Actively Understanding the Dynamics and Risks of the Threat Intelligence Ecosystem

Tillson Galloway, Allen Chang, Omar Alrawi, Thanos Avgetidis, Manos Antonakakis, Fabian Monrose
Georgia Institute of Technology
{tillson, allen.chang, alrawi, avgetidis, manos, fabian}@gatech.edu

***Abstract*—Despite the billions of dollars invested in the threat intelligence (TI) ecosystem—a globally distributed network of security vendors and altruists who drive critical cybersecurity operations—we lack an understanding of how it functions, including its dynamics and vulnerabilities. To fill that void, we propose a novel measurement framework that tracks binaries as they traverse the ecosystem by monitoring for watermarked network Indicators of Compromise (IoCs). By analyzing each stage of the propagation chain of submitted TI (submission, extraction, sharing, and disruption), we uncover an ecosystem where dissemination almost always leads to the disruption of threats, but vendors who selectively share the TI they extract limit the ecosystem's utility. Further, we find that attempts to curtail threats are often slowed by 'bottleneck' vendors delaying the sharing of TI by hours to days.**

Critically, we identify several threats to the ecosystem's supply chain, some of which are presently exploited in the wild. Unnecessary active probing by vendors, shallow extraction of dropped files, and easy-to-predict sandbox environment fingerprints all threaten the health of the ecosystem. To address these issues, we provide actionable recommendations for vendors and practitioners that improve the safety of the TI supply chain, including detection signatures for known abuse patterns. We collaborated with vendors through a responsible disclosure process, gaining insight into the operational constraints underlying these weaknesses. Finally, we provide a set of ethical best practices for researchers actively measuring the threat intelligence ecosystem.

I. Introduction

Threat intelligence (TI) involves the process of gathering, analyzing, and utilizing information about cyber threats to improve the security of an organization [1]. The TI market is rapidly expanding and is projected to surpass $15 billion by 2026 [2]. This growth is driven by the increasing demand for high-quality data that enables organizations to react better to emerging threats. The globally distributed vendors and individual contributors within the ecosystem generate a wide and intricate web of indicators derived from analyzing malicious artifacts such as binaries, phishing websites, and emails [3]. The sharing in this ecosystem is driven by a combination of altruism and commercial agreements, and ultimately TI is shared with defenders. For these defenders, navigation and assessment of TI dataset quality and relevance are pressing challenges [4].

Researchers and practitioners have explored various aspects of the threat intelligence ecosystem to better understand its sources, sharing, and implementations [5]–[10]. Although prior work has demonstrated how TI can be extracted and measured, fundamental gaps remain in our understanding of the ecosystem's topology, supply chain, and threat surface. Limited to passive and locally scoped datasets, existing studies fail to capture the global dynamics of indicator of compromise (IoC) propagation and the nuances of TI sharing or timeliness. To address these shortcomings, we adopt an active measurement approach that examines real-time interactions among antivirus engines, sandboxes, TI platforms, and blocklisting services, enabling us to reconstruct the ecosystem's topology and assess detection delays. This method also reveals the behaviors of closed-source vendors, broadening visibility into otherwise opaque areas of the ecosystem. Further, our findings highlight ongoing adversarial exploitation, where we identified over 800 binaries uploaded to VirusTotal embedding sandbox-specific IPs or URLs within a 90-day window in early 2025. These artifacts show that adversaries deliberately employ evasion strategies targeting known TI infrastructure.

Our novel measurement framework leverages active probing to trace IoCs across the global TI ecosystem. We design and submit benign yet suspicious binaries to 30 security vendors—spanning antivirus engines, sandboxes, and TI platforms—each embedded with custom file-dropping and provenance-tracking mechanisms. The framework allows us to observe the analysis and sharing of IoCs through submission, sandbox analysis, sharing, blocking, and takedown. Our approach reveals previously unreported vendor behaviors and sharing relationships, including those related to vendors without open data feeds. To further characterize ecosystem actors, we introduce a clustering technique that fingerprints entities based on their network and system-level responses. Crucially, our work addresses ethical concerns often neglected in earlier studies by explicitly accounting for human deception, resource overhead, and privacy. We consider three research questions (RQs):

- **RQ1 [Propagation]:** How do security vendors differ in their ability to analyze malware and share extracted indicators of compromise (IoCs) across the ecosystem?
- **RQ2 [Disruption]:** How do differences in analysis and sharing affect the speed and effectiveness with which vendors block IoCs or take down (or suspend) infrastructure?
- **RQ3 [Evasion]:** How are adversaries exploiting gaps in analysis, sharing, or disruption, and what strategies can improve the ecosystem's resilience against such evasions?

Our study reveals a stratified structure in the TI ecosystem that directly impacts IoC sharing and response times. While sandbox analysis occurs rapidly and is widespread (performed

Network and Distributed System Security (NDSS) Symposium 2026
23 - 27 February 2026 , San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240102
www.ndss-symposium.org

by 67% of vendors), only 17% of vendors share the extracted TI, and only two of the 30 vendors contributed to downstream domain takedowns, highlighting a gap between TI extraction, sharing, and action (**RQ1**). We identify four central 'nexus' vendors that heavily enrich and redistribute TI, serving as aggregation points for the broader ecosystem. Most other vendors consume or extract intelligence but do not reshare, creating dependency bottlenecks and potential single points of failure. We further find that network IoCs are reshared far more frequently than binaries. In some cases, domain lookups were 20× more prevalent than dynamic analysis executions.

Although IoCs are typically extracted within seconds, we observe that sharing delays—often hours to days—propagate across the supply chain, increasing time-to-block or suspend by up to 20% (**RQ2**). By simulating adversarial evasion tactics against our measurements using known sandbox blocklists, we find that while central nexus vendors are rarely bypassed, evasions can reduce the number of vendors receiving extracted TI by 25%, indicating that these tactics can degrade TI coverage across the ecosystem (**RQ3**).

Finally, we discovered striking evidence of adversaries actively attempting to evade sandboxes via fingerprinting techniques and show that insufficient diversification of sandbox fingerprints puts vendors at risk for these evasive tactics. We communicated our findings to 30 security vendors through a responsible disclosure process. Through this effort, we provided vendors with recommendations and detection rules for reducing their exposure to the weaknesses discovered in our work.

In summary, the contributions of our paper are as follows:

1) **Understanding the TI supply chain.** We introduce a novel, data-driven methodology for measuring the behaviors and relationships that drive the TI supply chain, enabling quantitative analysis of its completeness, coverage, timeliness, and information-sharing dynamics. Using telemetry collected from over 60 vendors, we find that domains are shared far more frequently than malware binaries, a disparity that may hinder coordinated detection engineering and remediation efforts.
2) **Assessing Fingerprints.** We systematically measure the diversity and reuse of sandbox fingerprints across non-premium vendors, finding that most expose themselves to evasion and vendor attribution attacks by relying on uniform and predictable environments.
3) **Adversarial tactics and community recommendations.** We identify real-world malware samples exploiting systemic weaknesses in the TI ecosystem, with observed abuse persisting as recently as June 2025. Building on these findings, we provide practical recommendations and detection signatures to help vendors and operators mitigate such abuses. Finally, we outline ethical guidelines for future researchers conducting active probing of TI infrastructures to ensure responsible and transparent experimentation.

II. Background & Related Work

A. Background — Threat Intelligence

Broadly, threat intelligence (TI) comes in operational, strategic, and tactical forms. Operational and strategic TI sources typically include high-level findings related to attacker behaviors, trends, and attribution. We primarily focus on data-driven tactical TI, which includes indicators of compromise (IoCs), including binaries, file hashes, domains, IPs, and URLs. Defenders use these IoCs for detection, incident response, and threat hunting.

The TI ecosystem contains many organizations, including vendors, operators, researchers, and adversaries. Vendors are commercial or non-profit security organizations that extract and share TI. In addition, vendors can use this TI for detection engineering and for disrupting adversaries' campaigns. Operators (or practitioners) are the beneficiaries of the enrichment, sharing, and detections provided by TI. They use threat intelligence in operational environments, including security monitoring, forensic analysis, and attribution, and lawful takedowns. In contrast to vendors, operators are concerned with protecting a network or investigating a specific threat. Operators may offer security products and services, but are not directly involved in the production of threat intelligence. However, in performing their tasks, operators may generate relevant data. Operators can include security teams within an organization, a managed security service provider (MSSP), a forensic law firm, or law enforcement. Researchers study the ecosystem for the purpose of discovering and minimizing threats. They differ from vendors in that they do not offer services or products that are directly used by customers.

B. Related work

Studies related to TI sharing can be broadly grouped into four categories: sharing practices and barriers, evaluation and quality assessment of feeds, usage of TI platforms in research, and methods for indicator extraction and automated analysis.

Sharing between organizations. Existing studies have explored the benefits and challenges associated with TI sharing among organizations. The barriers identified include trust in upstream sources, privacy concerns, and organizational readiness [3], [10]–[12]. Adoption of TI within organizations has been linked to factors such as organizational size, resource availability, and the financial cost of acquiring commercial feeds [13], [14]. Sauerwein et al. [9] highlighted informal and ad-hoc processes that organizations often use in sourcing and utilizing TI, which can lead to inconsistent threat responses. In contrast, our work actively tracks the end-to-end flow of TI from extraction to disruption. We explicitly address privacy concerns and integrate provenance tracking to improve our understanding of TI sharing relationships.

Assessment of TI Feeds. Researchers have proposed metrics and approaches to evaluate the quality and usefulness of TI feeds. Evaluations include assessments of IoC labeling accuracy, indicator timeliness and shelf life, and correlations among different TI sources [7], [8], [15]–[21]. Bouwman et al. [6] demonstrated differences between commercial and open-source TI feeds in terms of overlap, coverage, and timeliness, while Kuhrer et al. [5] showed limited coverage of public blocklists. Our approach differs by actively measuring the TI supply chain to observe real-time generation and propagation of IoCs, allowing us to assess the ecosystem's resilience against manipulation and evasion tactics directly.

Usage of TI platforms. Public TI platforms, such as VirusTotal, have become central to many security workflows. Researchers often rely on these platforms to obtain detection verdicts and malware sandbox analysis reports for evasive samples [22], [23]. Prior evaluations of these sandboxes provide insight into the static signatures [24], [25] and behaviors [26] present in dynamic analysis environments.

IoC extract and analysis. Automated methods for IoC extraction employ graph mining, natural language processing, and various machine learning techniques [27]–[29]. Machine learning models have also been proposed for assessing indicator relevance and automatic blocking or takedown of malicious domains [30], [31]. Recent studies have identified vulnerabilities in automated domain-blocking systems, demonstrating susceptibility to evasion and poisoning attacks, some of which can be mitigated by adversarial training [31], [32].

Our work. Prior studies of TI platforms [24], [25] have primarily analyzed sandbox environments in isolation, with a key theme of identifying static or behavioral fingerprints that malware may exploit for evasion. Our work differs in several important ways. We repurpose these fingerprints to conduct active, ecosystem-scale measurements of vendor practices and relationships, extending the scope of analysis beyond isolated sandbox evasion. We introduce a novel telemetry mechanism that uses watermarked indicators of compromise that serve as provenance trails that trace how binaries propagate across the ecosystem. Finally, we provide empirical evidence that fingerprint-based evasion is not merely a theoretical concern, documenting concrete cases where cybercriminals actively exploit these weaknesses.

Additionally, our study extends prior assessments of various TI feeds, largely centering on IoCs associated with known malware samples (e.g., STIX objects [21] and network-level IoCs [7], [8], [15]–[20]). In contrast, our provenance-based methodology enables active inference of vendor-to-vendor sharing relationships, revealing the underlying structure of information exchange within the TI ecosystem. This capability allows practitioners to identify propagation bottlenecks and can be used to attribute controlled data leaks. Furthermore, our active measurement framework captures the downstream operational effects of TI sharing, including firewall-level blocking and registrar takedowns, providing visibility into how shared intelligence translates into concrete defensive actions. Finally, unlike all these works, we differentiate between the sharing of derived threat intelligence (e.g., network IoCs and file hashes) and the exchange of full binary artifacts, the latter offering deeper insight into the threat's behavior and enabling richer downstream analysis.

