situations that may or not be intentionally triggered, such as single packets triggering input handling bugs (eventually generated by device fingerprinting



```
2f 30 30 52 57 30 30 30 30 30 33 32 43 0d
2f 41 30 30 52 57 30 30 30 30 30 30 30 31 30 30 30 31[...]
```

Figure 4.16: Example of PCOM/ASCII request and reply in the reference scenario

procedures) or brute force attacks causing resource exhaustion (such as SYN floods). This may lead to DoS scenarios.

Such DoS attacks are especially relevant in the Critical Infrastructure domain, due to their potential impact. An IACS-targeted DoS may have disastrous consequences, ranging from service or production interruptions (availability is considered a topmost priority for most automation infrastructures) to physical damage or even loss of human life [Cruz et al., 2016].

By analysing the network traffic generated by PCOM applications such as [Unitronics, Inc, 2019b] [Red Lion, 2019], it was possible to identify a series of reserved commands used for PLC administrative purposes, which are not part of the publicly documented command set [Unitronics, Inc, 2014]. When used for their legitimate purpose, these commands provide a convenient way to remotely control a PLC, allowing an operator to recover it from a failure or to reset the device after a reprogramming operation.

However, due to the lack of authentication and authorization mechanisms, an attacker can abuse administrative commands for malicious purposes. In particular, the `STOP` or the `RESET` commands can be used to prevent a PLC from communicating with the HMI or other PLCs. Other attacks are also possible. For instance, an attacker may hijack a TCP connection to block `RESET` or `START` operations triggered by the legitimate operator. In this case, on top of traditional ARP poisoning, the attacker should acknowledge such PCOM/TCP requests so the sender believes they were successful while preventing them from reaching the PLC.

In order to study these vulnerabilities, another Metasploit auxiliary module was developed to send PCOM/ASCII administrative commands such as start, stop and reset operations [Rosa, 2019b]. While these non-documented commands were discovered and explored in PCOM/ASCII mode, research showed no evidence of similar commands in the PCOM/Binary mode.

4.2.7 Reprogramming the PLC

Another attack strategy is to reprogram the PLC, in order to modify its behaviour, for instance to induce permanent damage to the physical equipment or process under control. A skilled attacker may use this technique as part of a long-term strategy, by introducing subtle changes to the process control tasks that can go unnoticed for a long time, eventually being detected only when permanent damage is already unavoidable (as demonstrated by the well-known Stuxnet incident [Langner, 2013]). This is in sharp contrast with the immediately noticeable effects of attacks such as DoS.

PCOM supports several options for pushing the ladder logic project to the PLC RAM or flash memory. Although actively used by applications [Unitronics, Inc, 2019b] [Red Lion, 2019], such options and their specifications in terms of the PCOM packet structure are not publicly documented.

Nevertheless, there is an option that allows pushing (and, later, recovering) the PLC project, which is used by [Unitronics, Inc, 2019b]. If the PLC was programmed with that option, the lack of authentication and authorization makes it possible to use a rogue setup to retrieve the original project, change it as desired, and download it again to the PLC. Moreover, since no encryption is used, it is also possible to reconstruct the entire packet sequence or even to change such packets on-the-fly to recreate a MITM attack.

Observed network traffic showed that a multi-part PCOM/Binary operation (with command code `0x41`) was used to transfer a relatively large data block to the PLC. Moreover, it was possible to group all the message parts, contained in the PCOM/Binary data field, since each one is identified by little-endian sequential value (starting from 0 and incremented by 1) at bytes 16th to 18th. Additionally, it was also possible to identify the data format – a plain PKZIP file – by looking for its signature. Similarly, when uploading the PLC project to Visilogic, a PKZIP file is also transferred from the PLC. Since PCOM communications are not encrypted, as long as the malicious attacker is able to access the communication he should be able to reconstruct this file.

Moreover, the PKZIP file, not encrypted, contains a single file inside, a Microsoft Jet4

Database. The database is protected by a master password. Nevertheless, Jet4 databases are known to obfuscate the password in the file header, so the database password can be easily guessed. In the aforementioned scenario, as an example, it was possible to change the `XOR` ladder element to an `AND` element simply by updating its element type within the database, after guessing the database password. This required experimentation with several undocumented functions and values. A simpler path, in some scenarios, would be to use a rogue application deployment to retrieve and reprogram the PLC ladder logic.

4.2.8 Fuzzing the PCOM Protocol

In the previous subsections several attack cases were discussed, using the reference PCOM scenario to illustrate their fundamental operation and deployment procedures. It was possible to perform these attacks after conducting an investigation about the inner workings of the PCOM protocol, based on publicly available documentation and analysis of network traffic.

In order to go beyond, a network fuzzer can be used. Network fuzzers are tools that automate and test different types of input conditions by generating random or semi-randomized values. For the PCOM protocol, known functions could be validated, for instance, against malformed packets such as protocol violations, bad check-sums and unexpected fields values.

Fuzzing the PCOM protocol might also be useful to explore undocumented functions, protocol vulnerabilities or simply to discover specific inputs that might crash a PLC (e.g. a buffer overflow), using an automated procedure. This automation is also valuable to assess how different PLC models handle exceptional conditions or to discover supported PCOM functions in each model. Although outside the scope of this thesis, those vulnerabilities might apply not only to PCOM PLCs but also to software applications used in the HMI side.

A PCOM fuzzer was developed in the form of a Metasploit auxiliary module, to send PCOM-specific packets and evaluate the behavior of the PLC. This module allows specific testing options such as sending randomized values of some specific protocol fields (e.g. PCOM/ASCII command codes) or completely random messages.

Table 4.1 provides a brief summary of the tests and results performed during the conducted experiments. Non-conclusive results were observed, as the tested PLCs sometimes do not reply or reply with invalid or empty results. Nevertheless, none of the tried combinations resulted in PLC DoS conditions – which is positive from a security point-of-view. Based on Test 1, it was observed that tested PLCs accept and reply to malformed-packets containing invalid PCOM/TCP length values. Nevertheless, based on Tests 3 and 4, it appears that the tested PCOM PLCs acknowledge all the TCP requests but do not respond to bad PCOM requests (e.g. they do not respond to packets with bad checksums or non-existent Unit IDs). Opposed to other protocols, PCOM does not seem to return any exception or code to invalid requests. This makes it harder to identify all the hidden functions and features of a given PLC – which is positive from a security point-of-view. Moreover, the tested PCOM PLCs seem to reject consecutive connections if they are sent too fast (Test 2), forcing scouting processes to slow down.

Table 4.1: Summary of the results of the PCOM fuzzing tests

Test ID	Mode	Description	Result
1	Both	Invalid PCOM/TCP Length	PLC accept requests
2	ASCII	Consecutive connections	PLC rejects too fast connections
3	Both	Bad check-sum	No reply from PLC
4	ASCII	Non-existent Unit IDs	No reply from PLC
5	ASCII	Random command codes	No reply
6	ASCII	R&W with random params	No reply or invalid

4.2.9 PCOM Datasets - a Manifold Contribution

Datasets are vital to study and understand how protocols work, as well as for the development and validation of security tools, being used for model training and validation (e.g. in the case of Machine Learning classifiers) or for establishing nominal parameters for statistical analysis. As far as the author of this thesis knows, when it comes to the PCOM protocol, there were no publicly available datasets.

In the particular case of the research effort hereby documented, a PCOM dataset was fundamental to support the tools development process and to validate their behavior. Moreover, the author of this thesis believes there might be several undocumented and undiscovered PCOM/Binary functions used during the reprogramming step – having a dataset would be instrumental to verify their existence, semantics and structure.

In order to help bridge this gap, a set of labeled individual PCOM/TCP captures in libpcap format was publicly released [Rosa, 2019d], each associated to a single PCOM operation.

4.3 Mitigation Strategies

The two previous sections focused on the security of SCADA protocols. First, with a discussion on how SCADA systems can be attacked, from a practical standpoint, and then with a more extensive analysis of the security of the PCOM protocol. This Section complements the general discussion of mitigation strategies already provided in Section 2.2 with the presentation of a Snort ruleset specifically developed for the PCOM protocol.

Several strategies might be pursued to address the security shortcomings of protocols such as Modbus or PCOM. One of the first approaches, if not the most obvious, would imply a complete redesign of the protocol to add missing security features. For instance, as already mentioned in Section 2.2, the Modbus Organization released in October 2018 a new Modbus/TCP Security protocol specification [Modbus Organization, Inc, 2018] with enhanced support for confidentiality, data integrity, authentication and access control. However, the majority of the huge amount of already installed Modbus devices will not be able to upgrade to this new, more secure version. For other protocols, such as PCOM, the situation is even worse, since there is no evidence pointing towards ongoing security-focused improvements.

Considering this situation, several strategies may be considered to mitigate the aforementioned issues, as described next. Rather than definitive solutions (something that would imply revising the protocol), these proposals intend to provide the means to detect and block potentially unwanted communications in existing IACS for which it will be still necessary to keep legacy devices and protocols for a long period of time.

A NIDS instance (e.g. Snort) can only be effective for a specific environment as long as it is loaded with an adequate ruleset covering the protocol mix being used, such as the set of Snort Rules to handle PCOM/TCP traffic [Rosa, 2019c] created by the authors of this thesis, derived from the work presented in the previous section. Next, some details of this ruleset are briefly presented, as an example of how to build such rulesets in general.

Each rule was designed to match a single PCOM command code, so that it is possible to distinguish not only administrative remote commands, but also reads and writes of different types of operands. For instance, one might want to block system related operands (i.e. `SB`, `SI`, `SL`) and allow the access to other operands. One might also want to distinguish between read and writes in some of the operations. Each rule searches for specific byte values corresponding to the command codes by using hard-coded positions (according to the protocol specification). To improve readability, those command values were specified using their ASCII representation for PCOM/ASCII and hexadecimal values for PCOM/Binary rules. Figure 4.17 shows an excerpt of those rules. Such rules specify portions of the TCP payload (based on the `offset` and `depth` Snort keywords) and match them to hard-coded values (i.e. the signatures of each protocol).

In ASCII mode, the command code of a request may have 2 or 3 bytes, whereas the replies will always have only 2 bytes. In the reply, the command code appears at a different offset.

```
alert tcp any any -> any 20256 (flow:established; content:"ID"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Identification (ID)"; \
classtype:attempted-recon; sid: 1000001; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"CCS"; \
offset: 9; depth:3; msg:"PCOM/ASCII Request - Stop Device (CCE)"; \
classtype:attempted-dos; sid: 1000004; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"UG"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Get UnitID (UG)"; \
classtype:attempted-recon; sid: 1000007; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"MI"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Read Memory Integers (MI)"; \
classtype:attempted-recon; sid: 1000027; rev:1;)
alert tcp any any -> any 20256 (flow:established; content:"SB"; \
offset: 9; depth:2; msg:"PCOM/ASCII Request - Write Memory Bits (SB)"; \
classtype:attempted-recon; sid: 1000038; rev:1;)
```

Figure 4.17: Excerpt of PCOM rules for Snort NIDS

Such variations in PCOM/ASCII can be also accommodated by using Snort `offset` and `depth` keywords. In both modes, the keyword `byte_test` was used to verify whether the payload is encoded in PCOM/ASCII or PCOM/Binary mode, based on the value of the third byte.

Nevertheless, in PCOM/Binary mode such hard-coded rules might only be able to detect the operation type (e.g. 12th byte at 77 for reading operands), being unable to distinguish the operand types and addresses (which are structured in data blocks with variable size), because in this mode a single packet might request more than one operand type. This calls for a dedicated PCOM Snort preprocessor, not only to unlock more complex Deep Packet Inspection (DPI) logic but also to improve the range of available keyword options to further improve the readability of such rules.

Finally, as already discussed in Section 2.2, other approaches could also be used to further improve the detection capabilities within a given IACS environment, such as specialized data diodes and honeypots. Nevertheless, the availability of such tools is still rather limited, especially outside the couple of more popular SCADA protocols. For PCOM, for instance, there are no data diodes or specialized honeypots available.

4.4 Summary

The first part of this chapter described the sequence of attacks involved in an intrusion process, in the scope of a specific IACS. In a SCADA environment, the reconnaissance step it is often similar to other Information Technology (IT) network scans, with the main difference being the additional PLC enumeration (e.g. through the usage of the Modbus `unitID` field). Specific network segmentation and setups might also pose additional barriers and further difficulties for the attacker at this stage.

Simple service disruption was straightforward, since after the attacker has access to the network it is a matter of redirecting Modbus traffic (causing the disruption) or even flooding the PLCs, as they typically have limited resources. The communication hijacking attack that was implemented was more complex and tightly coupled to the field processes of the victim IACS. This complexity derived from several factors, such as the need to stealthly mimic part of the physical process behavior to deceive the system operators.

These experiments, conducted from the attacker's perspective, allowed to dig into technical details of involved protocols and, ultimately, to understand how to design and implement appropriate countermeasures. By no means this work covered all the potential attack scenarios within a SCADA environment – it was intended as a more practical approach to some of attacks scenarios described in the literature in an abstract manner.

The second part of this chapter presented a security analysis of the PCOM protocol. In contrast with the more usual approach to this type of analysis, often based on ad-hoc procedures mostly focused on filling a CVE report, the work hereby documented goes full-circle, providing several public contributions that are instrumental for infrastructure operators, security pentesters, auditors and researchers alike.

Starting from a purpose-built research testbed and following the PTES methodology, several vulnerabilities of the PCOM protocol were scouted, pinpointed, identified and explored to implement and test distinct use cases. Together with an effective Open-source intelligence (OSINT) effort, which provided information about the core PCOM functions, a security analysis procedure was pursued to showcase how valid PCOM messages can be leveraged by a malicious actor to ultimately gain full control over a controlled process. Finally, several mitigation strategies were proposed, albeit a more definitive solution would imply an entire redesign of the protocol.

Like other contemporary SCADA protocols, it was found that PCOM lacks security features such as confidentiality or integrity, being vulnerable to several types of network attacks that might be used to disclose information about the process under control, affect the integrity of inflight data, manipulate runtime device registers or even disrupt the process. Nevertheless, despite the importance of such issues, it must be clearly stated that PCOM is no worse than its contemporary counterparts. As discussed in Chapter 2, IACS were originally designed based on the preconceived notion of air-gaped systems and focused mainly considered on the functional aspects of communications. Today, given all the discussed security concerns, it became necessary to have the means to monitor, assess and enforce the security of such protocols, especially in mission-critical environments. Among other approaches, this implies creating protocol-aware modules for NIDS, honeypots or data-diodes. Section 4.3 presented some details about how to create such modules, based on the example of a NIDS ruleset for PCOM.

As already mentioned in the beginning of this chapter, the work hereby described directly resulted in two research papers and a several open-source contributions to widely used tools such as Snort, Wireshark, Nmap, Metasploit and Scapy.

5
Event Streaming Layer

The previous chapter presented the exploratory analysis of the security of Supervisory Control And Data Acquisition (SCADA) protocols, to better understand how cyber-attacks on Industrial Automation and Control Systems (IACS) can be performed. Isolated detection mechanisms, such as SCADA-aware rulesets for Network Intrusion Detection System (NIDS), were also discussed. Nevertheless, such isolated mechanisms are not enough. As already discussed, more flexible and comprehensive solutions are necessary, such as the holistic data-driven framework introduced in Chapter 3.

This Chapter is devoted to the Event Streaming Layer, one of the core modules of the proposed framework. The event streaming layer fulfills two main purposes: (1) to provide an efficient, distributed and decoupled mechanism for inter-component communication with *exactly-once* processing guarantees; and (2), to provide domain-level processing capabilities. The idea is that different (and heterogeneous) security probes can leverage the event streaming layer to push their outputs and evidences (i.e. events) to a highly flexible messaging system. Moreover, as discussed in Chapter 2, such stream of events can represent not only reports of security incidents but also other types of information, such as application logs, network traffic or telemetry data. Therefore, as part of the event streaming layer, domain processors are used to deploy stream processing mechanisms at domain level. Those mechanisms, following the hybrid edge/cloud paradigm mindset, enable a preprocessing (and decentralized) stage near the collection points – optimizing the amount of unstructured data pushed to the upper layers (i.e. the data analytics layer). Moreover, such preprocessing stage is also strategical to support the correlation of events from different components, aggregate and filter different types of events, or even extract additional aggregated features to use as input of Machine-Learning (ML)-based anomaly detection mechanisms.

This chapter starts with the description of the overall proposed data-driven workflow and the proposed event format – a custom data model to exchange and describe all the cyber-physical events (Section 5.1). The details of the event streaming layer (i.e. the messaging system and the domain processors) are discussed in Sections 5.2 and 5.3, respectively. Section 5.3 also presents a reference use case which puts in evidence the benefits of using the intermediary preprocessing stage (more specifically, a time-windowed feature aggregation scenario to extract a set of additional network based features is described). Those features were later leveraged to implement an anomaly detection scenario based on network aggregated statistics (cf. Chapter 6).

Finally, an evaluation of the messaging system itself is presented in Section 5.4. Rather than focusing only on raw performance numbers (which might differ in different deployments), this evaluation discusses how the different settings can be adjusted to achieve different service goals and distinct deployment needs.

Part of the work presented in this chapter has also been published in [Rosa et al., 2021].

5.1 Data-driven Workflow and Event Datamodel

The specification of the full event datamodel results from joint work of the University of Coimbra team involved in the ATENA project, where the author of this thesis had a key role.

As mentioned before, the Intrusion Detection Message Exchange Format (IDMEF) format [Feinstein et al., 2007] was adopted in CockpitCI to represent security incidents using Extensible Markup Language (XML) notation in a semi-structured way, useful in the processing at upper layers. Nevertheless, since then, alternative formats were proposed (cf. Section 2.5). For instance, the Incident Object Description Exchange Format (IODEF) [Kampanakis and Suzuki, 2017], which is upward compatible with IDMEF, was proposed as a more generic way of exchanging information between Computer Security Incidents teams. Still, both formats were designed on top of a XML structure. Thus, they were not optimized for on-disk storage, long-term archiving or in-memory processing [Kampanakis and Suzuki, 2017].

Taking the inspiration from IDMEF, a custom datamodel (designated as Intrusion and Anomaly Detection System (IADS) datamodel) was specifically proposed in the scope of ATENA. This datamodel could simultaneously be used to represent a generic event (i.e. not only a security event but also events such as telemetry data), avoid the complexity of other existing formats and fit into the increasingly demanding (Big Data) IACS environments.

Ultimately, the idea was to use a vendor-agnostic format with a balance between flexibility (to represent highly heterogeneous events), syntax and performance. The performance side, often overlooked in security-related formats, is an increasingly important factor as we move towards distributed data-driven architectures.

According to this datamodel, each IADS message has three main parts:

- An Universal Unique Identifier (UUID), type 4 (RFC 4122 [Leach et al., 2005]) – a random-generated 128-bit number that uniquely identifies the event in the entire distributed IADS architecture;

- A `metadata` block holding a list of elements that the event has gone through (`origins`) and a SHA-256 checksum of the remaining payload;
- And the `payload` itself, encompassing an unbounded typed list of events (`events`) – since one event might hold and represent several particular occurrences, for instance as a result of a previous aggregation step.

Each `origin`, starting by the component where the event was generated, contains: a `Uniform Resource Identifier (URI)` (following the RFC 3986 syntax [Berners-Lee et al., 2005]), a UserAgent (RFC 7231 [Fielding and Reschke, 2014]) and a `Timestamp` of nanosecond precision (RFC 3339/ ISO 8601 [Zopf, 2002]).

Each `event` is composed of: a `type`, a `URIofType`, a `Severity` (following the eight levels as described in RFC 5424 [Gerhards, 2009]) and the `data` itself. Five types were initially specified (`Application`, `Network`, `Host`, `CyberPhysical`, `Other`). This gives the flexibility of either specifying a new specific event type through the `URIofType` field or using the generic `Other` type containing only a list of `meaning` and `content` pairs.

On top of this syntax, Apache Avro [Apache Software Foundation, 2020e] was used for data serialization, due to its suitability for Big Data and heterogeneous environments. Avro supports rich data structures, binary data formats (as well as human-readable JavaScript Object Notation (JSON) representations), multi-language, dynamic typing and optional static code generation. Figure 5.1 shows an example of an event using the IADS datamodel to represent an ARP-based MITM, generated by a NIDS.

This datamodel is used to exchange messages between the different IADS components (e.g. it is used to push network-related metrics to the messaging system and by domain processors to describe the computed aggregated metrics). Likewise, as later described (cf. Section 5.4), it is also used in the evaluation of the messaging system itself.

Figure 5.2 illustrates the overall data-driven workflow. Probes are components strategically deployed to collect data. Each probe, besides the sensor logic itself, is composed of a management adapter and an event adapter. For sake of clarity, even though these two internal probe components were not designed by the author of this thesis, they are briefly mentioned here. The event adaptor, a custom YAML Ain't Markup Language (YAML) configuration file and a set of regular expressions, was used to map different output formats (e.g. syslog) into a unified data format, as well as to push them into the streaming layer. The event management adapter, using a configuration file and a custom protocol on top of

```json
{
  "uuid": "99e01a19-87be-4e27-9fd4-b100afd32f57",
  "Metadata": {
    "Origins": [
      {
        "URI": "IADS:\\\\172.27.249.20",
        "UserAgent": "snort/2.9.11.1",
        "Timestamp": "2018-10-12T17:18:39.412586811-04:00"
      }
    ],
    "Checksum": "5f5e7c6a73d8bd008a1fbfcb2b8304364c2118d6c6cd6bf2f1557ad83aff0e76"
  },
  "Payload": {
    "Events": [
      {
        "URIofType": "/schema/NetEvent.txt",
        "Type": "Network",
        "Severity": "emerg",
        "Data": {
          "Subtype": "Attack",
          "Entities": {
            "Sources": [
              {
                "ProtoName": "Ethernet",
                "Values": [
                  "b8:27:eb:d9:bc:e0"
                ]
              }
            ],
            "Destinations": [
              {
                "ProtoName": "IPv4",
                "Values": [
                  "172.27.248.213"
                ]
              },
              {
                "ProtoName": "Ethernet",
                "Values": [
                  "00:0d:22:09:bd:dd"
                ]
              }
            ]
          },
          "Items": [
            {
              "Meaning": "Attack",
              "Content": "Attempted ARP cache overwrite attack"
            }
          ]
        }
      }
    ]
  }
}
```

Figure 5.1: Example of a event represented with the proposed datamodel.

the Message Queuing Telemetry Transport (MQTT), was used for remote management of the probe.

Each probe is responsible for formatting events into the proposed datamodel and for pushing them to preconfigured topics of the messaging system (as detailed bellow). Next, for each domain, according to the specific deployment, different domain processor tasks are used to consume, preprocess and push the events to one or more output topics. Later, those events are consumed and processed by different tasks, part of the global analytics layer. The final outcomes from the data analytics layer (i.e. the alarms resulted from the analytics tasks, as discussed in Chapter 6) are then pushed to another topic and/or consumed by a web interface which exposes events to the SCADA operator.

