ECHELONGRAPH · THREAT INTELLIGENCE

The State of Internet Exposure

2026 — Inaugural Edition · What the whole internet can already see

An evidence-based field report: candidates found in public data, then checked with identified requests of our own *(corrected 2026-09-27)* *(data-store check corrected 2026-09-29)* **— not a survey, not a forecast.**

BY THE NUMBERS — WINDOW 29 MAY – 28 JUNE 2026 *(window corrected 2026-09-29)*

2,050
AI services confirmed active & open
of 82,416 discovered; 2 MCP servers *(withdrawn 2026-09-29)*

21,299
hosts on CISA-KEV exploited vulns
of 36,416 CVE-exposed

6,607
open, unauthenticated databases
120 countries

2,571
exposed .env files
≈2,600 AWS keys

619
secrets in public Git repos
478 repositories

134
hijackable subdomains
dangling CNAME

340,552
CVEs tracked & enriched
1,621 CISA-KEV

137
countries observed
public data, then identified checks *(corrected 2026-09-27)*

✓ Certified by EchelonGraph

Observation window: 29 May – 28 June 2026 · **Generated:** 28 June 2026

All figures are derived from EchelonGraph's exposure radars, several from Shodan data. Shodan data is owned by Shodan, which holds its copyright (© Shodan). EchelonGraph claims no ownership of it or copyright in it. *(corrected 2026-09-27)* Aggregate & host-redacted.

For CISOs, CTOs & CEOs. Shareable. © 2026 EchelonGraph, for this report's text and compilation *(corrected 2026-09-27)* — echelongraph.io

Certification & Disclaimer

Correction, 2026-09-29: read this first. Every sentence of this report that describes how we found, checked, contacted, read or identified anything is superseded by the table what each radar sent, below. Where a chapter says more or less than the table, the table is right; the dated notes below add detail to it. This report's corrections are dated. Its figures are unchanged, except the 2 MCP servers, whose count is withdrawn in the MCP correction below.

Correction, 2026-09-27. This report counted 36,991 AI services as "authenticated" or "secured", and described every one of them as a service that had demanded a credential, sat behind a login or other access control, and was "working as intended". That is not a count of services we saw enforce authentication. When this data was collected, our verifier also filed a service in that bucket when it could *not* tell — the methodology chapter says so: "any inconclusive result is treated as authenticated" — for example a JavaScript-only app shell whose login state is decided in the browser (MinIO's web console is one), a deeper probe that matched neither an open nor a locked response, or a redirect chain that never reached a page. Those services answered, but we never saw them ask for credentials. Our verifier now records such a service as *inconclusive*, counted neither as exposed nor as protected, and this report now labels the 36,991 "auth observed or not determined". The observations from this window are no longer held, so we cannot say how the 36,991 split between the two, and we do not print a replacement number. This correction concerns that bucket only; each sentence it changed is marked "corrected 2026-09-27".

Correction, 2026-09-27: the method. This report describes its method as passive, detect-only and drawn from public data, and says that where a finding needed confirming we took exactly one action: "a single anonymous, read-only HTTP GET". That understates what we sent. The *candidates* came from public data, as described. Confirming them was not passive: our AI-service verifier sent several GETs to one host — its page, following redirects, then known API paths for that kind of service — and the exposed-configuration check requested a list of known configuration paths on one host and read the body of each exposed file, up to a size cap, which is how a leaked secret is detected (what it read was redacted before storage). Those requests carried our identity; they were not anonymous. None of them logged in, submitted a credential or exploited anything. Where

this report says “passive” or “passively”, read: candidates found in public data. Whether we then sent anything to a host differs by radar; the table in the next correction, what each radar sent, says which, and for the KEV-exposure figures and the banner-counted data-store figures we sent nothing *(amended 2026-09-29: this read “read: candidates found in public data, then checked with identified requests of our own”, which is true of some radars only)*. What each radar sends today is published at echelongraph.io/responsible-disclosure. No figure in this report changes; this correction concerns the description of the method only. The sentences of the cover and the disclaimer that it changed carry the same “corrected 2026-09-27” mark.

Correction, 2026-09-29: what each radar sent. The method correction above told readers to read “passive” as “candidates found in public data, then checked with identified requests of our own”. That was true of some radars and not of others, so, read that way, the sentences that call the KEV-exposure and data-store figures passive claimed checks we never made. In this report “passive” means only that the candidates were found in public data. Whether we then sent anything to a host is set out for each radar below; where a chapter says more or less, this table is the authority, and the dated notes add detail to it. Each row was derived from the radar’s code at every change made to it between 29 May and 28 June 2026, not from its last version alone, and a fact the code does not settle is marked “not recorded”. None of these requests logged in, submitted a credential or exploited anything. This table changes no figure.

Radar	Candidates came from	Sent to the host?	What we sent, and how often	What we read, and what we kept
Shadow AI (the whole window)	Certificate Transparency logs, read through crt.sh, and Shodan search results (© Shodan)	Yes	For a host name a DNS lookup, then GET requests: the host’s page, following up to four redirects (for a Shodan address, over http and then https); the target of a JavaScript redirect, if the page had one; up to four known API paths for its kind of service; and in some cases its page again. Every stored row was re-checked in turn, 50 rows every 5 minutes, least recently checked first. Each result of a Shodan sweep was checked when the sweep returned it, every 6 hours until 22 June and every 12 hours after; until 22 June the sweep also ran each time our service restarted, and from 22 June it was skipped when Shodan’s query credits were low. Until 21 June these rates were per instance: our service kept at least one instance running, at least two from 7 June, and could run up to 50, each on its own schedule; before 13 June 23:10 UTC each instance also ran its own Shodan sweep, with no lock, and	Up to 64 KiB of each answer, read to classify it; kept the verdict and a one-line note of what answered (status code, path, matched word), never the page

Radar	Candidates came from	Sent to the host?	What we sent, and how often	What we read, and what we kept
			until 21 June an instance skipped its sweep only while another's was running. From 21 June one instance ran them. Anyone could also re-check a listed row with the public radar's re-check button, at most once per row every 5 minutes. The requests named us in the User-Agent (until 13 June with a link to our home page), from 13 June with a contact address too, and carried no receipt.	
KEV exposure (from 7 June)	Shodan search results (© Shodan): software and versions read from service banners, matched to CVE and CISA-KEV records	No	Nothing was sent to these hosts; the figures rest on Shodan's data alone. Re-checks read Shodan's host records (from 24 June its free InternetDB), not the host. From 22 June, every 6 hours on each instance of our main service, which kept at least two running and could run up to 50, and whenever a company's exposure report, or a disclosure or onboarding draft for it, was opened in our internal admin tools, a DNS lookup of a host name Shodan listed for the address, to attribute it to a company: a name resolution, not a request to the host.	Shodan's records and the matched CVEs; nothing was read from a host
Exposed data stores (from 7 June)	Shodan search results (© Shodan) and, from 27 June, when Shodan's query credits ran out, LeakIX's open-service detections	Only to candidates on web-fronted engines, among them the 480 Kibana and CouchDB stores the report counts. The other 6,127 counted stores were not contacted.	For Kibana, CouchDB, Elasticsearch, ClickHouse, Neo4j and RethinkDB, a GET of the engine's authentication-gated path over http and then https: two requests a check, four for Elasticsearch, which has two paths (on 7 June a first version, for about half an hour, asked other paths; see the data-store correction below). Repeated for each host the search still returned, every 6 hours until 22 June and every 12 hours after, whenever the search ran, and until 20 June also each time our service restarted; on 28 June two revisions of our service were started with a setting that forces a scan at startup. Until 21 June these rates were per instance: our service kept at least two instances running and could run up to 50, each on its own schedule; from 21 June one instance ran them. From about 24 June Shodan's credits ran low and the search, and with it the check, was skipped; from 27 June the search went to LeakIX, whose web-engine results got the same check. Re-checks of stored hosts read Shodan, not the host. From 22 June, the same DNS lookup as the KEV-exposure row. The requests named us in the User-Agent, until 13 June as "Mozilla/5.0	Up to 128 KiB of every answer that was not a 401 or 403, an authentication challenge or a redirect; for an open store, kept only its PII/PCI class and the name fragments that triggered it. For the other 6,127, Shodan's or LeakIX's record

Radar	Candidates came from	Sent to the host?	What we sent, and how often	What we read, and what we kept
			(compatible; EchelonGraph-Verify/1.0)", and from 13 June carried a contact address and a signed receipt.	
Public-Git credentials (from 7 June)	GitHub's public event stream (api.github.com) and GH Archive's hourly event files	No	Nothing was sent to a repository's owner or to any service a credential belongs to: we read the commits from GitHub's public API and GH Archive. From 11 June a match was also checked for its format, checksum or encoding, on the text alone; no credential was ever used or tried.	The commits' changed lines; kept a masked sample and a salted hash of each credential, with its repository, commit and file, never the credential
Exposed configuration (from 21 June)	Host names in Certificate Transparency logs, and IP addresses the Shadow AI radar found in Shodan's search results (© Shodan), each requested on the standard web ports. How many of the 750 hosts were such addresses: not recorded.	Yes	In its first version, early on 21 June, a GET of each of 6 configuration paths over https and then http; from later that day a HEAD request to find the answering scheme, a request for a random path (to detect a host that answers every path) and a GET of each of 14 configuration paths, and from 24 June of 23, one of which, a heap dump, was cut to its first 128 bytes with a Range header. Every 2 hours (on 21 June, at first 6 and then 3 hours), working through the candidate hosts in turn and, from later on 21 June, re-probing every host with a finding each cycle; a restart did not bring a scan forward. On 21 June, until 16:10 UTC, these rates were per instance: our service kept at least two instances running and could run up to 50, each on its own schedule; after that one instance ran them. From 24 June, for a finding on a bare address, a reverse and a forward DNS lookup to name it. Each request named us and carried a signed receipt.	Up to 256 KiB of each file's body, to find a secret; the secret was redacted before storage (a masked sample and a hash), never stored whole
Subdomain takeover (from 21 June)	Host names in Certificate Transparency logs (the chapter's table says "Passive DNS"), and from 22 June the subdomains that Certificate Transparency logs, read through crt.sh, list for a curated	Yes, to subdomains whose DNS record delegated them to a known service	DNS lookups (the CNAME; from 22 June, for some services, a wildcard or a verification record), then a GET of the root page over https and, if that failed or returned nothing, over http, following no redirect. Every 3 hours (on 21 June, at first 12 and then 6 hours) for the candidates from the logs; from 22 June, every 12 hours, for up to 300 subdomains of each watchlist domain, whose findings were stored in the table this report counts; and from 22 June, whenever a company's exposure report, or a disclosure or onboarding draft for it, was opened in our internal admin tools, live on up to 80 of that domain's candidates,	Up to 96 KiB of the page; kept the verdict, the CNAME, the service and the matched fingerprint text, never the page

Radar	Candidates came from	Sent to the host?	What we sent, and how often	What we read, and what we kept
	watchlist of company domains		which could downgrade a stored finding. A restart did not bring a scan forward. On 21 June, until 16:10 UTC, these rates were per instance: our service kept at least two instances running and could run up to 50, each on its own schedule; after that one instance ran them. Each request named us and carried a signed receipt. How many of the report's takeover findings came from the watchlist: not recorded.	

One more request is not a radar's: from 22 June, opening a disclosure draft for a company in our internal admin tools fetched that company's published security.txt from its domain, to find its security contact: a GET of /.well-known/security.txt and, if that gave no contact, of /security.txt, over https, following no redirect and reading up to 64 KiB. The requests named us, with a contact address, and carried no receipt. Our responsible-disclosure page describes this fetch.

Correction, 2026-09-29: the MCP servers. This report counted 2 MCP servers among its 2,050 confirmed-open AI services, "reachable on the public internet with no authentication in the way", in the window 29 May – 28 June 2026. That count is withdrawn. The check behind it fetched the server's web page with GET requests and counted the server open when the page contained words such as "tools" and "jsonrpc". A GET cannot perform the MCP handshake, so that check could not tell whether a server demands authorization; we deleted it on 2026-08-06. The protocol-level check we have run since does not count even a completed handshake as open, because a server can accept it from an anonymous client and still demand credentials when a tool is called, and we never call a tool: an MCP server counts as requiring authorization only when it refuses our server/discover request, or the older initialize handshake, with HTTP 401 or 403 and publishes matching OAuth protected-resource metadata (RFC 9728); anything short of that is reported as not assessed, never as open. The 2,050 include those 2; this correction concerns them only. The 1,563 "MCP Server" discoveries are candidates matched by name or banner; none was sent an MCP request. This window's observations are no longer held. Read from our public API on 2026-09-29 (02:34 UTC), the Shadow AI radar held 1,339 MCP-server observations, largely derived from Shodan data (© Shodan), and counted none of them open; the AI-Exposure Radar never reports an MCP server as open, and its public counts were not split by service type until 30 September 2026, when it began publishing its MCP counts, adjudicated under RFC 9728, on echelongraph.io/shadow-ai-radar. Each sentence that counted MCP servers as open is marked "MCP count withdrawn 2026-09-29".

Correction, 2026-09-29: the breakdown of the 2,050. In four places this report gives its 2,050 confirmed-open AI services as 1,743 open LLM proxies, 171 live vector databases, 2 MCP servers and 81 AI-workflow services, as though those four made up the whole. They sum to 1,997, not 2,050. The remaining 53 are not accounted for in this report, and we do not know why the four fall short of the total. This window's observations are no longer held, so we cannot now say which categories those 53 were in, and we do not assign them to one. The 2,050 itself is the report's liveness count, and with the other three liveness states it sums to the 82,416 discovered; this correction concerns its breakdown only, and the 2 MCP servers remain withdrawn by the correction above. Each of the four sentences is left as published and marked "partial: these four sum to 1,997 of the 2,050; corrected 2026-09-29".

Correction, 2026-09-30: breakdowns that do not sum. In the places below this report gives the parts of a total as though they made up the whole, and they fall short of it. Two are lists we cannot complete. In the chapter "The AI Attack Surface", the table of the 82,416 discovered services by category gives AI Workflow 35,675, LLM Proxy 19,572, Model Store 18,317, Notebook 4,859, MCP Server 1,563 and Vector DB 589. Those six sum to 80,575, not 82,416: 1,841 are not accounted for. The chapter "Mitigation & Remediation" gives the same six counts again, and they sum to the same 80,575. In the executive summary, the 619 secrets found in 478 public Git repositories are given by provider as 249 generic, 228 Google, 63 AWS, 53 Telegram, 13 Discord, 2 OpenAI and 2 HuggingFace. Those seven sum to 610, not 619: 9 are not accounted for. "Mitigation & Remediation" gives the first five of them, which sum to 606, and "The CISO Playbook" the first three, which sum to 540, then names Telegram, Discord, OpenAI and Hugging Face without counts; the chapter "Secret Sprawl" counts those four as 70. The executive summary's severity counts for the 619, 301 high, 273 medium and 45 critical, do sum to 619. We do not know why either list falls short of its total, and the observations behind them can no longer tell us: this window's Shadow AI observations are no longer held, and the public-Git radar deletes a finding 60 days after it was last seen, so of its findings first seen by 28 June 2026, 3 remain. We cannot now say which categories the 1,841 were in, or which providers the 9 belonged to, and we do not assign them to any. The MCP Server row's 1,563, and the MCP 1,563 in "Mitigation & Remediation", are candidates, as the MCP correction above says. The other places give some of the parts as if they were all of them, and elsewhere the report itself shows they are not. The executive summary's table of open data stores by engine lists six engines, which sum to 6,591 of the 6,607; the seventh, CouchDB with 16, is in the table of the chapter "Exposed Data Stores", whose seven engines sum to 6,607. "The CISO Playbook" gives the 6,607 as evidence in two table cells with some of its engines: Redis 4,243, Memcached 1,756 and Kibana 464, which sum to 6,463, and Redis and Memcached alone, which sum to 5,999. "Mitigation & Remediation" lists the 7,952 subdomains in a table of five fronting services, which sum to

7,706; the chapter “Subdomain Takeover” gives the same five as the top five providers. “Past Mistakes & Lessons” gives the web secret findings by severity as 1,386 critical and 1,304 high, which sum to 2,690; the table of them by severity in “Secret Sprawl” also counts 46 medium, 2,736 in all. “The Path Forward” gives the 82,416 AI services as 36,991 auth observed or not determined and 42,593 resolved in DNS only, which sum to 79,584; the other two liveness states are the 2,050 confirmed open, which the same paragraph names, and the 782 unreachable. No figure in this report changes; this correction concerns the sentences and tables it names. Each is left as published and marked “partial”, with the sum its parts give, the total, and “corrected 2026-09-30”.

Correction, 2026-09-29: the data-store check. This report says we confirmed that a data store was open “via a single read-only probe”, and that for web-fronted engines we sent “one anonymous, read-only GET”. The first overstates what we checked; the second understates what we sent. Of the 6,607 open data stores, 480 — Kibana 464 and CouchDB 16 — were confirmed by a check of our own. That check requested the engine’s one known authentication-gated path — a saved-objects query on Kibana, the list of databases on CouchDB — over http and then https, so one check could send a host two GET requests. The same check was used for Elasticsearch (two paths, its index list and cluster health, so up to four requests), ClickHouse (a query that selects a constant, which the server records in its query log), Neo4j and RethinkDB. It ran again for each host the search still returned: every 6 hours until 22 June and every 12 hours after, whenever the search ran, and until 20 June also each time our service restarted; we redeployed it 71 times between 7 and 20 June. After that a restart did not start a scan, except on 28 June, when two revisions of our service were started with a setting that forces one at startup. Until 21 June these rates were per instance: our service kept at least two instances running and could run up to 50, each on its own schedule, and until 20 June each instance scanned when it started; from 21 June one instance ran them. The search spends Shodan query credits: from about 24 June they ran low and the search, and with it the check, was skipped, and from 27 June, with the credits exhausted, the search went to LeakIX, whose web-engine results got the same check. So a host could receive these requests many times over the weeks the radar ran, on the schedule above only while the credits lasted. For about half an hour on 7 June the check’s first version asked Kibana for its status page instead, and CouchDB for two paths, its welcome page and its list of databases, so up to four requests; what that version found was discarded and those stores checked again by the version described here. The requests carried our name, and from 13 June 2026 a signed receipt; they were not anonymous. The check read up to 128 KiB of every answer that was not a 401 or 403, an authentication challenge or a redirect, and when a store answered openly we kept only the PII and PCI classification and the name fragments that triggered it, never the answer itself or any record. The other 6,127 — Redis 4,243, Memcached 1,756, Cassandra 75, MongoDB 37 and InfluxDB 16 — were counted from public internet-scan records, and we did not contact them: Shodan’s banners (© Shodan)

and, from 27 June, LeakIX's open-service detections, a fallback we used when Shodan's query credits ran out. The fallback stored nothing until a fix on 28 June, whose commit records that a scan forced at startup then stored 130 rows through it. The methodology chapter names four of those engines; InfluxDB had no check of ours either. Our own later work found that such banners can be stale or mis-parsed, which is why, since 2 July 2026, after this window, we connect to those engines ourselves before counting them; what that sends is published at echelongraph.io/exposed-databases. The radar behind these counts was built on 7 June 2026, so they cover at most 7 – 28 June, not the whole window. We never logged in, submitted a credential or read a record. The disclaimer's "then confirmed with identified requests of our own" overstates in the same way, as do the marked sentences that say we confirmed or checked all of these counts: those 6,127, like the KEV-exposure counts, which the methodology chapter describes as banner fingerprinting, rest on public scan data alone. Where this report calls the data-store or KEV-exposure figures passive, or passively collected, read: found in public data and, but for those 480, not checked by us; the table "what each radar sent" above says what each radar sent. No figure in this report changes; this correction concerns the description of the method only. Each sentence it changed is marked "data-store check corrected 2026-09-29".

Correction, 2026-09-29: the observation window. This report says its findings were collected over 29 May – 28 June 2026, "roughly four to five weeks", and in places calls a count a measurement of "a single month". That holds for the Shadow AI radar only. The other five radars were built inside the window, and none of their observations can predate the day each was built: the KEV-exposure, exposed-data-store and public-Git credential radars on 7 June 2026, and the exposed-configuration and subdomain-takeover radars on 21 June 2026. Their counts were therefore collected over at most three weeks (7 – 28 June) and at most one week (21 – 28 June) respectively. Where a radar draws on public scan data, that data can be older than our collection of it. The 7,624 new CVEs are counted over the trailing 30 days of the CVE feed and are not affected. Three sentences of the vulnerability-landscape chapter give the 7,624 as a five-week count; it is the feed's trailing 30 days. No figure in this report changes; this correction concerns the stated length of observation only. Each sentence that presents the whole window, or its length, as the span of those five radars' observations, and each of those three, is marked "window corrected 2026-09-29".

Correction, 2026-09-29: the takeover check. This report says a subdomain was counted takeover-able when its "single anonymous read-only GET" returned a service's "unclaimed" fingerprint. That understates what we sent. For each subdomain whose DNS record delegated it to a known service, the check requested the subdomain's root page over https and, if that failed or returned nothing, over http, so one check could send two GET requests; it followed no redirects and read up to 96 KiB of the page. The check was

repeated on every scan cycle, as often as every three hours, so a subdomain could receive these requests many times during the week the radar ran. From 22 June the same check was also run every 12 hours on the subdomains that Certificate Transparency logs list for a curated watchlist of company domains, up to 300 for each domain, and what it found was stored in the table this report counts; how many of the report's takeover findings came from the watchlist is not recorded, because this window's observations are no longer held. From 22 June, too, opening a company's exposure report, or a disclosure or onboarding draft for it, in our internal admin tools re-ran the check live on up to 80 of that domain's candidates, and a finding that no longer held could be downgraded in the same table. Each request carried our identity and a signed receipt; they were not anonymous. The chapter's table calls the detection method "Passive DNS": the names came from Certificate Transparency logs, and each CNAME was resolved live by our own DNS query. We never registered or claimed a resource. No figure in this report changes; this correction concerns the description of the method only. Each passage it changed is marked "takeover check corrected 2026-09-29".

Correction, 2026-09-30: the takeover platforms. This report observed 7,952 subdomains delegated to third-party services and confirmed 134 of them vulnerable to takeover. Its platform counts, Shopify 4,869, GitHub Pages 1,563, Heroku 773, SmugMug 331 and Fastly 170, are counts of the 7,952 observed: the chapter "Subdomain Takeover" gives them as subdomains observed, by provider, and they sum to 7,706. In the chapter "Mitigation & Remediation" the same five counts follow "the confirmed cases cluster on platforms where a CNAME outlives the backing resource", as though they counted the 134 confirmed; Shopify's 4,869 alone is more than 36 times the 134. Three more sentences say which platforms the 134 are on: in "Subdomain Takeover", that they are "concentrated on the self-service providers … including Shopify, GitHub Pages, and Heroku"; in "Risk Analysis: Toxic Combinations", that they are "predominantly against Shopify, GitHub Pages, and Heroku targets"; and in "The CISO Playbook", that each is "a dangling CNAME aimed at a deprovisioned Shopify, GitHub Pages, Heroku, SmugMug or Fastly resource". This report gives no count of the 134 by platform, and the check that confirmed them recognised more services than these five. Whether that split can still be derived: the radar keeps one row for each subdomain, holding only its latest verdict. Each re-check that finds the subdomain still delegated to a service overwrites that verdict, and a live re-check in our internal admin tools can downgrade it; no earlier verdict is kept, and a row whose verdict has not been refreshed for 14 days is deleted. A verdict from this window survives only in a row that has since been neither overwritten, downgraded nor deleted. We have not queried the radar's data for this correction, so the split is not re-derived here, and we do not assign the 134 to platforms. No figure in this report changes; this correction concerns those four sentences only. The counts are left as published and marked "these are counts of the 7,952 observed, not of the 134 confirmed; corrected 2026-09-30", and

each of the other three “these platforms are named from the 7,952 observed; the report does not say which platforms the 134 confirmed are on; corrected 2026-09-30”.

Certification. EchelonGraph certifies that every metric in this report is derived directly from its production exposure-intelligence systems as of 28 June 2026. No figure has been estimated, modelled, or extrapolated; each traces to a query result over observed data.

Disclaimer — what this is, and how we found it. This report describes exposure whose candidates EchelonGraph found *in public data* — Certificate Transparency logs, public internet scan data, public source repositories, and public DNS — and then confirmed with identified requests of our own *(corrected 2026-09-27)* *(data-store check corrected 2026-09-29)*; see the method corrections above. We **never authenticate, never log in, and never exploit**. We detect that a service is reachable and (where determinable) unauthenticated; we do **not** read the data a service holds, but the exposed-configuration check does read an exposed file’s body, up to a size cap, to detect a secret *(corrected 2026-09-27)* *(method superseded 2026-09-29)*. Findings are reported in aggregate and host-redacted, under a responsible-disclosure posture. This is an observational field report, **not a penetration test or an audit of any specific organization**. The absence of a finding here is **not** evidence of safety. Classification (e.g. PII/PCI indicators) is deliberately conservative, and attribution to an organization is made only when ownership is DNS-provable. The observation window is approximately five weeks *(window corrected 2026-09-29)*; this is an inaugural baseline, not a multi-year trend line. Want to see your own exposure? Run the free Surface Scanner.

Contents

1. Executive Summary
2. Methodology & How We Found This
3. The AI Attack Surface
4. Known-Exploited Vulnerability Exposure
5. Exposed Data Stores
6. Secret Sprawl: Exposed Keys & Credentials
7. Subdomain Takeover
8. The Vulnerability Landscape
9. Trends & Trajectory
10. Risk Analysis: Toxic Combinations
11. Mitigation & Remediation
12. Gaps & What We Are Not Seeing
13. The CISO Playbook
14. Past Mistakes & Lessons
15. The Path Forward

Executive Summary

Every organization maintains two attack surfaces. The first is the one it knows about — the assets in its CMDB, the services behind its firewall, the cloud accounts on its bill. The second is the one the entire internet can already see: the forgotten database left listening on a public IP, the AI proxy a team spun up over a weekend, the production host running a version of software that attackers are exploiting this week, the credential committed to a public repository at 2 a.m. The gap between those two surfaces is where breaches happen. This report measures the second surface — the externally observable one — directly, from the same vantage point an adversary uses.

The findings below were collected over a four-to-five-week observation window from **2026-05-29 to 2026-06-28** *(window corrected 2026-09-29)*. This is an inaugural baseline, not a multi-year trend; readers should treat the absolute counts as a snapshot of the present state, and the patterns — not the slopes — as the signal. The methodology is deliberately constrained: every observation is **passive, drawn from public data, and detect-only**. We never authenticate, never exploit, and never read the contents of an exposed system *(method superseded 2026-09-29)*. Findings are aggregated and host-redacted, and confirmed exposures are handled through responsible disclosure to the verifiable owner. The point is not to publish a target list. The point is to establish what is already visible — because if a passive observer can see it, so can a motivated one.

The same dynamic recurs in nearly every major breach of the last decade. Equifax (2017) ran a known-vulnerable, internet-reachable web component for months after a patch existed. Capital One (2019) and the DeepSeek open database (2025) were misconfigurations — an asset reachable from the public internet with access control missing or bypassable, not a novel zero-day. MOVEit (2023) and the OpenSSH *regreSSHion* flaw (CVE-2024-6387) were widely deployed software whose exposed instances became a mass-exploitation event the moment a working exploit circulated. None of these required the attacker to be inside first. Each began with something the whole internet could already see. This report quantifies how much of that visible surface still exists today.

The thesis: the surface is large, it is reachable, and most of it is unwatched

Across six independent passive radars, we observed exposure at internet scale spanning **137 countries for AI services alone** and **161 countries for hosts running known-vulnerable software**. The recurring theme is not exotic attack technique — it is the ordinary failure mode of fast-moving engineering: a store left open, a patch not applied, a DNS record not retired, a secret not rotated. These are not edge cases. They are the default state of an unmanaged external surface, and they are observable from outside without privileged access of any kind.

2,050
AI services confirmed active & open
of 82,416 discovered; 2 MCP servers *(withdrawn 2026-09-29)*

21,299
hosts on CISA-KEV exploited vulns
of 36,416 CVE-exposed

6,607
open, unauthenticated databases
120 countries

2,571
exposed .env files
≈2,600 AWS keys

619
secrets in public Git repos
478 repositories

134
hijackable subdomains
dangling CNAME

340,552
CVEs tracked & enriched
1,621 CISA-KEV

137
countries observed
public data, then identified checks *(corrected 2026-09-27)*

Headline metrics across the six EchelonGraph exposure radars, observation window 2026-05-29 to 2026-06-28 *(window corrected 2026-09-29)*. Counts are passive, public-data, detect-only.

By the numbers

EchelonGraph State of Internet Exposure 2026 — headline figures. Window: 2026-05-29 to 2026-06-28 *(window corrected 2026-09-29)*.

Radar	Headline figure	Reach
Shadow AI footprint	82,416 AI services discovered; 2,050 confirmed active and open	137 countries
KEV exposure	36,416 hosts running known-CVE software; 21,299 exposed to CISA-KEV actively-exploited vulnerabilities	161 countries / 6,519 orgs
Exposed data stores	6,607 open, unauthenticated databases	120 countries / 1,723 orgs
Secret sprawl (web)	750 hosts serving secrets; ~2,600 AWS credentials	—
Leaked credentials (public Git)	619 secrets across 478 repositories	—
Subdomain takeover	7,952 dangling records observed; 134 confirmed vulnerable	—

For context on the vulnerability data that underpins the KEV findings, EchelonGraph tracks a corpus of 340,552 CVEs, of which 1,621 appear on the CISA Known Exploited Vulnerabilities catalog, 326 are linked to ransomware, 33,409 are rated CRITICAL, and 7,624 were published in the last 30 days alone. Exposure is not a static problem against a fixed list of flaws; the list itself grows by thousands of entries a month.

