Exposure is All You Need: Using LLMs to Identify Critical Endpoints at Internet Scale

Stijn Pletinckx*
University of California, Santa Barbara
Santa Barbara, USA
stijn@ucsb.edu

Milad Nasr
Anthropic
San Francisco, USA
milad@srxzr.com

Sanghyun Hong
Oregon State University
Corvallis, USA
sanghyun.hong@oregonstate.edu

Zakir Durumeric
Stanford University
Stanford, USA
zakir@cs.stanford.edu

Nicholas Carlini
Anthropic
San Francisco, USA
nicholas@carlini.com

Abstract

The Internet hosts billions of publicly accessible services, many of which inadvertently expose sensitive data or allow device manipulation. Historically, understanding the security implications of exposed Internet services required manual analysis or a priori knowledge, neither of which scales to the diversity of devices and services in the wild. In this paper, we show that Large Language Models (LLMs) change this dynamic and can identify security-relevant exposure without predefined rules or fingerprints. After evaluating their efficacy, we leverage LLMs to conduct the first large-scale measurement of HTTP(S)-based exposure and RTSP camera feeds. We use LLMs to build a hierarchical taxonomy of services and develop a multi-stage pipeline that combines lightweight ML classifiers with targeted LLM analysis to identify and triage security-critical exposure. We examine five types of security-critical exposure: directory listings, business applications and dashboards, IoT device interfaces, pages showing stack traces, and camera feeds. Our approach identifies thousands of dangerously exposed services, including healthcare management systems, financial documents, API keys, and private cameras.

CCS Concepts

• **Security and privacy** → *Web application security*; Systems security; Network security.

ACM Reference Format:
Stijn Pletinckx, Milad Nasr, Sanghyun Hong, Zakir Durumeric, and Nicholas Carlini. 2026. Exposure is All You Need: Using LLMs to Identify Critical Endpoints at Internet Scale. In *19th Workshop on Artificial Intelligence and Security (AISec '26), November 15–19, 2026, The Hague, Netherlands.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3847352.3848119

*Work done while at Stanford University.

AISec '26, The Hague, Netherlands

ACM ISBN 979-8-4007-3031-3/2026/11
https://doi.org/10.1145/3847352.3848119

1 Introduction

Publicly exposed services on the Internet are the top attack vector into organizations [43]. Historically, most attacks have targeted popular applications with known vulnerabilities or commonly deployed protocols that allow easily testing compromised credentials (e.g., RDP and IKE). This prioritization stems from economies of scale where an automated workflow can efficiently check the exploitability of millions of hosts. However, beyond popular software packages, the Internet is also composed of a long tail of other devices, services, and protocols that are often misconfigured, including device configuration panels, development tools, and back office applications that directly expose sensitive data. It is easy for an attacker or security analyst to recognize the security risk posed by a web interface on a case-by-case basis, but their heterogeneity makes them difficult to interpret programmatically, historically keeping them out of reach of many attackers.

Large Language Models (LLMs) change this calculus. In cybersecurity, LLMs have shown tremendous potential for automating tasks that previously required manual analysis, including weaponizing software vulnerabilities [25] and building polymorphic malware [26]. LLMs are particularly potent because of their ability to adapt to novel situations in ways classical programs cannot—the very type of task that has previously hindered understanding and remediating the long-tail of Internet service exposure.

In this paper, we show that LLMs can automatically identify security-relevant exposures in the long tail of the Internet. Our work suggests that the obscurity of many Internet services no longer provides security: LLMs can interpret unfamiliar interfaces, reason about context, and effectively triage the most critically exposed services. We argue that this paradigm shift will change how adversaries attack organizations.

We approach this study in two phases. In the first phase, we construct a hierarchical clustering of HTTP(S)-based Internet services (Section 3). This categorizes the types of exposed services and hints at the long tail of previously undetected exposure. We identify two broad classes of security-relevant exposure. First, instance-specific information disclosure (e.g., leaking passwords, secret keys, or other sensitive PII). Second, a long tail of unauthenticated business applications, administrative dashboards, industrial control interfaces, and other critical devices.

In the second phase, we show how LLMs can triage risk by investigating four common types of HTTP(S) exposure. First, we analyze public directory listings. We build an LLM-guided crawler that explores directory listings and determines the sensitivity of exposed data. We analyze 234,188 directory listings and identify that 7,629 expose sensitive data, including passports, financial statements, and immigration documents. Second, we analyze unauthenticated business applications and dashboards. We find 24,500 sensitive cases, including healthcare management systems. Third, we study IoT and ICS administration interfaces; we identify 12,022 control systems, including interfaces that allow controlling gas turbines. Fourth, we identify sites with errors or stack traces, including 20,612 sites that leak sensitive data like cryptographic keys and database credentials. Last, we consider a different modality of data: RTSP-based cameras. We find that 74.6% of unauthenticated camera feeds (7,796 systems) capture sensitive content, such as hospital rooms.

In sum, this paper makes the following contributions:

- We present the first hierarchical categorization of sensitive HTTP-based services on the public IPv4 address space.
- We analyze five categories of web-based exposure and show how LLMs can scale the identification and severity-ranking of exposed services, sensitive data exposures, and unprotected devices across different modalities.
- We show that service exposure (e.g., a directory listing) does not imply the presence of sensitive data, highlighting the importance of efficient triage beyond discovery.

Our work indicates that the threat model for Internet-facing services has changed and enables defenders to systematically reason about their attack surface.

2 Background

Our work draws on techniques from Internet scanning and Large Language Models (LLMs). We provide background on both below:

2.1 Internet-Wide Scanning

ZMap [23] and other high-speed network scanners have enabled answering a wide range of security measurement questions, ranging from mapping compromised SSH servers [39] to uncovering attacks against email delivery [19]. They have also enabled understanding Internet exposure [8, 33, 37, 38, 45] and performing large-scale vulnerability notification campaigns [22, 34, 52]. In most studies, researchers first identify responsive hosts (e.g., on TCP/443) using a stateless scanner like ZMap and then follow up with a stateful L7 protocol handshake (e.g., TLS handshake) to collect service configuration data [20]. Researchers subsequently analyze collected data to understand underlying software and hardware, service configuration, and ownership [8, 50]. Beyond static fingerprints, much of this analysis remains manual and domain-specific [2, 3, 16, 18, 21].

2.2 Large Language Models

LLMs are highly capable, general-purpose analyzers that accept text, images, audio, or video and produce structured natural-language interpretations [6, 14, 41]. LLMs already play a growing role in cybersecurity, supporting incident triage, red teaming, automated log interpretation, and cyber-training environments (e.g., CyBench [55], CyberGym [53]). Adversaries have leveraged LLMs for phishing generation, exploit discovery, and attack orchestration [1, 7]. LLMs are inexpensive compared to human analysts, but costly at scale: processing 1 MB of data costs roughly $10 [6, 41]. While costs continue to fall [4], analyzing hundreds of millions of Internet service responses remains prohibitively expensive. Our results show that we do not need to operate on the full corpus. Lightweight models can filter out benign content, allowing LLMs to focus on the small fraction of ambiguous or potentially risky exposures.

2.3 Threat Model

We consider a remote adversary with access to Internet service data as could be found through Internet scanning or a service like Shodan or Censys, but no privileged access. Their goal is reconnaissance: to identify services that expose sensitive data or security-sensitive functionality. We do not assume credentials, prior knowledge of specific applications, or the ability to bypass authentication.

2.4 Challenges

For adversaries, the barrier to conducting reconnaissance continues to drop. Internet-wide scanning has been readily available for over a decade [20, 23], and platforms like Censys [21] and Shodan provide indexed, searchable snapshots of the IPv4 address space. What has historically been missing is the ability to *interpret* this data at scale. For defenders, comprehensively preventing initial access through Internet exposure is significantly harder. Security teams operate under severe resource constraints: workforce shortages [28] lead to understaffed SOCs, many of which already suffer from alert fatigue [46]. At the same time, recent studies show that most compromises are first discovered by the attacker instead of the defender [36], highlighting the imbalance. Our approach shows how defenders can leverage LLMs to gain more comprehensive visibility into and prioritize improving their attack surface.

3 Identifying and Clustering Exposure

We start by prompting LLMs to determine whether the presence of an HTTP(S) interface creates a security risk and to grade the severity of the exposure. From there, we label types of exposure and build a hierarchical classification.

3.1 Dataset

We collect a corpus of 20M random HTTP(S) hosts from the November 3, 2025 snapshot of IPv4 services from Censys [21]. We sample across all ports given prior work that has shown that HTTP services on non-standard ports are more likely to serve security-sensitive services [29, 30].