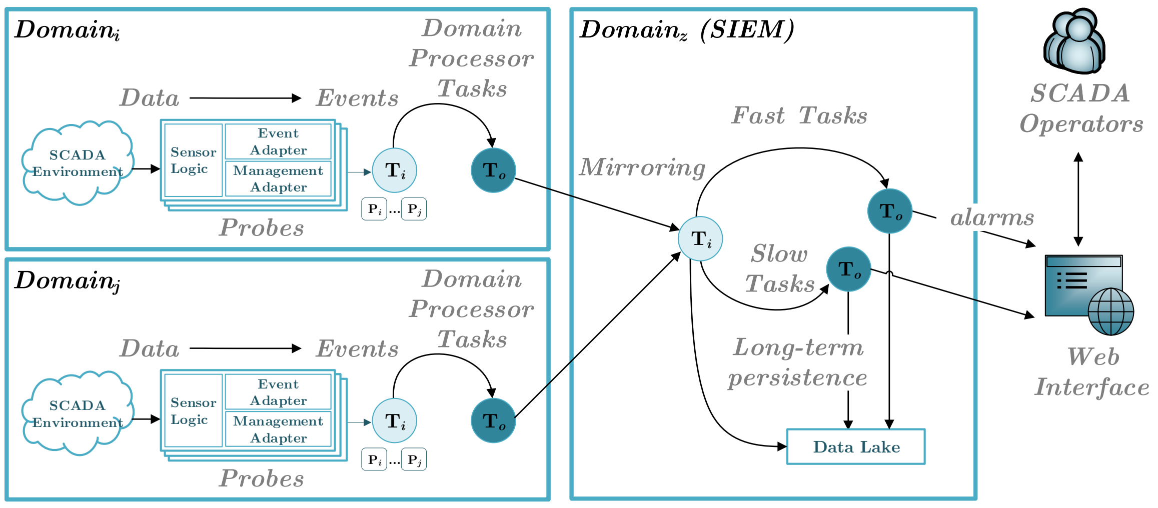


Figure 5.2: Event data-driven workflow.

5.2 Messaging System

This section focuses on the messaging system, a key part of the event streaming layer, as highlighted in Figure 5.3.

The messaging system is composed of the messaging nodes, each containing a message broker, and arranged in clusters to store and serve the events generated by the distinct components. Messaging clusters and brokers can scale-in/-out and spread geographically to meet the different service goals of different deployment scenarios. Moreover, the messages can also be replicated across multiple topics, and each topic can be partitioned across several message brokers. This flexibility constitutes a key advantage for supporting different messaging requirements within the same deployment (e.g. high-severity messages vs. low-priority telemetry data).

Similarly to probes, each messaging node also includes a management adapter (to remotely control the broker settings) and an instrumentation exporter – that exposes the instrumentation metrics from the messaging node through a instrumentation backend and a HTTP endpoint [Linux Foundation, 2018]. Such metrics are a fundamental part of the monitoring process, in order to monitor the behavior of each broker (e.g. plot the number of messages per node or per topic).

Finally, the messaging system also includes a schema registry, used to store and serve the proposed datamodel (and future versions) to all the components (e.g. probes and processing tasks) through a common endpoint, as well as a connector API that allows seamless integration of third-party producers and consumers through dedicated connectors.

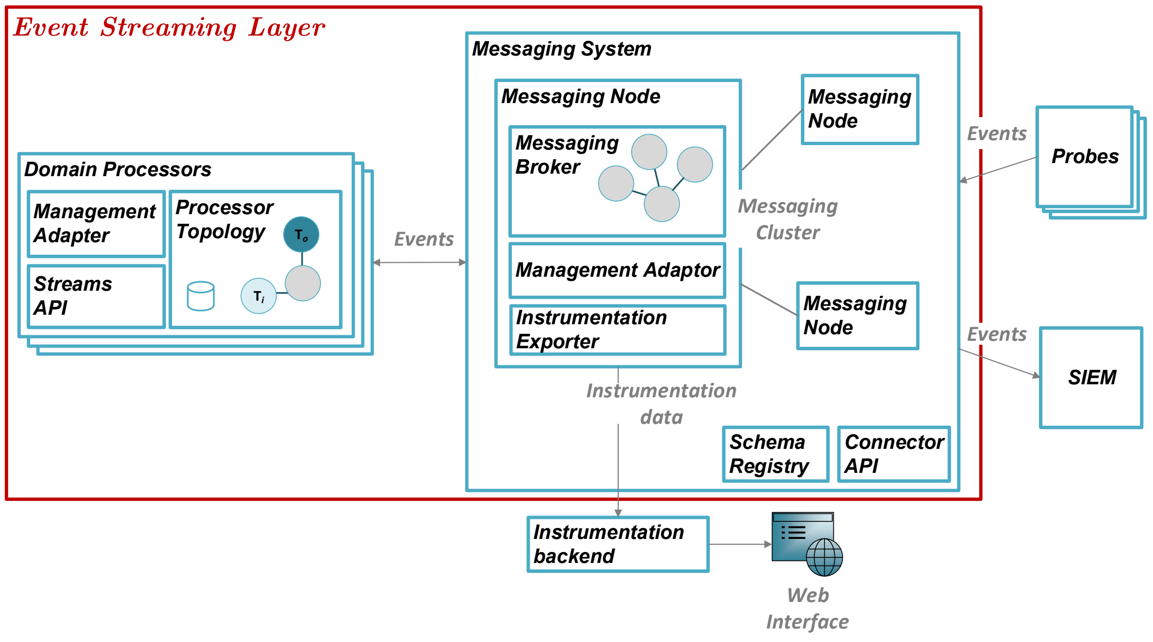


Figure 5.3: Event Streaming Layer Reference Architecture.

Apache Kafka [Apache Software Foundation, 2020b] was used to implement the proposed messaging system (i.e. the messaging brokers, schema registry and connectors API), since it is a distributed messaging system capable of achieving high message throughput without sacrificing latency – potentially supporting millions of messages per second, as required by larger IACS. Moreover, it fits the idea of supporting different types of message priorities and service goals. Each message, a key-value abstraction, is always persisted to disk – thus durable. Kafka also has multiple fault-tolerance and durability capabilities (as later detailed).

The Kafka connector API [Shapira et al., 2017] was used for seamless integration of multiple connectors through a common interface. Such connectors easily ingest and export data to additional third-party components (e.g. a database). Moreover, they rely on the built-in Kafka messaging system to support efficient, distributed and fault-tolerant operations.

The Confluent Schema Registry [Confluent, 2020] was used for schema management. It natively supports the Avro format (used in the datamodel) and leverages the Kafka messaging system to provide a decoupled way of serving different versions of message schemas.

Security-wise, Kafka natively supports Secure Sockets Layer (SSL) encryption on the fly between broker and clients, authentication through SSL certificates or Simple Authentication and Security Layer (SASL), and Access-control list (ACL)-based authorization. Thus, Kafka can support the needs of secure inter-component communication in the proposed IADS.

Another useful Kafka feature, opposed to pure queuing mechanisms, is that the messages are not deleted after being consumed. Instead, Kafka uses a sequential offset to identify each record within a partition and has the notion of consumer groups. Moreover, a message can be load-balanced across different consumer members, and multiple consumer groups can consume the same message. For instance, within a IACS environment, an off-the-path task can persist all the events (e.g. via Kafka Connect API to the Data Lake) along with multiple tasks cooperatively aggregating the same events – further supporting the concept of flexible processing schemes and topologies.

Finally, the Kafka ecosystem is widely used (including more than 80% of all Fortune 100 companies [Apache Software Foundation, 2020b]), has an active community and considerable third-party integration and built-in stream processing capabilities, which were also decisive factors to choose this framework.

As discussed before, regardless of the physical setup, for smaller scenarios a single cluster can support all the IADS operations, whereas a local messaging cluster per domain can be used to take advantage of data locality.

Figure 5.4 illustrates a scenario where the messaging system is deployed with two Kafka clusters (one local and one global). The figure also shows how different topics can be partitioned and replicated across multiple brokers. In the presented scenario, two topics are depicted: a priority topic with only two partitions but replicated three times, and a telemetry topic with four partitions but only replicated two times. Such an overly simplified setup is shown as an example of how different events can be grouped in different topics of interest with distinct service goals. In this example, even if any two brokers fail, the previously committed priority messages would still be available (due to the replication factor of 3). The same two failures would result in data loss in the telemetry messages topic – which, in the presented scenario, only tolerates one broker failure.

The specific setup, topics, and partitions of each broker heavily depend on each deployment's needs, being unrealistic to define them *a priori*. Even so, the proposed messaging system provides a flexible approach to accommodate different deployment scenarios.

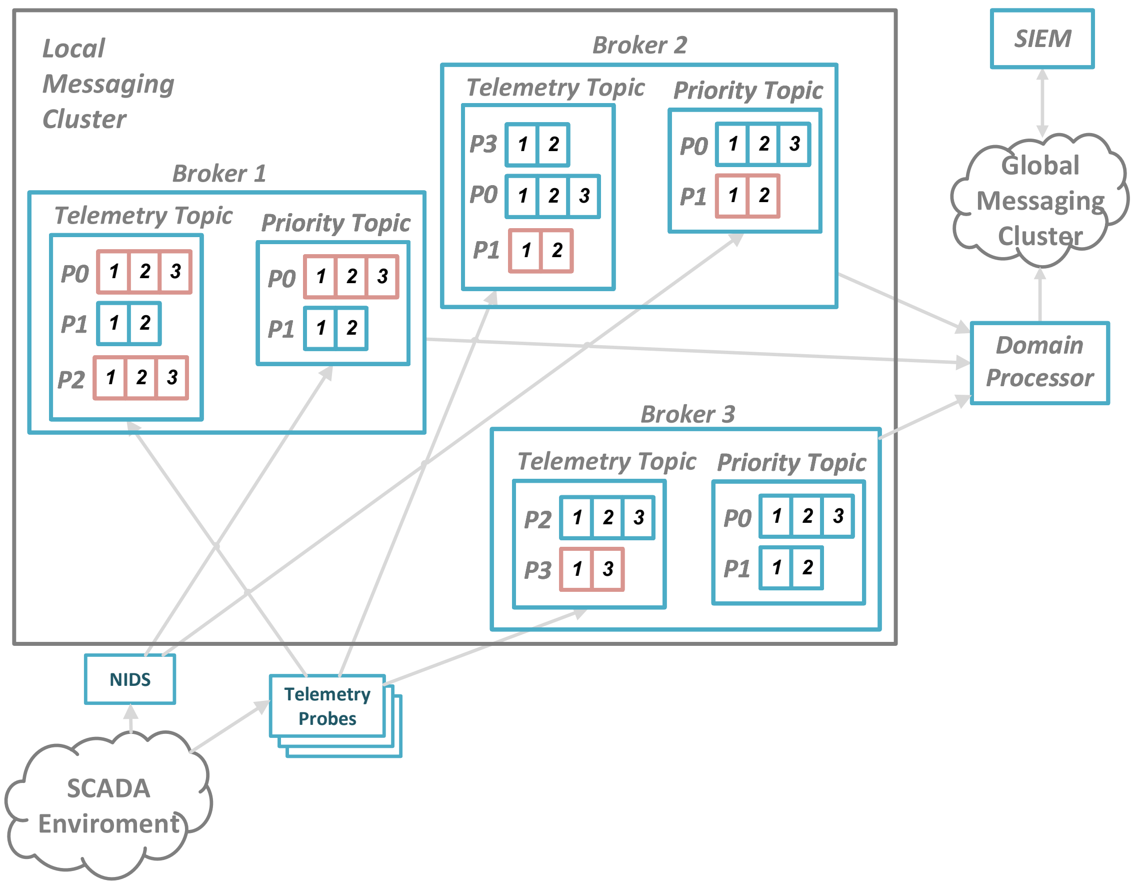


Figure 5.4: Base messaging system scenario with two Kafka clusters and different topics settings

5.3 Domain Processors

As discussed earlier, virtually, there are no limits to the amount of security probes. Multiple probes may report the same occurrence or attack (e.g. a network scan can be detected, for instance, by a NIDS and a honeypot). Moreover, the same attack may be successively reported by the same probe (e.g. long network-based attacks may get reported multiple times by the same NIDS instance). Some anomaly detection algorithms may not need the output of all probes – probes can simply generate more data than needed (or irrelevant data) for the anomaly detection task. On the other hand, simpler probes may lack contextual information or fail to comply with the event format (e.g. third-party probes hard to customize). In summary, there are several reasons why we need to optimize event flows by grouping them (e.g. fewer events containing the information from multiple occurrences). Many of these scenarios imply the need for some sort of preprocessing.

To address this need, domain processors are part of the proposed event streaming layer, being responsible for implementing the first-preprocessing step. Moreover, domain processors decouple this preprocessing stage from the remaining components (e.g. a given security probe), enabling a more modular design where different domain processors can be dynamically deployed, on-demand, to support different preprocessing requirements (e.g. different message priorities). In that sense, domain processors are responsible for correlating, transforming or enriching event messages on a per (logical and/or physical) domain basis. Domain processors may also implement all sorts of messaging and routing patterns, such as those commonly found in enterprise applications (e.g. Content-Based Router, Recipient List, Routing slip, Resequencer) [Hohpe and Woolf, 2004].

Anomaly detection algorithms often rely on aggregated features (e.g. averages, sums). For instance, network aggregated statistics can be seen as a valuable indicator of the behaviour of SCADA environment. For static environments, where the number of hosts and communications remains constant over the time, a small variation on the network traffic might indicate an anomaly or a normal but unusual event. The usage of domain processors enables the efficient computation of those features at domain level, thus optimizing the entire processing pipeline. This enables a balance between computing those statistics locally at the probe, on resource-constrained environments and devices, or pushing everything to a global data analytics layer which might run on a distant location or even on a third-party cloud provider.

An early assessment of how classic event correlation tools could (Esper, SEC, Nodebrain and Prelude) fulfil the event processing and correlation needs [Rosa et al., 2015] was already summarized in Section 2.4.1. Nevertheless, as referred before, those tools are mostly focused on rule-based event correlation and not oriented towards distributed and Big Data environments – thus, they were not included in the final design of the proposed approach.

Instead, the proposed approach uses domain processors, built on top of the Apache Kafka Streams Domain Specific Language (DSL) API [Apache Software Foundation, 2020c]. Despite being different components, the tight integration between the messaging nodes (based on Kafka brokers) and domain processors (Kafka Streams-based applications) simplifies the platform design. Moreover, in this way no additional processing framework is required to implement the preprocessing tasks – Streams-based applications can run as independent applications (e.g. a microservice) on top of the Kafka Streams API. Stateful streaming operations leverage Kafka's topics as a persistent storage layer. Likewise, multiple instances or threads of the same application can be deployed to work collaboratively and support application-level failures – natively supported by the Kafka Streams API by automatically restarting the same task in a different instance.

It should be noted that while there is no limit to the number of different applications (domain processors) in the proposed design, the number of instances of the same application is bounded by the number of partitions. In the previous example (cf. Figure 5.4) this means two and four instances for the priority and telemetry topics, respectively.

Each domain processor is composed of the Kafka Streams API itself, the management adapter (similar to the management adapter of probes and messaging nodes) and a topology. The topology includes a source (the topics where the probes produce) and a sink (the input topics of the SIEM layer). Additional intermediate topics can also be used for supporting stateful operations (e.g aggregations), in contrast to stateless operations (please refer to Kafka documentation [Apache Software Foundation, 2020c] for such distinction).

Figure 5.5 shows an example of how Kafka Streams DSL [Apache Software Foundation, 2020c] are leveraged to implement a feature aggregation task. Such tasks are the foundation of the domain processor concept. In the presented example, the domain processor extracts additional aggregated features grouped by time windows, on top of a stream of individual network packet features.

In this example, the initial `KStream <K, V>` is instantiated from one or more input topics, where `V` refers to the IADS messages. Next, all the messages are mapped to a common key, to perform a global aggregation. This is followed by a further split of the aggregated stream (a `KGroupedStream`) into a feed of time windows (a `TimeWindowedKStream`). Additional aggregation scenarios are possible by using a different `K` mapping in the previous step (e.g.

using the `uuid` field of the proposed event datamodel for unique counts, the `origin` field for further aggregation by message sources or the `severity` field to distinguish between different message priorities). Similarly, different types of time windows are supported (e.g. tumbling, hopping, sliding).

Afterwards, each individual feature, encoded within the IADS message using a list of *meaning*, *content* field pairs, is extracted into a new aggregated message. The output (the computed result of the aggregation) is continuously stored on a per-event basis. The presented example uses one-pass algorithms (e.g. Weldford's algorithm for computing the variance) to optimize the computation of the aggregated features. Like the original messages, the aggregated messages contain the aggregated features encoded using a list of *meaning* and *content* field pairs.

Finally, the aggregation result is transformed back to a plain `KStream<K, V>` (containing the final aggregated stream of messages) and ultimately pushed to the sink topic. It should be noted that the Kafka Streams API supports different types of window behaviors. For instance, each message in the input topic might turn into an output message, where each output message represents an intermediate result of the aggregation (default behavior). This example uses the `suppress` option to produce a single message, containing the final aggregation per time window. Moreover, these time windows might be event-driven, either using the event time of each message or triggered by the wall-clock time (using schedulers).

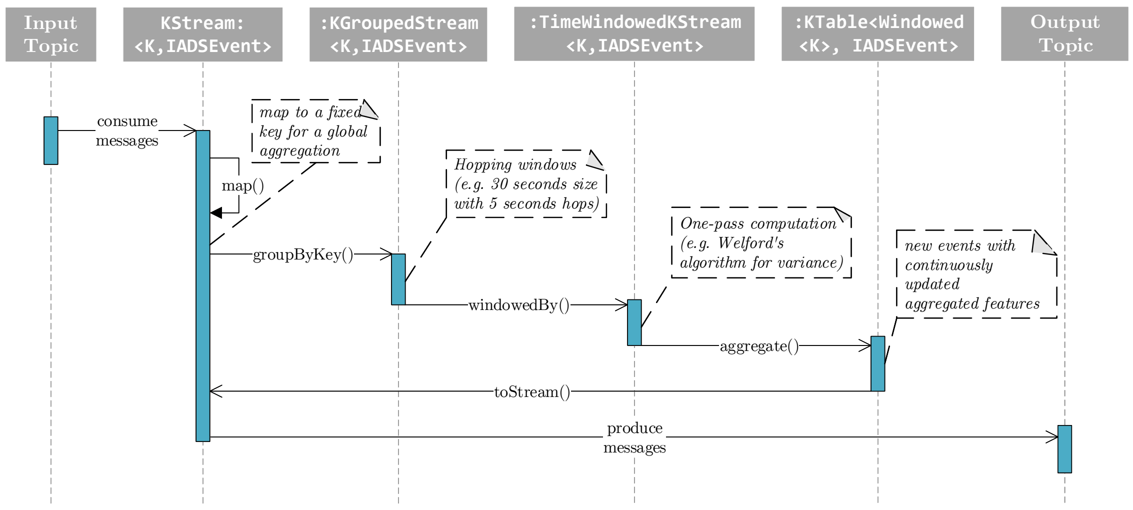

Input Topic
KStream: <K,IADSEvent>
:KGroupedStream <K,IADSEvent>
:TimeWindowedKStream <K,IADSEvent>
:KTable<Windowed <K>, IADSEvent>
Output Topic
consume messages
map to a fixed key for a global aggregation
map()
Hopping windows (e.g. 30 seconds size with 5 seconds hops)
groupByKey()
One-pass computation (e.g. Welford's algorithm for variance)
new events with continuously updated aggregated features
windowedBy()
aggregate()
toStream()
produce messages

Figure 5.5: Example of a feature aggregation by time windows using Kafka Streams DSL API.

5.4 Evaluation of the Event Messaging System

As mentioned before, the event messaging layer plays an important role on how components communicate with each other. Therefore, it is important to understand how its settings can be used to tune the platform to meet the requirements for next-generation IACS, such as event processing capacity and scalability.

For such purpose, a set of practical experiments was conducted using a Kafka cluster with three brokers – allowing to explore different replication and partition settings. Each broker was running on a different virtual machine with 8 vCPUs, 32GB RAM and 850GB of disk. All the tests were conducted in a Dell PowerEdge R440 host with a Intel(R) Xeon(R) Gold 5120 CPU (28 vCPUs), 256GB RAM and 3.2TB datastore (4x 10K RPM SAS HDD in RAID6 via PERC H740P controller), running an ESXi 6.7 hypervisor instance. An additional virtual machine (with 8 vCPUs, 8GB RAM and 20GB) was setup to work as a Kafka client (both producer and consumer). All the virtual machines, attached to the same vSwitch (with traffic shaping disabled), run CentOS 7.3 x64 with XFS. The broker instances were based on Confluent Docker images 4.0.0 (Apache Kafka 1.0.0).

Apache Kafka can be configured and optimized for different service goals, such as throughput, latency, durability or availability [Byzek, 2019]. Within a SCADA environment, we expect to handle at least two different types of messages: 1) high-priority alarms, as a result of the output of specialized detection probes, and 2) low-priority events such as telemetry data. Such distinction is important to fine-tune the platform for each use case. In the first case, it is critical to ensure low latency, meaning that the event should be forwarded and processed as soon as it is received, while in the second scenario it is more important to maximize the throughput by optimizing the number of events and the network overhead. For instance, a Kafka producer might be configured with the *batch-size* and *linger.ms* options to control whether the messages are sent as soon as they are ready or add a delay based on size or time respectively. Moreover, producers can send events using different levels of acknowledgments: asynchronous (i.e. without waiting for broker acknowledgments), synchronous with asynchronous replication (i.e. waits for an acknowledgment after the leader commit) or synchronous with synchronous replication, meaning the producer waits for the acknowledgment after an adjustable (*min.insync.replicas*) number of replica acknowledgments.

Since there are too many alternative factors to explore in such a scenario, a previous literature review [Kreps, 2014a] was used to filter out less relevant factors, in order to produce a manageable set of factors to consider in the experiments: the message size, the acknowledgment level, the buffer size, the number of partitions and the replication factor.

On the side of the messaging broker, different settings impact the overall performance, at both cluster and topic levels. Different topics can have different levels of replication and partitioning. The replication factor increases availability, critical for scenarios with strict fault-tolerance requirements (e.g. a topic with a replication number of 3 tolerates 2 broker failures). The partition number, on the other hand, is used as a parallel approach to improve both broker and client performance. Different partitions from the same topic can spread among different brokers and, in the same way, each partition can be assigned to different consumers. Nevertheless, a high number of partitions might lead to an increase of downtime in the case of a broker failure, due to the partition reassignment process. Regarding the message size, no limit is enforced by the message data model. Based on conducted experiments, a typical message containing all the mandatory fields varies between 2-5KB before Avro encoding and between 500-1000 bytes after encoding.

This test consisted of sending 1 million raw messages using the native Kafka client tool *kafka-producer-perf-test*, with different configurations for the client and broker. The following results summarize the obtained values in terms of the number of messages per second, throughput and latency. All the tests were repeated 10 times, with the confidence interval being computed using a Student's T distribution with a confidence level of 95%.

