Propagating Threat Scores With a TLS Ecosystem Graph Model Derived by Active Measurements

Markus Sosnowski, Patrick Sattler, Johannes Zirngibl, Tim Betzer, and Georg Carle
Technical University of Munich, Germany
TUM School of Computation, Information and Technology; Department of Computer Engineering
Chair of Network Architectures and Services
{sosnowski, sattler, zirngibl, betzer, carle}@net.in.tum.de

***Abstract*—The Internet is shaped by independent actors and heterogeneous deployments. With the wide adoption of Transport Layer Security (TLS), a whole ecosystem of intertwined entities emerged. Acquiring a comprehensive view allows searching for previously unknown malicious entities and providing valuable cyber-threat intelligence. Actively collected Internet-wide Domain Name System (DNS) and TLS meta-data can provide the basis for such large-scale analyses. However, in order to efficiently navigate the vast volumes of data, an effective methodology is required. This work proposes a graph model of the TLS ecosystem that utilizes the relationships between servers, domains, and certificates. A Probabilistic Threat Propagation (PTP) algorithm is then used to propagate a threat score from existing blocklists to related nodes. We conducted a one-year-long measurement study of 13 monthly active Internet-wide DNS and TLS measurements to evaluate the methodology. The latest measurement found four highly suspicious clusters among the nodes with high threat scores. External threat intelligence services were used to confirm a high rate of maliciousness in the rest of the newly found servers. With the help of optimized thresholds, we identified 557 domains and 11 IP addresses throughout the last year before they were known to be malicious. Up to 40% of the identified nodes appeared on average three months later on the input blocklist. This work proposes a versatile graph model to analyze the TLS ecosystem and a PTP analysis to help security researchers focus on suspicious subsets of the Internet when searching for unknown threats.**

***Index Terms*—TLS, DNS, Probabilistic Threat Propagation, Labeled Property Graph, Blocklists, Internet-wide Measurements**

I. Introduction

The widespread adoption of Transport Layer Security (TLS) [1] has led to the development of an interconnected ecosystem that, in conjunction with the Domain Name System (DNS) and the Web Public Key Infrastructure (WebPKI), underpins the majority of the current Internet.

Notably, active measurements of this TLS ecosystem can enable novel possibilities to identify malicious activity because interesting meta-data about servers, their configuration, or the actor behind the deployments (*e.g.*, Refs. [2, 3, 4]) can be collected, especially considering the increased usage of TLS by cyber-criminals [5]. Blocklists like the abuse.ch SSLBL [6] (listing certificates used by botnet Command and Control (C2) servers) indicate that certificates can reveal relations among C2 server IP addresses. Moreover, domains might resolve to these IP addresses and could be embedded in other certificates. Leveraging such relations can provide a detailed view of malicious infrastructure and offer valuable cyber-threat intelligence. However, a comprehensive model and efficient approach for navigating the vast amount of collectible data are necessary to provide this information.

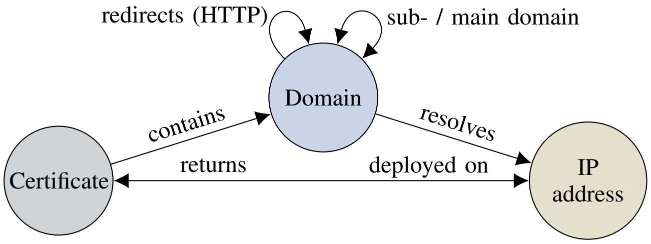


Figure 1: Internet-wide TLS Ecosystem Graph (ITEG) Schema

This work investigates a graph-based model of active DNS and TLS measurement data to leverage the extracted relations and transmit a threat score across entities using a Probabilistic Threat Propagation (PTP) [7] algorithm to find new, potentially malicious servers.

This paper makes the following contribution:

i) A versatile graph model of the TLS ecosystem built around the deliberate actions of an actor controlling a domain, IP address, or certificate; *cf.*, Figure 1.
ii) The application of a message-based implementation of PTP to propagate a threat score throughout the graph using blocklists as input.
iii) A one-year-long measurement study covering 13 monthly Internet-wide DNS and TLS measurements. We analyzed newly found domains and IP addresses with a high threat score via manual inspection and external threat intelligence services. Additionally, we evaluated whether they would appear on the respective blocklists over time.
iv) Published results, scripts, and example graph [8].

II. Methodology

The TLS ecosystem is an interplay of the DNS [9], X.509 Public Key Infrastructures (PKIs) [10], and the applications using TLS (*e.g.*, HTTPS) to provide a level of trust in our communication over the Internet. This work proposes modeling actively collected DNS and TLS meta-data as a

978-3-903176-64-5 © 2024 IFIP

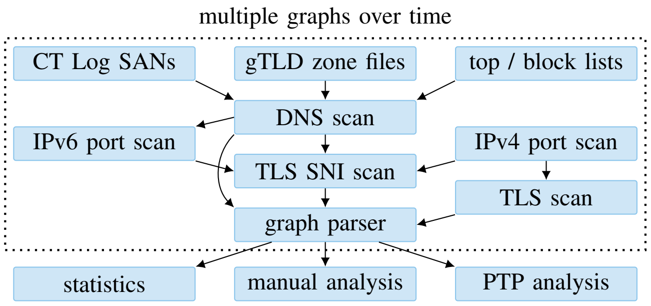


Figure 2: Data Collection and Graph Processing Pipeline

Labeled Property Graph to streamline processing, simplify inspection, and benefit from well-known graph algorithms, ultimately aiding security-relevant use cases.

Figure 2 describes our data collection and processing pipeline. It worked as follows: we started with domain lists (downloaded from external sources and extracted from the Certificate Transparency (CT) log) covering a large portion of the Internet and resolved each domain to one or more IP addresses (IPv4 and IPv6). Afterward, we conducted Internet-wide TLS scans of the entire IPv4 address space and all the (IP address, domain) tuples we collected during the DNS scan. In the second case, we used the domains as Server Name Indications (SNIs) to collect certificates bound to specific names. The TLS scans targeted TCP port 443, and we stored the received Hypertext Transfer Protocol (HTTP) headers. We leave an extension with QUIC scans to future work. As Refs. [11, 12] point out, QUIC is currently mostly used by Content Delivery Networks (CDNs) and hypergiants.

An Apache Spark [13] application transformed the DNS and TLS scan results into a property graph and performed a PTP analysis on the data. The graph and the calculated PTP scores were the basis for our later analyses.

The data collection is further elaborated in Section II-B, while Section II-C explains modeling this data as ITEG, and Section II-D proposes a message-based PTP on the ITEG.

A. *Why Modeling a Property Graph?*

Modeling real-world data as graphs is intuitive, especially if the data source is already a network like the Internet. The idea is not new; *e.g.*, the RDF [14] tries to model arbitrary resources on the Internet as a single Internet-wide graph.

A *graph* is an abstraction and simplification that expresses connected data using only *nodes* (or *vertices*) and *edges* [15]. The most popular graph model variant is a *Property Graph*, where nodes and edges can contain properties (expressed as key-value pairs), edges are named and directed, and they always have a start and end node (*cf.*, Ref. [15]). A *Labeled Property Graph* additionally contains labels attached to nodes that distinguish different types. We selected a Labeled Property Graph for our modeling because it is a simple and intuitive representation powerful enough to express the complex data we collect through our active measurements.

