Rolling the DICE: A Device Identification and Classification Engine to detect vulnerable devices facing the Internet

Yaben, Ricardo ; Vasilomanolakis, Emmanouil

Published in:
Proceedings of the 9th Network Traffic Measurement and Analysis Conference (TMA conference 2025)

Publication date:
2025

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Yaben, R., & Vasilomanolakis, E. (in press). Rolling the DICE: A Device Identification and Classification Engine to detect vulnerable devices facing the Internet. In *Proceedings of the 9th Network Traffic Measurement and Analysis Conference (TMA conference 2025)* IFIP.

Rolling the DICE: A Device Identification and Classification Engine to detect vulnerable devices facing the Internet

Ricardo Yaben
Technical University of Denmark
Kongens Lyngby, Denmark
rmyl@dtu.dk

Emmanouil Vasilomanolakis
Technical University of Denmark
Kongens Lyngby, Denmark
emmva@dtu.dk

***Abstract*—In recent years, the Internet has experienced a significant surge in connected devices, with an ever-growing number of sensors and monitoring systems—spanning industries and domestic networks—now exposed to the Internet and reliant on our ability to keep them secure (e.g., in healthcare, home automation, and manufacturing). However, securing Internet-facing devices is no trivial task. Applying patches, firewall rules, and strong credentials are only small steps during their security life-cycle. Since these steps work in tandem, failing even a few can significantly increase the risk of compromise. The cybersecurity community continues to build on its efforts to mitigate this issue from many fronts, all while investigating society's new challenges with technology and their security implications. To aid in this task, we present DICE, a modular Device Identification and Classification Engine to detect vulnerabilities on Internet-facing devices. DICE assists in most phases of the identification process, from automating Internet-wide scans to labeling results. In addition, DICE can help notify the affected device owners – an ongoing issue across the literature – by creating detailed reports and mitigation strategies. As proof of concept, we share preliminary implementations of various modules to identify recurrent issues in 8 protocols widely used in IoT and OT devices. These modules aim to discover security pitfalls beyond common vulnerabilities, such as signs of abandonment, obsolescence, and security negligence.**

***Index Terms*—device fingerprinting, Internet measurements, IoT, OT, vulnerability identification**

I. Introduction

As the number of Internet-facing devices continues to grow, so do concerns about their security. Ensuring these devices remain secure while exposed to the Internet is an increasingly complex challenge with potentially severe consequences. Although the field of cybersecurity is maturing and societal demand is rising, its complexity is evolving just as rapidly. We are reminded of this reality daily, as cybersecurity incidents soar and have a larger impact [1]. Therefore, the community's involvement is paramount in understanding the issue comprehensively and mitigating its threats. In this regard, notable efforts have been made to propose tools and methods to measure and analyze the Internet's population and behavior, such as Nmap, masscan, and the ZMap ecosystem; Scopus alone contains more than 600 publications on the topic of Internet measurements in the last 25 years and has grown every year since ZMap's release. As Durumeric et al. [2] mentioned in their review of ZMap's usage over the years, Internet scanning tools have been integral to studying Internet behavior and uncovering widespread security issues. According to their analysis, these tools are also found in many vulnerability scanning solutions (e.g., Palo Alto's Cortex Xpanse, Rapid7 InsightVM, and Nuclei) and Internet scanning services (e.g., Shodan, Censys, and ShadowRunner). However, despite the numerous publications and security tools available, one of the most common limitations – and promises to solve in future work – is the lack of reproducible and comparable results [3, 4]. Most studies lack transparency on their methodologies (e.g., publicly available probes, labeling, and classification systems), forcing authors to spend significant efforts re-implementing the state of the art instead of focusing on the issue at hand.

This paper introduces our current work in progress: DICE, a Device Identification and Classification Engine. DICE is primarily designed as a modular vulnerability identification engine for Internet-wide surveys, capable of profiling devices and orchestrating targeted scans. Our goal is to create a foundation for modern Internet surveys and address the aforementioned limitations. We are developing DICE as a common platform for the community to strategize, execute, analyze, and compare Internet measurements. As a proof of concept, we also share an early implementation of DICE modules to identify security issues in IoT and Operational Technology (OT) devices exposed to the Internet. We hope to inspire the community to establish new and refined requirements for Internet measurements and create a common language for this space, rolling DICE modules, and measure their effects. Our contributions are as follows:

- We introduce DICE, a modular Device Identification and Classification Engine for Internet measurements. We describe how DICE addresses common limitations in the literature, and explore further use cases and research directions.
- We demonstrate an example implementation of DICE

978-3-903176-74-4

signatures to detect indicators of security misuse, such as misconfigurations and abandonment. These signatures are applied to scanning results from a previous measurement study targeting eight widely used protocols in IoT and OT.