Figure 5.6 shows how the message size negatively impacts the rate of processed messages for different acknowledgement levels. The rate sharply dropped from over 100,000 messages per second (messages with 1 KByte) to 20,000 messages per second (7 KByte). This is an expected behaviour, since Kafka is optimized for small message sizes [Cloudera, 2019]. Likewise, the measured difference between full acknowledgement (acks = all) and no acknowledgement (acks = 0) also decreased as the message size grows.

Figure 5.7 shows how message size affects performance. For 1KB messages, more than 80,000 records per second were measured, a value that decreased sharply for larger sizes. A similar trend is observed for throughput.

Figure 5.8 shows the effect of different topic configurations, namely the number of partitions and the replication factor. For fully synchronous producers the replication factor imposes a severe negative impact, as each message needs to be committed by all the replicas before being acknowledged to the producer.

Overall, the practical experiments presented in the section provide a better understanding of how the proposed approach fits into the goal of providing a flexible framework, able to fit different scale, latency, distribution and resilience requirements. Obtained results show that the design of the event streaming layer, based on Apache Kafka, is not only flexible enough

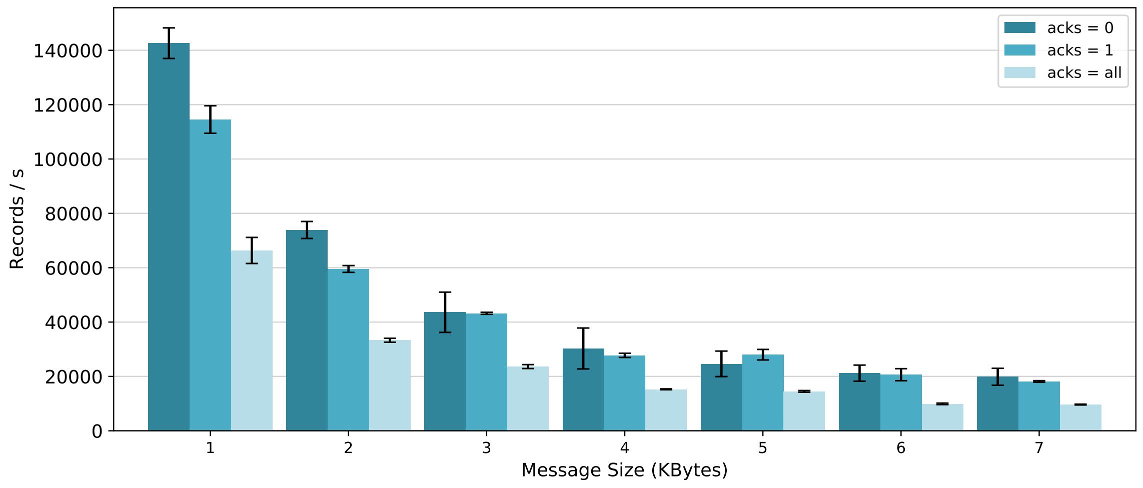


Figure 5.6: Average records / s using different acknowledgment levels and the message sizes for a 1 Million messages production test. Three brokers, 3 partitions, replication factor 3. Error bars shows the 95% Confidence Interval

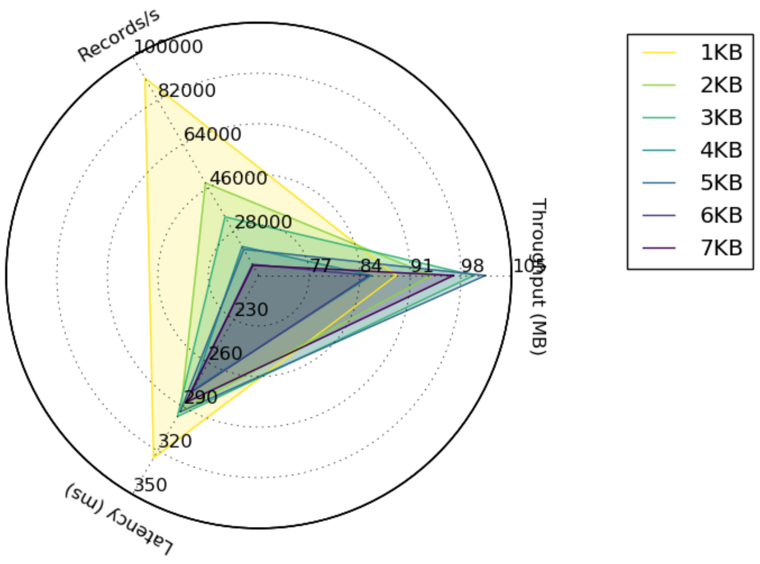


Figure 5.7: Average records / s, latency and throughput for different message sizes, Batch Size 16384 bytes, Replication Factor 3, Number of partitions 1.

to meet different deployment scenarios, but also able to be used as an efficient messaging broker mechanism capable of coping with the large number of events envisioned on more complex IACS environments.

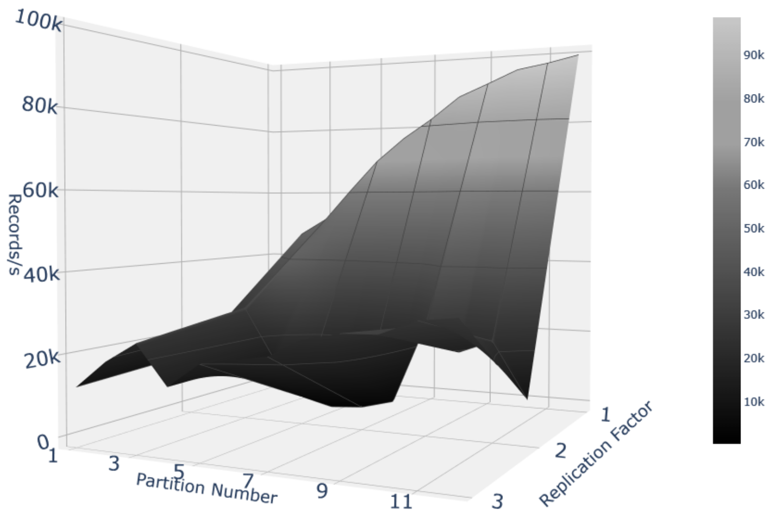


Figure 5.8: Average records / s based for different partition and replication values, 3KB message size, batch Size = 16384, acknowledgments = all.

5.5 Summary

The holistic intrusion and anomaly detection approach proposed in this thesis heavily relies on the idea of monitoring multiple and heterogeneous types of data sources. In that sense, this chapter presented the proposed data-driven workflow (i.e. how the different components communicate to each other) and the event datamodel (i.e. the schema used to describe the collected data). These two concepts are the foundation for leveraging the inputs from a wide range of heterogeneous components.

This chapter also described the event streaming layer (i.e. the messaging system and the domain processors). This layer provides an efficient, distributed and decoupled mechanism for inter-component communication, along with a way of filtering, aggregating and preprocesssing all sorts of events. This intermediate processing layer represents a step towards optimizing what is pushed to the upper layers. Moreover, this chapter detailed how to use Apache Kafka and Kafka Streams to design the proposed event streaming layer. By using Kafka for both messaging and preprocessing, it becomes easier to deal with different scenario requirements, such as low-latency, high-throughput or durability, while keeping adequate performance levels. As an example, a time-windowed feature aggregation scenario for extracting additional aggregated network statistics was presented.

A series of practical experiments was conducted to better assess the flexibility of the proposed approach, in terms of performance, scalability, resilience and distribution. Obtained results highlight the flexibility, which allows the framework to be deployed in (and optimized for) IACS scenarios with a wide range of requirements.

The next chapter describes the data analytics layer, which supports the deployment of different types of anomaly detection mechanisms, as part of the overall intrusion detection process. The next chapter will also explore the additional aggregated features extracted at the preprocessing stage, that will be used as input for various ML-based classification algorithms.

6
Data Analytics Layer

The event streaming layer was discussed in the previous chapter, detailing how distinct components communicate with each other to address the processing needs at domain/edge levels. This chapter describes the data analytics layer, discussing how it enables global situational awareness, based on the previously collected data.

Contrary to the local security components (e.g. honeypots, HIDS) and domain-level processing mechanisms previously discussed, the hereby proposed data analytics layer provides a global insight of the entire Industrial Automation and Control Systems (IACS) environment. For instance, ML-based classification algorithms running at global level can infer the probability of a cyber-security attack, based on processing of events and features extracted from each individual domain. This layer builds upon a distributed computation framework capable of efficiently running different types of algorithms, along with the concept of lambda architecture (cf. Section 2.4), which supports both Stream Processing (fast path) and Batch Processing (slow path) mechanisms.

This chapter starts by introducing the proposed data analytics layer (Section 6.1), including the presentation of the reference architecture and the proof of concept implementation. Next, it is discussed how this layer can be used to flexibly support different anomaly detection techniques and deployment scenarios (Section 6.2).

Section 6.3 presents a practical use case based on a data exfiltration attack scenario, showing how the proposed framework can be used to accommodate various Machine-Learning (ML)-based techniques. This scenario is a good example of a situation where relying solely on rule-based Network Intrusion Detection System (NIDS) would not feasible – since it would require predefined rules with known rogue servers or pre-established threshold-like strategies (e.g. based on "large" network packet lengths) which could be easily bypassed by skilled attackers. Instead, supervised ML-based algorithms are proposed and extensively evaluated, not to decide which one is better but to showcase the flexibility of the proposed framework in supporting and combining multiple algorithms.

Based on that use case, Section 6.4 presents the results of a more extensive evaluation study addressing the performance of those techniques, on top of the proposed framework. Finally, Section 6.5 provides a closing summary.

Part of the work presented in this chapter has been published in [Rosa et al., 2021].

6.1 Reference Architecture and Proof-of-Concept Implementation

When considering a Smart Grid (cf. Section 2.2), multiple security probes are expected to be deployed across the entire infrastructure. Nevertheless, each probe is focused on its own scope and specific tasks (e.g. analyse energy-related measurements; collect network traffic from a specific link; monitor suspicious events within a SCADA server). On the other hand, at global level, the deployment of additional anomaly detection mechanisms can help understanding the environment as a whole. As discussed in Section 2.3, ML-based algorithms are often proposed in the literature to detect deviant behaviors based on physical process values in IACS environments. Moreover, network-based indicators are often used to spot network traffic deviations (e.g. a sudden spike in the amount of traffic, deviant connections patterns, uncommon payload values). In the proposed approach, such inputs can be combined and analysed from a global standpoint at the data analytics layer.

The data analytics layer serves as an aggregation point for all the events originated from the multiple domains and components (the fast path). Additionally, it also supports computationally intensive processing tasks (the slow path) in a more efficient, flexible and scalable manner – opposed to requiring local components to have the computational resources required to run those tasks.

6.1.1 Reference Architecture

Figure 6.1 shows the reference architecture of the proposed data analytics layer, which is composed of processing nodes (i.e. the nodes used to run the algorithms) and the data lake (used to persist all the events, as discussed in the next section).

The underlying idea is to have an elastic approach where multiple computation nodes, arranged into a cluster, can cooperatively and dynamically work and scale-in/-out according to requirements. A master node (the cluster manager) orchestrates the cluster, being responsible for assigning and allocating resources on a set of additional slave nodes (the workers). For each application, the master node elects one of the slave nodes to run the *main()* application function (the driver program) and distribute the workload to the remaining nodes. Finally, each of those workers, containing a set of executors, is used to run the processing tasks.

These processing tasks, which can be both slow or fast processing mechanisms, consume the previously collected events and, by leveraging a cluster of distributed processing nodes,

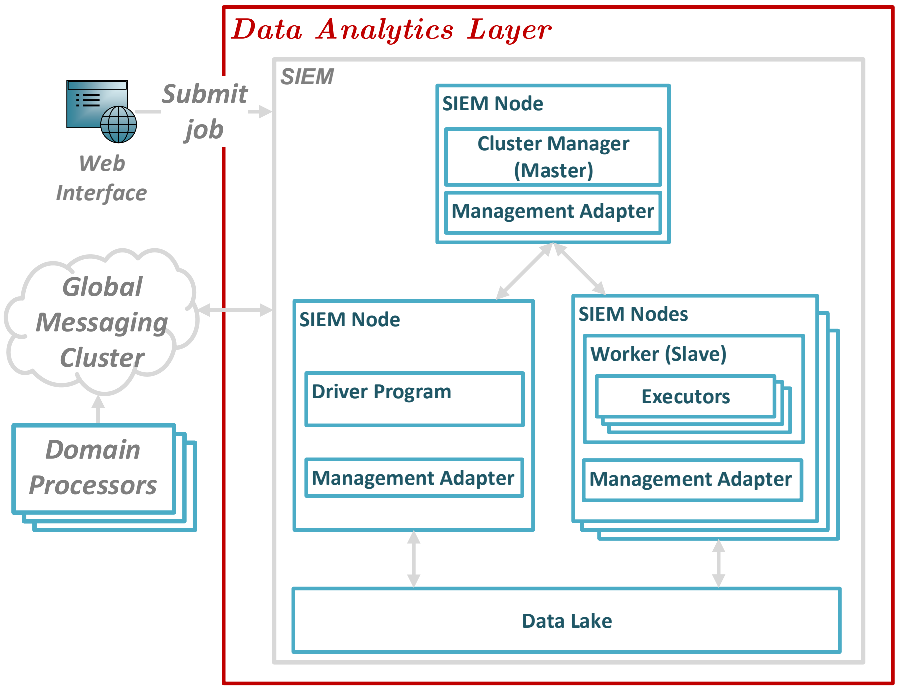


Figure 6.1: Reference Architecture of the Data Analytics Layer.

may for instance implement multiple ML-based anomaly detection algorithms. Moreover, this approach allows to combine different anomaly detection approaches (e.g. using different ML-algorithms, different processing schemes or even different types of features) to cope with a wide-range of different cyber-attacks and other types of anomalies.

Another key component of the analytics layer is the data lake, which has the function of persisting generated events. As previously discussed, an IACS can encompass a multitude of assets and protocols, spread along different logical and physical locations. Similarly to the messaging system and processing nodes, a distributed data storage approach is proposed for the data lake. This way, the proposed data lake accommodates a wide range of requirements, including complex scenarios involving massive amounts of events with strong scalability and availability needs.

In the proposed data-driven workflow, events are first pushed (and persisted) to a topic and messaging node. However, the main goal of the streaming layer is not the long term storage, but rather to support the intercommunication between all the components. This need of long term persistent storage is fulfilled by the proposed data lake.

This approach allows to retain events at scale, which is particularly relevant for IACS environments. Rather than relying on the storage from the messaging system, those persisted events can later be directly retrieved and used, for instance for offline processing analysis for forensics and compliance auditing purposes.

6.1.2 Proof of Concept Implementation

The remaining of this section provides some details about the proof-of-concept implementation of the data analytics layer.

Instead of using the popular TensorFlow [Abadi et al., 2015] or Scikit-learn [Pedregosa et al., 2011] frameworks – which are mainly focused on ML – Apache Spark [Apache Software Foundation, 2020d] was selected as the underlying computation platform. Spark is a general-purpose computation framework with native support of both streaming and batch processing, which provides additional flexibility to implement different types of intrusion and anomaly detection tasks.

Moreover, Spark supports the proposed distributed data processing approach, where a scalable number of processing nodes may cooperatively distribute the workload of different processing tasks. Spark also provides significant improvements over traditional map-reduce alternatives, by using an in-memory approach and an efficient Directed Acyclic Graph (DAG) scheduler [Hazarika et al., 2017].

Having said that, it is important to mention that Spark was not truly designed with stream processing in mind. Instead, a micro-batching approach is used as a low-latency processing mechanism. The latest versions include an experimental feature (*continuous processing*) that supposedly enables processing of each record as soon as it becomes available, claiming latencies around 1ms – however, this feature was not used in the scope of this thesis.

Feature-wise, Spark also supports a set of APIs for Streaming Processing, machine learning and dataframe manipulation. Notably, the newest Structured Streaming API [Apache Software Foundation, 2020d] allows to further unify, abstract and simplify the way both bounded and unbounded datasets are manipulated.

Finally, features such as the native Kafka integration (allowing the usage of Kafka as source/sink data sources), the built-in Avro support and a wide range of implementations and libraries of ML algorithms were also decisive factors to the choice of Apache Spark.

Regarding the proof-of-concept implementation of the Data Lake, Apache Cassandra was used. Cassandra is a distributed NoSQL row-based database that supports continuous availability (with eventual consistency), high-performance and linear scalability [Apache

Software Foundation, 2020a]. It is capable of seamless scale-in/-out and tolerates failures by using a shared-nothing principle. Each row supports from basic types to composed collections (e.g. lists, sets, maps) and allows to store both structured and unstructured data.

Figure 6.2 provides an overview of how Kafka Connect API was used to interface the event streaming layer and the data lake. As discussed in Chapter 5, Kafka Connect, part of the overall Kafka framework, allows to set up a set of connectors (both sinks and sources) to consume and produce from/to a topic. Such connectors can be used together with the schema registry for schema management and additional converters (e.g. to unlock the mapping of complex fields).

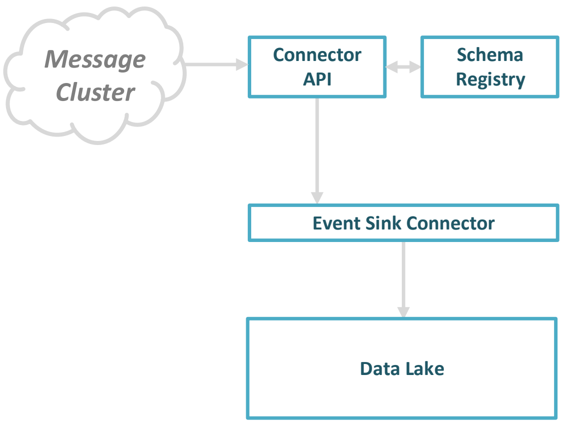


Figure 6.2: Messaging system and Data integration pipeline.

Rather than requiring additional libraries and code, the usage of such connectors, based on configuration files (as exemplified in Figure 6.3), offers a more simple, flexible and decoupled approach to ingest and export data from/to the messaging system.

In the proposed architecture, a Cassandra Connector from Landoop [Landoop, 2018] was used to convert on-the-fly between events (in Avro format) and Cassandra rows and columns. As reference, Figure 6.4 shows an excerpt of how event data fields might map into Cassandra data types.

6.2 Anomaly Detection on Top of the Data Analytics Layer

As already discussed, the vast majority of the surveyed IACS literature addresses either generic layer 2/3 attacks (e.g. network scans or ARP-based attacks) or specific application layer attacks leveraging Supervisory Control And Data Acquisition (SCADA) protocols.

```
{
    "name":   "cassandra-sink-atena",
    "config":  {
        "connector.class":
"com.datamountaineer.streamreactor.connect.cassandra.sink.CassandraSinkConn
ector",
        "tasks.max":   "1",
        "topics":   "kafka-topic",
        "connect.cassandra.kcql":   "INSERT INTO Events SELECT * FROM
kafka-topic withstructure;",
        "connect.cassandra.contact.points":   "cassandra.disney.dei.uc.pt",
        "connect.cassandra.port":   "9042",
         "connect.cassandra.key.space":   "atena",
        "connect.cassandra.username":   "<ommited>",
        "connect.cassandra.password":   "<ommited>"
    }
}
```

Figure 6.3: Cassandra sink connector configuration for Kafka Connect.

```
CREATE KEYSPACE atena WITH REPLICATION = {'class' : 'SimpleStrategy',
'replication_factor' : 3};
use atena;

CREATE TYPE Origin  ( URI text, UserAgent text, Timestamp text);
CREATE TYPE Origins( origins frozen<list<list<Origin>>>);
CREATE TYPE Content( bytes text, string text,);
CREATE TYPE Item( Meaning text, Content frozen<Content>);
CREATE TYPE Other( Code int, Items frozen<list<Item>>);
CREATE TYPE Network( /*[…]*/);
CREATE TYPE Data( Network frozen<Other>, Other frozen<Other> /*[…]*/);
CREATE TYPE Event ( URIofType text, Type text, Severity text, Data frozen<Data>,
Other blob);
CREATE TYPE Payload ( events frozen<list<Event>>,);
CREATE TYPE Metadata( origins frozen<list<Origin>>, checksum text );
CREATE TABLE Message( uuid text PRIMARY KEY, metadata Metadata, payload Payload);
```

Figure 6.4: Excerpt of Cassandra User-defined Type (UDT) fields used to represent the events.

Some of those attacks can be detected and mitigated using signature-based mechanisms (e.g. a NIDS with SCADA-specific rules) or additional SCADA-specific probes and components (cf. Section 2.2.3), while other attacks require anomaly-based detection techniques, such as those based on ML. Other approaches, such as leveraging the analysis of the physical IACS process values to detect subtle variations in the behaviour of the SCADA process (cf. Section 2.2), also rely on anomaly detection techniques.

Contrary to other researches, instead of focusing on a particular ML-algorithm, this thesis proposes a data-driven anomaly detection strategy that is not tied to a single protocol, a single attack scenario or a single detection technique. The proposed framework is able to collect data from all sorts of probes, to flexibly and dynamically support a wide range of deployment scenarios (local processing vs. global processing, resilience, scalability, etc.) and to integrate multiple detection techniques. Within this framework, the data analytics layer provides the support for integrating e combining those data-driven detection techniques.

Among such techniques, ML-based detection algorithms seem to be especially interesting for the proposed platform, due to their ability to detect anomalies based on large amounts of data without requiring specific rulesets defined in advance. This fits particularly well into the vision of a framework able to capture and process large amounts of data from various domains and geographic locations.

ML-based techniques have the advantage of not being tied to network-based scenarios – they mainly depend on the features and the algorithm itself. Thus, they can be applied to a wide range of scenarios. Such approaches follow a standard procedure of collecting and preprocessing a set of features and, later, using those features as the input of a given algorithm. Such algorithms are then used to predict the probability of the occurrence of anomalies.

The proposed framework supports those techniques by using the concept of a common ML-based pipeline that can be easily extended and applied to multiple anomaly detection tasks. The proposed anomaly detection approach (described below) uses the messaging system and the data analytics layer to create a complete pipeline for ML-based algorithms.

Figure 6.5 provides an overview of the proposed ML pipeline. The first step, referring to the training stage, starts by loading a previously acquired dataset (containing a list of features). The initial dataset is then split into two additional datasets (for training and validating). Moreover, an optional *GridSearch* step can be used to optimize the ML model parameters. Later, using the training dataset and leveraging the Spark ML API, the classification model is fitted (trained).