We propose a graph-based model of the TLS ecosystem. However, this does not mean the data has to be stored in a graph database. For example, we implemented our message passing in Section II-D using only a series of SQL joins and aggregations on Apache Spark [13] Dataframes because it allowed us to optimize the algorithm better. Conversely, graph databases can sometimes improve performance because they were optimized for typical graph problems; *e.g.*, Abedrabbo *et al.* [16] observed a "significant degregation of performance" at depths 4 and 5 when comparing graph traversals using a relational and graph database. Additionally, graph-based libraries and databases can provide a convenient interface to interact and visualize data, *e.g.*, we used the Graph Frames [17] library to compute the node degrees in Section III-A and Neo4J [18] to visualize and manually inspect our findings.

B. *Internet-wide DNS and TLS Scans*

We collected data about the TLS ecosystem with the help of DNS and TLS measurements. Input for the DNS scans were:

i) domains from gTLD zone files obtained from the Centralized Zone Data Service (CZDS), including *.com*, *.net*, and *.org*;

ii) top and blocklists, *i.a.*, Majestic [19], Umbrella [20], Chrome UX Report [21], Chromium HSTS Preload [22], Cloudflare Radar [23], and Openphish [24]; and

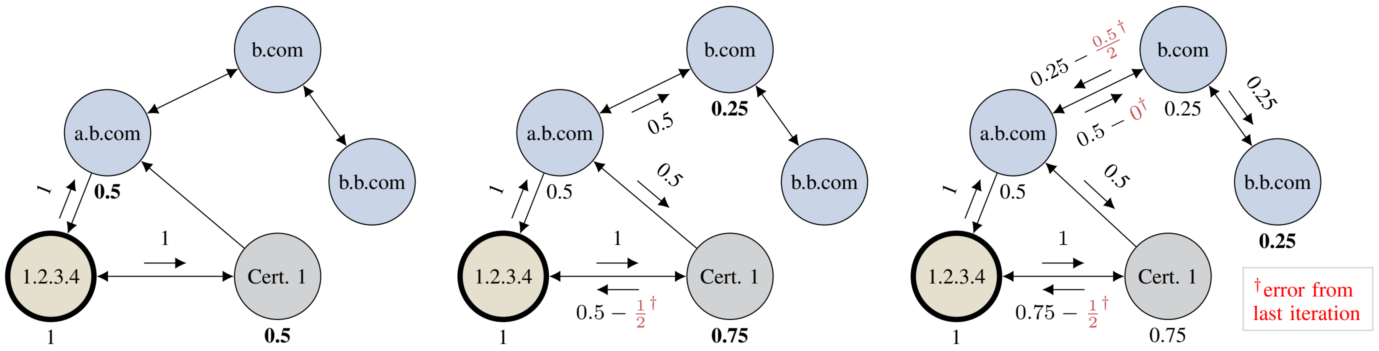
iii) domains we extracted from the Subject Alternative Name (SAN) extension in CT log certificates.

We issued A and AAAA queries using MassDNS [25] and a local Unbound [26] resolver. For our TLS measurements, we focused on TLS-capable addresses on port 443, the standard HTTPS port, expecting a high rate of deployed servers supporting TLS. For IPv4, we used ZMap [27] on the complete address space to identify servers with an open port 443. Afterward, we used the Goscanner [28] to conduct a full TLS handshake without a SNI. We relied on the previous A and AAAA DNS resolutions to collect further TLS data bound to specific domains. We scanned each resolved IPv6 address for an open port 443 using ZMapv6 [29]. Afterwards, we filtered all (domain, IP address)-pairs for targets with an open port 443 based on our previous ZMap(v6) scans and established a TLS connection to all targets with the respective domain as SNI. Selecting targets with an open port reduced the scanning overhead and network load. We adhered to the ethical considerations in the appendix.

C. *An Internet-wide TLS Ecosystem Graph (ITEG) Model*

This work proposes to model the actively collected data from the TLS ecosystem as a single Internet-wide graph. This section describes the model and the reasons for its design.

Essentially, the TLS ecosystem links resources (identified by domains) to physical locations (IP addresses) and ensures, with the help of certificates, that only servers eligible to serve a particular resource do so. In other words, it creates a cyclic relation between a domain, certificate, and IP address, illustrated in Figure 1. A domain can *resolve* to one or multiple IP addresses via the DNS. When performing a TLS handshake



(a) Iteration 1 (b) Iteration 2 (c) Iteration 3

Figure 3: Example PTP on the ITEG. The IP address "1.2.3.4" is on the input list, hence, has its score fixed to one. To prevent nodes falsely increasing their own scores, PTP subtracts the potential error from the last iteration on bidirectional edges, *e.g.*, between *b.com* and *a.b.com*. Scores of zero are hidden.

with the resolved IP address, the server *returns* a certificate to authorize its response. The certificate *contains* one or multiple domains with the SAN [10] extension. Domains are hierarchical per-design expressed in the *subdomain* and *main domain* relation (*e.g.*, `www.example.org` has an edge to `example.org`). Because we performed our scans on port 443, we received several *redirects* via the HTTP to different domains (*e.g.*, from `example.com` to `example.org`). An important design criterion was that each edge should reflect an actor's intent who is controlling the node. For example, the owner of a domain controls which IP addresses are resolved; however, the owner of an IP address cannot control which domains point to the address. Similarly, the server behind an IP address controls which certificates it returns, and the creator of a certificate chooses the included domains and on which servers the private key is deployed. We decided not to distinguish between valid or self-signed certificates to simplify the model and because the graph already contains this information to a certain degree, as discussed in Section V-3.

In summary, the TLS ecosystem can be modeled with an Internet-wide Labeled Property Graph. Relations in the ITEG express a deliberate action of the actor controlling an entity.

D. Probabilistic Threat Propagation (PTP) on the ITEG

This work leverages the graph structure of the TLS ecosystem to propagate threat indications between connected nodes and find new, potentially malicious nodes given a set of input hints. We used the PTP algorithm developed by Carter *et al.* [7] to achieve this propagation. They originally designed their approach to propagate a threat score among a graph derived from IP addresses and domains a web proxy server observed. However, we will show that it can be also used on the ITEG.

The directions in the ITEG express a deliberate action of someone controlling a node; therefore, we can use the edges to propagate a score and find other relevant nodes. We decided to propagate scores in reversed graph direction. The intuition of this decision was that nodes deliberately pointing to a known malicious node might also be malicious, but the contrary can not be assumed. For example, a benign actor cannot prevent malicious domains from being resolved to his benign IP addresses. Although, domains resolving to a known malicious IP address might be misconfigured, compromised, or even belong to the respective malicious actor. The other ITEG relations follow the same intuition.