3.2 Identifying Security-Relevant Exposure

We prompted Claude Sonnet 4.5 to analyze 1.1M unique HTML bodies, deduplicated from 2M HTTP(S) services sampled from our Censys dataset. We specifically asked Claude to: (1) tag the exposure as intentionally public or unintentionally exposed; and (2) to grade the severity of each incidentally exposed service from low to critical using a pre-defined rubric. We provide both our prompts and rubric in Appendix A. Overall, Claude characterized 7.3% of bodies as

Table 1: Human validation of the LLM exposure labels. Each service was independently reviewed by two of four human annotators. Percentages in the middle three columns are relative to the number of services with each LLM label. The final column reports LLM agreement conditional on the two humans agreeing.

LLM Label	N	Consensus = LLM	Consensus ≠ LLM	Human Disagree	LLM Agree | Consensus
Exposed	90	57 (63.3%)	22 (24.4%)	11 (12.2%)	72.2%
Public	110	92 (83.6%)	8 (7.3%)	10 (9.1%)	92.0%
Overall	200	149 (74.5%)	30 (15.0%)	21 (10.5%)	83.2%

unintentionally exposed with the severities of: 9% Critical, 38% High, 48% Medium, and 4% Low.

Validation. To measure agreement between LLM and human judgments, four human annotators participated in a manual review, with each example independently reviewed by two annotators. Annotators labeled each example as either intentionally public or unintentionally exposed, and could additionally select "impossible to tell" when the available content was insufficient to make a determination. For example, some services rendered an entirely blank page because they embedded content, such as an `iframe`, that was no longer available. We exclude an example from the validation if either annotator selected "impossible to tell," since no meaningful comparison with the LLM label can be made in these cases. The resulting validation set contains 200 services. Of these, the LLM labeled 90 as unintentionally exposed and 110 as intentionally public.

Among the 179 services for which the two human reviewers reach a consensus, the LLM agrees with the human-consensus label for 149 (83.2%). As Table 1 shows, this agreement differs between the two classes. For services labeled intentionally public, the LLM agrees with the human consensus in 92.0% of cases, compared with 72.2% for services labeled unintentionally exposed.

Much of this variance stems from the fact that many of these decisions are not black-and-white; the inter-rater agreement between the human annotators was 89.5%, only slightly above the human-LLM agreement rate. As an example, both humans and the LLMs frequently disagreed about whether or not pages with error messages, but no sensitive data, were unintentionally exposed, or poorly designed but intentionally public.

3.3 Open-Set Keyword Tagging

We next categorize the types of security problem posed by identified exposure. We perform an open-set categorization where we prompted Claude to tag each risky HTTP body with 5–10 of its own keywords to describe the problem, but do not prescribe the set of possible keywords ahead of time. We show the top keywords in Table 2. The value of open-set keyword tagging is that it enables discovering new types of exposure that we may not have expected a priori. Below are several example of these categories:

- `battery-backup-system`: UPS backup systems with network interfaces that occasionally report their location, charge state, and power draw; state-changing functionality (such as power-down) is behind an authentication page.
- `xor-encryption`: several services include client-side JavaScript-based encryption that uses a simple XOR cipher to either obfuscate malicious payloads or encrypt passwords.
- `real-time-gps-coordinates`: services track the real-time status of electric cars, providing their precise location, state of charge, total miles driven, and other metrics.
- `certificate-private-keys`: servers that host their root directory occasionally contain private `.pem` files.

Table 2: Top Keywords from Open-Vocabulary Annotation

Keyword	Services (%)
aws-credentials	11.33
authentication-bypass	10.30
plesk-control-panel	9.98
hardcoded-credentials	9.13
hardcoded-secrets	7.89
secret-access-key	7.11
sentry-dsn	5.87
version-disclosure	5.84
authentication-tokens	5.70
unauthenticated-access	5.50
internal-ip-exposure	5.47
authentication-bypass-risk	5.34
router-admin-panel	5.03
aws-credentials-exposed	4.84
authentication-page	4.77
router-admin-interface	4.52
network-infrastructure	4.40
web-hosting-panel	4.07
javascript-embedded-secrets	3.93
api-keys	3.32

Because we perform open-set categorization, we find that Claude frequently assigns different keywords to the same concept; the remainder of this section addresses this challenge.

3.4 Hierarchical Clustering

Next, we describe our approach for hierarchically clustering types of exposures. Given the initial open-set keywords used above, we first provide an LLM with the full list of keywords and ask it to return high-level categories that describe the data. Because we have a large number of unique keywords, we first conducted this analysis on batches of keywords, then iteratively prompted Claude to merge them into higher-level categories until we arrived at 17 high-level keyword categories. Then, we performed a second closed-set keyword mapping where we asked Claude to categorize each exposure using the 17 keyword categories. Only a small fraction of pages (<0.5%) have no appropriate keyword applied.

Then, given this first level of categorization, we repeat the process: for each first-level category of content, we again construct new keywords that further describe the content, then we create categories from these keywords, giving us a second level to the tree. By repeating this process an arbitrary number of times, we can describe our content to an arbitrary degree of granularity. We show our results in Figure 1. As an example, "information-disclosure," "credential-exposure," and "admin-interface datasets"

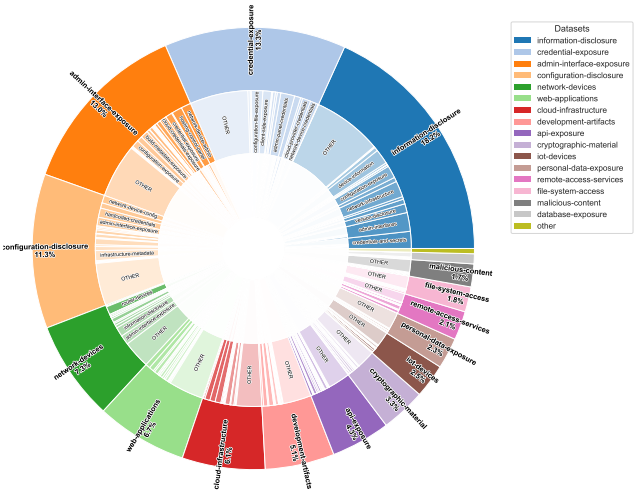


Figure 1: Exposed Services by Content Category

hold 249,948, 181,966, and 178,886 keywords respectively. Inside the admin-interface wedge, hosting control panels (8.10%), network-device admin consoles (7.79%), and build-metadata leaks (6.00%) together account for over 20% of all admin-interfaces.

4 Scaling Identification and Triage

On any given day, Internet scanning platforms like Censys collect over a terabyte of service content, which would be infeasible to process with LLMs available today: capable models like GPT-5 or Claude Sonnet 4.5 cost approximately $1 USD per megabyte of input processed, and $10 USD per megabyte of output produced. It would cost millions of dollars to process one day's worth of data. We therefore approach this problem by searching for ways that still make extensive use of LLMs, but also to combine them with classical machine learning techniques to reduce cost.

Building on our categories of exposure, we consider how to identify candidate services for deeper LLM-based investigation using more cost-effective models. To do this, we proceed in three steps. First, given our initial dataset of 1% of IPv4 services that we have already annotated and categorized, we use this to construct a training dataset of HTTP(S) services from each exposure category (e.g., directory listings). Then, we train a simple-and-small machine learning model (e.g., a bag-of-words model with logistic regression) on this dataset. We do not expect that this model will be perfect, but rather optimize for a method that has high recall (i.e., it does not miss potential security risks), which we can use as a filtering mechanism to identify services that need LLM-based consideration. Once we have done this, we finally describe a method to *sort* the data by severity, by having the LLM compare pages (pairwise) and then rank them by a calculated Elo-like score. We show that this pairwise algorithm is more precise than asking LLMs to output an objective classification without context.

Table 3: Classifier performance for distinguishing exposed services from benign content, pooled over the four categories.

Classifier	AUROC	AUPRC	TPR at 0.5% FPR
Ridge-TFIDF	0.98	0.72	0.60
SVM	0.98	0.78	0.55
SGD-Hash	0.98	0.82	0.62
LR	0.96	0.72	0.48
CNB-TFIDF	0.95	0.58	0.38
LR-Elastic	0.96	0.39	0.12
FastText	0.50	0.04	0.01
Embed-LR	0.50	0.04	0.01

4.1 Labeling a Training (and Testing) Dataset

Constructing a labeled training and testing dataset is straightforward given the categorized services from the prior section. We select some harmful category (e.g., `directory-listings`) and choose all examples that have this category as the positive class; we then select 100,000 other examples that do not have this category as the negative class. We then randomly partition these into a training and testing set, and use this for training a future ML model. In this way, the small ML model is trained to emulate the behavior of the much larger LLM but at a small fraction of the cost.