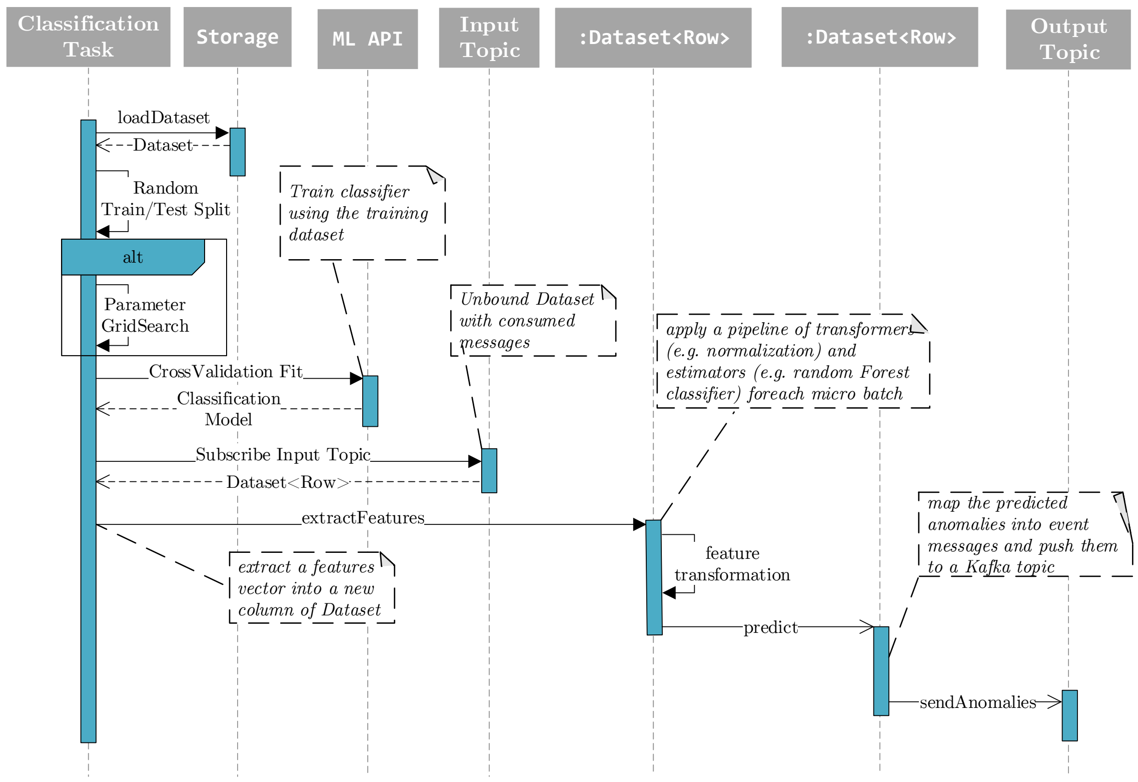

Classification Task
Storage
ML API
Input Topic
:Dataset<Row>
:Dataset<Row>
Output Topic
loadDataset
Dataset
Random Train/Test Split
Train classifier using the training dataset
alt
Parameter GridSearch
Unbound Dataset with consumed messages
apply a pipeline of transformers (e.g. normalization) and estimators (e.g. random Forest classifier) foreach micro batch
CrossValidation Fit
Classification Model
Subscribe Input Topic
Dataset<Row>
map the predicted anomalies into event messages and push them to a Kafka topic
extractFeatures
feature transformation
extract a features vector into a new column of Dataset
predict
sendAnomalies

Figure 6.5: Proposed ML-based anomaly detection pipeline based on Spark APIs.

For the prediction step (i.e. the classification of the events as normal or anomalous), a *Kafka Source* is configured to subscribe to multiple domain level topics, using the Spark Structured Streaming API (that supports different types of input sources). Each Kafka message (`K V` pair) coming from the domain processors is directly mapped into a `Row` of an unbounded Spark `Dataset<Row>`. The `Row` contains a list of columns including, amongst others, both `K` and `V` encoded as binary values, as well as the topic and partition metadata.

Plain Avro messages can be directly decoded at this step to primitive and complex Spark column types, based on the Avro schema. For a common Avro variant used by Confluent Kafka Registry [Confluent, 2020], where each message contains also the schema management information, an extra conversion step was necessary. Additionally, a custom lookup function is also used for converting known feature pairs (based on *meaning* / *content* field entries) and saving them as an additional `Vector` column of the initial `Dataset<Row>`.

Then, by leveraging the Spark ML API, each feature vector goes through a pipeline, meaning a chain of *transformers* (e.g. one-hot-encoding) and an *estimator* (a previously trained classification algorithm). The classification result is also stored into a *label* column. Finally, for each detected anomaly, a new message is derived and pushed back to a Kafka topic using a *Kafka sink*.

One of the main advantages of such a ML pipeline is that it can be reused in the context of different types of features and ML algorithms. The next section presents and discusses a concrete use case based on a scenario of data exfiltration to show how to leverage such a ML pipeline concept for anomaly detection.

6.3 The Use Case of Data Exfiltration

The use case presented next details a data exfiltration scenario based on Domain Name System (DNS) tunnels. In SCADA systems, data-exfiltration techniques are relevant because they allow the attacker to stealthily exfiltrate confidential data such as PLC's ladder logic, process data or operators credentials.

Whereas DNS is not a SCADA-specific protocol, it is a real threat to IACS. This generic protocol is used to map domain names into IP addresses, and is generally present in SCADA networks. Moreover, being a fundamental network protocol used by several services, it is often not blocked by firewalls. DNS tunnels are one of the more typical data exfiltration techniques, not just in IACS domains but also in general [Alcaraz et al., 2019].

Such data exfiltration techniques are difficult to detect by signature-based mechanisms. For this reason, they constitute a good reference use case to showcase and explore the benefits of ML-based techniques in the overall proposed framework.

Data exfiltration techniques typically rely on the establishment of side channels to Command and Control servers (C&C). Later, these channels are used to remotely manage the compromised hosts and to exfiltrate sensitive data. DNS tunneling is one of the techniques that allow creating a two-way communication channel by encoding data on both DNS queries (as part of the queried domain name) and DNS responses (in the resource record data). Various ML-based approaches have been proposed in the literature for detecting DNS tunneling, mainly using payload analysis (based on packet header fields) or traffic analysis (based on DNS session metrics) [Yassine et al., 2018] [Nadler et al., 2019] [Nuojua et al., 2017] [Yu et al., 2016]. The remaining of this section discusses how those techniques can be leveraged in the proposed framework.

Figure 6.6 illustrates the setup used to recreate a data exfiltration scenario using DNS tunnels. On the attacker's side, a server was setup in the cloud to behave as an authoritative

DNS server for two previously registered domains. Then, two popular DNS tunneling tools (dnscat2 [Bowes, 2019] and iodine [Ekman and Andersson, 2019]) were used to produce 23 scenario variations, including simple tunnel handshakes using different DNS record types to encrypted sessions, interactive shells, and the exfiltration of a complete PLC project.

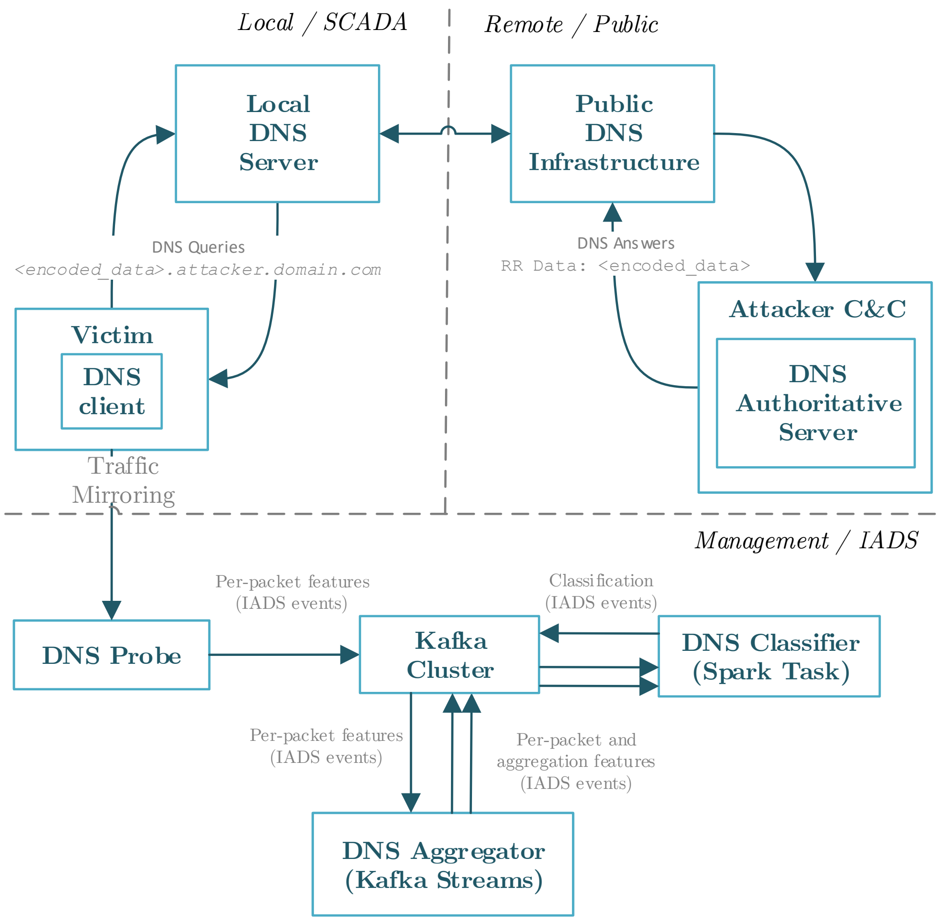

Local / SCADA
Remote / Public
Local DNS Server
Public DNS Infrastructure
DNS Queries
<encoded_data>.attacker.domain.com
DNS Answers
RR Data: <encoded_data>
Victim
DNS client
Attacker C&C
DNS Authoritative Server
Traffic Mirroring
Management / IADS
Per-packet features (IADS events)
Classification (IADS events)
DNS Probe
Kafka Cluster
DNS Classifier (Spark Task)
Per-packet features (IADS events)
Per-packet and aggregation features (IADS events)
DNS Aggregator (Kafka Streams)

Figure 6.6: DNS tunneling scenario

On the detection side, to showcase the different anomaly detection capabilities, a DNS probe was used for extracting packet-specific features (as described below) from the DNS network traffic (this DNS probe, conceived in the context of the ATENA project, was not developed by the author of this thesis). Then, a domain processor capable of extracting additional features, based on the aggregation of the messages into time windows, was created. Finally, an anomaly detection pipeline was created for training/classifying DNS traffic as normal or abnormal (i.e. a possible attempt of DNS tunneling).

Three datasets resulted from the experiments (cf. Table 6.1). Dataset *DS1* contains over 15,000 records (one per each DNS packet), including all the anomalous and normal DNS traffic related with software updates, network services and arbitrary DNS requests. Dataset *DS2* contains the aggregated features – one record per each time-based window – using hopping windows of 30 seconds with 5 seconds hops. Dataset *DS3* contains 25 features derived from DS1, including the average and the standard deviation for each feature on DS1, as well as the number of packets per window.

Table 6.1: DNS tunneling datasets

	Number of Records		
Class	DS1	DS2*	DS3*
Normal	8458	868	868
Anomaly	7410	276	276
Total	**15868**	**1144**	**1144**

*Original data before oversampling using SMOTE technique.

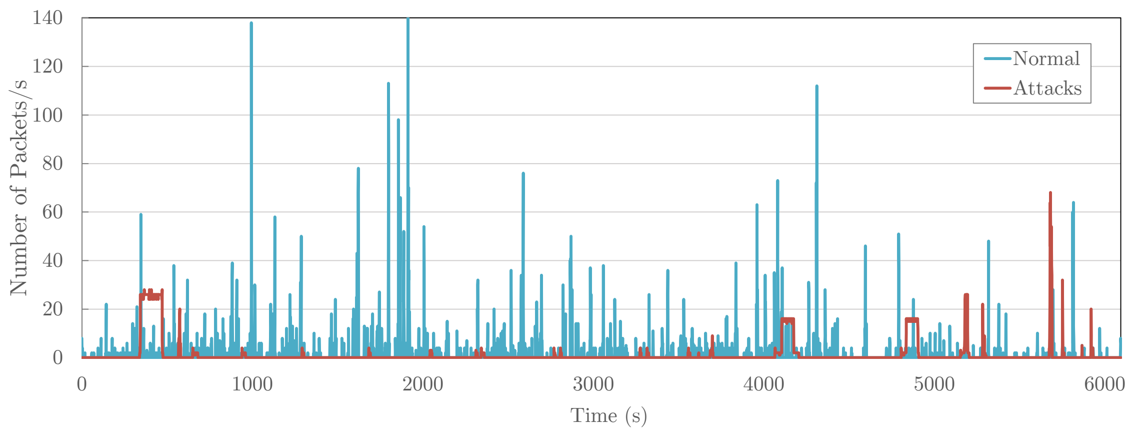


Figure 6.7: Distribution of network packets over time

In order to ensure a more balanced distribution between the number of normal and anomalous events, the datasets *ds2* and *ds3* were oversampled using the SMOTE technique [Chawla et al., 2002],

Despite being originated from the same scenario, the three datasets are fundamentally different and were used to feed different anomaly detection approaches. There are trade-offs between them. *DS1* derives from a larger dataset where each record, containing individual features of a single network packet, can be processed in near real-time. This means less computation on the preprocessing step but more at the classification algorithm. In *DS2*, only aggregated events are used to train the anomaly detection system, meaning more computation at the domain processing but fewer events to classify in the upper layers. Moreover, there is a distinction between *DS2* and *DS3*. *DS3* uses DNS-specific aggregated features, whereas *DS2* pursues a more generic approach (that can be applied to other protocols, including SCADA protocols) by using only generic network-level features. Whereas the final accuracy in both cases is totally dependent on the chosen features, a single packet might be insufficient to represent a cyber-attack. On the other hand, an aggregated record might be enough to flag an unusually large amount of network flows but might hide information behind all the underlying aggregation statistics for slow, low-intensity attacks. Both *DS2* and *DS3* used overlapping time windows to reduce the latency between each new record, while maintaining larger aggregation time windows.

For anomaly detection, a machine learning approach was used to evaluate the performance of several supervised algorithms for binary classification, namely plain Decision Trees, Random Forests, Gradient-boosted trees (GBT)s, Multilayer Perceptron classifier (MLPC), linear Support Vector Machine (SVM), Naiver Bayes, Logistic Regression, all natively supported by Spark ML API [Apache Spark, 2019], as well as two additional ML techniques more recently referred in the literature: XGBoost [Chen and Guestrin, 2016] and LightGBM [Ke et al., 2017].

Each record was labeled and marked as anomalous if either the request or response contains one of the malicious domain names – therefore, this feature was not considered for any algorithm. The full list of used features, in line with previous works [Yassine et al., 2018] [Nadler et al., 2019] [Nuojua et al., 2017] [Yu et al., 2016], is enumerated in Table 6.2. Their feature histograms, depicted in Figures 6.8, 6.9 and 6.10, show their pairwise distribution.

The chosen features are mainly focused on: (1) the size of both request and response packets – DNS tunnelling packets are typically larger because they carry extra information; and (2) how different the domain names are from normal DNS requests – since they are used to encode data, they are typically less natural than words used in normal DNS requests. Additionally, a binary feature was also included to indicate if the domain is listed in the Alexa top sites list [Alexa Internet, Inc, 2020], as an indicator of whether this is a common DNS name. Other common features, such as layer 2/3 fields or time-related measurements, were explicitly not used. This means this approach is not constrained to specific network details or limited to high-throughout attacks. Opposed to other research works, it was decided to

Table 6.2: Features list by dataset

DS1	**DS2**	**DS3**
Frame Size	Count	Count
Query length	Frame Size Mean	Mean of each feature of DS1
Query Shannon entropy	Frame Size Std	Std of each feature of DS1
Query char percentage	Frame Size Var	PIT Mean
Query non hex percentage	PIT Mean	PIT Std
Query Alexa top 1M	PIT Std	
Reply length	PIT Var	
Reply Shannon entropy		
Reply char percentage		
Reply non hex percentage		
Reply Alexa top 1M		

Packet Inter-arrival Time (PIT), Standard Deviation (Std), Variance (Var).

leave out the DNS record type. Although some record types are preferred over others for data exfiltration, using this feature for training purposes can result in inefficient models for classification of all DNS tunnel variants – the collected datasets contain a combination of several tunnelling samples using different record types. Features extracted from both queries and replies were included. Although there are some correlations between these features, since we might have data encoded on both directions or only one way, it was decided to not perform any feature reduction, due to the reduced number of items. Figures 6.11, 6.12 and 6.13 show the heat maps of the Pearson correlation coefficients between the features for each dataset, whereas Figures 6.14, 6.15 and 6.16 show the mutual information between each feature.

As described earlier in this chapter, the pipeline was developed on top of the dataframe-based Spark ML API. This allows a more user-friendly approach, by introducing the notion of *pipelines*, a set of *transformers* and *estimators* that can be applied directly to a dataframe to reassemble a complete workflow. For all the aforementioned algorithms, a random 70/30 split was adopted (where 70% of the dataset was used for training, validation and hyper-parameter tuning and the remaining 30% was used for the final testing). Depending on the specific algorithms (as detailed below), the first step of the pipeline is the feature transformation (e.g. scaling, one-hot encoding, etc.). The next step consisted of a 10 k-fold cross-validation combined with a grid parameter search where the best model of each algorithm was chosen, to maximize the area under the ROC curve (AUC). Finally, each model was tested using the remaining 30% of the dataset.

The performance of several tree-based approaches was evaluated, including plain Decision

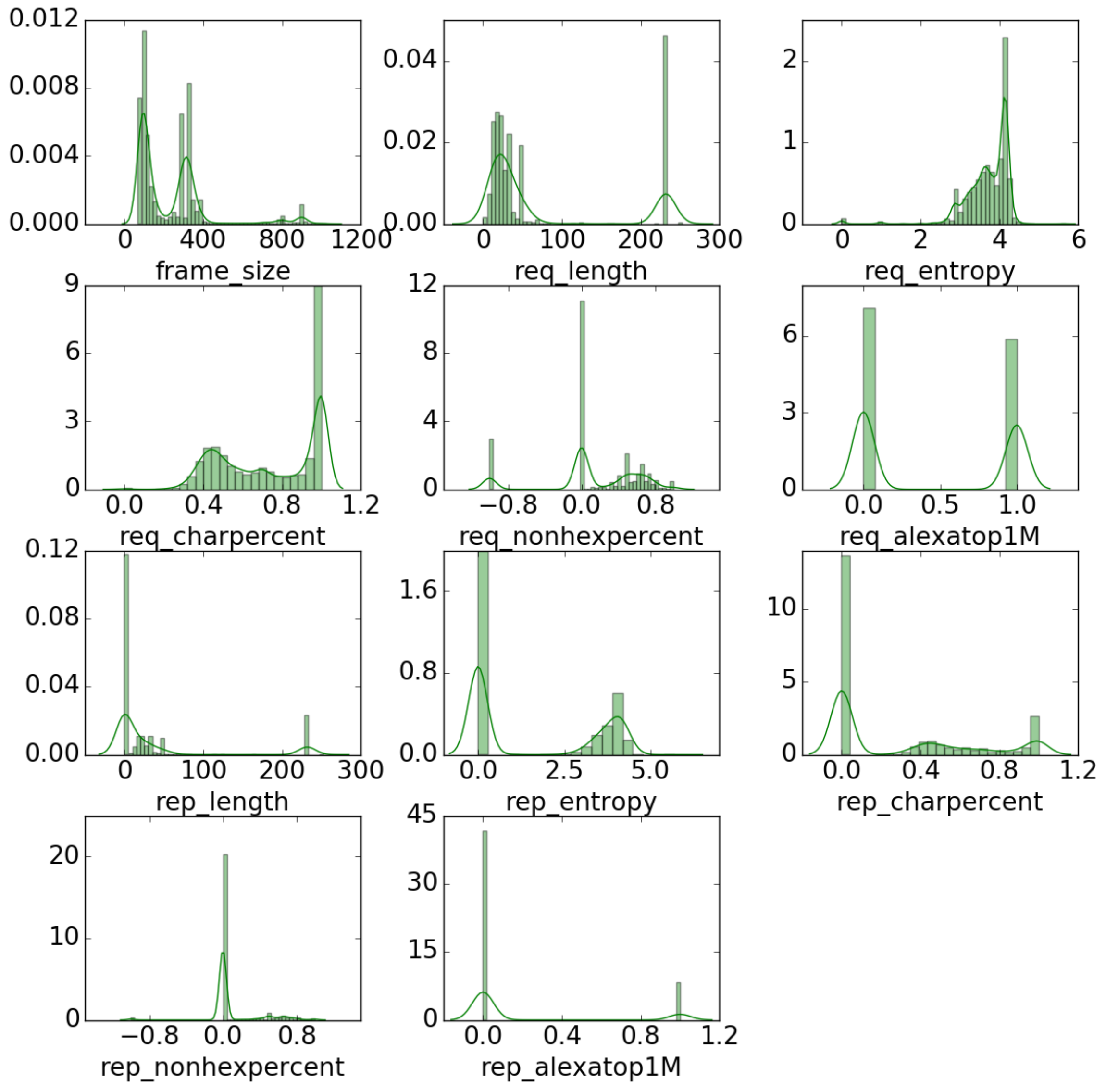


Figure 6.8: Features Histogram for Dataset DS1

Trees, Random Forests,Gradient-boosted trees (GBT) [Apache Spark, 2019], XGBoost [Chen and Guestrin, 2016] and LightGBM [Ke et al., 2017]. They are known to produce acceptable results without requiring a lot of feature engineering. Moreover, since the used implementations can already handle categorical features, no feature transformation was applied for those approaches.