Algorithm 1: Message-based Approximate PTP

```
Input: The $ITEG$ (without loops), a blocklist as $Input$,
       and a convergence parameter $\varepsilon$
repeat
    for $(src, dst) \in ITEG.edges$ do
        $message \leftarrow dst[score]$
        $prev \leftarrow$ score send over edge $(dst, src)$ in the
          previous iteration
        if $\exists\, prev$ and $dst \notin Input$ then
            // remove error caused by $prev$
            $message \leftarrow message - \frac{prev}{dst[outDegree]}$
        send $message$ to $e.src$
    for $n \in ITEG.nodes$ do
        $M \leftarrow$ messages received by $n$
        $n[score] \leftarrow \begin{cases} 1 & \text{, if } n \in Input \\ \frac{1}{|M|}\sum_{m \in M} m & \text{, otherwise} \end{cases}$
until $no\ score\ changed\ more\ than\ \varepsilon$
```

We implemented the *Approximate Inference* ($O(n)$) of the PTP algorithm [7] because the *Exact Inference* ($O(n^2)$) would not scale to our large graph. We realized it as a message-passing algorithm, described in Algorithm 1. Our PTP implementation runs for multiple iterations until convergence. We detected a convergence if neither node changed its score more than an input parameter ε. Additional inputs are the ITEG without loops (edges from and to the same node) and a blocklist from which the scores should spread (all nodes on the blocklist will have their scores fixed to one throughout all iterations). Some values were pre-computed to save time, *e.g.*, the *outDegree* of a node. For each iteration, a node would send its score to all nodes pointing to it; in the case of bidirectional edges, we subtracted the portion of the score that was directly caused by the destination in the previous iteration, reducing the error created by nodes falsely increasing their own score. Limiting this error is a central design aspect of PTP (more details in Ref. [7]). However, when using the *Approximate Inference*, cycles in the graph can still cause scores to flow back to the originating node. The approximation is the reason we need ε because the graph does not converge to a final state, the error is only reduced in each iteration. However, Carter *et al.* [7] argue this error is negligible. Then, we computed the average of received scores and assigned it as the new node score. Finally, nodes with high scores can be analyzed.

Figure 3 shows an example run over three iterations:

1) Initially, the IP address *1.2.3.4* is placed on a blocklist, and thus, has a fixed score of one. In the first iteration, this score is passed on to all neighbors. The recipients *a.b.com* and *Cert. 1* will also receive a score of zero from *b.com* and *a.b.com*, respectively. We send scores in reversed edge direction; hence, *a.b.com* receives no score from *Cert. 1*. Then, the new scores of *a.b.com* and *Cert. 1* are set to *0.5*, the average of received messages.
2) In the second iteration, *Cert. 1* will increase its score to 0.75, as *a.b.com* now sends a score of 0.5 instead of zero. Similarly, *b.com* will increase its score to 0.25. We ignore messages sent to *1.2.3.4* because its score is fixed to one.
3) In the third iteration, we observe an interesting effect between *a.b.com* and *b.com* due to the bidirectional edge. Without the PTP error correction, the 0.25 score from *b.com* would flow back to *a.b.com*, falsely increasing the score in every iteration. The *b.com* node has an outgoing degree of two, hence the portion received from *a.b.com* is $\frac{prev}{outDegree} = \frac{0.5}{2} = 0.25$, resulting in a score of zero sent to *a.b.com* after the error correction. In the example, the scores will remain stable after the third iteration. However, the third iteration also shows the downside of the *Approximate Inference*, as we can see a score of 0.25 falsely flowing back to *1.2.3.4* via the cycle over *a.b.com* and *Cert. 1*. Although, it is irrelevant in the example since *1.2.3.4*'s score is fixed.

In conclusion, a message-based PTP allows the propagation of a threat score across the ITEG such that new, potentially malicious IP addresses, domains, and certificates can be found. Existing threat intelligence serves as algorithm input.

E. Potential Optimization

As we developed our methodology, we used top lists as input due to their smaller size, allowing for faster development. However, we soon realized that our methodology heavily relied on collecting critical pieces of information in a larger puzzle, which was impossible with a small input. This means that the completeness of the graph is very crucial to the efficacy of our approach, and the more data available, the higher the chances of discovering something interesting. Nevertheless, compared to all nodes, only a few blocked nodes exist. Since the PTP algorithm is based on locality, only nodes close to blocked ones can achieve a high score. Thus, we optimized the implementation significantly by considering only nodes that received a score greater than zero, enabling us to run each iteration in minutes instead of hours. We ran the algorithm separately for each blocklist, which enabled us to retrace how concrete scores were created. However, merging the input blocklists into a single list, could save processing resources. Nonetheless, we discovered in Section III-D that the blocklists behaved differently, and blocklist-type specific thresholds provided the best detection results. Hence, knowing the type of blocklist helps interpreting the results. Future work may involve exploring more advanced models combining numerous blocklists to provide better threat intelligence.

III. Evaluation

With the help of the methods described in the previous section, we performed 13 monthly Internet-wide DNS and TLS measurements starting January 1, 2023. This data was the basis of 13 ITEGs, which modeled the TLS ecosystem during the respective month. For each ITEG, it took approximately one week to finish scanning, parsing, and calculating the PTP scores. We could have created at most one graph every week, but we decided to do it once a month to minimize computational and storage costs. However, this could mean we may have missed some findings.

Before we present our results, we want to provide an explanation to help interpret the results. Our methodology allowed us to find numerous suspicious domains, IP addresses, and certificates. However, it is important to note that there is no definitive way for us to determine if a suspicious entity is actually malicious or not. To gain insights into the nature of these suspicious entities we will be using external threat intelligence services to check whether the entities also attracted the attention of other actors searching for threats on the Internet; but again, their output is usually only an indicator as well, and it remains unclear whether a domain or IP address is truly malicious or harmless. Nevertheless, by highlighting the overlap between our findings and the data known to these services, we aim to illustrate that our approach can provide valuable indications of malicious activity.

A. Overview

This section provides an overview of collected measurement data, the resulting ITEG, and the blocklist data used as input for the PTP algorithm. We will analyze multiple graphs over time in Section III-D; however, for simplicity and because we mainly analyzed our latest graph in the following sections, we focus this overview on data from February 1, 2024. The statistics are similar for the other time frames.

Table I presents an overview of the measurements necessary to create a single ITEG. It took five days, on average, to complete all scans. We conducted 6.3×10^8 DNS resolutions of domains extracted from CT Log SANs and downloaded from external services. We scanned domains present in both sources only once. The CT Log SANs provided us with the most domains. However, the success rate was higher for the downloaded domains. We used the A and AAAA

Table I: Overview of the measurements necessary to create a single ITEG from Jan. 1, 2024.

	Total Measurements	Success Rate
DNS Scan	6.3×10^8	78.1 %
⌞ CT Log SANs	4.9×10^8	72.1 %
⌞ domain lists	2.8×10^8	95.5 %
TLS IPv4 SNI Scan	5.5×10^8	90.7 %
TLS IPv6 SNI Scan	1.5×10^8	83.1 %
TLS Full IPv4 Scan	5.4×10^7	77.2 %

Table II: Overview of the ITEG from Jan. 1, 2024. Listing the number and distribution of node and edge types.

Type	Amount	Distribution
ITEG nodes	9.0×10^8	
⌞ domains	6.3×10^8	70.0 %
⌞ certificates	1.7×10^8	19.1 %
⌞ IP addresses	9.8×10^7	10.9 %
ITEG edges	3.2×10^9	
⌞ resolves	1.3×10^9	39.6 %
⌞ returns	4.0×10^8	12.6 %
⌞ deployed on	4.0×10^8	12.6 %
⌞ contains	3.6×10^8	11.4 %
⌞ main domain	3.6×10^8	11.2 %
⌞ subdomain	3.6×10^8	11.2 %
⌞ redirects	4.5×10^7	1.4 %

resolutions for our TLS SNI scans over IPv4 and IPv6 only if we previously detected an open port 443 (*cf.*, Section II-B). Interestingly, the success rate over IPv4 was higher than over IPv6. The full IPv4 address space scan had the lowest success rate compared to the other TLS scans. Although we detected open HTTPS ports, we observed TCP resets, TLS protocol errors, or nothing, and the connection ran into a timeout.