4.2 Classification via Traditional ML

We evaluate eight fast classifiers that represent diverse modeling paradigms: FastText models, bag-of-words–based linear models, and embedding-based classifiers. We use logistic regression (LR), support vector machine (SVM), stochastic gradient descent with hashing (SGD-Hash), ridge regression with TF-IDF features (Ridge-TFIDF), complement naïve Bayes with TF-IDF (CNB-TFIDF), elastic-net regularized logistic regression (LR-Elastic), a supervised FastText classifier (FastText), and a sentence-transformer–based embedding classifier combined with LR (Embed-LR). We implement all models through a unified training and evaluation pipeline to ensure consistent pre-processing, feature extraction, data splits, and predictions. We evaluate each classifier on four categories: Directory Listings, Uncommon IoTs, Business Applications and Dashboards, and Error Traces. We use a 70/30 train-test split.

Performance Comparison. We evaluate how well each classifier distinguishes security-critical services from benign services using the Area Under the Receiver Operating Characteristic Curve (AUROC), Area Under Precision-Recall Curve (AUPRC), and the True Positive Rate (TPR) at a strict 0.5% False Positive Rate (FPR). We report both AUROC and AUPRC to account for class imbalance, as positive samples comprise only 3.7% of the test set. Under such imbalance, AUROC can remain high despite poor precision, while AUPRC directly reflects performance on the minority class (AUPRC of a random classifier is 0.037 [10] for reference). As seen in Table 3, Ridge-TFIDF, SVM, and SGD-Hash achieve the highest overall AUROC, and SGD-Hash is also the strongest on AUPRC and at the low-FPR operating point. CNB-TFIDF has the lowest AUROC of the scoring classifiers and is correspondingly weaker at that operating point. LR and LR-Elastic obtain comparable overall AUROCs, but LR performs considerably better under the strict FPR constraint. In contrast, FastText and Embed-LR perform at chance

level. FastText defaults to classifying all pages as benign because of the label imbalance (96% benign). Embed-LR exhibits the opposite behavior: its high recall of 0.91 but low precision of 0.10 indicates that it fires on nearly all pages, effectively classifying most samples as exposed. Appendix B provides additional data on the prediction agreement among the ML classifiers.

Despite not achieving the highest AUROC, we select LR as the primary classifier for subsequent analyses. Its AUPRC of 0.72 matches Ridge-TFIDF's and trails the best classifier by 0.10, and it offers deterministic and interpretable linear decision boundaries, and minimal implementation overhead. These properties make it a practical choice for lightweight, large-scale security analysis. Under the low-FPR regime (0.5%), LR achieves strong separability for most exposure types: Directory Listings and Error Traces achieve high TPRs of 0.94 and 0.66. Dashboards and Uncommon IoTs show moderate detectability, with TPRs of 0.28 and 0.21, respectively (see Figure 2).

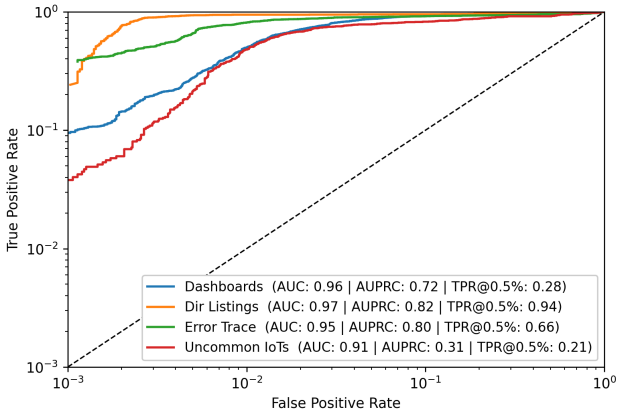


Figure 2: ROC curves for LR in log-log scale. Labels show AUROC, AUPRC and TPR at 0.5% FPR for each data category.

4.3 Priority Ranking

The final step in the process—once we have completed a first filtering pass with our simple ML methods and followed up with the full LLM evaluation pass—is to rank-order exposures by severity. This allows a human analyst to more efficiently triage the data, processing content according to its (predicted) severity.

Absolute Ranking. Our first approach was to have an LLM categorize severity according to a 10-point scale, and then sort the pages according to the ranking. However, we encountered several problems. First, if we ask LLMs to provide an absolute ranking, LLMs have very little internal consistency between what counts as a severity of 8 compared to a severity of 10 across runs. Indeed, the same webpage with small unimportant changes is frequently categorized with different severity levels on different runs of the LLM. While the differences are relatively small (e.g., scoring a 7 vs. a 10), this significantly affects the final ordering algorithm. We hypothesize this is because having an objective definition of what an "8" vs a "9" means is challenging. Even with a detailed rubric, we are still left with two problems. First, upon manual inspection, we found it did not match our intuitions because not all cases could be covered by the rubric; and second, when a large number of pages are given the same categorization (e.g., of a score of 8), ordering within this bucket is random.

Table 4: Distribution of Content Sensitivity. The majority of the sensitive data resides on exposed dashboards. However, RTSP camera feeds have the highest proportion of High and Critical exposures (74.6%).

Severity	Business App. & Dashboards	Error/Stack Traces	Uncommon IoT and ICS	RTSP Camera Feeds
Low	459 (1.9%)	747 (3.6%)	463 (3.9%)	1,041 (9.9%)
Medium	16,953 (69.2%)	17,538 (85.1%)	8,110 (67.4%)	1,613 (15.4%)
High	4,797 (19.6%)	1,857 (9.0%)	2,183 (18.2%)	6,572 (62.9%)
Critical	2,291 (9.4%)	470 (2.3%)	1,266 (10.5%)	1,224 (11.7%)
Total	24,500	20,612	12,022	10,450

Pairwise Comparisons. Instead of asking the LLM to make absolute judgments on the severity of a given exposure, we perform a (nearly) proper sorting algorithm, and ask the LLM to compare two pages and pick which is a more severe security risk. Because there is still some noise in the sorting method, we cannot use classical sorting algorithms from any algorithms textbook. Instead, we draw inspiration from ranking the quality of participants when entering in a competition. Concretely, we treat each page as a "player" with a latent severity score and use an Elo pairwise comparison scheme: whenever the LLM judges page A to be more severe than page B, we update their scores and increase winner Elo rating and decrease the loser Elo rating. To increase the convergence of the algorithm, instead of randomly pairing together pages, we run a first *Swiss tournament*[1] and then calculate Elo based on the game results of this tournament. We find that twenty rounds is sufficient to identify the pages with highest severity.

Evaluating the Two Approaches. We performed a small-scale evaluation where we randomly showed an author of this paper the top-50 most severe webpages as rated by these two ranking approaches. One at a time (e.g., comparing one of the first-50 pages as determined by pairwise rankings to one of the top-50 pages as determined by absolute ranking) the author annotated which of the two pages they determined to be more severe. This annotation was performed blind (i.e., without providing additional information about which algorithm was used) and randomized (so the pages did not occur in the same order). The pairwise-ranking algorithm significantly out-performed the absolute ranking: For 39 of the top 50 pages, the manual annotator rated the pairwise page as being higher severity than the absolute method.[2] This clearly demonstrates the superiority of pairwise comparisons.

5 Internet Exposure

Leveraging the Section 4 filtering models and frontier LLMs, we analyze four types of common exposure: directory listings, business

[1] In a Swiss tournament of N rounds, on each round, participants are sorted according to their current win/loss score. Then, the player #1 competes against player #2, the player ranked #3 against #4, etc. Players are then re-ranked, and the process repeats.
[2] Note that even if the pairwise method was *perfectly aligned* with human preferences and the absolute method was merely approximately correct, achieving a score of 50/50 may be impossible: we may compare the 50th most severe page according to pairwise ranking with the 1st most severe page according to the absolute ranking. Other comparison methods have more severe quirks, and we believe this method is the least bad method of comparison.

