UNIVERSITY OF PIRAEUS
School of Information and Communication Technologies
Department of Informatics

Thesis

Thesis Title: Τίτλος Διατριβής:	Assessing the Discriminative Power of YARA Rules for Nation-State Malware Attribution Αξιολόγηση της Διακριτικής Ικανότητας των Κανόνων YARA για την Απόδοση Κακόβουλου Λογισμικού σε Κρατικούς Δρώντες
Student's name-surname:	Michail Vintzilaios
Father's name:	Vasileios
Student's ID No:	Π22022
Supervisor:	Patsakis Konstantinos, Professor

June 2026 / Ιούνιος 2026

Copyright Declaration

Acknowledgments

I would like to express my gratitude to the people who supported me throughout the course of this thesis. First and foremost, I would like to thank my supervisor, Constantinos Patsakis, for giving me the opportunity to explore and research a field that has interested me throughout my Bachelor's years. Second, I want to thank Dr. Theodoros Apostolopoulos for his continuous help and coordination throughout the development of this thesis. He provided me with the guidance necessary to create a work I am very proud of, and I look forward to our future work together. Lastly, I am deeply grateful for the valuable advice on Machine Learning from Ippokratis Pantelidis, as well as for the support and encouragement of my family, and my friends Thanos, Nikos, and Socrates, who helped me structure and refine my ideas.

Abstract

Attributing malware to the Advanced Persistent Threat (APT) group or nation-state behind it is a challenging but invaluable task, complicated by the deliberate camouflage and false-flag techniques these actors employ. Most existing attribution approaches rely on dynamic analysis or machine-learning models that are costly to run and examine samples in isolation, without revealing the shared tooling that links families together. This thesis investigates a lightweight, purely static alternative: representing each APT family by its *YARA-rule match profile*, the set of publicly available rules its samples trigger, and examines whether these profiles carry a discriminative signal about the family's country of origin and its relationships to other families. We assembled a dataset of 14,938 verified Windows PE/DLL samples spanning 78 APT families across 12 countries, scanned them with a large corpus of public YARA rules filtered to exclude attribution-encoding and non-technical rules, and clustered the families with agglomerative hierarchical clustering, evaluating the results with chance-corrected metrics and permutation-based significance tests. We find that YARA profiles carry only a weak signal about a family's country of origin. A per-country analysis further shows that this faint same-country tendency comes almost entirely from North Korea, whose families cluster tightly together, whereas no other country, including the most heavily represented ones, shows comparable cohesion. The profiles are far more useful; however, for uncovering relationships between individual families: pairs that MITRE ATT&CK records as the same actor or as related tend to cluster closer than chance. Static YARA signals are therefore of limited use for country-level attribution but genuinely captures shared tooling between individual families.

Scientific area: Cybersecurity; Malware Analysis; Machine Learning.
Keywords: APT attribution; YARA rules; malware clustering; hierarchical clustering; nation-state malware.

Περίληψη

Η απόδοση κακόβουλου λογισμικού (malware) στην ομάδα Προηγμένης Μόνιμης Απειλής (Advanced Persistent Threat – APT) ή στο κράτος που την υποστηρίζει είναι δύσκολο αλλά πολύτιμο έργο, το οποίο περιπλέκεται από τις τεχνικές απόκρυψης και παραπλάνησης (false-flag) που χρησιμοποιούν αυτοί οι δρώντες. Οι περισσότερες υπάρχουσες προσεγγίσεις βασίζονται σε δυναμική ανάλυση ή σε μοντέλα μηχανικής μάθησης, τα οποία είναι δαπανηρά και εξετάζουν τα δείγματα μεμονωμένα, χωρίς να αποκαλύπτουν τα κοινά εργαλεία (shared tooling) που συνδέουν τις οικογένειες. Η παρούσα εργασία διερευνά μια αμιγώς στατική εναλλακτική: την αναπαράσταση κάθε οικογένειας APT από το προφίλ αντιστοίχισης κανόνων YARA (YARA-rule match profile). Συγκεντρώσαμε 14.938 επαληθευμένα δείγματα Windows PE/DLL από 78 οικογένειες APT και 12 χώρες, τα σαρώσαμε με εκτενές σύνολο δημόσιων κανόνων YARA – φιλτραρισμένο ώστε να εξαιρεθούν κανόνες που ενσωματώνουν ενδείξεις απόδοσης – και ομαδοποιήσαμε τις οικογένειες με συσσωρευτική ιεραρχική συσταδοποίηση, αξιολογώντας τα αποτελέσματα με μετρικές διορθωμένες ως προς την τυχαιότητα και ελέγχους σημαντικότητας βάσει μεταθέσεων. Διαπιστώνουμε ότι τα προφίλ YARA φέρουν ασθενές μόνο σήμα για τη χώρα προέλευσης, το οποίο προέρχεται σχεδόν εξ ολοκλήρου από τη Βόρεια Κορέα· καμία άλλη χώρα δεν εμφανίζει αντίστοιχη συνοχή. Τα προφίλ αποδεικνύονται πολύ πιο χρήσιμα για την ανακάλυψη σχέσεων μεταξύ μεμονωμένων οικογενειών: ζεύγη που το MITRE ATT&CK καταγράφει ως τον ίδιο δρώντα ή ως συσχετιζόμενα τείνουν να συσταδοποιούνται πιο κοντά από ό,τι θα αναμενόταν τυχαία.

Επιστημονική περιοχή: Ασφάλεια Υπολογιστών· Ανάλυση Κακόβουλου Λογισμικού· Μηχανική Μάθηση.
Λέξεις-κλειδιά: απόδοση APT· κανόνες YARA· ομαδοποίηση κακόβουλου λογισμικού· ιεραρχική ομαδοποίηση· κρατικό κακόβουλο λογισμικό.

In loving memory of George Gratsias

Contents

List of Figures

List of Tables

1 Introduction

Over the past years, APT groups have become increasingly prominent actors in the cyber-attack landscape [1]. Their sophistication and destructiveness make them a primary concern for both enterprises and governments. To defend against such attacks more effectively, profiling these actors and attributing malware to a specific APT family or country can be immensely valuable [2]. However, the advanced camouflage that APT groups employ to evade attribution and plant false flags makes attributing both individual malware samples and the families behind them a highly challenging task for malware analysts [3].

Although reliable attribution methods do exist, many of them rely on machine-learning models and dynamic analysis, which are time-consuming and costly [4, 5]. Moreover, such approaches typically attribute samples in isolation, without revealing the underlying structure of relationships and shared tooling between APT families – information that would allow us to view not only each family on its own, but also the broader landscape of the groups and countries behind them [6].

Despite adversaries' efforts to mislead analysts, some malware artifacts and behavioral indicators remain very hard to disguise. This is captured well by the Pyramid of Pain, introduced in 2013 by David J. Bianco [7]. As seen in figure 1.1, the pyramid ranks indicators by how difficult they are for an adversary to change. As a result, Tools, the software an attacker relies on, which leaves detectable static artifacts in the binary, sit near the top, second only to the adversary's Tactics, Techniques, and Procedures (TTPs). Because TTPs are behavioral and can rarely be recovered through static analysis, we focus on Tools.

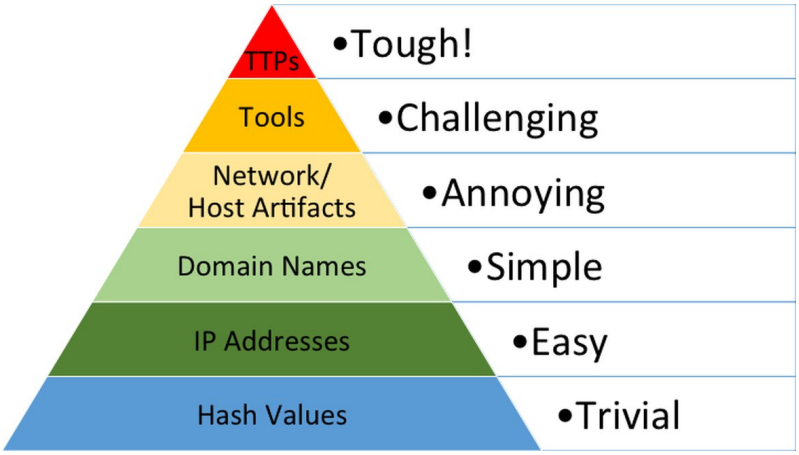


Figure 1.1: The Pyramid of Pain [7]: indicators ranked by how costly they are for an adversary to change.

To detect such tooling directly from static artifacts, we rely on YARA [8]. YARA rules offer a more effective and flexible method for malware detection than traditional hashing because they rely on customizable rules [9]. This heuristic, pattern-matching nature provides adaptability against malware variants associated with specific threat actors, which often evade traditional detection systems, ultimately aiding accurate attribution and the development of countermeasures. YARA is also highly valuable in detecting the polymorphic and stealthy behaviors that characterize sophisticated malware [10]. However, advanced threat actors can still evade detection by modifying their code to bypass common YARA rules. Automatically generated rules are particularly susceptible to these evasion techniques. While highly sophisticated rules are more effective at identifying malware variants, writing them requires analysts to possess significant experience and a deep understanding of malware analysis [9, 11].

In this thesis, we evaluate whether YARA rules can exploit the difficulty of hiding tooling to reveal inter-country or inter-family signals, and thereby help analysts cluster malware samples to their corresponding countries using only static artifacts. The same approach could also help uncover inter-family and inter-country relationships by scanning APT samples, supporting attribution efforts, and helping researchers understand shared tooling and connections among APT families and countries.

In other words, this thesis investigates whether the YARA-rule match profile of an APT malware family, a purely static representation of which publicly available rules its samples trigger, carries discriminative signal about the family's country of origin and its relationships to other families. We address three research

questions:

- **RQ1 (Country separation)**. Can APT families be grouped by their country of origin using only their YARA-rule match profiles, with no dynamic analysis or analyst-provided attribution?
- **RQ2 (Country cohesion)**. Independently of any fixed partition, do families from the same country merge closer in the clustering hierarchy than would be expected by chance, and which countries, if any, drive this effect?
- **RQ3 (Family relationships).** Do YARA-rule profiles recover known relationships between individual families, beyond what shared nationality alone would explain?

To answer these questions, we assembled a dataset of 14,938 verified Windows PE/DLL malware samples spanning 78 APT families across 12 countries, and scanned them against a large corpus of publicly available YARA rules. Rules referencing a specific actor or campaign, targeting non-PE formats, or consisting solely of ephemeral indicators of compromise were filtered out, so that the remaining rules reflect technical tooling rather than encoded attribution. We aggregated the resulting matches to the family level as rule-firing rates and clustered the families using agglomerative hierarchical clustering across three representations (raw firing rate, binary presence, and TF-IDF). Clustering quality was measured with chance-corrected metrics (ARI, AMI) as well as homogeneity and completeness, and every finding was tested for significance and robustness with permutation methods: a maximum-statistic test that corrects for the feature-selection threshold, a Mantel test and per-country cohesion test for geographic structure, and a same-country ranking analysis for family relationships, the latter using ground-truth pairs derived automatically from MITRE ATT&CK and guarded by a leakage check.

The main contributions of this thesis are:

- A reproducible representation of APT families by their static YARA-rule match profiles, together with a filtered rule set and pipeline designed to exclude attribution-encoding rules.
- A rigorous, chance-corrected evaluation of whether static YARA signal alone can separate nation-states, establishing a real but weak country-level signal.
- A per-country cohesion analysis that localizes this signal, identifying North Korea as the only robustly cohesive country.
- A leakage-controlled method that recovers known inter-family relationships from YARA profiles, showing that they capture genuine shared tooling rather than encoded attribution.
- An honest characterization of the approach's limits, including the dominance of commodity tooling and the effect of obfuscation on static signal.

The remainder of this thesis is organized as follows. Section 2 reviews related work on APT attribution, malware-family clustering, and YARA-based detection, positioning our contribution with respect to existing approaches. Section 3 provides the background on APTs, malware analysis, YARA, and the clustering and evaluation techniques used. Section 4 describes the construction of the malware and YARA-rule datasets and the clustering methodology. Section 5 presents the results of the three analyses, and Section 6 discusses their interpretation, implications, and limitations. Section 7 concludes and outlines directions for future work.

2 Related Work

The problem of attributing malware and cyber-attacks to the actor or nation-state behind them has attracted significant attention from the research community and practitioners, and existing approaches differ mainly in the type of evidence they exploit. In this section we review this body of work along three axes that together frame our contribution. We first survey approaches to APT and nation-state attribution, distinguishing those that rely on dynamic behavior, on cyber-threat-intelligence text, and on static artifacts. We then turn to work that clusters malware and APT families or recovers the relationships between them, which is closest to our own analysis. Finally, we review how YARA has been used in prior work, and argue that its rules have not yet been exploited as a static feature space for country-level attribution, which is the gap this thesis addresses.

2.1 APT and Nation-State Attribution

The line of work most directly related to ours attributes malware to the nation-state that developed it. Rosenberg et al. [3] introduced DeepAPT, the first end-to-end nation-state attribution classifier. It feeds the sandbox report of a sample, i.e. its *dynamic* behavior, to a deep neural network and, on a test set of one thousand Chinese- and Russian-developed samples, reaches an accuracy of 94.6%. The same authors later extended the approach with transfer learning, jointly solving nation-state attribution and APT-family classification and raising the reported accuracy to 98.6% [12]. More recently, Shenderovitz and Nissim [13] proposed Bon-APT, which segments the API-call sequences observed during dynamic analysis to detect, attribute, and explain APT malware; they evaluate it on an unusually large collection of $12{,}655$ samples spanning 188 groups and 17 nations and report attribution at both the group and the nation level. These works establish that a country-of-origin signal exists in malware, but they share two properties that motivate our study, they depend on costly dynamic execution, and they classify each sample in isolation rather than exposing the shared tooling that links families and countries together. Closer to our own emphasis on a lightweight, mostly static pipeline, Kida and Olukoya [14] attribute both the country and the APT group using fuzzy hashes of the binary as input to machine-learning classifiers, reaching accuracies of 89% and 87.5% respectively while avoiding a sandbox; like ours, their method treats a compact static representation of the file as the feature, though they classify individual samples rather than clustering families by publicly available rule matches.

A second group of studies attributes malware using machine learning over features extracted from the binary or its execution. Rani et al. [15] extract operational TTPs directly from malware samples and map them to MITRE ATT&CK for attribution, arguing, as we do, that raw samples are a more faithful source than processed threat reports. Xu et al. [16] combine Adaboost feature selection with LightGBM to classify APT malware, while Li et al. [17] target the same problem in the IoT setting with an imbalance-aware random-forest variant. Particularly relevant to our static viewpoint, Wang et al. [18] attribute malware through binary similarity, mining time-series shapelets over sequences of basic blocks to obtain interpretable, path-level features. Mei et al. [19] instead attribute the APT organization from tool traces collected in a sandbox using a PSO-optimized multi-class SVM. Most closely aligned with our clustering perspective, Haddadpajouh et al. [20] propose a multi-view fuzzy consensus clustering model that fuses twelve views of a malware payload to attribute it to its actor, demonstrating that unsupervised grouping of samples carries an attribution signal, albeit over a handful of APT families and without the chance-corrected evaluation we adopt.

A large and comparatively less related body of work attributes attacks not from the malware itself but from cyber-threat intelligence (CTI) expressed as free text, TTPs, or indicators of compromise. Several studies cast attribution as a text-classification problem over analyst reports, including NO-DOUBT by Perry et al. [21] and the deep-learning extension of Naveen and Puzis [22], while Noor et al. attribute threats from high-level indicators of compromise [23] and later empirically compare high- against low-level attack patterns [24]. A further strand reasons over the ATT&CK matrix and playbooks of tactics. Edie et al. [25] mine association rules over TTPs and attribute them with a weighted Jaccard similarity; Kim et al. [26] and Shin et al. [27] vectorize the ATT&CK matrix and compare groups by cosine similarity; and Sachidananda et al. [28] correlate and attribute multi-stage alerts. Others enrich these representations with behavioral or graph-based features. For instance, Irshad and Siddiqui with hybrid technical and behavioral traits [29], Böge et al. with the command sequences of threat actors [30], Xiao et al. with a multimodal, multilevel fusion of IoC graphs [31], Wu et al. with TTP profiles mined from IoT honeypots [32], and Xu et al. with a

heterogeneous graph attention network over web-attack chains [33]. Knowledge-graph formulations of the same problem have also appeared [34, 35], together with argumentation-based reasoners that combine technical and social evidence [36] and network-forensic pipelines over traffic [37]. This text-driven paradigm continues to evolve. Indeed, Rani et al. [38] attribute APTs by encoding the TTP sequences found in threat reports along kill-chain phases and matching them to group profiles with a custom, order-aware similarity, while Guru et al. [39] evaluate off-the-shelf large language models for extracting TTPs and attributing actors, reporting that they surpass a random-guessing baseline but yield noisy technique sets. All of these methods depend on the availability and quality of analyst-written intelligence, whereas our approach requires only the binary and a corpus of public rules.

Finally, attribution is complicated by the deliberate anonymity of malware authors and by active deception. Research on binary authorship attribution seeks to recover an author's stylistic fingerprint from code. BinAuthor exploits an author's coding habits to identify the writer of a binary [40], and the survey of Gray et al. [41] reviews such techniques together with the adversarial methods used to defeat them. Bartholomew and Guerrero-Saade [42] document how sophisticated actors plant false flags to muddy attribution, a reminder—central to our motivation— that indicators an adversary can cheaply alter are unreliable, and that attribution should rest on artifacts that are costly to disguise.