We used the actively collected data from Table I to create an ITEG. Table II shows an overview of the number of resulting nodes and edges. In general, the graphs contained fewer entries because we included information only once. The table reveals that domains and their resolutions dominated the graph. We observed almost twice as many certificates as IP addresses, highlighting the importance of SNI scans to get a comprehensive view of the TLS ecosystem. To provide another perspective on the ITEG, we investigated node degrees in Figure 4. Node degrees are a straightforward but effective metric for understanding the structure of a graph. The *inDegree* is defined for each node as the number of incoming edges. The figure shows how the edges in our graph are not evenly distributed and accumulate around a few nodes. In particular, 90% of all edges pointing to IP Addresses accumulate on only 2% (1.3×10^7), meaning that few addresses were responsible for most connections, resulting in a high centralization. We could attribute the top two IP addresses to a domain parking service from GoDaddy, as shown by Zirngibl *et al.* [30]. The rest of the top ten addresses were from Cloudflare and website hosters (*i.e.*, Wix.com and Squarespace).

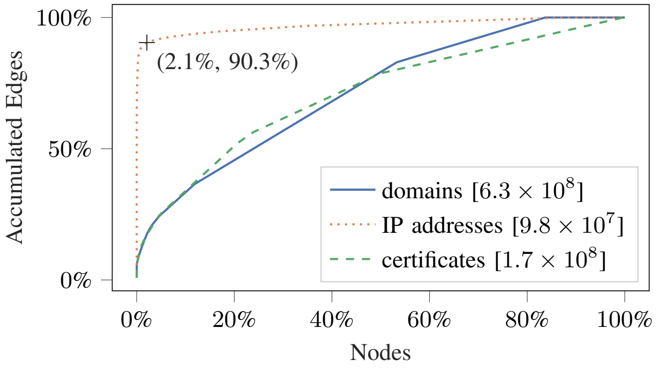


Figure 4: Nodes ordered by the fraction of edges they accumulate. The figure shows a high centralization on IP addresses (90% of the edges congregate around only 2% of the addresses). The legend also lists the total number of nodes.

Table III: Used blocklists, number of listed entries, the portion also observed in the ITEG, and Iterations (Iter.) necessary until PTP convergence on Jan. 1, 2024.

Blocklist	Entries	Observed	Iter.
abuse.ch Feodo	174 IP addresses	34 (19.5,%)	12
Blocklist.de Strongips	472 IP addresses	161 (34.1%)	9
abuse.ch SSLBL	5 577 Certificates	19 (0.3%)	14
Openphish	9 289 Domains	3 461 (37.3%)	42

On the ITEG, we ran the PTP algorithm described in Section II-D for each blocklist in Table III. We downloaded each list daily and merged all entries from the last month as algorithm input. For example, if the scans for the ITEG on Feb. 1, 2024, were finished on Feb. 6, 2024, we considered the daily blocklist from Jan. 1, 2024, to Feb. 6., 2024, as PTP input. As seen in the table, our scans observed only a fraction of the blocked entries. Unresponsive IP addresses, unresolvable domains, and unseen certificates were not included in the graph. The low rates were expected, especially for the SSLBL, because certificates are a strong indicator that do not have to be removed timely after CNC servers stop using them. The PTP algorithm worked as follows: we fixed the threat score to one on each node on the input blocklist. Then, we ran the algorithm using $\varepsilon = 0.01$ until convergence; this took 9 to 42 iterations, depending on the input.

This section gave an overview of the collected data necessary to create a single ITEG and to run the PTP algorithm. Statistics about the graph's structure revealed a high centralization. In the following sections, we will evaluate the generated graph and calculated threat scores more closely.

B. Manual Analysis of Suspicious Clusters

This section shows four outliers that the ITEG modeling and the PTP scores revealed in the scans from January 1, 2024.

As shown in Table IV, the first outliers were 1.5×10^5 seemingly random domains (*e.g.*, `figtbnfjxbqjyl[.]in`) that resolved to a single IP address returning a blocked certificate. A blocked certificate has a SHA-1 hash that was published on the abuse.ch SSLBL blocklist. We did not see these domains in any other context, so the PTP algorithm assigned

Table IV: Suspicious groups of domains and IP addresses identified by a high and uniform threat score.

Suspicious Cluster	Score	Size
1. Seemingly random domains resolving to an IP address with a blocked certificate	100%	1.5×10^5
2. `unbouncepages` subdomains	33%	3.8×10^4
3. IP addresses without known domain returning a blocked certificate	100%	2.7×10^4
4. Seemingly random domains redirecting to a blocked Openphish domain	51%	3.1×10^3

a threat score of 100%. We extracted only 0.7% of these domains from the CT log; most originated from the zone files retrieved from the CZDS. We assume this large group of domains was automatically generated. The second outliers were 3.8×10^4 subdomains from `unbouncepages[.]com`. Because Openphish listed the main domain, all subdomains were assigned a score of 33%. Unbouncepages might not be malicious (at the time of writing, it was not listed on the blocklist anymore); although, we think the high threat score for its subdomains is justified assuming the information on the blocklist is correct. The third outliers were 2.7×10^4 IP addresses we observed only in our full IPv4 address scan returning a blocked self-signed certificate. None of our domains resolved to these addresses or were they involved in any other context; hence, the PTP algorithm assigned them a threat score of 100%. The SSLBL contains certificates that botnet C2 servers have used [6], some are default certificates (*e.g.*, embedded in web servers), meaning that the found addresses do not have to be malicious, but they are suspicious. The last outliers were 3.1×10^3 domains with a uniform score of 51%. The domains resolved to an unremarkable IP address, but were redirected to a domain on the Openphish blocklist (*i.e.*, `407979[.]com`). We believe the majority of this group was also generated because they seemed to be made of random 6-character strings with sometimes a prepended "www" (*e.g.*, `www11666x[.]com`). Similar to the above, only 8.0% were from the CT log, and the rest were CZDS domains.

To conclude, the ITEG can help manually analyze the TLS ecosystem. The PTP algorithm allowed us to quickly find new blocklist-related IP addresses and domains for a more thorough analysis. Moreover, we found several highly suspicious groups of IP addresses and domains.

C. Comparison with External Threat Intelligence Services

The ITEG allowed us to use a PTP algorithm to propagate a threat score from a set of input nodes listed on blocklists. In the following, we investigate IP addresses and domains with a high score in the graph from February 1, 2024, and check their status with the external threat intelligence services VirusTotal (VT) [31] and Google Safe Browsing (GSB) [32].

VT provides aggregated threat intelligence, and we used it to classify an IP address or domain as *malicious*, *suspicious*, *harmless*, or—in case of a Not Found error—as *unknown*. GSB also provides information about malicious websites; however, their Application Programming Interface (API) returned only a *malicious* flag or no data in other cases. Both are unable to check IPv6 addresses or certificates. We combined both sources for our evaluation and flagged an entry as *malicious* if either service identified it as *malicious* or *suspicious*. If present, we used the VT *harmless* class and flagged the rest as *unknown*. However, the VT API was rate-limited, and we could check only a small set of nodes; hence, we removed the first three large clusters we had already identified in the previous Section III-B as suspicious from this analysis.