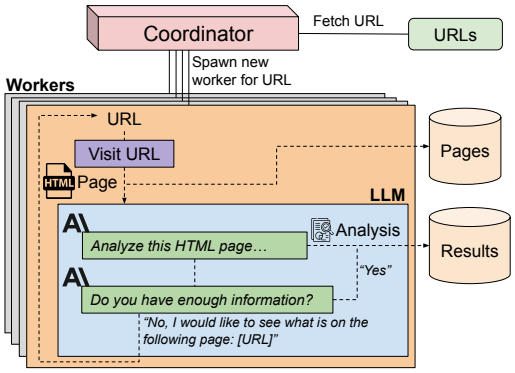


Figure 3: LLM-Guided Web Scraper. The scraper starts by fetching the index page of the HTTP service and feeding it into the LLM for analysis. The LLM decides whether it needs to explore the directory listing further, or whether it has enough information to determine whether the service is unintentionally exposed. If further exploration is needed, the LLM provides the next URL to visit, which is then fed into the scraper to repeat the process.

applications and dashboards, uncommon IoT devices and ICS, and errors and stack traces. We use a Censys snapshot from November 3, 2025, which captures 20 million hosts serving HTTP on at least one port. Using the ML filtering method, we extract 238,252 potential directory listings, 112,176 dashboards, 124,100 uncommon IoT devices, and 128,999 potential pages with error/stack traces. We then de-duplicate and process these pages with our LLM pipeline; 27–45% of pages remain by category. Additionally, we leverage LLMs' capability of handling different input modalities by studying a fifth category: camera feeds. Because this last category has much less data (Censys finds only 4.1M RTSP streams in total), we do not need to apply any pre-filtering.

In what follows, we detail how our pipeline analyzes content from each category, demonstrating the types of exposures it surfaces and their severity. For most categories, we can directly analyze the results from triaging the pages. For directory listings and RTSP camera feeds, we describe additional steps needed to collect the data (e.g., scraping web pages or downloading camera snapshots). Tables 4 and 5 provide a summary of the results.

5.1 Open Directory Listings

We begin by showing that LLMs can effectively uncover open directories containing sensitive data. Most directory listings are benign: by default, many HTTP servers provide a directory listing when there is no `index.html` file provided. Other directory listings, however, are the result of misconfigured servers, and expose sensitive data. This is nearly impossible to detect using conventional tools because directories might contain any type of data. Using manual analysis, however, a human can rather quickly get an idea of whether a directory listing is showing `.html` files or, for example, copies of one's passport. In what follows, we show how LLMs possess a similar "intuition" for making this distinction, and enable at-scale triage of exposed directory listings.

5.1.1 Analysis Pipeline. We design an LLM-guided scraper that, rather than performing a conventional breadth/depth-first exploration, relies on the LLM to determine which pages to visit. While directory names often reveal what type of content to expect inside the folder, diversity in naming conventions and semantics makes it infeasible to manually craft a set of keywords that indicate the presence of sensitive information. An LLM, however, has a much better intuition given its affinity with natural language. As such, instead of visiting all folders in a directory listing, we present the choice of folders to an LLM and ask which one is the most interesting to explore further (with the possibility to backtrack).

Figure 3 depicts the workflow of our scraper. For each directory listing, we fetch the index page and ask the LLM to categorize the severity of the exposed content according to a rubric similar to that used in Section 3. For instance, critical data may include personal files/documents/photos, large databases, financial records, medical records, etc.

After analyzing the page, we ask the LLM whether it has enough information to make a determination about the directory listing, or if it needs to explore the directory further. In case of the latter, the LLM will respond with the URL of the next folder to analyze. We then verify that this URL is: (1) present on the web page, and (2) still on the same IP address as the previous URL. Doing so, we avoid the LLM generating bogus URLs, malicious requests, or URLs outside of the directory listing. After obtaining the new URL, we repeat the process until the LLM decides it has enough information to reach a conclusion about the directory listing. We then save this conclusion and proceed with the next directory listing.

5.1.2 Results. We run our scraping pipeline on 234,188 directory listings identified by our conventional ML model. 90,871 responded with an HTTP status code outside the `2xx` range, yielding no page for the LLM to analyze. Additionally, 53,939 hosts were unresponsive or had other connection errors. We, thus, end up with 89,378 (38.2%) analyzed directory listings.

As expected, the majority of directory listings do not expose sensitive information (91.5%). These are directory listings showing empty files or folders or `.html` files without sensitive data. Without automated triage, one has to manually vet through 81,749 innocuous web pages to find the 7,629 that contain sensitive data. 7,629 open directories exposed some form of sensitive data (Table 5). The majority (almost 60%) belong to the "medium" category, and expose mostly source code directories that leak internal system configuration such as database backends, network layouts, or admin interfaces. The "high" and "critical" categories capture the most worrisome content and account for roughly 30% of the exposed directory listings. They show government audit documents, employment contracts, patient hospital records, SSH keys, AWS keys, database backups, and payment logs.

Figure 4 shows the top 100 ports hosting an unintentionally exposed directory listing. The default HTTP(S) ports (80 and 443) serve 1,638 of these instances, which accounts for 21.5% of all exposed directory listings. That is, almost 80% reside on unexpected ports, which are prohibitively hard to scan for [29]. Combined with the hidden nature of these services (our critical examples account for less than 3% of all scanned directory listings), makes this problem much akin to searching for a needle in a haystack. Our

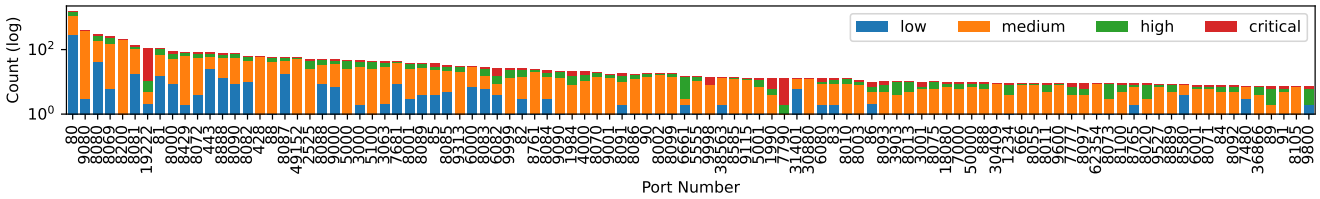


Figure 4: Top 100 Ports Hosting Risky Directory Listings. The default HTTP(S) ports (TCP/80 and TCP/443) account for 21.5% of directory listings, demonstrating that most exposure resides on non-standard ports.

Table 5: Unintentionally Exposed Directory Listing Severity. While the majority of services have the "medium" severity level, we find that almost 30% contain "high" or even "critical" levels of unintentionally exposed data.

Severity Level	Number	Examples of Exposed Data
Critical	633 (8.3%)	Employment contracts, hospital patient files, AWS keys, passport copies, bankruptcy case documents (including email exchanges with lawyers, expert reports, etc.), SSH keys, `.mp3` files of phone calls (with phone numbers and timestamps), payment logs, directory of car registrations and driver licenses, server backups, SQL database backups, employee time-tracking logs.
High	1,632 (21.4%)	Government audit documents, Virtual Machine (VM) images, system configuration files, machine file access, picture albums of families, internal business software, point-of-sale system and logs, computer backup, pirated movies and TV shows, excel report sheets, firmware images.
Medium	4,513 (59.1%)	Pictures of invoices, server configuration file, `phpinfo()` file, source code directories with some files exposing internal settings and system configurations.
Low	851 (11.2%)	System information files, large data dumps or backups (directory with 100+ files using structured naming conventions), source code directories, school assignments.

approach, however, found all these services in less than 38 hours, and scraped on average 4.8 pages per directory listing. LLMs, thus, scale the discovery and triage of sensitive content in exposed directory listings. We break down our findings by security level in Table 5 and discuss several specific cases below.

Vehicle Licensing and Registration Certificates. Two directory listings—both hosted within the same AS—hosted copies of vehicle registration documents from Brazil. Each document shows the car model, license plate, owner, and *"Cadastro de Pessoas Físicas"* which is the Brazilian equivalent of the American SSN. All documents are timestamped within the last 6 months, suggesting that this is either a new "database" or that documents are periodically removed. These documents could enable identity fraud or targeted phishing attacks.

Customer Support Calls. One directory listing contained an organized folder structure storing 200+ `.mp3` and `.wav` files with recorded phone calls from clients to a Russian tech support system. The timestamps on the files are all between 9am and 5pm, indicating a business-hour pattern. The filenames also contain the phone number that initiated the call. The content of the phone calls reveals personal information about clients, such as purchase information and delivery addresses. Adversaries could extract such personal information from the calls, and use it to either scam the company behind the support system (by impersonating as a legitimate client and asking for shipments to be delivered to a different address) or scam the customer (by impersonating as the business and asking for payment or other personal details).

GoTAK Positioning Data. GoTAK is a situational awareness company that uses Team Awareness Kits (TAK) to provide geospatial infrastructures for coordination efforts among teams. Their equipment is mostly used by the military and law enforcement. We found a directory listing hosting location data (using `.kml` files) from a GoTAK server. The exposure of this data could potentially compromise the location of equipment and individuals, and have impact on public safety. Besides location data, the directory listing also stores private keys of users.