2.2 Malware and APT-Family Clustering

A separate line of research, which our relationship analysis builds upon, groups malware or APT families and studies the connections between them rather than labeling samples individually. Chen et al. [43] cluster APT groups by a weighted similarity computed over ATT&CK techniques, software, target nations, and industries extracted from CTI reports, seeking to reveal collaborative patterns among groups. Li et al. [44] pursue the complementary goal of *automatically* discovering correlations between APT groups, modeling their behavior with rough-set theory and inferring links through a link-prediction scheme. Both aim, as we do in our third research question, to recover known and latent relationships between actors; unlike ours, they derive these relationships from analyst-curated technique inventories rather than from the static tooling present in the samples themselves. The same goal has been pursued at the level of adversary behavior by several groups. Wang et al. [45] cluster APT groups using a similarity measure derived from a *semantic* analysis of their TTPs, arguing that earlier document-clustering efforts captured only surface-level similarity; Xu et al. [46] quantify the relevance between an organization and its affiliated partners from the attribution traces in threat reports; and Wang et al. [47] assess whether a set of activities is organisationally *consistent*—that is, attributable to a single actor—with a self-contained model that requires neither external knowledge bases nor system logs, a question directly analogous to the same-actor alias recovery in our third research question.

Two studies are especially close to our methodology, as they apply hierarchical clustering to adversary knowledge to surface relationships that are not explicit in the data. Ding et al. [48] build a cybersecurity knowledge graph from ATT&CK, CAPEC, CWE, and vulnerability databases and then use hierarchical clustering to discover *hidden* patterns—for instance, which techniques are frequently used jointly by hacker groups—going beyond the one- and multi-hop queries that such graphs normally support. Al-Shaer et al. [49] likewise apply hierarchical clustering to the ATT&CK technique co-occurrences reported for APT and software groups, extracting 98 statistically significant technique associations and validating them with mutual information so that observed techniques can predict unobserved ones. Both works confirm that agglomerative hierarchical clustering, assessed with information-theoretic and significance-based criteria, can recover meaningful structure among adversaries, precisely the combination of tools we adopt in this thesis. The essential difference is one of representation. They cluster over analyst-curated ATT&CK techniques, whereas we cluster families directly from the static YARA-rule matches produced by their samples, requiring no manual technique labeling.

The influence of static features on malware-family clustering has been examined directly by Vitale et al. [50], who measure how often eleven popular static similarity features—fuzzy hashes, structural hashes, certificate- and icon-based features—erroneously group samples from different families across nearly 80,000 labeled Windows binaries. Their finding that build tools such as packers and installers severely distort static clustering is directly relevant to us, since it explains why commodity tooling can dominate a static signal and justifies both our rule-filtering step and our use of chance-corrected metrics. Earlier, Al Shamsi et al. [51] applied agglomerative hierarchical clustering—the same family of algorithms we adopt—to sequences of

API calls to discover behavioral similarities and new malware families, though on dynamic rather than static features. In a deliberately lightweight and static direction closer to ours, Ali Khan and Al-Awami [52] forgo machine learning entirely and detect IoT (ELF) malware using fixed-threshold similarity hashes, among them a Lempel–Ziv Jaccard distance, the same Jaccard-based notion of proximity we employ, although their aim is binary detection rather than actor attribution. Collectively, this work shows that clustering can expose structure among malware, but that the choice of feature representation strongly determines whether that structure reflects genuine relatedness or merely shared, non-discriminative components.

2.3 YARA-Based Detection and Rule Quality

YARA is widely used as a practical instrument for malware triaging and detection. A representative line of work by Naik et al. combines YARA with similarity-preserving hashes, they triage ransomware by comparing fuzzy hashing, import hashing, and YARA rules [53], and subsequently augment YARA rules with fuzzy hashing to raise detection rates without inflating rule complexity [9, 54]. Beyond triaging, YARA rules have been used to detect polymorphic malware in combination with mutation engines [55], generated automatically as family signatures through semi-supervised clustering of Android applications [56], coupled with fuzzy hashing and VirusTotal for Android profiling [57], and embedded in multi-step pipelines for classifying unknown samples [58]. In all of these settings, YARA serves as a detector or triaging filter, not as a feature space for characterizing the actor behind a sample.

Because our method treats rule matches as features, the quality and behavior of the underlying rules is important to us. Culling and Vandeven [11] contrast basic string-based rules with advanced rules that also encode behavioral and structural characteristics, showing that the latter generalize far better across variants of the same family. Canfora et al. [59] formalize this concern by defining robustness and looseness metrics for rules and analyzing tens of thousands of them, confirming that writing style strongly affects the quality of the indicators a rule captures. These observations directly inform our decision to filter out low-complexity, hash-like, and attribution-encoding rules before clustering. The work closest to using YARA for attribution is that of Dimitrov and Nikolov [10], who hand-craft YARA and Sigma rules, mapped to ATT&CK, to detect the Chinese state-sponsored “Typhoon” groups. Their rules are manually crafted and target a single, named group, whereas we take a complementary and inverse view. Rather than writing bespoke rules to recognize a known actor, we ask whether the profile of matches produced by a large corpus of *existing* public rules carries, purely statically, a discriminative signal about a family’s country of origin and its relationships to other families, a question, to the best of our knowledge, not previously studied.

3 Background

3.1 Advanced Persistent Threats (APT)

In today's digital landscape, cyberattacks vary significantly according to their target, objective, and overall sophistication. Highly complex, well-resourced attacks characterized by specific motivations, clear objectives, minimal attribution signals, and sophisticated evasion techniques are defined as Advanced Persistent Threats (APTs) [60, 61].

Carrying out APT operations requires a considerable amount of financial, technical, and human resources. Consequently, the threat actors behind these attacks are typically nation-states or corporate entities. These operations leverage highly sophisticated evasion techniques alongside advanced zero-day vulnerabilities [61]. Furthermore, attackers actively camouflage their tools and deliberately encourage false attribution through techniques like false-flag indicators [62, 63], making APTs exceptionally difficult to detect and trace to a specific threat actor [3].

At a foundational level, the primary purpose of APT operations is to obtain strategic information, execute infrastructure sabotage, or both [64]. Threat actors achieve these goals by deploying carefully developed Tactics, Techniques, and Procedures (TTPs)–such as those cataloged in the MITRE ATT&CK [65] framework–explicitly tailored to specific organizations or nation-states [66]. Rather than seeking quick financial payouts, APT operations deliberately target high-value commercial enterprises, government bodies, and critical infrastructure networks to secure long-term geopolitical and economic advantages [67, 68, 69, 70]. Attackers pursue these broader objectives through strategic espionage, infrastructure disruption, information operations, and state-sponsored cyber warfare [71]. This thesis focuses specifically on nation-state-sponsored APTs. Lockheed Martin created a famous model called the Cyber Kill Chain to explain how complex cyberattacks work by breaking them down into a step-by-step process [72]. In this model, an attacker must successfully complete every single stage in exact order to reach their final goal. The major benefit for security teams is that they only need to detect and block one of these steps to ruin the hacker's entire plan. Furthermore, the earlier the attack is stopped in the chain, the less damage the attacker can do. Because of this, the Cyber Kill Chain has become the standard blueprint the cybersecurity industry uses to understand and fight advanced threats.

To better understand APT behavior, many authors have described their attack lifecycles in various stages. Hutchins et al. [73] introduced an intrusion kill chain model to describe the distinct phases of an APT attack. In this model, to reach its final objective, an adversary must complete every stage of the kill chain in sequence. The kill chain consists of seven phases: reconnaissance, weaponization, delivery, exploitation, installation, command and control (C2), and actions on objectives. Various other authors have subsequently described APT attack lifecycles using alternative phase models [74, 75, 76].

3.1.1 Tactics, Techniques and Procedures (TTP)

In 2013, MITRE began developing ATT&CK® [77], a curated taxonomy of adversary Tactics, Techniques, and Procedures (TTPs) aligned with the various phases of an attack lifecycle. This threat model was initially created within a MITRE research project called FMX, with the goal of enhancing post-compromise detection of threats penetrating enterprise networks through telemetry data and behavioral analytics.

In this knowledge base, MITRE defines a set of techniques, sub-techniques, and procedures that attackers perform to achieve their tactical objectives (tactics). Tactics represent the adversary's overarching goals, answering why an attacker would employ the specific techniques or sub-techniques associated with that category.

Techniques explain how an adversary achieves a tactic. Additionally, some techniques are divided into sub-techniques, which provide a more granular description of how these objectives can be accomplished. Because there are many different ways for attackers to achieve their tactical goals, each tactic category contains multiple techniques, and similarly, many techniques contain multiple sub-techniques.

Procedures refer to the specific, observed real-world implementations of these techniques and sub-techniques used by adversaries [65]. The structural relationships among these tactics, techniques, and sub-techniques are visualized in the MITRE ATT&CK Matrix [78].

3.2 Malware Analysis

Malware analysis aims to extract as much information as possible from a discovered binary sample to identify and comprehend the associated threat [79]. There are two primary techniques for achieving this: static and dynamic analysis [80].

3.2.1 Portable Executable (PE) & DLL

While various binary executable file formats exist across different operating systems–such as ELF for Linux and Mach-O for macOS [81, 82], this methodology exclusively focuses on the Portable Executable (PE) and Dynamic-Link Library (DLL) formats. These represent the native executable architectures used in the Microsoft Windows operating system [83]. Because Windows maintains a dominant global market share in desktop and enterprise computing, PE and DLL binaries consequently constitute the vast majority of active malware threats [84]. Restricting our dataset to these specific formats ensures that our analysis targets the most prevalent and impactful vectors utilized in modern cyber-espionage and geopolitical threat campaigns.

3.2.2 Static Malware Analysis

Static analysis examines a malware's binary structure to determine its functionality without executing the malicious file [85]. By employing techniques such as behavioral pattern matching, string extraction, and the analysis of API calls, functions, and data structures, analysts can extract specific characteristics to define the type of file or software [86]. The execution can take different forms, from inside a malware sandbox to emulation, and from symbolic execution to bare metal an other hybrid forms [87, 88, 89, 90, 91, 92]. Nonetheless, malware obfuscation poses a significant challenge to this process. Because a malware's functionality and intent can be hidden using encryption or packing, these evasion techniques often lead to the extraction of inaccurate or unusable information [79]. Consequently, obfuscated executables must be decrypted or unpacked prior to analysis [93]. Despite these challenges, static analysis remains a foundational technique due to its efficiency, scalability, and ability to expose code paths that might not execute during dynamic analysis [94]. Therefore, it is the first line of defense. Numerous automated static analysis tools are available that leverage the known syntax and structural properties of a binary to extract these characteristics [79]. A primary example of such a tool is YARA [95].

3.2.3 Dynamic Malware Analysis

Dynamic analysis examines malware behavior by executing the binary within a controlled environment, such as a sandbox or a virtual machine [96]. The malware can be monitored during runtime, or the system's state can be analyzed after execution [85]. Observable malicious functionality typically includes invoked system and API calls, network activity, file system changes, and registry modifications [79, 93]. Dynamic analysis complements static techniques by allowing the analyst to record the malware's actual behavior, as the mere existence of an action string in a binary does not guarantee its execution [85, 94]. Furthermore, by leveraging cloud-based environments, researchers can dynamically detonate malware at scale to observe these real-time behavioral patterns, allowing them to develop stronger defensive countermeasures [86].

3.3 Yara Rules

YARA, widely referred to as the pattern-matching "Swiss Army knife" for malware researchers [8], is a tool developed to assist analysts in the identification and classification of collected samples [95]. Developed by Victor Alvarez in 2008, YARA was introduced as a "simple and rule-driven" tool to supplement manual malware analysis [97]. Its primary goal was not to replace human analysis but to complement it by mitigating human limitations, such as memory constraints and the difficulty of sharing scattered information across analyst teams. It resolved these issues by transforming the collective knowledge of malware analysts into a set of machine-readable rules [97]. In other words, YARA rules identify malicious files and active processes by defining the unique strings or raw byte sequences associated with a specific threat, and subsequently scanning the target file for those exact patterns [9].

3.3.1 Syntax

A rule consists of text or binary strings connected by logical operators within a structured, computer-understandable syntax [97]. More specifically, a YARA rule consists of two primary sections: the strings section and the condition section. Additionally, rules may include an optional meta section containing descriptive information about the rule itself [98].

Strings

Strings are the predetermined signatures expected to match when scanning a target. Typically, the strings section defines sequences that are unique to malware samples and absent from legitimate programs [56]. YARA supports three types of strings: hexadecimal strings, text strings, and regular expressions. Text strings are the simplest type: ASCII-encoded, case-sensitive sequences used for exact matching. However, they can be made more flexible by appending special modifiers directly after the string declaration. Hexadecimal strings integrate four special features that enhance matching flexibility: wildcards, not operators, jumps, and alternatives. Finally, regular expressions can also be accompanied by modifiers, making them one of the most powerful features in YARA [98].

Condition

The condition section contains the Boolean logic that determines whether the scanned file satisfies the rule using the defined strings. Conditions can incorporate Boolean, relational, arithmetic, and bitwise operators [98].

Listing 1 presents an example of a YARA rule, taken from Alvarez's publication [97]. The rule triggers on any file that contains the strings `foo` or `bar` and lacks the hexadecimal string `$c=0x12 0x34 0x56 0x78 0x9A`. In this example, the author has also included a metadata section that describes the rule as "Bad file". While this is a basic example, analysts can construct much more complex rules utilizing the various string types and modifiers defined previously, as well as conditions. Additionally, YARA rules can import modules to extend their core capabilities. Each module provides unique functions that become available upon importation into the YARA file. For instance, when analyzing Portable Executable (PE) binary files, the pe module is highly valuable, as it includes variables and functions specifically designed for parsing the PE file format [98].

Listing 1: Example of a basic YARA rule structure.

```
rule exampleRule
{
    meta:
        description = "Bad file"
    strings:
        $a = "foo"
        $b = "bar"
        $c = { 0x12 0x34 0x56 0x78 0x9A }
    condition:
        ($a or $b) and not $c
}
```

3.4 Clustering

Clustering is a form of unsupervised learning. Because unsupervised learning operates without labeled data, its primary goal is to discover hidden structures within a dataset [99]. Clustering involves a set of techniques designed to group data objects into subsets, ensuring that objects within the same cluster are highly similar to one another while remaining distinctly dissimilar to observations in other clusters. The level of similarity (or dissimilarity) between two objects is typically assessed using distance metrics applied to their features, which is often a domain-specific consideration [100, 99]. As an unsupervised technique, clustering is frequently performed as part of exploratory data analysis to understand structural information without relying on predefined labels [99, 101].

Figure 3.1 provides an example of a set of observations in a two-dimensional space. The discrimination among the three well-formed groups is visually obvious to the human eye because the data has only two dimensions (where dimensions equal the number of features per observation). However, real-world datasets typically consist of multi-dimensional data that is impossible to visualize. Therefore, clustering algorithms are required to automatically identify these hidden formations [101].

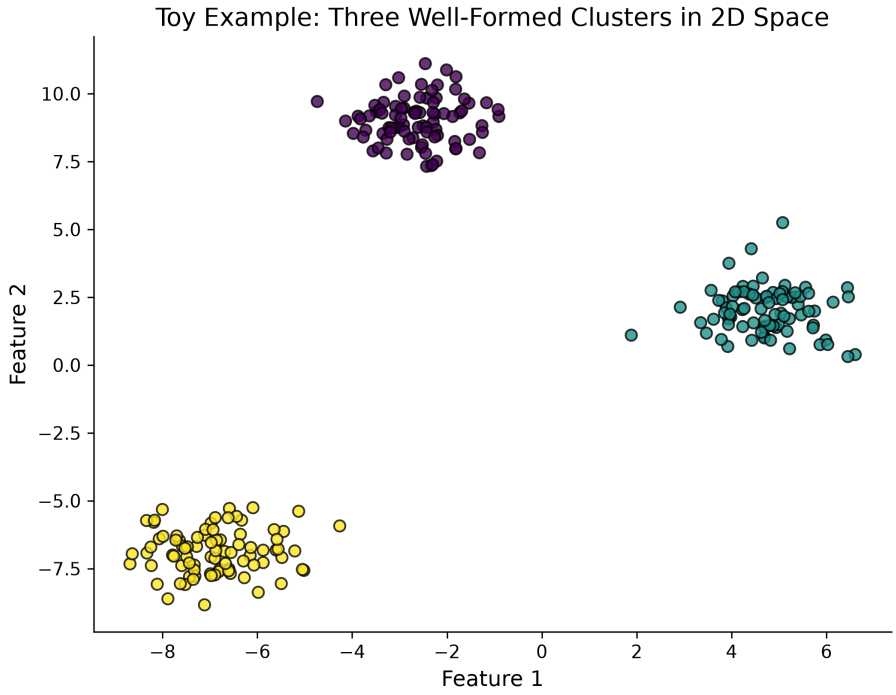


Figure 3.1: A two-dimensional visualization of clustered data points demonstrating distinct group separations.

Furthermore, different clustering algorithms may generate completely different subset formations from the exact same dataset [100]. Due to their broad applicability, there is ongoing research and a wide variety of clustering algorithms available [102].

3.4.1 Distance Metrics

To use clustering algorithms, we need to define a level of proximity between pairs of objects in our dataset. This proximity measure represents either a similarity or a dissimilarity between two objects [103]. Specifically, our methodology uses the Jaccard distance (a dissimilarity measure) and Cosine similarity to define the relationship between two samples in our dataset.

Jaccard Distance

When dealing with binary or nominal data, proximity is typically measured using matching coefficients. Let v_i and v_k be the binary feature vectors of the i^{th} and k^{th} observations (rows), respectively. The Jaccard coefficient between the two vectors is based on the contingency table shown in Table 3.1:

In Table 3.1, a_{11} represents the total number of attributes that are 1 in both binary vectors. a_{10} represents the number of attributes that are 1 in v_i and 0 in v_k, and so forth. Consequently, the sum $a_{11} + a_{10} + a_{01} + a_{00} = d$, where d equals the total number of features in our dataset.

Based on this, the Jaccard metric is defined as:

$$D_{\text{Jac}}(v_i, v_k) = \frac{a_{11}}{a_{11} + a_{01} + a_{10}} = \frac{a_{11}}{d - a_{00}} \tag{3.1}$$

Table 3.1: Contingency table for binary feature vectors v_i and v_k

		v_k = 1	v_k = 0
v_i	1	a_{11}	a_{10}
	0	a_{01}	a_{00}

It is important to note that the Jaccard metric accounts for the matches of 1s and ignores the matches of 0s. In other words, it bases similarity on the common ”presence” of a feature in two observations rather than on their common ”absence” [103].