II. Related Work

Internet surveys are modern methods of studying the Internet and its population based on remote-host interrogation techniques. Tools such as ZMap and Masscan made scanning the Internet possible within hours and with fewer resources than their spiritual predecessor, Nmap. This method has proven to be extremely useful, with many studies uncovering widespread security issues such as Heartbleed [5], security concerns in IoT and OT [6, 7, 8, 9, 10], identifying or fingerprinting vulnerable devices [11, 12], issues in honeypots [13], monitoring events such as tracking botnets [1] and the impact of war on critical Infrastructure [14]. Their limitations are also well studied, with multiple publications on the implications of vantage point location [15], Internet churn, scanning velocity, and blocking behavior [16]. The literature even has multiple examples of strategies and guidelines to conduct Internet measurements [3]. However, the lack of reproducible and comparable methods threatens to slow down the good pace achieved in recent years, even when this issue is commonly voiced across the literature and is treated as one of the most fundamental impediments in Internet measurement studies and surveys [3, 17, 4, 18]. In response to these challenges, we propose DICE, an engine that lays the groundwork towards standardizing processes and methods for conducting and comparing Internet measurements.

III. DICE: The Engine

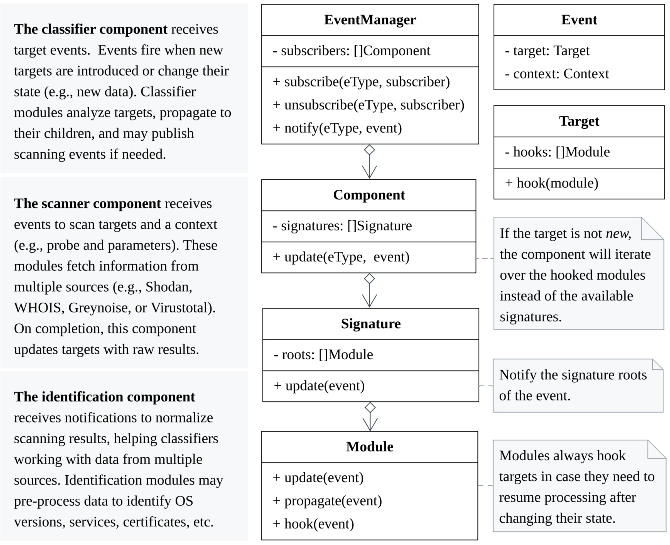
This section introduces DICE's design principles and core concepts of its structure. In addition, we include use cases to explain how DICE can help mitigate the field's limitations.

A. Design Principles

DICE combines many design principles from previous attempts at creating similar solutions. Our design goals are meant to develop an engine that can assist at most stages of an Internet measurement as needed, whether it is classifying devices or orchestrating complex measurements.

a) Connected: Scanning, identifying, and classifying are separate tasks that DICE tackles simultaneously. DICE distributes each task in a component, which should remain separate for independent use (e.g., for re-classifying, verifying, comparing, or extending results). However, measurements with higher complexity and internal dependencies require DICE components to stay connected – known as the rule of composition.

b) Scanner agnostic: At its core, DICE is designed to support widely adopted scanners while preserving user flexibility to choose, integrate, or develop their own scanning tools. Its guiding philosophy is to advance the field by sharing resources and providing access to the tools and materials used in measurements. Imposing restrictions on scanner choices would undermine its usefulness; therefore, DICE remains loosely coupled with scanners and deliberately unopinionated.



The classifier component receives target events. Events fire when new targets are introduced or change their state (e.g., new data). Classifier modules analyze targets, propagate to their children, and may publish scanning events if needed.