For plain Decision Trees and Random Forests, the split was chosen to maximize the Information Gain (IG), according to Equation 6.1 [Apache Spark, 2019], where D represents the

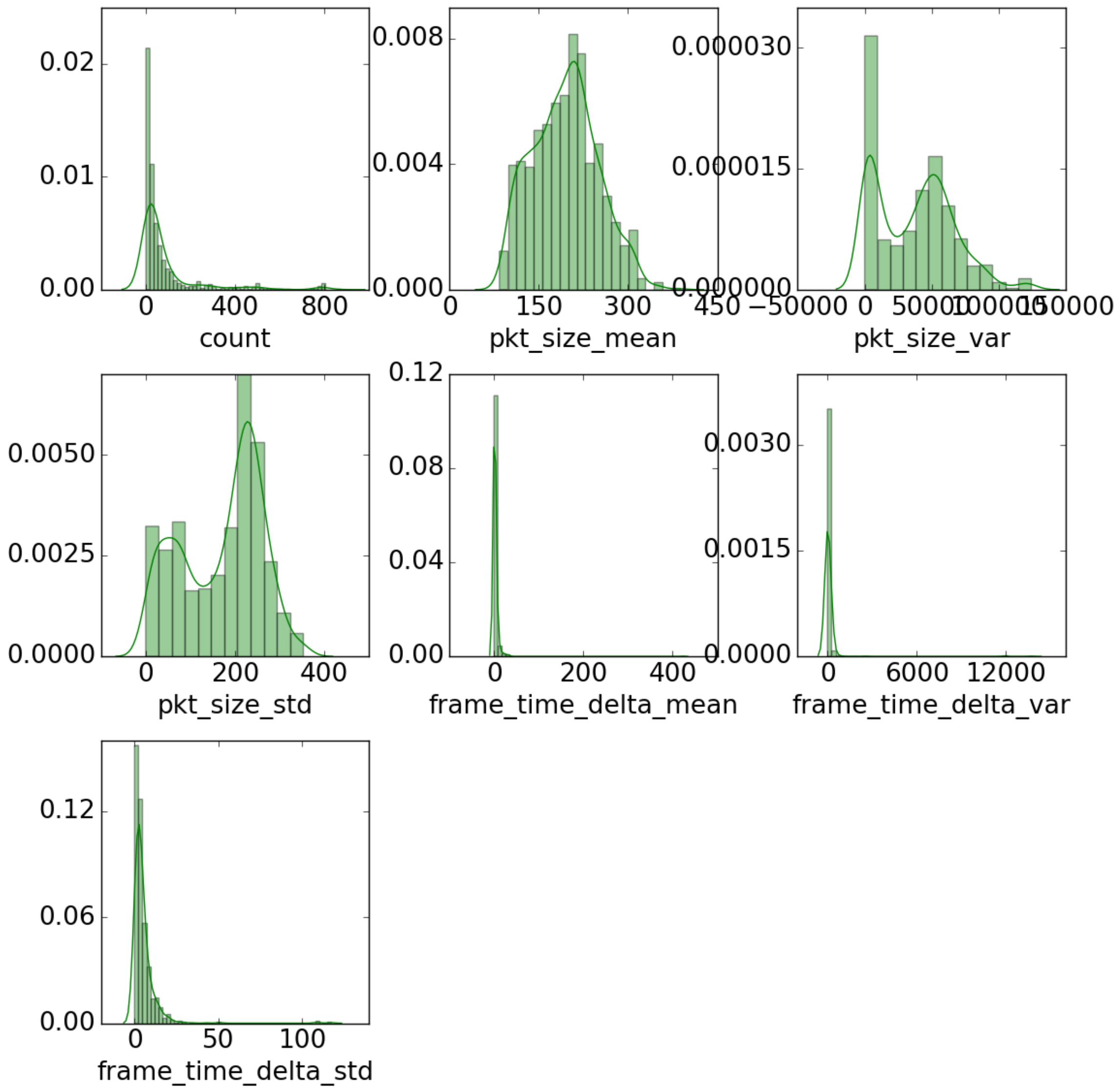


Figure 6.9: Features Histogram for Dataset DS2

dataset, s a split, and N the size of the dataset. The Impurity in each node was calculated using the Gini impurity Equation [Apache Spark, 2019]. Random forests, composed of several independent decision trees built from a random subset of data and features, are a

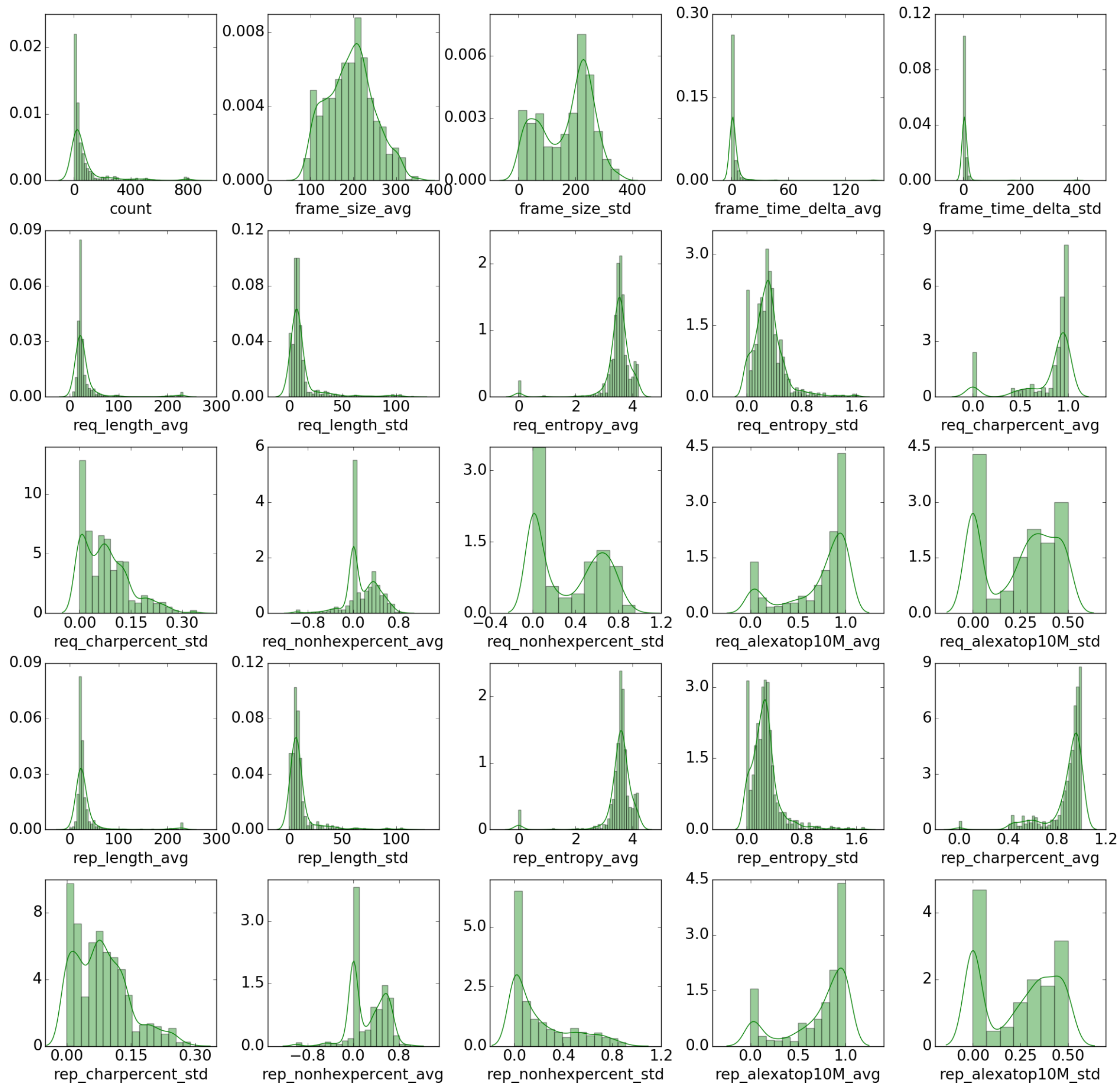


Figure 6.10: Feature Histogram for Dataset DS3

popular choice that typically helps reducing variance and over-fitting.

$$
\begin{aligned}
IG(D,s) = Impurity(D) \\
-\frac{N_{left}}{N} Impurity(D_{left}) \\
-\frac{N_{right}}{N} Impurity(D_{right})
\end{aligned}
\tag{6.1}
$$

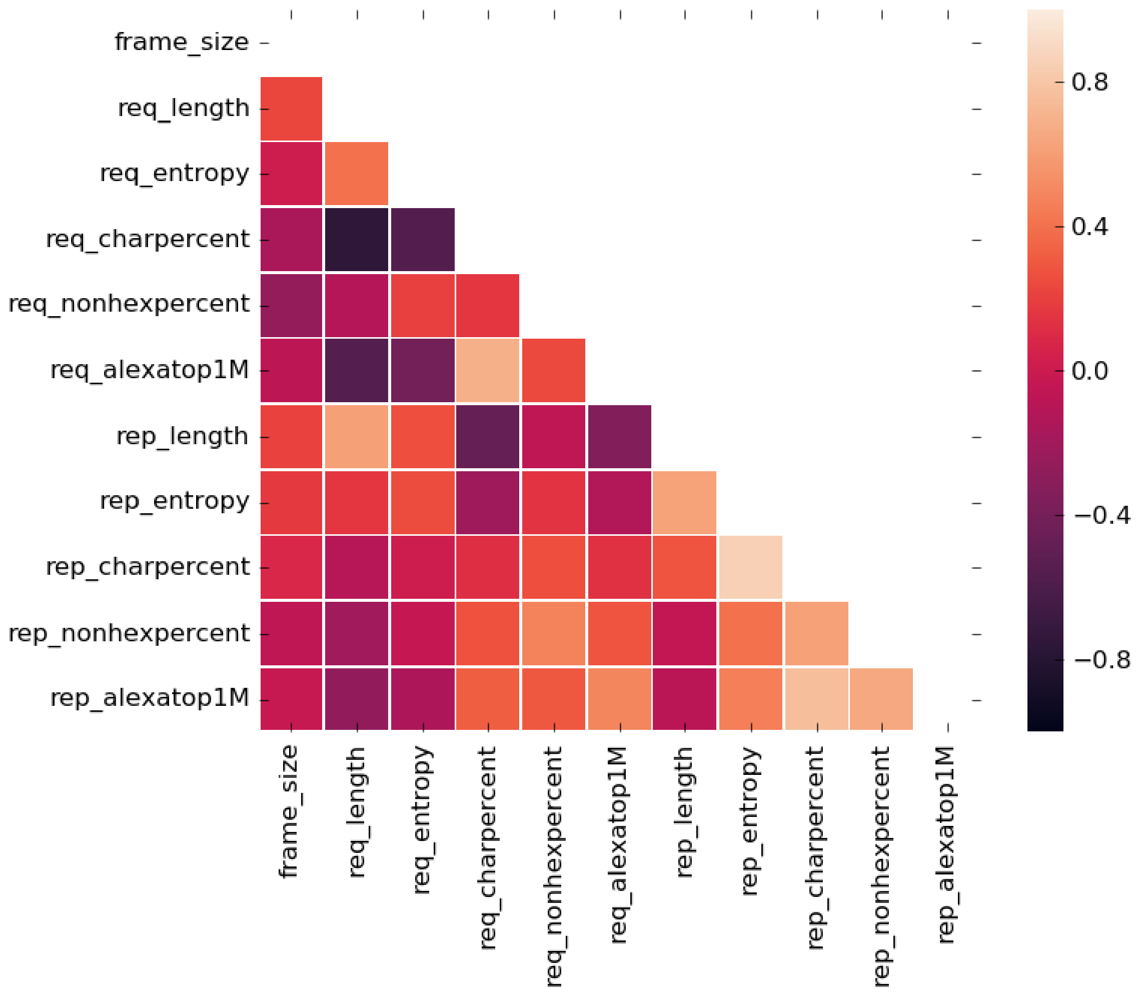


Figure 6.11: Pearson Correlation for DS1 Features

Opposed to random forests, GBTs use a sequential process where at each interaction they try to improve their result by incorporating the knowledge from the previous steps. The native Spark implementation was evaluated using a loss function according to equation 6.2 [Apache Spark, 2019]. For XGBoost and LightGBM, despite none of them being natively supported by Spark, both offer an easy integration path by using Spark ML Dataframe APIs. For XGBoost, the `binary:logistic` objective option [Chen and Guestrin, 2016] was used. For LightGBM, the `gbdt` boosting setting [Ke et al., 2017] was used.

$$2\sum_{i=1}^{N}\log(1+\exp(-2y_iF(x_i))) \tag{6.2}$$

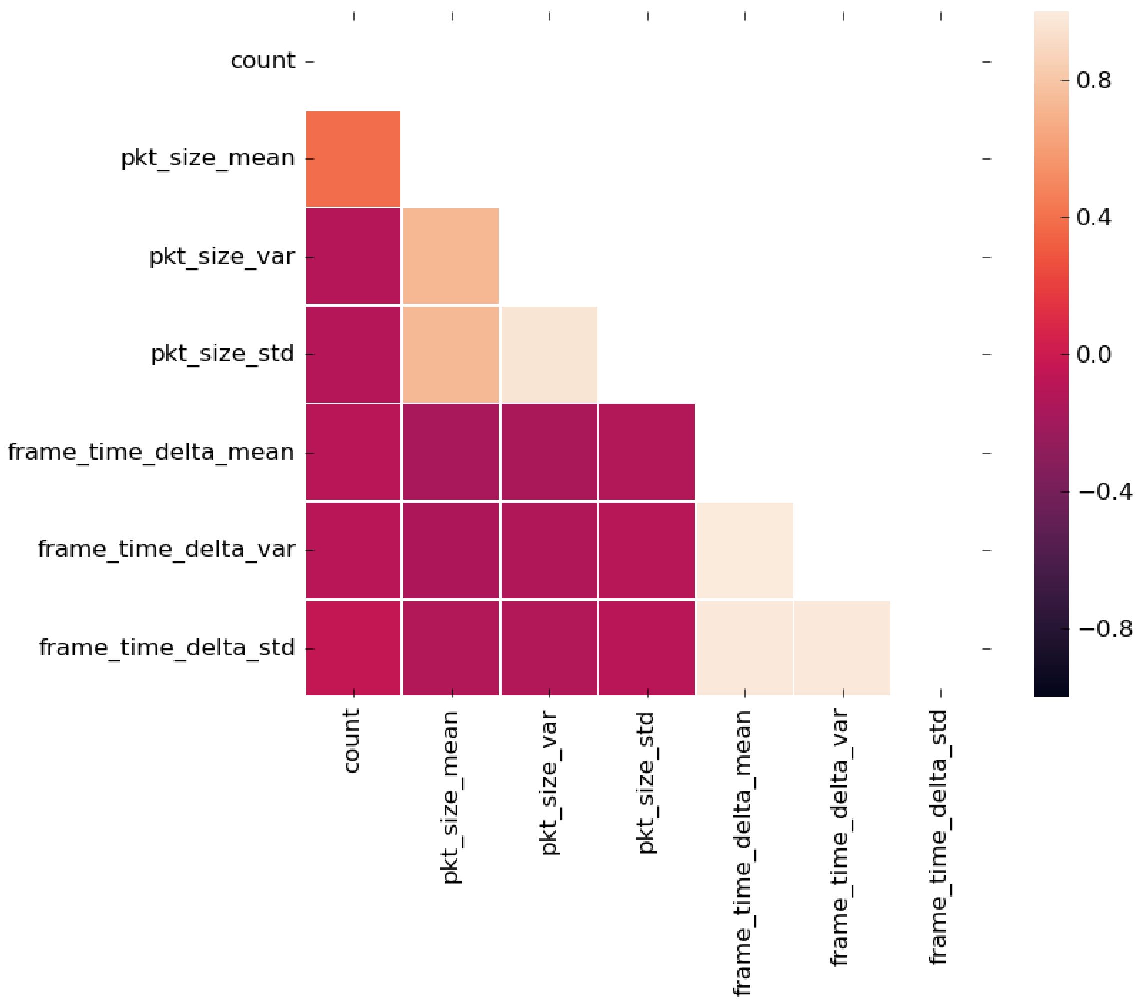


Figure 6.12: Pearson Correlation for DS2 Features

Feedforward artificial neural networks, based on the native Spark MLPC [Apache Spark, 2019], were also evaluated. Different network architectures were assessed, depending on the dataset. For DS1, 11, 7 and 2 were set as the number of nodes in the input, hidden and output layers, respectively. For DS2, the 7, 5 and 2 combination was used. Finally, for DS3, 25, 14 and 2 were used. The number of nodes within the hidden layer was chosen based on the average of the number of nodes in the input and output layers. The hidden layer uses the sigmoid activation function, while the output layer uses the softmax function [Apache Spark, 2019]. For the case of Naive Bayes, a feature scaler was used to transform the features between 0 and 1.

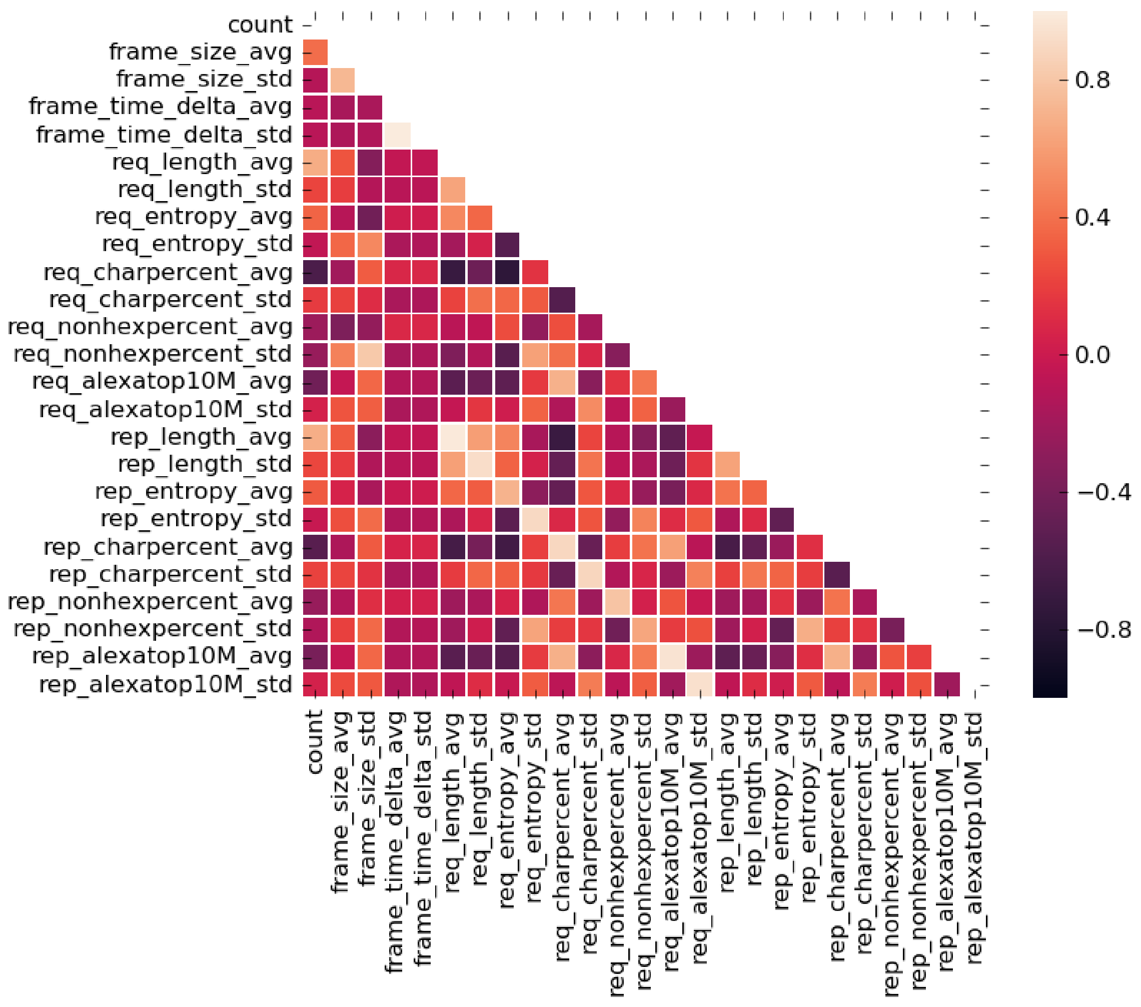


Figure 6.13: Pearson Correlation for DS3 Features

6.4 Evaluation and Performance Indicators

This section provides an overview of the performance indicators of each method against the different datasets, according to the following equations, where TP, TN, FP and FN stand for True Positives, True Negatives, False Positives and False Negatives, respectively:

$$Precision = \frac{TP}{TP + FN} \tag{6.3}$$

$$Recall = \frac{TP}{TP + FP} \tag{6.4}$$

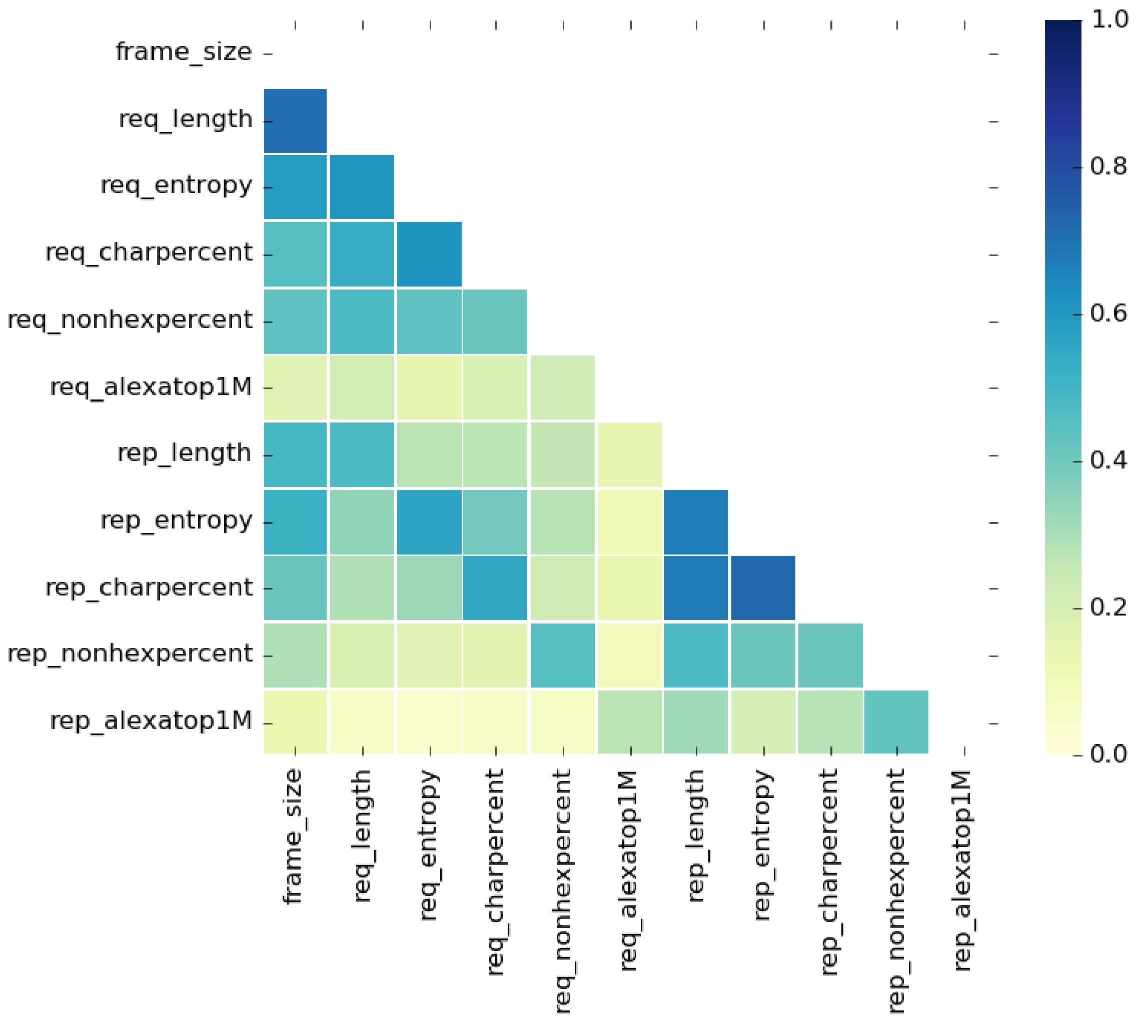


Figure 6.14: DS1 Feature Pairwise Mutual Information

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.5}$$

$$F1 = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{6.6}$$

Table 6.3 shows the results of the various machine learning algorithms assessed using DS1. All the methods yield results above 90% of accuracy. Whereas no algorithm outperforms the remaining in all the measured indicators, Gradient Boosting trees (namely XGboost and LightGBM) provided the best results. XGboost achieved a perfect precision score.