Cosine Similarity

Let i and k be two rows in our dataset, and v_i and v_k be the feature vectors describing the two observations, respectively. Assuming each feature vector consists of two attributes, the two observation points can be visualized in a 2D space, as shown in Figure 3.2.

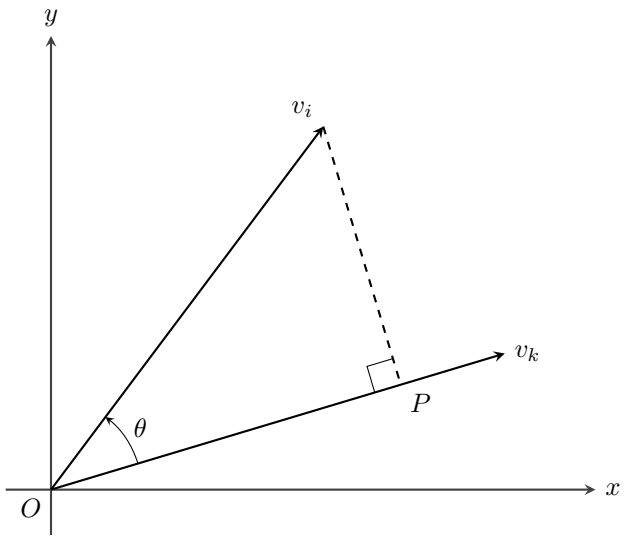


Figure 3.2: Geometric representation of the two observations $v_i = (x_i, y_i)$ and $v_k = (x_k, y_k)$.

The cosine of the angle θ formed by the two vectors serves as a measure of similarity. The larger the cosine value (closer to 1), the more parallel the two vectors are. In contrast, a smaller cosine value indicates that the directions of the two vectors are further apart. As is well known, the cosine of an angle is not influenced by the length (magnitude) of the vectors. Thus, it is clear that cosine similarity does not account for the magnitude of the feature vectors [104]. In general, if v_i and v_k are two observations in our dataset, the Cosine Distance is defined as [101]:

$$D_{\cos}(v_i, v_k) = 1 - \frac{v_i^\top v_k}{\|v_i\|\|v_k\|} \tag{3.2}$$

3.4.2 Dimensionality Reduction

The number of features the objects in our dataset have is known as the dataset’s dimensionality.

Curse of Dimensionality

As the number of dimensions (features) in our dataset increases, the space that the data occupies becomes increasingly sparse. Thus, the definition of distance, which is crucial for clustering algorithms, loses its meaning. As a result, many clustering algorithms perform poorly on high-dimensional data. This problem is known in data analysis as the curse of dimensionality [105].

Non-Negative Matrix Factorization (NMF)

Non-Negative Matrix Factorization (NMF) is a dimensionality reduction and feature extraction technique particularly well-suited for datasets composed strictly of non-negative values. Given an original data matrix X of size $M \times N$ (where M represents the original high-dimensional features and N represents the observations), NMF seeks to approximate this matrix as the product of two lower-dimensional, non-negative matrices: a basis matrix U and a coefficient matrix V. Mathematically, this decomposition is represented as:

$$X \approx U \times V \tag{3.3}$$

To successfully achieve dimensionality reduction, an inner dimension L is chosen such that $L \ll \min(M, N)$. By forcing the algorithm to reconstruct the original data matrix through this narrow bottleneck of L latent features, the NMF model must learn the most fundamental, intrinsic patterns of the dataset. Consequently, each observation is re-represented as a linear combination of these fewer latent basis vectors. This process effectively compacts the data, extracting the underlying structure and mitigating the curse of dimensionality before further clustering algorithms are applied [106, 107]. The primary advantage of NMF lies in its strict non-negativity constraint, which naturally yields a parts-based representation of the data. By reconstructing the original dataset strictly through the additive combination of smaller sub-matrices, NMF ensures that the resulting latent features remain highly interpretable [108].

3.4.3 Evaluation Metrics

To evaluate the quality of our clustering results, we must define appropriate evaluation metrics. There are two primary categories of clustering evaluation indices: intrinsic (internal) and extrinsic (external) [109]. In this research, because the ground-truth labels for our dataset are known, we employ extrinsic evaluation methods.

Adjusted Rand Index

The Rand Index, introduced by Rand in 1971, is an evaluation metric based on pair counting [110]. Let $S = \{s_1, s_2, \ldots, s_N\}$ be a dataset of N instances. When ground-truth labels are available, we can compare two partitions of S: C, the clustering produced by our unsupervised method, and T, the ground-truth clustering. Considering all $\binom{N}{2}$ possible pairs of data points, the Rand Index counts the pairs on which the two clusterings *agree*, namely those assigned to the same cluster in both C and T, and those assigned to different clusters in both C and T [111]. Denoting by N_s the number of pairs of the first type and by N_d the number of pairs of the second, the Rand Index is defined as

$$RI(\mathbf{C}, \mathbf{T}) = \frac{N_s + N_d}{\binom{N}{2}}. \tag{3.4}$$

The Rand Index takes values between 0 and 1, where 0 indicates that the two clusterings agree on no pairs, and 1 indicates that they are identical.

While the Rand Index provides a straightforward measure of agreement, it suffers from a significant limitation: its expected value for two random partitions is not constant. If we compare completely random clusterings, the baseline RI can vary widely depending on the number of clusters and the number of data points within each cluster. This shifting baseline makes it difficult to determine whether a moderately high

RI indicates a genuinely meaningful clustering or is merely a byproduct of random chance.

To address this issue, Hubert and Arabie (1985) proposed the Adjusted Rand Index (ARI) [112]. The ARI corrects the original index by establishing a baseline where the expected value of two random clusterings is exactly zero. To calculate this expected value, they modeled randomness using the generalized hypergeometric distribution. In practical terms, this means that the ”random guess” baseline is calculated by assuming the two clusterings are generated randomly, while strictly maintaining their original number of clusters and cluster sizes.

The adjustment follows a standard statistical correction framework, formulated as:

$$\text{Adjusted Index} = \frac{\text{Index} - \text{Expected Index}}{\text{Max Index} - \text{Expected Index}} \tag{3.5}$$

In this formula, the *Expected Index* represents the average agreement score obtained if the clusterings were completely independent, while the *Max Index* represents the theoretical maximum possible agreement (typically 1).

By applying this correction, the ARI yields a value bounded between -1 and 1. An ARI of 1 indicates that the two clusterings are identical. An ARI of 0 implies that the clustering agreement is exactly what would be expected by pure random chance, and negative values indicate an agreement worse than expected by chance.

Adjusted Mutual Information

Similar to the Adjusted Rand Index, Vinh, Epps, and Bailey developed a chance-corrected metric for another fundamental class of clustering comparison criteria: information-theoretic measures [111]. Building upon foundational concepts such as Mutual Information [113] and the Variation of Information [114], the authors introduced the Adjusted Mutual Information (AMI).

The AMI serves as an extrinsic, information-theoretic evaluation metric. While traditional information-theoretic measures effectively quantify the mutual dependence or shared information between two clusterings, they often lack a constant baseline value. The AMI was specifically designed to rectify this by ensuring that the metric lies within a predetermined range–bounded above by 1–and that it maintains a constant baseline of zero when clusters are generated by chance. This fixed baseline significantly enhances the intuitiveness and reliability of clustering comparisons.

The mathematical intuition behind the AMI stems from a notable vulnerability in unadjusted measures like the standard Mutual Information (MI) or Normalized Mutual Information (NMI). When comparing clusterings, particularly when the ratio of data points to clusters is small, unadjusted measures exhibit a considerable inherent bias; their scores are artificially inflated by random chance. To correct this, the AMI calculates the *expected* mutual information. This expected value represents the average mutual information that would be observed if the data points were distributed randomly, while maintaining the original number of clusters and their sizes.

By subtracting this expected value from the observed mutual information, the metric isolates the genuine structural agreement between the partitions. The AMI thus follows the exact same standard statistical correction framework used by the ARI: subtracting the expected index from the raw index, and dividing by the maximum possible index minus the expected index [111].

$$AMI(U,V) = \frac{I(U,V) - E\{I(M)\}}{\sqrt{H(U)H(V)} - E\{I(M)\}} \tag{3.6}$$

Homogeneity and Completeness

When Rosenberg and Hirschberg introduced the V-measure, they also established two intuitive evaluation metrics: homogeneity and completeness [115]. A clustering satisfies the homogeneity criterion if each cluster contains *only* data points belonging to a single class. Conversely, a clustering satisfies the completeness criterion if *all* data points belonging to a specific class are assigned to the same cluster. It is entirely possible

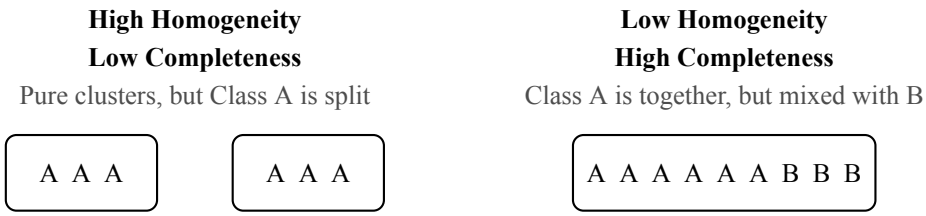


Figure 3.3: High homogeneity with low completeness (left) versus high completeness with low homogeneity (right).

for a clustering result to exhibit high homogeneity but low completeness, and vice versa. Figure 3.3 presents an example of each scenario.

3.4.4 Hierarchical Clustering

Hierarchical clustering produces a binary merge tree, in which the leaves represent individual data elements as singleton clusters. The algorithm iteratively merges the two least dissimilar subsets at a time until it reaches a single overarching subset, known as the root, which contains all the elements of the dataset [101]. This tree-like representation of the dataset is called a dendrogram. Furthermore, this specific bottom-up merging technique defines a category called agglomerative hierarchical clustering, which is the most widely used form of the algorithm [99].

The dissimilarity between data points is quantified using a distance metric. Popular choices for measuring the distance between two individual observations include Cosine similarity, Hamming distance, Jaccard distance, Manhattan distance, and Minkowski distance. However, while these metrics describe the distance between two discrete points in an n-dimensional space, the algorithm requires a method for calculating the dissimilarity between two clusters (subsets) or between a single element and an existing cluster. These subset-level measurements are known as linkage criteria. The three primary types of linkage are Single Linkage, Complete Linkage, and Group Average Linkage [101].

Linkage Criteria

Hierarchical clustering algorithms iteratively merge data, which can involve grouping two individual observations (singletons), merging a singleton with an existing cluster, or combining two established clusters. So far, we have defined two distance metrics for merging individual observations. Now, we introduce three linkage criteria for evaluating the distance between clusters.

Let $D(v_i, v_k)$ denote the distance between two individual points, computed using one of the aforementioned proximity measures. Furthermore, let $\Delta(C_A, C_B)$ represent the calculated distance between two subsets (clusters) of observations, C_A and C_B. The three primary linkage criteria used to determine this inter-cluster distance are defined as follows [101]:

1. *Single Linkage:*
$$\Delta(C_A, C_B) = \min_{v_i \in C_A, v_k \in C_B} D(v_i, v_k) \tag{3.7}$$
2. *Complete Linkage:*
$$\Delta(C_A, C_B) = \max_{v_i \in C_A, v_k \in C_B} D(v_i, v_k) \tag{3.8}$$
3. *Group Average Linkage:*
$$\Delta(C_A, C_B) = \frac{1}{|C_A||C_B|} \sum_{v_i \in C_A} \sum_{v_k \in C_B} D(v_i, v_k) \tag{3.9}$$

Dendrogram Interpretation

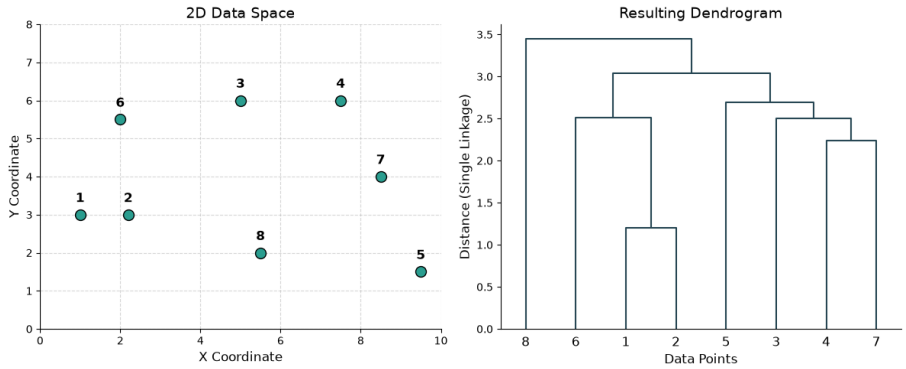


Figure 3.4: A dendrogram illustrating the hierarchical relationships and fusion distances between eight observations.

A dendrogram visually represents the hierarchical relationships and similarities between data points. Looking at the resulting dendrogram in Figure 3.4, each leaf along the bottom horizontal axis corresponds to one of the eight observations from the two-dimensional data space shown on the left. As we move vertically up the tree, individual leaves fuse together to form branches. The fundamental rule for interpreting this structure is that the vertical axis, representing distance or height, measures the dissimilarity between observations.

Data points that merge lower on the tree are more similar to one another. For example, observations 1 and 2 are the most similar in the dataset, fusing at the lowest point on the graph at a distance of approximately 1.2. Observations 4 and 7 are the next most similar pair. Conversely, fusions that occur higher up indicate a greater difference between the points. Observation 8 is the most distinct point in the dataset, as it fuses only with the other observations near the very top of the dendrogram, at a distance of roughly 3.45.

It is a common mistake to judge the similarity of two observations based on how close they are on the horizontal axis. However, this is not correct. For instance, observation 6 sits directly next to observation 1 on the horizontal axis, but observation 1 is actually much more similar to observation 2, as proven by their low vertical fusion height. Therefore, conclusions about similarity must be drawn strictly based on the vertical-axis location at which branches containing those observations first fuse.

A key advantage of a dendrogram is that it allows researchers to generate any desired number of clusters from a single chart by making a horizontal cut across the tree. The distinct branches that fall below the cut line become the clusters. Making a cut at a height of 3.2 results in two clusters: one containing only observation 8 and the other containing the remaining points. Lowering the cut to a height of 2.8 yields three clusters, separating observation 8, the group of 6, 1, and 2, and the group of 5, 3, 4, and 7. By evaluating the heights of these fusions, one can visually determine a sensible number of clusters for a specific dataset [99].

Generic Algorithm for Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering operates as a bottom-up approach, relying on a predefined linkage distance function, denoted as Δ, which is derived from a chosen distance metric between individual elements. The algorithm initializes by assigning each data point x_i in the dataset X to its own single-element cluster, $G_i = \{x_i\}$. These initial clusters are placed into an active list.

The core of the algorithm proceeds iteratively: as long as there is more than one cluster remaining in the list, it identifies the pair of clusters, G_i and G_j, that exhibit the smallest linkage distance $\Delta(G_i, G_j)$. Then these two most similar groups are merged to form a new combined cluster $G_{i,j} = G_i \cup G_j$. This new cluster is added to the active list, while the two original clusters are removed.

This merging process repeats until only a single cluster remains. This final group, encompassing the entire dataset X, constitutes the root of the resulting dendrogram. Because the procedure begins with $n = |X|$ individual leaves and continually merges them until a single comprehensive root is formed, the algorithm necessitates exactly $n - 1$ distinct merging operations to complete [101].

3.5 The Mantel Test

The Mantel test is a permutation-based procedure for assessing the association between two distance (or dissimilarity) matrices defined over the same set of objects [116]. It was originally proposed by Mantel to detect space–time clustering of disease cases, but it has since become a standard tool for comparing dissimilarity structures, particularly in ecology and related fields [117]. The test answers a simple question: given two ways of measuring how far apart the same n objects are, do the two resulting distance matrices vary together?

Let $\mathbf{X}$ and $\mathbf{Y}$ be two symmetric $n \times n$ distance matrices computed over the same objects. The standardized Mantel statistic is the Pearson correlation between the corresponding upper-triangular entries of the two matrices,

$$r_M = \frac{1}{d-1} \sum_{i<j} \left(\frac{x_{ij} - \bar{x}}{s_x} \right) \left(\frac{y_{ij} - \bar{y}}{s_y} \right), \tag{3.10}$$

where $d = \binom{n}{2}$ is the number of distinct off-diagonal pairs, and $\bar{x}, s_x$ (respectively $\bar{y}, s_y$) denote the mean and standard deviation of the off-diagonal entries of $\mathbf{X}$ (respectively $\mathbf{Y}$). The coefficient r_M ranges from -1 to 1 and is interpreted like an ordinary correlation: positive values indicate that large distances in one matrix tend to coincide with large distances in the other.

Since the data points in the matrices are dependent, we cannot use standard mathematical formulas to find the significance of r_M. Instead, we use a random shuffling method called permutation [116, 117]. If we assume there is no real connection between the two matrices, the exact order of the data in one matrix does not matter. To test this, we randomly shuffle the rows and matching columns of one matrix at the same time and recalculate r_M. By repeating this process many times, we build a baseline of random results. The p-value is simply the percentage of these random shuffles that produce a score as extreme as, or more extreme than, our actual r_M. This shuffling technique is accurate without requiring specific data distributions and works perfectly even when the distances depend on each other, which is why it is so widely used for comparing distance or dissimilarity matrices [117].

3.6 Term frequency and weighting

In the context of Information Retrieval, terms occurring in a collection of documents are weighted according to their frequency within each document, as well as the number of documents in which they appear [118]. More specifically, given a collection of N documents, the Term Frequency of term t in document d ($TF_{t,d}$) is defined as the number of times term t occurs in document d. The Document Frequency (df_t), in turn, is defined as the number of documents in the collection that contain term t. Terms that occur in many documents are less discriminative and should therefore be down-weighted. To capture this intuition, we define the Inverse Document Frequency (IDF):

$$IDF_t = \log \frac{N}{df_t} \tag{3.11}$$

Consequently, rare terms are assigned higher IDF values than more common ones.