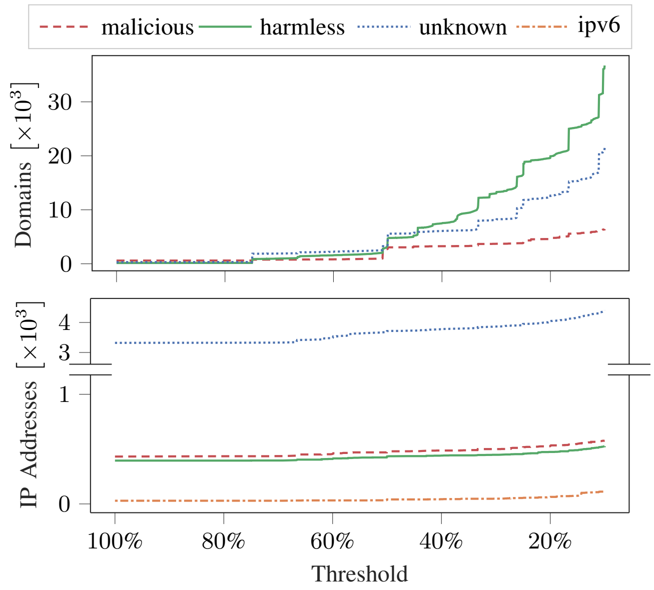


Figure 5: Cumulative number of domains and IP addresses with a threat score above the threshold, found via PTP, and classified according to the VT and GSB labels. Excluding the input blocklists and the first three clusters from Table IV.

Table III shows the cumulative number of domains and IP address with threat scores above the shown threshold grouped by the results from VT and GSB. We only analyzed nodes with a score above 10% because the number of affected nodes increased exponentially below this threshold. VT and GSB had no data about several domains and most IP addresses. The figure shows that we found an increasing amount of domains the lower we set the threshold. Most domains were flagged as *harmless*; however, we found an increasing number known as *malicious*. A large portion of the identified domains were unknown to VT and GSB. A different picture is revealed for IP addresses. By far, the largest portion of found IP addresses had a threat score of 100%, mainly due to servers returning a blocked certificate from the SSLBL. We found only a few additional addresses by lowering the threshold; among them were IPv6 addresses we could not check via either service.

For this analysis, we included the smallest cluster from Section III-B to evaluate it with VT and GSB. As a reminder, we identified 3.1×10^3 likely automatically generated domains (they had a very similar naming schema) that were redirected to a blocked phishing domain. These domains cause the

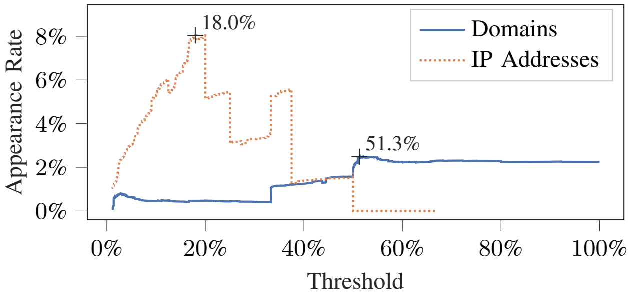


Figure 6: Rate of nodes with a score above the depicted threshold that appeared later on the blocklist. Marked are the thresholds providing the maximal rate.

small sharp increment at the 51% threshold in Figure 5. Interestingly, VT and GSB labeled only 53% of the identified domains *malicious*, 24% *harmless*, and 23% were unknown. Considering all of the indicators, we believe the entire cluster should be classified as *malicious*. It is unclear if VT and GSB did not add these domains due to a lack of visibility or if they were never operationalized. Nevertheless, this highlights our work's importance in identifying gaps in threat detection approaches. However, our approach is only intended as an indicator, and we will discuss false positives in Section V-2.

To conclude, the ITEG modeling and a PTP analysis can be used to propagate thread scores. Many of the newly found domains and IP addresses were already labeled as malicious *by other sources for threat intelligence, highlighting that our approach can identify threat-relevant subsets of the Internet. Moreover, we saw indicators that our work could help fill gaps in the knowledge base of threat intelligence services. However, our findings should not be directly treated as* malicious*, but as a starting point for a more thorough investigation.*

D. A Perspective Over Time

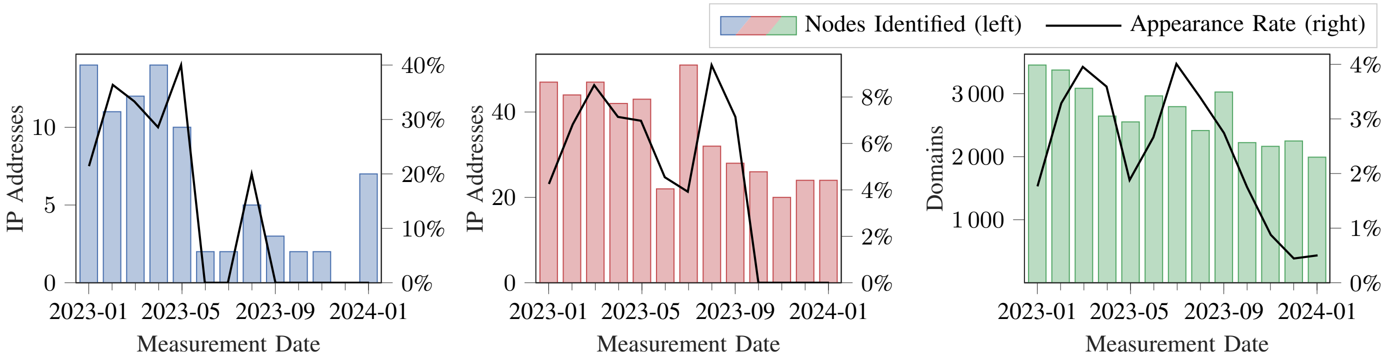
One goal of this work was to develop an approach that can find entities on the Internet before becoming known to be malicious. This section analyzes multiple ITEGs over time and whether nodes for which we calculated a high score appeared later on the blocklist. We analyzed 13 monthly measurements starting from January 1, 2023. For each month, we conducted the Internet-wide scans described in Section II-B, created an ITEG, and performed the PTP algorithm with the blocklists from Table III as input. We defined a *new appearance* as an IP address or domain that was not input for the PTP algorithm and which appeared on the input blocklist only after the ITEG measurements were completed and before Feb. 10, 2024. For example, the ITEG from May 1 would use the blocklists downloaded from April 1 to May 7 (because the last TLS scan finished on May 7) as PTP input and the blocklists from May 8, 2023, to Feb. 9, 2024, for the evaluation in this section. Although we identified certificates with high threat scores, none appeared later on the SSLBL. For this reason, we focus on domains and IP addresses in this section.

To investigate whether nodes with a high score appeared later on the respective blocklist, we first had to identify a set of relevant nodes. A straightforward approach considers all nodes with a score above a certain threshold. For this analysis, we wanted to use the best-performing thresholds. Hence, we calculated an Appearance Rate for domains and IP addresses for each threshold above 1% across all 13 ITEGs to choose the best option. We define the *Appearance Rate* as the portion of relevant nodes that later appeared on the blocklist. For example, if ten IP addresses scored above 50% and two appeared later on the blocklist, then the IP address Appearance Rate for a threshold of 50% would be 20%. Figure 6 illustrates the calculated rates. Then, we selected the best-performing thresholds of 51.3% and 18.0% to identify domains and IP address, respectively.