The eight findings to act on this quarter

The remainder of this report develops each radar in depth. For an executive audience, the actionable signal reduces to eight findings. Each is stated with its headline number and the question it should prompt on your own surface.

1. Your AI build-out has created a public footprint of 82,416 services — and you almost certainly cannot enumerate it

We discovered **82,416 AI-related services** across 137 countries: AI-workflow platforms, LLM proxies, model stores, notebooks, MCP servers, and vector databases. The honest reading of that number matters, because most of it is *not* dangerous on its own. The discovered footprint breaks down as follows.

Shadow AI liveness breakdown. "Discovered" is the AI footprint visible in public data; only the active/open set was confirmed reachable and unauthenticated. *(corrected 2026-09-27)*

State	Services	What it means
Auth observed or not determined	36,991	Answered, but not confirmed open: a login or access control was observed, or our verifier could not tell *(corrected 2026-09-27)*
Resolved (DNS-only)	42,593	Name resolves but no live service answered — inventory signal, not an open door
Confirmed active & open	2,050	Live and unauthenticated — the set that warrants action
Unreachable	782	No response on probe

The **2,050 confirmed-active, open** services are the ones that matter: **1,743 open LLM proxies, 171 live vector databases, 2 MCP servers, and 81 AI-workflow instances** *(partial: these four sum to 1,997 of the 2,050; corrected 2026-09-29)*, reachable on the public internet with no authentication in the way *(MCP count withdrawn 2026-09-29)*. An open LLM proxy is an uncapped bill and a data-exfiltration channel; an unauthenticated vector database is a verbatim copy of whatever proprietary corpus was embedded into it. This is the DeepSeek class of problem — an AI asset stood up in a hurry and left open — observed at scale. The so-what for a CISO: the 36,991 that answered without being confirmed open *(corrected 2026-09-27)* are evidence that your teams are deploying AI faster than security can inventory it, and the 2,050 are proof that some of those deployments are already open. The discovery is feasible from outside; the question is whether you find your share first. Note carefully that the auth-observed-or-not-determined and DNS-only sets are **not** counted as exposed and must never be characterized as such *(corrected 2026-09-27)*.

> **2,050 AI services were confirmed active and open on the public internet — 1,743 of them LLM proxies — out of an 82,416-service footprint spanning 137 countries.**

2. 21,299 hosts are exposed to vulnerabilities attackers are exploiting right now

We identified **36,416 hosts running software with known CVEs** — 165,717 distinct host-to-CVE pairings across 6,519 organizations and 161 countries. The subset that should drive remediation priority is the **21,299 hosts exposed to vulnerabilities on the CISA KEV catalog** (50 distinct KEV CVEs): not "you have a vulnerability," but "you are running, on a publicly reachable host, the specific flaw that is under active exploitation." A further **3,475 hosts carry ransomware-linked CVEs**, and **36,365 carry high-EPSS** (high exploit-probability) vulnerabilities. The exposed software population is dominated by the internet's load-bearing infrastructure.

Most-exposed software by host count, and the most-exposed individual CVEs by host count.

Software	Hosts	Top exposed CVE	Hosts
OpenSSH	12,889	CVE-2023-48795 (Terrapin / OpenSSH)	9,942
Apache Tomcat	5,219	CVE-2023-38408 (OpenSSH)	9,923
Apache httpd	3,491	CVE-2025-26465 (OpenSSH)	8,525
MongoDB	3,230	CVE-2023-44487 (HTTP/2 Rapid Reset / Tomcat)	5,859
VMware ESXi	2,816	CVE-2024-6387 (regreSSHion)	5,713
Citrix NetScaler	1,709	CVE-2025-24813 (Tomcat)	5,218

The so-what: *regreSSHion* (CVE-2024-6387) appears on 5,713 exposed hosts, and the Citrix NetScaler family — the lineage behind the "Citrix Bleed" exploitation wave — is present on 1,709. These are precisely the products that turn into mass-compromise events when an exploit circulates, which is the pattern Equifax, MOVEit, and the regreSSHion disclosure each followed. Across the population, host severity tallies **24,292 CRITICAL, 26,482 HIGH, and 19,329 MEDIUM**. Patch prioritization keyed to KEV and EPSS — not raw CVSS — is the single highest-leverage action a security team can take against this surface. Full host-redacted detail is available at the KEV exposure radar.

3. 6,607 databases are open to the internet with no authentication at all

We confirmed **6,607 open, unauthenticated databases** across 1,723 organizations and 120 countries *(data-store check corrected 2026-09-29)*. A critical framing applies here, and it is non-negotiable: our classification is **conservative**. We detect that the

store is open via a single read-only probe; we do **not** read its contents *(data-store check corrected 2026-09-29)*. We therefore make no claim about the volume of data inside — there is no "millions of records" assertion in this report, because we never looked. Only **3 stores were conservatively classified as holding PII and 1 as in PCI scope** on the basis of externally visible metadata alone; the true figure is unknowable without doing the thing an attacker would do, which we will not do. The exposure is concentrated in caching and search engines that ship insecure-by-default.

Open unauthenticated data stores by engine (hosts) and by country *(partial: its six engines sum to 6,591 of the 6,607; the seventh, CouchDB with 16, is in the Exposed Data Stores chapter's table; corrected 2026-09-30)*.

Engine	Hosts	Country	Hosts
Redis	4,243	United States	1,254
Memcached	1,756	China	1,092
Kibana	464	Germany	772
Cassandra	75	France	307
MongoDB	37	Singapore	248
InfluxDB	16	India	221

The so-what: an open Redis or Memcached instance is not merely a data-leak risk — it is a foothold, frequently writable, and a recurring initial-access vector in commodity intrusions. The DeepSeek disclosure (2025) was a single open database that made global news; this radar found 6,607 doors of the same kind, standing open today. Detail at the exposed databases radar.

4. ~2,600 AWS credentials are being served by the applications meant to protect them

We found **750 hosts serving secrets directly to the public internet** — applications handing out the very configuration they were never meant to expose. The dominant exposure vector is the environment file: **2,571 served `.env` files**, alongside 80 exposed Git configs, 61 exposed Git directories, 17 credential files, 3 database dumps, and 2 private keys. The credential material recovered from those served files is overwhelmingly cloud keys: **1,319 AWS secret keys and 1,284 AWS access-key IDs — on the order of 2,600 AWS credentials** — plus 2 GitHub tokens and a PEM key.

Severity rows tally 1,386 critical and 1,304 high. As with every radar, we detect that the file is being *served*; we never read or test a key, because using it would be the attack *(method superseded 2026-09-29)*.

The so-what: a leaked AWS key is not a future risk, it is a present one — automated harvesters scrape `.env` endpoints continuously, and a live key can be abused within minutes of exposure. Capital One (2019) demonstrated what a single mis-scoped cloud credential can unlock. Every served `.env` on this list is a standing invitation. Detail at the exposed AI keys radar.

5. 619 live secrets are sitting in 478 public Git repositories

Separately from web-served secrets, we identified **619 secrets across 478 public Git repositories**. The provider breakdown — 249 generic, 228 Google, 63 AWS, 53 Telegram, 13 Discord, 2 OpenAI, 2 HuggingFace *(partial: these seven sum to 610 of the 619; corrected 2026-09-30)* — shows that credential leakage now spans far beyond cloud IAM into bot tokens, messaging platforms, and, increasingly, AI-service keys. Severity tallies 301 high, 273 medium, and 45 critical. The so-what: source control is a primary exfiltration surface, and a committed secret is exposed from the moment of `git push` — rotation, not deletion, is the only effective response, because the history persists. Detail at the leaked credentials radar.

6. 134 confirmed subdomain takeovers let an attacker speak as your brand

We observed **7,952 dangling DNS records** and **confirmed 134 as vulnerable to subdomain takeover** — a forgotten CNAME pointing at a deprovisioned SaaS endpoint that an attacker can re-claim and serve content from. The exposure clusters on the platforms teams adopt and abandon fastest: Shopify (4,869 observed dangling records), GitHub Pages (1,563), Heroku (773), SmugMug (331), and Fastly (170). The so-what: a hijacked subdomain inherits your domain's trust — it is a turnkey phishing, cookie-theft, and brand-impersonation platform under `yourcompany.com`. The 134 confirmed cases are the ones where the takeover is demonstrably claimable; the gap between 7,952 observed and 134 confirmed reflects deliberately conservative validation, not a lack of risk in the remainder. Detail at the subdomain takeover radar.

7. The vulnerability backlog grows faster than any patch cadence — by 7,624 new CVEs in 30 days

The KEV findings sit on top of a structural problem: the supply of vulnerabilities is accelerating. The EchelonGraph corpus holds **340,552 CVEs**, including **33,409 CRITICAL and 103,468 HIGH**, with **7,624 new entries in the last 30 days** and **5,190 AI-related CVEs**. Of the total, 1,621 are CISA-KEV and 326 are ransomware-linked. The so-what: a remediation program scoped to "patch the criticals" is mathematically losing, because criticals arrive in the tens of thousands. The defensible posture is exposure-led prioritization — fix what is reachable, exploited, and yours, in that order — which is exactly what radars 2 through 6 enable. The full enrichment (CVSS v3/v4, EPSS, KEV tiering, SSVC, ransomware and AI flags) is available at EchelonGraph Pulse.

8. Almost none of this surface is being watched on the owner's behalf

The unifying finding is the one that does not reduce to a single radar. Every exposure above was discoverable passively, from public data, by an outside observer with no special access. In each historical breach we cited, a researcher or an attacker found the open door before the owner did — the only variable was who got there first, and what they did next. The so-what for the board: the external attack surface is already enumerated by adversaries continuously; the asymmetry is that most organizations are not enumerating their own. Closing that asymmetry — knowing what the internet can see about you, before it is used against you — is the single strategic posture this report argues for. Organizations can begin with a self-directed check of their own surface at the surface scanner, map blast radius at the attack graph, and tie exposure to obligations at the compliance view.

How to read the rest of this report

The chapters that follow take each radar in turn: methodology, the full data, what it means, who it hits hardest, and the parallels to incidents the reader will recognize. Two cautions carry throughout. First, this is a **baseline** — a four-to-five-week first measurement *(window corrected 2026-09-29)*, not a trend line; we report what is, and resist over-reading direction from a single window. Second, every figure in this report is an **observed, passively collected, detect-only** count, handled with host redaction and responsible disclosure. We measure the open door. We do not walk through it.

Methodology & How We Found This

> *2026-09-29: every sentence of this chapter that describes how we found, checked, contacted, read or identified anything is superseded by the table what each radar sent; where they differ, the table is right.*

Every figure in this report was produced the same way: by looking at data that is already public, from the outside, without ever logging in, sending an exploit, or touching a single record. That constraint is not a limitation we apologize for — it is the entire point. An attacker probing your perimeter sees exactly what we see. The difference is that we tell you, and we tell only you, before someone less friendly arrives at the same open door. This chapter documents precisely how each of the seven radars works, what our liveness and confidence labels actually mean, how we decide whether a finding belongs to a named company, and — just as importantly — what these numbers are not. Read this chapter before you read any number in the rest of the report. The numbers are only as trustworthy as the method that produced them, and we would rather you understand the method than over-trust the number.

The stance: passive, public-data, detect-only

The most important fact about this report is the smallest: we never authenticated to anything, we never wrote anything, we never validated a found credential by using it, and we never claimed a dangling resource. Each of those actions would, under the Computer Fraud and Abuse Act and its international analogues, constitute unauthorized access — and each would also make our findings legally and ethically unusable. So we built the constraint into the architecture rather than the policy. Our radars consume four classes of strictly public data:

- **Certificate Transparency logs** — the append-only public ledgers every certificate authority is required to write to. When an organization provisions TLS for `llm.internal-tool.example.com`, that hostname becomes a permanent public record, whether or not the service behind it was ever meant to face the internet.

- **Internet-wide scan banners (Shodan)** — the service fingerprints, software names, and version strings that hosts volunteer to anyone who connects to an open port.
- **Public DNS** — forward, reverse, and CNAME records that anyone can resolve.
- **Public source-code commits (GitHub / GH-Archive)** — the diffs of every commit pushed to a public repository.

Where a finding required a confirmation step beyond passive data — for example, distinguishing a genuinely open database from one that merely *looks* open behind a client-side login — we permitted ourselves exactly one action: a single anonymous, read-only HTTP GET, the same request a browser makes when it loads a page *(data-store check corrected 2026-09-29)*. Nothing more. We confirm that the door is open *(data-store check corrected 2026-09-29)*. We do not walk through it. Every outbound probe also identifies itself (see Identified scanning below), so any administrator who sees us in their logs can confirm it was us and reach us to ask us to stop *(method superseded 2026-09-29)*.

> **Detect-only, end to end. Zero authentications. Zero exploit attempts. Zero records read *(method superseded 2026-09-29)*. Where confirmation was needed, the strongest action we took was one anonymous read-only GET *(corrected 2026-09-27)*.**

The observation window: a baseline, not a trend

This is our inaugural report, and honesty about its time horizon matters more than the impressiveness of any single figure. All findings were collected between **29 May 2026 and 28 June 2026** — roughly four to five weeks *(window corrected 2026-09-29)*. That makes this a *baseline*, not a multi-year trend line. We deliberately do not draw growth curves, year-over-year deltas, or directional claims from this window; we have no prior year to compare against. Where a per-month split appears — for example, the AI attack-surface count rising from 10,679 discovered footprints in May to 71,737 in June — that jump reflects our scanner fleet ramping its coverage of the public internet, not a real-world surge in exposure over those weeks. We flag it as such wherever it appears, and you should read it that way. Future editions, with multiple baselines behind them, will support trend analysis. This one establishes the starting line.

Radar 1 — AI Attack Surface (Certificate Transparency + Shodan)

So what: the AI build-out is creating internet-facing services faster than security teams can inventory them, and Certificate Transparency makes every one of them publicly discoverable the moment it gets a certificate. This radar tails CT logs and Shodan for the signatures of AI infrastructure — LLM proxies, vector databases, Jupyter notebooks, model stores, MCP servers, and AI-workflow tools (Flowise, Langflow, and the rest of the zoo) — using a large, maintained library of hostname patterns and Shodan service dorks. Across the window we discovered **82,416** AI-related footprints spanning **137 countries**. See the live Shadow AI Radar.

The discipline that makes that 82,416 number defensible is what we do *after* discovery. A certificate or a banner only proves a hostname exists — not that anything is listening, and certainly not that it is open. So every discovered footprint is run through a liveness verifier that performs DNS resolution plus a lightweight, category-aware HTTPS probe (for a notebook, `/api/kernels`; for an Ollama-class proxy, `/api/tags`; for a Flowise-class workflow tool, `/api/config`). The verifier is conservative by construction: any sign of an authentication gate — a 401 or 403, a redirect to a login page, login-form markers in the body — or any inconclusive result is treated as *authenticated*, never as open. *(Corrected 2026-09-27: which is why this report now labels that bucket “auth observed or not determined”; see the correction at the front.)* We only ever label a host active when we positively confirmed there was no gate in front of it. This collapses the single largest class of false positive in surface scanning and is the reason the headline number and the "active" number differ by more than an order of magnitude.

What "discovered," "auth observed or not determined," "resolved," and "confirmed active" actually mean *(corrected 2026-09-27)*

So what: this is the most consequential paragraph in the report, because the difference between these labels is the difference between a real exposure and a non-event — and conflating them is how vendors manufacture scary headlines. We do not conflate them. Of the 82,416 AI footprints we discovered, here is the exact breakdown:

AI footprint liveness states (n = 82,416)

State	Hosts	What it means
Auth observed or not determined	36,991	The service answered, and either a login gate or other access control was observed, or our verifier could not tell whether one was there *(corrected 2026-09-27)*. **We never call these "exposed."**
Resolved (DNS-only)	42,593	The hostname resolves, but the service returned a 4xx/5xx or was unreachable over HTTPS. A record exists; no open service is behind it.
Confirmed active / open	2,050	HTTPS responded 200–299 with *no* login gate. A real, anonymously reachable AI service. This — and only this — is genuine exposure.
Unreachable	782	The hostname does not resolve at all — dead or never hosted.

The 2,050 confirmed-active set decomposes into **1,743 open LLM proxies**, **171 live vector databases**, **2 MCP servers** *(MCP count withdrawn 2026-09-29)*, and **81 AI-workflow tools** *(partial: these four sum to 1,997 of the 2,050; corrected 2026-09-29)*. When this report says "exposed AI services," it means those 2,050 — not the 82,416 we discovered, and emphatically not the 36,991 that answered without being confirmed open *(corrected 2026-09-27)*. We state this plainly because the temptation in this industry is to report the discovery count as the exposure count. We consider that dishonest, and we do not do it.

> **82,416 discovered ≠ 82,416 exposed. Only 2,050 were confirmed anonymously reachable with no authentication. The other 80,366 were not confirmed open: auth observed or not determined, a DNS record with nothing reachable behind it, or a dead hostname *(corrected 2026-09-27)*.**

From 82,416 discovered AI footprints to 2,050 confirmed-active exposures: how the liveness verifier filters discovery down to genuine, anonymously reachable services.

Radar 2 — KEV Exposure (banner fingerprinting × CVE correlation)

So what: "you have a vulnerability" is noise; "you are running, on the public internet, the exact software version that attackers are exploiting this week" is a fire alarm. This radar produces the second statement, not the first. We take Shodan's view of what software a host is running — product and version, parsed from the service banner — and correlate it against our CVE corpus by product and version match. We then keep a finding only if it is either CISA-KEV-listed (confirmed exploited in the wild) or carries a high EPSS exploit-probability score. A plain CVE with no exploitation signal is discarded; the radar's entire purpose is to surface what is reachable *and* under active attack. See KEV Exposure.

Across the window this surfaced **36,416 hosts** *(window corrected 2026-09-29)* running software with known CVEs — **165,717** distinct host-to-CVE pairs across **6,519 organizations** and **161 countries**. Of those hosts, **21,299** are exposed to a vulnerability on CISA's *actively-exploited* KEV catalog, mapping to **50 distinct KEV CVEs**; **3,475** run software tied to ransomware campaigns. The honest framing is the one we used in the previous sentence: 36,416 hosts run software with *known* CVEs, and 21,299 of them are exposed to *actively-exploited* ones. We keep those two populations distinct because they carry very different urgency. The top exposed software by host count — OpenSSH (12,889), Apache Tomcat (5,219), Apache httpd (3,491), MongoDB (3,230), VMware ESXi (2,816) — and the top CVEs, led by the Terrapin SSH attack (CVE-2023-48795, 9,942 hosts) and regreSSHion (CVE-2024-6387, 5,713 hosts), are analyzed in the KEV chapter. One caveat travels with all banner-based correlation: a version string can be stale or back-patched, so a banner match is strong evidence of exposure but not proof a host is unpatched. We surface the correlation; we do not assert exploitation has occurred.

Radar 3 — Exposed Data Stores (banner + a single read-only verifier probe) *(data-store check corrected 2026-09-29)*

So what: an unauthenticated database on the public internet is the DeepSeek-class failure — a production datastore left open on the internet with no password, readable by anyone who found it. We hunt that class at scale. The challenge is that a banner alone cannot always tell an open store from a secured one: modern web-fronted engines (Kibana, Elasticsearch) render their login screens client-side, so a password-walled instance emits the same banner as an open one. So for HTTP-fronted engines we send one anonymous, read-only GET to an *authentication-gated* endpoint — one that returns 401 or a login redirect when the instance is secured — and report the host only if it

answers with open, data-serving content *(data-store check corrected 2026-09-29)*. We deliberately avoid endpoints that stay public even on secured instances (Kibana's `/api/status`, CouchDB's welcome page) precisely because they would generate false positives. Raw-protocol engines (Redis, MongoDB, Memcached, Cassandra) are not probed at all; their banner already encodes auth state (Redis announces `NOAUTH`), and we never speak their wire protocol *(data-store check corrected 2026-09-29)*. See Exposed Databases.

This surfaced **6,607 open, unauthenticated databases** across **1,723 organizations** and **120 countries**, led by Redis (4,243 hosts) and Memcached (1,756). The single most important sentence in this entire report follows, and it governs how every data-store number must be read:

> **We detect that the store is open. We never read its contents *(method superseded 2026-09-29)*. Our classification is conservative — of 6,607 open stores, we flagged only 3 as likely holding PII and 1 as likely PCI, from metadata alone. We make no claim about how many records any store holds.**

You will not find the phrase "millions of records exposed" anywhere in this report, because we did not read a single record and we cannot honestly count what we did not read *(method superseded 2026-09-29)*. The PII and PCI classifications are conservative inferences from structural metadata — field and index names visible without authentication — not from data we extracted. When in doubt, we did not classify. This is the opposite of the prevailing breach-report style, and it is deliberate.

Radar 4 — Secret Sprawl on the web (path probing + redacting regex)

So what: in the rush to ship, applications routinely serve their own configuration straight to the internet — a `.env` file, a `.git` directory, a credential file sitting at a public URL with no authentication — and those files frequently contain cloud keys in plaintext. This radar checks public hosts for those well-known exposure paths and, when one is being served, runs the body through a redacting secret detector. We confirmed that the file is being served; we never read, tested, or used a single key *(method superseded 2026-09-29)*. Doing so would *be* the attack. Across the window this surfaced **750 hosts** *(window corrected 2026-09-29)*, dominated by exposed

environment files (2,571 instances), and within those bodies roughly **2,600 AWS credentials** (1,319 secret keys and 1,284 access-key IDs). See Exposed AI Keys.

Radar 5 — Leaked Credentials in public Git (commit diffs + a six-stage validator)

So what: credentials committed to public repositories are a top root-cause of breaches, but the naive approach — regex-matching commit text — produces so many false positives (documentation examples, placeholder values) that the output is unusable. We engineered for precision over recall: we would rather miss a real credential than show you a fake one. A regex match is only the first of six gates a candidate must clear before it is ever stored or shown:

1. **Diff-awareness** — only newly added (+) lines can introduce a secret; removed, context, and hunk-header lines are ignored.
2. **Known-example rejection** — famous documentation and test credentials are dropped.
3. **Placeholder rejection** — templated values, env-references, and `your-key-here` strings are dropped.
4. **Structural verification** — provider checksums and decoders that *prove* a value is well-formed (GitHub token CRC32, JWT base64+JSON decode, AWS base32). "Verified" here means structurally verified by shape — **never** confirmed live, because confirming a key live would require using it.
5. **Example-context rejection** — for unverified values, a test/docs/example file path drops the hit (the classic "token pasted as a code sample" false positive).
6. **Entropy and shape** — unverified token values must clear a Shannon-entropy floor and character-diversity check, with no long repeats or monotonic sequences.

Only candidates that survive all six are surfaced — and even then the raw secret is never stored; we persist a redacted form and a fingerprint. This yielded **619 secrets across 478 repositories**, spanning generic (249), Google (228), AWS (63), and other providers. Because "verified" means structurally verified by checksum and never confirmed by use, no number in this radar should be read as "619 live, working keys" — it is "619 strings that are shaped like real credentials and survived a six-stage anti-false-positive gauntlet." See Leaked Credentials.

Radar 6 — Subdomain Takeover (dangling-CNAME fingerprinting)

So what: when a subdomain's CNAME still points at a third-party service whose underlying resource has been deprovisioned — a cancelled SaaS, a deleted bucket — an attacker can register that resource and serve their own content from *your* subdomain, enabling phishing, cookie theft, and OAuth abuse. We detect these the way the established tooling (subjack, nuclei) does: a subdomain whose CNAME delegates to a known service *and* whose single anonymous read-only GET returns that service's "unclaimed" fingerprint is takeover-able *(takeover check corrected 2026-09-29)*. We never register or claim the dangling resource — that would be exploitation; we store only the matched fingerprint as evidence.

The gap between observation and confirmation here is enormous and instructive: we observed **7,952 subdomains** delegating to takeover-prone services, but confirmed only **134 as actually vulnerable**. The other 7,818 mostly resolve to live, properly-claimed services — a CNAME to a known provider is *not* a finding unless the resource behind it is genuinely unclaimed. We report the 134, not the 7,952, as the exposure. Two guardrails suppress the most common false positives: a domain the owner has provably verified with the provider cannot be claimed by anyone else (downgraded), and a wildcard delegation where unrelated subdomains share the same target is an intentional configuration, not a forgotten record (downgraded). See Subdomain Takeover.

Observed vs. confirmed across the radars: why we report the smaller number

Radar	Discovered / observed	Confirmed exposure	What the gap is
AI Attack Surface	82,416	2,050 active	Auth observed or not determined, DNS-only, or dead hostnames *(corrected 2026-09-27)*
KEV Exposure	36,416 CVE hosts	21,299 KEV-exposed	Known-CVE vs. actively-exploited
Subdomain Takeover	7,952 observed	134 vulnerable	Live/claimed delegations are not findings
Exposed Data Stores	6,607 open	3 PII / 1 PCI (conservative)	We detect "open," never read contents *(method superseded 2026-09-29)*
Leaked Credentials	match candidates	619 (post 6-gate)	Structurally verified, never confirmed live

The CVE corpus behind the correlation

So what: the KEV-exposure radar is only as good as the vulnerability intelligence it correlates against, so the corpus deserves its own accounting. It comprises **340,552 CVEs**, of which **1,621** are CISA-KEV-listed, **326** are ransomware-associated KEV entries, **33,409** are rated CRITICAL, and **5,190** are AI-related. Each CVE is enriched per-record with CVSS v3 and v4, FIRST EPSS exploit-probability, CISA-KEV status with our own tiering, SSVC decision points, GitHub Security Advisory data, ransomware and AI-relatedness flags, and a synthesized EchelonGraph score. The full scoring methodology — including where we diverge from NVD and where we deliberately do not claim superiority — is documented separately; here it is enough to know that the KEV correlation rides on this enriched, continuously-refreshed corpus rather than a static feed. See CVE Pulse.

Attribution: forward-confirmed reverse DNS, and rare by design

So what: the most dangerous thing a report like this can do is attribute an exposure to the wrong company — both because it is wrong, and because telling Company A about Company B's open database leaks B's exposure to A. So we attribute conservatively,

on provable ownership only, and we leave the unattributable in an aggregate bucket rather than guess. There are two cases:

- **Host-based findings** (an exposed config file, a takeover, a spoofable domain): we attribute only when the host *is* the company's registrable domain or a subdomain of it — because only the domain owner can create records under `*.their-domain`.
- **IP-based findings** (KEV hits, exposed databases): we require **forward-confirmed reverse DNS (FCrDNS)**. The IP's reverse-DNS hostname must resolve *forward* back to that same IP *and* sit under the company's domain. Reverse DNS alone is forgeable; the matching forward record is not. A lookup that times out is treated as "not confirmed" — we exclude rather than guess.

Anything that fails this bar — an organization-name match alone, a PaaS-hosted application whose individual tenant we cannot identify, an IP that does not forward-confirm — is never attributed to a named company and never contacted. The cost of a false attribution is far higher than the cost of missing a true one. As a direct consequence, confident company-level attribution is *rare*: the figures in this report are aggregate and host-redacted, and the small subset where we can prove ownership is what feeds responsible disclosure, not publication.

Identified scanning and responsible disclosure

So what: anonymous internet-wide scanning is indistinguishable from reconnaissance for an attack, which is why we made ours the opposite of anonymous. Every outbound probe carries a self-describing `User-Agent` naming EchelonGraph and linking to our responsible-disclosure page, plus an RFC 7231 `From` header with a monitored contact address *(method superseded 2026-09-29: before 13 June 2026 no request carried a `From` header or linked to our responsible-disclosure page; the Shadow AI radar's User-Agent linked to our home page, and the data-store check's read "Mozilla/5.0 (compatible; EchelonGraph-Verify/1.0)")*. Direct host probes additionally carry a one-off signed receipt — an HMAC-SHA256 token over the IP, port, radar, and timestamp — that any administrator can paste into a public verifier to confirm a request genuinely came from us (and, by the same token, identify impostors who merely put our name in their User-Agent) *(method superseded 2026-09-29: the Shadow AI radar's requests, behind the 82,416 and 2,050 figures, carried no receipt at any time in this window, nor did the data-store check's before 13 June or the security.txt fetch the table describes, so a request of ours without a receipt is no sign of an impostor)*. Where we can prove ownership, the platform renders a disclosure *draft* — the verified exposures and their

specific fixes — that a human sends from their own mailbox. We never auto-send, never include a raw secret in the body, and never lead with a sales pitch. The unattributable majority is reported only in aggregate, as it appears throughout this document.

What this report is not

We close the methodology with its boundaries, stated as plainly as we can, because a finding misunderstood is worse than a finding never made:

- **This is not a penetration test.** It is a record of what is observable from public data, from the outside, without authentication or exploitation. We confirmed open doors; we did not enter any building.
- **Absence of a finding is not proof of safety.** If your organization does not appear in our data, it means we did not observe a qualifying exposure in this window through these sources — not that you have none. We see what Certificate Transparency, Shodan, public DNS, and public Git reveal, on the cadence we scan them, nothing more.
- **This is a four-to-five-week baseline *(window corrected 2026-09-29)*, not a trend.** No directional or year-over-year claim is supported; month-over-month movement reflects scanner ramp, not real-world change.
- **"Discovered" is not "exposed," "known-CVE" is not "actively-exploited," and "structurally verified" is not "confirmed live."** Each distinction is preserved in every figure, and the smaller, more defensible number is always the one we headline.
- **We never read data, so we never count records** *(method superseded 2026-09-29)*. Any claim about the contents or record-count of an exposed store would be fabrication; we make none.
- **Attribution is provable or absent.** Aggregate figures are exactly that; company-level claims exist only where ownership is DNS-provable.

The historical incidents this report references — the Equifax breach, the Capital One exposure, the DeepSeek open database, the MOVEit and regreSSHion campaigns — share a single property: in each, the failing surface was reachable and observable before it was abused. Someone could have looked on the victim's behalf and did not. That is the gap this methodology exists to close: a passive, identified, responsibly-disclosed view of the open doors anyone on the internet can already see, delivered to the people who can close them. Test your own surface at our self-check.