TF-IDF Weighting

TF-IDF weighting assigns a weight to each term t in each document d, defined as:

$$\text{TF-IDF}_{t,d} = TF_{t,d} \times IDF_t \tag{3.12}$$

This formulation yields:

1. higher weights for terms that occur frequently in few documents;
2. lower weights for terms that occur rarely in a document, or across many documents;
3. the lowest weights for terms that occur in all documents.

Table 3.2: Worked example of TF-IDF weighting over $N = 4$ documents (logarithm base 10).

Term t	$TF_{t,d}$	df_t	$IDF_t = \log \frac{N}{df_t}$	TF-IDF$_{t,d}$
the	5	4	0.000	0.000
data	1	3	0.125	0.125
network	3	2	0.301	0.903
malware	2	1	0.602	1.204

In summary, the scheme rewards terms with high discriminative power while penalizing less informative ones. Table 3.2 illustrates a worked example of TF-IDF weighting for a collection of $N = 4$ documents.

4 Methodology

4.1 Malware Dataset

Sample Acquisition

The dataset was assembled from two malware repositories: VXUnderground [119] and MalwareBazaar [120]. VXUnderground samples were drawn from its locally maintained, APT-sorted corpus, where each sample's initial APT label was inferred from the name of the collection folder in which it resided. MalwareBazaar samples were collected separately by querying the platform for APT group and tag names taken from public APT attribution lists and its available APT labels; matching samples were then downloaded and, where available, extracted.

As an external reference, we used the ADAPT dataset (`ADAPTDataset.csv`) [121], which provided hash-level label confirmation, APT alias normalization, and country mapping where possible. This reference comprised $6{,}455$ labeled hashes.

Verification and Deduplication

Each candidate sample was validated as a genuine Windows Portable Executable (PE) file through header and file-type inspection. Only PE executables and dynamic-link libraries (DLLs) were retained, while all non-PE files were discarded. The retained samples were then normalized by their SHA-256 digest, and duplicate hashes appearing across both sources were collapsed into a single sample record.

Labeling and Alias Normalization

Initial family labels were derived from each sample's source: VXUnderground labels came primarily from its APT-sorted folder structure, whereas MalwareBazaar labels came from its query and tag metadata. When a sample's SHA-256 hash matched an ADAPT entry directly, the ADAPT label was used to confirm or override the source label.

Because the same actor frequently appears under several names, APT aliases were canonicalized so that equivalent names mapped to a single family label. For example, variants such as *APT29*/*APT 29*, *Lazarus Group*/*Lazarus*, and *Gamaredon Group*/*Gamaredon* were unified. This normalization was performed using a manually curated alias dictionary together with ADAPT-derived alias metadata, ensuring that every variant of an actor was mapped to one canonical APT name.

Country Attribution

Country labels were assigned by mapping each canonical APT name to its reported country of origin. This mapping drew on the ETDA APT group profiles [122] and a locally compiled `apt_attributions.csv` file aggregating attributions from established threat-intelligence sources, including MITRE ATT&CK, FireEye/-Mandiant, Kaspersky, CrowdStrike, Palo Alto Networks, Proofpoint, Microsoft, SecureWorks, Cisco Talos, ESET, Trend Micro, APTmap, and the MISP Galaxy [123], together with ADAPT hash-level metadata and MalwareBazaar country fields where available. Ambiguous or conflicting attributions were manually reviewed against public threat-intelligence reporting and corrected prior to inclusion.

Conflict Handling

Where multiple sources disagreed on a sample's label, the conflict was recorded and resolved using a fixed order of precedence: direct hash-level ADAPT matches were preferred first, followed by VXUnderground folder labels, and finally MalwareBazaar tag metadata. Samples whose labels remained vague, or for which no reliable group or country mapping could be established, were excluded from the final dataset.

Minimum-Support Filtering and Final Dataset

To ensure that the dataset was statistically usable for clustering, only APT families with sufficient sample support were retained, applying a minimum threshold of 20 samples per family. After all verification, deduplication, labeling, and filtering steps, the final raw dataset comprised 14,938 verified PE/DLL samples, spanning 78 APT families across 12 countries.

4.2 Yara Rules Dataset

Our dataset consists exclusively of publicly available YARA rules. In total, we gathered a corpus of 15,660 YARA files containing 164,136 YARA rules. A list of all the sources from which we obtained these rules is available at the end of this thesis. We excluded repositories targeting file formats other than PE executables and DLLs, as our malware dataset consists solely of these file types.

Two special cases: IceWater and HydraDragon

During the selection of our YARA rule dataset, we came across these two well-known and incredibly large repositories [124, 125]. After manual examination of their content, we decided to discard the IceWater repository and pick only a handful of YARA files from the HydraDragon repository. The first decision was based on the fact that the IceWater repository contains rules that are automatically generated and are mostly hash-based fingerprints and small hexadecimal patterns with minimum complexity. Thus, their discriminative power is assumed to be insignificant, if any. The second decision was based on the HydraDragon repository containing many YARA files labeled as deprecated, duplicated, or obsolete. Consequently, we excluded these files from our dataset.

Automated Yara Rules Filtering

The goal of this step is to act as a first filtering process that will keep only useful, informative, and discriminative YARA rules targeting PE executables that indicate shared tooling and a common development environment. In other words, we try to keep rules that do not discriminate nation-states and APT families based on their technical artifacts, but rather on analysts' prior family- or infrastructure-specific knowledge.

Parsing is performed with plyara [126], a dedicated YARA parser. This ensures that import statements, private and global rules, inter-rule dependencies, and the distinction between strings, hex patterns, and comments are all handled correctly. Because a single file may contain many rules, the script operates at the granularity of the individual rule: it removes only the unsuitable rules and re-emits the surviving ones together with their import lines and any private or global rules they depend on, so that the output still compiles.

A rule is discarded if it satisfies any of the following conditions:

1. **Actor / APT attribution.** The rule name, tags, or metadata reference a known threat actor, either through a naming convention (`APT##`, `FIN#`, `TA###`, `UNC####`, MITRE group identifiers) or through an explicit group or campaign name (e.g. Lazarus, Turla, Fancy Bear). Such rules are removed to prevent attribution from leaking directly into the feature set.

2. **Non-Windows target format.** The rule targets a format other than Windows PE. This could be formats including PDF, Office/VBA macros, Android, iOS/macOS (Mach-O), ELF/Linux, or web shells, identified either from the author's own labels or from format-specific content markers. Rules that additionally exhibit a Windows-PE signal are retained.

3. **Indicators of Compromise (IOCs).** The rule contains only hard-coded IP address, URL, domain, e-mail address, or filesystem path. These are excluded since they provide only ephemeral and/or infrastructure-specific information, thus lacking robustness to potential malware variants and discriminative value. In general, we use the following logic: If a rule consists of Indicators of Compromise exclusively, then it is discarded. If a rule contains at least one string, structural anchor (e.g. import hash, entropy, or PE-structure checks), code, or hex pattern, then it is kept.

4. **Compilation failure.** The rule does not compile, as validated with yara-python [127].

Every rule's fate is recorded in a per-rule decision log, providing a fully auditable account of which rules were kept or removed and why.

After this initial filtering, we are left with 12,206 YARA files containing 113,344 YARA rules.

4.3 Binary YARA Rules Match Matrix

Having prepared both the curated malware dataset and the YARA rules dataset, we scan the malware samples using our corpus of YARA rules. The result is exported as a binary YARA rule match matrix (or simply a YARA hit matrix).

Let $S = \{s_1, \ldots, s_n\}$ denote the set of malware samples and $R = \{r_1, \ldots, r_m\}$ the set of YARA rules. The binary hit matrix $H \in \{0,1\}^{n \times m}$ is defined as

$$H_{ij} = \begin{cases} 1, & \text{if rule } r_j \text{ matched sample } s_i, \\ 0, & \text{otherwise,} \end{cases} \qquad i \in \{1, \ldots, n\},\ j \in \{1, \ldots, m\}. \tag{4.1}$$

Additional columns indicate the family of each malware sample as well as its country of origin. An excerpt of this matrix can be seen in Table 4.1. Because the vast majority of rules never matched any sample, we retain only rules that fired on at least one sample; this reduces the $113{,}344$ scanned rules to the $8{,}962$ columns of the hit matrix.

Table 4.1: Excerpt of the binary YARA-rule match matrix ($14{,}938$ samples $\times$ $8{,}962$ rules).

Sample	Family	Country	r_1	r_2	r_3	r_4	r_5	$\cdots$
s_1	Donot Team	India	0	0	0	0	0	$\cdots$
s_2	Lazarus	North Korea	1	0	0	0	0	$\cdots$
s_3	APT32	Vietnam	0	0	0	0	0	$\cdots$
s_4	Kimsuky	North Korea	0	0	0	0	0	$\cdots$
s_5	Turla	Russia	1	1	1	1	1	$\cdots$
s_6	LOTUS BLOSSOM	China	1	1	1	1	1	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

4.4 Dataset Preprocessing

Data Cleaning

We begin our methodology by exploring and cleaning our dataset. To account for duplicate rules or rules with identical behavior, we remove identical columns. We also replace any potential NaN values in our dataset with 0, assuming that the specific rule did not fire on that sample. As part of this step, we additionally performed a correlation analysis, which revealed that a subset of features were highly correlated. However, removing these correlated features did not improve clustering performance. Since this removal offered no measurable benefit and would introduce an additional, somewhat arbitrary threshold (the correlation cut-off), we retained the full feature set, keeping every feature as a directly interpretable YARA rule. After this initial cleaning step, our feature matrix is reduced from 8,962 to 2,566 YARA rules.

Data Aggregation

Since our goal in this research is to evaluate whether YARA rules have the discriminative power to reveal shared tooling among same-country APTs, our clustering must be performed at the APT family level. To work with and cluster families, we calculate the activation frequency of each YARA rule for each APT family. Specifically, we divide the number of samples that triggered a specific YARA rule by the total

number of malware samples within that family. Thus, each feature represents the percentage of samples in a family that triggered a given rule. After this process, we end up with a dataset of 78 rows (APT families) and 2,566 columns (YARA rules).

Table 4.2: Excerpt of the aggregated, family-level matrix (78 families $\times$ 2,566 rules).

Family	Country	r_1	r_2	r_3	r_4	r_5	$\cdots$
APT-C-44	Algeria	0.88	0.88	0.88	0.12	0.12	$\cdots$
APT-Q-27	China	0.82	0.82	0.82	0.12	0.15	$\cdots$
APT1	China	0.01	0.01	0.07	0.93	0.94	$\cdots$
APT10	China	0.05	0.05	0.11	0.71	0.35	$\cdots$
APT15	China	0.00	0.00	0.22	0.99	0.99	$\cdots$
APT23	China	0.01	0.02	0.04	0.46	0.42	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$

Figure 4.1 presents the number of APT families per country. It is evident that China dominates our dataset, with more than twice as many Russian families. It is worth mentioning that 6 countries consist of only 1 family. We also discuss the effects of the class imbalance in the Discussion.

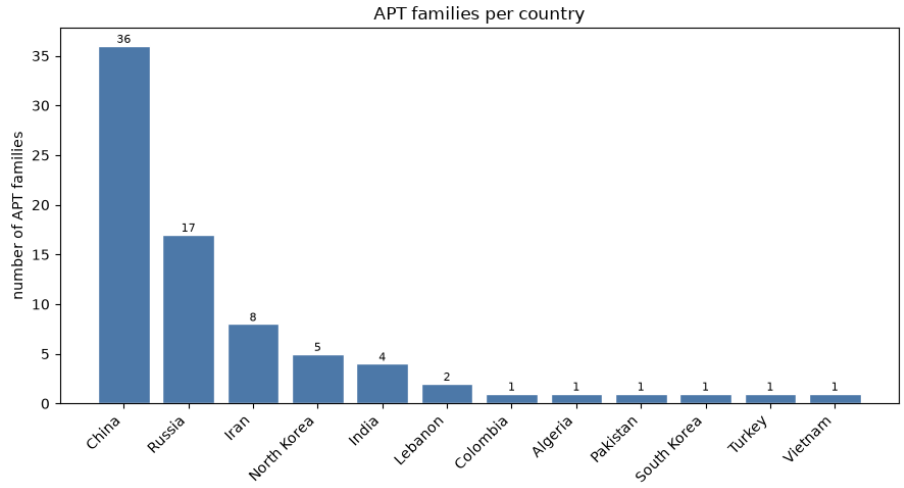


Figure 4.1: Number of APT families per country.

Feature Selection

The final step of data preprocessing applies a frequency-based feature selection that retains only rules lying within an intermediate document-frequency band, following the principle that terms which are too rare or too frequent carry little discriminative power. Here each APT family is treated as a document and each YARA rule as a term.

As the lower bound, we discard YARA rules that are present in only one family. This is done for two reasons. First, rules that fire in a single family provide no useful signal for clustering. Second, this step acts as an additional filter to ensure that any ephemeral, family- or infrastructure-specific YARA rules that slipped through the rule-selection phase are definitively excluded from our clustering model.

As the upper bound, we apply a maximum document-frequency threshold of $\text{max\_df} = 0.85$, discarding rules that fire in more than 85% of the families (66 of 78) — the corpus-specific stop words of our feature space: rules so ubiquitous that they detect common tooling or generic binary traits rather than distinguishing characteristics. Removing them reduces the dimensionality of the feature space and prevents these non-discriminative rules from dominating the distance computations, so that the clustering relies only

on informative rules. The same threshold is applied to all three representations. We further examine the effect of this parameter in Figure 4.2, where the Adjusted Rand Index and Adjusted Mutual Information of the baseline clustering (cosine distance, average linkage) are plotted against the upper family-frequency bound, providing insight into how the recovered structure responds to the choice of threshold. Finally, we aimed to remove any zero-variance features, since they provide no discriminative power; however, none were present in our dataset. After these steps, our final dataset consists of 1,364 YARA rules as features.

Sparsity of the Feature Matrix

The aggregated, family-level matrix on which the clustering is performed is highly sparse. It comprises 78 families and 1,364 rules (106,392 entries), of which only 19,295 (18.14%) are non-zero; equivalently, 81.86% of the matrix is zero. A typical family triggers a median of just 216 rules — about 16% of the rule set. This sparsity is a direct consequence of most YARA rules being highly specific, firing on only a handful of families. It also motivates our choice of distance metrics: the Cosine and Jaccard distances, which disregard jointly absent features (co-absences), are appropriate for such data, whereas metrics that count shared zeros as evidence of similarity would be dominated by the overwhelming number of zero entries.

Table 4.3: Sparsity of the aggregated family-level feature matrix.

Property	Value
Families (rows)	78
Rules (columns)	1,364
Total entries	106,392
Non-zero entries	19,295
Density	18.14%
Sparsity	**81.86%**

4.5 Hierarchical Clustering

We use Agglomerative Hierarchical Clustering: its bottom-up, nested-merging, summarized by a dendrogram approach suits the hierarchical structure of APT families, which are linked by shared tooling rather than flat categories. We evaluate three data representations, each fed into the model:

1. **Firing-rate matrix (baseline):** the unmodified aggregated matrix shown in Table 4.2, where each entry is the fraction of a family's samples that triggered a given rule. We cluster on these values directly, without any transformation, to assess the matrix's inherent clustering capability.

2. **Binary presence matrix:** each firing rate is reduced to a presence/absence indicator. In other words: 1 if the rule triggered on at least one sample of the family, 0 otherwise. This discards *how often* a rule fired within a family and keeps only *whether* it fired.

3. **TF-IDF–weighted matrix:** the firing rates re-weighted by inverse family-frequency, which increases the weight of rules appearing in few families (the more discriminative ones) and lowers the weight of ubiquitous, generic rules.

For each representation, we consider only distance metrics that are well suited to sparse data, ignore co-absences (matching zeros), and are appropriate to the representation's data type (binary or continuous). Two metrics satisfy these criteria: Jaccard and Cosine. Accordingly, we use both Jaccard and Cosine distance for the binary matrix, and Cosine distance alone for the continuous matrices (the baseline and TF-IDF representations). Each distance metric is then paired with one of three linkage criteria – single, complete, and average – and we report the results for every combination. To account for agreement arising by chance, we evaluate the clustering with the Adjusted Rand Index (ARI) and the Adjusted Mutual Information (AMI), both of which are chance-corrected. We further report Homogeneity and Completeness, which measure,

respectively, how well each cluster corresponds to a single country and how well the families of each country are kept together within a single cluster. We organize the analysis into three parts, moving from coarse, country-level structure to fine-grained relationships between individual APT families.

Analysis 1: Country-level clustering. We cluster the families into $k = 12$ groups (the number of countries) and evaluate the resulting partition against the ground-truth country labels using the metrics defined above.

Analysis 2: Country cohesion. Rather than committing to a single partition, we examine the dendrogram directly and ask whether families of the same country tend to merge earlier (at a lower height) than families of different countries. We quantify this with a Mantel test between the cophenetic distances of the baseline dendrogram and a binary same-/different-country indicator matrix, and further decompose it per country, testing whether the families of each individual country merge closer than a randomly selected group of the same size. Both tests obtain significance through 9,999 label permutations and are repeated across all feature-selection thresholds to confirm the result is not an artifact of a single configuration.

Analysis 3: Recovery of known family relationships. Finally, we ask whether the rule profiles recover genuine relationships between individual APT families, beyond shared nationality. Rather than curating relationships by hand, we derive them from the MITRE ATT&CK knowledge base [128]: for each family we retrieve its documented aliases (alternative names for the same actor) and the other groups that its description explicitly references. This yields two sets of pairs whose members are both present in our dataset: *(i) same-actor* pairs, where two of our families are aliases of a single MITRE group, and *(ii) related* pairs, where MITRE's description of one group links it to another. Because every such pair happens to be intra-country, we assess recovery relative to other same-country pairs: at each feature-selection threshold we rank each pair against all same-country pairs by cophenetic distance and record whether it falls among the closest 5% and 10%. We then report the fraction of thresholds at which a pair reaches each cut-off; a pair that does so far more often than the corresponding 5% and 10% chance rates is recovered as genuinely closer than an ordinary same-country pair — a family-level signal beyond shared nationality.