III. Measurement Methodology

Measuring the lifecycle of threat intelligence in near real-time is challenging. In what follows, we outline three requirements for conducting such a study, as well as the design choices we made to facilitate a sound analysis thereof.

To address our three research questions on propagation, disruption, and evasion, we examine three key attributes of the threat intelligence ecosystem: (a) *Extraction*, which refers to how threat intelligence is generated from binaries through sandbox analysis; (b) *Communities*, which capture how IoCs are shared and propagated among ecosystem participants; and (c) *Timeliness*, which concerns how quickly vendors extract, classify, and act on intelligence.

We study the *extraction* attribute by examining the number of distinct sandboxes used by a vendor as well as the depth to which a submission is analyzed. A vendor who analyzes both a binary and its dropped files is considered more comprehensive (deeper analysis) than one who does not. For the *community* attribute, we deduce sharing channels using a novel method for tracking binaries and their IoCs. By incorporating a provenance trail that creates unique binaries and IoCs, we can learn with whom vendors share information. We measure the connectivity of sub-communities using graph metrics to infer relationships. The constructed provenance trail enables us to study IoCs *timeliness*, or delays in analysis and sharing. Moreover, we actively check open source intelligence and blocklists for the IoCs to measure the time vendors take to act on intelligence that is gathered.

Requirement I: Fingerprinting Analysis Tools. Security vendors often analyze binaries using a diverse set of tools across multiple environments (e.g., varying operating systems, virtual memory configurations, or network setups). These environment-dependent analyses produce threat intelligence that we characterize as the *extraction* attribute of the TI ecosystem. Relying solely on network-level features to identify a vendor would obscure important aspects of their extraction tactics (such as shared tools and the use of cloud infrastructure or proxies). To capture these nuances, our measurement pipeline must record high-signal fingerprints of the sandbox environment, enabling accurate attribution and deeper insight into vendor-specific analysis behavior.

Requirement II: Adaptive Intelligence Tracking. Security vendors frequently share threat intelligence that was either generated in-house or aggregated from external sources, whether free or commercial. However, this sharing is often selective in that vendors may disclose only certain binaries or limit sharing to the network indicators produced during execution. During dynamic analysis, a vendor might execute a binary multiple times across different environments or using varied toolchains, yielding diverse outputs. In some cases, a vendor may share extracted indicators with another vendor, who then investigates those indicators without ever possessing the original binary. As a result, for an external network observer, it becomes infeasible to distinguish between the sharing of binaries and the sharing of derivative indicators. This ambiguity highlights the need for a robust measurement pipeline capable of tracking threat intelligence as it flows across vendors and analysis stages.

Requirement III: Triggering Malicious Verdicts. Security vendors often implement filtering mechanisms to ensure their feeds provide useful data [3], [33]. For an indicator to be included in a feed, it is reasonable to assume that the binary associated with the indicator must first be perceived as malicious. Hence, the binaries used in the study must induce suspicious behavior that would trigger malicious verdicts.

A. Design

To meet these requirements, we designed the measurement pipeline depicted in Figure 1. Our design consists of a *Generator* and an *Observer*. The Generator automatically produces binaries that, when executed, create unique indicators that can be tracked remotely and passively by the Observer.

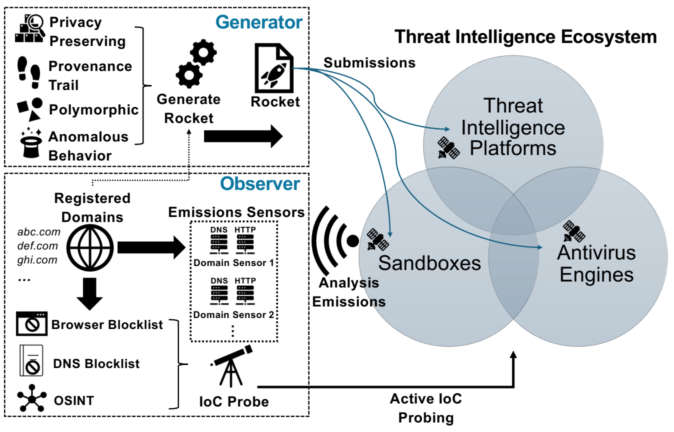

Fig. 1: **Measurement Pipeline.** The Generator creates binaries (Rockets) and deploys them. The Observer monitors for analysis emissions from Rockets and their payloads (Satellites).

1) Generator: In the spirit of information discovery terminology, our Generator creates a unique binary referred to as a *Rocket*. The binary is a defanged (i.e., modified so that it is non-harmful) program with suspicious behavior. When the Rocket is executed, it emits a unique domain name indicator containing encoded information about the specific sandbox environment. Then, the Rocket generates a modified copy of itself and stores (or *drops*) this copy on the host. The dropped binary is called the *Satellite*. Together, the Rocket and Satellite allow us to track the sharing of the binaries and their IoCs.

To ensure that an observer can disambiguate executions of the binaries from lookups of IoCs, we leverage HTTP as the primary signaling channel. We implement different modules for the aforementioned requirements: (1) a *recursive method* that uniquely tracks binaries as they traverse the ecosystem. Randomly generated IDs are assigned to each execution. Additionally, the Rocket drops a copy of itself. This new 'Satellite' is uniquely connected to the Rocket. (2) a *privacy-preserving* module that collects features of the analysis environment where the binary is run, hashes each attribute, and encodes the resulting fingerprint as part of the domain name. (3) a *defanged keylogging behavior* that mimics basic malware capabilities. This is designed to induce automated detections, blocking, and takedown techniques.

Our system fingerprint consists of three features selected based on a pilot experiment aimed at validating prior studies [24], [34]. In that pilot, we profiled a handful of sandboxes using 16 features described in these works, but ultimately settled on three features based on *high mutual information, robustness, and privacy considerations*. Specifics of the pilot are in Appendix A. Unlike prior work, we took steps to protect any information that we deemed potentially sensitive among the information collected by the Rocket.

The system shares the hashed fingerprint with the Observer server via a simple HTTP request, where the execution information is encoded in the domain name. However, encoding the fingerprint in a domain name is not straightforward and requires careful analysis to address several practical issues. First, the DNS protocol limits queries to 253 characters. Secondly, although a fully qualified DNS name can be up to 253 characters in length, a given subdomain is limited to 63 characters. Thirdly, the case-insensitivity of domain names constrains our encoding strategy. To ensure we can track sharing to a reasonable depth, we limit our labels to the 36 alphanumeric characters (`[a-z0-9]`). Specifically, the Satellite generates a domain name for each execution in the format $\langle \mathbf{b} \,\|\, \epsilon_{\mathbf{n}}\mathbf{H}_{\mathbf{n}}|\ldots|\epsilon_{\mathbf{1}}\mathbf{H}_{\mathbf{1}}\rangle$, where b is the ID of the binary that we initially submitted to the vendor, ϵ_i is a randomly generated string identifying a distinct execution of the binary, $\mathbf{H}_i$ is a sandbox fingerprint.

Conceptually, this mechanism can be viewed as a type of provenance tracking. We encode information in subdomain labels based on an incremental chained mechanism similar to hash chain approaches used to provide tamper-evident [35] logs in the domain of computer forensics [36]. Specifically, we opted for a *low overhead* approach where the trail is simply the concatenation of multiple execution IDs and fingerprints from a binary's provenance trail. Each fingerprint ($\mathbf{H}_i$) is itself the concatenation of the hashed values of the sandbox's system manufacturer, RAM, and OS install date information. These features are a subset of features used in prior work [24] that performed well in a pilot experiment and are not overly privacy-exposing. The feature selection process is discussed further in Appendix A. By design, tracking is restricted to a maximum depth of 5; no new Satellites are dropped beyond that level (this is to prevent infinite loops of executions that might occur after the binary is submitted to a vendor).

Whenever the rocket is executed, it issues an HTTP request, which will first resolve the domain using DNS. If the resolution is successful, the Rocket sends an HTTP GET request (with a custom User-Agent header) to the resolved IP containing the fingerprint as both a URI and a subdomain. Finally, the Rocket drops (but does not execute) a Satellite file on the host. When executed, Satellites have the same defanged behavior as the Rocket. Crucially, the new Satellite contains the fingerprint and unique ID of its parent, which are appended to the DNS hostname. This enables us to track the sharing of dropped binaries when they are executed by another vendor.

2) Observer: We deploy *Emission Sensors* for each registered domain. Those sensors record emissions received from the binaries. Each sensor runs a DNS authority and an HTTP server. The DNS authority responds to all queries for the registered domain, including all subdomain labels. The HTTP server logs the time, host, user-agent, method, URI, and body for all requests, and returns a *502* response. The sensors use separate IPv4 addresses for DNS and HTTP. These sensors are open to the internet and exposed to scanners and miscellaneous internet noise.

To measure timeliness, the *Observer* also implements an *IoC Probe* that queries several TI sources for the binaries' IoCs. The probe serves as a proxy for inferring how quickly security vendors put the threat intelligence into practice. We actively check if (*1*) the domains or IP addresses have been added to Google Safe Browsing or FireHol lists, (*2*) the domains have been blocked by Quad9, Google, Cloudflare, or Palo Alto DNS Security, or (*3*) if the domains, IPs, or related hashes, appear in the databases of VirusTotal or AlienVault Open Threat Exchange (OTX)[1]. If so, the related IoCs are saved along with the number of detections, the assigned AV labels, when available. We also inspect (*4*) the feed of a well-known commercial threat intelligence vendor with whom we have a contractual arrangement. This vendor's feeds include hashes, domains, URLs, and IP addresses that are deemed malicious.

An example of emissions is shown on the left-hand side of Table I. We return to how vendor labels (in the right-hand side) are assigned in Section III-C.

B. Sourcing security vendors

We started with VirusTotal, a platform that aggregates verdicts from numerous antivirus and sandbox vendors. From this collection of antivirus vendors, we included 20 vendors that have a dedicated web portal for submitting binaries. We then examined the 18 sandboxes that VirusTotal uses. However, we could not utilize any of them because they either required a commercial license, vetted trial via sales team, or the service is not publicly available. To find other reputable sandbox services, we turned to search engines and used specific keywords (e.g., 'malware sandbox,' 'malware detector,' 'dynamic binary analysis'). That search led to security articles that listed a number of commercial and open source sandboxes. We also examined the contents of a popular malware analysis GitHub repository [37]. Through this process, we identified 15 new vendors, most of which met our criteria (freely available and no human in the loop for registration). We included five additional well-known vendors given our prior knowledge. The full list of 40 vendors considered in this paper is in Table XI.

Although there is no agreed upon categorization in the literature, we group vendors based on their primary functions:

- **Threat intelligence (TI) platforms** produce, aggregate, and share threat intelligence from multiple sources. Examples include VirusTotal and MalwareBazaar.
- **Antivirus companies** provide commercial tools that use threat intelligence. They also produce threat intelligence by analyzing specific types of threats (e.g., malware, phishing). Examples include Kaspersky and Sophos.
- **Malware sandbox services** offer services that produce threat intelligence associated with malicious binaries. Examples include HybridAnalysis and Any.Run.

The explicit distinction we make here is that TI Platforms and Antivirus vendors utilize sandbox technologies, but sandboxes are not their core technology offering. In contrast, Sandbox services offer dynamic malware analysis as their product.

C. Labeling vendors

Uploading a binary to a single vendor leads to numerous executions, some of which are spawned by a diverse set of other vendors. To label vendors, we employed a three-step process combining vendor dynamic analysis reports, expert rules, and unsupervised clustering. This approach generates labels for *known* vendors using dynamic analysis reports and human labeling, then generates labels for *external* vendors using an ML-based technique. Specifically, we assign vendor labels to system and network (AS country and organization) fingerprint tuples collected from the Observer logs. *To limit reputational harm and disclosure of potentially private inter-vendor relationships, we generate pseudonyms for each vendor (e.g.,* `INT-AV-1` *for an antivirus) and refer only to these in the results.* The order of the labels is given by the total number of executions observed after submissions to each vendor (see Table II).