The AI Attack Surface

The most important number in this chapter is the one we did *not* report. Across the observation window we discovered 82,416 internet-facing AI services in 137 countries. We could have published that figure as "82,416 exposed AI systems" and it would have travelled. It would also have been wrong. The honest finding is smaller, harder-won, and far more useful to a security leader: of those 82,416, the overwhelming majority are either services we did not confirm open (auth observed or not determined) or nothing more than a DNS record *(corrected 2026-09-27)*. The genuinely dangerous set — services we confirmed to be live, reachable, and answering without credentials — numbers 2,050. That distinction is the entire point of this chapter, and we ask you to carry it into every conversation that follows.

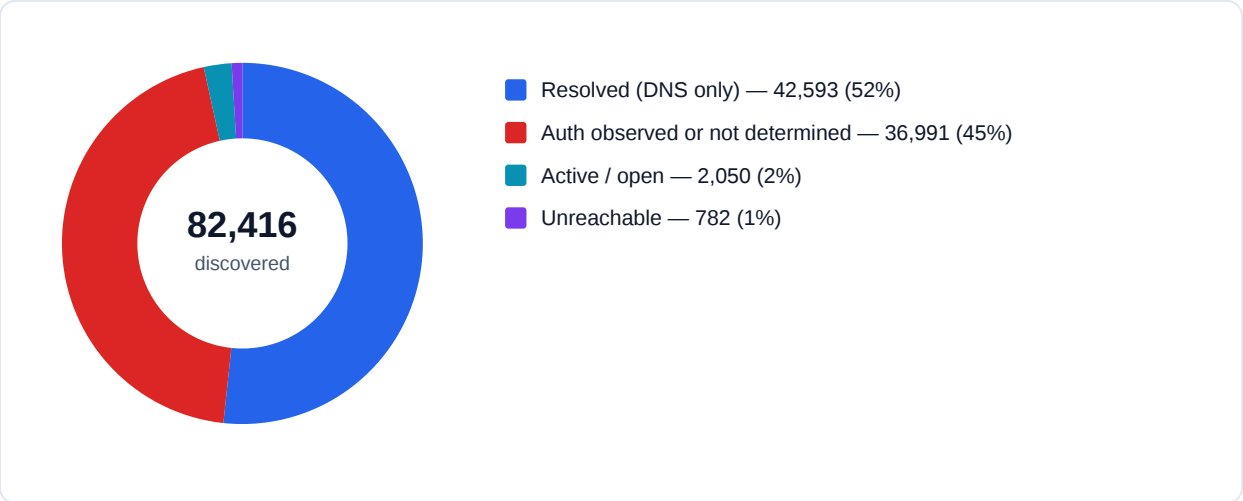
> **82,416 AI services discovered. 2,050 confirmed active and open. The gap between those two numbers is where most "AI exposure" reporting goes wrong.**

What we counted, and what we refused to count

"Shadow AI" has become a boardroom phrase without a boardroom-grade definition. We use it narrowly: AI infrastructure that an organization is running on the public internet, often outside the visibility of its own security team — model-serving proxies, vector databases, notebooks, model registries, agent frameworks, and the emerging class of Model Context Protocol (MCP) servers that broker tool access to large language models. We find these passively, from Certificate Transparency logs and public internet scan data. We never authenticate, never log in, and never send a prompt. We observe that a service exists, we classify what it is, and — where it is determinable without interaction — we record whether it answered us without asking for a credential *(method superseded 2026-09-29)*.

That last step is where discipline matters. A hostname appearing in a Certificate Transparency log proves only that someone requested a TLS certificate for it. It does not prove the service is running, reachable, or unprotected. Treating certificate

evidence as exposure evidence is the single most common error in this category, and it inflates headline numbers by an order of magnitude. We therefore resolve every discovered footprint into one of four liveness states, and we report all four.



The AI footprint by liveness state. Only the 2,050 "active / open" services were confirmed to answer us without authentication; 42,593 are DNS records with no reachable service, and 36,991 answered but were not confirmed open — auth observed or not determined *(corrected 2026-09-27)*. n = 82,416.

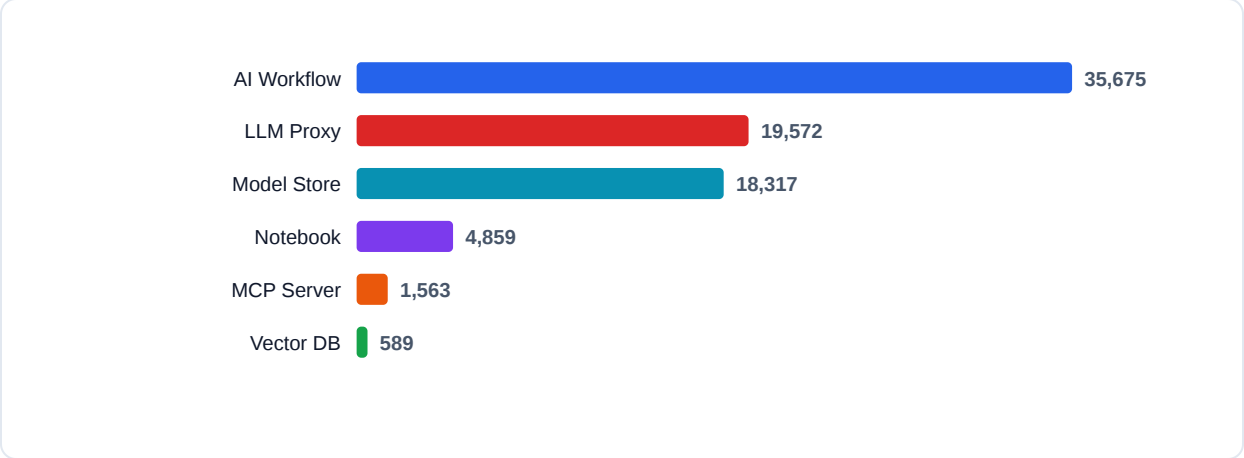
Liveness state	Services	What it means for risk
Resolved (DNS only)	42,593	A hostname resolves, but no AI service answered. Almost always noise — parked names, retired endpoints, internal-only records. **Not an exposure.**
Auth observed or not determined	36,991	A live AI service that either demanded a credential or answered in a way our verifier could not classify *(corrected 2026-09-27)*. **We do not call these "exposed."**
Active / open	2,050	A live AI service that responded without authentication. **This is the real attack surface.**
Unreachable	782	Discovered, but no successful connection during the window — firewalled, rate-limited, or transiently down.

We dwell on the 36,991 in the auth-observed-or-not-determined bucket deliberately, because they are the part of the story that resists a clean headline *(corrected 2026-09-27)*. Some are organizations running AI infrastructure on the public internet behind a login; for others our verifier could not tell, and we do not know how the 36,991 split between the two. That an LLM proxy is internet-reachable is not, by itself, a finding;

reachable-and-authenticated is the normal, defensible posture for a great deal of production AI. Counting the 36,991 as exposed would be the same mistake in the opposite direction — manufacturing alarm out of services we did not confirm open *(corrected 2026-09-27)*. The number that should occupy a CISO's attention is 2,050.

The shape of the footprint: where AI is sprawling

Before narrowing to the active set, it is worth understanding the composition of everything we discovered, because it maps the directions in which AI infrastructure is proliferating fastest. The 82,416 services break down by category as follows *(partial: this table sums to 80,575 of the 82,416; corrected 2026-09-30)*.



Discovered AI infrastructure by type (all liveness states). AI-workflow and orchestration platforms dominate the footprint, followed by LLM proxies and model stores.

Category	Discovered	What it is
AI Workflow	35,675	Agent frameworks and orchestration platforms (Langflow, AnythingLLM, ComfyUI and similar) that chain models, tools, and data.
LLM Proxy	19,572	Gateways that front one or more language models — the inference edge of an AI application.
Model Store	18,317	Registries and repositories that host model artifacts and weights.
Notebook	4,859	Interactive data-science environments (Jupyter and kin) — frequently rich in code, credentials, and data.
MCP Server	1,563	Model Context Protocol endpoints that grant models access to tools, files, and external systems. *(candidates, not confirmed MCP servers; see the 2026-09-29 correction)*
Vector DB	589	Embedding stores — the memory layer of retrieval-augmented applications, holding vectorized copies of private corpora.

The dominance of AI-workflow and orchestration platforms is the structural signal here. The fastest-growing slice of the AI attack surface is not the model itself but the connective tissue around it — the agent frameworks, proxies, and protocol servers that wire models to tools and data. These are typically deployed by application teams rather than platform or security teams, they ship with permissive defaults, and they accumulate access. That is precisely the profile of infrastructure that drifts out of a security organization's line of sight.

The real risk: 1,743 open LLM proxies and 171 live vector databases

Within the 2,050 active and open services, two categories carry disproportionate consequence. We confirmed 1,743 open LLM proxies and 171 live vector databases answering on the public internet with no authentication, alongside a small number of open MCP *(MCP count withdrawn 2026-09-29)* and AI-workflow endpoints. These are not parked names or polite 401 responses. They are running services that returned data to an unauthenticated request.

> **1,743 open LLM proxies and 171 live, unauthenticated vector databases — answering anyone on the internet, no credential required.**

An **open LLM proxy** is, in practical terms, someone else's inference budget and someone else's model — handed to the entire internet. The immediate harm is the obvious one: unmetered consumption of a paid model, billed to the operator. The more serious harms are second-order. A proxy is a position of trust between an application and a model; depending on configuration it may expose system prompts, leak the contents of prior conversations held in context, accept prompt-injection that redirects the model's behaviour, or — where the proxy fronts tool-calling — become a pivot into whatever the model is permitted to do. An unauthenticated proxy is not merely a billing leak; it is an open door into an application's reasoning layer.

A **live, unauthenticated vector database** is the more acute exposure of the two, and it is the one we want security leaders to internalize. A vector store is the memory of a retrieval-augmented application. Organizations populate it by embedding their private corpora — support tickets, internal wikis, contracts, source code, customer records — into vectors so a model can retrieve them. Crucially, those stores frequently retain the original text alongside the vector, as metadata, so the application can show a human the source passage. An open vector database, therefore, is not an abstract index of numbers. It is, very often, a queryable copy of the exact private documents an organization fed into its AI — sitting on the internet without a password.

Why AI infrastructure is uniquely dangerous

A traditional exposed database is a serious problem; an exposed AI service can be a worse one, for a reason that is specific to how these systems are used. AI infrastructure concentrates raw, unstructured, high-sensitivity data at the exact moment it is least protected.

- **The payloads are raw and unredacted.** Conventional applications pass structured, minimized fields between tiers. AI systems pass whole documents, full conversations, and complete records — because the model needs context to be useful. The data crossing an LLM proxy or sitting in a vector store has typically *not* been through the redaction and minimization that a normal data tier would apply. Personally identifiable information arrives intact, inside the payload, as a matter of course.

- **Retrieval stores hold copies, not references.** A vector database does not point at the system of record; it ingests a duplicate. Every access control, retention policy, and audit boundary that governs the source system has to be re-implemented on the copy — and when the copy is stood up by an application team in a hurry, it usually is not.
- **The tooling is young and ships open by default.** Much of this software is pre-1.0. Authentication is frequently opt-in rather than mandatory, default credentials are common, and the developer-convenience defaults that make a tool easy to try are the same defaults that make it dangerous to expose. Services that do demand credentials show the secure path exists *(corrected 2026-09-27)*; the 2,050 open ones show how easy it is to skip.
- **Agentic services hold access, not just data.** An MCP server or an agent framework is provisioned with credentials and tool access so the model can *act*. Exposing one does not merely leak information; it can hand an attacker the model's permissions — to read files, call internal APIs, or reach systems the model was trusted to reach.

The combination is what makes the category distinctive. An exposed AI service tends to leak the most sensitive data in the most usable form, from infrastructure the security team did not know was there.

The parallel: DeepSeek's open database, 2025

This is not a hypothetical. In early 2025, security researchers reported that an internet-facing database belonging to the AI company DeepSeek was publicly accessible without authentication, and that it exposed operational data associated with the service. The episode became a reference point for AI-era exposure for a simple reason: a frontier-scale AI provider, under intense growth pressure, left a backing store open to the internet — the same failure mode that has produced data exposures for two decades, now sitting directly behind an AI product.

We invoke the DeepSeek incident as a pattern, not to restate its particulars, because the pattern is exactly what our data describes at scale. The 171 open vector databases and 1,743 open LLM proxies we confirmed are 1,914 instances of the same class of mistake — an AI backing service reachable without a credential — distributed across the internet, most of them at organizations that have no idea the door is open. DeepSeek made headlines because of the name attached to it. The thousand-plus equivalents in this dataset are anonymous only because no researcher has knocked yet.

The deeper continuity is with the pre-AI canon of exposure incidents. The mechanics that produced the 2017 Equifax breach and the 2019 Capital One breach — an unpatched edge, an over-permissioned path to data, a store reachable from where it should not have been — have not changed. What has changed is the *content* behind the open door. Where an exposed store once leaked structured database rows, an exposed vector store leaks the raw documents an organization trusted to its AI, and an open proxy leaks the live reasoning of the application itself. AI did not invent the failure; it raised the value of the loot.

Reading the trajectory honestly

We discovered 10,679 of these services in May and 71,737 in June. That is not a measurement of the internet's AI footprint growing sixfold in a month. The jump is overwhelmingly the result of our own discovery capacity ramping during this inaugural window — more Certificate Transparency patterns, more scan coverage, more categories under watch. We surface the monthly split for transparency, and we caution explicitly against reading it as a growth rate. This is a baseline, established over roughly five weeks. Its value is as a fixed point to measure *against* in future editions, not as a trend in itself.

What the baseline does establish, firmly, is the shape of the problem. AI infrastructure is now a first-class category of internet exposure, distributed across 137 countries, dominated by orchestration and proxy layers that live outside traditional security visibility. Most of it is either inert or was not confirmed open *(corrected 2026-09-27)*. A meaningful, concrete minority — 2,050 services, with 1,743 open proxies and 171 open vector databases at its core — is live, unauthenticated, and holding exactly the kind of data and access that makes AI systems worth attacking.

For the security leader. Your exposure here is almost never a system your security team chose to expose; it is one an application team stood up and forgot to gate. Inventory every AI service your organization runs on the public internet — proxies, vector stores, notebooks, MCP and agent endpoints — and verify that each one demands a credential. Treat an unauthenticated vector database as a data breach in waiting, not an infrastructure tidy-up. The free Surface Scanner will show you what the internet can already see; the Shadow AI Radar tracks the wider footprint this chapter is drawn from.

Known-Exploited Vulnerability Exposure

The single most actionable fact in this report is that adversaries do not need to discover anything. The vulnerabilities that compromise organizations are, overwhelmingly, already catalogued, already weaponized, and already being used in the wild. Across the observation window of 29 May to 28 June 2026 *(window corrected 2026-09-29)*, EchelonGraph passively fingerprinted **36,416 internet-facing hosts running software with at least one known CVE**, spanning 165,717 distinct host-to-CVE pairs, **6,519 organizations** and **161 countries**. These are not theoretical weaknesses. Of those hosts, **21,299 are exposed to at least one vulnerability on the CISA Known Exploited Vulnerabilities (KEV) catalog** — vulnerabilities for which CISA has confirmed active exploitation against real victims. The gap between "a patch exists" and "the patch is applied" is the gap through which most breaches walk.

> **21,299 hosts are exposed to vulnerabilities that attackers are confirmed to be exploiting right now — across 50 distinct CISA-KEV CVEs.**

This finding is the through-line of the modern breach record. Equifax in 2017 was not undone by a novel exploit; it was a known Apache Struts flaw (CVE-2017-5638) with a patch available for months before the intrusion. Capital One in 2019 turned on a misconfiguration layered over a reachable service. The 2023 MOVEit campaign weaponized a single SQL-injection flaw across thousands of organizations simultaneously, because the same vulnerable file-transfer appliance sat on the internet edge of so many enterprises at once. In every case, the precondition was the same one we measure here: a known weakness, exposed, unpatched, at scale. Our window is an inaugural baseline of roughly four to five weeks *(window corrected 2026-09-29)*, not a multi-year trend — but the structure it reveals is consistent with a decade of incident data.

What "known-exploited" actually means — and why it changes the math

Most vulnerability programs drown in volume. Our broader corpus tracks 340,552 CVEs, of which 103,468 are rated HIGH and 33,409 CRITICAL by CVSS. No security team can remediate at that scale, and treating every CRITICAL as equally urgent is how real emergencies get buried under paperwork. The KEV catalog exists precisely to cut through this: it is the subset where exploitation is not predicted but observed. A vulnerability on the KEV list has crossed the line from "an attacker could" to "an attacker is."

That distinction reorders priorities. CVSS asks how bad a flaw would be if exploited. EPSS (the Exploit Prediction Scoring System) estimates the probability it will be exploited in the near term. KEV reports that it already has been. We enrich every host-to-CVE pairing with all three signals, plus ransomware-campaign association and our own EchelonGraph tiering, so that an operator can sort 165,717 exposures down to the handful that warrant a 2 a.m. response. Across the exposed population we identified **36,365 hosts carrying a high-EPSS vulnerability** and — most consequentially — **3,475 hosts exposed to a vulnerability tied to a known ransomware campaign**. Ransomware association is the signal that most reliably precedes a bad week. These are the exposures where the path from internet scan to encrypted estate has already been walked by someone else.

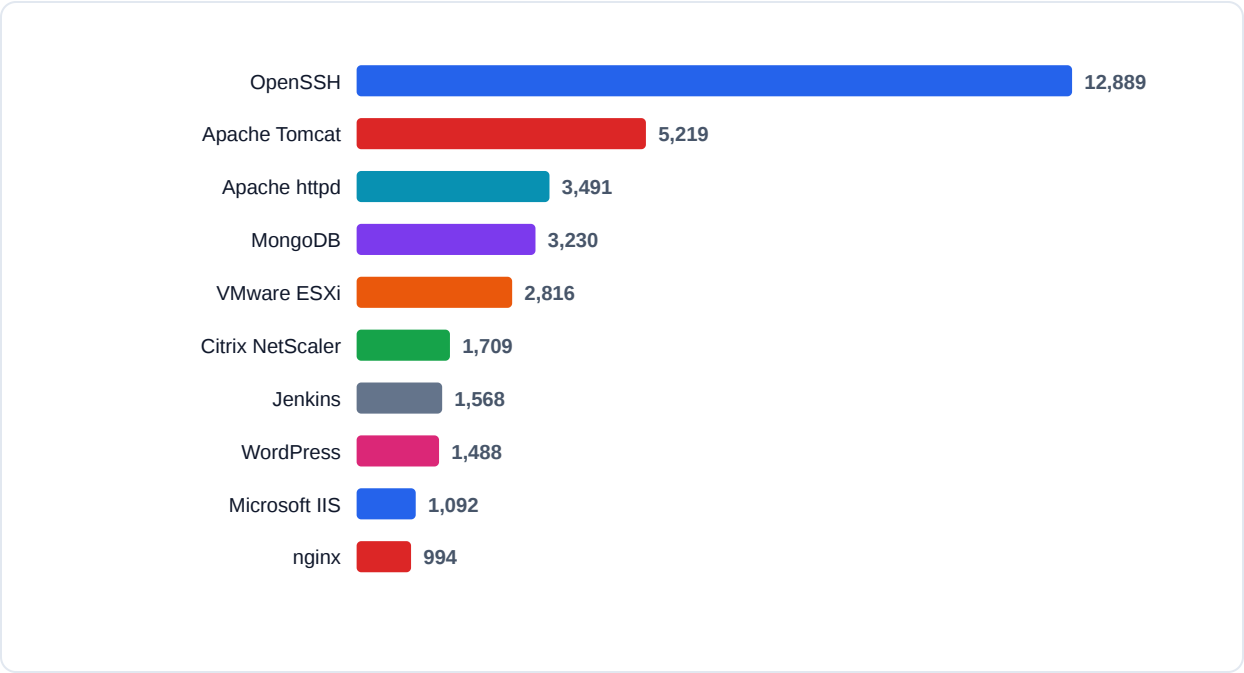
KEV-exposure population at a glance (29 May – 28 Jun 2026) *(window corrected 2026-09-29)*

Measure	Count	What it tells you
Hosts running known-CVE software	36,416	The addressable exposure surface
Distinct host:CVE pairs	165,717	Average host carries multiple known flaws
Hosts on CISA-KEV (actively exploited)	21,299	Confirmed-exploited, not theoretical
Distinct KEV CVEs observed	50	A small, knowable, patchable set
Hosts with high-EPSS vulnerabilities	36,365	High near-term exploitation probability
Hosts on ransomware-linked CVEs	3,475	Pre-positioned for extortion campaigns
Organizations affected	6,519	Breadth across distinct owners
Countries	161	A global, not regional, condition

The arithmetic is sobering. With 165,717 host-to-CVE pairs spread across 36,416 hosts, the average exposed host is not carrying one overdue patch — it is carrying roughly four to five. Exposure compounds: a host left unpatched against one known flaw is, in practice, a host left unpatched against several, because the same operational failure (no inventory, no patch cadence, no ownership) produces all of them at once.

The concentration is the opportunity: ten software families carry the load

The most useful thing about this data is how concentrated it is. The 36,416 exposed hosts are not spread thinly across thousands of obscure products. A short list of widely deployed software families accounts for the overwhelming majority of exposure, and that concentration is a defender's advantage: a remediation campaign aimed at fewer than a dozen products would retire most of the risk in this dataset.



Most-exposed software running KEV-class vulnerabilities, by host count (top 10 of the 36,416-host population)

OpenSSH dominates, appearing on **12,889 hosts** — more than the next three products combined. That is not a comment on OpenSSH's engineering, which is among the most scrutinized in open source; it is a reflection of ubiquity. SSH is the front door to nearly every Linux server on earth, and a backlog in patching it shows up everywhere at once. Behind it sit the workhorses of the public internet: **Apache Tomcat (5,219 hosts)**, **Apache httpd (3,491)**, **MongoDB (3,230)**, **VMware ESXi (2,816)**, and **Citrix NetScaler (1,709)**, followed by **Jenkins (1,568)**, **WordPress (1,488)**, **Microsoft IIS (1,092)**, and **nginx (994)**.

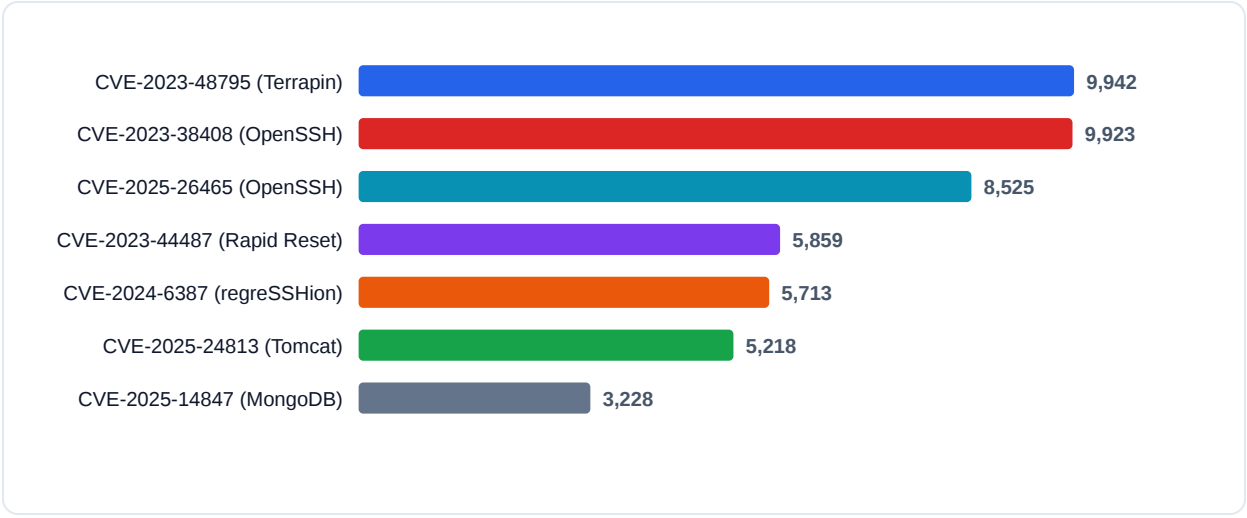
Top exposed software families by host count

#	Software	Hosts	Role / why it matters at the edge
1	OpenSSH	12,889	Remote administration; the front door to Linux fleets
2	Apache Tomcat	5,219	Java application server; frequent RCE target
3	Apache httpd	3,491	General-purpose web server
4	MongoDB	3,230	Data store; doubles as a data-exposure risk
5	VMware ESXi	2,816	Hypervisor; a single host = many workloads, prime ransomware target
6	Citrix NetScaler	1,709	Remote-access gateway; pre-auth flaws yield network entry
7	Jenkins	1,568	CI/CD orchestrator; pipeline compromise = supply-chain reach
8	WordPress	1,488	CMS; vast plugin attack surface
9	Microsoft IIS	1,092	Windows web server
10	nginx	994	Web server / reverse proxy

Two clusters in this list deserve a CISO's particular attention, because they sit at the points of maximum blast radius. **VMware ESXi** is the substrate beneath the data center: compromise one hypervisor and you do not own one server, you own every virtual machine it hosts. ESXi has accordingly become the favored terminal target of ransomware crews, who encrypt at the hypervisor layer to take down dozens of workloads in a single action. Its presence on 2,816 exposed hosts is, line for line, the highest-leverage finding in the table. **Citrix NetScaler** and other remote-access gateways occupy the opposite end of the kill chain — the entry point. These appliances exist to terminate untrusted traffic from the internet, which means a pre-authentication flaw in one is a direct, credential-free route into the internal network. The industry has watched this category produce mass-exploitation events repeatedly; 1,709 exposed NetScaler hosts is 1,709 standing invitations.

Six CVEs, and where the real story diverges from the headline

Drilling from products to specific vulnerabilities sharpens the picture further. The most prevalent individual CVEs in our data concentrate heavily in two technologies — SSH and Tomcat — and the ranking carries a nuance worth stating plainly.



Most-prevalent CVEs across the exposed population, by host count

Top individual CVEs by host count

CVE	Common name	Affected software	Hosts
CVE-2023-48795	Terrapin	OpenSSH	9,942
CVE-2023-38408	OpenSSH ssh-agent PKCS#11 RCE	OpenSSH	9,923
CVE-2025-26465	OpenSSH	OpenSSH	8,525
CVE-2023-44487	HTTP/2 Rapid Reset	Apache Tomcat	5,859
CVE-2024-6387	regreSSHion	OpenSSH	5,713
CVE-2025-24813	Tomcat RCE	Apache Tomcat	5,218

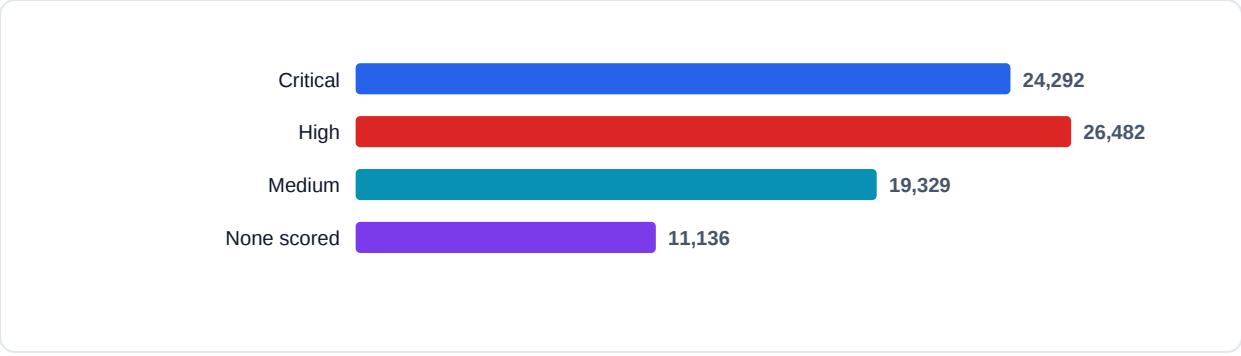
The honest reading of this table requires distinguishing severity from prevalence — a distinction practitioners routinely collapse, to their cost. The two most widespread CVEs, **Terrapin (CVE-2023-48795, 9,942 hosts)** and **HTTP/2 Rapid Reset (CVE-2023-44487, 5,859 hosts)**, are protocol-level weaknesses rather than direct remote-code-execution holes. Terrapin is a prefix-truncation weakness in the SSH transport that can downgrade connection integrity; Rapid Reset is a denial-of-service amplification in the HTTP/2 stream-multiplexing design that was used to drive record-

breaking volumetric attacks against major providers. They are genuine and worth fixing, but they are not "one packet to shell." Treating their large host counts as 9,942 imminent takeovers would overstate the case, and this report will not do that.

The teeth in the table belong to the remote-code-execution entries, even where their host counts are lower. **regreSSHion (CVE-2024-6387)**, present on **5,713 hosts**, is the one to lose sleep over: an unauthenticated remote-code-execution flaw in OpenSSH's server, reachable before any login, on the single most exposed service in the entire dataset. An RCE in the front door of the internet is the archetype of a vulnerability that converts a passive scan into an active intrusion. Alongside it, **CVE-2023-38408** (an OpenSSH ssh-agent PKCS#11 RCE, 9,923 hosts) and **CVE-2025-24813** (a Tomcat RCE, 5,218 hosts) round out the code-execution cluster. The practical instruction is to read this table in two passes: first by prevalence, to understand where your fleet most likely overlaps the herd, then by exploit class, to decide what gets patched tonight versus this sprint. The RCEs get tonight.