5.2 Business Applications and Dashboards

The second category of exposed service we investigate is business applications and dashboards. These services display unauthenticated interfaces or dashboards providing access to or insights into company infrastructures. The pages vary significantly in terms of design, control flow, language, and, most importantly, what they measure or control. In our data, we found 24,500 (21.8%) publicly accessible applications and dashboards that expose sensitive data. The majority (69.2%) fall into the "medium" classification. 29%, however, are considered "high" or "critical" and expose dashboards showing user requests to news feeds, customer reports with detailed financing/loan records, crontab managers, or phpMyAdmin interfaces. We discuss three in more detail:

News Feed Server Dashboard. One instance shows an HTTP server dashboard from what appears to be a news feed service. The dashboard displays all incoming requests of users to each of the different news feeds. The counter shows 804,173 requests being displayed. The web page also shows a table with currently active connections (36 at the moment of capture), showing the requested feed, but also the username and password, all of which are displayed in plain text. Besides directly leaking private information such as username and password, exposing people's news interest could reveal other personal information such as political views, religious beliefs, or sexual orientation.

Development Dashboard for Healthcare Software. We found an instance that shows a dashboard giving administrative access to a healthcare management system. The page displays links to an HIV and Diabetes portal, medical officer interfaces, and patient applications. On the development side, it gives access to tools such as Jenkins build environments, Docker, Redis, and a Postgres admin panel. The dashboard allows the user to stop containers, and provides real-time logs and statistics that refresh every five seconds.

phpMyAdmin Dashboard with Government Database. Lastly, we found a phpMyAdmin dashboard from the government of Zimbabwe that provides access to a database labeled `governmentdb` showing records of government employees, including emails and employee questionnaire answers.

5.3 IoT Devices and Industrial Control Systems

Next, we look at exposed IoT devices and industrial control systems (ICS) interfaces. From the 124,100 IoT/ICS interfaces, 12,022 (9.7%) exposed sensitive information. Below, we discuss two examples.

Chinese Control Systems. We found an ICS dashboard providing an overview of 100 electric and water utility meters. For each meter, the dashboard provides their address, the address of the contractor, meter type, status, and usage. More worryingly, each meter displays several control buttons, one of which translates to "close circuit/close valve," suggesting direct control over units.

Russian Gas Turbine. One dashboard displays an interactive overview of a Russian gas turbine, showing different sensors and their values. The diagram reports on temperatures, pressure, flow rates, etc. Additionally, the page links to 11 other gas turbines displaying the same telemetry for different units.

5.4 Application Errors and Stack Traces

For the fourth category, we investigate pages with errors or stack traces. Programming errors can lead to application crashes, which often leave behind a "stack trace" of the function call(s) that caused the crash. For the application developer, it is important to have as much detail as possible in the stack trace to enable debugging. However, exposing such details in production environments can leak sensitive information. For example, if code contains hard-coded database credentials, and a crash occurs during a database operation, the stack trace might leak those credentials.

Many error pages and stack traces provide little to no sensitive information, and likely display just a general error message. This makes it time consuming and impractical to triage at scale. LLMs, however, can easily distinguish meaningless error messages from leaked internal information. Note that we only include stack traces that appear when visiting the index page of a website. A more aggressive approach would be to actively trigger an error in the code (e.g., by providing invalid input), making the page crash and potentially show a stack trace containing sensitive information. For ethical reasons, we avoid causing unnecessary load on our targets, and refrain from trying this approach.

In total, we find 20,612 (38%) pages showing an error/stack trace that reveals sensitive information such as infrastructure details or even login credentials. In contrast to the other categories, error/stack traces contain less variety in the data they expose. We are less likely to find information such as temperature measurements or personal addresses compared to, for example, IoT panels and directory listings, and obtain mostly login credentials and (internal) endpoint locations. We discuss two case studies below.

Django Debug Trace. The Python web framework Django contains the `DEBUG` parameter which, if set to `True`, causes the application to display a detailed error page when encountering an error. It will do so in lieu of a standard HTTP error code with no content body. The framework has built-in mechanisms to obfuscate secret information such as API keys and passwords [17]. However, if the user's keys are directly passed as a `String` to the function causing the error, the debug trace will display the credentials nonetheless. The framework, therefore, explicitly advises to not have debug mode enabled for pages in production.

ThinkPHP Error Page. One ThinkPHP error page caused by a call to a non-existent controller contains a "Server/Request Data" section which includes sensitive information such as database credentials, WeChat API credentials, and an Alibaba Cloud access key. The error also displays the PHP version number and the ID of the Docker container in which the application is running. The page indicates that debug mode is enabled.

5.5 RTSP Camera Feeds

The Real-Time Streaming Protocol (RTSP) is an application-layer protocol used for streaming and controlling media [48]. The applications of RTSP are diverse, but the protocol is generally known for its use in IP cameras. While it is easy to scan for available RTSP endpoints, it is much harder to determine whether the streamed media contains sensitive information (e.g., private homes), or public information (e.g., traffic cameras), without manual analysis. We demonstrate, however, that LLMs can now scale such an endeavor, making it feasible to discover private camera streams at scale.

5.5.1 Methodology. We begin by fetching all RTSP endpoints found by Censys on October 30, 2025, providing 4.1M hosts. We attempt to connect over RTSP to each service, and fetch one frame of streamed video. Over 99% of RTSP hosts have either become unresponsive since scanned by Censys, or require authentication, inhibiting us from retrieving any media. 10,450 hosts responded with a valid `.jpg` file. For each of these hosts, we send the image to GPT-5 and prompt the model to summarize the scene and reason about whether the feed appears unintentionally exposed or contains sensitive content. If the model marks an endpoint as exposed, it must additionally assign one of four severity levels (low, medium, high, or critical) according to a rubric that distinguishes private home and medical interiors, offices and semi-public spaces, and clearly public or generic views, and justify this choice in free text. The rubric explicitly instructs the model to down-rank intentionally public webcams (e.g., traffic or weather cameras) and configuration or error screens, which we discard from further analysis.

We repeat this process every hour for the 10,450 hosts over one week. For each host, we collect the results of each of the hourly probes, and ask the model to analyze all the previous labels for each stream and make a summary about what the camera displays and the criticality of the exposed image. Because our summary stems from week-long hourly snapshots, we can detect potential errors of

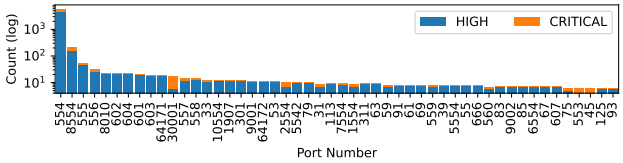
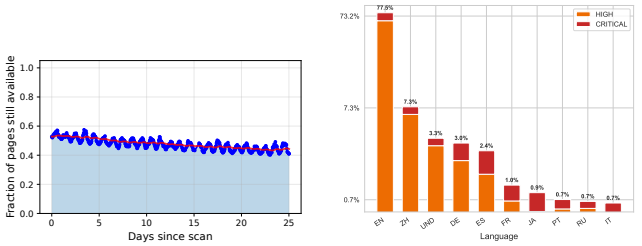


Figure 5: Top 50 Ports Hosting the Most Severe Unintentionally Exposed RTSP services. The RTSP standardized port (554) accounts for 74.2% of the entries, contrasting HTTP where most exposed services use non-standard ports.



(a) Exposed Services Over Time **(b) Exposed Content Language**

Figure 6: Exposure Characteristics. (a) Once a service has been exposed for at least one day, it is likely to remain online and unchanged for several weeks. (b) We discover content over a wide range of languages; because pages can contain multiple languages, results do not sum to 100%.

the LLM. That is, if an LLM were to label an image incorrectly (i.e., describing a scene that is not present in the image), the summary would notice such inconsistent descriptions, enabling us to discard such scenes. Indeed inconsistent labels are ranked lower compared to images that consistently show sensitive content.

5.5.2 Results. Contrary to the other categories, we found much more severe cases among RTSP feeds: 74.6% of RTSP camera feeds show "high" or "critical" levels of sensitive data. The port number distribution is also different compared to the HTTP services: 74.2% of the "high" and "critical" services use the default port (Figure 5). We note that 516 hosts (4.9%) stopped responding to our probes during the course of our week-long study, suggesting that either the camera was taken down, or that our scanning machine got blocked. We found many concerning examples of sensitive camera feeds. We note that none of these images were stored on any machine; we only save the analysis output of the LLM. As such, no author has actually seen any of the images to preserve the privacy of the people exposed in the video feeds (see Section 8 for details on the ethics of our research).

Private Bedrooms and Bathrooms. We find feeds of private bedrooms and bathrooms, which can potentially capture people in their most intimate setting. Since we scan hourly, we indeed find many snapshots where the LLM indicated that people are actually present in the room.