The overall process is as follows:

Step 1: Dynamic analysis reports. Whenever possible, we gathered dynamic analysis reports from vendors. These reports contain information collected during binary execution in a sandbox, including the domain names that it queried. Because we encode a uniquely generated execution ID (ϵ) in the subdomain, we can directly link the execution to the vendor who produced the dynamic analysis report. Twelve of the sandbox and TI vendors to whom we submitted a Rocket produced such reports, which allowed us to link 122 (25% of) executions *unequivocally* to their originating vendor.

Step 2: Collaborative hand-labeling. To link the remaining executions with their respective vendors, two Ph.D. students, both with experience in threat hunting, collaboratively analyzed the remaining logs. Each student was given a list of vendors with any HTTP emissions observed and their associated Observer logs, which included each execution's system and network fingerprints and fingerprint trail (if one exists). They were instructed to annotate as many executions as possible with their associated vendor given this information and any available internet sources (e.g., VirusTotal or AS metadata [38]), but to leave executions unlabeled when uncertain. To do this, the students constructed boolean rules, which are logical conjunctions that identify a vendor based on its system and network fingerprint. These boolean rules were then applied to label fingerprints, and therefore, executions. For example, the following rule identifies *Polaris Security* (a fictional company) by matching either the full sandbox fingerprint or the AS organization name. $\mathbf{H} = f_{1-3}$ are the system's hashed manufacturer, RAM, and OS install date, respectively.

$$\left(f_{1=\text{``}\texttt{iy}\text{''}} \wedge f_{2=\text{``}\texttt{v}\text{''}} \wedge f_{3=\text{``}\texttt{pkq}\text{''}}\right) \vee \mathbf{IP}_{org=\text{``}\texttt{Polaris Security LLC}\text{''}}$$

During the first stage, the students worked independently without sharing results to compose rules which, when applied, provided labels for 56 fingerprints corresponding to 381 executions. Cohen's Kappa score [39] for the independent labeling was 0.87. Afterwards, the researchers collaboratively resolved the disagreements. There were no instances where they disagreed on the identity of a labeled fingerprint, only cases where either researcher left a label blank. The disagreements fell into one of two categories: typos (1/7), and overly specific assumptions about the network fingerprint (6/7). The researchers quickly agreed on updated signatures that resolved their minor discrepancies. In total, 55% of executions were labeled by either dynamic analysis reports or collaborative hand-labeling.

[1]Now LevelBlue OTX. We use the former name due to its higher recognition within the security community.

TABLE I: **Example of observer log entries.** The left-hand side of the table contains observed executions (⚙) and IoC probes (🔍), including signatures encoded in the hostname (binary ID b, reversed execution history containing execution IDs (ϵ_i), system fingerprints $\mathbf{H_i}$), and the requesting client's IP. The right-hand side shows analysis of the logs. The logs are divided into two chains of execution trails.

Chain	≣	$\langle \mathbf{b} \,\|\, \epsilon_n \mathbf{H_n} \vert \ldots \vert \epsilon_1 \mathbf{H_1} \rangle$	Client	Observer	Binary	Type	Vendor
1	t_1	`i||`**`qr64jqbphb`**	34.1.0.0/16	HTTP	Rocket	⚙	`INT-TI-3`
	t_2	`i||`**`p7zy5fbpxd`**`|qr64jqbphb`	41.42.0.0/16	HTTP	Satellite A1	⚙	`EXT-DE-1`
	t_3	`i||`**`szbqjqbphb`**`|p7zy5fbpxd|qr64jqbphb`	34.1.0.0/16	HTTP	Satellite A2	⚙	`INT-TI-3`
2	t_4	`i||`**`8f5beasgad`**	84.52.0.0/16	HTTP	Rocket	⚙	`INT-SB-1`
	t_5	`i||`**`4bqdjqbphb`**`|8f5beasgad`	34.1.0.0/16	HTTP	Satellite B1	⚙	`INT-TI-3`
	t_6	`i||`**`4bqdjqbphb`**`|8f5beasgad`	123.5.0.0/16	DNS	Satellite B1	🔍	`EXT-NZ-1`

Step 3: Clustering external vendors. Numerous dynamic analysis executions originated from vendors to whom we did not submit binaries. To label these vendors, we applied an unsupervised learning approach to cluster similar executions together. Two sandboxes are deemed similar if they share a subset of system features (RAM, CPU manufacturer, and OS install date) and network features (organization and country, derived from their autonomous systems).

Our clustering method (Algorithm 1 in Appendix B) works by partitioning a set of fingerprints into clusters sharing similar system and network fingerprints. The clusters are assembled agglomeratively. In each iteration, fingerprint clusters with a minimum distance between fingerprints less than a predefined threshold are merged. We use the normalized Hamming distance (i.e., the proportion of features that differ between two fingerprints). The distance between two clusters is determined by finding the smallest distance between two fingerprints, one from each cluster. To find the max-distance threshold δ, we evaluated the performance of every possible choice of δ on clustering the hand-labeled ground truth.

We derive the label names from the network attributes and assign them to the clusters. For example, a client's IP originates from a Tor exit nodes [40] or VPNs [41] is assigned a cluster label of `EXT-TOR-1` or `EXT-VPN-1`. In all other cases, we used the AS country to generate labels (e.g., `EXT-RU-1`). If a cluster contains executions from multiple countries, the cluster is labeled by a majority vote with ties broken randomly.

Illustrative Example: Table I shows the outcome of a submission to a vendor. The Rocket was executed twice, generating two executions (at t_1 and t_4). In *Chain 1*, two Satellites (A1 and A2) are executed sequentially. Interestingly, `INT-TI-3` is the origin of the execution at t_1 and t_3, but not t_2. We can therefore infer that `EXT-DE-2` received Satellite A1 from `EXT-TI-3`, executed it (thus generating Satellite A2), and then shared Satellite A2 back to `EXT-TI-3`. We call this a cyclic relationship. *Chain 2* begins with `INT-SB-1` executing the Rocket, then `INT-TI-3` executing its corresponding Satellite (B1). Later, an IP address in New Zealand probes the domain name generated by B1, indicating that `INT-TI-3` has shared B1's corresponding domain IoC.

IV. Results

A. Experiment setup

We answered the three research questions by submitting Rocket binaries to the 30 vendors shown in Table XI while the Observer monitored for watermarked IoCs. We implemented the DNS server using *gdnsd* that runs as an authority for each domain. The IP address allocated by the cloud service provider was vetted for residual trust by checking historical blocklists [42]. The servers were hosted in the DigitalOcean North American region. We coordinated with DigitalOcean and used Namecheap as our registrar for the experiments.

We registered a total of nine domains. Given that our findings rely on security vendors having no prior observations of the IoCs [43], [44], we opted for domains in a TLD that became available in 2017 and only chose domains with no prior registrations in DNS zonefiles or WHOIS databases. To avoid any cross-contamination between experiments, we reserved three domains per vendor category.

Filtering aberrant data. DNS and HTTP server logs contain noise from Internet scans, crawls, or attacks. To ensure that we do not use such data in our analyses, we distinguish these unsolicited requests from binary executions with a unique HTTP User-Agent and Host header. That said, we only received HTTP requests for 88% of executions observed in DNS, possibly due to network failures or outbound firewall rules in the sandboxes. We focus on the HTTP logs when studying sharing dynamics, unless noted otherwise.

B. Data collection and clustering

To answer the above research questions, we conducted a measurement (referred to hereafter as Experiment I) where we submitted unique Rockets to 30 vendors. The Observer recorded the telemetry it received, from which we generated a set of labeled clusters.

1) Telemetry: We grouped the 170 unique system and network fingerprints observed in Experiment I into 62 clusters. We hand labeled 19 clusters for known vendors, and the remaining 43 were generated by the clustering algorithm discussed in Section III-C. We annotated the 43 external (`EXT-`) clusters with geographic/network information (i.e., country, continent, or known VPN/Tor/Cloud IP). We emphasize that our geographical observations about a cluster are artifacts of the network's location, not necessarily the vendor's locale. Table II summarizes the telemetry observed in Experiment I. In total, we observed 1,033 total executions from 21 of the 30 Rockets. We observed executions for submissions to all the sandboxes, 8/10 of the TI platforms, and 8/10 of the antivirus portals. Two of the Rockets were observed only in the DNS logs. On average, submissions to TI Platforms led to 61 executions per vendor, far higher than the number of executions seen in antivirus and sandbox submissions. In addition, executions from Rockets submitted

TABLE II: **Summary of results.** Submissions resulting in executions observed only in DNS are marked by an * symbol.

	Submitted Vendor	Execs.	ASNs		Countries	
			HTTP	DNS	HTTP	DNS
Antiviruses	`INT-AV-1`	84	27	24	1	10
	`INT-AV-2`	36	5	3	1	1
	`INT-AV-3`	4	2	4	2	2
	`INT-AV-4`	2	2	1	1	1
	`INT-AV-5`	1	1	1	1	1
	`INT-AV-6`	1	1	1	1	1
	`INT-AV-7`	1	1	1	1	1
	`INT-AV-8*`	1	0	1	0	1
Sandboxes	`INT-SB-1`	102	37	30	11	10
	`INT-SB-2`	3	2	2	2	2
	`INT-SB-3`	2	2	1	1	2
	`INT-SB-4`	2	1	1	1	1
	`INT-SB-5`	1	1	1	1	1
	`INT-SB-6`	1	1	1	1	1
	`INT-SB-7*`	1	0	1	0	1
TI Platforms	`INT-TI-1`	507	89	48	36	23
	`INT-TI-2`	235	5	36	19	14
	`INT-TI-3`	45	16	14	7	5
	`INT-TI-4`	2	2	2	2	2
	`INT-TI-5`	1	1	1	1	1
	`INT-TI-6`	1	1	1	1	1

TABLE III: **Evaluation for clustering algorithms.** Results from cross-validation on hand-labeled executions.

Algorithm	Dataset CV AMI			Best δ
	Train	Val	Test	
Agglomerative	0.88	0.85	0.86	$2/5$
DBSCAN [45]	0.83	0.81	0.73	$2/5$
OPTICS [46]	0.81	0.83	0.74	$2/5$

to TI Platforms resulted in a greater geographic diversity of executions than to Antivirus or Sandboxes.

Only 17/21 Rockets were executed by the vendor to which they were originally submitted—the exceptions are `INT-TI-1`, which only aggregates sandbox results from other vendors, and `INT-TI-2`. Dynamic analysis is a feature of `INT-TI-2`, but our Experiment I binary was not executed (although binaries in subsequent experiments were). We verified this finding by inspecting the Rocket's report on `INT-TI-2`'s website, which confirmed that the binary was uploaded but not executed.

2) Clustering evaluation: We evaluated the quality of the vendor labels generated by Algorithm 1 by benchmarking them against 89/267 fingerprints derived from dynamic analysis or collaborative labeling (see Section III-C). We selected hyperparameters for each clustering algorithm by optimizing over five-fold cross-validation stratified over the 12 out of 20 vendors with three or more unique fingerprints. We add vendors with fewer than three fingerprints to the test set. The results are shown in Table III.

We primarily evaluated our results using the adjusted mutual-information index (AMI), which quantifies the agreement between predicted clusters and ground truth labels while accounting for chance. Compared to similar metrics, AMI works well when measuring imbalanced, few-datapoint clusters [47]. Across all models, $\delta = 2/5$, which groups executions sharing at least three features into the same cluster, maximized the AMI and outperformed random guessing by 5–6$\times$ in the validation and test sets. This indicates that while sandboxes from different vendors might coincidentally share one type of feature, it is less likely that they overlap across three dimensions. Of the three clustering approaches we evaluated, agglomerative clustering yielded the highest validation AMI and is therefore used throughout the rest of the paper to cluster executions.