The scanner component receives events to scan targets and a context (e.g., probe and parameters). These modules fetch information from multiple sources (e.g., Shodan, WHOIS, Greynoise, or Virustotal). On completion, this component updates targets with raw results.

The identification component receives notifications to normalize scanning results, helping classifiers working with data from multiple sources. Identification modules may pre-process data to identify OS versions, services, certificates, etc.

EventManager
- subscribers: []Component
+ subscribe(eType, subscriber)
+ unsubscribe(eType, subscriber)
+ notify(eType, event)
Event
- target: Target
- context: Context
Target
- hooks: []Module
+ hook(module)
Component
- signatures: []Signature
+ update(eType, event)
If the target is not *new*, the component will iterate over the hooked modules instead of the available signatures.
Signature
- roots: []Module
+ update(event)
Notify the signature roots of the event.
Module
+ update(event)
+ propagate(event)
+ hook(event)
Modules always hook targets in case they need to resume processing after changing their state.

Fig. 1. DICE's implementation of observer pattern and descriptions of the three main components: *scanner*, *classifier*, and *identifier*.

c) Support a wide range of measurements: We adopt the concept of *measurement modules* described by Paxson et al. [19], and heavily used in designing most modern scanning tools (e.g., Nmap, Masscan, ZMap, and ZGrab). In DICE, we separate the engine from the measurement, which becomes a chain of probes, identification rules, and classifiers to survey the Internet.

d) Self-reflective: As an engine for orchestrating and comparing Internet measurements, DICE must provide testable metrics that reflect the measurement state, performance, and results summaries. Such metrics are rarely shared in existing studies, making evaluating and comparing measurements difficult. Sharing these metrics enables use cases such as observability analysis, performance tuning, and classification modeling. Future versions of DICE could leverage these insights to diagnose measurement issues and recommend improvements.

e) Help navigating results: DICE functions should remain within the scope of the engine, i.e., orchestrating measurements, and identifying and classifying devices. However, its usefulness would be significantly limited without built-in methods for navigating and interpreting results. This additional functionality enables result comparison and report generation, streamlining the process of conducting responsible disclosures.

B. Structure

This section provides a brief overview of DICE's core concepts, as depicted in Figure 1: *components*, *signatures*, and *modules*.

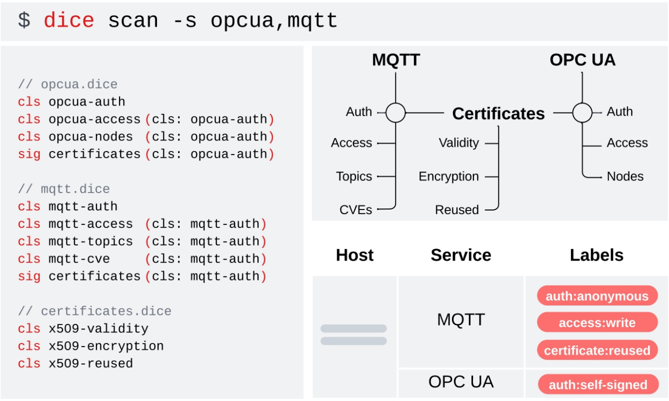
a) Components: DICE splits the phases of a measurement into *components*: scanning, identification, and classification. These components are *modular* to support a wide range of measurements, designed to work simultaneously as needed, and supervised by the engine. Module behavior depends on the parent component: (1) scanning modules take targets and dispatch scanning results, (2) identification modules normalize these results, and (3) classification modules assign labels and may propose further scans. Targets are DICE's inputs, references to an IP address in a measurement, and hold all related data to that address. DICE outputs targets as profiles of the collected addresses, their discovered services, fingerprints, and labels assigned. DICE enables this functionality by adopting an event-driven architecture with the observer pattern, propagating targets across component modules to manage state transitions and assign labels.

b) Signatures: Conceptually, components in DICE are made of Directed Acyclic Graphs (DAGs) in which the vertices correspond to modules. DAGs are powerful tools helping DICE to add dependencies between modules and avoid loops. First, DAGs are directed, i.e., graphs are made of vertices connected by edges with a direction (DAGs can have only one edge between two vertices). Then, DAGs are acyclic, i.e., the graph does not contain loops. Other rule-based systems with similar characteristics name their rulesets *signatures* (e.g., Suricata and Snort). We also adopt this naming convention for simplicity and refer to DAGs of modules in DICE as signatures. Signatures maintain their mathematical properties, allowing DICE to combine and embed other signatures to form a larger one. This approach is particularly useful for collecting signatures (e.g., identifying IoT issues) and describing measurements as configuration files, including other parameters such as the scanner choice. This design helps ease sharing and comparing measurements.