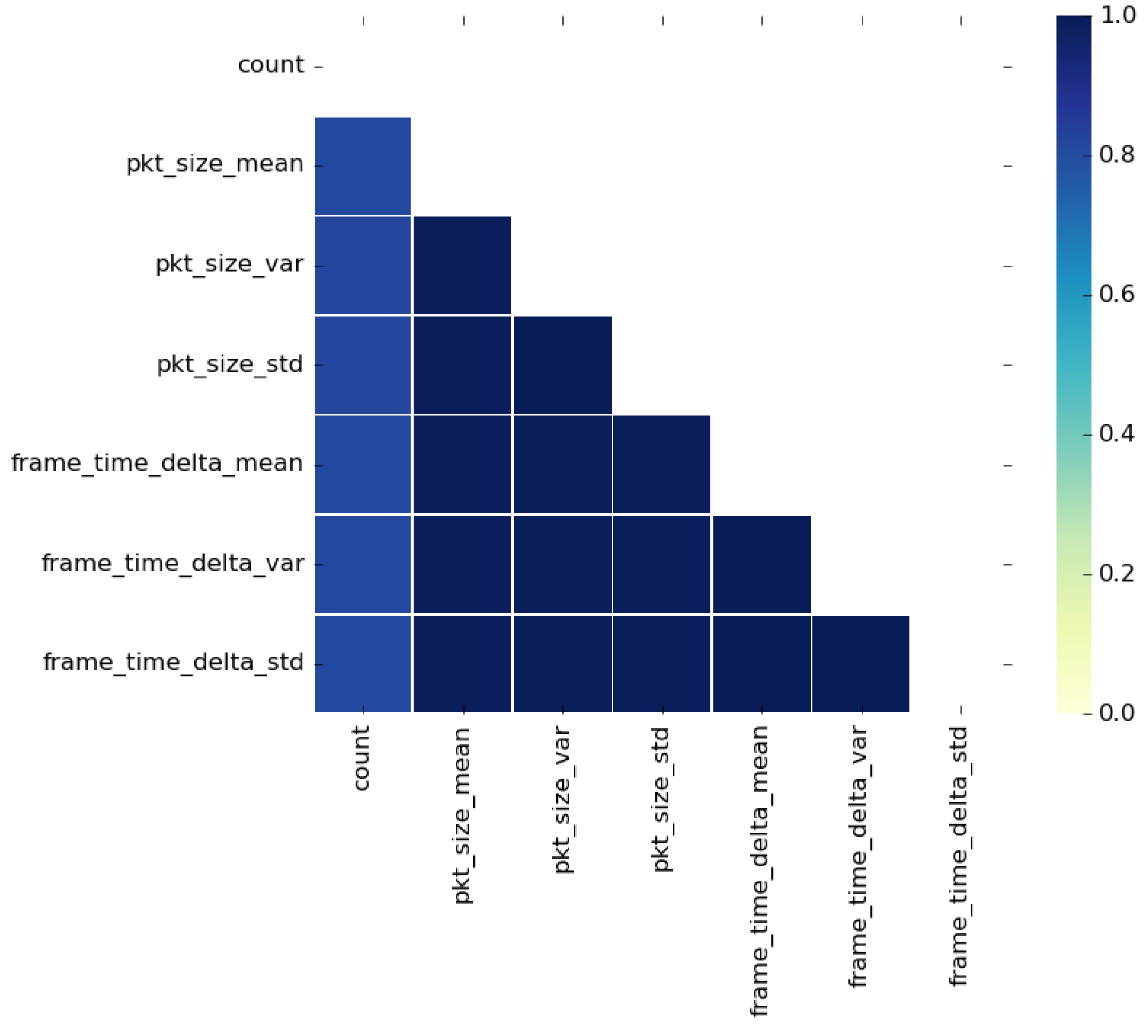


Figure 6.15: DS2 Feature Pairwise Mutual Information

Table 6.4 shows the results of the several machine learning algorithms assessed using DS2. As expected, the overall results were lower when compared to DS1. This can be explained by the reduced number of features and records of DS2. It is a trade-off between a less accurate but faster and more generic model. Since application layer-related features were not used, this approach remains valid for detecting large deviations in the traffic patterns of other protocols. Several methods surpass the 90% accuracy level. Tree-based methods obtained, once again, more consistent values – with an advantage for the Random Forests approach. In this low traffic scenario, a large amount of packets can be easily spotted. In larger networks, this might be less accurate or require additional fine-tuning of the time windows.

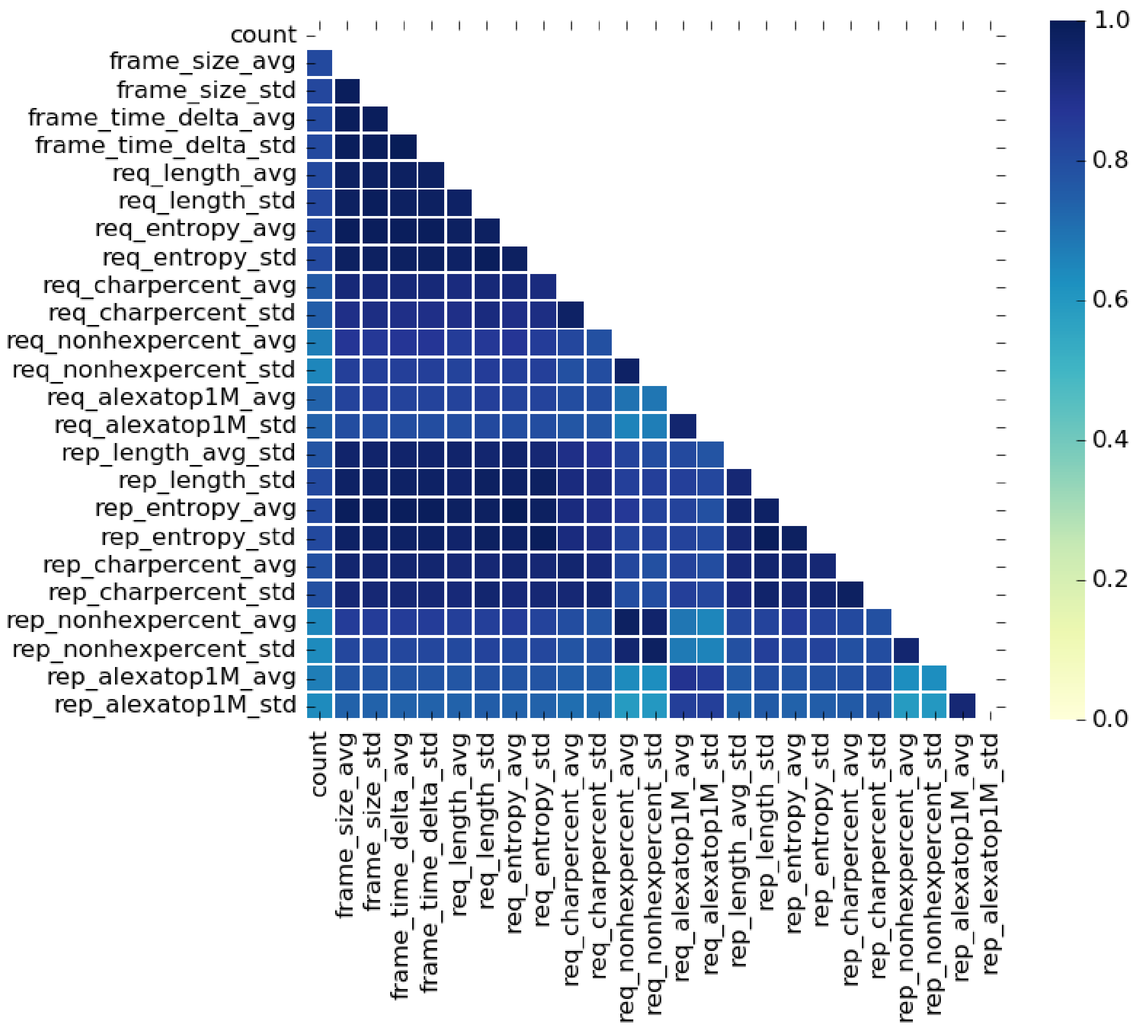


Figure 6.16: DS3 Feature Pairwise Mutual Information

Table 6.5 shows the results of the various machine learning algorithms assessed with DS3. By using a large number of features, including DNS-specific ones, there was an improvement when compared with DS2 results. As before, tree-based approaches show a slight overall advantage, with XGBoost reaching an Area under ROC Curve (AUC) of 0.986. Similarly to the DS1 results, it is also remarkable that all the algorithms obtained an AUC score above 0.9.

Figure 6.17 shows the average impact of each feature value (using the SHapley Additive exPlanation (SHAP) [Lundberg and Lee, 2017]) on the XGboost model using the DS3 dataset. As expected, the presence of the DNS name in the Alexa top sites list was among

Table 6.3: Summary of the key performance indicators for the DNS tunneling Detection using DS1

Technique	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.9914	0.9945	0.9872	0.9909	0.9912
Random Forests	0.9925	0.9982	0.9859	0.992	0.9921
GBT	0.9914	0.9945	0.9872	0.9909	0.9912
XGBoost	0.9989	1.0	0.9977	0.9989	0.9989
LightGBM	0.9989	0.9995	0.9982	0.9989	0.9989
Linear SVM	0.9776	0.9772	0.9754	0.9763	0.9775
MLPC	0.9895	0.9909	0.9868	0.9888	0.9893
NaiveBayes	0.9215	0.9115	0.9235	0.9174	0.9216
Logistic Regression	0.9202	0.9061	0.9271	0.9165	0.9206

Area under ROC (UAC).

Table 6.4: Summary of the key performance indicators for the DNS tunneling Detection using DS2

Technique	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.8955	0.8776	0.5811	0.6992	0.7798
Random Forests	0.9237	0.8615	0.7568	0.8058	0.8623
GBT	0.9068	0.7887	0.7568	0.7724	0.8516
XGBoost	0.9124	0.8525	0.7027	0.7704	0.8353
LightGBM	0.8983	0.8276	0.6486	0.7273	0.8065
Linear SVM	0.8927	0.95	0.5135	0.6667	0.7532
MLPC	0.8909	0.8909	0.6203	0.7313	0.7982
Naive Bayes	0.8079	1.0	0.0811	0.15	0.5405
Logistic Regression	0.8898	0.8302	0.5946	0.6929	0.7812

Area under ROC (UAC).

the most significant features. DNS exfiltration attacks mostly use dedicated DNS records that are unlikely to appear in a list of popular domains. Similarly, the average length of DNS responses also played a significant role in the classification. This depends on the underlying network traffic, since networks with more traffic might benefit the attacker by smoothing such values.

The training times are another important indicator to understand how the different algorithms

Table 6.5: Summary of the key performance indicators for the DNS tunneling Detection using DS3

Technique	Accuracy	Precision	Recall	F1	AUC
Decision Tree	0.9753	0.981	0.9699	0.9754	0.9753
Random Forests	0.9867	0.9887	0.985	0.9868	0.9867
GBT	0.981	0.9848	0.9774	0.9811	0.981
XGBoost	0.9886	0.9924	0.985	0.9887	0.9886
LightGBM	0.9734	0.9737	0.9737	0.9737	0.9734
Linear SVM	0.981	0.9923	0.9699	0.981	0.9811
MLPC	0.9867	0.9924	0.9812	0.9868	0.9868
Naive Bayes	0.9411	0.9916	0.891	0.9386	0.9416
Logistic Regression	0.981	0.9812	0.9812	0.9812	0.981

Area under ROC (UAC).

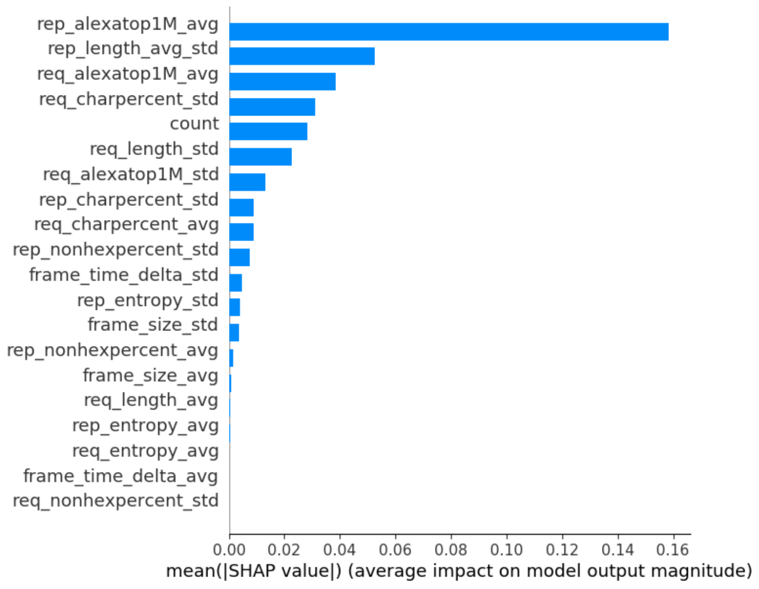


Figure 6.17: Feature impact (based on the SHAP value) for XGBoost Model using DS3 dataset

perform within the distributed computation platform. Such times are mainly influenced by the number of records, the algorithm itself, its parameters and the capacity of distributing the workload and run tasks on parallel, the computational resources available (e.g. Central Processing Unit (CPU), Graphics Processing Unit (GPU), Field-programmable gate array

(FPGA)) and the Spark cluster deployment setup (e.g. the number of Spark workers, the number of executors, number of cores per executor).

Table 6.6 represents the wall-clock time spent to train 70% of the original dataset (the training data) without any grid parameter search. These experiments were conducted using a 3-worker Spark cluster, each one running on a different virtual machine with 8 cores and 16GB RAM each, on a Dell PowerEdge R440 host with an Intel(R) Xeon(R) Gold 5120 CPU (28 vCPUs), 256GB RAM and 3.2TB datastore (4x10K RPM SAS HDD in RAID6 via PERC H740P controller), running an ESXi 6.7 hypervisor instance. For experimental purposes, the Spark deployment runs on the top of a Docker container based on official NVIDIA/CUDA images. This allowed keeping the results consistent across all the tests, including a GPU-based approach comparison.

Table 6.6: Summary of the training times (s)

Technique	DS1	DS2	DS3
Decision Tree	25.03	10.97	12.15
Random Forests	25.86	11.04	11.68
GBT	78.00	54.32	61.34
XGBoost	29.26	18.26	17.68
LightGBM	18.81	6.94	7.83
Linear SVM	173.84	56.18	52.60
MLPC	52.33	31.30	41.22
Naive Bayes	6.56	2.62	2.92
Logistic Regression	9.73	17.74	19.09

Given its greater number of records, training times with DS1 were generally longer than with DS2 and DS3. Such differences are expected to increase even more with larger datasets, and might become a deciding factor when selecting the most appropriate algorithms for a given use case. For the same number of records, in general DS2 lead to shorter training times, when compared with DS3, since it holds a significantly shorter number of features (25 for DS3 versus 7 for DS2). Naive Bayes and Logistic regression were among the fastest methods, but also the most inaccurate. Linear SVM was significantly slower without any meaningful advantage when compared to tree-based approaches. On the other hand, LightGBM obtained a good balance between the training times and classification performance.

A further experiment assessed the impact of available computational resources on the training time for larger datasets (using synthetic data generated from DS1). Figure 6.18

shows training times for XGBoost, considering datasets with different sizes. Two different CPU models (Intel Core i7-6700HQ and Intel Xeon Gold 5120, both using the `hist` XGBoost tree method) and a NVIDIA GeForce GTX 1060 using `gpu_hist` were considered.

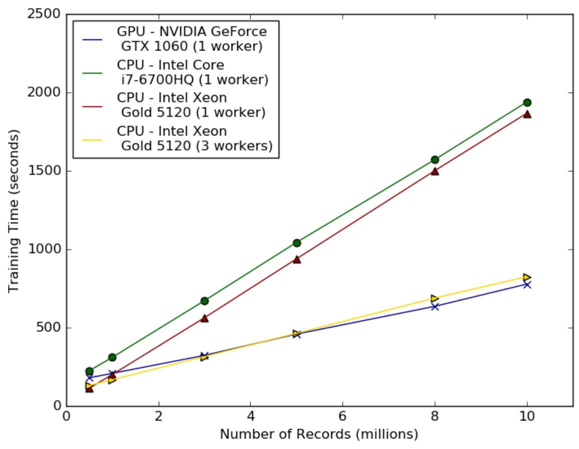


Figure 6.18: Comparison between CPU and GPU for XGBoost training times using synthetic data derived from DS1

For the Intel Core i7-6700HQ and NVIDIA GeForce GTX 1060 scenarios, tests were performed using a single worker and a single executor with 7 cores, plus one additional core for the driver. For the Intel Xeon Gold 5120 CPU, two scenarios were used: a single worker scenario with one executor; and a scenario with a 3 workers cluster with one executor per worker, in a total of 23 cores (plus one additional core for the driver).

For the smaller datasets, the Intel Xeon Gold 5120 CPU executing a single worker (thus avoiding additional synchronization overhead) provided the best results. As expected, as the number of records increases there is a clear advantage of pursuing a distributed approach with multiple workers or using a GPU-based approach. In our setup, after 5 million records, the NVIDIA GeForce GTX 1060 outperformed all the remaining approaches.

More than trying to determine what are the most appropriated algorithms for each of the possible types of of anomaly and cyber-attacks within a IACS environment, the conducted experiments show the benefits of the proposed approach in terms of easily integrating different computation and anomaly detection techniques, its flexibility in supporting different distribution models (edge/domain processing vs. global processing), and different performance and scalability requirements.

6.5 Summary

The proposed data analytics layer was inspired by the concept of an evolved Security Information and Event Management (SIEM), more geared towards Big Data and ML-based anomaly detection mechanisms. Within a highly distributed IACS environment, such a layer can be used to efficiently process cyber-physical events at a global level, as well as to detect anomalies that cannot be easily spotted by signature or threshold-based mechanisms. Moreover, as discussed before, the proposed approach mainly relies on the usage of general-purpose distributed computation frameworks which, from a security standpoint, can be leveraged to implement all sorts of anomaly detection algorithms in a more efficient and flexible manner.

As a reference, the entire process of detecting a data ex-filtration operation – a real threat for IACS environments – was discussed. This use case highlights the benefits of the analytics layer to detect complex threats, and demonstrates the flexibility of the proposed approach in what relates with incorporating and combining different types of algorithms. The focus was not on the specific algorithms, but on understanding how distinct mechanisms could be integrated in the proposed framework.

Regarding the proof-of-concept implementation, Apache Spark has proven to be a good match for developing such an analytics layer, enabling efficient and distributed computation capabilities. Besides performance, the Spark rich feature set and its APIs for third-party integration have also proven to be a good choice, supporting the idea of a unified approach for both streaming and batch processing.

7 Conclusions

7.1 Synthesis

The main objectives of this thesis were introduced in Chapter 1 and are shortly revisited in this section.

The underlying research question of this thesis was *how to improve the security of next-generation of Industrial Automation and Control Systems (IACS) through a holistic data-driven framework.*

First, an analysis of related works addressing the cyber-security of IACS was provided (cf. Chapter 2). The biggest perceived gap in the literature relates with the challenge of combining the wide array of data sources, probes, security components and intrusion detection systems that characterize the next generation of IACS. To fulfill that gap, recent advances on the fields of Big Data and event processing were analysed, with the purpose of assessing their suitability to IACS security. This work is aligned with the first objective of this thesis, as stated in Section 1.2: *review, identify and assess intrusion and anomaly detection algorithms and techniques potentially suitable for IACS.*

Next, several practical experiments and attack scenarios were devised and implemented in the testbed of an energy operator, to explore different cyber-security scenarios and IACS protocols, in order to better understand the dynamics of cyber-attacks and the vulnerabilities of Supervisory Control And Data Acquisition (SCADA) tools and protocols. These experiments were also used as part of the evaluation and demonstration activities of two European research projects. Overall, this work, described in Chapter 4, aligns with the second objective of this thesis: *explore different cyber-security scenarios and IACS protocols.*

Finally, as part of the third objective ("*the design and evaluation of a holistic data-driven framework*"), a data-driven and holistic framework was proposed for monitoring the cyber-security of next-generation IACS. The various threats surrounding the complexities of a IACS system demand a more comprehensive, modular, flexible and distributed approach, able to combine multiple techniques and deployment scenarios, according to the specific needs of each IACS. The concept of a data-driven and holistic monitoring framework was proposed (cf. Chapter 3) and its two key components (the streaming layer and the data analytics layer) were presented and evaluated in more detail (Chapter 5 and Chapter 6, respectively). This framework provides a powerful way of decoupling data collection from data processing, domain-wise and location-wise, while still enabling local and global processing (which complement each other) and supporting a wide range of distribution, scalability and performance requirements. Moreover, as shown in Chapter 6, different anomaly detection techniques can be easily integrated into this framework, making it possible to select the best sets of techniques for each specific IACS and each use case.

7.2 Main Contributions

The work conducted in this thesis has resulted in the following contributions.

- **Contribution 1. Security analysis of SCADA protocols in the scope of practical attack scenarios.** This contribution, reflected mostly in Chapter 4, focused first on the practical exploration of attack scenarios using the Modbus protocol to create disruption in the target IACS (an electrical grid). Based on widely known vulnerabilities, this line of work focused on discovering, from the attackers perspective, how SCADA systems can be effectively exploited from a practical standpoint, to create incidents on electrical grid scenarios.

  Next, a detailed assessment of PCOM was conducted, from a security standpoint. The lack of previous literature about the security of PCOM motivated a more ambitious analysis, from both the attacker's and the defendant's points of view, in order to raise awareness and show how less known SCADA protocols can be used to conduct attacks in IACS. This work eventually led to several contributions to open source tools such as Wireshark, Snort, Metasploit, Nmap and Scapy, which can be useful to infrastructure operators, security pentesters, auditors and researchers alike.

- **Contribution 2. Conceptualization and design of an holistic data-driven framework for intrusion and anomaly detection in IACS scenarios.** The second contribution provided by this thesis refers to the conceptualization and the design of a data-driven, holistic framework framework for Intrusion and Anomaly Detection System (IADS). By contrast to other works, this thesis focuses on the strategy for combining the data and the knowledge from different sources into a more comprehensive IADS approach. The advantage of having different types of specific security mechanisms, at both local and global levels, is a key driving motivation for this work. The proposed framework combines local security probes, event pre-processing mechanisms and anomaly detection techniques which, altogether, enable a broader situational awareness of the security of the IACS and support its monitoring as a whole. This is a particular relevant aspect in the context of next-generation IACS. As we rapidly move into more distributed and highly complex environments, it is important to have the means to understand and verify the correct behavior of the entire infrastructure Moreover, the proposed approach also supports early detection of different types of cyber-physical anomalies, by supporting (near) real-time collection and processing of events. This contribution is presented in Chapter 3, Chapter 5 and Chapter 6.