Among the errors, we identified six instances of over-clustering (one vendor is separated into two clusters), but no instances of under-clustering (multiple vendors are included in one cluster). Over-clustering is the result of a vendor's sandboxes differing in at least three fingerprints, which indicates that a vendor at least partially mitigates basic profiling techniques. The presence of over-clustering is a sign of a healthy ecosystem. `INT-TI-3`'s fingerprints were overclustered the most frequently and were spread across six clusters. Overall, this is likely because `INT-TI-3` aggregates the results of numerous dynamic analysis engines, each of which use different configurations and networks. We also found evidence of over-clustering in the external vendor set. For example, two vendors had similar system fingerprints, but their ASes were slightly different (but clearly related to the same corporate entity). Such naming inconsistencies are prevalent within the AS ecosystem [48].

While increasing the distance metric beyond $\delta = 2/5$ improved over-clustering, it introduced under-clustering. This under-clustering typically occurred when multiple vendors shared common attributes, such as the same AS country or RAM configurations, which led to distinct vendors being incorrectly merged. In addition to maximizing AMI, our choice of $\delta = 2/5$ maximizes vendor disambiguation while avoiding inadvertently merging vendors. Thus, our results approximate an upper bound on the number of vendors.

C. IoC propagation: Extraction and Sharing phases

To answer **RQ1** (comparing members of the TI ecosystem in their processing of IoCs), we analyze two key phases of an IoC's Propagation Chain (Figure 2): Extraction and Sharing.

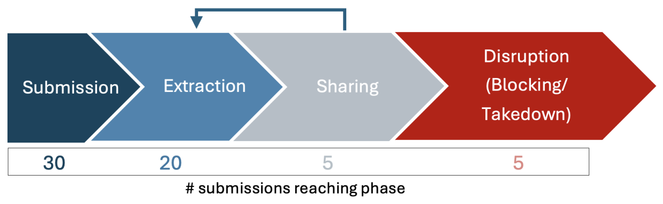

Fig. 2: Phases of an IoC's Propagation Chain

Extraction: Our analysis begins in the extraction phase, where vendors dynamically analyze submitted binaries to extract IoCs. Dynamic analysis is widespread: 20 of the 30 Rockets we submitted were eventually analyzed, and the first analyses almost always occurred within 30 minutes. 7 of the 10 unanalyzed Rockets were uploaded to antivirus vendors, indicating that Sandboxes and TI Platforms vendors are typically capable of IoC extraction.

Vendors vary in their processes for thwarting sandbox profiling and evasion. Figure 3 shows that most vendors generally have three or fewer unique system fingerprints. Additionally, most vendors do not vary their AS organization

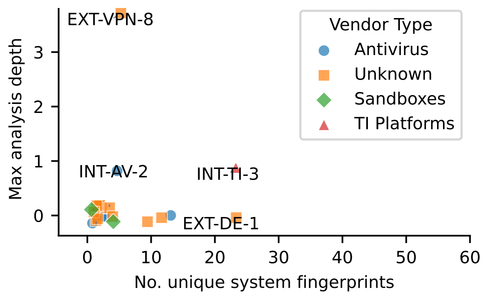

Fig. 3: **Vendor analysis tactics**. Figure shows depth and diversity of fingerprints for different vendor types.

or country, even if they do vary their system fingerprints—some examples are shown in Table IV. Vendors who mitigate both system and network fingerprints become difficult to evade—a high degree of fingerprint diversification from `INT-TI-3` led to our algorithm over-clustering it.

We also found that web search engines' indexing of dynamic analysis reports sometimes allow us to learn more about external vendors. For example, of the 16 IPs associated with `EXT-DE-1`, four were referenced in IoC reports hosted on a public threat feed website, whose homepage explains how they collect publicly available malware samples and run them in a sandbox. This shows how even when a vendor lacks a public submission portal, its extraction and sharing of TI from public sources can reveal its identity.

While most vendors executed the Rocket, we found evidence suggesting that many vendors do not analyze dropped files recursively. That is, most vendors only execute the Rocket, but not any satellites. This is shown in Figure 3, where the *maximum analysis depth* of a cluster is the length of the longest unbroken chain of executions related to a vendor as determined by the provenance trail.

Takeaway: The use of environmental keying by malware [49] underscores the limitations of single-environment analysis. Vendors that execute samples across diverse environments are more likely to uncover hidden or conditional behaviors, leading to higher-fidelity IoC extraction. Likewise, the trend toward multi-stage attacks [50] highlights the need to analyze not just initial payloads but also the full execution chain, including all dropped artifacts. Vendors that reconstruct these chains provide a deeper understanding of the malware's lifecycle and enhance the completeness of threat intelligence.

TABLE IV: **System and network fingerprints**. Fingerprints are f_1 = ram, f_2 = sysManufacturer, and f_3 = installDate.

	System				Network	
Cluster	f_1	f_2	f_3	Total	Country	Org
`EXT-DE-1`	3	28	55	56	1	9
`EXT-VPN-2`	1	1	1	1	19	28
`INT-TI-3`	6	22	7	23	4	5
`INT-AV-1`	3	1	13	13	1	1
`INT-AV-2`	1	1	3	3	2	10

TABLE V: **Node degrees in vendor sharing graph**.

Cluster	In-degree	Out-degree
`INT-AV-1`	4	14
`INT-AV-2`	5	0
`INT-AV-7`	2	0
`INT-AV-5`	6	0
`INT-AV-6`	3	0
`INT-AV-12`	1	0
`INT-AV-3`	0	1
`INT-SB-5`	1	0
`INT-SB-1`	1	17
`INT-SB-4`	5	0
`INT-SB-6`	2	0
`INT-SB-3`	1	0
`INT-SB-2`	0	1
`INT-TI-1`	0	37
`INT-TI-3`	5	16
`INT-TI-2`	0	15
`INT-TI-5`	1	0
`INT-TI-4`	0	2

Sharing: Next, we examine sharing of IoCs, which occurred following 16% of the submissions. Figure 4 shows how Rockets uploaded to one vendor spread to other vendors, where they are then re-analyzed to extract additional information. Vendors either act as producers, consumers, or both, as shown by their in- and out-degrees in Table V. Unsurprisingly, TI platforms generally have a high out-degree, indicating sharing with numerous other vendors. Antivirus providers rarely share, with the exception of one antivirus provider who consistently re-submitted Rockets to `INT-TI-3`. Similarly, only 1/6 sandbox vendors shared, which indicates a gap between the public-facing TI uploaded to popular sandboxes and their dispersion via TI platforms.

The high amount of sharing surrounding popular TI platforms positions them as central members of their sharing communities. We refer to these vendors as *nexus* vendors, and define them as vendors in the sharing graph with non-zero in- and out-degrees—i.e., they both consume binaries from a third-party vendor and share new binaries with other vendors. Although `INT-TI-1` does not meet this criteria, we also identify it as a nexus vendor as the company operates a publicly available crowdsourced threat feed. Information on its website references the fact that many samples are uploaded by both automated and human sources on a daily basis. Outbound sharing from a nexus vendor is typically facilitated 'as-a-service' by free or paid APIs (e.g., VirusTotal's popular file stream service or Malware Bazaar's API). Inbound sharing with a nexus vendor stems from an attempt to share a binary with the wider community, or as a side effect of a vendor seeking additional intelligence about a binary (e.g., learning about malicious convictions or related IoCs).

Many relationships between intelligence producers, sharers, and consumers in this ecosystem are asymmetric: there is a clear distinction between vendors who are primarily intelligence producers (low in-degree, high out-degree), redistributors (high in-degree, high out-degree), and consumers (high in-degree, low out-degree). This relationship is shown in Table V when the in-degree and the out-degree differ. In general, Antivirus vendors consume more than they share. Sandboxes generally do not share either, even though sandbox reports for our submissions are all publicly accessible.

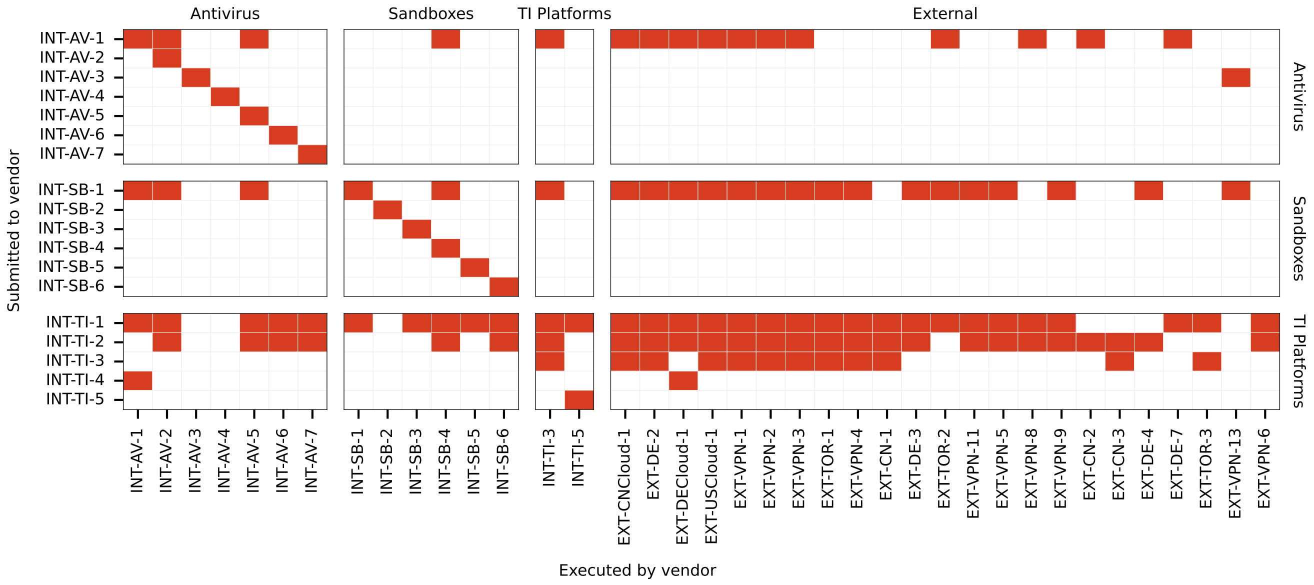

Fig. 4: **Sharing relationships.** Rows represent vendors to which we submitted a binary. External vendors that executed a binary from a single source are omitted.

Takeaway: Although few vendors independently perform deep, recursive analysis of dropped binaries (Figure 3), such analysis often emerges through cyclic inter-vendor relationships. In these cycles, one vendor analyzes an initial sample (the Rocket), extracts a dropped file (a Satellite), and another vendor subsequently analyzes and shares that file. This emergent form of distributed analysis enables broader IoC extraction by uncovering behaviors that may be hidden in isolated environments—particularly those relying on sandbox evasion. However, this decentralized approach also fragments visibility: parent-child relationships between binaries often get lost across vendors, impeding threat correlation, hunting, and attribution.

D. *Propagation of network IoCs*

To fully answer **RQ1**, we now turn to an analysis of the differences between the sharing of binaries versus the sharing of domain or URL IoCs. Recall that our design allows us to distinguish between those cases due to several factors. First, both the domain and the URL contacted by our binary include a unique format containing the binary ID, the system fingerprint, and the provenance trail. Second, we distinguish between IoC lookups and emissions from the execution of a binary using a unique user agent. Third, we distinguish between domain and URL IoCs based on the presence of the URI.

1) Sharing: The received telemetry shows that domain indicators are shared more frequently than the actual binaries. Moreover, several Antivirus and Sandbox vendors do not share binaries but do share IoCs. Like with binaries, much of this sharing is facilitated by nexus vendors. Table VI shows how six vendors share domain IoCs with nexus vendors `INT-TI-3` and `INT-TI-2`, who make them available on their public-facing platforms. Moreover, we found evidence of data stratification in the ecosystem, where IoCs are selectively shared with downstream vendors—six vendors appear to share domain indicators with nexuses, but only four also share URL IoCs. Further, more than 50% of the indicators produced by each vendor remain unshared.