c) Modules: Simply put, modules are processing units that assign labels or add further information to targets, including scanning results. DICE communicates with modules over RPC, simplifying the development of new modules and supporting most programming languages. Modules vary in complexity, from simple filtering rules to complex classification models. DICE feeds targets to all roots of the loaded classification signatures (roots are vertices without an inbound edge). Classification modules may check for prerequisites before processing targets and suggest new scanning signatures. Targets that match a module's evaluation are assigned a label and propagated to the module's children.

C. Use cases and research directions

DICE's modular structure and dependency features allow us to define and share whole processing pipelines and measurement metadata (e.g., monitoring metrics, aggregations, and configurations). We anticipate DICE will be used beyond the covered in the literature and help in other areas of interest, e.g.: (1) those derived from longitudinal studies, such as outages and cyber-security developments, (2) artificial intelligence on DICE classified datasets and new modules using these models, (3) IP and port prediction techniques to improve efficiency and scanning coverage, (4) mitigating Internet churn for time-sensitive measurements, etc. We look forward to seeing what other research questions DICE may support.



```
$ dice scan -s opcua,mqtt
```

```
// opcua.dice
cls opcua-auth
cls opcua-access (cls: opcua-auth)
cls opcua-nodes  (cls: opcua-auth)
sig certificates (cls: opcua-auth)

// mqtt.dice
cls mqtt-auth
cls mqtt-access  (cls: mqtt-auth)
cls mqtt-topics  (cls: mqtt-auth)
cls mqtt-cve     (cls: mqtt-auth)
sig certificates (cls: mqtt-auth)

// certificates.dice
cls x509-validity
cls x509-encryption
cls x509-reused
```

Host	Service	Labels
	MQTT	auth:anonymous access:write certificate:reused
	OPC UA	auth:self-signed

Fig. 2. Example of DICE scanning pipeline targeting Internet-facing devices exposing vulnerable OPC UA or MQTT services.

Figure 2 shows a simplified example of DICE pipelines to scan the Internet for vulnerable devices exposing MQTT and OPC UA services. In this example, DICE loads the signatures to scan, identify, and classify hosts exposing these services, and any additional signatures included as internal dependencies, such as certificate classifiers or other services. Signatures describe their layout, referencing modules and how they are linked (left side). In this case, the MQTT `access` module should only trigger on targets with `auth` labels, such as successful anonymous authentications or through self-signed certificates. Similarly, certificate modules trigger when a service includes this information during the communication – dependencies are represented as circles in the graph. Lastly, DICE outputs session metadata, scanning results, identification fingerprints, and classification labels to a new database. These results can be used to conduct responsible disclosure campaigns and for further analysis.

To inspire the community, we share an early implementation of signatures to identify vulnerable IoT and OT devices following a classification criterion similar to the ones presented in [20] (see [21]). These signatures include modules to identify access control issues, allowing untrusted clients to establish anonymous connections and access internal information. Moreover, we evaluate server certificates for validity issues, reuses, and insecure configurations. The included identification modules fingerprint hosts at different granulation levels based on the information they leak, and classify those leaking sensitive information (e.g., server state) or suffering from poor maintenance (e.g., deprecated versions). Most of these modules build upon previously assigned labels (e.g., authorization rules can only be tested on targets with authentication labels), exploiting the signature dependency features in DICE to create meaningful profiles of Internet-exposed devices. Finally, we use a dataset from early 2025 with 8 protocols as an input, and show a summary of the results.

Acknowledgments

This work is part of the project *Digital ghost ships: unveiling the threat of misconfigured and obsolete systems*, funded by the Independent Research Fund Denmark (grant number: 2035-00030B)

References

[1] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *26th USENIX Security Symposium (USENIX Security 17)*. Vancouver, BC: USENIX Association, Aug. 2017, pp. 1093–1110.