Severity at the host level: critical exposure is not the exception

Aggregating across every known flaw on every exposed host produces a severity distribution that should reset any assumption that critical exposure is rare. Of the host-level findings, **24,292 land at CRITICAL severity, 26,482 at HIGH, and 19,329 at MEDIUM**. (Hosts carry multiple findings, so these severity rows count exposures, not unique machines — many hosts contribute to more than one row, consistent with the four-to-five-flaws-per-host average noted above.)



Host-level KEV exposure by severity rating

Severity distribution of host-level findings

Severity	Findings	Operational implication
Critical	24,292	Immediate remediation; assume active interest
High	26,482	Prioritized patch cycle
Medium	19,329	Scheduled remediation; monitor for escalation

The signal here is that CRITICAL and HIGH together vastly outweigh MEDIUM. The exposed internet is not lightly bruised; it carries serious, exploitable weaknesses at the top of the severity scale as its normal state. For a CISO, the strategic takeaway is that the problem is not detection — these flaws are catalogued, fingerprintable, and visible to anyone with a scanner, attacker or defender alike — but cadence. The vulnerabilities are known. The exploits are public. The patches exist. What is missing, at 21,299 hosts, is the operational discipline to close the loop before someone else closes it for you.

The patch gap: why "a fix is available" is the most dangerous phrase in security

Every number in this chapter is a measurement of one thing under different lights: the patch gap, the interval between a fix becoming available and that fix being applied to a production system. It is the most studied and least solved problem in operational security, and our data quantifies its current width across the public internet.

The gap persists for reasons that are organizational before they are technical:

- **You cannot patch what you do not know you run.** Asset inventory is the unglamorous foundation of the whole discipline, and it is routinely incomplete. Shadow IT, forgotten cloud instances, contractor-stood-up appliances, and acquired infrastructure all run software that no one is tracking — and therefore no one is patching. The 6,519 organizations in this dataset did not choose to expose 165,717 known flaws; in most cases they simply did not have a current picture of what they were running.
- **Edge appliances are the worst-patched class of all.** The Citrix NetScaler and VMware ESXi findings are not accidents. Gateways, hypervisors, and firewalls are perceived as "infrastructure that just works," are change-controlled to the point of paralysis, and frequently sit outside the patch automation that covers ordinary servers. They are simultaneously the highest-value targets and the slowest to be remediated — the exact inversion of what risk would dictate.

- **Patching is a change, and changes carry risk.** Every patch is a small bet that the fix will not break production. Operators who have been burned by a bad update rationally hesitate, and that hesitation, multiplied across a fleet, becomes a standing exposure window. The answer is not to patch recklessly but to make patching routine, tested, and reversible — to lower the cost of the change so the bet becomes easy.
- **Internet-facing assets are the least forgiving place to carry this gap.** An unpatched internal system is a latent risk; an unpatched internet-facing one is a live one, scanned continuously by adversaries running the same fingerprinting that produced this report. The 21,299 KEV-exposed hosts are not waiting to be discovered. They have been discovered, repeatedly, by everyone looking.

The historical record removes any doubt about where this leads. regreSSHion, the unauthenticated OpenSSH RCE on 5,713 hosts in our data, is exactly the shape of flaw that produces a mass-exploitation event. The MOVEit campaign demonstrated the model end to end: a single known vulnerability in a single widely deployed appliance, exploited in parallel across every exposed instance an adversary could find, with victim organizations learning they were affected only after their data had been taken. The precondition every time is a known, exposed, unpatched weakness — the precise condition we count at 21,299 hosts. The vulnerabilities in this chapter are not predictions of future breaches. They are the standing inventory from which the next campaign will draw.

What to do with this

The concentration that makes this exposure dangerous also makes it tractable. Fifty KEV CVEs, ten software families, and six dominant vulnerabilities account for the bulk of the risk in 165,717 exposures — a target set small enough to act against deliberately rather than drown in. We recommend a sequenced response:

1. **Treat the KEV catalog as your priority queue, not your CVSS spreadsheet.** The 21,299 KEV-exposed hosts are where exploitation is confirmed. Remediate these before anything that is merely rated CRITICAL but not known-exploited. Active exploitation outranks theoretical severity every time.
2. **Escalate the 3,475 ransomware-linked exposures to incident tempo.** A ransomware-associated, internet-facing, unpatched host is the most reliable single predictor of a major incident in this dataset. These do not wait for the next maintenance window.

3. **Lead with the high-blast-radius edge: ESXi first, remote-access gateways second.** The hypervisor is where ransomware terminates; the gateway is where intrusions begin. Patching both retires disproportionate risk per host touched.
4. **Read the top-CVE list by exploit class, not headline.** Patch the RCEs — regreSSHion, the OpenSSH agent flaw, the Tomcat RCE — on the tightest possible clock. Schedule the protocol-level weaknesses (Terrapin, Rapid Reset) into the normal cycle. Do not let large host counts on lower-severity flaws crowd out the genuinely critical few.
5. **Close the inventory gap so the patch gap cannot reopen.** Every exposure here traces to something that was running but not being watched. Continuous external discovery — knowing what you actually expose, from the attacker's vantage point — is the precondition for every other step. You can map your own exposure against this dataset with EchelonGraph's surface scanner, trace blast radius through your estate in the attack graph, and track the live KEV-exposed internet on the KEV Exposure radar.

The recurring lesson of the breach record — Equifax, Capital One, MOVEit, and the campaigns that will follow them — is not that attackers are uniquely sophisticated. It is that defenders leave known doors unlocked at scale. This chapter is a census of those doors as they stood across roughly five weeks of mid-2026 *(window corrected 2026-09-29)*: 36,416 hosts, 21,299 of them open to vulnerabilities attackers are using today. The fix for nearly every one of them already exists. The only question this data leaves open is who applies it first.

Exposed Data Stores

A database that answers a query from anyone on the internet, without first asking who is asking, is not a misconfiguration in the abstract. It is an open door to whatever sits behind it — session tokens, customer records, telemetry, message queues, cached credentials. The category is unforgiving because the failure mode is silent: the store works exactly as designed, serves traffic, and shows no error. Nothing in the application breaks. The only signal that anything is wrong is that the wrong people can read it, and by the time that signal arrives it is usually a ransom note or a press inquiry.

Across the observation window of 29 May to 28 June 2026 *(window corrected 2026-09-29)*, EchelonGraph's passive discovery identified **6,607 open, unauthenticated databases** reachable from the public internet, spread across **1,723 organizations** and **120 countries**. These are not hosts with weak passwords or guessable credentials. These are stores that accept connections and return data with no authentication step at all — the equivalent of a filing cabinet left on the sidewalk with the drawers open.

> **6,607 open unauthenticated databases, across 1,723 organizations in 120 countries — each one reachable with no login.**

What we measured, and what we did not

This figure deserves precision, because the temptation in this category is to inflate. We did not. Our methodology is passive, public-data, and strictly detect-only: we observe that a store is listening and that it answers an unauthenticated handshake *(data-store check corrected 2026-09-29)*. **We see the open door. We do not walk through it.** We never read contents, never enumerate keyspaces *(data-store check corrected 2026-09-29)*, never count rows, and never exfiltrate a single record. Consequently this chapter contains no claim about "millions of exposed records," because we did not look inside to count, and any such number would be a fabrication *(method superseded 2026-09-29)*.

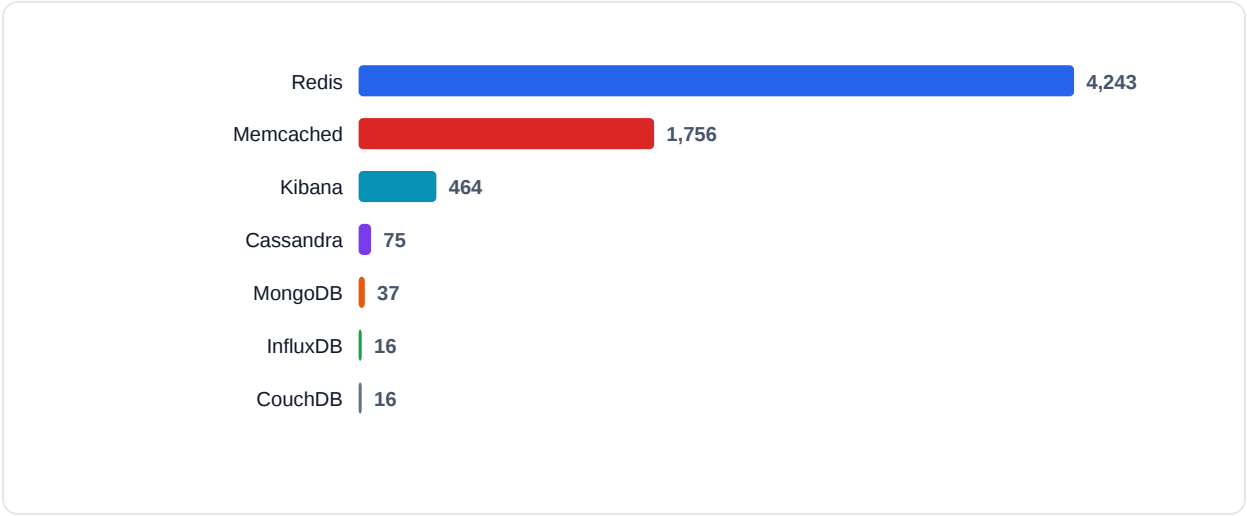
The classification is deliberately conservative. Of the open stores observed, only a small number could be confirmed — from public banner metadata alone, without

inspecting data *(data-store check corrected 2026-09-29)* — as carrying regulated content: **three were classified as holding PII and one as in PCI scope**. That is a floor, not a ceiling. It reflects what is provable from the outside without reading the store, and it almost certainly understates the true regulatory exposure, because most open databases reveal nothing about their contents from the doorway *(method superseded 2026-09-29)*. The honest reading is this: 6,607 doors stand open; the contents behind the overwhelming majority are unknown to us, by design, and should be assumed sensitive by their owners until proven otherwise. You can check your own exposure with the surface scanner, and the live radar behind these figures is published at exposed-databases.

One framing caveat carries over from the broader report and applies here. This is an inaugural baseline gathered over roughly four to five weeks *(window corrected 2026-09-29)*, not a multi-year trend line. The numbers describe a snapshot of the internet as it was during a single month *(window corrected 2026-09-29)*. They are large enough to be alarming on their own terms; they are not yet a trajectory.

The shape of the exposure: caches and search nodes, not "the database"

The most important finding in this chapter is a structural one, and it changes how the risk should be triaged. When people imagine an exposed database, they picture the system of record — the customer table, the orders ledger, the primary store. That is not what the internet is leaking. The exposure is dominated by two categories of infrastructure that operators rarely think of as "databases" at all: in-memory caches and search and observability nodes.



Open unauthenticated data stores by engine, observed 29 May – 28 June 2026 (host counts) *(data-store check corrected 2026-09-29) (window corrected 2026-09-29)*. Redis and Memcached together account for the overwhelming majority of exposures.

Exposed unauthenticated data stores by engine (host counts)

Engine	Type	Open hosts
Redis	In-memory cache / key-value store	4,243
Memcached	In-memory cache	1,756
Kibana	Search / observability front-end	464
Cassandra	Wide-column store	75
MongoDB	Document store	37
InfluxDB	Time-series store	16
CouchDB	Document store	16

Redis alone accounts for 4,243 of the 6,607 open stores — nearly two in three. Add Memcached's 1,756 and the two in-memory caches together make up the great majority of the entire exposed population. This is the central lesson of the data, and it is a lesson about defaults and mental models rather than about exotic attacks.

Redis and Memcached were both designed for a world inside a trusted network boundary. For much of their history neither shipped with authentication enabled by default; both were built to be fast, simple, and adjacent to the application, on the assumption that a firewall or a private subnet stood between them and the internet. When that assumption fails — a security group left open, a container published to a

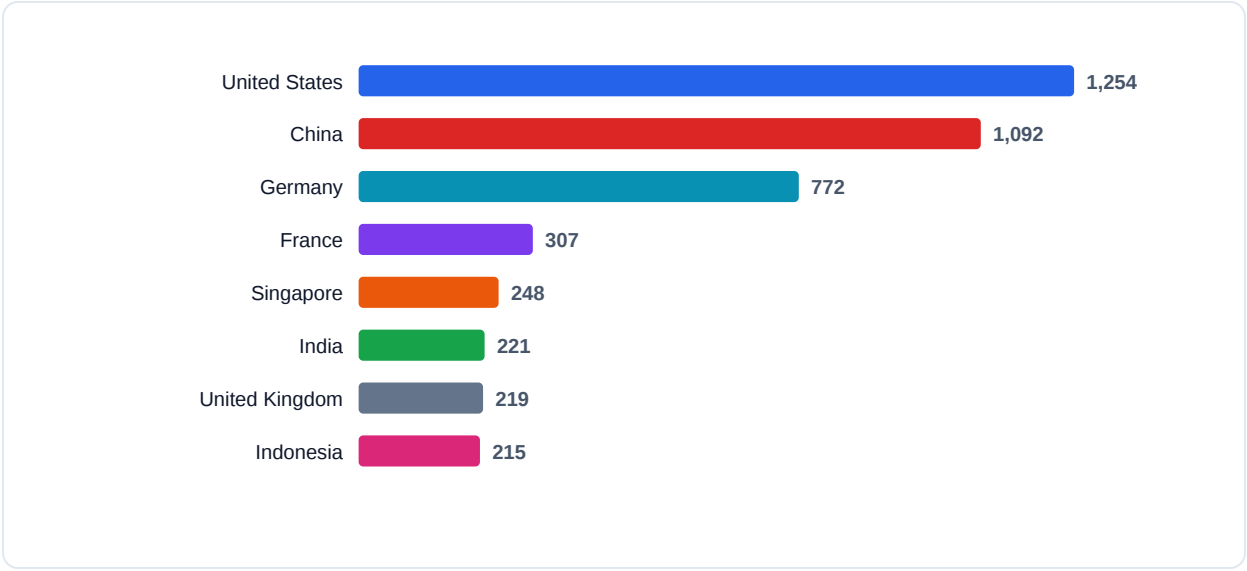
public interface, a cloud instance with a public IP and a permissive ingress rule — the store is simply on the internet, answering to anyone, with no credential to stop them. The operator did not "turn off" authentication. They never turned it on, because the threat model they inherited said they did not need to.

The risk is frequently dismissed because of what these stores nominally hold. "It's only a cache" is the reflex. That reflex is wrong on two counts. First, caches routinely hold exactly the material an attacker wants: session identifiers, authentication tokens, rate-limit state, password-reset nonces, queued jobs, and copies of records pulled forward from the primary database for speed. A session token read from an open cache is a logged-in user. Second, an exposed Redis or Memcached instance is not only a confidentiality problem; it is often a foothold. Write access to an unauthenticated store lets an attacker poison cached values, corrupt application state, and in well-documented Redis cases abuse the store's own persistence and scripting features to achieve code execution on the host. The cache is not the periphery of the system. It is frequently the soft center.

The 464 exposed Kibana instances carry a related but distinct danger. Kibana is the window onto an Elasticsearch cluster — logs, metrics, traces, and whatever application data has been shipped into the index for analysis. An open Kibana front-end is a search interface over an organization's operational telemetry, and operational telemetry is where secrets go to hide: tokens printed in debug logs, full request bodies, stack traces with connection strings. The remaining engines — Cassandra, MongoDB, InfluxDB, CouchDB — appear in smaller numbers, but each represents a system that can hold a system of record outright. MongoDB in particular has a long public history in this category, and its presence here, however modest in count, is a reminder that the document stores are not immune; they are simply less numerous in this sample.

Where the open doors are

Exposure of this kind is not concentrated in any one jurisdiction or any one cloud. It tracks the global distribution of internet-facing infrastructure, which means it tracks the global distribution of cloud regions and hosting density. The 6,607 stores span 120 countries; the table below shows the six with the highest observed counts.



Top six countries by count of exposed unauthenticated data stores, 29 May – 28 June 2026 *(window corrected 2026-09-29)*.

Exposed unauthenticated data stores by country (top six, host counts)

Country	Open hosts
United States	1,254
China	1,092
Germany	772
France	307
Singapore	248
India	221

The United States (1,254) and China (1,092) lead, together accounting for more than a third of all observed exposures and reflecting the two largest concentrations of public cloud and hosting capacity in the world. Germany (772) and France (307) follow, with Singapore (248) and India (221) rounding out the top tier — each a major regional cloud hub. The pattern is unsurprising and, for a security leader, clarifying: this is not a problem confined to one regulatory regime or one provider's customers. Wherever an organization stands up infrastructure quickly and at scale, the same default-trust assumptions follow it, and a fraction of those deployments end up listening on the open internet. The remaining 114 countries in the dataset each contribute fewer hosts, but their collective tail confirms that the exposure is genuinely global rather than an artifact of a handful of careless networks.

For multinational organizations the geographic spread carries a compliance edge worth naming. An open store in a European region is a candidate GDPR matter the moment it holds personal data; one in PCI scope is a candidate for cardholder-data findings regardless of where it sits. Because we deliberately do not read contents, we cannot tell an operator which of their stores crosses those lines — but they can, and the geographic and engine breakdown above is a map of where to look first *(method superseded 2026-09-29)*. Mapping that exposure against regulatory obligations is the work the compliance surface is built for.

The precedent is not hypothetical

The reason this category warrants a chapter of its own, rather than a footnote, is that the open-database failure mode has produced some of the defining data-loss events of the last decade. The pattern is consistent: a store that should have been private was reachable, no credential stood in the way, and the contents left the building before anyone with authority noticed.

The pattern reaches the newest organizations as readily as the oldest. The rush to ship — whether a startup, an AI lab, or a global enterprise standing up a new region — recreates the trusted-network assumption that the default-no-auth cache was built for, and the internet is indifferent to the intention. A store stood up fast, on the assumption that something else would keep it private, is exactly the door this chapter counts.

Landmark public breaches over the years reinforce the same structural point even when the specific technical mechanism differed. Catastrophic exposures of personal and financial data have repeatedly traced back to data that was reachable when it should not have been; supply-chain campaigns against widely used file-transfer and data-handling systems have shown how a single reachable system can become an exfiltration vector across many downstream organizations at once. None of these were caused by an open Redis instance, and we make no such claim. What they share with the 6,607 stores in this dataset is the underlying lesson: when sensitive data is reachable and the barrier protecting it fails or was never raised, scale is on the attacker's side, and the gap between exposure and exploitation is measured in hours, not months.

That is the weight behind the conservative count. We classified only three stores as PII-bearing and one as PCI-scoped, because that is all we could prove from the doorway without reading inside *(method superseded 2026-09-29)*. But every one of the 6,607 open doors is, from the perspective of its owner, a store whose contents are now governed by whoever finds it first. The right posture is not to take comfort in the small

confirmed-sensitive number; it is to assume that any unauthenticated store reachable from the internet is compromised until taken offline and re-secured.

What to do before the radar finds you

The remediation for this category is unglamorous and, mercifully, well understood. It does not require new tooling so much as the discipline to apply what already exists.

1. **Assume no store belongs on the public internet until proven otherwise.** The default for any cache, document store, search node, or time-series database should be a private network with no public ingress. Public reachability should be a deliberate, reviewed, and rare decision — never an accident of a default security group or a published container port.
2. **Turn authentication on, everywhere, including the caches.** The engines most represented here — Redis and Memcached — are precisely the ones operators are most likely to leave open because "it's just a cache." Enable authentication and, where available, transport encryption, on every store regardless of what it nominally holds. The cache holds tokens; treat it like it holds tokens.
3. **Close the gap between "internal" and "reachable."** Most of these exposures stem from a single broken assumption: that a network boundary that once existed still exists. Continuously verify, from the outside, what your organization actually presents to the internet. An external, attacker's-eye view is the only reliable way to catch the store that someone published last week.
4. **Inventory before you remediate.** You cannot secure a store you do not know is running. The geographic and engine breakdowns above are a triage order: in a large estate, the in-memory caches are both the most numerous exposure and the easiest to overlook, and the search front-ends are the highest-value single targets.

The 6,607 figure is a measurement of how often the boundary fails in practice, across the whole internet, in a single month *(window corrected 2026-09-29)*. It is large because the assumption that produces it — that the store sits safely behind a wall — is one of the most durable and most frequently violated assumptions in modern infrastructure. The stores in this dataset are not exotic. They are ordinary caches and search nodes, run by ordinary organizations, that ended up answering to the wrong audience. The work is to find yours before someone else does, and to do it knowing that the door, once open, does not announce itself.

EchelonGraph's view of this surface, the engine and country breakdowns behind these figures, and the means to check your own estate are maintained at the exposed-databases radar and the surface scanner.

CHAPTER 6

Secret Sprawl: Exposed Keys & Credentials

A credential is a skeleton key. Unlike a software vulnerability, it requires no exploit chain, no memory-corruption primitive, no race window — an attacker who finds a valid access key simply authenticates and is, from the platform's point of view, you. There is no patch for a leaked secret; the only remedy is rotation, and rotation only helps if you know the secret leaked. This is what makes credential exposure the quietest and most consequential class of finding in this report. The internet is not merely running vulnerable software. It is publishing the keys to that software, in plaintext, on web servers and in public source repositories, where any party with a browser or a `git clone` can collect them at scale.

Over the observation window of 29 May to 28 June 2026 *(window corrected 2026-09-29)*, EchelonGraph passively catalogued credential exposure across two distinct surfaces: secrets served directly off the public web (misconfigured web roots leaking configuration files), and secrets committed to public Git repositories. Both were collected detect-only — we fingerprint the *presence* and *type* of a secret from publicly retrievable artifacts; we never use a discovered credential, never authenticate, never read the account behind it *(method superseded 2026-09-29)*. The numbers below are therefore a conservative floor. They describe what the entire internet could already see during a single four-to-five-week baseline *(window corrected 2026-09-29)*, not a multi-year accumulation.

> **≈2,600 live AWS credentials were recoverable from 2,571 exposed `.env` files across just 750 web hosts — alongside 619 more secrets sitting in 478 public Git repositories.**

The web surface: configuration files served as plaintext

The dominant failure mode is mundane and almost entirely self-inflicted: an application's environment file, deployed into a directory the web server will serve, with no access control in front of it. The `.env` file — a convention popularized by twelve-

factor application design and frameworks such as Laravel, Symfony, Django and Node.js — is meant to live *beside* an application and never be reachable over HTTP. When the document root is misconfigured, or the file is dropped into `public/` by mistake, a request to `/.env` returns the application's entire secret inventory: database passwords, third-party API keys, signing secrets, SMTP credentials, and cloud access keys.

Across 750 hosts, EchelonGraph observed 2,571 distinct exposed `.env` files — the single largest category by a wide margin. The remaining exposure types are smaller in count but, in several cases, more severe per instance:

Exposed-secret artifacts on the public web (29 May – 28 Jun 2026; 750 hosts) *(window corrected 2026-09-29)*

Exposure type	Count	What it leaks
`.env` file	2,571	Full application secret inventory — DB, API keys, cloud keys
git-config exposed	80	`.git/config` — remote URLs, often with embedded tokens
git-exposed (repo on web root)	61	Reconstructable source tree, including committed secrets and history
cred-file	17	Standalone credential files (e.g. cloud SDK / service-account files)
db-dump	3	Database export served over HTTP
private-key	2	Private key material reachable without authentication

The two git categories deserve specific attention. An exposed `.git/config` (80 hosts) frequently embeds the credential used to clone the repository directly in the remote URL. An exposed `.git` directory served from the web root (61 hosts) is worse still: it lets an attacker reconstruct the application's entire source tree *including its commit history*, which is the single most reliable place to find secrets that a developer believed they had removed. A key deleted in the working tree but never purged from history is fully recoverable. The 3 database dumps and 2 reachable private keys are low in count but are each effectively a complete compromise of the asset behind them; we classify them as critical without qualification.

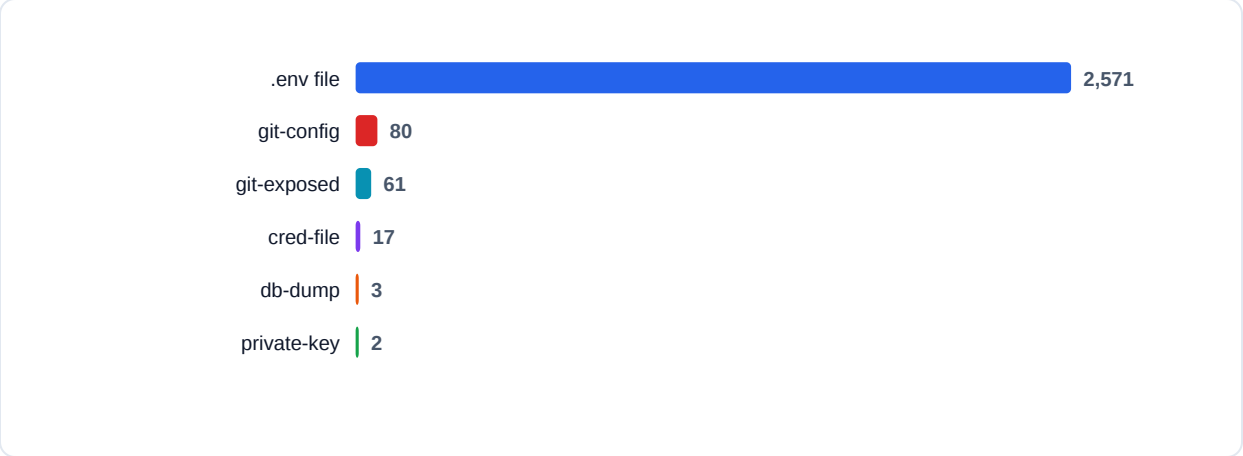
What the keys actually are

The composition of the secrets matters more than the file count, because not all secrets are equal. A leaked SMTP password lets an attacker send mail; a leaked cloud access key lets an attacker spend money, exfiltrate data, and pivot. The overwhelming majority of what we recovered from the web surface is cloud infrastructure credentials.

Secret material identified on the public web

Secret type	Count
AWS secret access key	1,319
AWS access key ID	1,284
GitHub token	2
PEM private key	1

The access-key ID and the secret access key are the two halves of an AWS credential pair; observed together, they constitute roughly 2,600 usable AWS credentials harvestable from the public web during a single baseline window *(window corrected 2026-09-29)*. An AWS key pair is not a password to one application — it is a programmatic identity inside an account, and what it can do is bounded only by the IAM policy attached to it. In practice, far too many of these keys are over-permissioned: granted broad or administrative access for convenience during development and never scoped down. That is precisely the failure mode that turns a single leaked key into a full account compromise.



Exposed secrets on the public web, by artifact type (n=750 hosts). The `.env` file dominates at 2,571 instances; git-config, git-exposed, cred-file, db-dump and private-key form the long tail.

Mapped to severity, the picture is stark: of the secret rows we classified, 1,386 were **critical** and 1,304 were **high**, with only 46 rated medium. There is almost no benign tail here. Web-served secret exposure is, by its nature, a high-severity finding — the artifact is reachable by anyone, the credential is live until rotated, and the discovery cost to an adversary is a single unauthenticated HTTP request.

Web secret findings by severity

Severity	Findings
Critical	1,386
High	1,304
Medium	46

The second surface: secrets committed to public Git

If the web surface is about files that should never have been served, the Git surface is about secrets that should never have been committed. Source-code hosting is the other end of the same pipe: a developer hardcodes a token to make something work, commits it, and pushes it to a public repository — at which point the secret is not only visible in the current tree but permanently embedded in the immutable commit history, replicated to every fork and clone, and indexed by anyone who watches the public event firehose. Removing the file in a later commit does not remove the secret; only a history rewrite plus rotation does, and the credential should be assumed compromised the moment it lands in a public repo.

EchelonGraph observed 619 secrets across 478 public repositories during the window *(window corrected 2026-09-29)*. The provider distribution is broader than the web surface and tells a different story — one increasingly shaped by the AI build-out:

Leaked secrets in public Git repositories, by provider (largest buckets; 619 secrets across 478 repos)

Provider / type	Count
Generic (high-entropy / unattributed)	249
Google (API / service credentials)	228
AWS	63
Telegram (bot tokens)	53
Discord (bot / webhook tokens)	13
OpenAI (API keys)	2
HuggingFace (access tokens)	2

Several patterns are worth naming. **Google credentials (228)** rival the generic high-entropy bucket and lead all named providers — API keys and service-account material that, like AWS keys, can grant programmatic access to cloud projects and data. **Telegram and Discord bot tokens (53 and 13)** reflect the proliferation of automation and bot frameworks whose tokens are routinely hardcoded; a leaked bot token can let an attacker hijack the bot, read channel traffic, or use it as a command-and-control or data-exfiltration channel. And while **OpenAI and HuggingFace keys (2 each)** are small in absolute count, they are an early signal of an emerging category: AI service credentials. A leaked LLM API key is a direct financial-abuse vector — an attacker runs inference on the victim's account, and the bill arrives at month's end — and a model-registry token can expose proprietary or fine-tuned models. This dovetails with the AI attack surface documented in Chapter 3: organizations are not only standing up AI infrastructure faster than they can secure it, they are leaking the keys to it.

By severity, the Git surface skews high but carries a real critical core: 301 high, 273 medium, and 45 critical. The medium share is larger than on the web surface — many committed secrets are lower-privilege tokens or are partially mitigated by scope — but 45 critical leaks in a single baseline *(window corrected 2026-09-29)*, each a live credential to production infrastructure, is not a rounding error.

Git-leaked secrets by severity (n=619)

Severity	Findings
High	301
Medium	273
Critical	45

It is the permissions, not just the key

The defining lesson of credential exposure is that the damage is governed less by the leak itself than by what the leaked identity is allowed to do. A leaked secret, on its own, is just a string. Its blast radius is determined entirely by the privileges attached to it: a read-only token scoped to a single resource is an incident to clean up; an administrative key is a full account compromise. The same convenience culture that puts a key in a web-served `.env` file — or hardcodes it into a commit — tends also to grant that key more privilege than the task it was created for actually needs.