4.6 Significance of the Feature-Selection Threshold

Figure 4.2 shows how the baseline clustering scores respond to the upper family-frequency threshold max_fam. Both ARI and AMI stay low across the whole range — again pointing to a weak country signal — and AMI reaches its highest value at max_fam $= 66$, the setting we adopt for the baseline. In the figure, min_fam is fixed at 2 and the dashed line (right axis) shows the number of YARA rules kept at each threshold.

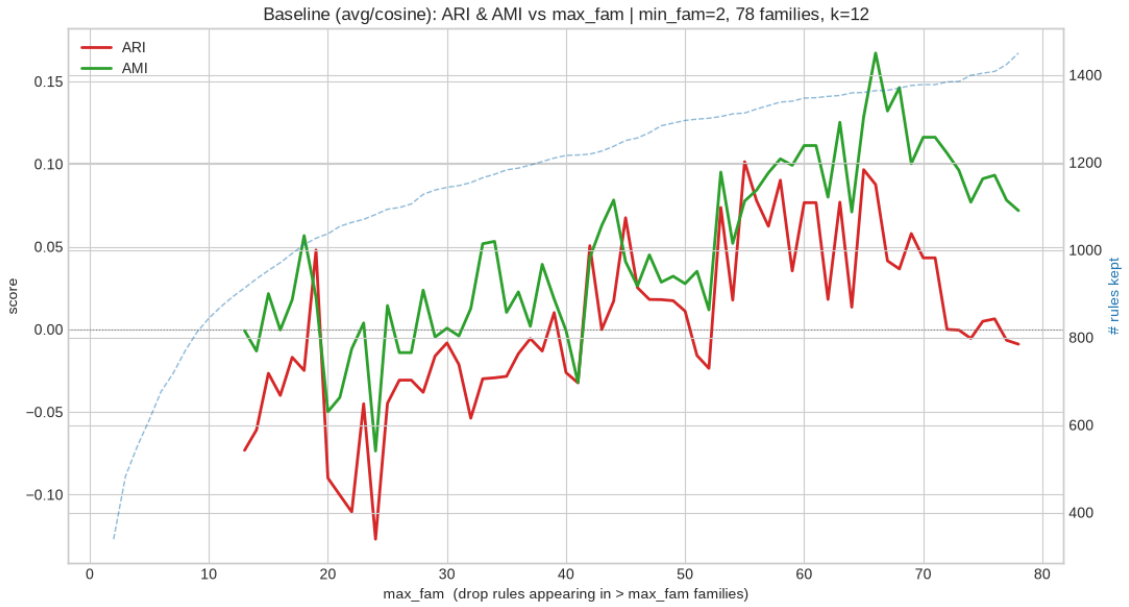


Figure 4.2: ARI and AMI of the baseline clustering as the threshold max_fam is varied.

Because we evaluate every possible threshold and then report the best one, this peak could be misleading: the highest score might simply be the luckiest threshold out of many rather than a genuine pattern (a multiple-comparisons problem). To rule this out, we apply a permutation test based on the *maximum statistic*, a standard correction for searching over multiple configurations. We first record the highest AMI obtained on the true data across all thresholds. We then permute the country labels, breaking any real geographic association, recompute the clustering across all thresholds, and store the maximum AMI achieved. Repeating this $9{,}999$ times builds a null distribution of the best score obtainable by chance *when the null is also allowed to pick its optimal threshold*. The empirical p-value is the fraction of permutations whose best AMI equals or exceeds the observed maximum:

$$p = \frac{\#\{\text{permutations with } \max_{\text{max\_fam}} \text{AMI} \geq \text{observed max AMI}\}}{\text{total number of permutations}}. \tag{4.2}$$

5 Results

5.1 Analysis 1

Baseline Representation

Table 5.1 presents the clustering results for the baseline firing-rate representation. Average linkage gives the best overall performance, achieving the highest ARI (tied with single linkage) and the highest AMI. Figure 5.2 shows the corresponding merge tree, and Figure 5.1a shows the country composition of the twelve clusters obtained with this configuration. In Figures 5.1a, 5.1b and 5.1c, each horizontal bar is a cluster, colored by the countries of its member families; a perfectly country-aligned clustering would therefore yield single-color bars.

Table 5.1: Analysis 1 results for the baseline (firing-rate) representation with cosine distance, $k = 12$.

Linkage	ARI	AMI	Homogeneity	Completeness
single	0.088	0.107	0.218	0.462
average	0.088	0.167	0.352	0.362
complete	-0.006	0.057	0.334	0.264

Binary Representation

Table 5.2 presents the results for the binary representation, which yields the worst clustering performance of the three. No metric–linkage combination reaches an AMI above 0.1, and every ARI score is negative. The corresponding cluster composition is shown in Figure 5.1b.

Table 5.2: Analysis 1 results for the binary (presence/absence) representation, $k = 12$.

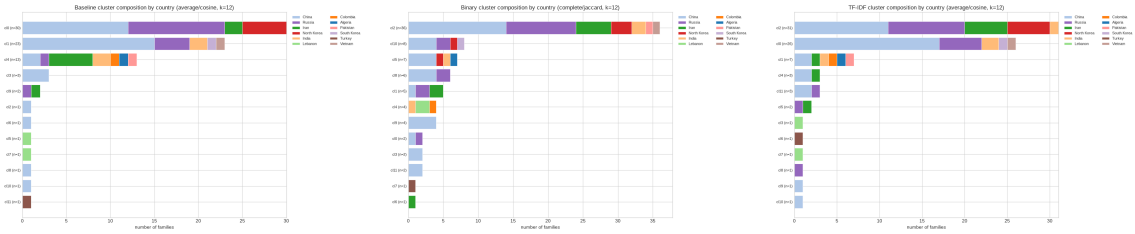
Linkage	Metric	ARI	AMI	Homogeneity	Completeness
complete	jaccard	-0.023	0.073	0.323	0.289
single	jaccard	-0.034	-0.006	0.153	0.323
single	cosine	-0.034	-0.006	0.153	0.323
complete	cosine	-0.046	0.033	0.282	0.260
average	cosine	-0.056	-0.033	0.152	0.273
average	jaccard	-0.078	-0.016	0.204	0.245

TF-IDF Representation

Table 5.3 presents the results for the TF-IDF representation. Average linkage performs best among the three, attaining the highest ARI (0.068) and AMI (0.125). The corresponding cluster composition is shown in Figure 5.1c.

Table 5.3: Analysis 1 results for the TF-IDF representation with cosine distance, $k = 12$.

Linkage	ARI	AMI	Homogeneity	Completeness
average	0.068	0.125	0.323	0.333
complete	0.040	0.102	0.365	0.296
single	0.029	0.035	0.176	0.373



(a) Baseline **(b) Binary** **(c) TF-IDF**

Figure 5.1: Country composition of the $k = 12$ clusters for each representation.

5.1.1 Significance of the Feature-Selection Threshold

Since the baseline representation achieved the best clustering scores in Analysis 1, we assess the significance of its result with a maximum-statistic permutation test that corrects for the search over max_fam. The highest AMI attained over all thresholds (0.167, at max_fam $= 66$) exceeds the 95th percentile of the best scores produced by 9,999 shuffled-label permutations (0.158), giving $p = 0.030$. The peak is therefore statistically significant even after the null model is allowed to select its own best threshold, confirming that it is not an artifact of the multiple thresholds tested. We accordingly regard the baseline partition as robust to the feature-selection process and adopt it as the basis for the subsequent dendrogram analysis.

5.2 Analysis 2

5.2.1 Mantel Test and Per-Country Cohesion

Because the baseline achieved the best clustering performance in Analysis 1, and the permutation test above confirmed that this result is robust to the feature-selection threshold, we focus the remaining dendrogram analysis exclusively on the baseline matrix.

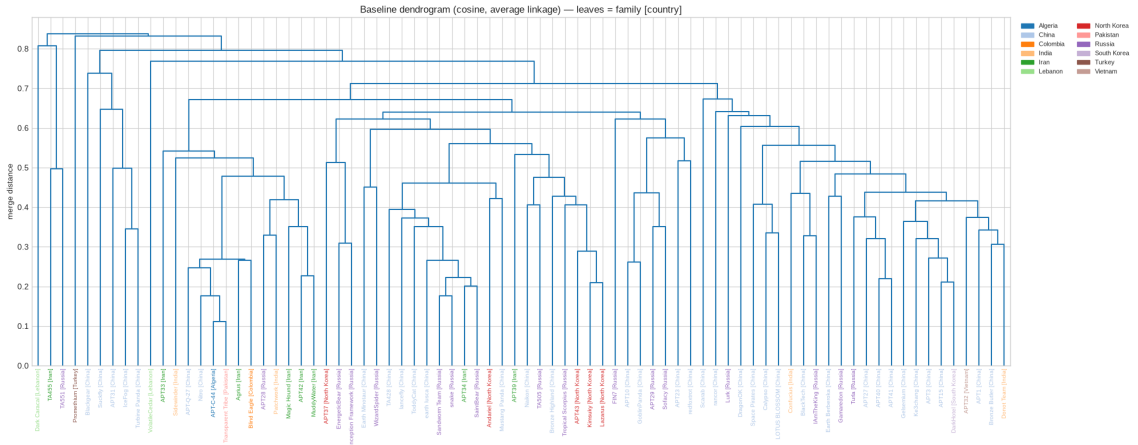


Figure 5.2: Dendrogram of the 78 APT families for the baseline representation, with leaves colored by country.

Table 5.4: Maximum-statistic permutation test for the baseline representation ($9{,}999$ label permutations).

Representation	Best AMI	Null max (95th pct)	p
Baseline	0.167	0.158	0.030

The Mantel test on the baseline dendrogram (Figure 5.2) returns a weak positive correlation at the selected threshold ($r_M = 0.072$) that marginally missed statistical significance ($p = 0.056$). Evaluated across all feature-selection thresholds; however, the coefficient remained positive 86% of the time, indicating a faint but directionally consistent tendency for APT families from the same country to merge earlier in the tree.

Table 5.5 reports the per-country cohesion results. There, the $\%p{<}0.05$ column gives the percentage of thresholds at which a country's families clustered significantly closer than a random group of the same size ($9{,}999$ permutations), and the %dir column the percentage of thresholds at which the country's average cophenetic distance was strictly below the global average. Lebanon ($n = 2$) is omitted, as its sample size is insufficient for the statistical test. North Korea is highly cohesive: its families merge significantly closer than random chance at 92% of the tested thresholds, and its average merge distance is strictly below the global average at 100% of them. Other countries do not exhibit this structural unity: Iran and Russia achieve significance at no more than 15% of the thresholds, while China—despite being the largest group in the dataset—reaches significance at only 3%.

Table 5.5: Per-country cohesion for the baseline representation, across all feature-selection thresholds.

Country	n	$\%p{<}0.05$	%dir
North Korea	5	92	100
Iran	8	15	82
Russia	17	11	83
China	36	3	39
India	4	0	71

5.3 Analysis 3

5.3.1 Recovery of Known Family Relationships

To test whether the rule profiles recover known relationships between families, we took ground-truth pairs from the MITRE ATT&CK knowledge base and kept only those where both families are in our dataset. This gives four *same-actor* pairs (families MITRE lists as aliases of one group) and nine *related* pairs (families that MITRE's group descriptions link to each other).

All of these pairs are from the same country, so a pair could look close simply because same-country families tend to cluster together (the weak signal from Analysis 2). To account for this, we compare each pair only against other same-country pairs, and we do so at every feature-selection threshold instead of averaging. At each max_fam value, we cluster the baseline matrix and, among all 811 same-country pairs, check whether the MITRE pair is among the closest 5% and the closest 10% by cophenetic distance. Because this analysis sweeps the upper threshold max_fam rather than fixing it, it draws on the full min_fam ≥ 2 pool of $1{,}450$ rules, rather than the fixed $[2, 66]$ band of $1{,}364$ rules used for the country-level clustering. Because we use the full set of same-country pairs, this ranking is exact and needs no permutations. For each pair we report the percentage of thresholds at which it reaches each cut-off, over the 66 usable thresholds (of 77; the lowest were dropped because some families have no active rule in the band, leaving the cosine distance undefined). A random same-country pair would reach the 5% and 10% cut-offs only about 5% and 10% of the time, which are therefore the chance rates.

Table 5.6: Recovery of MITRE-derived family pairs across the 66 feature-selection thresholds.

Pair	Tier	Top 5%	Top 10%
APT43 – Kimsuky	same-actor	100.0	100.0
APT15 – Ke3chang	same-actor	47.0	56.1
APT28 – Sofacy	same-actor	22.7	25.8
Turla – snake	same-actor	0.0	24.2
APT43 – Lazarus	related	60.6	63.6
Kimsuky – Lazarus	related	60.6	63.6
APT42 – Magic Hound	related	39.4	51.5
Andariel – Lazarus	related	33.3	66.7
APT28 – Sandworm Team	related	0.0	15.2
Sandworm Team – Sofacy	related	0.0	15.2
APT37 – Lazarus	related	0.0	13.6
APT41 – Earth Lusca	related	0.0	10.6
Confucius – Patchwork	related	0.0	4.5

Table 5.6 reports the results. Among the same-actor pairs, APT43–Kimsuky reaches the closest 5% at all 66 thresholds (100.0%), APT15–Ke3chang at 47.0%, APT28–Sofacy at 22.7%, and Turla–snake at 0.0% (24.2% at the 10% cut-off). Among the related pairs, APT43–Lazarus and Kimsuky–Lazarus reach the 5% cut-off at 60.6% of thresholds, APT42–Magic Hound at 39.4%, and Andariel–Lazarus at 33.3%; the remaining five related pairs reach it at 0.0%, with 10% cut-off values between 4.5% and 15.2%. The lowest value at the 10% cut-off is Confucius–Patchwork at 4.5%.

6 Discussion

6.1 Country Clustering

The baseline representation gave the best result among the three representations, followed closely by TF-IDF. Clustering on the binary representation performed poorly and clearly gave the worst results of the three, returning negative ARI scores on every combination. This indicates that the mere presence or absence of a YARA rule match in a family's samples is not enough to discriminate between countries: we also need to know how common each rule's firing is within each family. It is also worth mentioning that TF-IDF weighting did not lead to better results. This suggests that rules that fire in many families (which TF-IDF down-weights) are not simply noise but also carry meaningful signal for country discrimination. Overall, the baseline and TF-IDF results indicate a real but weak signal and limited discriminative power of YARA rules for separating nation-states. The ARI scores are lower than the AMI scores, possibly because ARI penalizes a country being split across several clusters. However, for large countries with many APT families whose tooling may vary a lot, such differences between families are expected, so splitting a country into more than one cluster is not necessarily a bad outcome. This is also why we chose AMI as the primary ranking score in our results. In the two best representations, Homogeneity and Completeness are low-to-moderate and fairly balanced.

Looking at the composition bar charts of the Baseline and TF-IDF representations, we observe many singleton clusters. Many of them are Chinese families, meaning China is quite spread out – which makes sense given the number of samples we gathered for this country. On inspection, these singletons do not reflect families with special tooling. They are singletons either because they fire very few rules (a sparse profile that sits far from everything) or because the rules they do fire are common ones shared with families from other countries, leaving them without a close same-country neighbor. Lebanon is a clear example: its two families end up as separate singletons not because of a distinct Lebanese toolset, but because they are dissimilar even to each other – each one resembles families from different foreign countries. The same generic-outlier effect explains Turkey's single family. All other smaller countries blend into Russia or China without any distinct pattern in the result. This can be explained by the class imbalance in our dataset, where countries with more samples tend to dominate the smaller, less-represented ones.

In the dendrogram, we can also see that some same-country families are among the first to merge. For example, APT43, Kimsuky, and Lazarus merge before blending with other countries. As we show later in this section, this could indicate that these families work closely together and are very tightly related – something that YARA rules can apparently make visible.

The permutation test for the baseline representation showed that these weakly positive results are not the product of a cherry-picked hyperparameter that spikes at a single point, but rather a statistically significant, repeatable positive signal.

Overall, despite the discriminative power YARA rules seem to have on some specific tasks, they perform poorly at general country separation. This could be due to several reasons. First, it could be due to the limited number of samples we have for some smaller countries. These few samples might not be representative of the country, making it much harder to separate them from the larger ones. At the same time, even though China and Russia are both well represented in the samples we gathered, they are not clearly distinguished from each other. This could indicate that these two countries base their malware on very common tooling, which YARA detects but that acts purely as noise. Lastly, as discussed in the Background section, APT samples are sophisticated binaries that make heavy use of obfuscation and packing. Their true tooling and attack methods may therefore remain concealed behind these techniques, in parts of the binary that YARA rules cannot scan at all.

On the other hand, these countries might indeed use specialized software that could carry discriminative power for country attribution, but the publicly available YARA rules mostly target more common tooling while ignoring these specialized, unique methods completely – something that was expected, since these rules are not written specifically for APT malware identification.

6.2 Other techniques used

Dimensionality Reduction

One could wonder whether the weak scores are a result of the ”curse of dimensionality” that our dataset appears to suffer from. To answer this, before clustering the baseline representation, we applied dimensionality reduction using Non-Negative Matrix Factorization (NMF) [106]. We chose this technique because it is one of the easiest to interpret [108], and since every individual YARA rule is useful for interpreting the results, interpretability was important to us. The results were no better than those we originally obtained, so we did not integrate it into our methodology.

Magnitude-Aware Metric

For our continuous representations, we used cosine distance, which ignores the magnitude of the feature vectors. To ensure this did not affect our results, we repeated the baseline clustering using the Bray–Curtis metric, which is magnitude-aware [129]. It did not improve the results – if anything, it scored slightly lower – so accounting for vector magnitude offered no benefit, and we kept cosine.

Overclustering

One might also wonder whether clustering into more than k = 12 groups could increase completeness and homogeneity, since large countries could split into several clusters based on differences among their families, while also giving smaller countries a chance to separate from the large buckets where they merge with larger countries. To answer this, we performed over-clustering, sweeping over k values from 12 to 40. The best AMI was achieved with the already-tested k = 12, and the second-best with k = 15. Since the results were not different, we did not include this in our methodology.