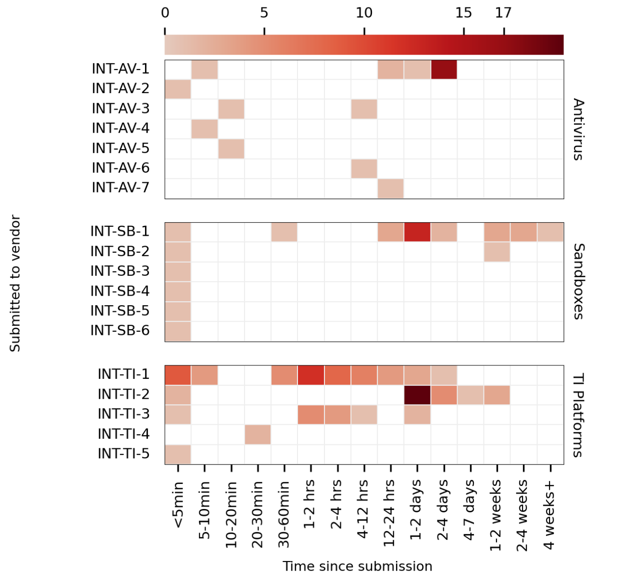

Fig. 5: **Binary sharing velocity.** This figure shows the number of clusters executing Rocket binaries for the first time.

2) Sharing velocity: We now examine timeliness, or the speed with which participants actively probe network IoCs after they are first generated by a Rocket or Satellite (Figure 6). Unlike the short lifecycle of binary sharing processes (Figure 5), probes to domain IoCs are more frequent and continue to occur weeks after the domain was generated.

In some extreme cases where IoCs are shared widely via nexus vendors, we observed more than 100 lookups/day for an IoC for the first week—more than 20× the number of dynamic analysis executions observed during the same timeframe.

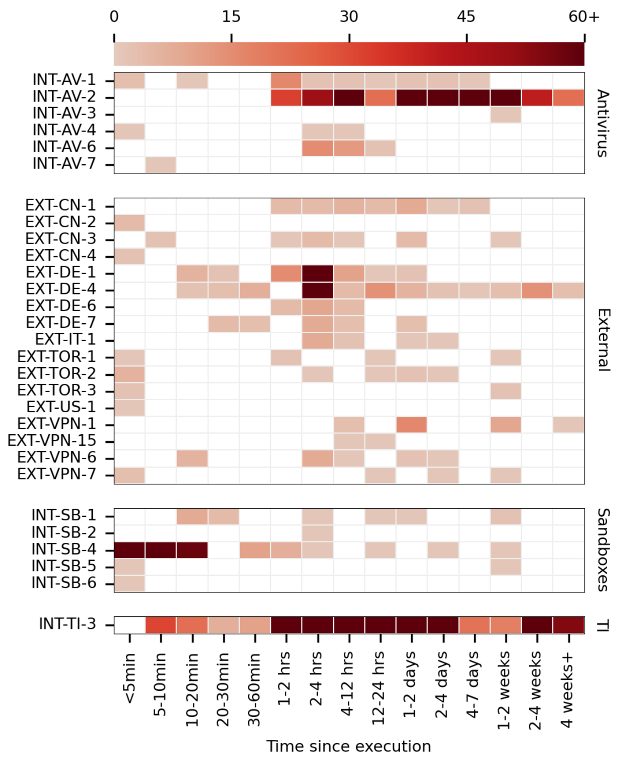

Fig. 6: **Domain IoC probes.** Figure shows probes over time grouped by the cluster executing each binary.

TABLE VI: **IoCs shared with nexus vendors**. As gathered from the VirusTotal and AlienVault OTX APIs.

Cluster	All Execs.	INT-TI-3 Domains	INT-TI-3 URLs	INT-TI-2 Domains
INT-AV-2	213	16	0	12
INT-TI-3	110	45	33	11
INT-AV-1	15	1	0	1
EXT-DE-1	83	6	5	0
EXT-TOR-1	19	2	0	0
EXT-CN-1	6	1	0	0

E. Disruption

To answer **RQ2**, we consider three categories of actions that disrupt adversaries' use of IoCs: blocking on commercial platforms, blocking on free platforms, and domain takedown. Table VII shows that free/commercial blocking typically occurs within hours after sharing, while takedown occurs around 10 days after sharing. Domain takedown is a more serious form of disruption than blocking, and therefore extra scrutiny (either by human or an automated system [30], [51]) is required.

Interestingly, although many disrupted domains are shared with many vendors, a Student's t-test [52] found no statistically meaningful correlation ($p \gg 0.05$) between the number of URL detections reported by VirusTotal partner vendors and the eventual blocking or takedown of those domains (Table VII). Even more strikingly, despite an average of 27.5 vendors identifying the associated binary as malicious, the corresponding URLs receive only about three malicious verdicts on average. Detection coverage is even more sparse at the domain level, averaging around one detection per domain.

TABLE VII: **Domain blocking and suspension**

		Duration before action			# VT
		DNS	OSINT	Takedown	detections
Exp 1	TI Platform	⊘ (1h)	○	⊘ (10d)	3
	Antivirus	⊘ (2d)	○	○	5
	Sandbox	⊘ (1d)	○	○	1
Exp 2	TI Platform	⊘ (13h)	⊘ (1h)	○	1
	Antivirus	○	○	⊘ (11d)	2
	Sandbox	○	○	⊘ (8d)	1
Exp 3	TI Platform	○	⊘ (1h)	⊘ (9d)	1
	Antivirus	○	○	⊘ (11d)	2
	Sandbox	○	○	⊘ (8d)	1

⊘ - Blocked ○ - No Action Taken (90 days out)

As TI traverses the ecosystem, the processing time from each vendor introduces a propagation delay, which can delay the blocking or takedown of an IoC (**RQ2**). This delay is shown in Figure 5 and partially explains the gap between binary submission and detection. For example, INT-AV-1 appears to share with a nexus vendor only after around two days, which causes a burst in executions and ultimately, takedown in 2/3 experiments. INT-TI-3 also has a small propagation delay of around one hour, leading to the lower bound for blocking on the commercial DNS provider. More generally, while sandbox and sharing processes begin seconds after production, executions continue to increase–around 50% of the total executions occurred after three days, primarily due to cyclic submissions to nexus vendors.

Case Study: To showcase how threat intelligence propagates and results in disruption, we examined one submission which led to a domain takedown. Figure 7 shows how one Rocket submitted to a TI vendor was shared, resulting in the blocking and takedown of its communicating Observer domain. Our registrar publicly attributed the domain's takedown to inclusion on this blocklist. We found evidence of a related Satellite's SHA256 hash also in this blocklist, implying that this vendor extracted the domain from this Satellite. This is consistent with prior work [51], which also identified this registrar and blocklist provider as popular takedown/sinkholing services.

Takeaway: Disruption depends on several factors, including extraction quality, sharing completeness, and timeliness. We found that suspension by registrars occurs between 8-11 days, while blocking by DNS firewalls and recursive resolvers often occurs within 24 hours. Further, disruption depends on the cooperation of multiple vendors in an IoC's Propagation Chain. Delaying TI sharing can provide a vendor with feed exclusivity [4] but can prolong widespread blocking and domain takedown.

F. Impact of obfuscation

One troubling finding from Experiment I was the fact that most vendors did *not* analyze the dropped Satellites,

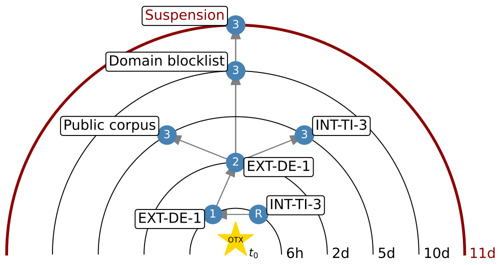

Fig. 7: **Case study.** Each blue circle depicts an execution or action taken on a Rocket (R) or Satellite (numbered).

which could enable adversaries to evade sandboxes (**RQ3**) through packing. To gain deeper insights into this behavior, we conducted a separate measurement (Experiment II) with a setup similar to the previous one, but with the exception that the Rocket is packed with encryption and compression using the UPX packing algorithm. When the packed binary is executed, it decrypts the Rocket and drops it onto the file system. Like the first experiment, the dropped binary is not automatically executed. The packer also generates network emissions that indicate that it has been executed.

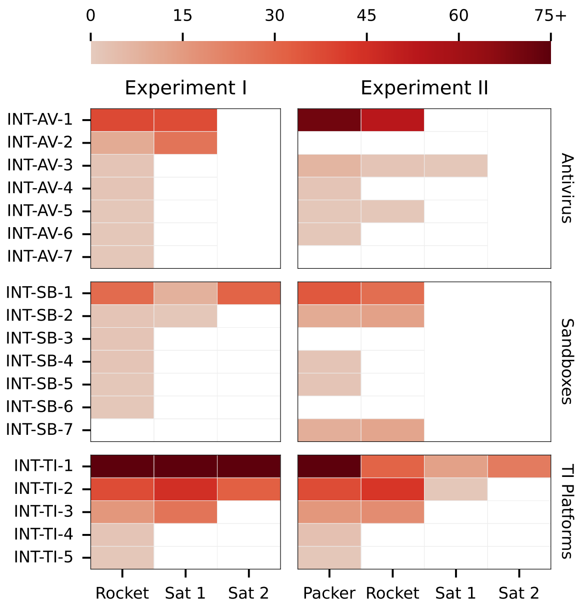

Fig. 8: **Packed binaries.** Comparison of binary executions between Expts. I and II.

Unfortunately, packed Rockets were executed 35% less than unpacked ones (Figure 8), despite being convicted more often by VirusTotal vendors (Table VIII). To assess whether vendors were convicting files based on the presence of the packer or the actual content of the dropped payload, we conducted a final measurement (Experiment III), where the same packing algorithm was used to encrypt and compress the popular *WinRAR* compression utility.

We found that the anomalous nature of packed files makes the outer shell more toxic. VirusTotal also labeled the submitted binaries in Experiment II/III as 'droppers' due to the use of the UPX packing algorithm, while the Rocket in Experiment I was not tagged as a dropper (despite the fact that Rockets drop Satellites!). Instead, Experiment I was tagged only as a 'keylogger'. Conversely, the Experiment II packer was not tagged as a keylogger, despite it containing a packed keylogger.

TABLE VIII: **VirusTotal detections.** Antivirus detections for the binaries submitted to vendor portals (72 total detectors).

Submitted to	Detections I	Detections II	Detections III	Analyses I	Analyses II	Analyses III
`INT-AV-1`	35	37	**40**	2	3	2
`INT-SB-1`	32	36	**38**	1	1	3
`INT-SB-6`	-	-	**33**	-	-	1
`INT-TI-1`	33	**37**	34	114	114	117
`INT-TI-2`	34	37	**38**	1	1	3
`INT-TI-3`	31	**32**	31	1	2	1

Takeaway: These variations in detection rates highlight how vendors differ in their conviction criteria, and that the results of recursive analyses are generally not leveraged, indicating a gap in vendors' processes. The fact that 85% of antiviruses and 57% of sandboxes did not execute the packed Rocket suggests that the responsibility of recursively executing binaries falls onto the users of submission portals or threat feeds. Unfortunately, recipients of IoC feeds are often disallowed from performing further extraction that may produce active egress traffic. Additionally, because threat investigators are forced to manually link relationships and behaviors of the packers, Rockets, and Satellites, it becomes easy to miss the full picture of a campaign and its TTPs.