That is the exact risk profile of the ≈2,600 AWS keys in this chapter. Each one is, by itself, just a string. Its blast radius is determined entirely by the IAM policy behind it — and over-broad grants turn a single leaked key into a programmatic identity that can spend money, read data, and pivot across the account. The combination is the toxic part: a broadly-scoped key, reachable by anyone, never rotated. Two of those three conditions are within the operator's direct control before any attacker is involved.

> **Why "least privilege" is the load-bearing control.** You cannot guarantee a key will never leak — humans misconfigure web roots and commit secrets, and they always will. What you *can* guarantee is that when a key leaks, it opens as little as possible. A read-only key scoped to one bucket is an incident; an administrator key is a breach. Scoping every credential to its minimum necessary permission converts the unavoidable leak from catastrophic to survivable — it is the one control that limits blast radius after every other safeguard has already failed.

Why this is structurally worse than a vulnerability

It is worth stating plainly why credential sprawl deserves separate treatment from the known-exploited-vulnerability exposure in Chapter 4. A vulnerability requires an exploit;

a leaked credential requires only a login. A vulnerability is often noisy to exploit and may trip detection; authenticating with a valid key is, by design, indistinguishable from legitimate use — the access looks exactly like the developer it was stolen from. And a vulnerability has a vendor patch with a clear before-and-after; a leaked secret has no patch, only rotation, and rotation is only triggered if someone realizes the leak occurred. The absence of an exploit step is precisely what makes this class so dangerous: it removes the friction and the signal that defenders normally rely on.

- **No exploit required.** Possession of the credential is the attack. The barrier to entry is a browser or a clone command.
- **No patch exists.** The only fix is rotation — and rotation depends on detection, which most organizations lack for their own externally-visible secrets.
- **Detection-evading by construction.** Valid-credential access blends into normal authenticated traffic and rarely fires an alert.
- **Permanent in Git.** Once a secret is in public commit history, deleting the file does not remove it; assume it is compromised the moment it is pushed.

What this means for defenders

Two surfaces, one root cause: secrets handled as plaintext data rather than as managed, scoped, rotatable identities. The remediation pattern is well understood and does not require new technology — it requires discipline applied consistently.

1. **Get secrets out of files and into a secrets manager.** Application secrets should be injected at runtime from a managed store, never committed to source and never written into a web-served directory. This single change eliminates both surfaces in this chapter.
2. **Scope every credential to least privilege.** Assume each key will eventually leak and ensure that when it does, it opens the minimum possible. This is the control that most directly limits the blast radius of the ≈2,600 keys above.
3. **Add pre-commit and CI secret scanning.** Catch the secret before it is pushed, not after. The 478 repositories in this chapter are evidence that secrets reach public Git at scale without it.
4. **Block `.env`, `.git`, and credential files at the web server.** Deny-by-default for dotfiles and version-control directories at the edge stops the dominant 2,571-file web exposure outright, independent of how the application is deployed.

5. **Rotate on the assumption of exposure, and monitor your own external footprint.** Any secret that has ever touched a public surface should be rotated and the old value revoked. You cannot rotate what you do not know has leaked — continuous external secret discovery is the detection layer that makes rotation possible.

An organization can verify its own exposure against the same passive methodology used to produce these figures via the EchelonGraph external surface scanner and the dedicated exposed-keys and leaked-credentials radars. The cost of looking is a single scan; the cost of not looking is measured in the gap between when a key leaks and when you find out — a gap that, for the keys catalogued here, is currently the entire internet's to exploit.

CHAPTER 7

Subdomain Takeover

A subdomain takeover is the rare exposure that hands an attacker a legitimate piece of your brand. Where most of the surfaces in this report leak something the attacker can read, a dangling subdomain lets the attacker *publish* — to serve content, set cookies, and accept credentials under a hostname that the browser, the corporate proxy, and the human reader all treat as yours. The address bar is genuine. The lock icon is genuine. Only the content underneath has been replaced. That combination is what makes the class disproportionately useful for phishing and session theft, and disproportionately hard for victims to spot.

Across the observation window of 29 May to 28 June 2026 *(window corrected 2026-09-29)*, EchelonGraph passively catalogued **7,952** subdomains delegated to third-party hosting services via CNAME records, and identified **134** of them as confirmed vulnerable to takeover through a dangling CNAME — a delegation pointing at a provider resource that no longer exists and can be re-claimed. These were detected, not exploited: the platform observed the public DNS delegation and the provider's unclaimed-resource response, and never registered, served from, or otherwise took control of any name. The figures below are a single inaugural baseline drawn over roughly four-to-five weeks *(window corrected 2026-09-29)*, not a multi-year trend.

> **134 subdomains confirmed hijackable via a dangling CNAME — 1.7% of the 7,952 third-party-delegated names observed. Each is a working, trusted hostname an attacker can claim and serve content from.**

Why a dangling CNAME is dangerous

The mechanism is mundane, which is precisely why it persists. An organization points `blog.example.com` or `shop.example.com` at a managed service — a Shopify storefront, a GitHub Pages site, a Heroku app — by adding a CNAME record that aliases the subdomain to a provider-controlled hostname. Months or years later the

underlying service is decommissioned: the store is closed, the repository deleted, the app torn down. The provider releases the backing resource. The DNS record, however, is rarely cleaned up at the same time, because DNS and application lifecycles sit with different teams and different change processes. The subdomain is now *dangling*: it still resolves, still points at the provider, but the target no longer exists.

On most platforms that allow customers to claim arbitrary hostnames, an attacker who notices the dangling record can create a new account, register the same hostname the CNAME still points to, and immediately begin serving content. No DNS access to the victim's zone is required — the victim's own delegation does the work. From that moment the attacker controls a live, fully-resolving page on the victim's domain, served over the provider's infrastructure and, in the common case, fronted by a provider-issued TLS certificate that turns the browser padlock green. To every external observer the page is part of the victim's web estate.

The damage follows from the trust that a hostname carries:

- **Credential phishing that survives scrutiny.** The single most effective control users are told to apply — "check the URL" — fails here, because the URL is correct. A login form on a hijacked `support.example.com` or `careers.example.com` defeats the link-inspection habit, passes mail-gateway domain reputation checks, and clears the URL allow-lists that security-awareness training tells staff to rely on. The phish is hosted on the brand it impersonates.
- **Cookie theft and session hijacking.** Browsers scope cookies by domain. A cookie set with `Domain=.example.com`, or any session that trusts the parent domain, is readable from any subdomain of `example.com` — including the one the attacker now controls. A hijacked subdomain can therefore read session cookies that were never meant for it, and can set its own cookies that the parent application will subsequently honor (session fixation). Single sign-on and OAuth flows that treat sibling subdomains as same-origin-adjacent are especially exposed.
- **Content Security Policy and all-list bypass.** Security policies frequently trust `*.example.com` wholesale — for script sources, frame ancestors, CORS origins, and redirect targets. A hijacked subdomain is inside that trust boundary, so it can host malicious script, frame the real application for clickjacking, or act as a sanctioned open-redirect that launders attacker URLs through the legitimate brand.

- **Reputation and deliverability collateral.** A subdomain caught serving malware or phishing is the organization's domain being blocklisted, flagged by safe-browsing services, and cited in abuse reports — with the cleanup, delisting, and customer-trust cost landing on the victim, not the attacker.

None of this requires a software vulnerability, a leaked credential, or an exploit chain. It requires only a CNAME that outlived the thing it pointed at. The barrier to entry is a free provider account.

Where the dangling names point

The 7,952 delegated subdomains concentrate on a small number of high-volume hosting platforms — the same managed services that make it trivial to stand up a site, and equally trivial to leave a record behind when that site is removed. The distribution below is by observed subdomain (host) across the top five providers.

Third-party-delegated subdomains by hosting service (hosts observed, top five providers)

Subdomains delegated to third-party services, by provider (observation window 29 May – 28 Jun 2026) *(window corrected 2026-09-29)*

Hosting service	Subdomains observed	Share of top five
Shopify	4,869	63.2%
GitHub Pages	1,563	20.3%
Heroku	773	10.0%
SmugMug	331	4.3%
Fastly	170	2.2%
Top five total	**7,706**	**100%**

Two patterns in this distribution deserve a CISO's attention. First, the providers that dominate are the ones whose value proposition is self-service: anyone can open a Shopify store, publish a GitHub Pages site, or deploy a Heroku app in minutes, which is exactly the property an attacker needs to re-claim a released hostname. Second, the volume is driven by marketing and developer convenience, not by core engineering. Storefronts, campaign microsites, documentation pages, status pages, and project sites are stood up by teams outside the central infrastructure function and are torn down

without a DNS de-provisioning step. The CNAME is created in a hurry and forgotten at leisure.

Of the 7,952 names observed, **134** were confirmed to be in the dangling, claimable state — concentrated on the self-service providers where re-registration of an unclaimed hostname is open to any account, including Shopify, GitHub Pages, and Heroku *(these platforms are named from the 7,952 observed; the report does not say which platforms the 134 confirmed are on; corrected 2026-09-30)*. The remainder resolve to live, claimed services and are not vulnerable; their presence in the dataset is simply the population from which takeovers emerge. The ratio matters more than either number alone: roughly **one in sixty** externally-delegated subdomains in this sample was a working takeover candidate. For an enterprise with hundreds or thousands of third-party-hosted names, that ratio implies a standing inventory of live, brand-trusted footholds that no one is watching.

Subdomain takeover exposure at a glance

Metric	Value
Third-party-delegated subdomains observed	7,952
Confirmed vulnerable (dangling CNAME, claimable)	134
Confirmed-vulnerable rate	1.7%
Distinct hosting services in top five	5
Detection method	Passive DNS + provider unclaimed-resource signal; detect-only *(takeover check corrected 2026-09-29)*

Who this hits

Subdomain takeover is not a problem of immature organizations. It is a problem of *large* organizations, because exposure scales with the breadth of an estate and the number of teams allowed to create DNS records. The most exposed profiles are:

- **Consumer and retail brands** with sprawling Shopify and marketing-platform estates — one storefront or landing page per campaign, region, or product line — and no lifecycle owner for the records left behind when a campaign ends.

- **Engineering-heavy companies** that publish documentation, status pages, and project sites on GitHub Pages and PaaS providers, where a deleted repository or torn-down app routinely outlives its CNAME.
- **Organizations that have grown by acquisition**, inheriting DNS zones whose history no current employee fully knows, including delegations to services that were cancelled before the acquisition closed.
- **Any brand whose customers are accustomed to logging in** at subdomains — `portal.`, `account.`, `my.`, `secure.` — because a hijacked sibling under the same parent domain inherits the credibility, and frequently the cookie scope, of the real one.

The defender's structural disadvantage is asymmetry of attention. Attackers enumerate an organization's subdomains continuously and cheaply, using the same public certificate-transparency logs and DNS data that this report draws on; finding a dangling record is a scriptable, undifferentiated commodity activity. Defenders, by contrast, typically discover dangling records only after an incident — when the hijacked subdomain is already serving a phishing page and the abuse reports have started. The window between a service being decommissioned and the dangling record being noticed by the right party is the entire exposure, and on the volunteer-to-clean-up side that window is open-ended.

Incident parallels

Subdomain takeover lives in the same family as the breaches that have defined the last decade of cloud and web exposure, and it is instructive to place it beside them. Where Equifax in 2017 turned on an unpatched, known-vulnerable component — the pattern this report quantifies in its known-exploited-vulnerability exposure — and where Capital One in 2019 turned on a misconfigured, internet-reachable resource — the pattern catalogued in this report's exposed data stores — subdomain takeover turns on neither a vulnerability nor a misconfiguration in the running system. It turns on a record that points at nothing, and a provider that lets a stranger fill the gap.

That distinction is what makes it dangerous in a way the others are not. The Equifax and Capital One classes are at least theoretically visible to the asset owner: the vulnerable host is in the inventory, the open store is in the cloud account. A dangling CNAME points *away* from the organization, to infrastructure it no longer owns or operates, which is exactly why it falls out of every inventory the organization keeps of itself. It is an exposure of subtraction — the danger is the absence of a resource, not

the presence of one — and conventional asset management is built to find what exists, not to notice what has quietly ceased to.

The post-exploitation use is also distinct. The mass-exploitation events of recent years — the MOVEit file-transfer campaign, the regreSSHion remote-code-execution exposure in OpenSSH that this report finds running on thousands of hosts — are about reaching *into* a victim. Subdomain takeover is about reaching *out* from a victim's identity: using the trusted brand to attack the brand's own customers, partners, and staff. The blast radius is not the compromised host; it is everyone who trusts the domain. We do not attribute or detail any specific takeover incident here — our methodology is passive and detect-only — but the structural logic is identical across every public case of the class: a trusted name, a forgotten record, a free account, and a phishing page that the victim's own DNS vouches for.

What to do about it

The remediation is unusually clean for a problem this consequential, because the fix is the deletion of a record rather than the patching of a system.

1. **Remove the delegation, not just the service.** Make DNS de-provisioning a mandatory, gating step in every decommission runbook. When a Shopify store, GitHub Pages site, Heroku app, or any externally-hosted property is retired, the corresponding CNAME must be deleted in the same change. The dangling record exists because these two actions are owned by different teams; closing that gap closes the exposure class at its source.
2. **Inventory outward-facing delegations continuously.** Enumerate every CNAME in every zone that points to a third-party service and reconcile it against a live-resource check, on a schedule — not once. Certificate-transparency logs and passive DNS make the attacker's enumeration trivial; the defender must run the same enumeration first, and treat any delegation whose target returns an unclaimed-resource signal as a live incident.
3. **Constrain who can create records and where they can point.** Treat the authority to add a CNAME to a corporate zone as a privileged action with an owner and an expiry, and prefer a small set of sanctioned hosting providers over an open field, so that the population to monitor stays bounded.
4. **Reduce the blast radius of a takeover before it happens.** Scope session cookies to the exact host that needs them rather than the parent domain; avoid `Domain=.example.com` for anything sensitive. Tighten Content Security Policy, CORS, and redirect allow-lists so they do not trust `*.example.com`

wholesale. These controls do not prevent a takeover, but they sharply limit what a hijacked subdomain can do to the rest of the estate.

Organizations can establish their own exposure directly. EchelonGraph publishes the aggregate subdomain-takeover findings from this study, and a self-service external surface scan that surfaces an organization's own third-party-delegated subdomains and flags dangling records before an attacker claims them. Because the entire estate of trusted hostnames is the blast radius, the same names belong in the attack-graph view that models how a single trusted foothold connects to identity, session, and downstream systems — so that a forgotten storefront is evaluated as the phishing and session-theft platform it can become, not merely as a stale line in a DNS zone.

CHAPTER 8

The Vulnerability Landscape

The preceding chapters described *exposure* — what an organization has inadvertently left reachable on the public internet. This chapter describes the raw material that adversaries pair with that exposure: the vulnerabilities themselves. A reachable host is only a target once a usable flaw exists in the software it runs. The two halves are inseparable. The KEV-exposure findings in Chapter 4 are precisely the intersection of an exposed surface and a known, weaponized flaw; this chapter steps back to the full corpus from which that intersection is drawn.

The headline problem is not that vulnerabilities exist — they always have — but that they are now disclosed faster than any human triage queue can absorb, and that the severity score most teams still anchor on is a poor predictor of which ones will actually be used against them. EchelonGraph tracks and enriches the complete public CVE record so that this chapter can speak to scale, composition, and — most importantly — the shift in how defensible organizations decide what to fix first.

The corpus, in scale

As of the close of the observation window, EchelonGraph tracked and enriched **340,552** distinct CVEs — effectively the entire published catalogue of software vulnerabilities with an assigned identifier. This is not a sample. It is the population, and its composition matters because it dictates the arithmetic every security team is up against.

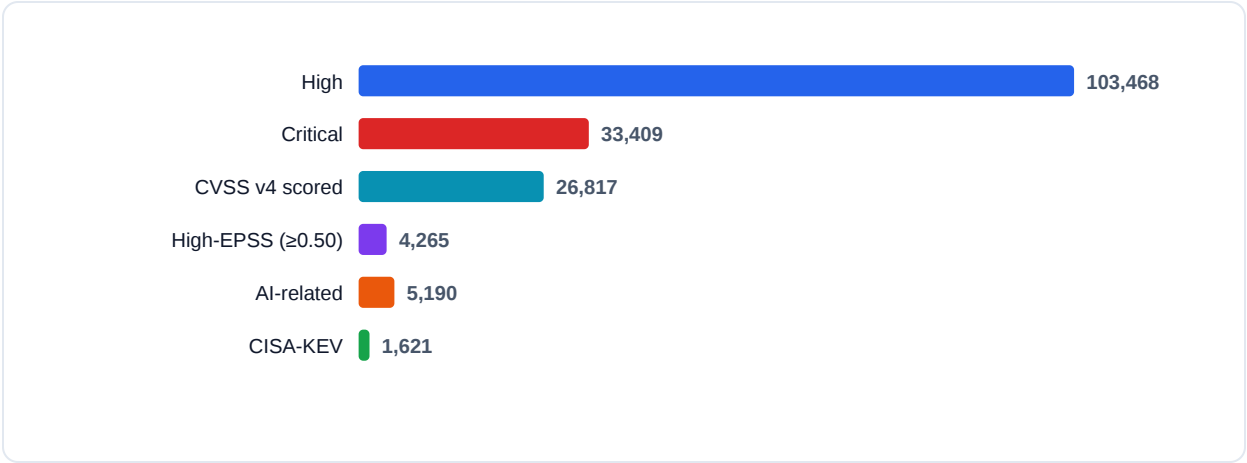
> **Of 340,552 tracked CVEs, 33,409 are rated Critical and 103,468 are rated High — roughly two in five carry a severity that, taken at face value, demands urgent action.**

That figure is the crux of the modern triage crisis. If a Critical or High rating were a reliable instruction to drop everything and patch, the instruction would arrive for 136,877 vulnerabilities — the sum of the two top bands. No organization — not the largest bank, not a hyperscaler — has the engineering capacity to treat 136,877 items as urgent.

Severity alone, applied literally, produces a backlog that is not a prioritization at all. It is noise with a red label.

The tracked CVE corpus by class (n = 340,552)

Class	Count	What it signals
Total CVEs tracked & enriched	340,552	The full published catalogue
Critical severity	33,409	Highest CVSS band
High severity	103,468	Second band; large by volume
CVSS v4 scored	26,817	Newer scoring standard, adoption growing
AI-related	5,190	Flaws in or adjacent to AI/ML systems
High-EPSS (exploitation likely)	4,265	Statistically probable to be exploited
CISA-KEV (known exploited)	1,621	Confirmed used in the wild
Ransomware-linked KEV	326	Tied to known ransomware operations
Newly published, last 30 days	7,624	Net new arrivals in the window



The CVE corpus by class (n = 340,552). The two largest bars — High and Critical severity — dwarf the bars that actually predict exploitation: High-EPSS and CISA-KEV. The mismatch in scale is the triage problem rendered visually.

The signal hides in the small numbers

Read the table again, but invert the instinct. The large numbers — 103,468 High, 33,409 Critical — are the distraction. The numbers that should govern a remediation queue are the small ones at the bottom.

Only **1,621** of the 340,552 tracked CVEs appear on CISA's Known Exploited Vulnerabilities catalogue — the authoritative public record of flaws confirmed to be exploited in the wild. That is less than half of one percent of the corpus. A further **4,265** carry a high Exploit Prediction Scoring System (EPSS) probability, meaning the data-driven forecast says exploitation is statistically likely even where it has not yet been publicly confirmed. And just **326** — a few hundred out of a third of a million — are tied to known ransomware operations, the category most likely to produce a board-level incident.

> **The vulnerabilities confirmed exploited in the wild number 1,621 — under 0.5% of the catalogue. The remaining 99.5% are real, but they are not where adversaries are spending their time.**

This is the single most important reframing in the chapter. The defensible posture is not "patch everything Critical." It is "find the few hundred to few thousand vulnerabilities that are actually being used, confirm whether your exposed surface runs the affected software, and fix those first." The corpus is enormous; the actionable subset is small, knowable, and changes the math from impossible to tractable.

From CVSS-only triage to EPSS, KEV and SSVC

For most of the last decade, vulnerability management ran on a single number: the CVSS base score. It is a useful measure of *intrinsic technical severity* — how bad the flaw is in the abstract, assuming an attacker reaches it. What CVSS was never designed to answer is the question a CISO actually asks: *is this one going to be used against us, and does it matter in our environment?* Three complementary signals have emerged to close that gap, and EchelonGraph enriches every CVE in the corpus with all of them.

The four signals — what each answers, and its limits

Signal	Question it answers	What it does not tell you
CVSS (v3 / v4)	How technically severe is this flaw in the abstract?	Whether anyone is, or will be, exploiting it
EPSS	What is the probability it will be exploited in the near term?	Whether it is severe, or relevant to your assets
CISA-KEV	Is it confirmed exploited in the wild, right now?	How likely future exploitation is for items not yet listed
SSVC	Given exploitation, exposure and impact, what should I *do*?	A raw number — it outputs a decision, not a score

The four are not competitors; they are layers. CVSS establishes whether a flaw is worth caring about at all. EPSS — the Exploit Prediction Scoring System — converts the question of exploitation from a guess into a probability, recalculated daily from observed attacker behaviour, telemetry and exploit availability. CISA-KEV is the ground-truth backstop: a binary, human-curated "this is being exploited today, act now." And SSVC — the Stakeholder-Specific Vulnerability Categorization model — fuses exploitation status, system exposure and mission impact into an explicit decision: *Track*, *Track**, *Attend*, or *Act*. It is the only one of the four that outputs an action rather than a metric, which is why mature programs increasingly anchor on it.

The practical consequence is a reordering. Under CVSS-only triage, a Critical-rated flaw with no known exploitation and a negligible EPSS sits in the queue ahead of a "merely High" flaw that is on the KEV list and tied to ransomware. That ordering is exactly backwards relative to risk. Layering EPSS and KEV on top of CVSS — and ideally resolving to an SSVC decision — inverts it back to something a defender can rationally execute. EchelonGraph's per-CVE enrichment carries CVSS v3 and v4, EPSS, CISA-KEV status with its own tiering, an SSVC decision, GHSA cross-references, ransomware and AI-related flags, and a composite EchelonGraph score, precisely so a team does not have to assemble these signals by hand for each item.

Every CVE in the corpus is enriched with multiple decision signals — CVSS v3/v4, EPSS, CISA-KEV with EchelonGraph tiering, SSVC, GHSA cross-references, and ransomware / AI-related flags — not a single severity number.

Where exposure and the corpus meet

This chapter and Chapter 4 are two readings of the same reality. The corpus described here is the universe of what *could* be exploited; the KEV-exposure findings are what is exploitable *and reachable today* across the observed surface. The intersection is sobering: **21,299** hosts were observed exposing software affected by CISA-KEV actively-exploited vulnerabilities, and **3,475** host:CVE pairs involved ransomware-linked flaws *(method superseded 2026-09-29)*. Those are not abstract corpus statistics — they are live, internet-facing instances where the small, dangerous subset of the catalogue has landed on a real, reachable machine.

It is worth stating plainly what that intersection has historically cost. The 2017 Equifax breach turned on a single known, patchable web-framework flaw left unremediated on an internet-facing system — a textbook case of a high-EPSS, ultimately KEV-class vulnerability meeting an exposed surface. The 2023 MOVEit Transfer campaign showed how one flaw in a widely deployed file-transfer product, once weaponized, cascades across thousands of organizations simultaneously the moment it moves from "disclosed" to "exploited in the wild." And the 2024 regreSSHion flaw in OpenSSH — observed in this very dataset across thousands of exposed hosts — is a reminder that the most ubiquitous, most trusted software is also the highest-value target precisely because it is everywhere. In each case the defensive failure was not ignorance of the flaw; it was an inability to distinguish the one that mattered from the tens of thousands that did not, and to act on it before the window closed.

Two pressures that bend the curve: velocity and AI

Two structural shifts make the triage problem worse over time, not better.

The first is **velocity**. In the roughly five-week observation window *(window corrected 2026-09-29)*, **7,624** new CVEs were published — net new arrivals on top of an already-saturated backlog. Disclosure now runs faster than human review can keep pace, which means the gap between "a flaw exists" and "a team has assessed it" widens continuously unless triage is automated against exploitation signals rather than handled item by item. The arithmetic only closes if the filter is EPSS- and KEV-driven from the outset.

The second is **AI**. The corpus already contains **5,190** AI-related vulnerabilities — flaws in or adjacent to AI and machine-learning systems, model-serving stacks, inference frameworks and the tooling around them. This is a vulnerability class that barely existed a few years ago and is now a measurable, growing slice of the catalogue. It maps

directly onto the AI attack surface documented in Chapter 3: as organizations stand up model proxies, vector databases and notebook environments, they introduce a fresh and rapidly evolving inventory of flaws that traditional vulnerability programs were never built to track. An AI-related CVE on an exposed, unauthenticated inference endpoint is the next decade's equivalent of an unpatched web framework on a public server.

Two pressures bending the vulnerability curve

Pressure	Observed figure	Why it compounds
Disclosure velocity	7,624 new CVEs in ~5 weeks *(window corrected 2026-09-29)*	Arrivals outrun human triage; backlog grows unless filtered by exploitation signal
AI-class vulnerabilities	5,190 AI-related in corpus	New surface (Chapter 3), immature tooling, fast-moving flaw inventory
Scoring transition	26,817 CVSS v4 scored	Dual v3/v4 world; programs must read both during the multi-year migration

The scoring transition itself is a quieter third pressure. With **26,817** CVEs now carrying CVSS v4 scores alongside the long-established v3, vulnerability programs are operating in a dual-standard world for the foreseeable future. A team that reads only v3, or only v4, will systematically misrank a growing share of the catalogue. EchelonGraph carries both for every applicable CVE so that a comparison is never apples-to-oranges.

What this means for the reader

The vulnerability landscape is not best understood as a number that goes up every year. It is best understood as a separation problem: an enormous, ever-growing corpus in which the genuinely dangerous subset is small, identifiable, and obscured by severity inflation. The organizations that fare well are not the ones that patch the most; they are the ones that patch the *right* few hundred items before the exploitation window opens.

- **Stop triaging on CVSS alone.** Severity tells you how bad a flaw is in the abstract; it does not tell you whether it will be used against you. Of the corpus, only 1,621 CVEs are confirmed exploited and 4,265 are statistically likely to be — that is the universe to anchor on, not the 136,877 rated High-or-Critical.
- **Layer EPSS, KEV and SSVC over CVSS.** Probability of exploitation (EPSS), confirmation of exploitation (KEV) and an explicit do-this decision (SSVC) turn an impossible backlog into a ranked, executable queue.

- **Intersect the corpus with your own exposure.** A KEV-class flaw matters to you only if you run the affected software on a reachable host. Pairing the catalogue against your external surface — as in Chapter 4's 21,299 exposed KEV hosts — is how a global statistic becomes a personal worklist. The free Surface Scanner is the starting point for that intersection.
- **Plan for velocity and for AI.** 7,624 new CVEs in five weeks *(window corrected 2026-09-29)* is the steady state, not a spike; 5,190 AI-related flaws are an early reading of a curve that is still bending upward. Triage that depends on human review of every item will fall further behind every month it runs unchanged.

The corpus is the map of everything that could go wrong. The enrichment signals — EPSS, KEV, SSVC, ransomware and AI flags, the composite EchelonGraph score available on every entry in the CVE Pulse feed — are the legend that tells a defender which roads adversaries are actually driving down. Read together, they convert a third of a million vulnerabilities from a source of paralysis into a short, ordered list of decisions.

CHAPTER 9

Trends & Trajectory

This chapter is the one place in the report where we ask not "what is exposed today?" but "which way is it moving?" — and it is also the chapter where intellectual honesty matters most. The figures elsewhere in this report are point-in-time counts of what EchelonGraph observed between 29 May and 28 June 2026 *(window corrected 2026-09-29)*. A trajectory needs a second point, and a single inaugural window roughly five weeks long cannot deliver a defensible multi-year trend line. We will not pretend otherwise. What this baseline *can* do is establish the floor — the numbers next year's edition will be measured against — and surface the directional signals that are already legible inside the window: the rate at which new exposure appears, the categories that are growing fastest, and the structural forces that make the next twelve months easy to reason about even without a longer history.

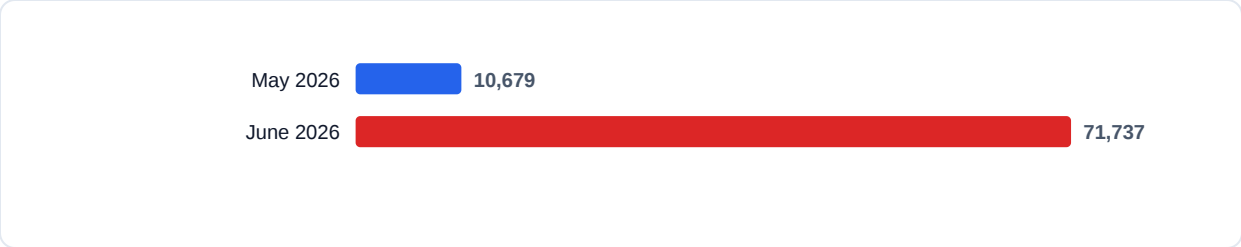
Read this chapter, therefore, as a baseline with annotations, not as a forecast. Where we point at a direction, we say why we believe the direction is real and not an artifact of our own measurement. Where a number looks like growth but is actually our scanners reaching further, we say that too, plainly, because the alternative — letting a coverage increase masquerade as a threat increase — is exactly the kind of dishonesty that has hollowed out the credibility of vendor "threat reports" for a decade.

The headline number is real, but it is half coverage and half signal

The single most arresting movement in the window is in AI infrastructure. EchelonGraph's discovery of AI-related services rose from 10,679 in May to 71,737 in June — a 6.7-fold month-over-month jump. Taken at face value that reads as an explosion of new AI deployment onto the public internet. Taken honestly, it is mostly the sound of our own instrument warming up.