- **Contribution 3. Integration and evaluation of different mechanisms for classifying network traffic.** The third main contribution of this thesis is the integration and evaluation of different Machine-Learning (ML)-based mechanisms to detect anomalies in IACS environments. Part of this work is reflected in the domain processors (cf. Section 5.3), which support the preprocessing of events generated from the probes and extract additional statistics on a time-windows fashion – therefore enabling efficient processing at edge or domain level, limiting the amount of data that needs to be pushed to the (global) data analytic layer.

  The other part of this contribution relates with the integration and evaluation of different ML-based techniques, in the scope of the data exfiltration use case (cf. Sections 6.3 and 6.4). Previously produced statistics per time windows (coming from the edge/domain processors) were processed at a global level to detect potential deviations in network traffic. This processing was made by using a pipeline for streamlining the integration of different anomaly techniques for detecting anomalies. An evaluation of those techniques was performed, based on the data exfiltration use case. This evaluation highlights the benefits of the proposed approach in terms of easily integrating different computation and anomaly detection techniques.

7.3 Future Work

This thesis focused on the design of the data-driven IADS. Still, the complexity of a IACS environment is one of the biggest challenges for the anomaly detection process. The feasibility and practical evaluation of additional SCADA-specific algorithms proposed in the literature is something that needs further work. A commercial product based on the proposed framework would be useless without a proper set of algorithms suitable for different domains and different deployment scenarios.

Moreover, even though this thesis argues that SCADA security is not only about network communication protocols, most of the evaluation work focused on network-based scenarios – due to logistic and practical reasons. Nevertheless, it would be interesting to explore other types of data sources, such as host-based events, useful for instance to monitor HMIs and other SCADA workstations. It would also be interesting to explore more process-specific data, as well as other protocols from the field network (similarly to what was made with PCOM).

Finally, there are additional security trends which are getting more popular now and might be worth exploring. The Zero Trust model [Greenwood, 2021] [Rose et al., 2019] for the security of the infrastructure is a promising approach, but it remains to be seen how it holds in a SCADA domain. Likewise, the advances in privacy-preserving mechanisms such as Fully Homomorphic Encryption [Pulido-Gaytan et al., 2020] can be increasingly relevant in the future to ensure the privacy of external data computation (e.g. to allow the execution of anomaly detection tasks on the top of sensitive data on a third-party provider). The MLOps concept, intended to provide more agile approaches to the ML-based life cycle, also deserves further research in the specific domain of IACS.

References

[5G-ACIA, 2019] 5G-ACIA (2019). A 5g traffic model for industrial use cases. `https://www.5g-acia.org/fileadmin/5G-ACIA/Publikationen/5G-ACIA_White_Paper_Traffic_Model/WP_5G_5G_Traffic_Model_for_Industrial_Use_Cases_22.10.19.pdf`.

[Abadi et al., 2015] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. `https://www.tensorflow.org/`.

[Adepu et al., 2018] Adepu, S., Kandasamy, N. K., and Mathur, A. (2018). Epic: An electric power testbed for research and training in cyber physical systems security. In *Computer Security*, pages 37–52. Springer.

[Agrawal and Agrawal, 2015] Agrawal, S. and Agrawal, J. (2015). Survey on anomaly detection using data mining techniques. *Procedia Computer Science*, 60(1):708–713. `http://dx.doi.org/10.1016/j.procs.2015.08.220`.

[Ahmed et al., 2017] Ahmed, C. M., Palleti, V. R., and Mathur, A. P. (2017). Wadi: a water distribution testbed for research in the design of secure cyber physical systems. In *Proceedings of the 3rd International Workshop on Cyber-Physical Systems for Smart Water Networks*, pages 25–28.

[Al-Hawawreh et al., 2019] Al-Hawawreh, M., den Hartog, F., and Sitnikova, E. (2019). Targeted ransomware: A new cyber threat to edge system of brownfield industrial internet of things. *IEEE Internet of Things Journal*, 6(4):7137–7151.

[Alcaraz et al., 2019] Alcaraz, C., Bernieri, G., Pascucci, F., Lopez, J., and Setola, R. (2019). Covert channels-based stealth attacks in industry 4.0. *IEEE Systems Journal*, 13(4):3980–3988.

[Alexa Internet, Inc, 2020] Alexa Internet, Inc (2020). Alexa - top sites. `https://www.alexa.com/topsites`.

[Allen-Bradley, 2018] Allen-Bradley (2018). Logix 5000 controllers data access. `https://literature.rockwellautomation.com/idc/groups/literature/documents/pm/1756-pm020_-en-p.pdf`.

[Aniello et al., 2011] Aniello, L., Lodi, G., and Baldoni, R. (2011). Inter-domain stealthy port scan detection through complex event processing. In *Proceedings of the 13th European Workshop on Dependable Computing*, pages 67–72.

[Antón et al., 2017] Antón, S. D., Fraunholz, D., Lipps, C., Pohl, F., Zimmermann, M., and Schotten, H. D. (2017). Two decades of scada exploitation: A brief history. In *Application, Information and Network Security (AINS), 2017 IEEE Conference on*, pages 98–104. IEEE.

[Anton et al., 2017] Anton, S. D., Fraunholz, D., Lipps, C., Pohl, F., Zimmermann, M., and Schotten, H. D. (2017). Two decades of SCADA exploitation: A brief history. In *2017 IEEE Conference on Application, Information and Network Security (AINS)*, pages 98–104, Miri. IEEE. `http://ieeexplore.ieee.org/document/8270432/`.

[Anton et al., 2019] Anton, S. D. D., Sinha, S., and Schotten, H. D. (2019). Anomaly-based Intrusion Detection in Industrial Data with SVM and Random Forests. `http://arxiv.org/abs/1907.10374`.

[Apache Software Foundation, 2020a] Apache Software Foundation (2020a). Apache cassandra. `https://cassandra.apache.org/`.

[Apache Software Foundation, 2020b] Apache Software Foundation (2020b). Apache kafka. `https://kafka.apache.org/`.

[Apache Software Foundation, 2020c] Apache Software Foundation (2020c). Apache kafka documentation. `https://kafka.apache.org/documentation/`.

[Apache Software Foundation, 2020d] Apache Software Foundation (2020d). Apache spark™ - unified analytics engine for big data. `https://spark.apache.org/`.

[Apache Software Foundation, 2020e] Apache Software Foundation (2020e). Welcome to apache avro! `https://avro.apache.org/`.

[Apache Spark, 2019] Apache Spark (2019). Machine learning library (mllib) guide. `https://spark.apache.org/docs/2.4.4/ml-guide.html`.

[Assante and Lee, 2015] Assante, M. J. and Lee, R. M. (2015). The industrial control system cyber kill chain. *SANS Institute InfoSec Reading Room*, 1.

[Basumallik et al., 2019] Basumallik, S., Ma, R., and Eftekharnejad, S. (2019). Packet-data anomaly detection in PMU-based state estimator using convolutional neural network. *International Journal of Electrical Power and Energy Systems*, 107(June 2018):690–702. `https://doi.org/10.1016/j.ijepes.2018.11.013`.

[Berners-Lee et al., 2005] Berners-Lee, T., Fielding, R. T., and Masinter, L. (2005). Uniform resource identifier (uri): Generic syntax. STD 66, RFC Editor. `http://www.rfc-editor.org/rfc/rfc3986.txt`.

[Biondi, 2020] Biondi, P. (2020). Scapy: the python-based interactive packet manipulation program. `https://scapy.readthedocs.io/`.

[Borges de Freitas et al., 2018] Borges de Freitas, M., Rosa, L., Cruz, T., and Simões, P. (2018). Sdn-enabled virtual data diode. In *Katsikas S. et al. (eds) Computer Security. SECPRE 2018, CyberICPS 2018. Lecture Notes in Computer Science, vol 11387.*, pages 102–118. Springer, Cham.

[Bosch et al., 1991] Bosch, R. et al. (1991). Can specification version 2.0. *Rober Bousch GmbH, Postfach*, 300240:72.

[Bowes, 2019] Bowes, R. (2019). dnscat2. `https://github.com/iagox86/dnscat2`.

[Byzek, 2019] Byzek, Y. (2019). Optimizing your apache kafka deployment. `https://cdn.confluent.io/wp-content/uploads/Optimizing_Your_Apache_Kafka_Deployment_White_Paper.pdf`.

[Capgemini Research Institute, 2019a] Capgemini Research Institute (2019a). Intelligent automation in energy and utilities. `https://www.capgemini.com/wp-content/uploads/2019/05/Digital-Report-%E2%80%93-Automation-in-Utilities-2.pdf`.

[Capgemini Research Institute, 2019b] Capgemini Research Institute (2019b). Reinventing cybersecurity with artificial intelligence. `https://www.capgemini.com/wp-content/uploads/2019/07/AI-in-Cybersecurity_Report_20190711_V06.pdf`.

[Casado and Younas, 2015] Casado, R. and Younas, M. (2015). Emerging trends and technologies in big data processing. *Concurrency and Computation: Practice and Experience*, 27(8):2078–2091.

[Chawla et al., 2002] Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

[Chen and Guestrin, 2016] Chen, T. and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 785–794, New York, NY, USA. ACM. `http://doi.acm.org/10.1145/2939672.2939785`.

[Chen et al., 2019] Chen, W., Tianjiao, L., Yu, T., and Dongsheng, X. (2019). Multi-level adaptive coupled method for industrial control networks safety based on machine learning. 120(May):268–275.

[Cherepanov, 2017] Cherepanov, A. (2017). Win32/industroyer, a new threat for industrial control systems. *White paper, ESET (June 2017)*.

[Cisco, 2020] Cisco (2020). Snort - network intrusion detection and prevention system. `https://www.snort.org/`.

[Cloudera, 2019] Cloudera (2019). Kafka administration - performance and resource considerations. `https://docs.cloudera.com/documentation/kafka/latest/topics/kafka_performance.html`.

[Confluent, 2020] Confluent (2020). Schema management — confluent platform. `https://docs.confluent.io/current/schema-registry/index.html`.

[Cruz et al., 2015] Cruz, T., Barrigas, J., Proença, J., Graziano, A., Panzieri, S., Lev, L., and Simões, P. (2015). Improving network security monitoring for industrial control systems. In *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pages 878–881. IEEE.

[Cruz et al., 2016] Cruz, T., Rosa, L., Proença, J., Maglaras, L., Aubigny, M., Lev, L., Jiang, J., and Simões, P. (2016). A Cybersecurity Detection Framework for Supervisory Control and Data Acquisition Systems. *IEEE Transactions on Industrial Informatics*, 12(6):2236–2246. doi:10.1109/TII.2016.2599841.

[Databricks, 2019] Databricks (2019). Delta architecture, a step beyond lambda architecture. `https://pages.databricks.com/`

201908-WB-Delta-Architecture-A-Step-Beyond-Lambda-Architecture_02-TY.html.

[Demertzis et al., 2019] Demertzis, K., Iliadis, L., and Kikiras, P. (2019). Cyber-Typhon : An Online Multi-task Anomaly Detection Framework. (May).

[Deneut, 2017] Deneut, T. (2017). Siemens profinet scanner. https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/scada/profinet_siemens.rb.

[Dewa and Maglaras, 2016] Dewa, Z. and Maglaras, L. A. (2016). Data Mining and Intrusion Detection Systems. 7(1):62–71.

[Digital Bond, Inc. and N-Dimension Solutions, Solana Networks, 2011] Digital Bond, Inc. and N-Dimension Solutions, Solana Networks (2011). A set of ics ids rules for use with suricata. https://github.com/digitalbond/Quickdraw-Suricata/blob/master/enip-rules.

[Ding et al., 2018] Ding, D., Han, Q.-l., Xiang, Y., Ge, X., and Zhang, X.-m. (2018). Neurocomputing A survey on security control and attack detection for industrial cyber-physical systems. *Neurocomputing*, 275:1674–1683. https://doi.org/10.1016/j.neucom.2017.10.009.

[Diogo, 2018] Diogo, J. (2018). Arp preprocessor patch. https://seclists.org/snort/2018/q4/39.

[DNP Users Group, 2005] DNP Users Group (2005). A dnp3 protocol primer. https://www.dnp.org/Portals/0/AboutUs/DNP3%20Primer%20Rev%20A.pdf.

[Dunkel et al., 2011] Dunkel, J., Fernández, A., Ortiz, R., and Ossowski, S. (2011). Event-driven architecture for decision support in traffic management systems. *Expert Systems with Applications*, 38(6):6530–6539.

[E-ISAC, SANS, 2016] E-ISAC, SANS (2016). Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388.

[Egozcue et al., 2012] Egozcue, E., Rodriguez, D., Ortiz, J., Villar, V., and Tarrafeta, L. (2012). Smart grid security-recommendation for europe and member states.

[Ekman and Andersson, 2019] Ekman, E. and Andersson, B. (2019). iodine. https://github.com/yarrick/iodine.

[Elasticsearch B.V., 2020] Elasticsearch B.V. (2020). Elasticsearch - a distributed restful search engine. `https://github.com/elastic/elasticsearch`.

[ENISA, 2011] ENISA (2011). Annex iii. ics security related standards, guidelines and policy documents.

[EsMnemon et al., 2018] EsMnemon, SOULLIE, A., TORRENTS, A., and CHEVALIER, M. (2018). Modbus client utility. `https://github.com/rapid7/metasploit-framework/blob/master//modules/auxiliary/scanner/scada/modbusclient.rb`.

[EsperTech Inc., 2014] EsperTech Inc. (2014). Event processing with esper and nesper. `http://esper.codehaus.org/`.

[European Commission, 2014] European Commission (2014). Benchmarking smart metering deployment in the eu-27 with a focus on electricity. `https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:52014SC0188`.

[Feick et al., 2018] Feick, M., Kleer, N., and Kohn, M. (2018). Fundamentals of real-time data processing architectures lambda and kappa. *SKILL 2018-Studierendenkonferenz Informatik*.

[Feinstein et al., 2007] Feinstein, B., Curry, D., and Debar, H. (2007). The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765. `https://rfc-editor.org/rfc/rfc4765.txt`.

[Ficco and Romano, 2011] Ficco, M. and Romano, L. (2011). A generic intrusion detection and diagnoser system based on complex event processing. In *2011 First International Conference on Data Compression, Communications and Processing*, pages 275–284. IEEE.

[Ficco et al., 2013] Ficco, M., Tasquier, L., and Aversa, R. (2013). Intrusion detection in cloud computing. In *2013 Eighth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 276–283. IEEE.

[Fielding and Reschke, 2014] Fielding, R. and Reschke, J. (2014). Hypertext transfer protocol (http/1.1): Semantics and content. RFC 7231, RFC Editor. `http://www.rfc-editor.org/rfc/rfc7231.txt`.

[Filkins, 2019] Filkins, B. (2019). Sans 2019 state of ot/ics cybersecurity survey.

[FireEye, 2020] FireEye (2020). Highly evasive attacker leverages solarwinds supply chain to compromise multiple global victims with sunburst backdoor. `https://www.fireeye.com/blog/threat-research/2020/12/`

evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor.html.

[Forgeat, 2015] Forgeat, J. (2015). Data processing architectures–lambda and kappa. *Ericsson Research Blog*.

[Gao et al., 2019] Gao, J., Dong, X., and Lu, T. (2019). Omni SCADA Intrusion Detection Using Deep Learning Algorithms. (July).

[Gao et al., 2016] Gao, Y., Xie, X., Parekh, M., and Bajramovic, E. (2016). SIEM: Policy-based monitoring of SCADA systems. In *Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI)*, volume P-259, pages 559–570.

[Gartner, 2020] Gartner (2020). Top 10 strategic technology trends for 2020. https://www.gartner.com/smarterwithgartner/gartner-top-10-strategic-technology-trends-for-2020/.

[Gerhards, 2009] Gerhards, R. (2009). The syslog protocol. RFC 5424, RFC Editor. http://www.rfc-editor.org/rfc/rfc5424.txt.

[Ghaeini and Tippenhauer, 2016] Ghaeini, H. R. and Tippenhauer, N. O. (2016). HAMIDS: Hierarchical monitoring intrusion detection system for industrial control systems. In *CPS-SPC 2016 - Proceedings of the 2nd ACM Workshop on Cyber-Physical Systems Security and PrivaCy, co-located with CCS 2016*, pages 103–111.

[Ghosh and Sampalli, 2019] Ghosh, S. and Sampalli, S. (2019). A Survey of Security in SCADA Networks: Current Issues and Future Challenges. *IEEE Access*, PP:1–1.

[Goh et al., 2016] Goh, J., Adepu, S., Junejo, K. N., and Mathur, A. (2016). A dataset to support research in the design of secure water treatment systems. In *International Conference on Critical Information Infrastructures Security*, pages 88–99. Springer.

[Gonzalez, 2015] Gonzalez, C. (2015). Engineering essentials: What is a programmable logic controller? https://www.machinedesign.com/learning-resources/engineering-essentials/article/21834250/engineering-essentials-what-is-a-programmable-logic-controller.

[Greenwood, 2021] Greenwood, D. (2021). Applying the principles of zero-trust architecture to protect sensitive and critical data. *Network Security*, 2021(6):7–9.

[Handa et al., 2019] Handa, A., Sharma, A., and Shukla, S. K. (2019). Machine learning in cybersecurity : A review. (January):1–7.

[Hazarika et al., 2017] Hazarika, A. V., Ram, G. J. S. R., and Jain, E. (2017). Performance comparision of hadoop and spark engine. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, pages 671–674. IEEE.

[Hemsley and Fisher, 2018] Hemsley, K. and Fisher, R. (2018). A history of cyber incidents and threats involving industrial control systems. In *International Conference on Critical Infrastructure Protection*, pages 215–242. Springer.

[Hilt, 2014a] Hilt, S. (2014a). Enip nmap scan. `https://svn.nmap.org/nmap/scripts/enip-info.nse`.

[Hilt, 2014b] Hilt, S. (2014b). S7 nmap scan. `https://svn.nmap.org/nmap/scripts/s7-info.nse`.

[Hilt, 2015] Hilt, S. (2015). Digital bond's ids/ips rules for ics and ics protocols. `https://github.com/digitalbond/Quickdraw-Snort/blob/master/all-quickdraw.rules`.

[Hindy et al., 2018] Hindy, H., Brosset, D., Bayne, E., Seeam, A., Tachtatzis, C., Atkinson, R., and Bellekens, X. (2018). A taxonomy and survey of intrusion detection system design techniques, network threats and datasets. *arXiv preprint arXiv:1806.03517*.

[Hink, 2020] Hink, R. C. B. (2020). Collection of resources for industrial control system cybersecurity. `https://github.com/rayborg/ICS_repo`.

[Hohpe and Woolf, 2004] Hohpe, G. and Woolf, B. (2004). *Enterprise integration patterns: Designing, building, and deploying messaging solutions.* Addison-Wesley Professional.

[Humayed et al., 2017] Humayed, A., Lin, J., Li, F., and Luo, B. (2017). Cyber-physical systems security—a survey. *IEEE Internet of Things Journal*, 4(6):1802–1831.

[International Electrotechnical Commission, 2018] International Electrotechnical Commission (2018). Industrial communication networks - network and system security - part 1-1: Terminology, concepts and models. `https://www.isa.org/store/products/product-detail/?productId=116720`.

[International Electrotechnical Commission, 2020a] International Electrotechnical Commission (2020a). Iec 60870-5:2020 ser - series telecontrol equipment and systems - part 5: Transmission protocols. `https://webstore.iec.ch/publication/3755`.

[International Electrotechnical Commission, 2020b] International Electrotechnical Commission (2020b). Iec 61850:2020 series - communication networks and systems for power utility automation. `https://webstore.iec.ch/publication/6028`.

[ISA, 2007] ISA, A. (2007). Isa-99. 00. 01-2007 security for industrial automation and control systems part 1 terminology, concepts, and models. *International Society for Automation*, 10.

[Iturbe et al., 2017] Iturbe, M., Garitano, I., Zurutuza, U., and Uribeetxeberria, R. (2017). Towards Large-Scale, Heterogeneous Anomaly Detection Systems in Industrial Networks: A Survey of Current Trends. *Security and Communication Networks*, 2017.

[Jakobson and Weissman, 1995] Jakobson, G. and Weissman, M. (1995). Real-time telecommunication network management: extending event correlation with temporal constraints. In *International Symposium on Integrated Network Management*, pages 290–301. Springer.

[Jiang et al., 2017] Jiang, J., Zhao, X., Wallace, S., Cotilla-Sanchez, E., and Bass, R. (2017). Mining PMU Data Streams to Improve Electric Power System Resilience. pages 95–102.

[Jordan, 2012] Jordan, R. (2012). Snort 2.9.2: Scada preprocessors. `https://blog.snort.org/2012/01/snort-292-scada-preprocessors.html`.

[Jormakka et al., 2007] Jormakka, J., Jormakka, H., and Väre, J. (2007). A lightweight management system for a military ad hoc network. In *International Conference on Information Networking*, pages 533–543. Springer.

[Kampanakis and Suzuki, 2017] Kampanakis, P. and Suzuki, M. (2017). Incident object description exchange format usage guidance. RFC 8274, RFC Editor.

[Kang et al., 2016] Kang, B., McLaughlin, K., and Sezer, S. (2016). Towards A Stateful Analysis Framework for Smart Grid Network Intrusion Detection. pages 124–131.

[Kasperky, 2017] Kasperky (2017). Threat landscape for industrial automation systems h2-2017. `https://ics-cert.kaspersky.com/reports/2018/03/26/threat-landscape-for-industrial-automation-systems-in-h2-2017`.

[Kasperky, 2018] Kasperky (2018). Threat landscape for industrial automation systems h2-2018. `https://ics-cert.kaspersky.com/reports/2019/03/27/threat-landscape-for-industrial-automation-systems-h2-2018/`.

[Kasperky, 2019] Kasperky (2019). Threat landscape for industrial automation systems h2-2019. `https://ics-cert.kaspersky.com/media/KASPERSKY_H22019_ICS_REPORT_FINAL_EN.pdf`.

[Kavanagh et al., 2020] Kavanagh, K. M., Rochford, O., and Bussa, T. (2020). Magic quadrant for security information and event management. *Gartner Group Research Note*.

[Ke et al., 2017] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, pages 3146–3154.

[Keliris et al., 2017] Keliris, A., Salehghaffari, H., Cairl, B., Krishnamurthy, P., Maniatakos, M., and Khorrami, F. (2017). Machine learning-based defense against process-Aware attacks on Industrial Control Systems. *Proceedings - International Test Conference*, pages 1–10.

[Khan et al., 2019] Khan, I. A., Pi, D., and Khan, Z. U. (2019). HML-IDS : A Hybrid-Multilevel Anomaly Prediction Approach for Intrusion Detection in SCADA Systems. *IEEE Access*, 7:89507–89521.

[Khodabakhsh et al., 2017] Khodabakhsh, A., Ari, I., and Bakir, M. (2017). Cloud-based Fault Detection and Classification for Oil & Gas Industry. i:1–6. `http://arxiv.org/abs/1705.04583`.

[Knapp and Langill, 2014] Knapp, E. D. and Langill, J. T. (2014). *Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems*. Syngress.