Childcare and Educational Facilities. We find several instances of surveillance cameras that directly (as well as in the periphery) monitor childcare facilities and other educational institutions (such as kindergartens). We capture images described by the LLM as "parents picking up their children" or "children playing."

Private Residential Areas. We found several cases of private surveillance cameras showing people's living rooms, kitchens, pools, backyards, etc. Again, due to our hourly scans, there was always at least one snapshot that captured people within the residential area under surveillance.

Hospital and Medical Care Facilities. One camera feed observes a ward room containing several patient beds. The room displays medical equipment such as IV stands, wheelchairs, and informational posters on the wall. In several snapshots, the LLM describes a caregiver providing care to (partially unclothed) patients.

6 Further Observations

Below, we discuss several observations from our analysis:

Exposure Length. The attack window to extract sensitive information varies widely. To measure how long sensitive information remains exposed, we download "high" and "critical" services daily and measure whether the sensitive content remains accessible. As seen in Figure 6a, nearly half of services become unavailable after the very first scan, but if they remain, they often will for weeks. Note, however, that a brief window of opportunity is often enough for an adversary to download leaked information [32, 44].

Language Distribution. For each high and critical severity endpoint, we infer the page's language (Figure 6b). Overall, 22.5% of these pages use a language other than English. This highlights an advantage of LLM-based analysis: a human analyst who only speaks English would miss nearly one quarter of critical exposures. For example, on a server containing over 200 `.mp3` files, the LLM recognized Russian phone numbers in the filenames and business-hour timestamps, correctly inferring that the files were customer-call recordings. In another directory listing, it identified a *Cadastro de Pessoas Físicas*, a Brazilian taxpayer identifier analogous to a U.S. Social Security number.

Estimated Cost. We roughly estimate the cost of our unoptimized implementation to understand its feasibility, both as a defensive tool and as a potential attack vector. Without any filtering (i.e., the techniques we developed in Section 4), analyzing one million web pages would cost approximately $20,000 USD. Having once performed this analysis, it then becomes possible to train the small and efficient classifiers. Once we have done this, we can easily reduce the analysis cost by $10 - 100\times$ depending on the desired FPR/TPR ratio. However, by applying cheaper language models, multiple stages of pipelining, or fine-tuning specialized models, it is likely possible that this cost could be further reduced significantly.

For a defender, auditing an organization's perimeter may already be economically practical depending on the size of the organization and frequency of audits. For an adversary, the current cost remains a barrier, but given that language model inference costs dropped year-over-year, this cost may soon be within reach.

7 Responsible Disclosure

We notified operators of the most severely misconfigured services. We searched for responsible contacts through WHOIS records,

abuse contacts, and, where available, contact information on websites themselves. In the small number of cases where we found a point of contact, none took action to resolve the data exposure.

Identifying responsible contacts proved to be a significant challenge. Often no contact information is available, and when there is, most of our requests were ignored. This is a fundamental tension: our pipeline surfaces *thousands* of sensitive endpoints, but traditional disclosure where a human contacts each affected operator individually does not scale to this volume. One natural option is to use LLMs (e.g., to identify the responsible party and draft a notification), but this carries its own risks: if the model misattributes ownership, we end up notifying the wrong entity. Future work should, therefore, explore how to scale this process using LLMs, without introducing new harms.

8 Ethical Considerations

Our study analyzes only public Internet services and we design our methodology to minimize unnecessarily downloading sensitive content. We begin by analyzing Censys data rather than conducting our own Internet scans [21]. We use LLMs to selectively access only the pages needed for classification rather than exhaustively crawling every service. On average, we visit 4.8 pages per service, limit directory listings to 10 requests per server, and query RTSP camera feeds only once per hour. Following established best practices [11, 20], every HTTP request includes an identifying `User-Agent` with contact information, and the scanning IP hosts a web page describing the study and its opt-out procedure. We received no opt-out requests. For the RTSP camera feeds, we never store any images on our machines and retain only the structured outputs from the LLM. The terms attached to our API access to OpenAI and Anthropic prohibit using scan inputs for model training [42].

We argue that publication provides defenders with timely insight into an approach that adversaries could independently develop, particularly while its cost still limits widespread use. Nevertheless, we do not release the collected dataset because it identifies specific vulnerable services and could facilitate exploitation. We provide a representative prompt and complete severity rubric in Appendix A, but withhold the full prompt set and pipeline code because together they form a near-turnkey scanning system. To support reproducibility and defensive research, we will share data and code with verified researchers.

9 Related Work

Internet-wide scanning has enabled researchers to analyze many classes of Internet exposure (e.g., [8, 9, 15, 33, 37, 38, 45, 47]). For example, Karl et al. [31] constructed checks for missing authentication across 25 administrative applications and measured vulnerability across the IPv4 address space. Beyond Internet scanning data, several works have also investigated data exposed in cloud storage buckets, which share many characteristics with the data exposed in open directories [12, 24]. Commercial scanning platforms annotate Internet-facing services using banner- and payload-based detections [21]. Huang et al. [27] show that these approaches can both miss vulnerable endpoints and produce false positives. These methods are effective when the analyst knows which service or vulnerability to search for, but fundamentally rely on predefined fingerprints, probes, templates, or application-specific logic.

Our work builds on several prior studies that have used LLMs for scoped security analyses. Ng and Mehrnezhad used ChatGPT to study IP cameras found by Shodan [40]. Rather than consulting a specific image model, they fed screenshots into ChatGPT for analysis. While similar in nature, their study is limited to 281 camera feeds, and provides no longitudinal analysis of the footage and severity levels. Moreover, they task ChatGPT to limit its analysis to a specific question: whether people are present in the image. Earlier work on IP cameras focused on network traffic patterns rather than the camera images themselves [5, 35, 49, 54].

We relied on an LLM-guided web crawler to further explore the directory listings in Section 5.1. Recently, several works have introduced ML-based approaches to web crawling, including Multi-Armed-Krawler (MAK), a web crawler that uses Reinforcement Learning (RL) to maximize page coverage [13]. In contrast, with our LLM-guided approach, we focus less on achieving maximal coverage, but rather focus on detecting unintentionally exposed content with a small number of page visits. Related to our approach is YuraScanner, an agent-based scanner capable of evaluating target websites for XSS vulnerabilities [51]. However, in YuraScanner, the LLM is capable of executing tasks on the website. In contrast, we limit our LLM to only *suggest* URLs to visit, which enables us to safely run it against public sites.

10 Conclusion

In this work, we showed that LLMs can identify misconfigured Internet services that have previously gone undetected by both defenders and adversaries. Across five common types of exposure, we demonstrated that LLMs can identify, categorize, and severity-rank thousands of misconfigured services without predefined fingerprints. Our work acts as a warning: as large language models continue to improve and drop in cost, the ability of adversaries to find security-critical misconfigurations across the long tail of services on the Internet will likely become a popular means of gaining initial access to, extorting, or otherwise attacking organizations.

Acknowledgments

This work was supported by a gift from Google, Inc. and the National Science Foundation under Grant Numbers #2319080 and #2540803. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding organizations.

References

[1] Khalifa Afane, Wenqi Wei, Ying Mao, Junaid Farooq, and Juntao Chen. 2024. Next-generation phishing: How LLM agents empower cyber attackers. In *IEEE International Conference on Big Data*.

[2] Taha Albakour, Oliver Gasser, Robert Beverly, and Georgios Smaragdakis. 2021. Third time's not a charm: exploiting snmpv3 for router fingerprinting. In *ACM internet measurement conference*.

[3] Taha Albakour, Oliver Gasser, Robert Beverly, and Georgios Smaragdakis. 2023. Illuminating router vendor diversity within providers and along network paths. In *ACM Internet Measurement Conference*.

[4] Sam Altman. 2025. *Three Observations*. Accessed: 2025-11-13. https://blog.samaltman.com/three-observations

[5] Jay Anand, Arunan Sivanathan, Ayyoob Hamza, and Hassan Habibi Gharakheili. 2021. PARVP: passively assessing risk of vulnerable passwords for HTTP authentication in networked cameras. In *Workshop on Descriptive Approaches to IoT*

Security, Network, and Application Configuration.

[6] Anthropic. 2025. Claude Sonnet 4.5 System Card. https://assets.anthropic.com/m/12f214efcc2f457a/original/Claude-Sonnet-4-5-System-Card.pdf.

[7] Anthropic. 2025. Disrupting the first reported AI-orchestrated cyber espionage campaign. https://www.anthropic.com/news/disrupting-AI-espionage.

[8] Manos Antonakakis, Tim April, Michael Bailey, Matthew Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J. Alex Halderman, Luca Invernizzi, Michalis Kallitsis, Deepak Kumar, Chaz Lever, Zane Ma, Joshua Mason, Damian Menscher, Chad Seaman, Nick Sullivan, Kurt Thomas, and Yi Zhou. 2017. Understanding the mirai botnet. In *USENIX Security Symposium*.