G. *Adversarial tactics in the wild*

1) Evidence of abuse: To answer **RQ3**, we found evidence of fingerprint-based evasion using VirusTotal's Retrohunt feature, which allows users to search the past 90 days of file uploads using YARA signatures. YARA is a widely used schema for matching combinations of string and byte patterns in arbitrary files. To identify malware attempting to evade sandbox detection, we crafted a YARA rule that matched all files containing each of the top three IPs observed in our experiments [53]. Presence of these IPs in a file likely indicates an evasion attempt. The search found 874 matches, 95% of which were first uploaded between March and June 2025. Nearly 98% of the matches were Windows or Python-based, but some were written in JavaScript (4 samples) and Java (2 samples). This suggests that the evasion technique may be employed in cross-platform attacks.

To better understand the threat objectives associated with the samples, we used malware family labels provided by VirusTotal and further enriched them using two automated labeling tools: AVClass2 [54] and ClarAVy [55]. This analysis revealed that the two most common malware families are well-known open-source information stealers that explicitly advertise anti-sandbox capabilities. Examining their source code [56], [57], we found that both families use nearly identical sets of blocklisted IP addresses to evade analysis (98% overlap). One author attributes this list to an external anti-VM blocklist project maintained on GitHub [58]. Both families detect broad system and network fingerprints, including PC names, usernames, and MAC addresses.

We also discovered five threat reports published between April 2023 and December 2024 that include a list of IP addresses that, when identified by the malware, halts its execution. All five of these lists overlapped exactly with the open source anti-VM blocklist. Digging deeper, we found further reporting [59] that claims that a 2024 stealer variant downloads the IP evasion list directly from GitHub, indicating a shift from hard-coding blocklisted IPs to dynamically loading them. By running a 90-day Retrohunt scan, we identified 34 files referencing the list's URL, mostly Python scripts and Windows executables.

Cluster	# IPs	Prop.	Prop. downstream	Avg. life (days)
`INT-TI-3`	4	0.11	0.52	725
`INT-AV-2`	3	0.06	0.21	504
`INT-SB-1`	1	1.00	1.00	954
`INT-AV-8`	1	0.08	0.67	93
`INT-SB-4`	1	0.17	0.43	685

TABLE IX: **IPs observed in both the public evasion list and our experiments.** Prop. indicates the proportion of IPs for a given vendor that were also observed in the evasion list.

2) Impact of evasion: We further investigated the impact of the public GitHub evasion list on ecosystem health by simulating its effects on our results, which highlights the impact of several threats to the ecosystem identified by our study. First, we find that IPs being used for a multiple year period makes it easy for adversaries to use the dataset to identify sandbox executions. Despite this, most vendors are not fully blocked by the evasion list—only one vendor, `INT-SB-1`, is fully blocked by the entries on the list. However, even partial blocking of a vendor can have downstream effects. By assuming a Satellite would not have been dropped if its parent binary had been blocked from execution by the public evasion list, we can track downstream effects of evasive IoCs. Table IX shows five examples of vendors with significant downstream blocking. In the most extreme case, a single IP belonging to `INT-AV-8` executes only 8% of the total binaries attributed to `INT-AV-8`, but removing the IP prevents 60% of downstream executions. This reduction in exposure of an adversary's behaviors and IoCs can result in a more covert campaign with fewer IoCs burned and a lower overall attack cost.

Takeaway: Our analysis provides evidence that publicly available IP blocklists are actively used by malware in the wild. Despite only searching for three IP addresses over a 90-day window, we uncovered over 900 matching samples, suggesting that the true scale of sandbox-evading malware is significantly larger. The prevalence of malware directly integrating evasion lists published on GitHub underscores how easily threat actors can implement evasive techniques. The use of such tactics can suppress dropped-file execution, limiting analysis by defenders and downstream vendors.

V. Discussion

A. *Implications & recommendations*

The TI community benefits from a robust ecosystem for sharing, consuming, and enriching IoCs. Processes that inadvertently cause active probing can violate data sharing agreements between data producers and inform adversaries of analysis (**RQ3**). To improve the ecosystem's resilience, we provide several recommendations:

For vendors. Our study revealed that while a few vendors thoroughly analyze malware, most conduct shallow analysis and ignore dropped files by the initial binary. By exploring strategies that take into consideration more comprehensive analysis techniques (e.g., execution of dropped files), vendors would be in a better position to uncover hidden threats and provide more relevant TI. Additionally, the lack of context for detection labels can easily lead to incorrect classifications of a binary or IoC's malware family [60]. Incorporating provenance information into detection processes could help provide more accurate threat labeling and attribution.

Vendors should reduce the risk of fingerprint-based evasion by diversifying their IP space. Simple strategies can include: the use of VPNs, or wider IP pools in multiple different ASNs for sandbox Internet connection in order to reduce the sandbox profiling capabilities of cybercriminals.

For operators. Operators should be more aware of the risks of uploading binaries to public scanning services, which can inadvertently help adversaries by revealing detection capabilities and response times. Our Generator/Observer system can be used to detect the unintended public exposure of controlled TI (e.g., IoCs designated TLP:AMBER or PAP:RED [61]). In this context, the binaries and domains generated by our system would serve as honeytokens [62]. The Observer component monitors for any probing of these honeytokens, while the Analysis Engine identifies and distinguishes the entities responsible for such activity. This approach can reveal breaches of data-sharing agreements between organizations or be used by defenders themselves to self-audit incident response and threat hunting workflows.

When evaluating TI feeds, it is imperative that operators assess how thoroughly and reliably vendors extract IoCs. Understanding which vendors perform deeper analysis and use more diverse detection environments allows operators to prioritize intelligence sources that best match their operational needs. It is also important to consider a vendor's potential upstream providers. As we showed, vendors who introduce a sharing delay can hinder widespread threat detection. Lastly, our system can help identify overlaps in TI sources, helping to reduce redundancy in storage and analysis.

For researchers. In scientific studies, the source and quality of data are paramount. Our findings emphasize the need to carefully consider the source of data when developing measurements or ML models for malware detection. Relying solely on data from antivirus vendors may result in unrepresentative models due to limited analysis depth and regional biases. TI platforms, which ingest data from multiple sources, offer more diverse and comprehensive datasets suitable for training robust models. In addition, researchers must account for potential redundancy and polymorphism prevalent in malware datasets. Studies [63] have shown that a significant portion of malware samples are polymorphic copies, which can skew analysis and lead to overestimation of threat diversity. Understanding the dependencies between vendors is also critical given the fact that many vendors reshare IoCs and detection labels, which can create the illusion of consensus [64]. Our work provides more supporting evidence for considering these inter-dependencies when deciding on detection thresholds. Furthermore, existing metrics used to evaluate feeds may not fully capture the dynamics of the ecosystem (e.g., binaries, domains, and URLs exhibit distinct sharing patterns), necessitating specific approaches to metric development. Lastly, researchers should account for aggressive IoC scanning during experiments and implement appropriate protections and data cleaning.

B. Responsible disclosure

We contacted each vendor to whom we had submitted binaries to inform them of our studies, coordinate public disclosure, and offer remediation support through detection capabilities. Each vendor received an anonymized version of the paper, their assigned anonymous ID, and our YARA rules. For vendors who did not respond initially, we sent a follow-up email after one month. Of the 30 contacts, 17 acknowledged our outreach. Many expressed appreciation for our detailed analysis and collaborative approach. Two sandbox vendors requested full anonymity, which we honored in Table XI.

Notably, views on our findings differed across the security community. While most vendors treated them as product vulnerabilities, one organization classified our report as strategic threat intelligence. Another concluded that the weaknesses we identified do not directly impact the confidentiality, integrity, or availability of customer data. From their perspective, our findings reflect performance gaps in the security product rather than vulnerabilities requiring immediate action.

These differing assessments were compounded by operational constraints that further complicate remediation. In particular, one issue the vendors brought to our attention was sandbox IP reputation. Network communication is often essential to dynamic analysis as C2 traffic can yield valuable threat intelligence, and some malware only activates in networked environments. However, this same communication can trigger abusive behavior. For example, the Mirai malware family conducts brute-force attacks against external hosts [65], which can result in the blocklisting of sandbox IPs. Because these IPs are typically leased or assigned by registries, vendors face challenges in rotating or replenishing them. This makes it difficult to diversify sandbox infrastructure without increasing exposure to fingerprint-based evasion. Despite their generally low reputation due to widespread abuse [66], VPNs could offer a practical tradeoff, enabling IP diversity while reducing the risk of long-term reputation damage.

C. Ethical recommendations

Our survey of prior studies that upload binaries ([22], [67]) and URLs ([68]) to VirusTotal revealed high variation in binary submission volume and inconsistent disclosure of research activities. Ethical standards [69] safeguarding humans and systems must always be followed. When only the effects of systems are being measured (like in this study), *disclosure of research activities* should always be readily available to inform humans. This includes embedding an opt-out link in the binary, all network communications, and all infrastructure when possible. Studies should have a clear end date, and infrastructure should be taken down after this date.

Special care must be taken when submitting *lookalike* artifacts (e.g., new polymorphic samples of real-world malware families [22] or of specific brand names [68]). These can cause targeted detections in commercial systems, leading to more widespread human intervention and can introduce artificial data into future research studies. In these cases, it is critical to clearly share the nature of the experiment with the TI ecosystem. When live infrastructure is involved, we recommend taking lookalike measurements quickly and then immediately replacing lookalike content with a disclosure page and opt-out URL. When possible, researchers should opt for ‘private’ analyses to minimize sharing.

Finally, researchers should take reasonable steps to minimize compute cost induced on vendors. While vendor-imposed API rate limiting provides guidance, the impact of downstream vendors and longitudinal studies should also be considered. We found that submissions to multiscanner websites generate between 15-100 total executions when downstream vendors are included. In cases where many binaries are being studied over a long period of time, cumulating in millions of total probes [15]), we recommend using backoff techniques to minimize the number of probes conducted when no changes are observed for an increasing number of days.

D. Limitations

First, although we took steps to include a representative sample of vendors, our analysis is limited to free sandbox providers. This constraint may affect the external validity of our findings, particularly when compared to premium sandbox services. Unlike free sandboxes, premium offerings typically execute binaries privately, which could reduce the extent of inter-vendor sharing observed. Moreover, the system and network fingerprints of premium sandboxes may differ from those of the free ones, potentially reflecting greater heterogeneity. We leave the exploration of these differences as a valuable direction for future work.

Beyond these premium vendors, there exists a long tail of vendors offering free sandbox, TI, and malicious file inspection services that were not directly evaluated in our study. Exhaustively cataloging all such vendors is not the primary objective of our work. Rather, our goal was to draw inductive observations about the broader ecosystem based on a representative set of 30 vendors.

Another limitation centers around the temporal stability and completeness of the fingerprints we collected. Signatures may change over time as vendors update their infrastructure or practices, meaning that longitudinal analyses would require periodically refreshing the features used for vendor labeling. Thus, our results should be viewed as a snapshot of the ecosystem that highlights its underlying processes, strengths, and weaknesses. Similarly, while the small number of features used in our experiments may lower clustering performance, it suffices for highlighting the ecosystem's processes.

Finally, our study does not address the human and organizational elements of disruption. In practice, takedown workflows rely on human reporting and analyst intervention, and registrars differ widely in their willingness to cooperate with anti-abuse efforts—an issue highlighted in recent litigation [70]. Use of these registrars would likely have reduced the takedown prevalence seen in our study.

VI. Conclusion

We present a novel approach for gaining a deeper understanding of how TI is disseminated among the ecosystem. Through an extensive empirical analysis, we showcase several important factors, including the fragility of an ecosystem that is currently highly dependent on a few key players. We observe clear evidence that adversaries exploit known weaknesses in sandbox infrastructures and sharing paths, and provide actionable recommendations to improve the resiliency of TI systems.

Acknowledgment

The authors thank the anonymous reviewers for their detailed feedback that helped improve the paper. Likewise, we thank Carrie Gates, Michael Silverman, and Teresa Walsh from the FS-ISAC community for their input regarding various aspects of this work. We also express our profound gratitude to the members of TI community who engaged with us during the responsible disclosure process. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001123C0035. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the U.S. Department of Defense (DoD).