[2] Z. Durumeric, D. Adrian, P. Stephens, E. Wustrow, and J. A. Halderman, "Ten years of zmap," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, 2024, pp. 139–148.

[3] V. Paxson, "Strategies for sound internet measurement," in *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 263–271.

[4] M. Safaei Pour, C. Nader, K. Friday, and E. Bou-Harb, "A comprehensive survey of recent internet measurement techniques for cyber security," *Computers & Security*, vol. 128, p. 103123, 2023.

[5] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman, "The matter of heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, ser. IMC '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 475–488.

[6] A. Mirian, Z. Ma, D. Adrian, M. Tischer, T. Chuenchujit, T. Yardley, R. Berthier, J. Mason, Z. Durumeric, J. A. Halderman, and M. Bailey, "An internet-wide view of ics devices," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, 2016, pp. 96–103.

[7] S. Srinivasa, J. M. Pedersen, and E. Vasilomanolakis, "Open for hire: Attack trends and misconfiguration pitfalls of iot devices," in *Proceedings of the 21st ACM Internet Measurement Conference*, ser. IMC '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 195–215.

[8] L. Markowsky and G. Markowsky, "Scanning for vulnerable devices in the internet of things," in *2015 IEEE 8th International conference on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS)*, vol. 1. IEEE, 2015, pp. 463–467.

[9] J. Rüth, T. Zimmermann, and O. Hohlfeld, "Hidden treasures – recycling large-scale internet measurements to study the internet's control plane," in *Passive and Active Measurement*, D. Choffnes and M. Barcellos, Eds. Cham: Springer International Publishing, 2019, pp. 51–67.

[10] B. Zhao, S. Ji, W.-H. Lee, C. Lin, H. Weng, J. Wu, P. Zhou, L. Fang, and R. Beyah, "A large-scale empirical study on the vulnerability of deployed iot devices," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 3, pp. 1826–1840, 2022.

[11] H. Kim, T. Kim, and D. Jang, "An intelligent improvement of internet-wide scan engine for fast discovery of vulnerable iot devices," *Symmetry*, vol. 10, no. 5, 2018.

[12] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 542–553.

[13] S. Morishita, T. Hoizumi, W. Ueno, R. Tanabe, C. Gañán, M. J. van Eeten, K. Yoshioka, and T. Matsumoto, "Detect me if you… oh wait. an internet-wide view of self-revealing honeypots," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, 2019, pp. 134–143.

[14] R. Singla, S. Srinivasa, N. Reddy, J. M. Pedersen, E. Vasilomanolakis, and R. Bettati, "An analysis of war impact on ukrainian critical infrastructure through network measurements," in *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, 2023, pp. 1–10.

[15] 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 *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 662–679.

[16] Z. Durumeric, M. Bailey, and J. A. Halderman, "An {Internet-Wide} view of {Internet-Wide} scanning," in *23rd USENIX Security Symposium (USENIX Security 14)*, 2014, pp. 65–78.

[17] Z. Durumeric, D. Adrian, P. Stephens, E. Wustrow, and J. A. Halderman, "Ten years of zmap," in *Proceedings of the 2024 ACM on Internet Measurement Conference*, ser. IMC '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 139–148.

[18] K. Claffy, D. Clark, J. Heidemann, F. Bustamante, M. Jonker, A. Schulman, and E. Zegura, "Workshop on overcoming measurement barriers to internet research (wombir 2021) final report," *SIGCOMM Comput. Commun. Rev.*, vol. 51, no. 3, p. 33–40, Jul. 2021.

[19] V. Paxson, J. Mahdavi, A. Adams, and M. Mathis, "An architecture for large scale internet measurement," *IEEE Communications Magazine*, vol. 36, no. 8, pp. 48–54, 1998.

[20] R. Yaben, N. Lundsgaard, J. August, and E. Vasilomanolakis, "Towards identifying neglected, obsolete, and abandoned iot and ot devices," in *2024 8th Network Traffic Measurement and Analysis Conference (TMA)*, 2024, pp. 1–10.

[21] R. Yaben and E. Vasilomanolakis, "Ricyaben/tma-2025-poster: Early implementation of DICE for the TMA conference (poster session)," May 2025. [Online]. Available: https://github.com/RicYaben/tma-2025-poster