6.3 Mantel's Test

The Mantel test found a real but weak link between how early two families merge and whether they come from the same country. The overall signal is faint, and it comes almost entirely from one country: North Korea. Its families are far more similar to each other than a random group of families, at almost every threshold we tested, while no other country – including China, the largest – shows this. This aligns with the view of some analysts that many North Korean groups are part of or closely tied to the Lazarus umbrella [130, 131]. So the YARA rules were able to pick up North Korea’s cohesion. This could suggest that these families not only share tooling, but that their shared tooling is distinctive enough to set them apart from other countries, since they tend to cluster together rather than blending with families from other countries lower in the dendrogram.

6.4 Inter-Family Relations

The family-pair analysis supports what we found about North Korea’s cohesion. The pairs APT43–Kimsuky, APT43–Lazarus and Kimsuky–Lazarus are recovered most strongly, which fits our expectation of close links between these families. The pairs APT15–Ke3chang (China), APT42–Magic Hound (Iran), and Andariel–Lazarus (North Korea) are also recovered strongly and agree with MITRE’s information on how these groups are associated. Because each pair is compared only against other pairs from the same country, this closeness is not just an effect of shared nationality – it points to a real link between the families themselves.

Overall, YARA rules do carry signal for relationships between families. This is true for only a few pairs, but where the signal exists, it is strong.

6.4.1 Are the recovered relationships just leaked attribution?

A fair worry is that some YARA rules are named after a specific threat actor or campaign. If two families both trigger such a rule, they might look related only because the rule already ”knows” who they are, not because they truly share tooling.

Our initial rule-filtering step already tries to remove such rules, but it relies on a hand-written list of names, and a hand-written list can never be complete. Because this is the one result where a leaked name would be fatal — it would make the relationship look real when it is not — we did a second, stricter check here just for this analysis. This time we did not type the names ourselves: we built the list automatically from MITRE ATT&CK, taking every group and campaign name and alias. We deliberately left out malware names, so that rules detecting a shared piece of malware (which is exactly the signal we want) are kept, while only rules carrying an actor or campaign name are treated as suspicious.

We then removed every rule whose name matched one of these terms and repeated the whole analysis. Only nine of the roughly $1{,}450$ rules were affected — a small number that in itself shows the first filtering step had already done most of the work — including genuinely attribution-named ones such as the Lazarus campaign *Operation Blockbuster* and rules mentioning *Winnti* or *Confucius*. After removing them, the results barely changed: every pair kept almost the same score, and the strongest links (such as APT43–Kimsuky) were identical.

This tells us that the recovered relationships do not come from names baked into the rules. The families are pulled together instead by rules that detect shared malware families, for example *DTrack* between Andariel and Lazarus or *TidePool* between APT15 and Ke3chang. In other words, what YARA is picking up is genuine shared tooling, not the analyst’s prior knowledge of who each group is.

6.5 Effect of the feature-selection threshold

Figure 4.2 shows how the clustering scores change as we move the upper threshold max_fam. Both ARI and AMI stay low over the whole range, which again points to a weak country signal, and AMI reaches its highest value around max_fam $= 66$ — the setting we use for the baseline. However, this peak is exactly the kind of result that can be misleading: because we tried every possible threshold and then reported the best one, the highest score could simply be the luckiest threshold out of many, rather than a real pattern. This is a multiple-comparisons problem, and it is the reason we do not take the peak at face value. To check it, we run the max-statistic permutation test described below, which asks whether the best AMI is higher than what chance would produce *even when the same search over all thresholds is allowed*. Because the test found the peak to be significant, we can treat it as a genuine, if weak, structure and not just a cherry-picked threshold.

7 Conclusion and Future Work

This thesis examined whether the YARA-rule match profiles of APT malware families carry discriminative signal about their origin and their relationships, using only static artifacts. Returning to our three research questions, the answers are mixed but consistent.

For **RQ1 (country separation)**, YARA profiles do carry a real but *weak* country-of-origin signal: the best clustering reached an Adjusted Mutual Information of 0.167, which a maximum-statistic permutation test confirmed to be statistically significant ($p = 0.030$) rather than an artifact of the feature-selection threshold. This is well above chance, yet far from the level needed for reliable country attribution.

For **RQ2 (country cohesion)**, this weak signal is not evenly distributed. The Mantel test showed only a faint tendency for same-country families to merge earlier, and the per-country analysis localized almost all of it to *North Korea*, whose families cluster significantly closer than chance across the large majority of thresholds. No other country — including China, the largest group — exhibited comparable cohesion, which we attribute to the dominance of common, commodity tooling and to the effect of obfuscation on static signal.

For **RQ3 (family relationships)**, the results are clearest and strongest. Even though country separation is weak, YARA profiles recover specific, documented relationships between individual families — both aliases of the same actor and groups MITRE describes as related — markedly more often than an ordinary same-country pair would, and therefore beyond what shared nationality alone explains. Notably, this recovery is driven by rules that detect shared malware families rather than by attribution encoded in rule names, and it remains unchanged after removing every rule matching a known actor or campaign name.

Together, these findings show that publicly available YARA rules are a poor instrument for nation-state attribution on their own, but a genuine one for uncovering shared tooling and close relationships between individual APT families — a lightweight, fully static signal that can complement, rather than replace, existing attribution techniques.

Several directions could strengthen or extend this work:

- **A larger and more balanced dataset.** Six countries were represented by a single family, and China and Russia dominated the corpus. Collecting more samples for the under-represented countries would reduce the class imbalance that lets large groups absorb smaller ones, and would allow firmer conclusions about countries other than North Korea.
- **Unpacking and deobfuscation before scanning.** Because much APT tooling is hidden behind packing and encryption, scanning raw binaries misses signal that static rules cannot reach. Unpacking or unpacking-aware scanning could expose tooling that is currently invisible and may sharpen the country-level signal.
- **Purpose-built rules instead of generic public ones.** The public corpus is dominated by commodity signatures that act as noise. YARA rules written specifically to capture APT tooling, or automatically generated per family, might discriminate far better than the general-purpose rules used here.
- **Combining static and dynamic signal.** YARA profiles capture the Tools level of the Pyramid of Pain but not behavioral TTPs. Fusing static YARA features with dynamic behavioral indicators could combine the low cost of static analysis with the higher discriminative power of behavior.
- **Cross-country relationships.** Every recovered pair in this study was intra-country, because no cross-country relationships survived the match to our dataset. A dataset that includes families known to share tooling across nations would test whether the recovery generalizes beyond a single country's ecosystem.
- **Temporal analysis.** Tracking how families' rule profiles evolve over time could reveal when tooling is shared, forked, or retired, adding a temporal dimension to the relationships recovered here.

Εκτεταμένη Περίληψη

Οι Προηγμένες Επίμονες Απειλές (Advanced Persistent Threats - APT) αφορούν στοχευμένες, άρτια χρηματοδοτούμενες κυβερνοεπιθέσεις, οι οποίες συνήθως καθοδηγούνται από κρατικούς δρώντες. Η απόδοση (attribution) ενός κακόβουλου λογισμικού σε μια συγκεκριμένη οικογένεια ή χώρα προέλευσης αποτελεί μια εξαιρετικά χρήσιμη, αλλά παράλληλα δύσκολη διαδικασία, καθώς οι επιτιθέμενοι χρησιμοποιούν εξελιγμένες τεχνικές συγκάλυψης και παραπλάνησης (false-flag). Οι υπάρχουσες μέθοδοι βασίζονται συχνά στη δυναμική ανάλυση ή σε μοντέλα μηχανικής μάθησης. Ωστόσο, οι προσεγγίσεις αυτές είναι υπολογιστικά δαπανηρές και εξετάζουν το κάθε δείγμα μεμονωμένα, με αποτέλεσμα να μην αναδεικνύουν τα κοινά εργαλεία (shared tooling) που ενδέχεται να συνδέουν διαφορετικές οικογένειες κακόβουλου λογισμικού μεταξύ τους. Στην παρούσα πτυχιακή εργασία, διερευνούμε εάν οι κανόνες YARA –μια κατεξοχήν μέθοδος στατικής ανάλυσης– μπορούν να προσφέρουν αξιόπιστες ενδείξεις σχετικά με τη χώρα προέλευσης και τις διασυνδέσεις μεταξύ οικογενειών APT, βασιζόμενοι αποκλειστικά σε στατικά χαρακτηριστικά.

Για τον σκοπό αυτό, δημιουργήσαμε ένα σύνολο δεδομένων από 14.938 επαληθευμένα δείγματα Windows PE/DLL, τα οποία προέρχονται από 78 οικογένειες APT και εκπροσωπούν 12 διαφορετικές χώρες. Στη συνέχεια, τα σαρώσαμε χρησιμοποιώντας ένα εκτενές σύνολο δημόσια διαθέσιμων κανόνων YARA. Προκειμένου τα αποτελέσματα να αντικατοπτρίζουν αποκλειστικά την υποκείμενη τεχνική υποδομή και όχι έτοιμες πληροφορίες ταυτοποίησης, εξαιρέσαμε από το σύνολο αυτό κανόνες που κατονόμαζαν συγκεκριμένους επιτιθέμενους ή εκστρατείες, κανόνες που αφορούσαν αρχεία εκτός του τύπου PE, καθώς και εκείνους που βασίζονταν αποκλειστικά σε παροδικούς δείκτες παραβίασης (IoC). Έτσι, η κάθε οικογένεια αναπαρίσταται πλέον από τη συχνότητα με την οποία ενεργοποιούνται οι κανόνες στα δείγματά της. Για την ανάλυση, εφαρμόσαμε συσσωρευτική ιεραρχική συσταδοποίηση (agglomerative hierarchical clustering) εξετάζοντας τρεις διαφορετικές προσεγγίσεις (συχνότητα ενεργοποίησης, δυαδική παρουσία και TF-IDF), χρησιμοποιώντας μετρικές απόστασης κατάλληλες για αραιά δεδομένα (Cosine και Jaccard). Τα αποτελέσματα αξιολογήθηκαν με μετρικές προσαρμοσμένες ως προς την τυχαιότητα (ARI, AMI, ομοιογένεια και πληρότητα).

Σε πρώτη φάση, ομαδοποιήσαμε τις οικογένειες σε $k = 12$ συστάδες, όσες δηλαδή και οι χώρες προέλευσης. Η μέθοδος της συχνότητας ενεργοποίησης παρουσίασε την καλύτερη απόδοση ($\text{AMI} = 0{,}167$), γεγονός που επιβεβαιώθηκε στατιστικά μέσω ελέγχου μεταθέσεων μέγιστης στατιστικής (maximum statistic permutation test), ο οποίος διορθώνει για την αναζήτηση βέλτιστου κατωφλίου ($p = 0{,}030$). Παρά ταύτα, η ικανότητα ταυτοποίησης της χώρας προέλευσης παρέμεινε περιορισμένη. Στη δεύτερη φάση, μέσω του ελέγχου Mantel και αναλύσεων συνοχής ανά χώρα, εξετάσαμε κατά πόσο οι οικογένειες που προέρχονται από το ίδιο κράτος τείνουν να ομαδοποιούνται πιο στενά από ό,τι θα περιμέναμε καθαρά τυχαία. Η συνολική συσχέτιση αποδείχθηκε ασθενής ($r_M = 0{,}072$) και οφειλόταν σχεδόν εξ ολοκλήρου στη Βόρεια Κορέα, η οποία ήταν και η μόνη χώρα που παρουσίασε στατιστικά σημαντική συνοχή. Αντιθέτως, η Κίνα και η Ρωσία –οι δύο μεγαλύτερες ομάδες– δεν εμφάνισαν αντίστοιχη εσωτερική δομή.

Στην τρίτη φάση, ελέγξαμε κατά πόσο τα προφίλ YARA μπορούν να επιβεβαιώσουν ήδη γνωστές συσχετίσεις μεταξύ μεμονωμένων οικογενειών, εξετάζοντας ζεύγη δεδομένων από το πλαίσιο MITRE ATT&CK (ομάδες του ίδιου δρώντα ή σχετιζόμενες ομάδες). Συγκρίνοντας το κάθε ζεύγος με τα υπόλοιπα της ίδιας χώρας, διαπιστώσαμε ότι, ανεξαρτήτως του κατωφλίου επιλογής χαρακτηριστικών, πολλά γνωστά ζεύγη (όπως APT43–Kimsuky και Kimsuky–Lazarus) συσταδοποιούνταν πολύ πιο συχνά από ό,τι θα αναμενόταν τυχαία. Το αποτέλεσμα αυτό παρέμεινε ουσιαστικά αμετάβλητο ακόμα και μετά την αφαίρεση οποιουδήποτε κανόνα περιείχε όνομα δράστη ή εκστρατείας. Αυτό υποδηλώνει ότι η ισχυρή συσχέτιση προκύπτει όντως από τη χρήση κοινών εργαλείων (όπως π.χ. τα κακόβουλα λογισμικά DTrack και TidePool) και όχι από έτοιμες πληροφορίες απόδοσης ενσωματωμένες στους κανόνες.

Συμπερασματικά, η αποκλειστική χρήση δημόσια διαθέσιμων κανόνων YARA αποτελεί μια αδύναμη μέθοδο για την απόδοση επιθέσεων σε επίπεδο κράτους, κυρίως λόγω της ευρείας χρήσης κοινών, μη διακριτών εργαλείων και των εκτεταμένων τεχνικών συσκότισης (obfuscation). Ωστόσο, τα προφίλ αυτά αποτυπώνουν με ακρίβεια τη διαμοιραζόμενη εργαλειοθήκη και τις στενές σχέσεις που αναπτύσσονται μεταξύ μεμονωμένων οικογενειών κακόβουλου λογισμικού. Συνεπώς, προσφέρουν ένα ελαφρύ, αμιγώς στατικό στοιχείο που μπορεί να λειτουργήσει συμπληρωματικά –και όχι ως αντικατάσταση– στις ήδη υπάρχουσες τεχνικές απόδοσης.

Terminology Table

Foreign term	Greek term
Advanced Persistent Threat (APT)	Προηγμένη Επίμονη Απειλή
Attribution	Απόδοση
Malware	Κακόβουλο λογισμικό
Static analysis	Στατική ανάλυση
Dynamic analysis	Δυναμική ανάλυση
Clustering	Ομαδοποίηση
Hierarchical clustering	Ιεραρχική ομαδοποίηση
Dendrogram	Δενδρόγραμμα
Linkage criterion	Κριτήριο σύνδεσης
Distance metric	Μετρική απόστασης
Cophenetic distance	Συνοφαινετική απόσταση
Feature	Χαρακτηριστικό
Dimensionality reduction	Μείωση διαστάσεων
Sparsity	Σποραδικότητα
Permutation test	Έλεγχος μεταθέσεων
Ground truth	Πραγματικές ετικέτες
Homogeneity	Ομοιογένεια
Completeness	Πληρότητα

Abbreviations

Abbreviation	Expansion
AMI	Adjusted Mutual Information
APT	Advanced Persistent Threat
ARI	Adjusted Rand Index
C2	Command and Control
CTI	Cyber-Threat Intelligence
DLL	Dynamic-Link Library
IDF	Inverse Document Frequency
IoC	Indicator of Compromise
MI	Mutual Information
MITRE ATT&CK	Adversarial Tactics, Techniques, and Common Knowledge
NMF	Non-Negative Matrix Factorization
NMI	Normalized Mutual Information
PE	Portable Executable
RI	Rand Index
TF	Term Frequency
TF-IDF	Term Frequency–Inverse Document Frequency
TTP	Tactics, Techniques, and Procedures
YARA	Yet Another Recursive Acronym

References

[1] Anooja Joy et al. “The Evolution of APT Techniques Targeting the Power Sector: Trends, Challenges, and Defense Strategies”. In: *IEEE Access* 14 (2026), pp. 8426–8449. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2026.3651833. URL: https://ieeexplore.ieee.org/abstract/document/11338756 (visited on 08/17/2026).

[2] Florian Skopik and Timea Pahi. “Under false flag: using technical artifacts for cyber attack attribution”. en. In: *Cybersecurity* 3.1 (Mar. 2020), p. 8. ISSN: 2523-3246. DOI: 10.1186/s42400-020-00048-4. URL: https://doi.org/10.1186/s42400-020-00048-4 (visited on 08/17/2026).

[3] Ishai Rosenberg, Guillaume Sicard, and Eli (Omid) David. “DeepAPT: Nation-State APT Attribution Using End-to-End Deep Neural Networks”. en. In: *Artificial Neural Networks and Machine Learning – ICANN 2017*. Ed. by Alessandra Lintas et al. Cham: Springer International Publishing, 2017, pp. 91–99. ISBN: 978-3-319-68612-7. DOI: 10.1007/978-3-319-68612-7_11.

[4] Danny Kim et al. “A Hybrid Static Tool to Increase the Usability and Scalability of Dynamic Detection of Malware”. In: *2018 13th International Conference on Malicious and Unwanted Software (MALWARE)*. Oct. 2018, pp. 115–123. DOI: 10.1109/MALWARE.2018.8659373. URL: https://ieeexplore.ieee.org/document/8659373/ (visited on 08/17/2026).

[5] Chih-Hung Lin, Hsing-Kuo Pao, and Jian-Wei Liao. “Efficient dynamic malware analysis using virtual time control mechanics”. In: *Computers & Security* 73 (Mar. 2018), pp. 359–373. ISSN: 0167-4048. DOI: 10.1016/j.cose.2017.11.010. URL: https://www.sciencedirect.com/science/article/pii/S016740481730247X (visited on 08/17/2026).

[6] Jingwen Li, Jianyi Liu, and Ru Zhang. “Advanced Persistent Threat Group Correlation Analysis via Attack Behavior Patterns and Rough Sets”. en. In: *Electronics* 13.6 (Jan. 2024), p. 1106. ISSN: 2079-9292. DOI: 10.3390/electronics13061106. URL: https://www.mdpi.com/2079-9292/13/6/1106 (visited on 05/21/2026).

[7] Davidjbianco. *Enterprise Detection & Response: The Pyramid of Pain*. Mar. 2013. URL: http://detect-respond.blogspot.com/2013/03/the-pyramid-of-pain.html (visited on 08/16/2026).

[8] *YARA - The pattern matching swiss knife for malware researchers*. URL: https://virustotal.github.io/yara/ (visited on 07/30/2026).