Figure 7 shows the number of nodes we identified for each of the 13 ITEGs and the percentage of these nodes that appeared later on the respective blocklist. The subfigures reveal that the performance of the overall approach highly depends on the input list and the time of the measurements. Sometimes, it worked very well, and other times, not at all. For example, our approach revealed ten new IP addresses when using the Feodo Tracker as input on April 1, 2023, where four of them appeared later on the blocklist. On the other hand, on December 1, 2023, we identified no new address; either our active measurements missed a relevant piece of data, or there was nothing to find. The Blocklist.de Strongips list was larger, and we identified more IP addresses, although the Appearance Rate was generally lower. The Openphish list contained many more blocked entries, and our approach revealed a few thousand additional domains. Although the Appearance Rate was only between 0.4% and 4.0%, this still means we found 557 domains before they appeared later on the list. Some identified nodes overlapped on consecutive scans; hence, the number of distinct addresses was 5 using Feodo and 6 with Strongips as input. 82% of the IP addresses appearing later on a blocklist returned the same certificate when we identified them and when we scanned them again after they appeared on the blocklist, indicating that the deployment was the same the whole time. The other 18% were unresponsive.

We can see that the rate of nodes appearing on the blocklists has decreased in the four most recent months. Several nodes we identified with a high score might be added only after writing this paper. Therefore, we analyzed the time until an entry appeared on a blocklist in Figure 8. On average, it took three months for an entry to appear, although it took much longer in some cases (350 days).

An example for a domain we identified before it was listed by Openphish was D1 (`bluewishlists[.]shop`). During our scans, we revealed its connection to the domain `usps[.]speed-mypkg[.]shop` (already listed on Openphish) because they both resolved to the same IP address and had the same valid Let's Encrypt certificate. Two days after we finished our scans, D1 was listed by Openphish. It is possible that the domain was operationalized only later or that the activity remained unnoticed for several days. Interestingly,



(a) abuse.ch Feodo

(b) Blocklist.de Strongips

(c) Openphish

Figure 7: Number of nodes identified using an ITEG and the described blocklist as PTP input on each measurement date. Also, showing the percentage of identified nodes that appeared later on the respective blocklist as Appearance Rate.

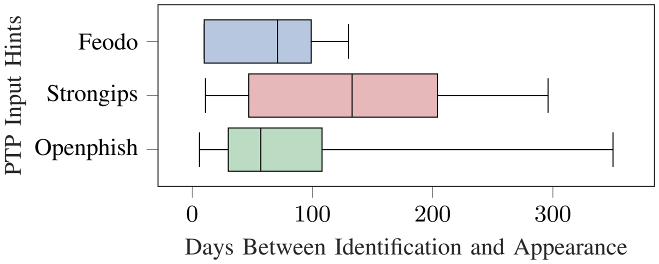


Figure 8: Time passed between our identification of an IP address or domain and its inclusion into the blocklist.

`usps[.]logistic-info[.]shop` was also included in the same Let's Encrypt certificate and we assigned it a high threat score. However, it was never added to the blocklist at the time of writing. The domain may have never been used, but we believe the threat score is justified due to its suspicious connection to an already blocked domain. The Openphish list only includes entries for which phishing was reported. Hence, this example shows how a low appearance rate is not necessary a problem of the blocklist nor our approach, but a result of different methodologies. Nevertheless, we had false positives, which we will discuss in Section V-2.

In conclusion, the ITEG modeling and PTP allows using existing blocklist data to identify additional IP addresses and domains before they appear on the respective blocklist. While the rate of nodes appearing is generally low, they are present, emphasizing that this work can find valuable cyber-threat intelligence as a starting point for further investigation. Depending on the input and measurement period, up to 40% *of the identified nodes showed up later on the blocklist.*

IV. Related Work

Multiple works (*e.g.*, Refs. [33, 34, 35]) performed Internet-wide active measurements to improve the understanding of the TLS ecosystem. VanderSloot *et al.* [36] found that IPv4 address space scans captured only one-third of their certificates, and a combination of CT logs and SNI scans is needed to acquire 99.4%, highlighting ne necessity of our SNI scans.

Refs. [3, 4, 37, 38, 39] have used data extracted from the TLS ecosystem to fingerprint and detect malicious C2 servers. However, their fingerprinting relies on multiple TLS connections to the same server, which can be a scalability issue. Our approach uses only a single connection.

Security-related Internet-wide graph modeling has been successfully applied on higher layers based on the web graph derived from hyperlinks (initially used for Google's PageRank [40]): Gyöngyi *et al.* [41] proposed "TrustRank", propagating a trust score to help identify spam, and Najafi *et al.* [42] introduced "MalRank", propagating a maliciousness score between connected nodes based on known threat intelligence. Because their work uses different data and graph algorithms, they can find other (possible complementary) types of threats compared to us.

Graph modeling is a common technique to model passively collected network data, as shown by Refs. [43, 44, 45] and the PTP paper [7]. Some consider TLS meta-data. The approaches range from score propagation to community detection or the application of graph neural networks.

Similar to this work, Simeonovski *et al.* [46] used DNS, IP addresses, and hosting information to derive a property graph and used taint-style propagation techniques to understand the impact of dependencies between Internet services and their providers on attacks. Like us, they propagated scores only in explicit directions reasoned with their domain knowledge.

To our knowledge, no work has applied a similar methodology than this paper to detect malicious activity. Neither has the feasibility been shown in a large-scale study over time.

V. Discussion

The ITEG modeling and PTP analysis creates new opportunities to analyze the TLS ecosystem and the Internet. We want to discuss some aspects in the following paragraphs.

1) IPv6 Addresses and Blocklists: We could not find IPv6 addresses on our blocklists, and VT and GSB did not support them either. However, we found IPv6 addresses related to the blocked entries; *e.g.*, 29 IPv6 addresses returned a blocked certificate in the scans from February 1, 2024. An approach similar to this work could be a starting point to find malicious IPv6 addresses so that blocklists include them in the future.

2) Examples for False Positives: Our methodology allowed us to find suspicious IP addresses and domains. However, we noticed also false positives. One cause was that the Openphish blocklist contained Uniform Resource Locators (URLs), but we only modeled domains. Most of the time this was acceptable; however, in some cases this introduced an error. For example, a URL on `sites.google.com` was blocked, but of course not the whole domain should be treated malicious. This affected smaller websites that redirected their domain to a google site URL and we falsely propagated a threat score to their domains. A similar case was the `bit.ly` domain (a URL shortener) that we falsely labeled as blocked because some URLs were blocked. This affected domains redirecting to `bit.ly`; however, it was an uncommon behavior and several of the domains we found this way were actually known as *malicious* by VT, but most of them as *harmless*. We could have prevented both cases by using a blocklist that considers domains only. Other cases for false positives were subdomains configured insecurely with one of the default certificates blocked by the SSLBL. These subdomains likely hosted services intended for internal or experimental use only (with names like `internal`, `ftp`, `webdisk`, etc.). Due to the score propagation this resulted in a threat score on the other (correctly configured) domains as well.

3) Valid Certificates in the ITEG: A core aspect of our TLS ecosystem is that it allows for a decentralized assessment of trust issued by Certificate Authoritys (CAs) through digital signatures. CAs check whether the private key owner also owns the subject names listed in the certificates. In the context of HTTPS the names are usually domains (rarely, IP addresses). However, we decided not to treat valid certificates (having a valid signature during scanning) differently in the graph and the PTP analysis: *i)* it makes modeling, parsing, and analyzing more straightforward and independent of root stores; *ii)* we focus on malicious activity where mostly self-signed certificates are used (e.g., all observed SSLBL certificates were self-signed); and *iii)* the graph already contains an intuition of validity as triangles between domains, IP addresses and certificates. Unless we were subject to a Man-in-the-Middle (MITM) attack, a CA like Let's Encrypt would issue a certificate for all requested domains that resolve to the IP address of the requesting server (basically, creating a triangle) via the ACME [47] protocol. Hence, to a certain extent, the validity of certificates is embedded in the ITEG edges; thus, it is considered in the PTP analysis. However, future work focusing on benign services might model validity differently.