[9] Ioannis Arakas, Panagiotis Pallis, Evangelos Markatos, and Georgios Smaragdakis. 2026. The Last of the Apaches: Investigating the State of Internet-facing End-of-Life Software. In *European Workshop on Systems Security*.

[10] Daniel Arp, Erwin Quiring, Feargus Pendlebury, Alexander Warnecke, Fabio Pierazzi, Christian Wressnegger, Lorenzo Cavallaro, and Konrad Rieck. 2022. Dos and Don'ts of Machine Learning in Computer Security. In *31st USENIX Security Symposium (USENIX Security 22)*. USENIX Association, Boston, MA, 3971–3988. https://www.usenix.org/conference/usenixsecurity22/presentation/arp

[11] Michael Bailey, David Dittrich, Erin Kenneally, and Doug Maughan. 2012. The Menlo Report. *IEEE Security & Privacy* (March 2012). doi:10.1109/MSP.2012.52

[12] Jack Cable, Drew Gregory, Liz Izhikevich, and Zakir Durumeric. 2021. Stratosphere: Finding vulnerable cloud storage buckets. In *International Symposium on Research in Attacks, Intrusions and Defenses*.

[13] Lorenzo Cazzaro, Stefano Calzavara, Maksim Kovalkov, Aleksei Stafeev, and Giancarlo Pellegrino. 2025. Less is More: Boosting Coverage of Web Crawling through Adversarial Multi-Armed Bandit. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[14] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, Luke Marris, Sam Petulla, Colin Gaffney, Asaf Aharoni, Nathan Lintz, Tiago Cardal Pais, Henrik Jacobsson, Idan Szpektor, Nan-Jiang Jiang, Krishna Haridasan, Ahmed Omran, Nikunj Saunshi, Dara Bahri, Gaurav Mishra, Eric Chu, Toby Boyd, Brad Hekman, Aaron Parisi, Chaoyi Zhang, et al. 2025. Gemini 2.5: Pushing the Frontier with Advanced Reasoning, Multimodality, Long Context, and Next Generation Agentic Capabilities. *arXiv:2507.06261* (2025).

[15] Markus Dahlmanns, Johannes Lohmöller, Ina Berenice Fink, Jan Pennekamp, Klaus Wehrle, and Martin Henze. 2020. Easing the conscience with OPC UA: an internet-wide study on insecure deployments. In *ACM Internet Measurement Conference*.

[16] Nurullah Demir, Yash Vekaria, Georgios Smaragdakis, and Zakir Durumeric. 2026. Keys on Doormats: Exposed API Credentials on the Web. *ACM Conference on Computer and Communications Security (CCS)* (2026).

[17] Django. 2026. Settings | Django Documentation. https://docs.djangoproject.com/en/dev/ref/settings/#debug.

[18] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. 2015. A search engine backed by Internet-wide scanning. In *ACM SIGSAC conference on computer and communications security*.

[19] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J Alex Halderman. 2015. Neither snow nor rain nor MITM... an empirical analysis of email delivery security. In *ACM Internet Measurement Conference*.

[20] Zakir Durumeric, David Adrian, Phillip Stephens, Eric Wustrow, and J. Alex Halderman. 2024. Ten Years of ZMap. In *ACM Internet Measurement Conference*.

[21] Zakir Durumeric, Hudson Clark, Jeff Cody, Elliot Cubit, Matt Ellison, Liz Izhikevich, and Ariana Mirian. 2025. Censys: A Map of Internet Hosts and Services. In *ACM SIGCOMM 2025 Conference*.

[22] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, and Alex Halderman. 2014. The matter of heartbleed. In *ACM Internet Measurement Conference*.

[23] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast internet-wide scanning and its security applications. In *USENIX Security Symposium*.

[24] Soufian El Yadmani, Olga Gadyatskaya, and Yury Zhauniarovich. 2025. The file that contained the keys has been removed: An empirical analysis of secret leaks in cloud buckets and responsible disclosure outcomes. In *IEEE Symposium on Security and Privacy*.

[25] Sergei Glazunov and Mark Brand. 2024. Project Naptime: Evaluating Offensive Security Capabilities of Large Language Models. https://googleprojectzero.blogspot.com/2024/06/project-naptime.html. Google Project Zero Blog.

[26] Google Threat Intelligence Group. 2025. GTIG AI Threat Tracker: Advances in Threat Actor Usage of AI Tools. https://cloud.google.com/blog/topics/threat-intelligence/threat-actor-usage-of-ai-tools. Google Cloud Security Blog.

[27] Szu-Chun Huang, Harm Griffioen, Max van der Horst, Georgios Smaragdakis, Michel van Eeten, and Yury Zhauniarovich. 2025. Trust but Verify: An Assessment of Vulnerability Tagging Services. In *USENIX Security Symposium*.

[28] ISC2. 2025. 2025 ISC2 Cybersecurity Workforce Study. https://www.isc2.org/Insights/2025/12/2025-ISC2-Cybersecurity-Workforce-Study

[29] Liz Izhikevich, Renata Teixeira, and Zakir Durumeric. 2021. LZR: Identifying Unexpected Internet Services. In *USENIX Security Symposium*.

[30] Liz Izhikevich, Renata Teixeira, and Zakir Durumeric. 2022. Predicting IPv4 services across all ports. In *ACM SIGCOMM*.

[31] Manuel Karl, Marius Musch, Guoli Ma, Martin Johns, and Sebastian Lekies. 2022. No keys to the kingdom required: a comprehensive investigation of missing authentication vulnerabilities in the wild. In *ACM Internet Measurement Conference*.

[32] Brian Kondracki, Michael Ferdman, and Nick Nikiforakis. 2024. Ready or Not, Here I Come: Characterizing the Security of Prematurely-public Web Applications. In *2024 Annual Computer Security Applications Conference (ACSAC)*. IEEE.

[33] Deepak Kumar, Kelly Shen, Benton Case, Deepali Garg, Galina Alperovich, Dmitry Kuznetsov, Rajarshi Gupta, and Zakir Durumeric. 2019. All things considered: An analysis of IoT devices on home networks. In *USENIX security symposium*.

[34] Frank Li, Zakir Durumeric, Jakub Czyz, Mohammad Karami, Michael Bailey, Damon McCoy, Stefan Savage, and Vern Paxson. 2016. You've got vulnerability: exploring effective vulnerability notifications. In *USENIX Security Symposium*.

[35] Jinyang Li, Zhenyu Li, Gareth Tyson, and Gaogang Xie. 2020. Your Privilege Gives Your Privacy Away: An Analysis of a Home Security Camera Service. In *IEEE Conference on Computer Communications*.

[36] Mandiant. [n. d.]. M-Trends 2026 Report. https://cloud.google.com/security/resources/m-trends

[37] Ariana Mirian, Zane Ma, David Adrian, Matthew Tischer, Thasphon Chuenchujit, Tim Yardley, Robin Berthier, Joshua Mason, Zakir Durumeric, J Alex Halderman, et al. 2016. An internet-wide view of ICS devices. In *Conference on Privacy, Security and Trust (PST)*.

[38] Martin Mladenov, Laszlo Erdodi, and Georgios Smaragdakis. 2025. All that glitters is not gold: Uncovering exposed industrial control systems and honeypots in the wild. In *IEEE European Symposium on Security and Privacy*.

[39] Cristian Munteanu, Georgios Smaragdakis, Anja Feldmann, and Tobias Fiebig. 2025. Catch-22: uncovering compromised hosts using SSH public keys. In *USENIX Security Symposium*.

[40] Cheok Ieng Ng and Maryam Mehrnezhad. 2024. Security and Privacy Evaluation of IP Cameras on Shodan. In *Conference on Research in Security Standardisation*.

[41] OpenAI. 2025. GPT-5 System Card. https://openai.com/index/gpt-5-system-card/.

[42] OpenAI. 2026. Enterprise privacy at OpenAI. https://openai.com/enterprise-privacy/.

[43] Palo Alto Networks. [n. d.]. Global Incident Response Report 2026. https://www.paloaltonetworks.com/resources/research/unit-42-incident-response-report

[44] Stijn Pletinckx, Thanh-Dat Nguyen, Tobias Fiebig, Christopher Kruegel, and Giovanni Vigna. 2023. Certifiably Vulnerable: Using Certificate Transparency Logs for Target Reconnaissance. In *IEEE European Symposium on Security and Privacy*.