[9] Nitin Naik et al. “Augmented YARA Rules Fused With Fuzzy Hashing in Ransomware Triaging”. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2019, pp. 625–632. DOI: 10.1109/SSCI44817.2019.9002773. URL: https://ieeexplore.ieee.org/abstract/document/9002773 (visited on 05/22/2026).

[10] Dimitar Dimitrov and Dimitar Nikolov. “LEVERAGING YARA AND SIGMA RULES TO DETECT CHINESE STATE-SPONSORED HACKING GROUPS OF THE ”TYPHOON” TYPE”. en. In: *ENVIRONMENT. TECHNOLOGY. RESOURCES. Proceedings of the International Scientific and Practical Conference* 2 (June 2025), pp. 107–113. ISSN: 2256-070X. DOI: 10.17770/etr2025vol2.8617. URL: https://archive-journals.rtu.lv/etr/article/view/5146 (visited on 05/19/2026).

[11] Christopher S Culling and Sally Vandeven. “Which YARA Rules Rule: Basic or Advanced?” en. In: ().

[12] Ishai Rosenberg, Guillaume Sicard, and Eli (Omid) David. “End-to-End Deep Neural Networks and Transfer Learning for Automatic Analysis of Nation-State Malware”. en. In: *Entropy* 20.5 (May 2018), p. 390. ISSN: 1099-4300. DOI: 10.3390/e20050390. URL: https://www.mdpi.com/1099-4300/20/5/390 (visited on 05/20/2026).

[13] Gil Shenderovitz and Nir Nissim. “Bon-APT: Detection, attribution, and explainability of APT malware using temporal segmentation of API calls”. In: *Computers & Security* 142 (July 2024), p. 103862. ISSN: 0167-4048. DOI: 10.1016/j.cose.2024.103862. URL: https://www.sciencedirect.com/science/article/pii/S0167404824001639 (visited on 05/20/2026).

[14] Michal Kida and Oluwafemi Olukoya. “Nation-State Threat Actor Attribution Using Fuzzy Hashing”. In: *IEEE Access* 11 (2023), pp. 1148–1165. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3233403. URL: https://ieeexplore.ieee.org/document/10004581 (visited on 05/17/2026).

[15] Nanda Rani et al. “Genesis of Cyber Threats: Towards Malware-based Advanced Persistent Threat (APT) Attribution”. English. In: IEEE Computer Society, Oct. 2024, pp. 399–408. ISBN: 979-8-3503-8674-5. DOI: 10.1109/TPS-ISA62245.2024.00052. URL: https://www.computer.org/csdl/proceedings-article/tps-isa/2024/867400a399/23ylJfQPs8o (visited on 05/17/2026).

[16] Na Xu et al. “An APT Malware Classification Method Based on Adaboost Feature Selection and LightGBM”. In: *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*. Oct. 2021, pp. 635–639. DOI: 10.1109/DSC53577.2021.00101. URL: https://ieeexplore.ieee.org/document/9750513 (visited on 05/18/2026).

[17] Shudong Li et al. “Attribution Classification Method of APT Malware in IoT Using Machine Learning Techniques”. en. In: *Security and Communication Networks* 2021.1 (2021). _eprint: https://onlinelibrary.wiley.com/doi p. 9396141. ISSN: 1939-0122. DOI: 10.1155/2021/9396141. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/9396141 (visited on 05/18/2026).

[18] Qinqin Wang et al. “APT Attribution for Malware Based on Time Series Shapelets”. In: *2022 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. ISSN: 2324-9013. Dec. 2022, pp. 769–777. DOI: 10.1109/TrustCom56396.2022.00108. URL: https://ieeexplore.ieee.org/document/10063628 (visited on 05/17/2026).

[19] Yangyang Mei et al. “A Hybrid Intelligent Approach to Attribute Advanced Persistent Threat Organization Using PSO-MSVM Algorithm”. In: *IEEE Transactions on Network and Service Management* 19.4 (Dec. 2022), pp. 4262–4272. ISSN: 1932-4537. DOI: 10.1109/TNSM.2022.3201928. URL: https://ieeexplore.ieee.org/document/9868098 (visited on 05/18/2026).

[20] Hamed Haddadpajouh et al. “MVFCC: A Multi-View Fuzzy Consensus Clustering Model for Malware Threat Attribution”. In: *IEEE Access* 8 (2020), pp. 139188–139198. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3012907. URL: https://ieeexplore.ieee.org/document/9152031 (visited on 05/18/2026).

[21] Lior Perry, Bracha Shapira, and Rami Puzis. “NO-DOUBT: Attack Attribution Based On Threat Intelligence Reports”. In: *2019 IEEE International Conference on Intelligence and Security Informatics (ISI)*. July 2019, pp. 80–85. DOI: 10.1109/ISI.2019.8823152. URL: https://ieeexplore.ieee.org/document/8823152 (visited on 05/18/2026).

[22] Naveen S, Rami Puzis, and Kumaresan Angappan. “Deep Learning for Threat Actor Attribution from Threat Reports”. In: *2020 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*. Sept. 2020, pp. 1–6. DOI: 10.1109/ICCCSP49186.2020.9315219. URL: https://ieeexplore.ieee.org/abstract/document/9315219 (visited on 05/18/2026).

[23] Umara Noor et al. “A machine learning-based FinTech cyber threat attribution framework using high-level indicators of compromise”. In: *Future Generation Computer Systems* 96 (July 2019), pp. 227–242. ISSN: 0167-739X. DOI: 10.1016/j.future.2019.02.013. URL: https://www.sciencedirect.com/science/article/pii/S0167739X18326141 (visited on 05/17/2026).

[24] Umara Noor et al. *A Machine Learning based Empirical Evaluation of Cyber Threat Actors High Level Attack Patterns over Low level Attack Patterns in Attributing Attacks*. en. July 2023. URL: https://arxiv.org/abs/2307.10252v1 (visited on 05/18/2026).

[25] Kelsie Edie, Cole Mckee, and Adam Duby. “Extending Threat Playbooks for Cyber Threat Intelligence: A Novel Approach for APT Attribution”. In: *2023 11th International Symposium on Digital Forensics and Security (ISDFS)*. May 2023, pp. 1–6. DOI: 10.1109/ISDFS58141.2023.10131867. URL: https://ieeexplore.ieee.org/document/10131867 (visited on 05/17/2026).

[26] Kyoungmin Kim et al. “Automatically Attributing Mobile Threat Actors by Vectorized ATT&CK Matrix and Paired Indicator”. en. In: *Sensors* 21.19 (Jan. 2021), p. 6522. ISSN: 1424-8220. DOI: 10.3390/s21196522. URL: https://www.mdpi.com/1424-8220/21/19/6522 (visited on 05/18/2026).

[27] Youngsup Shin et al. “ART: Automated Reclassification for Threat Actors based on ATT&CK Matrix Similarity”. In: *2021 World Automation Congress (WAC)*. ISSN: 2154-4824. Aug. 2021, pp. 15–20. DOI: 10.23919/WAC50355.2021.9559514. URL: https://ieeexplore.ieee.org/abstract/document/9559514 (visited on 05/18/2026).

[28] Vinay Sachidananda et al. “APTer: Towards the Investigation of APT Attribution”. In: *2023 IEEE Conference on Dependable and Secure Computing (DSC)*. Nov. 2023, pp. 1–10. DOI: 10.1109/DSC61021.2023.10354155. URL: https://ieeexplore.ieee.org/document/10354155 (visited on 05/18/2026).

[29] Ehtsham Irshad and Abdul Basit Siddiqui. “Context-aware cyber-threat attribution based on hybrid features”. In: *ICT Express* 10.3 (June 2024), pp. 553–569. ISSN: 2405-9595. DOI: 10.1016/j.icte.2024.04.005. URL: https://www.sciencedirect.com/science/article/pii/S2405959524000420 (visited on 05/17/2026).

[30] Emirhan Böge et al. “Unveiling Cyber Threat Actors: A Hybrid Deep Learning Approach for Behavior-Based Attribution”. In: *Digital Threats: Research and Practice* 6.1 (Feb. 2025), 2:1–2:20. DOI: 10.1145/3676284. URL: https://dl.acm.org/doi/10.1145/3676284 (visited on 05/17/2026).

[31] Nan Xiao et al. “APT-MMF: An advanced persistent threat actor attribution method based on multimodal and multilevel feature fusion”. In: *Computers & Security* 144 (Sept. 2024), p. 103960. ISSN: 0167-4048. DOI: 10.1016/j.cose.2024.103960. URL: https://www.sciencedirect.com/science/article/pii/S0167404824002657 (visited on 05/17/2026).

[32] Yixin Wu et al. “GroupTracer: Automatic Attacker TTP Profile Extraction and Group Cluster in Internet of Things”. en. In: *Security and Communication Networks* 2020.1 (2020). _eprint: https://onlinelibrary.wiley.com/do
p. 8842539. ISSN: 1939-0122. DOI: 10.1155/2020/8842539. URL: https://onlinelibrary.wiley.com/doi/abs/10.1155/2020/8842539 (visited on 05/18/2026).

[33] Yijia Xu et al. “HGHAN: Hacker group identification based on heterogeneous graph attention network”. In: *Information Sciences* 612 (Oct. 2022), pp. 848–863. ISSN: 0020-0255. DOI: 10.1016/j.ins.2022.08.097. URL: https://www.sciencedirect.com/science/article/pii/S0020025522010076 (visited on 05/18/2026).

[34] K. Subhash Chandra et al. “Cybersecurity Knowledge Graph: Advanced Persistent Threat Organization Attribution”. In: *2023 Third International Conference on Ubiquitous Computing and Intelligent Information Systems (ICUIS)*. Sept. 2023, pp. 350–356. DOI: 10.1109/ICUIS60567.2023.00064. URL: https://ieeexplore.ieee.org/abstract/document/10506026 (visited on 05/17/2026).

[35] Yitong Ren et al. “CSKG4APT: A Cybersecurity Knowledge Graph for Advanced Persistent Threat Organization Attribution”. In: *IEEE Transactions on Knowledge and Data Engineering* 35.6 (June 2023), pp. 5695–5709. ISSN: 1558-2191. DOI: 10.1109/TKDE.2022.3175719. URL: https://ieeexplore.ieee.org/abstract/document/9834133 (visited on 05/17/2026).

[36] Erisa Karafili, Linna Wang, and Emil C. Lupu. “An Argumentation-Based Reasoner to Assist Digital Investigation and Attribution of Cyber-Attacks”. In: *Forensic Science International: Digital Investigation* 32 (Apr. 2020), p. 300925. ISSN: 2666-2817. DOI: 10.1016/j.fsidi.2020.300925. URL: https://www.sciencedirect.com/science/article/pii/S2666281720300202 (visited on 05/21/2026).

[37] Yangyang Mei et al. “A Novel Network Forensic Framework for Advanced Persistent Threat Attack Attribution Through Deep Learning”. In: *IEEE Transactions on Intelligent Transportation Systems* 25.9 (Sept. 2024), pp. 12131–12140. ISSN: 1558-0016. DOI: 10.1109/TITS.2024.3360260. URL: https://ieeexplore.ieee.org/document/10440159 (visited on 05/17/2026).

[38] Nanda Rani et al. “Decoding shadows: Towards Tactics, Techniques, and Procedures (TTP)-based Advanced Persistent Threat (APT) attribution”. In: *Information Security Journal: A Global Perspective* 35.3 (May 2026), pp. 381–408. ISSN: 1939-3555. DOI: 10.1080/19393555.2025.2543450. URL: https://doi.org/10.1080/19393555.2025.2543450 (visited on 08/16/2026).

[39] Kyla Guru, Robert J. Moss, and Mykel J. Kochenderfer. “On Technique Identification and Threat-Actor Attribution using LLMs and Embedding Models”. In: *2025 International Conference on Cybersecurity and AI-Based Systems (Cyber-AI)*. Sept. 2025, pp. 332–339. DOI: 10.1109/CyberAI66431.2025.11233715. URL: https://ieeexplore.ieee.org/abstract/document/11233715 (visited on 05/22/2026).

[40] Saed Alrabaee et al. “On Leveraging Coding Habits for Effective Binary Authorship Attribution”. en. In: *Computer Security*. Ed. by Javier Lopez, Jianying Zhou, and Miguel Soriano. Cham: Springer International Publishing, 2018, pp. 26–47. ISBN: 978-3-319-99073-6. DOI: 10.1007/978-3-319-99073-6_2.

[41] Jason Gray, Daniele Sgandurra, and Lorenzo Cavallaro. *Identifying Authorship Style in Malicious Binaries: Techniques, Challenges & Datasets*. en. arXiv:2101.06124 [cs.CR]. Jan. 2021. DOI: 10.48550/arXiv.2101.06124. URL: http://arxiv.org/abs/2101.06124 (visited on 05/19/2026).

[42] Brian Bartholomew and Juan Andres Guerrero-Saade. “WAVE YOUR FALSE FLAGS! DECEPTION TACTICS MUDDYING ATTRIBUTION IN TARGETED ATTACKS”. en. In: ().

[43] Zheng-Shao Chen et al. “Clustering APT Groups Through Cyber Threat Intelligence by Weighted Similarity Measurement”. In: *IEEE Access* 12 (2024), pp. 141851–141865. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3469552. URL: https://ieeexplore.ieee.org/abstract/document/10697172 (visited on 05/20/2026).

[44] Jingwen Li, Jianyi Liu, and Ru Zhang. “Advanced Persistent Threat Group Correlation Analysis via Attack Behavior Patterns and Rough Sets”. en. In: *Electronics* 13.6 (Jan. 2024), p. 1106. ISSN: 2079-9292. DOI: 10.3390/electronics13061106. URL: https://www.mdpi.com/2079-9292/13/6/1106 (visited on 05/21/2026).

[45] Wenhao Wang et al. “Clustering Using a Similarity Measure Approach Based on Semantic Analysis of Adversary Behaviors”. In: *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*. July 2020, pp. 1–7. DOI: 10.1109/DSC50466.2020.9194468. URL: https://ieeexplore.ieee.org/document/9194468/ (visited on 05/21/2026).

[46] Jian Xu et al. “NetworkTrace: Probabilistic Relevant Pattern Recognition Approach to Attribution Trace Analysis”. In: *2017 IEEE Trustcom/BigDataSE/ICESS*. ISSN: 2324-9013. Aug. 2017, pp. 691–698. DOI: 10.1109/Trustcom/BigDataSE/ICESS.2017.301. URL: https://ieeexplore.ieee.org/abstract/document/8029504 (visited on 05/21/2026).

[47] Hesong Wang et al. “SCFAO: A Self-Contained Framework for APT Organizational Consistency Assessment”. In: *2025 IEEE 10th International Conference on Data Science in Cyberspace (DSC)*. Aug. 2025, pp. 589–596. DOI: 10.1109/DSC67331.2025.00083. URL: https://ieeexplore.ieee.org/document/11360396/ (visited on 05/22/2026).

[48] Zhaoyun Ding et al. “A Method for Discovering Hidden Patterns of Cybersecurity Knowledge Based on Hierarchical Clustering”. In: *2021 IEEE Sixth International Conference on Data Science in Cyberspace (DSC)*. Oct. 2021, pp. 334–338. DOI: 10.1109/DSC53577.2021.00053. URL: https://ieeexplore.ieee.org/abstract/document/9750532 (visited on 05/21/2026).

[49] Rawan Al-Shaer, Jonathan M. Spring, and Eliana Christou. “Learning the Associations of MITRE ATT & CK Adversarial Techniques”. In: *2020 IEEE Conference on Communications and Network Security (CNS)*. June 2020, pp. 1–9. DOI: 10.1109/CNS48642.2020.9162207. URL: https://ieeexplore.ieee.org/abstract/document/9162207 (visited on 05/21/2026).

[50] Antonino Vitale et al. “Family Ties: A Close Look at the Influence of Static Features on the Precision of Malware Family Clustering”. In: *2025 APWG Symposium on Electronic Crime Research (eCrime)*. ISSN: 2159-1245. Nov. 2025, pp. 1–13. DOI: 10.1109/eCrime66972.2025.11327864. URL: https://ieeexplore.ieee.org/abstract/document/11327864 (visited on 05/24/2026).

[51] Fatima Al Shamsi, Wei Lee Woon, and Zeyar Aung. “Discovering Similarities in Malware Behaviors by Clustering of API Call Sequences”. en. In: *Neural Information Processing*. Ed. by Long Cheng, Andrew Chi Sing Leung, and Seiichi Ozawa. Cham: Springer International Publishing, 2018, pp. 122–133. ISBN: 978-3-030-04212-7. DOI: 10.1007/978-3-030-04212-7_11.

[52] Mohammed Rauf Ali Khan and Louai Al-Awami. “No ML, Just Hashes: Code Similarity Distance for Detecting IoT Malware”. In: *2026 IEEE International Conference on Consumer Electronics (ICCE)*. ISSN: 2158-4001. Feb. 2026, pp. 1–6. DOI: 10.1109/ICCE67443.2026.11449610. URL: https://ieeexplore.ieee.org/abstract/document/11449610 (visited on 05/24/2026).

[53] Nitin Naik et al. “Cyberthreat hunting - Part 1: triaging ransomware using fuzzy hashing, import hashing and YARA rules: 2019 IEEE International Conference”. In: *2019 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. Fuzzy Systems (FUZZ-IEEE) 2019 IEEE International Conference (Oct. 2019). ISSN: 978-1-5386-1729-8. DOI: 10.1109/FUZZ-IEEE.2019.8858803.

[54] Nitin Naik et al. “Fuzzy Hashing Aided Enhanced YARA Rules for Malware Triaging”. In: *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. Dec. 2020, pp. 1138–1145. DOI: 10.1109/SSCI47803.2020.9308189. URL: https://ieeexplore.ieee.org/abstract/document/9308189 (visited on 05/22/2026).

[55] Shreyansh Swami et al. “Adaptive Detection of Polymorphic Malware: Leveraging Mutation Engines and YARA Rules for Enhanced Security”. en. In: ().

[56] Andrea Atzeni et al. “Countering Android Malware: A Scalable Semi-Supervised Approach for Family-Signature Generation”. In: *IEEE Access* 6 (2018), pp. 59540–59556. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2874502. URL: https://ieeexplore.ieee.org/abstract/document/8485352 (visited on 05/23/2026).