4) Mitigation: This work relies on malicious actors using the TLS ecosystem in ways that leak information about their deployments. Such leaks can arise out of necessity (*e.g.*, when a single IP address serves multiple services), limited resources and prioritization on other aspects, or human errors. A malicious actor may mitigate detection by isolating or obfuscating their deployments. The former can be achieved if every malicious activity has its own IP address, domain and certificate. If an address appears on a blocklist and fallback addresses are used, new domains and certificates must also be used to prevent information leakage. However, isolating deployments can be difficult, and the risk of human error remains. It is possible to obfuscate deployments from our PTP algorithm, *e.g.*, by including an unusual number of domains in a certificate. Although the algorithm would not assign a high score in such cases, it would be an outlier in the ITEG, detectible by other means.

5) Limitations: The data collected with our Internet scans tried to be as comprehensive as possible. However, we only scanned open ports 443 and used a single vantage point. Hence, we missed parts of the TLS ecosystem (*e.g.*, mail), and our results might be biased due to DNS load-balancing and location [48]. However, it is sufficient to show the potential of this work and provide valuable insights.

VI. Conclusion

This work investigated modeling Internet-wide active measurements as a single property graph and applying a PTP algorithm to find new, potentially malicious servers. To evaluate the methodology, we conducted a one-year-long measurement study of 13 monthly Internet-wide DNS and TLS scans and used the collected data to create respective Internet-wide TLS Ecosystem Graphs. In our latest graph, we found four highly suspicious clusters of domains and IP addresses, two likely due to automatic domain generation. We checked nodes outside these clusters with external threat intelligence services. We found that more IP address with a threat score above 10% were flagged as *malicious* than *harmless*, highlighting that our approach focused on malicious activity. Moreover, with the help of an optimized detection threshold, we identified 11 IP addresses and 557 domains throughout the last year before they were added to a blocklist. Depending on the input blocklist and the measurement period, up to 40% of the detected nodes appeared later on the respective blocklist. We do not think nodes with a high score should be directly considered as malicious; however, our approach narrows down the millions of possible domains and IP addresses, which can be a starting point for a more thorough investigation.

The results were obtained by modeling Internet-wide DNS and TLS measurements as a Labeled Property Graph. Existing blocklists were input for a PTP algorithm that propagated a threat score among IP addresses, domains, and certificates. Our proposed model explains and simplifies the massive data collectible from the TLS ecosystem. The proposed application of PTP can help to find previously unknown threats on the Internet and provide valuable threat intelligence.

This work could be extended in the future using additional data (*e.g.*, routing information) or alternative graph algorithms. Our modeling and data collection pipeline can be an addition to the tool-set of researchers analyzing the TLS ecosystem to answer security and non-security related questions.

Acknowledgment

The authors would like to thank the anonymous reviewers and our shepherd Gautam Akiwate for their valuable feedback. This work is partially funded by Germany Federal Ministry of Education and Research (BMBF) under the projects PRIMEnet (16KIS1370), 6G-life (16KISK001K) and 6G-ANNA (16KISK107), and by the German Research Foundation (HyperNIC, DFG grant no. CA595/13-1).

References

[1] C. Labovitz, "Internet traffic 2009-2019," in *Proc. Asia Pacific Regional Internet Conf. Operational Technologies*, 2019.

[2] R. Roberts and D. Levin, "When Certificate Transparency Is Too Transparent: Analyzing Information Leakage in HTTPS Domain Names," in *Proceedings of the 18th ACM Workshop on Privacy in the Electronic Society*, 2019, DOI:10.1145/3338498.3358655.

[3] M. Sosnowski, J. Zirngibl, P. Sattler, G. Carle, C. Grohnfeldt, M. Russo, and D. Sgandurra, "EFACTLS: Effective Active TLS Fingerprinting for Large-scale Server Deployment Characterization," *IEEE Transactions on Network and Service Management*, 2024, DOI:10.1109/TNSM.2024.3364526.

[4] M. Sosnowski, J. Zirngibl, P. Sattler, and G. Carle, "DissecTLS: A Scalable Active Scanner for TLS Server Configurations, Capabilities, and TLS Fingerprinting," in *Proc. Passive and Active Measurement (PAM)*, 2023, DOI:10.1007/978-3-031-28486-1_6.

[5] Sean Gallagher, "Nearly half of malware now use TLS to conceal communications," Apr 2021. [Online]. Available: https://news.sophos.com/en-us/2021/04/21/nearly-half-of-malware-now-use-tls-to-conceal-communications/

[6] abuse.ch, "SSL Blacklist," (accessed Feb. 21, 2024). [Online]. Available: https://sslbl.abuse.ch/

[7] K. M. Carter, N. Idika, and W. W. Streilein, "Probabilistic threat propagation for malicious activity detection," in *Proc. IEEE Int. Conference on Acoustics, Speech and Signal Processing*, 2013, DOI:10.1109/ICASSP.2013.6638196.

[8] M. Sosnowski, P. Sattler, J. Zirngibl, T. Betzer, and G. Carle. (2024) ITEG: Additional Material. [Online]. Available: https://tumi8.github.io/iteg/

[9] P. Mockapetris, "Domain names - concepts and facilities," RFC 1034, 1987, DOI:10.17487/RFC1034.

[10] S. Boeyen, S. Santesson, T. Polk, R. Housley, S. Farrell, and D. Cooper, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile," RFC 5280, 2008, DOI:10.17487/RFC5280.

[11] J. Zirngibl, P. Buschmann, P. Sattler, B. Jaeger, J. Aulbach, and G. Carle, "It's over 9000: Analyzing early QUIC Deployments with the Standardization on the Horizon," in *Proc. ACM Int. Measurement Conference (IMC)*, 2021, DOI:10.1145/3487552.3487826.

[12] J. Zirngibl, F. Gebauer, P. Sattler, M. Sosnowski, and G. Carle, "QUIC Hunter: Finding QUIC Deployments and Identifying Server Libraries Across the Internet," in *Proc. Passive and Active Measurement (PAM)*, 2024, DOI:10.1007/978-3-031-56252-5_13.

[13] Apache Software Foundation. (2023) Apache Spark™ - Unified engine for large-scale data analytics. (accessed Feb. 21, 2024). [Online]. Available: https://spark.apache.org/

[14] F. Manola and E. Miller, "RDF Primer," *W3C recommendation*, 2004, (accessed Feb. 21, 2024). [Online]. Available: https://www.w3.org/TR/rdf-primer/

[15] I. Robinson, J. Webber, and E. Eifrem, *Graph Databases*. O'Reilly Media, 2013.

[16] T. Abedrabbo, N. Watt, D. Fox, and A. Vukotic, *Neo4j in Action*. Manning, 2014.