> **AI-infrastructure discovery rose from 10,679 hosts in May to 71,737 in June — a 6.7× jump — but the bulk of that step is scanner ramp, not five weeks of real-world AI sprawl.**

The Shadow AI Radar expanded its discovery surface materially during the window: more Certificate Transparency search patterns, more banner-fingerprint dorks, and more read-only liveness probes came online across the period. When a scanner's reach grows inside the same window you are measuring, the count of things it finds grows with it — and most of what appeared in June was already sitting on the internet long before our crawler got to it. The correct interpretation of the May-to-June step is therefore: **this is what fuller coverage of an existing footprint looks like, not what one month of greenfield AI adoption looks like.** Treating it as the latter would overstate the velocity of the threat by a wide and unknowable margin.



AI-infrastructure discovery by month within the observation window. The May→June step is dominated by expanding scanner coverage of an already-existing footprint, not by five weeks of net-new AI deployment.

AI-infrastructure discovery by month (observation window)

Month	Hosts discovered	Share of window total	Primary driver
May 2026	10,679	13%	Initial radar coverage
June 2026	71,737	87%	Expanded radar coverage of existing footprint
Window total	82,416	100%	Discovered AI footprint (137 countries)

This is also why we anchor every AI figure to the word *discovered* rather than *exposed*. Of the 82,416 AI services found, 36,991 were auth observed or not determined *(corrected 2026-09-27)*, 42,593 resolved in DNS without a reachable service behind them, 782 were unreachable, and only 2,050 were confirmed active and open. The growth in the discovery count does not move the confirmed-open count proportionally — finding more of the internet's AI footprint mostly means finding more AI footprint that is *not confirmed open (corrected 2026-09-27)*. The trajectory worth watching across editions is therefore not the discovery total at all; it is the confirmed-active subset, where the active population is dominated by open LLM proxies (1,743), with live vector

databases (171) and exposed Model-Context-Protocol servers (2) *(MCP count withdrawn 2026-09-29)* as the smaller but more alarming tail. Next year, the honest growth question is: did the 2,050 move, and in which direction?

Velocity: what "new per day" actually means here

A CISO reading a threat report wants a velocity number — findings per day — to size the rate at which their own attack surface drifts. We can give one, but only with the caveat attached: a velocity computed over a five-week inaugural window, during which the measuring apparatus itself was changing, is a rough order-of-magnitude figure, not a precision instrument. With that stated plainly, the corpus-level signal is the most stable one we have, because the CVE feed does not depend on our scanner reach the way the radars do.

> **7,624 new CVEs were published in the last 30 days — roughly 250 a day — against a tracked corpus of 340,552. New vulnerability supply is the one velocity number in this report that is not an artifact of our own coverage.**

The vulnerability corpus grew by 7,624 CVEs in the trailing 30 days, against a total tracked corpus of 340,552 — on the order of 250 newly-published vulnerabilities every day, a touch over two percent of the entire historical corpus arriving in a single month. This figure is comparatively trustworthy as a trajectory signal precisely because it is upstream of EchelonGraph: CVEs are assigned by CNAs and flow through the public record regardless of how far our crawler reaches. The implication for defenders is structural and unglamorous — the rate at which the world manufactures new attacker opportunity is steady, high, and indifferent to any one organization's patch cadence. The 7,624-per-month inflow is the headwind every remediation program runs into, and it does not abate between report editions.

The exposure radars, by contrast, produce velocity figures we deliberately decline to annualize from this window. The KEV-exposure radar observed 36,416 hosts running software with known CVEs and 165,717 distinct host-to-CVE pairings; the exposed-database radar found 6,607 open, unauthenticated stores; public-Git credential leakage surfaced 619 secrets across 478 repositories. Each of these is a real count of real exposure observed in the window *(window corrected 2026-09-29)*. None should be read as "X per week, therefore 52X per year" — the radars were still expanding their reach,

and a linear extrapolation from a ramping baseline is exactly the error we just warned against in the AI numbers. We report them here as the floor each radar will be measured against, not as a rate.

Where the trajectory points — structural forces, not curve-fitting

Because the window is too short to fit curves to, the defensible way to talk about direction is to name the structural forces already visible in the data and reason about where they push. Three are strong enough to call.

First: AI infrastructure is becoming a first-class attack surface, and it is starting from a position of weak defaults. Even setting the inflated discovery growth aside, the confirmed-active picture is the tell. The open population is led by LLM proxies — the unguarded front doors to model inference, billing, and frequently the API keys behind them — and includes a tail of live vector databases, the substrate of retrieval-augmented systems and therefore a place organizations are actively pooling proprietary and customer data. This category did not meaningfully exist in any prior "state of the internet" baseline because the technology did not exist at scale. EchelonGraph's separate CVE tracking already enumerates 5,190 AI-related vulnerabilities in the corpus, so the disclosure pipeline for AI software is now a standing feature of the landscape, not a curiosity. The structural read is unambiguous: AI exposure is the category most likely to grow in genuine substance — not just coverage — between this edition and the next, and it is doing so while the security maturity of the average deployment lags the technology by years. The pattern to watch is the familiar one — an open, unauthenticated data store attached to a fast-moving build-out, reachable to anyone who looks rather than to anyone who breaks in — now recurring on AI infrastructure that has been stood up faster than it has been secured.

Second: the most-exposed software in the KEV data is old, load-bearing, and slow to change — which makes its trajectory predictable and grim. The KEV radar's top of the list is not exotic. It is OpenSSH (12,889 hosts), Apache Tomcat (5,219), Apache httpd (3,491), MongoDB (3,230) and VMware ESXi (2,816) — the structural plumbing of the internet. The single most-observed exposure is CVE-2023-48795, the Terrapin weakness in OpenSSH, on 9,942 hosts, with the regreSSHion flaw (CVE-2024-6387) on a further 5,713. Plumbing of this kind turns over on a multi-year cycle, not a monthly one. That property cuts two ways for the trajectory: the population is unlikely to spike, but it is equally unlikely to drain quickly, and 21,299 of the CVE-exposed hosts already carry vulnerabilities on CISA's actively-exploited (KEV) catalogue — including 3,475 tied to known ransomware operations. The honest

forward-look is that this is the slow, sticky tail of internet exposure: it will look much the same in the next edition unless an external shock — a wormable exploit, a regulatory mandate, a mass cloud-image refresh — forces it to move. A single edge-software vulnerability in a widely-deployed component, exploited at scale, is the classic shock here: it can convert a long tail of "patch eventually" into an industry-wide incident within days, which is precisely why a sticky population that looks stable can move suddenly.

Third: cheap, structural exposure classes are not self-correcting and should be expected to persist. Open databases (6,607, led by 4,243 Redis and 1,756 Memcached instances), web-exposed secrets (750 hosts, with roughly 2,600 AWS credentials in reachable `.env` files), and dangling-CNAME subdomain takeover (7,952 observed, 134 confirmed hijackable) share a common cause: they are configuration and lifecycle failures, not software flaws awaiting a patch. Nothing in the ecosystem applies steady downward pressure on them the way a vendor patch eventually does to a CVE. A misconfigured Redis instance stays open until a human notices; a leaked AWS key in a public repository stays valid until it is rotated; a forgotten CNAME points at a claimable host until someone reclaims or deletes it. It is exactly these unglamorous categories — unpatched internet-facing components and misconfigured cloud infrastructure, not exotic zero-days — that most often turn into headline losses, because they sit open long enough for someone to find them. Absent a deliberate intervention, the trajectory for all three is flat-to-rising, and they are the cheapest classes for any single organization to remove from its own footprint before the next edition lands.

What next year's edition will measure against

The discipline of an inaugural baseline is that it commits to numbers it can be held to. For the avoidance of any "moving the goalposts" later, the following are the floors the 2027 edition will be read against — and the specific signals that will tell us whether the trajectory we have reasoned about above is bending toward improvement or decay.

Inaugural-baseline anchors and the trajectory signal each one establishes

Dimension	2026 baseline	What a year-over-year move would tell us
AI services confirmed active & open	2,050	The honest AI trajectory — independent of scanner reach, since it counts only confirmed-open services
Hosts on actively-exploited (KEV) vulns	21,299	Whether the sticky, load-bearing software tail is draining or holding
Open, unauthenticated databases	6,607	Whether self-inflicted, non-self-correcting exposure is being addressed at all
Web-exposed secrets (hosts)	750	Credential-hygiene drift across the public internet
Secrets leaked in public Git repos	619	Whether developer-side secret discipline is improving upstream of deployment
Confirmed-hijackable subdomains	134	DNS lifecycle hygiene; the rate at which dangling records get reclaimed
New CVEs published, trailing 30 days	7,624	The exogenous supply of new attacker opportunity — the field's headwind

How to read these trajectories without fooling yourself. When the 2027 edition arrives, weight the discovery-count growth lightly and the confirmed-active and corpus-level numbers heavily. The radars' reach will have grown again, so a bigger discovery total is the expected null result, not news. The numbers that carry real signal are the ones that do not depend on how far our crawler reached: the confirmed-open subsets, the KEV-exposed host count, and the upstream CVE inflow. Those are where a genuine improvement — or a genuine deterioration — in the state of internet exposure will actually show up.

The honest summary of the trajectory is this. The internet's exposure is moving in two speeds at once. The legacy tail — exposed databases, leaked credentials, unpatched load-bearing software, dangling DNS — is large, sticky, and self-inflicted, and it will look broadly similar next year unless organizations choose to act on it; it is also, for exactly that reason, the most addressable. The AI tail is small in confirmed-open terms today

but is the one category where genuine, substantive growth — not measurement artifact — is the safe bet, arriving on infrastructure whose security maturity has not caught up with its adoption. The single number every organization can change before the next edition is its own: a five-minute pass through the free Surface Scanner turns this internet-wide trajectory into a specific, fixable list for one perimeter — and removes that perimeter from whatever the 2027 baseline records.

Risk Analysis: Toxic Combinations

The preceding chapters each measured one fault line in isolation: an open database here, a leaked key there, a host running software with a known-exploited flaw. Read that way, every finding has a tidy, bounded severity. Read the way an attacker reads it — as a graph, not a list — the severities do not add. They multiply. A single leaked credential is an incident; a leaked credential that unlocks an over-permissioned role, which reaches an unauthenticated data store, which sits one network hop from a host on an actively-exploited vulnerability, is a breach. This chapter is about the second reading. It is the most important chapter in the report, because it is the one that maps most directly onto how organizations actually get compromised.

We use the term *toxic combination* in its now-standard sense: two or more individually-tolerable conditions that, when present together on a reachable path, produce a risk far greater than the sum of their parts. The discipline that studies these paths is attack-path analysis, and the quantity it produces is *blast radius* — the set of assets, data, and identities an adversary can reach once a single foothold is established. EchelonGraph's Attack Graph exists precisely to compute that quantity. What follows is what our passive, external vantage point can already see about the raw materials for these chains, drawn entirely from the same observation window as the rest of this report.

The "so what": severity is not additive, it is multiplicative

Most vulnerability programs are organized around a queue of findings sorted by CVSS score. That model has a structural blind spot: it scores nodes, not paths. A medium-severity misconfiguration that bridges a low-severity exposure to a critical asset can be the single most dangerous thing in an estate, and a node-by-node queue will never surface it, because no individual node in the chain is rated "critical." Attackers do not respect that queue. They look for the shortest reachable path from anything they can touch to anything worth taking, and they are indifferent to how each link was scored in isolation.

Across this window we observed 6,519 organizations running internet-facing software with at least one known CVE and 1,723 organizations exposing at least one open, unauthenticated database — two populations that, where they overlap, describe an estate where the foothold and the prize are already both visible from the public internet *(window corrected 2026-09-29)*.

The remainder of this chapter does three things. First, it inventories the raw ingredients of a breach chain as we measured them externally. Second, it walks the canonical chains those ingredients compose into, each anchored to a real-world incident that followed exactly that shape. Third, it explains why the graph — and not the list — is the correct unit of analysis, and how an organization can compute its own blast radius rather than infer it from this report.

The ingredients, as observed externally

Every breach chain is assembled from a small set of recurring primitives. We are able to observe each of these primitives passively, from public data, without ever authenticating or reading any contents *(method superseded 2026-09-29)*. The table below maps each primitive to what we measured, and to the role it plays in a chain.

Chain primitive	What we observed (this window) *(window corrected 2026-09-29)*	Role in an attack path
Initial access — known-exploited vuln	21,299 hosts exposed to CISA-KEV actively-exploited vulnerabilities (50 distinct KEV CVEs), out of 36,416 hosts running software with known CVEs across 165,717 host:CVE pairs	The front door. A vulnerability already weaponized in the wild is the cheapest, most reliable foothold an external attacker has.
Initial access — leaked credential	619 secrets across 478 public repositories; an additional ~2,600 AWS credentials and 2,571 exposed `.env` files across 750 web hosts	An authenticated front door. A valid key skips exploitation entirely and often grants identity inside the target's own cloud.
Privilege / lateral reach	134 confirmed-vulnerable subdomain takeovers (of 7,952 dangling delegations observed)	A trust hijack. A hijacked subdomain inherits the parent domain's reputation — usable for credential phishing, cookie theft, or bypassing same-site controls.
The prize — unauthenticated data	6,607 open, unauthenticated databases across 1,723 organizations and 120 countries	The objective. A store reachable without authentication is exfiltration with no exploitation step required.
The prize — AI / model assets	2,050 confirmed-active, open AI services — including 1,743 open LLM proxies and 171 live vector databases — within a discovered footprint of 82,416	A new objective class. Open model proxies and vector stores expose prompts, embeddings, training data, and the keys behind them.

Two clarifications matter here, and we state them plainly because misreading either would inflate the risk. First, the 82,416-figure is our *discovered* AI footprint, not an exposed one: the large majority resolved in DNS only or were not confirmed open (auth observed or not determined) *(corrected 2026-09-27)*, and only 2,050 were confirmed reachable and open. An authenticated endpoint is not a finding. Second, for the open databases we detect the *open store* — that it answers without credentials — and we do not read what is inside *(data-store check corrected 2026-09-29)*. Our content

classification is deliberately conservative; across the entire population we flagged only 3 stores with PII indicators and 1 with PCI indicators. We make no claim about record counts. The danger of an open store is not a number we assert; it is the fact that an attacker, unlike us, would read it.

How the ingredients combine: the canonical chains

The primitives above are not dangerous because they exist. They are dangerous because they connect. Below are the four chains we consider most consequential, ordered roughly by how often the necessary ingredients co-occurred in our data. Each is presented as a sequence — entry, traversal, objective — and anchored to a publicly-documented breach that followed the same shape. We describe those incidents only at the level of their published, structural facts; we do not embellish them.

Anatomy of a breach chain: how individually-tolerable exposures compose into a single attack path

Chain 1 — Leaked key → cloud identity → open data store

So what: this is the chain that needs no exploit at all. When a valid credential is the entry point, there is no vulnerability to patch and no exploit to detect in flight — the attacker simply authenticates as you. It is the cheapest breach in the catalogue, and our data shows the raw materials are abundant.

1. **Entry.** An access key is committed to a public repository or left in a web-served `.env` file. We observed ~2,600 AWS credentials in web exposures and a further 63 AWS keys among the 619 secrets found across 478 public repositories — alongside 228 Google and 53 Telegram credentials in the same Git population.
2. **Traversal.** The key authenticates to the victim's own cloud. If the associated identity is over-permissioned — the single most common cloud misconfiguration — it can enumerate buckets, read secrets managers, and assume further roles.
3. **Objective.** The identity reaches a data store. In the worst case that store is one of the 6,607 already open to the entire internet, and the credential was never even required.

Incident parallel — Capital One, 2019. The publicly-reported breach followed this exact topology: a server-side request forgery weakness was used to obtain cloud credentials, the over-scoped role those credentials carried was used to enumerate storage, and

customer data held in cloud object storage was exfiltrated. The lesson the industry took, and the one our data restates, is that the credential and the permission boundary — not only the perimeter — are load-bearing. A leaked key is only as dangerous as the identity behind it; in practice, that identity is too often unbounded.

Chain 2 — Known-exploited vuln → foothold → lateral reach → data

So what: this is the chain the entire KEV program exists to interrupt, and it remains the best-instrumented path because the exploitation step is, by definition, already happening in the wild. When the front door is a vulnerability adversaries are *actively* exploiting, the only variable left is how long it stays open.

1. **Entry.** An internet-facing host runs software with an actively-exploited flaw. We observed 21,299 such hosts on CISA-KEV vulnerabilities, with the OpenSSH family — Terrapin (CVE-2023-48795, 9,942 hosts) and regreSSHion (CVE-2024-6387, 5,713 hosts) — and the Apache Tomcat and VMware ESXi populations among the most exposed *(method superseded 2026-09-29)*.
2. **Foothold & lateral reach.** Exploitation yields code execution or a session on the host. From there an attacker pivots — and a hijacked subdomain (134 confirmed vulnerable in our data) can supply the trusted-origin staging or phishing surface that widens the pivot.
3. **Objective.** The pivot terminates at data or at a deployment system. Of the KEV-exposed population, 3,475 hosts ran software tied to vulnerabilities with documented ransomware association — the chains most likely to end not in quiet exfiltration but in encryption.

Incident parallels — Equifax, 2017; MOVEit, 2023. Both began with a single exploited, internet-facing software flaw and ended in mass data loss, and both illustrate the same uncomfortable truth our numbers carry: the window between a vulnerability becoming known-exploited and an organization closing it is measured in the same units as the breach. When 21,299 hosts sit on vulnerabilities attackers are already using, the chain's first link is not hypothetical — it is the present tense.

Chain 3 — Exposed AI plane → harvested key → downstream blast radius

So what: the AI build-out has created a new entry class that most asset inventories do not yet enumerate, and it is uniquely dangerous because an exposed AI surface

frequently holds the keys to everything behind it. The proxy is not the prize; the credential it leaks is.

1. **Entry.** An LLM proxy or AI-workflow endpoint is deployed without authentication. Of the 2,050 confirmed-active, open AI services, 1,743 were open LLM proxies — gateways that commonly sit in front of, and are configured with, the API keys for commercial model providers and internal services.
2. **Harvest.** An open proxy can leak its upstream provider key, its system prompts, and the contents of any connected retrieval store. Where that store is one of the 171 live vector databases we observed, the embeddings — and the source documents behind them — are reachable directly.
3. **Downstream.** A harvested key is a credential, and a credential re-enters Chain 1. The blast radius of an exposed AI plane is therefore not the model; it is every system whose keys the model could see.

Incident parallel — the DeepSeek open database, 2025. The publicly-reported finding was an unauthenticated database, associated with an AI service, reachable from the open internet and exposing operational data and secrets. Structurally it is the union of Chain 1 and Chain 3: an AI deployment whose data plane was simply left open. It is the clearest public demonstration that the AI surface is not a separate, contained risk category — it is the same data-and-credential exposure as everything else, accelerating faster than inventories can track it. For context, our discovered AI footprint grew from 10,679 to 71,737 month-over-month within this window; the scanner ramp accounts for much of that, but the direction of travel for AI-surface area is not in doubt.

Chain 4 — Subdomain takeover → trusted-origin abuse → credential capture

So what: this chain weaponizes trust rather than software. A hijacked subdomain serves attacker content from the victim's own name, which defeats the human and machine heuristics that would otherwise flag the attack — users, cookies, and CORS policies all extend trust to it by default.

1. **Entry.** A DNS record points at a de-provisioned third-party service, and an attacker re-registers that service to claim the dangling name. We observed 134 confirmed-vulnerable takeovers — predominantly against Shopify, GitHub Pages, and Heroku targets *(these platforms are named from the 7,952*

observed; the report does not say which platforms the 134 confirmed are on; corrected 2026-09-30) — within 7,952 dangling delegations.

2. **Abuse.** Content served from the legitimate subdomain inherits the parent domain's reputation and, depending on cookie scoping and same-site policy, its session context.
3. **Capture.** The trusted origin is used to phish credentials or capture tokens — which, again, re-enter Chain 1 as valid keys, closing the loop.

Chain 4 is the connective tissue of the catalogue. It rarely is the whole breach, but it lowers the cost of every other chain by supplying a trusted launch point. This is exactly why path-based analysis matters: a single dangling CNAME, trivial in isolation, can be the cheapest edge in an attacker's graph.

Why the graph, not the list, is the unit of analysis

So what: if severity multiplies along paths, then the only honest way to measure risk is to measure the paths. A findings list — however well-sorted — cannot tell you that a medium-rated misconfiguration is the bridge between a low-rated leaked key and a critical data store. A graph can, because the bridge is an edge, and the danger is the route.

The recurring structural lesson across all four chains is that the link the defender under-weighted is the link the attacker used. In Chain 1 it was the permission boundary on an identity, not the leak itself. In Chain 2 it was the lateral path off the foothold, not the CVE score. In Chain 3 it was the key the proxy could see, not the proxy. In Chain 4 it was inherited trust, not any single vulnerable component. None of these links is the highest-severity node in its chain, and a node-ranked queue would deprioritize every one of them.

This is the gap EchelonGraph's Attack Graph is built to close. Rather than ranking findings, it models assets, identities, and exposures as nodes and the reachability between them as edges, then computes blast radius: given a foothold, what is reachable, and by what route. The chains in this chapter are the externally-visible silhouettes of those routes; the internal graph is where they become measurable, prioritizable, and — critically — interruptible at the single edge that severs the most paths.

Chain	Severs the most paths by fixing…	Why this edge, not the loudest node
1 — Leaked key → identity → data	The identity's permission boundary (least privilege)	Bounding the role neutralizes every leaked key that maps to it, present and future.
2 — KEV → foothold → data	The actively-exploited entry vuln, then segmentation of the foothold	Patching the front door removes the cheapest foothold; segmentation caps the blast radius if one is found.
3 — AI plane → key → downstream	Authentication on the AI surface and scoping of the keys it holds	An authenticated proxy stops being an entry; a scoped key limits what its compromise can reach.
4 — Takeover → trust → credentials	The dangling DNS delegation	Removing the record severs the trusted-origin edge that cheapens every other chain.

What this means for you

The organizations most exposed in this analysis are not those with the most findings — they are those whose findings *connect*. An estate with a hundred isolated medium-severity issues and no path between them is, counter-intuitively, safer than an estate with three findings that form a single chain from the public internet to a customer database. Findings counts measure work; reachable paths measure risk.

Three practical implications follow directly from the data in this chapter:

- **Hunt for chains, not just findings.** Cross-reference your own results across categories the way an attacker would: every leaked key against every over-permissioned role; every KEV-exposed host against its lateral reach; every open store against what could authenticate to it. The ingredients in our table are common precisely because they are rarely correlated in-house.
- **Prioritize the bridging edge.** When a path is identified, the highest-leverage fix is the edge that severs the most paths — which is frequently a medium-severity item a node-ranked queue would defer. The table above names the edge for each canonical chain.
- **Treat the AI surface as part of the same graph.** The fastest-growing entry class in our data is also the one most likely to hold downstream keys. It

belongs in the same blast-radius model as your databases and identities, not in a separate AI-security silo.

You do not have to infer your own chains from this report. The same external vantage point that produced these figures is available to you directly: the free Surface Scanner shows what the internet can already see of your estate, and the Attack Graph turns that surface into the paths an attacker would actually walk. The findings in the preceding chapters are the ingredients. This chapter is about the recipe — and the recipe is what breaches you.

CHAPTER 11

Mitigation & Remediation

The preceding chapters catalogued an exposure surface that is, in almost every instance, the product of a known and well-understood failure mode: a default left unchanged, a key committed to a public place, a patch deferred, a DNS record orphaned. None of the findings in this report required a novel exploit chain to reach. That is the central, uncomfortable finding of the 2026 baseline — and it is also the source of its optimism. Surfaces built from known mistakes can be closed with known fixes. The question is not *whether* the remediation is understood, but *in what order* a defender with finite hours should apply it.

This chapter answers that question. For each finding type observed between **29 May and 28 June 2026**, we give the concrete corrective action, who it most affects, and the real-world incident that demonstrates the cost of leaving it undone. We then rank every action by a single, deliberately unsentimental metric: **blast-radius reduction per engineering hour**. The intent is to let a CISO with one available sprint, or one available afternoon, spend it where the marginal risk retired is highest.

The prioritization principle: blast radius per hour, not severity per finding

Conventional vulnerability management ranks work by severity score. That ordering is necessary but insufficient, because it ignores two variables that dominate real-world outcomes: how much an attacker gains from the first successful step (blast radius), and how long the fix actually takes to apply (effort). A CRITICAL CVE on an internal host behind three network controls retires less risk per hour than an unauthenticated database that hands an attacker the data plane in a single unauthenticated request.

We therefore score each remediation along three axes — **blast radius** (what an attacker controls after the first success), **exploitation likelihood** (is it actively exploited, trivially discoverable, or already weaponized), and **effort to remediate** (minutes, hours, or a program) — and order the work by the ratio of risk retired to hours spent. The findings in this report are unusually amenable to this treatment because their blast radii

are extreme and their fixes are, for the most part, configuration changes rather than code rewrites.

> **Every high-priority finding class in this report can be remediated with configuration, key rotation, or patching — not custom engineering. The constraint is sequencing and discipline, not capability.**

Remediation actions ranked by estimated blast-radius reduction per engineering hour (relative; derived from observed finding counts and typical fix effort)

Priority 1 — Exposed data stores: put authentication in front of the data plane

So what: An open, unauthenticated database is not a vulnerability that an attacker must exploit — it is the objective, already achieved, waiting to be found. There is no privilege escalation, no lateral movement, no payload. The first connection is the breach. This is why it sits at the top of the priority order despite often carrying a lower nominal CVSS than a remote-code-execution CVE: the effort to reach the data is zero, and the blast radius is the entire store.

We observed **6,607 open, unauthenticated databases** across **1,723 organizations** and **120 countries** *(method superseded 2026-09-29)*. The distribution is dominated by in-memory and caching engines that ship with no authentication by default and are frequently bound to all interfaces during setup and never re-bound: **Redis (4,243 hosts)** and **Memcached (1,756 hosts)** together account for roughly nine in ten of the exposed stores. We classify this finding conservatively. We detect the open store; we do not read its contents *(data-store check corrected 2026-09-29)*, and we make no claim about the volume of records behind it. Within the observed set our conservative classification identified **3 stores carrying PII indicators and 1 carrying PCI indicators** — and we report only what the surface metadata supports.

Open unauthenticated data stores by engine (hosts observed)

Engine	Hosts	Default auth posture	Concrete fix
Redis	4,243	No auth; binds 0.0.0.0 if unconfigured	Enable `requirepass` / ACLs; bind to loopback or private subnet; enable TLS; enable protected-mode
Memcached	1,756	No auth; UDP enabled historically	Bind to localhost; disable UDP; enable SASL; place behind a private network boundary
Kibana	464	UI fronting Elasticsearch, often unauthenticated	Enable security realm / SSO; never expose the UI to the public internet; gateway-authenticate
Cassandra	75	`AllowAllAuthenticator` by default	Switch to `PasswordAuthenticator` + authorizer; restrict listen address; enable client encryption
MongoDB	37	Auth off unless explicitly enabled	Enable authorization; bind to private interface; enforce SCRAM; require TLS
InfluxDB	16	Auth optional	Enable auth; scope tokens; restrict bind address
CouchDB	16	Historically shipped in "admin party" mode	Create admin credentials; disable anonymous writes; restrict listen interface

The fix, concretely:

1. **Authenticate, then bind.** Enable the engine's native authentication (Redis `requirepass` and ACLs, Mongo `authorization: enabled`, Cassandra `PasswordAuthenticator`) *and* bind the listener to loopback or a private subnet. Do both — authentication without a network boundary still exposes the auth handshake and version banner; a network boundary without authentication fails the moment that boundary is misconfigured. Defense in depth here is two configuration lines.

2. **Default-deny at the perimeter.** The recurring root cause is a database port reachable from the public internet at all. Security-group and firewall policy should deny ingress to data-store ports (6379, 11211, 9042, 27017, 5601, et al.) by default and allow only named application sources.
3. **Encrypt in transit.** Enable TLS so that even an authorized but observed connection does not leak credentials or payloads.
4. **Detect drift.** The store that is private today is the store that is public after the next infrastructure change. Continuous external attack-surface monitoring (see our exposed-databases radar and the surface self-check) closes the gap between "configured correctly" and "still configured correctly."

Who it hits: Disproportionately, fast-moving teams that stood up a cache or search cluster for a single feature and never revisited its network posture — and the geographies where this report found the largest concentrations: the United States (1,254), China (1,092), Germany (772), France (307), Singapore (248) and India (221).

Incident parallel: The pattern is the one that exposed a major AI provider's ClickHouse instance in early 2025 — an internet-reachable database, no authentication required, operational and chat-history data sitting behind a single open port. The remediation in that case was the remediation here: bind the listener, require credentials. The lesson is that the most consequential data exposures of the decade have repeatedly been not exploits but open doors.

Priority 2 — Leaked & exposed credentials: assume compromise, rotate at the source of truth

So what: A leaked credential collapses the attacker's cost of entry to nothing and, unlike a vulnerable service, it travels — copied into forks, caches, mirrors, and third-party indexes the moment it is published. You cannot "un-leak" a key. The only sound response is to treat any exposed secret as already compromised and to invalidate it at the issuing authority. Deleting the commit or the file does not remediate the credential; rotation does.

We observed two distinct leakage surfaces. On the **web**, **750 hosts** exposed secret-bearing artifacts — overwhelmingly **.env files (2,571 exposure rows)**, alongside exposed `.git` configuration and directories, credential files, database dumps and private keys. The secrets recovered from these artifacts were dominated by cloud keys: **1,319 AWS secret access keys and 1,284 AWS access key IDs — roughly 2,600 AWS credentials in total**. Separately, in **public Git repositories**, we observed **619**

secrets across 478 repositories, spanning generic high-entropy secrets (249), Google credentials (228), AWS (63), and messaging-platform tokens for Telegram (53) and Discord (13) *(partial: these five sum to 606 of the 619; corrected 2026-09-30)*.

