ChatGPT

Passive OS Fingerprinting Tools and Datasets

Lesser-Known Passive OS Fingerprinting Tools

- **Satori (Python)** – Satori is an open-source passive OS fingerprinting tool, originally a Windows program revived in Python. It passively inspects multiple protocols (e.g. DHCP, TCP/IP stack, HTTP User-Agent, SMB, SSL/TLS JA3 signatures) to identify a host's OS [1] [2]. Satori does not send probes; instead it sniffs normal traffic and uses a database of fingerprints (for various TTLs, window sizes, DHCP options, etc.) to guess the OS. Maintained by Eric Kollmann, Satori is a modern alternative to classic p0f and comes with regularly updated fingerprint files [3].

- **Zardaxt (Python)** – Zardaxt is a relatively new passive TCP/IP fingerprinting tool designed as a more *maintained* replacement for p0f's OS detection. It runs on a server and examines the **first incoming SYN packet** of each TCP handshake, looking at header metadata like IP TTL, TCP window size, MSS, and TCP options order to infer the client's OS [4]. The author created Zardaxt as a Python reimplementation inspired by Satori, noting that p0f's signature database is outdated and that a simple Python tool is easier to extend than p0f's C code [5]. Zardaxt can output an OS probability score (for example, a SYN might be 57% likely Android vs 24% Windows, etc.) and even includes a live API for classifying traffic in real-time.

- **PRADS (C)** – *Passive Real-time Asset Detection System* (PRADS) is an open-source tool that listens to network traffic and catalogs assets (hosts and services) along with their OS fingerprints [6]. PRADS uses **digital TCP/IP fingerprints** to recognize operating systems (it performs both TCP and UDP OS fingerprinting) and can also match application-layer data (client/server banners) for a comprehensive fingerprint [7]. It never sends packets (completely passive) and was built to handle high-speed networks (including IPv6). PRADS is often mentioned as a modern replacement for older passive scanners like p0f, PADS, and Sancp [8]. It logs identified hosts with details such as their guessed OS, which can be fed into IDS/IPS or SIEM tools for network monitoring.

- **AUB's OS Fingerprinting Tool (C)** – An academic project from the American University of Beirut provides an open-source OS fingerprinting tool (and an associated detector for fingerprinting attacks) that is well-documented and modifiable [9] [10]. This tool issues a normal HTTP request to a target's web server and then analyzes the response packets' metadata to guess the OS. It uses a weighted combination of **ten TCP/IP stack features** to distinguish among 24 OS versions [11]. Notably, it looks at the initial TTL value, the IP "Don't Fragment" (DF) bit, patterns in the IP ID field, the default TCP window size, the total SYN packet size, presence/order of TCP options (like window scale, NOP, etc.), and even behavior of certain flags in TCP headers [11]. By aggregating these clues (each given a weight based on reliability), the tool outputs a likely OS (e.g. differentiating Windows vs Linux variants). Being open-source (in C with ~56 GitHub stars), this project is a good base for students to modify or extend the fingerprinting logic for research.

Open-Source Datasets for OS Fingerprinting

- **CESNET Idle OS Traffic Dataset (2025)** – A recently released public dataset focused on passive OS fingerprinting. It consists of *idle* background traffic captured from a wide range of operating systems (multiple families, types, and versions) in a lab environment [12]. For each OS (39 different OS instances, including various Windows 10/11, multiple Linux distributions like Ubuntu, Fedora, Debian in different versions, Android, BSD, etc.), the dataset provides raw packet captures (PCAPs) of that system's network traffic when no user activity is occurring [12]. Because this "idle" traffic includes things like periodic updates, NTP, DHCP, ARP, and other background network chatter characteristic to each OS, it's a rich source of fingerprinting features. The dataset also comes with flow records and extracted metadata (DNS queries, HTTP User-Agent strings, TLS handshakes, etc.) along with documentation of the VM configurations. This allows training a model to distinguish OS nuances — for example, Windows 10 vs Windows 11 might have different default TTLs or update server patterns, and Ubuntu 22.04 vs 24.04 may contact different domains or have slightly different TCP/IP defaults. The dataset is publicly downloadable (via Zenodo DOI) and suitable for machine learning, given that it includes ground-truth OS labels and no sensitive payloads [12].

- **nPrint OS Detection Dataset (CIC-IDS2017 Derived, 2021)** – This open dataset was created by re-labeling the well-known CIC IDS 2017 traffic captures with **per-device operating system labels** [13]. The original CIC-IDS2017 testbed included various machines (Windows, Ubuntu Linux, MacOS, etc.) generating normal and attack traffic. Researchers mapped each source IP address in the pcap to its host OS (information provided by CIC) and then split the traffic by host. The resulting dataset contains a classification task with **13 OS classes** (e.g., different versions of Windows, Linux, and other OS present in the lab) [13]. For practical use, the traffic is segmented into samples of 100 packets, each labeled with the source host's OS. Only packet headers (IPv4/TCP) are considered for features, making it ideal for training machine learning models on metadata like TTL, TCP window size, etc., without relying on payload. The entire dataset is under 1 GB (uncompressed) and is provided publicly (via Google Drive) along with a benchmark task description [14]. This dataset is suitable for algorithms like Random Forests or SVMs – indeed, it has an existing benchmark where methods are evaluated by their accuracy in identifying the OS from flow snippets. It offers a convenient starting point for training a model to classify OS purely from network metadata.