[57] Franklin Tchakounté et al. “LimonDroid: a system coupling three signature-based schemes for profiling Android malware”. en. In: *Iran Journal of Computer Science* 4.2 (June 2021), pp. 95–114. ISSN: 2520-8446. DOI: 10.1007/s42044-020-00068-w. URL: https://doi.org/10.1007/s42044-020-00068-w (visited on 05/23/2026).

[58] Arnaldo Sgueglia et al. “Poster: A Multi-step Approach for Classification of Malware Samples”. In: *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. CCS ’24. New York, NY, USA: Association for Computing Machinery, Dec. 2024, pp. 5009–5011. ISBN: 979-8-4007-0636-3. DOI: 10.1145/3658644.3691370. URL: https://dl.acm.org/doi/10.1145/3658644.3691370 (visited on 05/23/2026).

[59] Gerardo Canfora et al. “About the Robustness and Looseness of Yara Rules”. en. In: *Testing Software and Systems*. Ed. by Valentina Casola, Alessandra De Benedictis, and Massimiliano Rak. Cham: Springer International Publishing, 2020, pp. 104–120. ISBN: 978-3-030-64881-7. DOI: 10.1007/978-3-030-64881-7_7.

[60] Ping Chen, Lieven Desmet, and Christophe Huygens. “A Study on Advanced Persistent Threats”. en. In: *Communications and Multimedia Security*. Ed. by Bart De Decker and André Zúquete. Berlin, Heidelberg: Springer, 2014, pp. 63–72. ISBN: 978-3-662-44885-4. DOI: 10.1007/978-3-662-44885-4_5.

[61] Amit Sharma et al. “Advanced Persistent Threats (APT): evolution, anatomy, attribution and countermeasures”. en. In: *Journal of Ambient Intelligence and Humanized Computing* 14.7 (July 2023), pp. 9355–9381. ISSN: 1868-5145. DOI: 10.1007/s12652-023-04603-y. URL: https://doi.org/10.1007/s12652-023-04603-y (visited on 05/17/2026).

[62] Brian Bartholomew and Juan Andres Guerrero-Saade. “Wave your false flags! deception tactics muddying attribution in targeted attacks”. In: *Virus Bulletin Conference*. Vol. 9. 2016.

[63] Florian Skopik and Timea Pahi. “Under false flag: using technical artifacts for cyber attack attribution”. In: *Cybersecurity* 3.1 (2020), p. 8.

[64] Atif Ahmad et al. “Strategically-motivated advanced persistent threat: Definition, process, tactics and a disinformation model of counterattack”. en. In: *Computers & Security* 86 (Sept. 2019), pp. 402–418. ISSN: 01674048. DOI: 10.1016/j.cose.2019.07.001. URL: https://linkinghub.elsevier.com/retrieve/pii/S0167404818310988 (visited on 07/24/2026).

[65] Blake E Strom et al. “MITRE ATT&CKÒ: Design and Philosophy”. en. In: ().

[66] Nanda Rani, Bikash Saha, and Sandeep Kumar Shukla. “A comprehensive survey of automated Advanced Persistent Threat attribution: Taxonomy, methods, challenges and open research problems”. In: *Journal of Information Security and Applications* 92 (July 2025), p. 104076. ISSN: 2214-2126. DOI: 10.1016/j.jisa.2025.104076. URL: https://www.sciencedirect.com/science/article/pii/S2214212625001139 (visited on 05/17/2026).

[67] Murtaza Siddiqi, Abdul Aziz, and Kanwal Oad. “Advanced Persistent Threats Defense Techniques: A Review”. In: 1 (May 2016), pp. 53–65.

[68] *Petya Destructive Malware Variant Spreading via Stolen Credentials and EternalBlue Exploit | Mandiant*. en-US. URL: https://cloud.google.com/blog/topics/threat-intelligence/petya-ransomware-spreading-via-eternalblue-exploit (visited on 08/16/2026).

[69] *SolarWinds Supply Chain Attack Uses SUNBURST Backdoor*. en-US. URL: https://cloud.google.com/blog/topics/threat-intelligence/evasive-attacker-leverages-solarwinds-supply-chain-compromises-with-sunburst-backdoor (visited on 08/16/2026).

[70] Constantinos Patsakis, David Arroyo, and Fran Casino. “The malware as a service ecosystem”. In: *Malware: Handbook of prevention and detection*. Springer, 2024, pp. 371–394.

[71] Yangyang Mei et al. “A Review of Attribution Technical for APT Attacks”. In: *2022 7th IEEE International Conference on Data Science in Cyberspace (DSC)*. July 2022, pp. 512–518. DOI: 10.1109/DSC55868.2022.00077. URL: https://ieeexplore.ieee.org/document/9900213 (visited on 05/17/2026).

[72] *Cyber Kill Chain®*. en. URL: https://www.lockheedmartin.com/en-us/capabilities/cyber/cyber-kill-chain.html (visited on 09/01/2026).

[73] Eric M Hutchins, Michael J Cloppert, and Rohan M Amin. “Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains”. en. In: ().

[74] Paul Pols and Francisco Dominguez. “RAISING RESILIENCE AGAINST ADVANCED CYBER ATTACKS”. en. In: ().

[75] *Using An Expanded Cyber Kill Chain Model to Increase Attack Resiliency : Black Hat : Free Download, Borrow, and Streaming : Internet Archive*. URL: https://archive.org/details/youtube-1Dz12M7u-S8 (visited on 08/16/2026).

[76] Sungyoung Cho et al. “Cyber Kill Chain based Threat Taxonomy and its Application on Cyber Common Operational Picture”. In: *2018 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. June 2018, pp. 1–8. DOI: 10.1109/CyberSA.2018.8551383. URL: https://ieeexplore.ieee.org/document/8551383 (visited on 08/16/2026).

[77] *FAQ | MITRE ATT&CK®*. URL: https://attack.mitre.org/resources/faq/ (visited on 07/27/2026).

[78] *MITRE ATT&CK®*. URL: https://attack.mitre.org/ (visited on 07/25/2026).

[79] Anitta Patience Namanya et al. “The World of Malware: An Overview”. In: *2018 IEEE 6th International Conference on Future Internet of Things and Cloud (FiCloud)*. Aug. 2018, pp. 420–427. DOI: 10.1109/FiCloud.2018.00067. URL: https://ieeexplore.ieee.org/abstract/document/8458044 (visited on 07/27/2026).

[80] Sushil Jajodia, Pierangela Samarati, and Moti Yung, eds. *Encyclopedia of Cryptography, Security and Privacy*. en. Cham: Springer Nature Switzerland, 2025. ISBN: 978-3-030-71520-5 978-3-030-71522-9. DOI: 10.1007/978-3-030-71522-9. URL: https://link.springer.com/10.1007/978-3-030-71522-9 (visited on 07/30/2026).

[81] *Overview of the Mach-O Executable Format*. URL: https://developer.apple.com/library/archive/documentation/Performance/Conceptual/CodeFootprint/Articles/MachOOverview.html (visited on 08/16/2026).

[82] *Linux Foundation Referenced Specifications*. URL: https://refspecs.linuxfoundation.org/ (visited on 08/16/2026).

[83] GrantMeStrength. *PE Format - Win32 apps*. en-us. URL: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format (visited on 08/16/2026).

[84] *AV-ATLAS*. en. URL: https://portal.av-atlas.org/malware/statistics (visited on 08/16/2026).

[85] Michael Sikorski. *Practical malware analysis: the hands-on guide to dissecting malicious software*. en. San Francisco: No Starch Press, 2012. ISBN: 978-1-59327-290-6 978-1-59327-430-6.

[86] Sebastian Berrios et al. "Systematic Review: Malware Detection and Classification in Cybersecurity". en. In: *Applied Sciences* 15.14 (Jan. 2025), p. 7747. ISSN: 2076-3417. DOI: 10.3390/app15147747. URL: https://www.mdpi.com/2076-3417/15/14/7747 (visited on 07/29/2026).

[87] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. "Barebox: efficient malware analysis on bare-metal". In: *Proceedings of the 27th annual computer security applications conference*. 2011, pp. 403–412.

[88] Dhilung Kirat, Giovanni Vigna, and Christopher Kruegel. "{BareCloud}: Bare-metal analysis-based evasive malware detection". In: *23rd USENIX Security Symposium (USENIX Security 14)*. 2014, pp. 287–301.

[89] Sainadh Jamalpur et al. "Dynamic malware analysis using cuckoo sandbox". In: *2018 Second international conference on inventive communication and computational technologies (ICICCT)*. IEEE. 2018, pp. 1056–1060.

[90] Songsong Liu et al. "Enhancing malware analysis sandboxes with emulated user behavior". In: *Computers & Security* 115 (2022), p. 102613.

[91] Vasilis Vouvoutsis, Fran Casino, and Constantinos Patsakis. "On the effectiveness of binary emulation in malware classification". In: *J. Inf. Secur. Appl.* 68 (2022), p. 103258. DOI: 10.1016/J.JISA.2022.103258. URL: https://doi.org/10.1016/j.jisa.2022.103258.

[92] Vasilis Vouvoutsis, Fran Casino, and Constantinos Patsakis. "Beyond the sandbox: Leveraging symbolic execution for evasive malware classification". In: *Comput. Secur.* 149 (2025), p. 104193. DOI: 10.1016/J.COSE.2024.104193. URL: https://doi.org/10.1016/j.cose.2024.104193.

[93] Saranya Chandran et al. "From Static to AI-Driven Detection: A Comprehensive Review of Obfuscated Malware Techniques". In: *IEEE Access* 13 (2025), pp. 74335–74358. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3550781. URL: https://ieeexplore.ieee.org/abstract/document/10924165 (visited on 07/29/2026).

[94] Jayanthi Ramamoorthy, Narasimha K Shashidhar, and Cihan Varol. "Automated Static Analysis of Linux ELF Malware: Framework and Application". In: *2025 13th International Symposium on Digital Forensics and Security (ISDFS)*. ISSN: 2768-1831. Apr. 2025, pp. 1–5. DOI: 10.1109/ISDFS65363.2025.11012103. URL: https://ieeexplore.ieee.org/abstract/document/11012103 (visited on 07/30/2026).

[95] *Welcome to YARA's documentation! — yara 4.4.0 documentation*. URL: https://yara.readthedocs.io/en/stable/index.html (visited on 07/30/2026).

[96] Omer Aslan and Refik Samet. "A Comprehensive Review on Malware Detection Approaches". eng. In: *IEEE ACCESS* 8 (2020). ISSN: 2169-3536. DOI: 10.1109/access.2019.2963724. URL: https://avesis.ankara.edu.tr/yayin/e9c86c43-0dee-4c78-97dd-deb45cd72d59/a-comprehensive-review-on-malware-detection-approaches (visited on 07/27/2026).

[97] *Virus Bulletin :: Rule-driven malware identification and classification*. URL: https://www.virusbulletin.com/virusbulletin/2008/01/rule-driven-malware-identification-and-classification (visited on 07/30/2026).

[98] *Writing YARA rules — yara 4.5.0 documentation*. URL: https://yara.readthedocs.io/en/latest/writingrules.html#using-modules (visited on 07/31/2026).

[99] *An Introduction to Statistical Learning*. en-US. URL: https://www.statlearning.com (visited on 08/01/2026).

[100] *Data Mining: Concepts and Techniques | Guide books | ACM Digital Library*. en. URL: https://dl.acm.org/doi/book/10.5555/1972541 (visited on 08/01/2026).

[101] *Introduction to HPC with MPI for Data Science | Springer Nature Link*. URL: https://link.springer.com/book/10.1007/978-3-319-21903-5 (visited on 08/01/2026).

[102] *A rapid review of clustering algorithms - ScienceDirect*. URL: https://www.sciencedirect.com/science/article/pii/S2590005626002274 (visited on 08/16/2026).

[103] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. USA: Prentice-Hall, Inc., 1988. ISBN: 978-0-13-022278-7.

[104] Michael R. Anderberg. *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*. en. Google-Books-ID: 7YTiBQAAQBAJ. Academic Press, May 2014. ISBN: 978-1-4831-9139-3.

[105] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to data mining*. en. New internat. edition. Always learning. Harlow: Pearson, 2014. ISBN: 978-1-292-02615-2.

[106] Daniel D. Lee and H. Sebastian Seung. "Learning the parts of objects by non-negative matrix factorization". en. In: *Nature* 401.6755 (Oct. 1999), pp. 788–791. ISSN: 1476-4687. DOI: 10.1038/44565. URL: https://www.nature.com/articles/44565 (visited on 08/08/2026).

[107] Yu-Xiong Wang and Yu-Jin Zhang. "Nonnegative Matrix Factorization: A Comprehensive Review". In: *IEEE Transactions on Knowledge and Data Engineering* 25.6 (June 2013), pp. 1336–1353. ISSN: 1558-2191. DOI: 10.1109/TKDE.2012.51. URL: https://ieeexplore.ieee.org/abstract/document/6165290 (visited on 08/08/2026).

[108] Rachana Mehta, Shakti Mishra, and Snehanshu Saha. "Non-Negative Matrix Factorization in Recommender Models: Concept, Survey, and Future Direction". In: *IEEE Access* 13 (2025), pp. 192492–192517. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2025.3630750. URL: https://ieeexplore.ieee.org/abstract/document/11235932 (visited on 08/07/2026).

[109] *A comparison of extrinsic clustering evaluation metrics based on formal constraints | Discover Computing | Springer Nature Link*. URL: https://link.springer.com/article/10.1007/s10791-008-9066-8 (visited on 08/16/2026).

[110] *Objective Criteria for the Evaluation of Clustering Methods: Journal of the American Statistical Association: Vol 66, No 336*. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1971.10482356 (visited on 08/16/2026).

[111] Nguyen Xuan Vinh, Julien Epps, and James Bailey. "Information theoretic measures for clusterings comparison: is a correction for chance necessary?" In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ICML '09. New York, NY, USA: Association for Computing Machinery, June 2009, pp. 1073–1080. ISBN: 978-1-60558-516-1. DOI: 10.1145/1553374.1553511. URL: https://dl.acm.org/doi/10.1145/1553374.1553511 (visited on 08/08/2026).

[112] Lawrence Hubert and Phipps Arabie. "Comparing partitions". en. In: *Journal of Classification* 2.1 (Dec. 1985), pp. 193–218. ISSN: 1432-1343. DOI: 10.1007/BF01908075. URL: https://doi.org/10.1007/BF01908075 (visited on 08/16/2026).

[113] Alexander Strehl and Joydeep Ghosh. "Cluster Ensembles – A Knowledge Reuse Framework for Combining Multiple Partitions". en. In: ().

[114] *Comparing clusterings | Proceedings of the 22nd international conference on Machine learning*. URL: https://dl.acm.org/doi/abs/10.1145/1102351.1102424 (visited on 08/16/2026).

[115] Andrew Rosenberg and Julia Hirschberg. "V-Measure: A Conditional Entropy-Based External Cluster Evaluation Measure". en. In: ().

[116] *The Detection of Disease Clustering and a Generalized Regression Approach | Cancer Research | American Association for Cancer Research*. URL: https://aacrjournals.org/cancerres/article/27/2_Part_1/209/476508/The-Detection-of-Disease-Clustering-and-a (visited on 08/16/2026).

[117] P. Legendre and Louis Legendre. *Numerical Ecology*. en. Elsevier, July 2012. ISBN: 978-0-444-53869-7.

[118] Christopher Manning, Prabhakar Raghavan, and Hinrich Schuetze. "Introduction to Information Retrieval". en. In: (2009).

[119] *Vx Underground*. en. URL: https://vx-underground.org/ (visited on 08/16/2026).

[120] *MalwareBazaar | Malware sample exchange*. URL: https://bazaar.abuse.ch/ (visited on 08/16/2026).

[121] *SecPriv/adapt*. original-date: 2024-07-24T11:30:57Z. July 2026. URL: https://github.com/SecPriv/adapt (visited on 08/16/2026).

[122] *Threat Group Cards: A Threat Actor Encyclopedia*. URL: https://apt.etda.or.th/cgi-bin/aptgroups.cgi (visited on 08/16/2026).

[123] *MISP/misp-galaxy*. original-date: 2016-02-27T20:11:06Z. Aug. 2026. URL: https://github.com/MISP/misp-galaxy (visited on 08/16/2026).

[124] *HydraDragonAntivirus/HydraDragonAntivirus*. original-date: 2024-05-22T11:24:00Z. Aug. 2026. URL: https://github.com/HydraDragonAntivirus/HydraDragonAntivirus (visited on 08/16/2026).

[125] *SupportIntelligence/Icewater*. original-date: 2017-08-10T17:21:23Z. July 2026. URL: https://github.com/SupportIntelligence/Icewater (visited on 08/16/2026).

[126] *plyara/plyara*. original-date: 2018-06-05T00:53:20Z. Apr. 2026. URL: https://github.com/plyara/plyara (visited on 08/16/2026).

[127] *VirusTotal/yara-python*. original-date: 2015-09-11T09:37:35Z. Aug. 2026. URL: https://github.com/VirusTotal/yara-python (visited on 08/16/2026).

[128] *Groups | MITRE ATT&CK®*. URL: https://attack.mitre.org/groups/ (visited on 08/16/2026).

[129] J. Roger Bray and J. T. Curtis. "An Ordination of the Upland Forest Communities of Southern Wisconsin". In: *Ecological Monographs* 27.4 (1957), pp. 326–349. ISSN: 0012-9615. DOI: 10.2307/1942268. URL: https://www.jstor.org/stable/1942268 (visited on 08/16/2026).

[130] *Assessed Cyber Structure and Alignments of North Korea in 2023 | Mandiant*. en-US. URL: https://cloud.google.com/blog/topics/threat-intelligence/north-korea-cyber-structure-alignment-2023 (visited on 08/16/2026).

[131] *Lazarus Group, Labyrinth Chollima, HIDDEN COBRA, Guardians of Peace, ZINC, NICKEL ACADEMY, Diamond Sleet, Group G0032 | MITRE ATT&CK®*. URL: https://attack.mitre.org/groups/G0032/ (visited on 08/16/2026).