Credential exposure surfaces and the source-of-truth that must rotate

Surface	Volume	Predominant secret	Rotate at
Exposed .env / web artifacts	750 hosts; 2,571 .env rows	AWS keys (~2,600)	AWS IAM / KMS — deactivate key, issue new, scope to least privilege
Exposed `.git` config / directory	80 + 61	Embedded remotes & creds	Provider tokens + any credentials in history
Public Git repositories	619 secrets; 478 repos	Generic, Google, AWS, Telegram	Respective cloud / platform IAM and token consoles
Private keys / PEM	2 + 1	Private key material	Re-key the asset; revoke and re-issue certificates

The fix, concretely:

1. **Rotate at the issuing authority, immediately.** For the ~2,600 exposed AWS credentials, the corrective action is in AWS IAM: deactivate and delete the leaked access key, issue a replacement, and — critically — scope the replacement to least privilege so the next leak is less catastrophic than the last. For keys that protect data, rotate the underlying KMS key material and re-encrypt. For Google, platform, and messaging tokens, revoke in the respective console. Rotation, not deletion, is the remediation.
2. **Purge history, not just HEAD.** A secret deleted from the latest commit remains in Git history and in every clone. Rewrite history to remove it — but understand that history rewriting reduces future discoverability, it does not restore the secret's secrecy. Rotation must come first.
3. **Eliminate static long-lived keys.** The structural fix for "AWS keys in a .env on a public host" is to stop having long-lived keys at all: adopt short-lived, federated credentials (instance/workload identity, OIDC-issued role assumption) so there is no durable secret to leak.
4. **Stop the leak at commit time.** Deploy pre-commit and CI secret scanning, and serve `.env`, `.git`, dumps and key files from outside the web root with

explicit deny rules. The 2,571 exposed `.env` rows are, in nearly every case, a web server configured to serve dotfiles it should refuse.

5. **Hunt your own public footprint.** Continuously scan public Git and the open web for your organization's secrets — the same passive technique behind our leaked-credentials and exposed-keys radars — so that you, and not an adversary, are the first to find them.

Who it hits: Cloud-native organizations whose blast radius from a single AWS secret key can be the entire account. A leaked key with broad IAM permissions is functionally a master key; the "rotate plus least-privilege" pairing is what converts a catastrophe into an incident.

Incident parallel: The 2019 Capital One breach turned on cloud credentials and an over-permissioned role reaching storage that held the data of more than 100 million people. The credential was the pivot; the excessive privilege was the multiplier. Both halves of that lesson — rotate the credential, and scope it so the next one matters less — are the remediation for the ~2,600 AWS keys in this dataset.

Priority 3 — Known-exploited vulnerabilities: a KEV-first patch cadence

So what: A vulnerability on the CISA Known Exploited Vulnerabilities catalog is, by definition, one that adversaries are using in the wild right now. The window between "exposed" and "exploited" for these is not theoretical, and the prioritization is made for you: patch what is being exploited, before what merely could be.

We observed **36,416 hosts running software with known CVEs** — **165,717 host-to-CVE pairs** across **6,519 organizations** and **161 countries** *(method superseded 2026-09-29)*. Of those hosts, **21,299 are exposed to CISA-KEV actively-exploited vulnerabilities** (spanning 50 distinct KEV CVEs), and **3,475 carry a vulnerability with a documented ransomware association**. By host severity, **24,292 hosts carry a CRITICAL** exposure, 26,482 a HIGH, and 19,329 a MEDIUM.

> **21,299 of the 36,416 vulnerable hosts we observed are exposed to vulnerabilities that attackers are actively exploiting today — these define where patching must start *(method superseded 2026-09-29)*.**

The exposure concentrates in a short list of widely deployed software. **OpenSSH leads with 12,889 hosts**, followed by Apache Tomcat (5,219), Apache httpd (3,491), MongoDB (3,230) and VMware ESXi (2,816). At the CVE level the most-exposed flaws are the SSH transport and remote-code-execution issues that have defined recent advisories:

Most-exposed known-exploited CVEs by host count

CVE	Common name / component	Hosts	Remediation
CVE-2023-48795	Terrapin — OpenSSH/SSH transport	9,942	Upgrade OpenSSH; disable affected ChaCha20-Poly1305 / CBC-EtM modes per advisory
CVE-2023-38408	OpenSSH (ssh-agent PKCS#11)	9,923	Patch OpenSSH; restrict agent forwarding
CVE-2025-26465	OpenSSH	8,525	Apply vendor-patched OpenSSH build
CVE-2023-44487	HTTP/2 Rapid Reset — Tomcat & HTTP/2 stacks	5,859	Patch the HTTP/2 implementation; apply rate/stream limits
CVE-2024-6387	regreSSHion — OpenSSH RCE	5,713	Upgrade to fixed OpenSSH; `LoginGraceTime 0` as interim hardening per advisory
CVE-2025-24813	Apache Tomcat	5,218	Upgrade Tomcat to the patched release

The fix, concretely:

1. **Adopt a KEV-first SLA.** Bind your patch cadence to active exploitation: KEV-listed and ransomware-associated vulnerabilities on internet-facing hosts get an aggressive, contractually-short remediation window; everything else follows the standard severity cadence. The 21,299 KEV-exposed hosts are the queue, and the 3,475 ransomware-linked hosts are the front of it.

2. **Patch the long tail of SSH first.** The single highest-leverage action in this dataset is upgrading OpenSSH across the 12,889 affected hosts — one component closes the four most-exposed CVEs at once (Terrapin, the agent flaw, CVE-2025-26465 and regreSSHion). Where an immediate upgrade is impossible, apply the advisory's documented interim mitigations (cipher/MAC restrictions for Terrapin; `LoginGraceTime 0` for regreSSHion).
3. **Reduce exposure where you cannot patch.** Management interfaces — SSH, hypervisor consoles (ESXi), CI servers (Jenkins, 1,568 hosts) — should not be openly reachable. Place them behind VPN or a bastion. Removing the network reachability buys time when the patch cannot ship today.
4. **Let likelihood, not just severity, drive the queue.** We observed **36,365 hosts** carrying high-EPSS exposures — a forward-looking signal of probable exploitation. Combine KEV (confirmed) and high EPSS (predicted) to triage; our KEV-exposure radar and the CVE intelligence feed maintain this enrichment continuously.

Who it hits: Everyone — but the host-severity distribution (24,292 CRITICAL) and the breadth (6,519 organizations) mean that infrastructure-heavy operators running large fleets of SSH-exposed Linux, Tomcat, and hypervisor hosts carry the deepest queues.

Incident parallel: The 2017 Equifax breach is the canonical demonstration of the cost of patch latency: a known, patch-available vulnerability in a public-facing application, left unremediated past the available fix, became the entry point to the records of 147 million people. The MOVEit campaign of 2023 made the same point at fleet scale, with a single known flaw weaponized across thousands of organizations. regreSSHion (CVE-2024-6387), with 5,713 exposed hosts in this dataset, is precisely the class of known, patchable, internet-facing RCE that those incidents warn about.

Priority 4 — Subdomain takeover: DNS hygiene and the dangling-record sweep

So what: A dangling DNS record — a subdomain still pointing at a de-provisioned cloud resource — lets an attacker claim that resource and serve content from a hostname your users and systems still trust. The blast radius is your brand and your trust boundary: phishing on a legitimate domain, cookie theft, and bypass of allow-lists that name the subdomain. The fix is pure hygiene and costs minutes per record.

We observed **7,952 subdomains exhibiting takeover-relevant configurations, of which 134 are confirmed vulnerable** — a confirmation rate that underscores how the

corrective action is targeted, not sweeping. The confirmed cases cluster on platforms where a CNAME outlives the backing resource: **Shopify (4,869 observed)**, **GitHub Pages (1,563)**, **Heroku (773)**, SmugMug (331) and Fastly (170) *(these are counts of the 7,952 observed, not of the 134 confirmed; corrected 2026-09-30)*.

Subdomain-takeover surface by fronting service *(partial: these five sum to 7,706 of the 7,952; the Subdomain Takeover chapter gives them as the top five providers; corrected 2026-09-30)*

Service	Observed	Typical cause	Fix
Shopify	4,869	Store removed, CNAME retained	Delete the DNS record, or re-claim the resource
GitHub Pages	1,563	Repo/site deleted, CNAME retained	Remove CNAME or re-create and verify the Pages site
Heroku	773	App deleted, custom domain dangling	Remove the record or re-attach the app domain
SmugMug	331	Account/domain unbound	Delete the dangling CNAME
Fastly	170	Service unconfigured for the host	Remove record or bind the host to an active service

The fix, concretely:

1. **Sweep for dangling records.** Enumerate every DNS record and resolve whether its target resource still exists and is owned by you. The 134 confirmed cases are removable today — delete the orphaned CNAME, or re-claim the underlying resource if the hostname is still needed.
2. **Decommission in the right order.** The structural root cause is teardown sequencing: the backing resource is deleted before — or instead of — its DNS record. Make "remove the DNS record" the first step of any decommission, not an afterthought.
3. **Govern DNS centrally.** Takeovers thrive in fragmented DNS ownership where no one team can see the whole zone. Consolidate authority, require records to reference owned resources, and audit continuously — the technique behind our subdomain-takeover radar.

Who it hits: Organizations with large marketing and microsite footprints on SaaS hosting — exactly where Shopify, Pages, and Heroku CNAMEs accumulate and outlive the campaigns that created them.

Incident parallel: Dangling-record takeovers are a recurring source of brand-impersonation and phishing infrastructure built on otherwise-trusted domains. Because the hostname remains legitimate, downstream controls that allow-list it — and users who recognize it — are bypassed by design. The defense is unglamorous and decisive: do not let a name point at a thing you no longer own.

Priority 5 — AI infrastructure: egress controls, authentication, and treating models as data-bearing systems

So what: The AI build-out is creating a new infrastructure tier faster than it is being secured. The remediation is not exotic — it is the same authentication, network-boundary, and egress discipline applied to any data-bearing service — but it must be applied to a class of systems many organizations do not yet inventory as such. We are deliberately precise about scope: of the **82,416 AI services discovered** across **137 countries**, the great majority were not confirmed open. **36,991 are auth observed or not determined *(corrected 2026-09-27)*; 42,593 resolve in DNS only; 782 are unreachable; and 2,050 are confirmed active and open.** An authenticated Shadow AI endpoint is not an exposed one, and we never describe it as such.

> **Of 82,416 AI services discovered, 2,050 (about 2.5%) are confirmed active and open — 1,743 open LLM proxies, 171 live vector databases, 2 MCP *(MCP count withdrawn 2026-09-29)* and 81 AI-workflow endpoints *(partial: these four sum to 1,997 of the 2,050; corrected 2026-09-29)*.**

The remediable surface is that confirmed-active set. Open LLM proxies (1,743) can be abused for resource theft, prompt injection, and as an unmonitored exfiltration channel; live vector databases (171) can leak the embedded knowledge — frequently proprietary or personal data — that an organization has loaded into them. The discovery counts by category (AI-Workflow 35,675, LLM-Proxy 19,572, Model-Store 18,317, Notebook 4,859, MCP 1,563 *(candidates, not confirmed MCP servers; see the 2026-09-29 correction)*, Vector-DB 589) *(partial: these six sum to 80,575 of the 82,416; corrected 2026-09-30)* describe the *footprint to inventory*; the active counts describe the *work to do now*.

The fix, concretely:

1. **Authenticate every inference and model endpoint.** LLM proxies, model servers, notebooks, and MCP endpoints must require authentication. An open inference endpoint is an open API to your compute and, increasingly, to your data. This single control retires the bulk of the 1,743 open proxies.
2. **Treat vector databases as the sensitive data stores they are.** A vector database holds your embedded documents and their semantics. Apply the Priority-1 data-store playbook to it without exception: authenticate, bind to a private network, encrypt in transit, default-deny at the perimeter. The 171 live vector DBs are data exposures, not curiosities.
3. **Impose egress controls.** AI infrastructure is unusually dangerous on the outbound path: a compromised or manipulated model server, agent, or workflow runner can become an exfiltration and command channel. Restrict outbound network access from AI workloads to an explicit allow-list of required destinations, and monitor it. Egress control is the differentiating mitigation for this tier and the one most often missing.
4. **Inventory the shadow footprint.** The footprint grew from 10,679 discoveries in May to 71,737 in June as observation scaled — a measure of how much AI infrastructure exists outside central inventory. Discover your own AI surface continuously before treating it; our Shadow AI radar and the surface self-check exist for exactly this.
5. **Patch AI-specific vulnerabilities.** The CVE corpus already contains **5,190 AI-related vulnerabilities**; the model-serving and orchestration stack is now a patch surface like any other and belongs in the Priority-3 cadence.

Who it hits: Organizations racing to ship AI features — where a proxy, notebook, or vector store is stood up for velocity and inherits none of the controls the rest of the estate takes for granted.

Incident parallel: The exposed-database failures of the cloud era are now repeating in the AI tier — an unauthenticated store, reachable from the internet, holding data the operator assumed was private. The same root cause (authentication and network boundary omitted under deadline) is migrating to vector databases and model endpoints. The defense migrates with it unchanged.

The master prioritization: where to spend the next hour

Consolidating the above into a single ordered queue, ranked by blast-radius reduction per engineering hour. An organization triaging its own exposure should descend this

list, not its CVSS report:

Remediation queue, ordered by blast-radius reduction per hour

#	Finding class	Observed scale	Blast radius if unaddressed	Effort	Core action
1	Open data stores	6,607 DBs / 1,723 orgs	Direct data-plane access — the breach is the first connection	Minutes/host	Authenticate + bind to private network
2	Leaked / exposed credentials	~2,600 AWS keys (web); 619 secrets / 478 repos	Account/identity compromise; a broad key is a master key	Minutes–hours	Rotate at source-of-truth + least-privilege
3	Known-exploited CVEs	21,299 KEV-exposed; 3,475 ransomware-linked	Active in-the-wild exploitation, incl. ransomware	Hours/fleet	KEV-first patch SLA; upgrade OpenSSH first
4	Subdomain takeover	134 confirmed of 7,952	Brand abuse, phishing on trusted domains, allow-list bypass	Minutes/record	Delete dangling records; fix teardown order
5	AI infrastructure	2,050 active/open of 82,416	Compute theft, model/agent exfiltration, embedded-data leak	Hours–program	Auth + egress controls; treat vector DBs as data stores

Remediation is a cadence, not an event

Two disciplines determine whether the work in this chapter holds. The first is **verification**: a remediation is not complete when the change is deployed but when the exposure is confirmed gone from the outside. Every fix above should be re-tested from the attacker's vantage point — the same passive, external view that produced these

findings — because the gap between "configured correctly" and "observably closed" is where regressions live.

The second is **continuity**. This report is a single 4-to-5-week observation *(window corrected 2026-09-29)*; the surface it describes is not static. New databases are exposed, new keys are committed, new CVEs are weaponized, and new AI endpoints appear every week — the May-to-June discovery growth is evidence enough. Point-in-time remediation against a point-in-time scan decays immediately. The durable posture is a standing program that continuously discovers exposure, ranks it by blast radius per hour, remediates at the source of truth, and verifies externally — closing the loop before an adversary completes it first. Mapping how a single closed exposure cascades to retire downstream risk is what tools such as the attack-graph view, and the compliance mappings at our compliance surface, are built to make legible.

None of the findings in this report required an adversary to be sophisticated. None of the fixes require the defender to be. What both require is order, cadence, and the discipline to spend the next hour where it retires the most risk.

Gaps & What We Are Not Seeing

The most dangerous number in any security report is the one that is missing. Every figure in the preceding chapters describes exposure that EchelonGraph could observe *from the outside, passively, without authenticating (method superseded 2026-09-29)*. That vantage point is deliberate, and it is powerful: it is the same vantage point a remote, unauthenticated adversary occupies before they have spent a single credential. But it is a vantage point, not omniscience. This chapter is the part of the report most analysts skip and the part a CISO should read first. It states plainly what our methodology cannot see, where our counts are floors rather than totals, and which entire classes of risk fall outside the lens. Read defensively: the absence of a finding in this report is not evidence of safety, and the totals we publish are almost certainly lower than the truth.

> **Every number in this report is a floor, not a ceiling. We count only what the public internet already reveals — the internal attack surface, by design, is invisible to us.**

A passive, external lens misses everything behind the perimeter

The single largest blind spot is structural: we observe the internet-facing surface only. An attacker who has already obtained a foothold — through a phished credential, a third-party compromise, a malicious insider, or a supply-chain implant — operates inside a perimeter our scanners never cross. The lateral movement, privilege escalation, over-permissioned service accounts, flat internal networks, unsegmented VLANs, and trust relationships that turn a single compromised host into a domain-wide breach are entirely outside this dataset. EchelonGraph never authenticates, never logs in, and never exploits; by construction we see the front door, not the corridors behind it.

This matters because the most consequential breaches of the last decade did not end at the perimeter — they began there and then traversed inward. The 2017 Equifax breach turned a single unpatched internet-facing application into the loss of records for nearly half the United States adult population because, once inside, the attacker found

a network that did little to contain them. The 2019 Capital One incident converted one server-side request forgery against a misconfigured component into access to data held in cloud storage, because the identity and network boundaries that should have separated the entry point from the crown jewels were permissive. Our external view would, at best, have flagged the exposed entry point in cases like these. It would have said nothing about the internal blast radius — the very thing that turned an incident into a catastrophe. Modelling that internal blast radius requires inside-the-fence telemetry; it is the explicit job of an authenticated posture-management and attack-graph capability, not of a passive external scan.

Concretely, the following risk classes are **out of scope for this report by design**:

- **Lateral movement and east-west exposure** — internal network reachability, segmentation failures, and trust paths between hosts.
- **Identity and entitlement risk inside the tenant** — over-privileged IAM roles, standing access, privilege-escalation paths, and dormant credentials that never touch the public internet.
- **Endpoint and workload state** — unpatched software on machines with no public listener, malware resident on internal hosts, and runtime behaviour.
- **Insider threat and credential abuse** — legitimate access used illegitimately, which by definition looks authenticated.
- **Data-at-rest governance** — what sensitive data actually lives in a store, how it is classified, and who can reach it from inside.

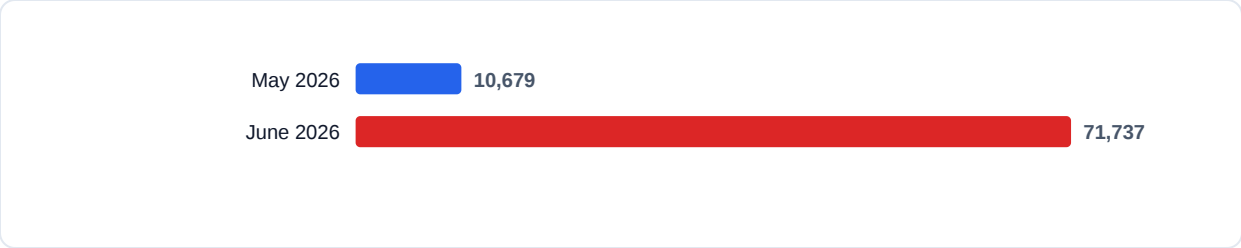
None of these are minor. In aggregate they are where most breach *impact* accrues. They are simply not visible from where we stand, and we will not pretend otherwise.

Our counts are floors: discovery is bounded by what is reachable in a five-week window *(window corrected 2026-09-29)*

Even within the external surface, our totals undercount reality for three compounding reasons, and a defender should mentally adjust every headline figure upward.

First, the observation window is short. This inaugural edition covers approximately five weeks, from 29 May to 28 June 2026 *(window corrected 2026-09-29)*. It is a baseline snapshot, not a multi-year trend line. Exposure is not static: databases are opened and closed, certificates are issued and revoked, repositories are made public and then scrubbed, and hosts rotate addresses daily. An open store that was briefly exposed in early May and re-secured before our window opened does not appear here,

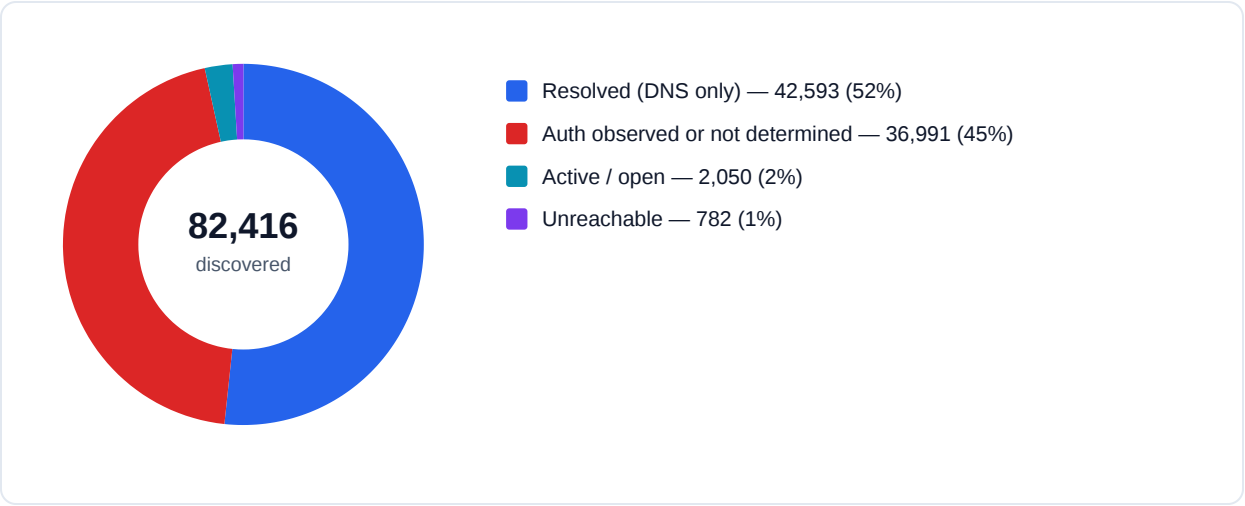
even though it was a real exposure with real risk *(window corrected 2026-09-29)*. Conversely, the scanner fleet itself ramped during the window — AI-infrastructure discovery rose from 10,679 records observed in May to 71,737 in June — which means our own coverage was still expanding as we measured. Month-over-month deltas in this edition reflect *our* growing reach as much as any change in the underlying internet; they should not be read as a real-world surge. We flag this explicitly rather than let the curve be misread as a trend.



AI-infrastructure discovery by month. The June step-change largely reflects scanner ramp-up within the observation window, not a real-world surge — a caution against over-reading short-baseline deltas.

Second, discovery is bounded by our seeds and sources. We find what Certificate Transparency logs, public internet-scan data, public source repositories, and public DNS reveal. A service with no TLS certificate logged to CT, on a non-standard port no public scanner sampled, behind a hostname never published in DNS, in a repository that was always private, is invisible to us — not because it is safe, but because it left no public trace we ingest. Internet-scale passive discovery is wide but never exhaustive. Where competitors might extrapolate a "total internet" estimate from a sample, we do not; we report observed counts only. That honesty costs us a bigger headline number and buys the reader a figure they can trust.

Third, the active-and-open subset is the conservative core of a much larger discovered footprint. Nowhere is the floor-not-ceiling principle more important than in the AI-infrastructure findings, where the gap between "discovered" and "confirmed exposed" is enormous and routinely misreported by others.



AI footprint by liveness (n=82,416). Only the 2,050 confirmed active-and-open services are counted as exposed; the 36,991 auth-observed-or-not-determined and 42,593 DNS-only hosts are explicitly not *(corrected 2026-09-27)*.

We discovered an AI-related footprint of 82,416 endpoints across 137 countries. It would be trivial — and wrong — to headline that as "82,416 exposed AI services." We do not, and no one should. Of that footprint, 36,991 endpoints are **auth observed or not determined**: they exist and they answered, and either demanded credentials we never supply or answered in a way our verifier could not classify *(corrected 2026-09-27)*. A further 42,593 **resolve in DNS only** — a hostname points somewhere, but we found no live, answering service behind it during our probes. Another 782 were **unreachable** at probe time. Only 2,050 endpoints were **confirmed active and genuinely open** — reachable and serving without authentication — and only those 2,050 are counted as exposed anywhere in this report. The honest breakdown:

AI footprint liveness state	Endpoints	Counted as “exposed”?
Discovered (total footprint)	82,416	No — discovery is not exposure
Auth observed or not determined	36,991	No — not confirmed open *(corrected 2026-09-27)*
Resolved (DNS only, no live service)	42,593	No — no answering service found
Unreachable at probe time	782	No
Confirmed active and open	2,050	Yes — the only figure we call exposure

The confirmed-open 2,050 decompose into 1,743 open LLM proxies, 171 live vector databases, 81 AI-workflow surfaces, and 2 Model Context Protocol servers *(MCP count withdrawn 2026-09-29) (partial: these four sum to 1,997 of the 2,050; corrected 2026-09-29)*. That a host requires authentication is a finding about its existence, never about its vulnerability. Counting the auth-observed-or-not-determined bucket as "exposed" would inflate the number roughly twentyfold *(corrected 2026-09-27)* and corrode the report's credibility. The price of that discipline is a smaller, slower, more defensible figure — and that is the right trade. The full discovered footprint is browsable at the Shadow AI Radar; the distinction between discovery and exposure is preserved there as it is here.

We detect the open door — we never read what is behind it *(method superseded 2026-09-29)*

A second deliberate limit governs the entire data-store and secret-exposure picture: we confirm that a store is reachable and unauthenticated, and we stop *(data-store check corrected 2026-09-29)*. We do not enumerate keys, count rows, sample documents, or read records. This is a hard ethical and legal boundary, not a technical limitation, and it shapes how our numbers must be interpreted.

Across the window we observed 6,607 open, unauthenticated databases spanning 1,723 organizations and 120 countries *(window corrected 2026-09-29) (method superseded 2026-09-29)* — Redis, Memcached, Kibana, Cassandra, MongoDB, InfluxDB, and CouchDB instances answering to anyone who connects. That is the count of **open stores**. It is emphatically not a count of exposed records. We classified only 3 of these stores as carrying probable personally identifiable information and 1 as carrying probable payment-card data, and that classification is deliberately conservative — inferred from non-content signals such as port, banner, schema metadata, and instance naming, never from reading the data *(method superseded 2026-09-29)*. The true sensitivity of the remaining stores is unknown to us by choice. We will never publish a "millions of records exposed" figure, because to produce one we would have to do the very thing an attacker does and we refuse to.

What we measure	What we deliberately do not measure
A store is reachable	How much data it holds
It answers without authentication	What records are inside
Engine type and version banner	The sensitivity of specific rows
Conservative PII/PCI indicators (3 / 1)	A record count of any kind

The history of this exact failure mode is instructive without our needing to read a byte *(method superseded 2026-09-29)*. The 2025 disclosure of an unauthenticated DeepSeek database — an open store reachable on the public internet — is the archetype: the exposure was the open door, and the moment a researcher (or adversary) connected, the contents were theirs. Our methodology would have flagged such a store as open. It would not, and ethically could not, have told you what was inside. That gap between "we know it is open" and "we know what it leaks" is permanent and intentional. The defender's correct response is to treat any open store as fully compromised until proven otherwise — precisely because the people who do read the contents will not announce themselves. The conservative open-store inventory is published, host-redacted, at Exposed Databases.

Attribution is rare, gated, and aggregate by design

A finding is only as actionable as our confidence in who owns it, and here too we have chosen restraint over reach. We attribute an exposure to a named organization only when ownership is independently provable through forward-confirmed reverse DNS — the reverse record for an address resolves back to a forward record under the claimed domain. FCrDNS is a high bar, and it fails far more often than it succeeds across the messy reality of cloud IP allocation, shared hosting, content-delivery networks, and dynamic addressing. The consequence is that attribution in this dataset is rare, and almost all findings are reported in aggregate and host-redacted rather than tied to a specific entity.

This is a feature, not a defect, but it bounds what the report can claim. We can tell you, with evidence, that 21,299 hosts are running software affected by vulnerabilities CISA lists as actively exploited; we generally cannot, and will not, name those hosts in this document. The organizational counts we do publish — 6,519 organizations touched by known-exploited vulnerability exposure, 1,723 with an open database, 6,607 stores in total — are themselves floors, because every host we could not confidently attribute is

excluded from the org tally even though it is included in the host tally. Responsible disclosure runs through private channels to the parties we can identify, not through this report. A reader should not infer that a low organizational count means few organizations are affected; it means few are affected *and* attributable to our standard. Owners who want a definitive, authenticated answer for their own estate should run the free Surface Scanner rather than infer their status from our aggregates.

What this report does not cover

For the avoidance of doubt, the following are explicitly outside the scope of this edition. Their absence here is a statement about our chosen vantage point, never a statement that the risk does not exist.

- **Internal and authenticated attack surface** — anything reachable only after a successful login or from inside the network boundary.
- **Cloud entitlement and identity risk** — IAM over-permissioning, privilege-escalation chains, standing access, and dormant keys that never face the internet. These demand an authenticated cloud-posture view, not a passive scan.
- **Endpoint, workload, and runtime state** — host-level patching of non-public software, in-memory malware, container runtime behaviour, and reachability of a vulnerability in execution.
- **Data classification and governance at rest** — what data a store holds and its regulatory weight; we confirm an open door, never its contents *(data-store check corrected 2026-09-29)*.
- **Active validation** — we never exploit, so a flagged vulnerability is a known-affected version, not a confirmed-exploitable instance in a given environment.
- **Application-layer logic flaws** — business-logic abuse, broken authorization within an authenticated app, and chained behaviours that only manifest in use.
- **Encrypted, private, or unindexed surface** — services and repositories that leave no public trace in CT, DNS, public scan data, or public source hosting.
- **Causation and intent** — we observe exposure; we do not assert that any specific exposure has been or will be breached *(method superseded 2026-09-29)*.