[45] Said Jawad Saidi, Anna Maria Mandalari, Roman Kolcun, Hamed Haddadi, Daniel J Dubois, David Choffnes, Georgios Smaragdakis, and Anja Feldmann. 2020. A haystack full of needles: Scalable detection of IoT devices in the wild. In *ACM Internet Measurement Conference*.

[46] SANS. 2025. SANS 2025 SOC Survey. https://www.sans.org/white-papers/sans-2025-soc-survey

[47] Takayuki Sasaki, Akira Fujita, Carlos H Gañán, Michel van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. 2022. Exposed infrastructures: Discovery, attacks and remediation of insecure ics remote management devices. In *IEEE Symposium on Security and Privacy*.

[48] Henning Schulzrinne, Anup Rao, Rob Lanphier, Magnus Westerlund, and Martin Stiemerling. 2016. Real-Time Streaming Protocol Version 2.0. RFC 7826.

[49] Yogeesh Seralathan, Tae Tom Oh, Suyash Jadhav, Jonathan Myers, Jaehoon Paul Jeong, Young Ho Kim, and Jeong Neyo Kim. 2018. IoT security vulnerability: A case study of a Web camera. In *International Conference on Advanced Communication Technology*.

[50] Drew Springall, Zakir Durumeric, and J Alex Halderman. 2016. FTP: The forgotten cloud. In *IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*.

[51] Aleksei Stafeev, Tim Recktenwald, Gianluca De Stefano, Soheil Khodayari, and Giancarlo Pellegrino. 2025. YuraScanner: Leveraging LLMs for Task-driven Web App Scanning. In *32nd Annual Network and Distributed System Security Symposium, NDSS, 2025, San Diego, California, USA, February 24-28, 2025*. The Internet Society. doi:10.14722/ndss.2025.240388

[52] Ben Stock, Giancarlo Pellegrino, Frank Li, Michael Backes, and Christian Rossow. 2018. Didn't You Hear Me?—Towards More Successful Web Vulnerability Notifications. (2018).

[53] Zhun Wang, Tianneng Shi, Jingxuan He, Matthew Cai, Jialin Zhang, and Dawn Song. 2025. CyberGym: Evaluating AI Agents' Real-World Cybersecurity Capabilities at Scale. arXiv:2506.02548 [cs.CR] https://arxiv.org/abs/2506.02548

[54] Haitao Xu, Fengyuan Xu, and Bo Chen. 2018. Internet Protocol Cameras with No Password Protection: An Empirical Investigation. In *Passive and Active Measurement (PAM)*.

[55] Andy K Zhang, Neil Perry, Riya Dulepet, Joey Ji, Celeste Menders, Justin W Lin, Eliot Jones, Gashon Hussein, Samantha Liu, Donovan Julian Jasper, Pura Peetathawatchai, Ari Glenn, Vikram Sivashankar, Daniel Zamoshchin, Leo Glikbarg, Derek Askaryar, Haoxiang Yang, Aolin Zhang, Rishi Alluri, Nathan Tran, Rinnara Sangpisit, Kenny O Oseleononmen, Dan Boneh, Daniel E. Ho, and Percy Liang. 2025. Cybench: A Framework for Evaluating Cybersecurity Capabilities and Risks of Language Models. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=tc90LV0yRL

A Example Prompt

We use many LLM prompts in this work. While sharing all of these is likely redundant, below we provide an example of one of these prompts. We will make our pipeline available to other researchers who are interested in extending this work.

```
Analyze this HTML response and summarize the content on the page in one paragraph. Explain this information in sufficient detail that if someone was not able to read the page, they would be able to understand exactly what is going on given your description.

Then, determine if this page was unintentionally exposed on the Internet, or if it leaks data.

Do not include pages that are: - Generic template pages that are likely to be repeated on the internet many times (e.g., generic or custom 404 pages, 500 pages that do not expose any sensitive data) - Login boxes or input fields that do not expose data (e.g., router login pages, admin login pages). By default assume password-protected pages have strong passwords. - Normal web servers hosting standard content - Blogs, personal homepages, company webpages advertising phone numbers for public use, or anything intended for public use.

Please respond with an output formatted as XML: <summary>description of the page content</summary> <yes_considerations>any considerations that imply the content is sensitive</yes_considerations> <no_considerations>any considerations that imply the content is not sensitive</no_considerations> <exposed>yes/no</exposed>

You can stop here if the answer is no. There is nothing further to do. You should not continue. ONLY IF you respond exposed is yes, then your next task is to categorize the severity as CRITICAL, HIGH, MEDIUM, or LOW. If exposed is no then abort early.

CRITICAL are similar to one of the following: - Actual personal files, documents, or photos that are clearly visible - Large database contents with real personal information (names, addresses, SSNs, etc.) - Private email contents or messaging systems with visible messages
- Financial records, medical records, or other highly sensitive documents
- Directory listings containing personal files with sensitive names
- Directory listings that contain large files that could be highly sensitive
- Real-time location information
- Home automation systems showing personal schedules, camera feeds, or location data without password authentication
- IoT devices that allow harmful state-changing operations (update firmware, change password, anything that makes physical-world changes like turn on/off motor/valve, unlock door, open/close, turn on/off lights, PTZ camera operations)
- Misconfigured IoT devices that leak the username/password at the login page

HIGH pages are like critical pages but are not as extreme: - Directory listings that appear to potentially include content that is damaging - Files that apear to be generic in nature (e.g., promotional images) - IoT devices that are not well-known (e.g., custom IoT devices for domain-specific applications); if you know details about this device, then it's not HIGH. - Routers that only expose WiFi passwords, SSIDs, or MAC addresses. - Pages with PII that's only names or email addresses, but nothing more sensitive (like SSNs or financials) - Any webpage that has non-default passwords (e.g., is not 'password' or 'admin' or 'root') - IoT devices that allow state-changing operations (restart, disconnect wireless, change name)

MEDIUM pages are those that don't severe content but are still harmful: - IoT devices that show potentially sensitive data without requiring login - Routers that expose default usernames&passwords.
- Pages that expose sensitive information about the current file system (e.g., logs that expose ENVIRONMENT or other sensitive data)
- Anything standard that you've seen before and has a password login page that's not covered above, e.g., regular routers that millions of people own and are likely to be patched and secure - Pages that disclose only a few internal IP addresses (like 192.168.0.1) without additional context. - Home automation systems that require login or show a default authentication page

LOW pages are those that contain anything else: - Empty directory listings or generic file browsers without visible sensitive content
- IoT pages that ask for login information - Login pages or authentication screens (even if misconfigured) - Error pages, default pages, or configuration pages - Public websites, blogs, or intentionally shared content - Generic server information or status pages - Pages that might contain sensitive data but don't actually show it - Administrative interfaces without actual data visible (like RabbitMQ, phpMyAdmin shells)

Include the following additional XML tags after the first XML tags you previously wrote down: <visible_data>specific sensitive data that is actually visible on the page; summarize the data and also provide exact quotes of the particular content that is exposed that should not be exposed</visible_data> <analysis_severity>analyze what this data means with regard to the rubric above; explain how each of the considerations for any of critical/high/medium apply to this particular context. do not make a decision here, ONLY evaluate how the criteria from the rubric apply</analysis_severity> <severity>CRITICAL/HIGH/MEDIUM/LOW</severity>
```

B Prediction Agreement in ML Classifiers

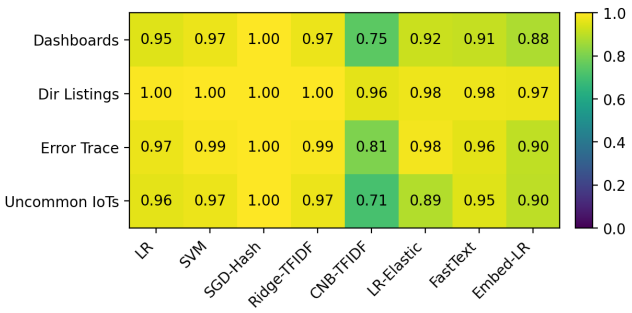


Figure 7: Pairwise prediction agreement between the best-performing model and all other classifiers, per dataset.

In Section 4, we examine how consistently the models agree on their predictions for each dataset, since substantial disagreement would mean that classifier choice could affect downstream security analyses. The pairwise agreement results in Figure 7 show that the best-performing model agrees closely with most TF-IDF-based linear models, with agreement typically between 0.94 and 1.00. CNB-TFIDF shows substantially lower agreement on Dashboards (0.75) and Uncommon IoTs (0.71). FastText shows moderate-to-high agreement (0.91–0.98) despite its near-chance ranking performance, because many binary predictions coincide under the strongly imbalanced class distribution. Embed-LR shows somewhat lower agreement than the strongest linear models, ranging from 0.88 to 0.97 across datasets.