References

[1] A. Zrahia, "Threat intelligence sharing between cybersecurity vendors: Network, dyadic, and agent views," *Journal of Cybersecurity*, 2018.

[2] A. Dhapte, "Threat intelligence market – forecast to 2035," Market Research Future (MRFR), Tech. Rep., Oct. 2025, 200 pages. [Online]. Available: https://www.marketresearchfuture.com/reports/threat-intelligence-market-4110.

[3] S. De Smale, R. van Dijk, X. Bouwman, J. van der Ham, and M. van Eeten, "No one drinks from the firehose: How organizations filter and prioritize vulnerability information," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2023.

[4] V. G. Li, M. Dunn, P. Pearce, D. McCoy, G. M. Voelker, and S. Savage, "Reading the tea leaves: A comparative analysis of threat intelligence," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2019.

[5] M. Kührer, C. Rossow, and T. Holz, "Paint it black: Evaluating the effectiveness of malware blacklists," in *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, The International Symposium on Research in Attacks, Intrusions, and Defenses, 2014.

[6] X. Bouwman, H. Griffioen, J. Egbers, C. Doerr, B. Klievink, and M. Van Eeten, "A different cup of TI? the added value of commercial threat intelligence," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2020.

[7] M. Wang, L. Yang, and W. Lou, "A comprehensive dynamic quality assessment method for cyber threat intelligence," in *Proceedings of the IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, IEEE, 2022.

[8] B. Tostes, L. Ventura, E. Lovat, M. Martins, and D. Menasche, "Learning when to say goodbye: What should be the shelf life of an indicator of compromise?" In *Proceedings of the International Conference on Cyber Security and Resilience (CSR)*, IEEE, 2023.

[9] C. Sauerwein, C. Sillaber, and R. Breu, "Shadow cyber threat intelligence and its use in information security and risk management processes," in *Proceedings of the Multikonferenz Wirtschaftsinformatik (MKWI)*, 2018.

[10] X. Bouwman, V. Le Pochat, P. Foremski, *et al.*, "Helping hands: Measuring the impact of a large threat intelligence sharing community," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2022.

[11] A. Zibak and A. Simpson, "Cyber threat information sharing: Perceived benefits and barriers," in *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, ACM, 2019.

[12] B. Kacsmar, K. Tilbury, F. Kerschbaum, and M. Mazmudar, *Caring about sharing: User perceptions of multiparty data sharing*, USENIX Security Symposium Talk, 2021.

[13] D. Fischer, M. Werchan, C. Sauerwein, and D. Stelzer, "An exploratory study on the use of threat intelligence sharing platforms in germany, austria, and switzerland," in *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, ACM, 2023.

[14] J. Kotsias, A. Ahmad, and R. Scheepers, "Adopting and integrating cyber-threat intelligence in a commercial organisation," *European Journal of Information Systems*, 2023.

[15] S. Zhu, J. Shi, L. Yang, *et al.*, "Measuring and modeling the label dynamics of online Anti-Malware engines," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2020.

[16] S. S. Roy, U. Karanjit, and S. Nilizadeh, "What remains uncaught?: Characterizing sparsely detected malicious urls on twitter," *MadWeb NDSS*, pp. 400–412, 2021.

[17] P. Pawlinski and A. Kompanek, *Evaluating threat intelligence feeds*, https://www.first.org/resources/papers/conf2016, FIRST Technical Colloquium, 2016.

[18] T. Schaberreiter, V. Kupfersberger, K. Rantos, *et al.*, "A quantitative evaluation of trust in the quality of cyber threat intelligence sources," in *Proceedings of the International Conference on Availability, Reliability and Security (ARES)*, ACM, 2019.

[19] A. Zibak, C. Sauerwein, and A. C. Simpson, "Threat intelligence quality dimensions for research and practice," *Digital Threats: Research and Practice*, 2022.

[20] D. Schlette, F. Böhm, M. Caselli, and G. Pernul, "Measuring and visualizing cyber threat intelligence quality," *International Journal of Information Security*, 2021.

[21] B. Jin, E. Kim, H. Lee, E. Bertino, D. Kim, and H. Kim, "Sharing cyber threat intelligence: Does it really help?" In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2024.

[22] K. Lucas, M. Sharif, L. Bauer, M. K. Reiter, and S. Shintre, "Malware makeover: Breaking ml-based static analysis by modifying executable bytes," in *Proceedings of the ACM Asia Conference on Computer and Communications Security (AsiaCCS)*, 2021, pp. 744–758.

[23] Z. Mao, Z. Fang, M. Li, and Y. Fan, "Evaderl: Evading pdf malware classifiers with deep reinforcement learning," *Security and Communication Networks*, vol. 2022, no. 1, p. 7 218 800, 2022.

[24] A. Yokoyama, K. Ishii, R. Tanabe, *et al.*, "Sandprint: Fingerprinting malware sandboxes to provide intelligence for sandbox evasion," in *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2016.

[25] N. Miramirkhani, M. P. Appini, N. Nikiforakis, and M. Polychronakis, "Spotless sandboxes: Evading malware analysis systems using wear-and-tear artifacts," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2017.

[26] A. Küchler, A. Mantovani, Y. Han, L. Bilge, and D. Balzarotti, "Does every second count? time-based evolution of malware behavior in sandboxes," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, Internet Society, 2021.

[27] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah, "Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[28] V. Jain, S. M. M. Alam, S. V. Krishnamurthy, and M. Faloutsos, "C2store: C2 server profiles at your fingertips," in *Proceedings of the ACM on Networking (ACM Netw.)*, ACM, 2023, pp. 1–21.

[29] A. Tundis, S. Ruppert, and M. Muhlhauser, "A feature-driven method for automating the assessment of osint cyber threat sources," *Computers & Security*, 2022.

[30] J. Spooren, T. Vissers, P. Janssen, W. Joosen, and L. Desmet, "Premadoma: An operational solution for DNS registries to prevent malicious domain registrations," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2019, pp. 557–567.
[31] F. Deniz, M. Nabeel, T. Yu, and I. Khalil, "Mantis: Detection of zero-day malicious domains leveraging low reputed hosting infrastructure," *arXiv preprint arXiv:2502.09788*, 2025.
[32] T. Galloway, K. Karakolios, Z. Ma, R. Perdisci, A. Keromytis, and M. Antonakakis, "Practical attacks against DNS reputation systems," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2024.
[33] H. Griffioen, T. Booij, and C. Doerr, "Quality evaluation of cyber threat intelligence feeds," in *Proceedings of the International Conference on Applied Cryptography and Network Security (ACNS)*, Springer, 2020.
[34] A. Klein and I. Kotler, *The Adventures of AV and the Leaky Sandbox*. Jul. 2017. [Online]. Available: https://www.blackhat.com/docs/us-17/thursday/us-17-Kotler-The-Adventures-Of-Av-And-The-Leaky-Sandbox-wp.pdf.
[35] R. T. Snodgrass, S. S. Yao, and C. Collberg, "Tamper detection in audit logs," in *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, 2004, pp. 504–515.
[36] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Transactions on Information and System Security*, vol. 2, no. 2, pp. 159–176, May 1999.
[37] Awesome Malware Analysis: A curated list of awesome malware analysis tools and resources. https://github.com/rshipp/awesome-malware-analysis. Accessed 12-10-2023.
[38] CAIDA, *The CAIDA AS Organizations Dataset*, https://publicdata.caida.org/datasets/as-organizations/, 2023.
[39] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.
[40] S. Ramanathan, J. Mirkovic, and M. Yu, "Blag: Improving the accuracy of blacklists," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2020.
[41] IPRegistry.co, *IP Geolocation and Threat Intelligence*, https://ipregistry.co/.
[42] Sicehice, *About sicehice*, https://www.sicehice.com/about, Accessed: 2024-11-11.
[43] C. Lever, R. Walls, Y. Nadji, D. Dagon, P. McDaniel, and M. Antonakakis, "Domain-z: 28 registrations later measuring the exploitation of residual trust in domains," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2016.
[44] E. Pauley, R. Sheatsley, B. Hoak, Q. Burke, Y. Beugin, and P. McDaniel, "Measuring and mitigating the risk of IP reuse on public clouds," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2022.
[45] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 1996, pp. 226–231.
[46] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 1999, pp. 49–60. DOI: 10.1145/304182.304187. [Online]. Available: https://doi.org/10.1145/304182.304187.
[47] S. Romano, N. X. Vinh, J. Bailey, and K. Verspoor, "Adjusting for chance clustering comparison measures," *Journal of Machine Learning Research*, vol. 17, no. 134, pp. 1–32, 2016.
[48] M. Ziv, L. Izhikevich, K. Ruth, K. Izhikevich, and Z. Durumeric, "Asdb: A system for classifying owners of autonomous systems," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2021, pp. 703–719.
[49] C. Song, P. Royal, and W. Lee, "Impeding automated malware analysis with environment-sensitive malware," in *Proceedings of the USENIX Workshop on Hot Topics in Security (HotSec)*, Bellevue, WA: USENIX Association, 2012.
[50] F. Barr-Smith, X. Ugarte-Pedrero, M. Graziano, R. Spolaor, and I. Martinovic, "Survivalism: Systematic analysis of windows malware living-off-the-land," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2021.
[51] E. Alowaisheq, "Cracking wall of confinement: Understanding and analyzing malicious domain takedowns," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2019.
[52] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.
[53] Nextron Systems, *SUSP_Defense_Evasion_Known_Hostnames_Jun23 - Valhalla Rule Info*, https://valhalla.nextron-systems.com/info/rule/SUSP_Defense_Evasion_Known_Hostnames_Jun23, Accessed: 2025-07-10, 2023.
[54] S. Sebastián and J. Caballero, "Avclass2: Massive malware tag extraction from av labels," in *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*, 2020.
[55] R. J. Joyce, D. Everett, M. Fuchs, E. Raff, and J. Holt, "Claravy: A tool for scalable and accurate malware family labeling," in *Companion Proceedings of the ACM Web Conference (The Web Conference)*, 2025, pp. 277–286.
[56] hackirby, *Skuld stealer*, https://github.com/hackirby/skuld/blob/main/modules/antivm/antivm.go, 2025.
[57] Luna-Grabber, *Luna-grabber*, https://github.com/Luna-Grabber/Luna-Grabber/blob/main/Luna-Grabber-main/luna.py, 2025.
[58] 6nz, *Virustotal-vm-blacklist*, https://github.com/6nz/virustotal-vm-blacklist, BSD-3-Clause License, 2025.
[59] CYFIRMA Research Team, "Kematian–stealer: A deep dive into a new information stealer," CYFIRMA, Technical Report, Jul. 2024, Accessed January 11, 2026. [Online]. Available: https://www.cyfirma.com/research/kematian-stealer-a-deep-dive-into-a-new-information-stealer/.
[60] H. Aghakhani, F. Gritti, F. Mecca, *et al.*, "When malware is packin' heat; limits of machine learning classifiers based on static analysis features," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2020.
[61] ANSSI / CERT-FR, *Anssi policy for sharing and handling its operational information*, French and English, Version française Version 1 (16 Nov 2022), Agence nationale de la sécurité des systèmes d'information (ANSSI), 2022. [Online]. Available: https://cert.ssi.gouv.fr/csirt/sharing-policy/.
[62] R. A. Petrunić, "Honeytokens as active defense," in *Proceedings of the 38th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2015, pp. 1313–1317.
[63] K. Van Liebergen, J. Caballero, P. Kotzias, and C. Gates, "A deep dive into the virustotal file feed," in *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, 2023.
[64] J. Wang, L. Wang, F. Dong, and H. Wang, "Re-measuring the label dynamics of online anti-malware engines from millions of samples," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2023.
[65] M. Antonakakis, T. April, M. Bailey, *et al.*, "Understanding the mirai botnet," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2017, pp. 1093–1110.
[66] R. Ramesh, "Investigating the vpn ecosystem through the lens of security, privacy, and usability," Ph.D. dissertation, PhD thesis, 2023.
[67] K. Eykholt, T. Lee, D. Schales, J. Jang, and I. Molloy, "URET: Universal robustness evaluation toolkit (for evasion)," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2023, pp. 3817–3833.
[68] P. Peng, L. Yang, L. Song, and G. Wang, "Opening the blackbox of virustotal: Analyzing online phishing scan engines," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, 2019, pp. 478–485.
[69] M. D. Bailey, D. Dittrich, E. Kenneally, and D. Maughan, "The menlo report," *IEEE Security & Privacy*, vol. 10, no. 2, pp. 71–75, 2012. DOI: 10.1109/MSP.2012.52. [Online]. Available: https://doi.org/10.1109/MSP.2012.52.
[70] B. Krebs, "Sued by meta, freenom halts domain registrations," *Krebs on Security*, Mar. 7, 2023, Accessed: 2025-11-13. [Online]. Available: https://krebsonsecurity.com/2023/03/sued-by-meta-freenom-halts-domain-registrations/.
[71] T. Kohno, Y. Acar, and W. Loh, "Ethical frameworks and computer security trolley problems: Foundations for conversations," in *Proceedings of the USENIX Security Symposium (USENIX Security)*, 2023.
[72] E. Pauley and P. McDaniel, "Understanding the ethical frameworks of internet measurement studies," in *Proceedings of the 2nd International Workshop on Ethics in Computer Security*, 2023.
[73] F. Hantke, S. Roth, R. Mrowczynski, C. Utz, and B. Stock, "Where are the red lines? towards ethical server-side scans in security and privacy research," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2024, pp. 4405–4423.