[Kravchik and Shabtai, 2018] Kravchik, M. and Shabtai, A. (2018). Detecting Cyber Attacks in Industrial Control Systems Using Convolutional Neural Networks. *Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy - CPS-SPC '18*, pages 72–83. `http://dl.acm.org/citation.cfm?doid=3264888.3264896`.

[Kreps, 2014a] Kreps, J. (2014a). Benchmarking apache kafka: 2 million writes per second (on three cheap machines). `https://engineering.linkedin.com/kafka/benchmarking-apache-kafka-2-million-writes-second-three-cheap-machines`.

[Kreps, 2014b] Kreps, J. (2014b). Questioning the lambda architecture. *Online article, July*, page 205.

[Lal and Suman, 2020] Lal, D. K. and Suman, U. (2020). A survey of real-time big data processing algorithms. In *Reliability and Risk Assessment in Engineering*, pages 3–10. Springer.

[Landoop, 2018] Landoop (2018). Lenses for apache kafka. `https://github.com/lensesio/stream-reactor`.

[Laney, 2001] Laney, D. (2001). 3d data management: Controlling data volume, velocity and variety. *META group research note*, 6(70):1.

[Lang, 2014] Lang, J.-P. (2014). Prelude oss edition. https://www.prelude-ids.org.

[Langner, 2013] Langner, R. (2013). To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve. *The Langner Group*.

[Leach et al., 2005] Leach, P. J., Mealling, M., and Salz, R. (2005). A universally unique identifier (uuid) urn namespace. RFC 4122, RFC Editor. http://www.rfc-editor.org/rfc/rfc4122.txt.

[Lee et al., 2016] Lee, J., Jeon, J., Lee, C., Lee, J., and Cho, J. (2016). An implementation of log visualization system combined SCADA Honeypot. pages 441–444.

[Lemay and Fernandez, 2016] Lemay, A. and Fernandez, J. M. (2016). Providing {SCADA} network data sets for intrusion detection research. In *9th Workshop on Cyber Security Experimentation and Test ({CSET} 16)*.

[Leszczyna, 2019] Leszczyna, R. (2019). *Cybersecurity in the Electricity Sector*. Springer.

[Lin et al., 2013] Lin, H., Slagell, A., Di Martino, C., Kalbarczyk, Z., and Iyer, R. K. (2013). Adapting bro into scada: building a specification-based intrusion detection system for the dnp3 protocol. In *Proceedings of the Eighth Annual Cyber Security and Information Intelligence Research Workshop*, pages 1–4.

[Linux Foundation, 2018] Linux Foundation (2018). Prometheus - monitoring system & time series database. https://prometheus.io/.

[Linux Foundation, 2019] Linux Foundation (2019). Delta lake - reliable data lakes at scale. https://delta.io/.

[Liu et al., 2010] Liu, D., Gu, T., and Xue, J.-P. (2010). Rule engine based on improvement rete algorithm. In *The 2010 International Conference on Apperceiving Computing and Intelligence Analysis Proceeding*, pages 346–349. IEEE.

[Lundberg and Lee, 2017] Lundberg, S. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. *arXiv preprint arXiv:1705.07874*.

[Manuel and Salvio, 2019] Manuel, J. and Salvio, J. (2019). Lockergoga: Ransomware targeting critical infrastructure. https://www.fortinet.com/blog/threat-research/lockergoga-ransomeware-targeting-critical-infrastructure.

[Martin, 2014] Martin, L. (2014). Cyber kill chain®. *URL: http://cyber. lockheedmartin. com/hubfs/Gaining the Advantage Cyber Kill Chain. pdf*.

[Marz and Warren, 2015] Marz, N. and Warren, J. (2015). *Big Data: Principles and best practices of scalable real-time data systems.* New York; Manning Publications Co.

[Maynard and Beecroft, 2015] Maynard, T. and Beecroft, N. (2015). Business blackout: The insurance implications of a cyber-attack on the us power grid. *Lloyd's of London. Accessed March*, 15:2019.

[McKinnon et al., 2020] McKinnon, C., Carroll, J., McDonald, A., Koukoura, S., Infield, D., and Soraghan, C. (2020). Comparison of new anomaly detection technique for wind turbine condition monitoring using gearbox scada data. *Energies*, 13(19):5152.

[McLaughlin and McAdam, 2016] McLaughlin, P. and McAdam, R. (2016). The undiscovered country: The future of industrial automation. *Honeywell Process Solutions. Honeywell.*

[Modbus Organization, Inc, 2006] Modbus Organization, Inc (2006). Modbus application protocol specification v1.1b. `http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf`.

[Modbus Organization, Inc, 2018] Modbus Organization, Inc (2018). Modbus/tcp security protocol specification. `http://modbus.org/docs/MB-TCP-Security-v21_2018-07-24.pdf`.

[Mohamed et al., 2020] Mohamed, A., Najafabadi, M. K., Wah, Y. B., Zaman, E. A. K., and Maskat, R. (2020). The state of the art and taxonomy of big data analytics: view from new big data framework. *Artificial Intelligence Review*, 53(2):989–1037.

[Monteiro et al., 2019] Monteiro, J. D., Rosa, L., and de Freitas, M. B. (2019). Dos exploitation of allen-bradley's legacy protocol (pccc). `https://github.com/rapid7/metasploit-framework/pull/11106`.

[Morris and Gao, 2014] Morris, T. and Gao, W. (2014). Industrial control system network traffic data sets to facilitate intrusion detection system research. *Critical infrastructure protection VIII—8th IFIP WG*, 11:17–19.

[Myers et al., 2011] Myers, J., Grimaila, M. R., and Mills, R. F. (2011). Log-based distributed security event detection using simple event correlator. In *2011 44th Hawaii International Conference on System Sciences*, pages 1–7. IEEE.

[Nadler et al., 2019] Nadler, A., Aminov, A., and Shabtai, A. (2019). Detection of malicious and low throughput data exfiltration over the dns protocol. *Computers & Security*, 80:36–53.

[Nazir et al., 2017] Nazir, S., Patel, S., and Patel, D. (2017). Assessing and augmenting scada cyber security: A survey of techniques. *Computers & Security*, 70:436–454.

[NETRESEC, 2015] NETRESEC (2015). Capture files from 4sics geek lounge.

[NETRESEC, 2020] NETRESEC (2020). Scada/ics network captures. `https://www.netresec.com/index.ashx?page=PcapFiles`.

[Nguyen et al., 2018] Nguyen, V. Q., Van Ma, L., Kim, J. Y., Kim, K., and Kim, J. (2018). Applications of anomaly detection using deep learning on time series data. *Proceedings - IEEE 16th International Conference on Dependable, Autonomic and Secure Computing, IEEE 16th International Conference on Pervasive Intelligence and Computing, IEEE 4th International Conference on Big Data Intelligence and Computing and IEEE 3*, pages 393–396.

[Nisioti et al., 2018] Nisioti, A., Mylonas, A., Yoo, P. D., and Katos, V. (2018). From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods. *IEEE Communications Surveys and Tutorials*, 20(4):3369–3388.

[NIST, 2018] NIST (2018). Framework for improving critical infrastructure cybersecurity. `https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04162018.pdf`.

[NodeBrain.org, 2014] NodeBrain.org (2014). Nodebrain, open source project. `http://nodebrain.sourceforge.net`.

[North American Electric Reliability Corporation, 2009] North American Electric Reliability Corporation (2009). Cip-009-2: Cyber security — recovery plans for critical cyber assets. `https://www.nerc.com/files/CIP-009-2.pdf`.

[Nuojua et al., 2017] Nuojua, V., David, G., and Hämäläinen, T. (2017). Dns tunneling detection techniques–classification, and theoretical comparison in case of a real apt campaign. In *Internet of Things, Smart Spaces, and Next Generation Networks and Systems*, pages 280–291. Springer.

[OPC Foundation, 2020] OPC Foundation (2020). Opc unified architecture (ua). `https://opcfoundation.org/about/opc-technologies/opc-ua/`.

[Open Information Security Foundation, 2020] Open Information Security Foundation (2020). Open source ids / ips / nsm engine. `https://suricata-ids.org/`.

[OSSEC, 2016] OSSEC (2016). Ossec - world's most widely used host intrusion detection system - hids. `https://www.ossec.net/`.

[Pathak et al., 2018] Pathak, A. R., Pandey, M., and Rautaray, S. (2018). Construing the big data based on taxonomy, analytics and approaches. *Iran Journal of Computer Science*, 1(4):237–259.

[Pawlinski et al., 2014] Pawlinski, P., Jaroszewski, P., Urbanowicz, J., Jacewicz, P., Zielony, P., Kijewski, P., and Gorzelak, K. (2014). Standards and tools for exchange and processing of actionable information. *European Union Agency for Network and Information Security, Heraklion, Greece.*

[Payment Card Industry, 2002] Payment Card Industry (2002). Data security standard, version 2.0, october 2010. `https://www.pcisecuritystandards.org/pdfs/pr_101028_standards_2.0.pdf`.

[Pedregosa et al., 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Penetration Testing Execution Standard Group, 2019] Penetration Testing Execution Standard Group (2019). Penetration testing execution standard. `http://www.pentest-standard.org/index.php/Main_Page`.

[Petri et al., 2013] Petri, G., Nunes, R. C., Lopez, V. L., Junior, T. C., and dos Santos, O. M. (2013). Kbam: Data model of a knowledge base for monitoring attacks. *LADC 2013-Fast Abstract, Rio de Janeiro, Brazil.*

[Phillips et al., 2020] Phillips, B., Gamess, E., and Krishnaprasad, S. (2020). An evaluation of machine learning-based anomaly detection in a scada system using the modbus protocol. In *Proceedings of the 2020 ACM Southeast Conference*, pages 188–196.

[Pulido-Gaytan et al., 2020] Pulido-Gaytan, L. B., Tchernykh, A., Cortés-Mendoza, J. M., Babenko, M., and Radchenko, G. (2020). A survey on privacy-preserving machine learning with fully homomorphic encryption. In *Latin American High Performance Computing Conference*, pages 115–129. Springer.

[Ramotsoela et al., 2019] Ramotsoela, D. T., Hancke, G. P., and Abu-Mahfouz, A. M. (2019). Attack detection in water distribution systems using machine learning. *Human-centric Computing and Information Sciences*, 9(1). `https://doi.org/10.1186/s13673-019-0175-8`.

[Red-hat, 2014] Red-hat (2014). Business rules management system (brms) solution. http://www.drools.org/.

[Red Lion, 2019] Red Lion (2019). Crimson®3.0. http://www.redlion.net/crimson-30.

[Rist et al., 2013] Rist, L., Vestergaard, J., Haslinger, D., Pasquale, A., and Smith, J. (2013). Conpot ics/scada honeypot. *Honeynet Project (conpot. org).*

[Rosa, 2019a] Rosa, L. (2019a). New module pcomclient. https://github.com/rapid7/metasploit-framework/pull/11219/.

[Rosa, 2019b] Rosa, L. (2019b). New pcom module to send admin commands. https://github.com/rapid7/metasploit-framework/pull/11220/.

[Rosa, 2019c] Rosa, L. (2019c). New snort rules for pcom protocol. https://marc.info/?l=snort-sigs&m=154746968717558.

[Rosa, 2019d] Rosa, L. (2019d). Pcom pcap captures. https://github.com/lmrosa/pcom-misc/tree/master/pcaps.

[Rosa, 2019e] Rosa, L. (2019e). pcomtcp: dissection of additional pcom/ascii fields. https://code.wireshark.org/review/#/c/31467/.

[Rosa, 2019f] Rosa, L. (2019f). pcomtcp: new built-in dissector for pcom protocol. https://code.wireshark.org/review/#/c/30823/.

[Rosa, 2019g] Rosa, L. (2019g). pcomtcp: Pcom/binary command to descriptions. https://code.wireshark.org/review/#/c/31858/.

[Rosa, 2019h] Rosa, L. (2019h). Scada cip enum scan. https://github.com/nmap/nmap/pull/1539.

[Rosa, 2019i] Rosa, L. (2019i). Scada scan to collect information from unitronics plcs via pcom protocol. https://github.com/nmap/nmap/pull/1445.

[Rosa et al., 2015] Rosa, L., Alves, P., Cruz, T., Simões, P., and Monteiro, E. (2015). A comparative study of correlation engines for security event management. In *Iccws 2015-The Proceedings of the 10th International Conference on Cyber Warfare and Security*, page 277.

[Rosa et al., 2020] Rosa, L., Borges de Freitas, M., Henriques, J., Quitério, P., Caldeira, F., Cruz, T., and Simões, P. (2020). Evolving the security paradigm for industrial iot environments. In *Cyber Security of Industrial Control Systems in the Future Internet Environment*, pages 69–90. IGI Global.

[Rosa et al., 2021] Rosa, L., Cruz, T., Freitas, M., Quitério, P., Henriques, J., Caldeira, F., Monteiro, E., and Simões, P. (2021). Intrusion and anomaly detection for the next-generation of industrial automation and control systems. *Future Generation Computer Systems*, 119:50–67.

[Rosa et al., 2017] Rosa, L., Cruz, T., Simões, P., Monteiro, E., and Lev, L. (2017). Attacking SCADA systems: A practical perspective. In *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, Lisbon, Portugal. IEEE. `http://ieeexplore.ieee.org/document/7987369/`.

[Rosa et al., 2019] Rosa, L., Freitas, M., Mazo, S., Monteiro, E., Cruz, T., and Simões, P. (2019). A comprehensive security analysis of a scada protocol: From osint to mitigation. *IEEE Access*, 7:42156–42168. doi:10.1109/ACCESS.2019.2906926.

[Rose et al., 2019] Rose, S., Borchert, O., Mitchell, S., and Connelly, S. (2019). Zero trust architecture. Technical report, National Institute of Standards and Technology.

[Roth and Patzke, 2019] Roth, F. and Patzke, T. (2019). Generic signature format for siem systems. `https://github.com/Neo23x0/sigma`.

[Rudakov, 2010] Rudakov, A. (2010). Enumerates scada modbus slave ids (sids) and collects their device information. `https://svn.nmap.org/nmap/scripts/modbus-discover.nse`.

[Saggi and Jain, 2018] Saggi, M. K. and Jain, S. (2018). A survey towards an integration of big data analytics to big insights for value-creation. *Information Processing & Management*, 54(5):758–790.

[SANS, 2017] SANS (2017). Defense use case no. 6 "modular ics malware". `https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_6.pdf`.

[Santamarta et al., 2012] Santamarta, R., Wightman, R., and todb (2012). Allen-bradley/rockwell automation ethernet/ip cip commands. `https://www.rapid7.com/db/modules/auxiliary/admin/scada/multi_cip_command`.

[Sarbanes, 2002] Sarbanes, P. (2002). Sarbanes-oxley act of 2002. In *The Public Company Accounting Reform and Investor Protection Act. Washington DC: US Congress*, page 55.

[Schiffer, 2016] Schiffer, V. (2016). The common industrial protocol (cip™) and the family of cip networks. `https://www.odva.org/wp-content/uploads/2020/06/PUB00123R1_Common-Industrial_Protocol_and_Family_of_CIP_Networks.pdf`.

[Shapira et al., 2017] Shapira, G., Palino, T., Sivaram, R., and Narkhede, N. (2017). *Kafka: the definitive guide*. O'Reilly Media, Incorporated.

[Shitharth and Prince Winston, 2017] Shitharth, S. and Prince Winston, D. (2017). An enhanced optimization based algorithm for intrusion detection in SCADA network. *Computers and Security*, 70:16–26. `https://doi.org/10.1016/j.cose.2017.04.012`.

[Simões et al., 2015] Simões, P., Cruz, T., Proença, J., and Monteiro, E. (2015). Specialized honeypots for scada systems. In *Cyber Security: Analytics, Technology and Automation*, pages 251–269. Springer.

[Slowik, 2018] Slowik, J. (2018). Anatomy of an attack: Detecting and defeating crashoverride. *VB2018, October*.

[Smart Grid Coordination Group, 2012] Smart Grid Coordination Group (2012). Smart grid reference architecture. `https://ec.europa.eu/energy/sites/ener/files/documents/xpert_group1_reference_architecture.pdf`.

[Sokolov et al., 2019] Sokolov, A. N., Alabugin, S. K., and Pyatnitsky, I. A. (2019). Traffic modeling by recurrent neural networks for intrusion detection in industrial control systems. In *2019 International Conference on Industrial Engineering, Applications and Manufacturing, ICIEAM 2019*, pages 1–5. IEEE.

[Steinberg et al., 2011] Steinberg, R. A., Rudd, C., Lacy, S., and Hanna, A. (2011). *ITIL service operation*. Stationery Office Limited.

[Stewart et al., 2017] Stewart, B., Rosa, L., Maglaras, L. A., Cruz, T. J., Ferrag, M. A., Simoes, P., and Janicke, H. (2017). A novel intrusion detection mechanism for scada systems which automatically adapts to network topology changes. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, 4(10). doi:eai.1-2-2017.152155.

[Stouffer et al., 2015] Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M., and Hahn, A. (2015). Nist special publication 800-82 revision 2. guide to industrial control systems (ics) security. *NIST*.

[Suleiman et al., 2015] Suleiman, H., Alqassem, I., Diabat, A., Arnautovic, E., and Svetinovic, D. (2015). Integrated smart grid systems security threat model. *Information Systems*, 53:147–160.

[Sundararajan et al., 2018] Sundararajan, A., Wei, L., Khan, T., Sarwat, A. I., and Rodrigo, D. (2018). A Tri-Modular Framework to Minimize Smart Grid Cyber-Attack Cognitive Gap in Utility Control Centers. *Proceedings - Resilience Week 2018, RWS 2018*, (Dm):117–123.

[Taormina et al., 2018] Taormina, R., Galelli, S., Tippenhauer, N. O., Salomons, E., Ostfeld, A., Eliades, D. G., Aghashahi, M., Sundararajan, R., Pourahmadi, M., Banks, M. K., Brentan, B. M., Herrera, M., Rasekh, A., Campbell, E., Montalvo, I., Lima, G., Izquierdo, J., Haddad, K., Gatsis, N., Taha, A., Somasundaram, S. L., Ayala-Cabrera, D., Chandy, S. E., Campbell, B., Biswas, P., Lo, C. S., Manzi, D., Luvizotto, Jr, E., Barker, Z. A., Giacomoni, M., Pasha, M. F. K., Shafiee, M. E., Abokifa, A. A., Housh, M., Kc, B., and Ohar, Z. (2018). The battle of the attack detection algorithms: Disclosing cyber attacks on water distribution networks. *Journal of Water Resources Planning and Management*, 144(8):04018048.

[Teixeira et al., 2018] Teixeira, M. A., Salman, T., Zolanvari, M., Jain, R., Meskin, N., and Samaka, M. (2018). Scada system testbed for cybersecurity research using machine learning approach. *Future Internet*, 10(8):76.

[Terzi et al., 2017] Terzi, D. S., Terzi, R., and Sagiroglu, S. (2017). Big data analytics for network anomaly detection from netflow data. In *2017 International Conference on Computer Science and Engineering (UBMK)*, pages 592–597. IEEE.

[The MITRE Corporation, 2020] The MITRE Corporation (2020). Mitre att&ck®. `https://attack.mitre.org/`.

[Theriault, 2019] Theriault, J. (2019). Pcom - a basic pcom implementation in python. `https://pypi.org/project/pcom/`.

[Udd et al., 2016] Udd, R., Asplund, M., Nadjm-Tehrani, S., Kazemtabrizi, M., and Ekstedt, M. (2016). Exploiting bro for intrusion detection in a scada system. In *Proceedings of the 2nd ACM International Workshop on Cyber-Physical System Security*, pages 44–51.

[Ujvarosi, 2016] Ujvarosi, A. (2016). Evolution of scada systems. *Bulletin of the Transilvania University of Brasov. Engineering Sciences. Series I*, 9(1):63.

[Unitronics, Inc, 2014] Unitronics, Inc (2014). Communication with the vision™ plc. `https://unitronicsplc.com/Download/SoftwareUtilities/Unitronics%20PCOM%20Protocol.pdf`.

[Unitronics, Inc, 2019a] Unitronics, Inc (2019a). .net driver.

[Unitronics, Inc, 2019b] Unitronics, Inc (2019b). Visilogic™for vision™and samba™. `https://unitronicsplc.com/software-visilogic-for-programmable-controllers`.

[Vaarandi, 2014] Vaarandi, R. (2014). Simple event correlator. `http://simple-evcorr.sourceforge.net`.

[Vávra and Hromada, 2016] Vávra, J. and Hromada, M. (2016). Comparison of the intrusion detection system rules in relation with the scada systems. In *Computer Science On-line Conference*, pages 159–169. Springer.

[Wightman, 2018a] Wightman, R. (2018a). Schneider modicon ladder logic upload/download. `https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/scada/modicon_stux_transfer.rb`.

[Wightman, 2018b] Wightman, R. (2018b). Schneider modicon remote start/stop command. `https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/admin/scada/modicon_command.rb`.

[Williams and Nicolett, 2005] Williams, A. T. and Nicolett, M. (2005). Improve it security with vulnerability management. `https://www.gartner.com/en/documents/480703`.

[Wireshark Foundation, 2019] Wireshark Foundation (2019). Wireshark - 7.4. expert information. `https://www.wireshark.org/docs/wsug_html_chunked/ChAdvExpert.html`.

[Wong et al., 2017] Wong, K., Dillabaugh, C., Seddigh, N., and Nandy, B. (2017). Enhancing suricata intrusion detection system for cyber security in scada networks. In *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–5. IEEE.

[Wu et al., 2012] Wu, Y., Sheng, Q. Z., Ranasinghe, D., and Yao, L. (2012). Peertrack: a platform for tracking and tracing objects in large-scale traceability networks. In *Proceedings of the 15th International Conference on Extending Database Technology*, pages 586–589.

[Yang et al., 2019] Yang, H., Cheng, L., and Chuah, M. C. (2019). Deep-Learning-Based Network Intrusion Detection for SCADA Systems. *2019 IEEE Conference on Communications and Network Security (CNS)*, pages 1–7.

[Yang et al., 2017] Yang, W., Haider, S. N., Zou, J.-h., and Zhao, Q.-c. (2017). Industrial Big Data Platform Based on Open Source Software. volume 54, pages 649–658.

[Yao et al., 2011] Yao, W., Chu, C.-H., and Li, Z. (2011). Leveraging complex event processing for smart hospitals using rfid. *Journal of Network and Computer Applications*, 34(3):799–810.

[Yassine et al., 2018] Yassine, S., Khalife, J., Chamoun, M., and El Ghor, H. (2018). A survey of dns tunnelling detection techniques using machine learning. In *BDCSIntell*, pages 63–66.

[Yu et al., 2016] Yu, B., Smith, L., Threefoot, M., and Olumofin, F. G. (2016). Behavior analysis based dns tunneling detection and classification with big data technologies. In *IoTBD*, pages 284–290.

[Zhu et al., 2011] Zhu, B., Joseph, A., and Sastry, S. (2011). A taxonomy of cyber attacks on scada systems. In *2011 IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing*, pages 380–388. IEEE.

[Zopf, 2002] Zopf, R. (2002). Real-time transport protocol (rtp) payload for comfort noise (cn). RFC 3389, RFC Editor.