[17] (2024) GraphFrames: DataFrame-based Graphs. (accessed Feb. 16, 2024). [Online]. Available: https://github.com/graphframes/graphframes

[18] Neo4j, Inc. . (2024) Neo4j Graph Database & Analytics | Graph Database Management System. (accessed Feb. 16, 2024). [Online]. Available: https://neo4j.com/

[19] Majestic, "The Majestic Million," (accessed Feb. 19, 2024). [Online]. Available: https://majestic.com/reports/majestic-million/

[20] Cisco. (2023) Umbrella Popularity List. (accessed Feb. 9, 2024). [Online]. Available: https://s3-us-west-1.amazonaws.com/umbrella-static/index.html

[21] Google. Chrome User Experience Report. (accessed Feb. 8, 2024). [Online]. Available: https://developer.chrome.com/docs/crux

[22] Chromium. HSTS Preload List. (accessed Feb. 9, 2024). [Online]. Available: https://hstspreload.org/

[23] Cloudflare. Cloudflare Radar: Domain Rankings. (accessed Feb. 8, 2024). [Online]. Available: https://radar.cloudflare.com/domains

[24] OpenPhish. (2024) OpenPhish - Phishing Intelligence. (accessed Feb. 9, 2024). [Online]. Available: https://openphish.com/

[25] Q. S. Birk Blechschmidt. (2021) MassDNS. (accessed Feb. 21, 2024). [Online]. Available: https://github.com/blechschmidt/massdns

[26] NLnet Labs. (2023) Unbound. Last accessed November 2, 2023. [Online]. Available: https://www.nlnetlabs.nl/projects/unbound

[27] Z. Durumeric, E. Wustrow, and J. A. Halderman, "ZMap: Fast internet-wide scanning and its security applications," in *Proc. USENIX Security Symposium*, 2013.

[28] O. Gasser, M. Sosnowski, P. Sattler, and J. Zirngibl. (2023) TUM goscanner. (accessed Feb. 21, 2024). [Online]. Available: https://github.com/tumi8/goscanner

[29] O. Gasser, Q. Scheitle, P. Foremski, Q. Lone, M. Korczynski, S. D. Strowes, L. Hendriks, and G. Carle, "Clusters in the Expanse: Understanding and Unbiasing IPv6 Hitlists," in *Proc. ACM Int. Measurement Conference (IMC)*, 2018, DOI:10.1145/3278532.3278564.

[30] J. Zirngibl, S. Deusch, P. Sattler, J. Aulbach, G. Carle, and M. Jonker, "Domain Parking: Largely Present, Rarely Considered!" in *Proc. Network Traffic Measurement and Analysis Conference (TMA)*, 2022.

[31] Chronicle Security. (2024) VirusTotal. (accessed Feb. 21, 2024). [Online]. Available: https://www.virustotal.com

[32] Google. (2024) Safe Browsing - Making the world's information safely accessible. (accessed Feb. 20, 2024). [Online]. Available: https://safebrowsing.google.com

[33] Z. Durumeric, J. Kasten, M. Bailey, and J. A. Halderman, "Analysis of the HTTPS certificate ecosystem," in *Proc. ACM Int. Measurement Conference (IMC)*, 2013, DOI:10.1145/2504730.2504755.

[34] R. Holz, J. Hiller, J. Amann, A. Razaghpanah, T. Jost, N. Vallina-Rodriguez, and O. Hohlfeld, "Tracking the Deployment of TLS 1.3 on the Web: A Story of Experimentation and Centralization," *SIGCOMM Comput. Commun. Rev.*, 2020, DOI:10.1145/3411740.3411742.

[35] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of Age: A Longitudinal Study of TLS Deployment," in *Proc. ACM Int. Measurement Conference (IMC)*, 2018, DOI:10.1145/3278532.3278568.

[36] B. VanderSloot, J. Amann, M. Bernhard, Z. Durumeric, M. Bailey, and J. A. Halderman, "Towards a complete view of the certificate ecosystem," in *Proc. ACM Int. Measurement Conference (IMC)*, 2016, DOI:10.1145/2987443.2987462.

[37] J. Althouse, A. Smart, R. Nunnally, Jr., and M. Brady, "Easily Identify Malicious Servers on the Internet with JARM," 2020, (accessed Feb. 20, 2024). [Online]. Available: https://engineering.salesforce.com/easily-identify-malicious-servers-on-the-internet-with-jarm-e095edac525a

[38] M. Sosnowski, J. Zirngibl, P. Sattler, G. Carle, C. Grohnfeldt, M. Russo, and D. Sgandurra, "Active TLS Stack Fingerprinting: Characterizing TLS Server Deployments at Scale," in *Proc. Network Traffic Measurement and Analysis Conference (TMA)*, 2022.

[39] E. Papadogiannaki and S. Ioannidis, "Pump Up the JARM: Studying the Evolution of Botnets Using Active TLS Fingerprinting," in *Proc. Int. Symposium on Computer Communications (ISCC)*, 2023, DOI:10.1109/ISCC58397.2023.10218210.

[40] S. Brin and L. Page, "The anatomy of a large-scale hypertextual Web search engine," *Computer Networks and ISDN Systems*, 1998, DOI:10.1016/S0169-7552(98)00110-X.

[41] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating Web Spam with Trustrank," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, 2004.

[42] P. Najafi, A. Mühle, W. Pünter, F. Cheng, and C. Meinel, "MalRank: A Measure of Maliciousness in SIEM-Based Knowledge Graphs," in *Proceedings of the 35th Annual Computer Security Applications Conference*, 2019, DOI:10.1145/3359789.3359791.

[43] D. Zhuang and J. M. Chang, "Enhanced PeerHunter: Detecting Peer-to-Peer Botnets Through Network-Flow Level Community Behavior Analysis," *IEEE Transactions on Information Forensics and Security*, 2019, DOI:10.1109/TIFS.2018.2881657.

[44] Š. Mandlík, T. Pevný, V. Šmídl, and L. Bajer, "Malicious Internet Entity Detection Using Local Graph Inference," *IEEE Transactions on Information Forensics and Security*, 2024, DOI:10.1109/TIFS.2024.3360867.

[45] Y. Kazato, Y. Nakagawa, and Y. Nakatani, “Improving Maliciousness Estimation of Indicator of Compromise Using Graph Convolutional Networks,” in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, 2020, DOI:10.1109/CCNC46108.2020.9045113.

[46] M. Simeonovski, G. Pellegrino, C. Rossow, and M. Backes, “Who Controls the Internet? Analyzing Global Threats Using Property Graph Traversals,” in *Proceedings of the 26th International Conference on World Wide Web*, 2017, DOI:10.1145/3038912.3052587.

[47] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten, “Automatic Certificate Management Environment (ACME),” RFC 8555, Mar. 2019, DOI:10.17487/RFC8555.

[48] G. Wan, L. Izhikevich, D. Adrian, K. Yoshioka, R. Holz, C. Rossow, and Z. Durumeric, “On the origin of scanning: the impact of location on Internet-wide scans,” in *Proc. ACM Int. Measurement Conference (IMC)*, 2020, DOI:10.1145/3419394.3424214.

[49] D. Dittrich, E. Kenneally *et al.*, “The Menlo Report: Ethical principles guiding information and communication technology research,” *US Department of Homeland Security*, 2012.

[50] C. Partridge and M. Allman, “Addressing Ethical Considerations in Network Measurement Papers,” in *Proceedings of the 2015 ACM SIGCOMM Workshop on Ethics in Networked Systems Research*, 2016, DOI:10.1145/2793013.2793014.

Appendix

Ethical Considerations

During our scans, we followed ethical considerations and best practices described by Dittrich *et al.* [49] and Partridge and Allman [50]. We scanned with a limited rate, used a blocklist containing opt-out requests, and informed about our scans using reverse DNS, a website on the scanning machine, and whois information. We responded to all requests regarding our scans and requested opt-outs. Our study collected no user data but focused on publicly visible server data. No human subjects were part of this study.