Ethical Considerations

Throughout the course of the study, we took several steps to ensure that we minimized harm by following ethical norms [71] and security research guidance [69], [72], [73]. In Section V, we highlight some of the lessons learned from our study and propose guidelines for future researchers actively probing production TI systems.

Human deception. Our study is designed to measure *automated systems*, not human analysts. To minimize collateral human deception, we included an opt-out link within all binaries and network requests. Furthermore, we implemented system-based notifications for human analysts by displaying messages on the screen. This approach ensures that human analysts were informed option to opt out, respecting the autonomy and privacy of potential participants. Furthermore, we initiated a responsible disclosure process after the experiments (detailed in Section V-B). We received two opt-out requests during the responsible disclosure phase. In both cases, the vendors agreed to be included in the study anonymously.

Compute cost. Our tracking mechanism involves submitting a self-replicating binary to vendors (i.e., each time the binary executes, a new binary is dropped on the host). Given the presence of sharing in the ecosystem, we must ensure that submitting our binary does not introduce a 'snowball effect' where self-replications increase exponentially as more and more binaries are shared. With this in mind, we aggressively limit our binaries' replication depth to a maximum of five layers. Our tracking mechanism allows us to estimate that the 90 binaries we submitted resulted in 1,827 executions. Each execution emitted only one HTTP/DNS request, minimizing network usage. Finally, when probing OSINT we probed public blocklists and vendor query interfaces for IoC detections no more than 200 times per hour.

Privacy. To protect the privacy of the vendors and to prevent unnecessary leakage of sensitive information, we anonymize the name of each vendor when presenting the results. Additionally, to minimize the impact of our probes, our system collects a minimal set of sandbox features and applies privacy-preserving hashing before any transmission. This limits potential exposure of potentially private information from the sandbox (e.g., machine names or user configuration data [24]) that external parties could abuse. Furthermore, our all binaries are defanged—they do not store or transmit any behaviors they may record.

Responsible disclosure. We disclosed our findings and provided recommendations (including signatures that could help detect attempts to evade execution on their platforms) to all vendors we directly uploaded binaries to. We discuss the insights gleaned from this process in Section V-B.

Appendix

Our fingerprinting methodology is based on the approach of SandPrint [24], which identifies unique features of sandbox environments. However, rather than using all 16 features considered in that work, we use a subset of features that maximizes mutual information while being privacy conscious. Our system fingerprint consists of hashed values of this set. To find the subset of features, we designed a binary that collected the same SandPrint features (along with the information gain) shown in Table X. To ensure no cross-contamination, that binary used domains unrelated to those in the final study. That binary was submitted to a small set of vendors during a pilot study. We clustered the emissions using the methodology in Section III.

Since each execution may produce multiple emissions and IDs are unique, emissions are first grouped by execution ID. We observed 532 executions and their execution IDs in this pilot. Executions were grouped using single-link hierarchical clustering on 16 sandbox features. To compute the distance threshold parameter of the hierarchical clustering algorithm, distance thresholds between similarities of 0.00 to 1.00 in increments of 0.05 were attempted sequentially; the distance threshold which correctly clustered the greatest number of hand-labeled clusters for a random subset of the dataset was selected and passed into the clustering a final time with the full execution dataset to source ground truth.

TABLE X: **Features from SandPrint [24] used in pilot and their associated information gain. The 🔒 symbol denotes features we viewed as potentially sensitive and choose not to consider as part of the hashed fingerprint.**

Field	Description	Info Gain	Sensitive
`host_name`	Host Name	5.98	🔒
`install_date`	Install Date	5.84	
`processor`	Processor	5.69	
`win_prod_id`	Windows Product ID	5.65	🔒
`mac`	MAC Address	5.32	🔒
`owner_name`	Owner Name	5.21	🔒
`sample_path`	Sample Path	5.20	🔒
`ip`	IP Address	5.11	
`sample_name`	Sample Name	4.97	
`default_gateway`	Default Gateway	4.77	
`sys_manufac`	System Manufacturer	4.28	
`disk_space`	Disk Space	4.21	
`dns`	DNS	4.14	
`ram`	RAM	4.00	
`os_information`	OS Information	2.30	
`org_name`	Organization Name	0.95	🔒

We analyzed each cluster to find the smallest subset of features with the largest amount of mutual information so that we can disambiguate vendors in a length-constrained subdomain. We compute mutual information between a feature and the cluster variable as $I(F_1; C) = H(F_1) + H(C) - H(F_1, C)$, where $H(\cdot)$ is the entropy of a random variable and $H(\cdot, \cdot)$ is the joint entropy of two random variables. We found the maximum by searching over all $2^{16} - 1$ nonempty subsets of the 16 features. For each subset, we computed the mutual information gain from the subset of features. To validate, we ran the single-link hierarchical clustering algorithm to check similarity between ground truth (clustering with all 16 features) and clustering with just information contained within the subset of features and performed pairwise comparisons of executions in the ground truth labeling and the subset labeling. We found that the `install_date`, `ram`, and `sys_manufac` features achieved the greatest cluster similarity for 3 features: in over 99% of pairs of executions, we achieved the same labeling between the two clustering algorithms. This is due to the fact that among all 560 possible triplets, the mutual information for the selected features tied for first place with another 160 subsets. Of those, 39 subsets did not include any feature marked as sensitive in Table X.

Satisfied with the high mutual information for those 3 features, We then built a new version of our Rocket that profiles only those features. To validate that the fingerprinting

TABLE XI: **Vendors considered in our study.** After disclosing our study details to the vendors, two requested anonymity, which we honor with the INT-category-n label.

	Vendor Name	Locale	Measured	Execs.	Additional Notes
Antivirus	Avast	CZ	✓	✓	
	ClamAV	US	✓	✓	
	Dr.Web	RU	✓	✓	
	FortiGuard Labs	US	✓	✓	
	Kaspersky	RU	✓	✓	
	Sophos	UK	✓	✓	
	Symantec	US	✓	✓	
	Avira	DE	✓	✗	
	BitDefender	RO	✓	✗	
	Comodo	UK/US	✓	✗	
	F-Secure	FI	✓	✗	
	GData	DE	✓	✓	
	Microsoft Defender	US	✓	✗	
	SonicWall	US	✓	✗	
	WebRoot	US	✓	✗	
	Adaware	CA	✗	✗	*Service unavailable*
	EMSISoft	NZ	✗	✗	*Submission error*
Sandboxes	Any.Run	UA	✓	✓	
	Hybrid Analysis	DE/US	✓	✓	
	Intezer	US	✓	✓	
	`INT-SB-1`	-	✓	✓	
	Pikker	EE	✓	✓	
	Filescan	US	✓	✓	
	`INT-SB-D`	-	✓	✓	*Requested redaction*
	CAPEv2	-	✗	✗	*Service unavailable*
	IOBit	CN	✗	✗	*Service unavailable*
	Amn Pardaz	IR	✗	✗	*Service unavailable*
	AntiScan	BZ	✗	✗	*Service unavailable*
	SecureHunter	US	✗	✗	*PDFs only*
	IRIS-H	-	✗	✗	*PDFs only*
	SecondWrite	US	✗	✗	*Requires vetted trial*
	VMRay	US	✗	✗	*Paid license*
	MalBeacon	US	✗	✗	*Anti-probing policy*
TI Platforms	AlienVault OTX	US	✓	✓	
	MWDB	PL	✓	✓	
	MalwareBazaar	CZ/NL	✓	✓	
	VirusTotal	US	✓	✓	
	US-CERT	US	✓	✓	
	`INT-TI-G`	-	✓	✓	*Requested redaction*
	OPSWAT	US	✓	✗	

and watermark trail functionalities worked correctly, we tested the rocket in a control with different sandboxes installed locally, where we manually copied the satellites dropped in one sandbox and executed them in another and then verified the trail worked as expected.

Most system fingerprints are easy to change or spoof, and a vendor who frequently alters system and network fingerprints could readily evade fingerprinting. Our experiment identified one vendor that changes its OS install date for every execution, but keeps all other fingerprints the same. Egress IP addresses are more difficult to spoof, but our experiments discovered vendors still manipulating IP addresses using proxies. Sandprint [24] notes that several features excluded from our measurement are trivial to change, for example, screen resolution or size.

A. Vendor Selection

We categorize inaccessibility to some of the vendors into six groups. First, for five of the vendors, their online submission portal or website was not accessible. For example *CAPEv2* had a datacenter incident that shut down their operation and did not recover in time for us to consider. Second, some vendors' web portal only accept a specific type of files because their online engine is limited for public use. Third, one of the vendors offers a free trial for their analysis service but required us to contact sales and set up a demo and ensure that we are a legitimate entity before providing us with a free trial. Since we do not intend to purchase the service, we opted not to waste the human resource by interfacing with sales representatives. Fourth, some vendors have explicit anti-probing policies such as MalBeacon's. We excluded those vendors from our study. Fifth, EMSISoft had an issue with their submission form that did not accept or upload the binary file correctly. We tried multiple times with different browsers, operating systems, and IP addresses but were not successful. Lastly, some vendors require a paid license. For the remaining vendors, we were able to submit a binary, but the binaries were not dynamically analyzed or may have been analyzed in isolation, and our *Observer* could not record the emissions.

B. Automatic labeling of unknown vendors

The clustering algorithm used to label external vendors is given in Algorithm 1.

Algorithm 1 Cluster Unknown Vendors

```
1: Input: Set of fingerprints $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$
2: Output: Set of hard clusters $\mathcal{C}$
3: $\mathcal{C} \leftarrow$ {Instantiate a cluster for each $\mathbf{x}_{1\ldots n}$}
4: $\mathcal{D}[c_i, c_j] \leftarrow$ Distance between clusters $(c_i, c_j)$
5: while minimum distance between clusters $\leq \delta$ do
6:     $c_i, c_j \leftarrow \arg\min_{c_i, c_j \in \mathcal{C}} \left(\min_{x \in c_i, y \in c_j} \mathcal{D}[x, y]\right)$
7:     Merge $c_i$ and $c_j$ into a single cluster, update $\mathcal{C}$
8:     Recompute distances $\mathcal{D}$ for the new set of clusters
9: end while
10: Return: $\mathcal{C}$
```