- **CESNET Network Flow OS Dataset (2024)** – Another open dataset (by Hulák *et al.*) provides millions of labeled network *flows* with OS annotations [15] [16]. Instead of raw packets, this dataset aggregates traffic into flows (e.g. using IPFIX records) captured from several real networks (university subnets and a private lab network). Each flow record is tagged with the **OS of the source device**, determined by a combination of methods (DHCP logs, HTTP User-Agent strings, TLS SNI, reverse DNS, and even Shodan lookups) to ensure accurate labeling [17]. The OS labels in this dataset are at the family level (five classes: Windows, Linux, macOS, iOS, Android) rather than specific versions [18], but the dataset is valuable for its richness and size – over 32 million flows in total. Crucially, it includes the kind of metadata features useful for passive fingerprinting: initial TTL (recorded and binned to nearest power of two), TCP window size, window scale, MSS, a bitfield for TCP options seen, IP layer flags (e.g. DF), packet counts, and more for each flow [16]. The data is available through an academic repository (CESNET/Zenodo) and is meant to support machine learning research – one can train models to predict OS from these flow features. Although the OS granularity is coarse in this set, it complements the above datasets by providing a **large-scale, real-world** sample where patterns in TTL, TCP options, and other header fields can be correlated with

broad OS categories [16]. This could be useful for validating a model's performance on real network traffic and ensuring it generalizes beyond lab conditions.

Each of the above datasets is suitable for a final-year project: they are publicly downloadable and contain the necessary packet metadata (TTL, window size, TCP options, DF bit, IP ID behavior, etc.) to train a machine learning model. By using these datasets, a student can train classifiers (Random Forest, SVM, or even neural networks) to passively fingerprint operating systems with no reliance on packet payloads (encrypted traffic is fine). This allows creation of a demo where the model infers a remote host's OS from a pcap or live capture, which would nicely showcase the **practical efficacy of passive OS fingerprinting** in an Internet Security project.

Sources:

1. Emms, S. *"5 Best Free and Open Source Passive OS Fingerprinting Tools." LinuxLinks* (Apr 28, 2024) – Descriptions of Satori, PRADS, etc. [1] [2]

2. Hjelmvik, E. *"Passive OS Fingerprinting." Netresec Blog* (Nov 5, 2011) – Overview of passive techniques and mention of tools like Satori [3].

3. Zardaxt GitHub README – Explanation of tool purpose and design (N. Tschacher, 2023) [4] [5].

4. PRADS GitHub and LinuxLinks review – Features of PRADS for passive asset/OS detection [7].

5. Ghali, C. *et al.*, AUB "OS-Fingerprinting" Tool README – Uses TTL, DF, IP ID, window size, TCP options order, etc. for OS detection (open-source) [11].

6. Bartoš, V. *et al. "CESNET Idle OS Traffic"* (CNSM 2025 dataset paper) – Public idle traffic dataset with multiple OS versions (PCAPs and metadata) [12].

7. Holland, J. *et al. "nPrint OS Detection Dataset."* (2021) – OS-labeled subset of CIC-IDS2017 traffic, 13-class OS classification task [13] [14].

8. Hulák, M. *et al. "Evaluation of TCP/IP-based OS fingerprinting... (new datasets)"* (Zenodo 2024) – Large flow dataset with OS labels and features like TTL, window size, MSS, etc. [16].

---

[1] [2] satori - passive OS fingerprinting tool - LinuxLinks
https://www.linuxlinks.com/satori-passive-os-fingerprinting-tool/

[3] Passive OS Fingerprinting
https://www.netresec.com/?page=Blog&month=2011-11&post=Passive-OS-Fingerprinting

[4] [5] GitHub - NikolaiT/zardaxt: Passive TCP/IP Fingerprinting Tool. Run this on your server and find out what Operating Systems your clients are *really* using.
https://github.com/NikolaiT/zardaxt

[6] [7] [8] PRADS - Passive Real-time Asset Detection System - LinuxLinks
https://www.linuxlinks.com/prads-passive-real-time-asset-detection-system/

[9] [10] [11] GitHub - cesarghali/OS-Fingerprinting: Operating system remote fingerprinting attack and detection tools
https://github.com/cesarghali/OS-Fingerprinting

[12] CESNET Idle OS Traffic: A Dataset of Idle Traffic of Various Operating Systems
https://dl.ifip.org/db/conf/cnsm/cnsm2025/1571196299.pdf

[13] [14] nprint os detection | pcapML Benchmarks
https://nprint.github.io/benchmarks/os_detection/nprint_os_detection.html

[15] [16] [17] [18] Evaluation of TCP/IP-based OS fingerprinting methods using new datasets
https://zenodo.org/records/12745822