Where this report ends, the work of an authenticated, inside-the-fence security program begins. Passive external intelligence and authenticated posture management are complements, not substitutes: the former tells you what every adversary already sees

before they touch your systems; the latter tells you what happens after they do. Mapping known-exploited exposure to your own assets starts at KEV Exposure, and mapping it to a control framework starts at Compliance. This chapter exists so that the strength of the numbers in the rest of the report is not mistaken for completeness. They are a floor, gathered honestly, from the outside. The ceiling — and the interior — is yours to measure.

CHAPTER 13

The CISO Playbook

Every preceding chapter described the internet as it is. This chapter is the only one written to be acted on. It converts thirty days of passive, detect-only observation *(window corrected 2026-09-29)* into a sequenced ninety-day program that a CISO can hand to an engineering and security team on a Monday morning. The structure is deliberate and the ordering is not arbitrary: each action is mapped to a specific number from this report, sequenced by return on effort, and scoped so that the highest-consequence, lowest-cost work happens first. The thesis of the entire report holds here too — almost none of this requires a new tool, a new headcount, or a new budget line. It requires knowing where to look, and looking before an adversary does.

We have written this as three windows — the first thirty days, days 31 through 60, and days 61 through 90 — because that is the cadence in which security work actually ships and the cadence in which a board expects to see movement. Treat the windows as priority tiers, not calendar law. If your organization can close the first window in a week, close it in a week. What matters is the order.

The board-level summary, in one paragraph

Read this paragraph into the minutes and move on; the rest of the chapter is the working detail behind it.

> **Across roughly four weeks of passive observation *(window corrected 2026-09-29) (data-store check corrected 2026-09-29)*, EchelonGraph found 6,607 open, unauthenticated databases, 21,299 hosts running software with vulnerabilities that CISA has confirmed are being actively exploited, ≈2,600 cloud credentials sitting in exposed web files, and 619 secrets committed to public source-code repositories — all visible to anyone on the internet, none requiring a single exploit to reach.**

The exposures in this report are not sophisticated attacks waiting to happen. They are configuration defaults left unchanged, credentials checked into the wrong place, certificates pointed at services that no longer exist, and patches not yet applied to vulnerabilities whose exploitation is a matter of public record. The remediation is correspondingly unglamorous: close the open door, rotate the leaked key, apply the known patch. The single most important decision a board can make this quarter is to fund the time to look at the organization the way the internet already does — from the outside, continuously, before the next breach disclosure makes the introduction for them. The actions that follow are ranked by that principle, and the first one is the one that, on this evidence, prevents the most damage for the least money.

The single highest-ROI move: find and close your open data stores

If you do one thing after reading this report, do this. Of every exposure class we measured, an open, unauthenticated database is the shortest path from "on the internet" to "data in someone else's hands." There is no vulnerability to weaponize, no credential to phish, no lateral movement to engineer. The store answers. We found 6,607 of them across 1,723 organizations and 120 countries in four weeks *(window corrected 2026-09-29)*, the overwhelming majority — 4,243 Redis and 1,756 Memcached instances — being caching and in-memory engines that ship with no authentication enabled by default and are routinely stood up "temporarily" and never locked down.

> **5,999 of the 6,607 open databases we observed — roughly nine in ten — were Redis or Memcached instances, two engines that bind to all interfaces with no password unless an operator explicitly says otherwise *(method superseded 2026-09-29)*.**

This is the DeepSeek 2025 failure mode: a single ClickHouse database left open to the internet, no authentication, full read access to internal logs and chat history, discovered by an outside researcher rather than the operator. It is also the structural lesson of the broader decade — Capital One in 2019 was a misconfigured access path to data that should never have been reachable; the pattern that recurs is not a clever zero-day but a store that was simply left answering. The work is concrete and finishable inside the first window.

We deliberately classified our findings conservatively — we detect the open store, we do not read its contents *(data-store check corrected 2026-09-29)*, and so we counted only what we could prove from metadata: 3 stores carrying probable personally identifiable information and 1 carrying probable payment-card data. The honest reading is not "only four databases matter." It is the opposite: 6,607 stores are open, and an attacker — unlike us — will read every one of them. You must assume each of yours that appears in this class is fully readable, because to an adversary it is.

Action 1 — Eliminate open data stores (highest priority)

Step	What to do	Mapped to
1.1	Enumerate every Redis, Memcached, Elasticsearch/Kibana, MongoDB, Cassandra, InfluxDB and CouchDB instance your organization runs, including developer and "temporary" ones.	6,607 open DBs; Redis 4,243, Memcached 1,756, Kibana 464 *(partial: these three sum to 6,463 of the 6,607; the other four engines are in the Exposed Data Stores chapter's table; corrected 2026-09-30)*
1.2	For each, confirm it is not reachable from the public internet. Bind to localhost or a private subnet; put a security group / firewall in front; require authentication.	Default-no-auth engines dominate the data set
1.3	Treat any instance that was internet-reachable as potentially read. Rotate anything it held that is a credential; assess the data it exposed for breach-notification obligations.	Conservative 3 PII / 1 PCI — assume yours is readable
1.4	Add an external check (scan your own ranges from outside) so a re-opened store is caught in days, not at breach disclosure.	Continuous external view

Days 1–30: stop the bleeding (the exposures with no exploit step)

The first window targets exposures where the distance between discovery and compromise is effectively zero — an attacker needs no vulnerability and no skill, only the address. Open data stores, above, lead this window. Three more belong beside them.

Rotate credentials that are already public. A leaked key is a valid key until you revoke it. We found ≈2,600 AWS credentials in exposed web files — 1,319 AWS secret keys and 1,284 AWS access-key IDs — overwhelmingly from 2,571 publicly served `.env` files, the single most common exposure type at 750 affected hosts. Separately, in public Git repositories, we found 619 secrets across 478 repositories — 249 generic secrets, 228 Google credentials, 63 AWS, plus Telegram, Discord, OpenAI and Hugging Face tokens *(partial: the three counts sum to 540 of the 619; the four providers named after them hold 70 more, and 9 are not accounted for; corrected 2026-09-30)*. These are not theoretical. The 2019 Capital One incident turned on credentials that granted more access than they should have; the lesson generalizes to every key in this set.

> **≈2,600 AWS credentials in exposed web files plus 619 secrets in 478 public Git repositories — every one of them valid until someone rotates it.**

- **Rotate first, investigate second.** For any key that could be in your namespace, revoke and reissue immediately; forensic attribution can follow. A rotated key costs you a deployment; an un-rotated public key costs you the account.
- **Remove the `.env` from the web root** and add a deny rule so dotfiles and `.git` directories are never served. 2,571 exposed `.env` files is a web-server-configuration problem with a one-line fix.
- **Scrub Git history, not just `HEAD`.** A secret removed in the latest commit is still in the log. Rotate the secret regardless — history rewriting does not un-leak what was already cloned.
- **Stand up automated secret scanning** on every repository and in CI so the next credential is caught before it merges, not four weeks later by a stranger.

Reclaim or remove dangling DNS. Subdomain takeover is the quiet entry in this report and one of the cheapest to fix. We observed 7,952 subdomains pointing at third-party services and confirmed 134 as currently hijackable — a dangling CNAME aimed at a deprovisioned Shopify, GitHub Pages, Heroku, SmugMug or Fastly resource *(these platforms are named from the 7,952 observed; the report does not say which platforms the 134 confirmed are on; corrected 2026-09-30)* that an attacker can simply claim and serve content from under your name, harvesting cookies, phishing your users, and bypassing same-origin trust. The fix is a DNS edit. Inventory every CNAME that points

off-domain, verify the target is still claimed by you, and delete the record the moment the backing service is decommissioned.

Govern the AI footprint you did not know you had. We discovered an AI footprint of 82,416 services across 137 countries. The honest framing matters and we hold to it: that is the *discovered* footprint, not an exposed one. The large majority was not confirmed open — 36,991 are auth observed or not determined *(corrected 2026-09-27)*, and 42,593 resolve in DNS only. What demands first-window attention is the confirmed-active-and-open set: 2,050 services answering with no authentication, including 1,743 open LLM proxies and 171 live vector databases. An open LLM proxy is an uncapped bill and a data-exfiltration channel; a live vector database is your retrieval-augmented knowledge base — often your internal documents — readable by anyone who finds it. The action is to inventory AI services the way you inventory any other asset, and to put authentication in front of every proxy and vector store before it joins the active-and-open column.

Window 1 targets, by observed count — the exposures with no exploit step between discovery and compromise

Days 31–60: close the known-exploited vulnerabilities

The second window moves from open doors to weak ones: software with vulnerabilities whose exploitation is not hypothetical but documented. This is where a patch-prioritization program earns its keep, because the data refutes the instinct to chase every critical CVE equally. Of 36,416 hosts we found running software with known CVEs, 21,299 were exposed to vulnerabilities on CISA's Known Exploited Vulnerabilities catalog — flaws confirmed to be actively exploited in the wild. That subset, not the raw critical count, is the queue.

> **21,299 of 36,416 vulnerable hosts — nearly three in five — carried a vulnerability that CISA has confirmed is being actively exploited. Patch that queue first.**

Two patterns make this window tractable. First, a small number of products carry most of the exposure: OpenSSH (12,889 hosts), Apache Tomcat (5,219), Apache httpd (3,491), MongoDB (3,230) and VMware ESXi (2,816) lead the list. Patching a handful of widely deployed packages retires a disproportionate share of risk. Second, a small number of CVEs dominate by host count — and several are household names. CVE-

2024-6387, the OpenSSH "regreSSHion" remote-code-execution flaw, appears on 5,713 hosts. CVE-2023-48795, the "Terrapin" SSH downgrade, appears on 9,942. CVE-2025-24813 in Tomcat appears on 5,218. These have public exploit context; their presence on an internet-facing host is a finding to action this window, not next quarter.

Most-exposed CVEs by host count — the second-window patch queue

CVE	Name / component	Hosts
CVE-2023-48795	Terrapin — OpenSSH	9,942
CVE-2023-38408	OpenSSH	9,923
CVE-2025-26465	OpenSSH	8,525
CVE-2023-44487	HTTP/2 Rapid Reset — Tomcat	5,859
CVE-2024-6387	regreSSHion — OpenSSH	5,713
CVE-2025-24813	Apache Tomcat	5,218

The Equifax 2017 breach is the canonical reminder of what this window is for: a known, patchable vulnerability in a widely deployed component, with a fix available, left unapplied on an internet-facing system. The cost of that gap was not technical sophistication on the attacker's side; it was time on the defender's. The same logic applies to ransomware exposure — 3,475 of our CVE-exposed hosts carried vulnerabilities specifically linked to ransomware operations, and ransomware crews work precisely this catalog of known, reliable, internet-reachable flaws.

1. **Sort your patch backlog by exploited-in-the-wild status, not by CVSS alone.** A known-exploited high outranks a theoretical critical. Map your inventory against the KEV catalog and let that ordering drive the sprint.
2. **Lead with the high-count packages.** OpenSSH, Tomcat, Apache httpd, ESXi and the like return the most risk-reduction per patch cycle because they are everywhere.
3. **Where a patch cannot ship this window, compensate.** Restrict the service to known networks, put it behind a gateway, or take it off the public internet — exposure removed is as good as patched for the duration.
4. **Prioritize the ransomware-linked subset** for emergency change windows; these are the flaws most likely to convert directly into an extortion event.

Days 61–90: build the muscle so this list never grows back

The third window is the one that determines whether next year's report finds your organization in the same state. The first two windows are remediation; this one is the operating discipline that keeps the exposure surface from silently re-accumulating. Every number in this report exists because something was stood up, mis-configured, or forgotten and never re-checked from the outside. The countermeasure is to make the outside view continuous and to own your inventory.

- **Adopt a continuous external view of your own perimeter.** This entire report was produced passively, from public data, in roughly four weeks *(window corrected 2026-09-29)*. An adversary's reconnaissance is identical. Run the same lens against your own ranges on a recurring schedule so that a re-opened database, a fresh `.env` leak, or a newly dangling CNAME surfaces in days rather than at disclosure. EchelonGraph's surface scanner exists for exactly this self-check.
- **Map blast radius before the incident, not during it.** The reason a single open store or one leaked key becomes a breach is that the exposed asset connects to more than its owner remembers. Understanding those paths in advance — which credential reaches which account, which host pivots to which data — converts firefighting into containment. This is the function of an attack-graph view of your environment.
- **Bring AI infrastructure into asset management.** The June scanner ramp in our own data — AI discovery rising from 10,679 services in May to 71,737 in June — reflects how fast this surface is being built across the internet. Inside your walls it is being built just as fast and just as informally. Every model endpoint, proxy and vector store needs an owner, an authentication requirement, and a place in the inventory, or it becomes next year's shadow finding.
- **Track exploited-vulnerability exposure as a standing metric.** "How many internet-facing hosts carry a known-exploited CVE today" is a single number a board can follow quarter over quarter. We track 340,552 CVEs — 1,621 on the CISA-KEV catalog, 326 of them ransomware-linked — and enrich each with CVSS, EPSS, KEV status, SSVC and ransomware context through the EchelonGraph pulse feed so prioritization is driven by exploitation reality rather than raw severity.
- **Wire exposure findings to compliance evidence.** Much of this work already maps to obligations you carry. Closing open stores, rotating credentials and

patching exploited vulnerabilities is also audit evidence; routing it through a compliance mapping means the same effort satisfies both the security and the regulatory ledger.

The ninety-day plan at a glance

The table below is the chapter compressed to one page — the artifact to hand to the team and to revisit at each window's close. The ordering, again, is the message: open doors before weak ones, and a durable external view before either grows back.

The EchelonGraph 30/60/90 exposure-reduction plan

Window	Objective	Lead actions	Mapped evidence
Highest ROI	Close open data stores	Enumerate, lock down, assume-read & rotate, add external recheck	6,607 open DBs (4,243 Redis, 1,756 Memcached) *(partial: these two sum to 5,999 of the 6,607; the other five engines are in the Exposed Data Stores chapter's table; corrected 2026-09-30)*
Days 1–30	Stop the bleeding — no-exploit exposures	Rotate public credentials; remove served `.env`/`.git`; fix dangling DNS; authenticate active-open AI	≈2,600 AWS creds; 619 Git secrets; 134 hijackable subdomains; 2,050 active-open AI services
Days 31–60	Patch the known-exploited	Reprioritize by KEV; patch high-count packages; compensate where no patch; ransomware-linked first	21,299 KEV-exposed hosts; 3,475 ransomware-linked; top: OpenSSH, Tomcat, httpd, MongoDB, ESXi
Days 61–90	Build durable discipline	Continuous external view; blast-radius mapping; AI in asset management; KEV metric; compliance wiring	340,552 CVEs tracked; 1,621 KEV / 326 ransomware-KEV; May→June AI ramp 10,679→71,737

A closing note on tone for the team this lands with. None of the four windows asks for heroics. The work is enumeration, configuration, rotation and patching — the unfashionable core of the discipline. Its difficulty is not technical; it is organizational, the difficulty of looking at yourself honestly and continuously from the outside. The organizations that will not appear in next year's edition of this report are not the ones with the largest security budgets. They are the ones that did this list, in this order, and then never stopped looking.

CHAPTER 14

Past Mistakes & Lessons

The exposures in this report are not novel. Every category we measured maps cleanly onto a breach the industry has already paid for — in regulatory fines, in stock-price collapse, in the resignation of named executives. The uncomfortable finding of our baseline is not that new mistakes are being made. It is that the *same* mistakes, with the same root causes, are still observable at internet scale years after the canonical incident made headlines. The patch exists. The guidance is written. The post-mortems are public. And yet the exposure persists.

This chapter takes three of the most-studied breaches of the last decade — Equifax (2017), Capital One (2019), and DeepSeek (2025) — and shows where the conditions that caused each one are still measurable in our 2026 data. We do not speculate about the internals of any victim's environment beyond what those organisations and their regulators have already disclosed publicly. The point is not to re-litigate any single incident; it is to demonstrate that the failure *pattern* is structural, recurring, and — critically — externally observable before the breach, not only after it.

> **Three landmark breaches. Three root causes. All three still observable, at scale, in a four-to-five-week passive scan of the public internet in 2026 *(window corrected 2026-09-29)*.**

Equifax (2017): the unpatched edge and the flat network behind it

So what: the breach that compromised the personal data of roughly 147 million people began with a single internet-facing application running a vulnerability for which a patch already existed, and was made catastrophic by a network that imposed no friction once that edge was crossed. Both halves of that pattern — the exposed unpatched edge and the lack of internal segmentation — are the dominant findings of our Known-Exploited Vulnerability dataset and the premise of our attack-graph model.

The publicly established facts of Equifax are narrow and we will not embellish them: an internet-facing Apache Struts component carried a known, actively-exploited remote-

code-execution vulnerability; the fix had been available for roughly two months before exploitation; and once attackers held a foothold, they were able to move and exfiltrate over an extended period without being stopped. Two governance failures sit underneath that: the organisation did not know the vulnerable component was running where it was, and the internal network treated a compromised web tier as trusted.

Our data says that combination is not a 2017 artefact. We observe **36,416 hosts** running software with known CVEs, and **21,299 of them** exposed to vulnerabilities on CISA's Known-Exploited Vulnerabilities catalog *(method superseded 2026-09-29)* — i.e. flaws with confirmed, in-the-wild exploitation, not theoretical CVSS scores. These are not edge cases; they are the load-bearing software of the internet:

Most-exposed software carrying known-exploited vulnerabilities (hosts)

Software	Hosts exposed	Equifax-pattern parallel
OpenSSH	12,889	Internet-facing service, patch available, exposure unchanged
Apache Tomcat	5,219	Public app tier — the same class of component as Struts
Apache httpd	3,491	Web tier reachable from the internet
MongoDB	3,230	Data tier reachable through the edge
VMware ESXi	2,816	Hypervisor exposure — blast radius is the whole host
Citrix NetScaler	1,709	Edge appliance, repeatedly weaponised in the wild

The single sharpest parallel in the dataset is **regreSSHion (CVE-2024-6387)**, an unauthenticated remote-code-execution flaw in OpenSSH, which we observe on **5,713 hosts**. A fix was published; the exposure did not move. This is the Equifax pattern stripped to its essence — the patch existed, the operator did not apply it, and the vulnerable service is still answering on the public internet. The Terrapin and related OpenSSH issues (CVE-2023-48795, 9,942 hosts; CVE-2023-38408, 9,923 hosts; CVE-2025-26465, 8,525 hosts) tell the same story at even greater scale.

The second half of the Equifax lesson — flatness — is what turns an exposed host into a company-ending event. A single compromised edge is a contained incident only if the network behind it refuses to cooperate. Across our KEV corpus we record **165,717**

host-to-CVE pairs across 6,519 organisations and 161 countries, with host-severity counts of **24,292 CRITICAL and 26,482 HIGH**. An attacker does not need all of them. They need one reachable foothold and a path inward — which is precisely what an attack-graph view of reachability is built to surface, and precisely what most organisations still cannot see about themselves.

- **The recurring mistake:** treating "patched in the catalogue" as "patched in production," and treating the internal network as a trust boundary rather than a hostile transit medium.
- **The externally-visible warning sign:** a known-exploited CVE answering on a public IP. We can see it from the outside; so can an adversary. The defender's disadvantage is purely one of attention, not information.
- **The fix that already exists:** exposure-driven patch prioritisation (KEV first, then high-EPSS), plus segmentation so that one foothold is not the whole estate. Both were known before 2017.

Capital One (2019): the over-permissioned credential reachable from the edge

So what: the breach that exposed roughly 100 million credit applications did not begin with a zero-day or a sophisticated implant. It began with a request that reached an internal metadata endpoint, retrieved a set of credentials, and used the permissions attached to those credentials to read storage those credentials never needed to touch. The root cause was not the entry technique; it was a credential with far more authority than its job required, reachable from a position an attacker could occupy. Our secret-sprawl and leaked-credentials datasets show that over-scoped, recoverable credentials remain abundant in the open.

The publicly disclosed mechanics are, again, narrow: a server-side request reached an internal credential-issuing endpoint; the credentials it returned were over-permissioned relative to the task; and those permissions allowed bulk read of customer data in object storage. The single transferable lesson is that *the permission, not the perimeter, was the breach.* A least-privileged credential in the same position would have failed safely.

Our scan does not (and by methodology cannot) read inside anyone's cloud account. What it can see is the upstream version of the same failure: long-lived cloud credentials left in places the public internet can reach. On the web we observe **750 hosts** leaking secrets through misconfigured exposure paths, dominated by **2,571 exposed environment (`.env`) files**. Inside that material we count roughly **2,600 AWS**

credentials — **1,319 AWS secret access keys** and **1,284 AWS access-key IDs** — sitting in the clear. Severity rows break down to **1,386 critical and 1,304 high** *(partial: these two sum to 2,690 of the 2,736 web secret findings in the Secret Sprawl chapter's table by severity, which also counts 46 medium; corrected 2026-09-30)*.

Recoverable cloud credentials observed in the open

Source surface	Finding	Count
Web (exposed files)	AWS secret access keys	1,319
Web (exposed files)	AWS access-key IDs	1,284
Web (exposed files)	Exposed `.env` files (carrier)	2,571
Public Git	Secrets across repositories	619 across 478 repos
Public Git	AWS-provider secrets	63
Public Git	Google-provider secrets	228

The Capital One lesson lands in two places. First, the credential should never have been recoverable from where it was — and yet our leaked-credentials radar finds **619 live secrets across 478 public repositories** (high 301, medium 273, critical 45), with cloud providers prominent: 228 Google, 63 AWS. Every one of those is a credential an attacker can simply pick up; there is no entry technique to develop. Second — and this is the half organisations consistently underweight — what determines blast radius is not that the key leaked, but *what the key can do.* Capital One's incident was severe because the recovered credential could read at scale. A leaked key bound to least privilege is an incident; a leaked key with broad read on a customer datastore is a disclosure event.

> **~2,600 AWS credentials observed sitting in exposed web files, plus 619 live secrets in 478 public repositories. The 2019 root cause is not historical — it is inventory.**

- **The recurring mistake:** issuing broad, long-lived credentials "to make it work," then leaving them recoverable from an internet-reachable position — a config endpoint, a committed `.env`, a public repo.

- **The externally-visible warning sign:** the credential is in the open. We find it passively; so does anyone running the same searches. Rotation is reactive; scoping is preventive.
- **The fix that already exists:** short-lived credentials, least privilege enforced at issuance, and secret-scanning on every public surface — repo, build, and web root. None of this is new guidance.

DeepSeek (2025): the open data store nobody meant to publish

So what: in early 2025 a fast-growing AI provider was found to have a database reachable on the public internet with no authentication, exposing operational data and secrets to anyone who pointed a browser at it. There was no exploit. There was no credential to steal. The store was simply open. Our exposed-database radar and the Shadow AI dataset show that "open by default, secured never" is the single most common — and most preventable — exposure on the internet today.

The publicly reported facts are stark in their simplicity: an internet-facing database, no authentication required, containing material it should never have been possible to read anonymously. We treat the specifics conservatively and do not characterise volumes we did not observe. What matters is the shape: this was not a breach of a defence, because there was no defence to breach.

Our methodology is deliberately built to find this class of failure without ever crossing the line that the incident itself crossed — we detect that a store is open and unauthenticated; **we do not connect, authenticate, or read its contents** *(data-store check corrected 2026-09-29) (method superseded 2026-09-29)*. On that basis we observe **6,607 open, unauthenticated databases across 1,723 organisations and 120 countries** *(method superseded 2026-09-29)*. The engine distribution is dominated by in-memory and search stores that ship without authentication and are routinely deployed straight onto the internet:

Open, unauthenticated data stores by engine (hosts)

Engine	Open hosts	Why it recurs
Redis	4,243	No auth by default; frequently bound to a public interface
Memcached	1,756	No native auth; trivially reachable when exposed
Kibana	464	Search/analytics UI fronting indexed data, left open
Cassandra	75	Misconfigured listener, auth disabled
MongoDB	37	The original "open by default" cautionary tale
InfluxDB / CouchDB	16 / 16	Time-series / document stores exposed unauthenticated

We classify this set conservatively — confirming only **3 hosts as carrying probable PII and 1 as in PCI scope** from external signals alone, precisely because we will not read contents to inflate the number *(method superseded 2026-09-29)*. The honest finding is the count of open doors, not a claim about what is behind them.

The DeepSeek incident sits at the intersection of two of our radars, because it was both an open database *and* an AI-infrastructure exposure. On the AI side, our Shadow AI dataset discovered an **82,416-host AI footprint across 137 countries**. We are precise about liveness, and we never call an authenticated endpoint "exposed": of that footprint, **36,991 are auth observed or not determined** *(corrected 2026-09-27)*, **42,593 resolve in DNS only**, **782 are unreachable**, and **2,050 are confirmed active and open.** Within that active set sit **171 live, reachable vector databases** — exactly the category of AI data store that the DeepSeek incident turned into a headline. The vector database is the new system of record for retrieval-augmented AI; left open, it is the 2025 update to the open MongoDB of the last decade.

> **6,607 open, unauthenticated databases — plus 171 live, reachable vector databases in the active AI footprint. The DeepSeek failure shape is the most common exposure on the internet, not an outlier.**

- **The recurring mistake:** deploying a data store with authentication disabled and binding it to a public interface, on the assumption that "nobody will find it." Internet-wide scanning finds it within hours.
- **The externally-visible warning sign:** the store answers, unauthenticated, on a public IP. This is the most detectable failure in the entire report — and the most embarrassing, because it requires no skill to exploit.
- **The fix that already exists:** authentication on by default, never bind a datastore to `0.0.0.0`, and treat AI data stores (vector DBs, model stores, notebooks) as production data tiers with the same controls — not as experiments exempt from policy.

What the industry keeps getting wrong

So what: across three breaches separated by eight years and three completely different technology stacks, the root causes rhyme. The errors are not exotic. They are governance failures that every framework already prohibits, repeated because the controls that prevent them are unglamorous and the exposures that result are invisible from the inside. Five patterns recur in our data with enough frequency to call them structural.

Each landmark breach root cause, mapped to the 2026 dataset that still shows it at scale

1. **"Known" is not "fixed."** A vulnerability in a catalogue and a patch in a vendor advisory change nothing until they reach production. We track **340,552 CVEs**, of which **1,621 are CISA-KEV** and **326 are linked to ransomware** — and we still observe 21,299 hosts exposed to actively-exploited flaws *(method superseded 2026-09-29)*. The gap between disclosure and remediation is where every Equifax-class breach lives. Prioritise by exploitation, not by score: that is what enriched KEV and EPSS signal is for.
2. **Permission, not perimeter, sets blast radius.** Every credential is assumed to leak eventually; the only question that matters is what it can do when it does. The industry keeps issuing broad, long-lived keys and then acting surprised when one of the ~2,600 AWS credentials we see in the open turns a foothold into an exfiltration. Least privilege is a blast-radius control, not a compliance checkbox.
3. **Default-open is a decision, even when it is an accident.** 6,607 unauthenticated databases and 171 live open vector stores did not get

attacked — they got *published*, by operators who never turned authentication on. "Secure by default" has to be enforced by tooling, because "remember to secure it later" demonstrably does not scale.

4. **You cannot defend an asset you do not know you own.** Equifax did not know the vulnerable component was where it was. Capital One's credential reached storage nobody was watching. Subdomain takeover — we observe **7,952 candidates and 134 confirmed-vulnerable** dangling records — is pure inventory failure: an asset that still points somewhere the owner no longer controls. Attack surface management is not optional tooling; it is the precondition for every other control.
5. **New technology inherits old mistakes — faster.** The AI build-out is repeating the cloud build-out's earliest errors at compressed speed. Our AI footprint grew from **10,679 hosts discovered in May to 71,737 in June** within this baseline window. Some of that is our own scanner ramping; the trajectory of the underlying build-out is not. Vector databases, model stores, and notebooks are being stood up faster than the security controls around them, which is exactly how the data-store mistakes of the 2010s are becoming the AI-store mistakes of the 2020s.

The throughline. None of these breaches required a novel attacker capability. They required a defender to not know something about their own external posture that an outsider could see plainly — an unpatched edge, an over-scoped key, an open store, a dangling record. Every figure in this report was obtained passively, from public data, without logging in or exploiting anything. If we can see it, so can an adversary. The defender's only structural disadvantage is attention, and attention is fixable. The recurring lesson of Equifax, Capital One, and DeepSeek is not that defence is hard — it is that the basics, applied with discipline and verified from the outside, would have prevented all three. Map your own exposure before someone else maps it for you: start with a passive external surface check.

CHAPTER 15

The Path Forward

The preceding fourteen chapters are an inventory of what the open internet can already see: 82,416 discovered AI services, 36,416 hosts running software with known vulnerabilities, 6,607 open and unauthenticated databases, thousands of secrets sitting in public files and public repositories, and 134 confirmed-hijackable subdomains. None of it required a single login, an exploit, or a privileged vantage point. It was all visible from the outside, to anyone who chose to look — which means it is already visible to the people you do not want looking. The question this final chapter answers is not *what is broken*; the earlier chapters covered that. It is *what a serious organization does differently* so that next year its assets are not in this dataset.

The uncomfortable through-line of this report is that almost none of these exposures were sophisticated. They were defaults left on, snapshots left public, credentials committed in haste, DNS records left dangling after a service was decommissioned, and patches deferred past the point at which the vulnerability became a CISA-Known-Exploited entry. Sophistication was on the attacker's side of the ledger, not the defender's. That asymmetry is the opportunity: an exposure that is cheap for an adversary to find is, by definition, cheap for the defender to find first. The path forward is to find it first, as a routine, before it is weaponized.

The core shift: exposure is a discipline, not a project

The organizations that stay out of next year's edition will be the ones that stop treating external exposure as a quarterly assessment and start treating it as a continuously-running function — closer in cadence to monitoring than to auditing. The exposures in this report are not static. The discovery data alone moved from 10,679 AI services observed in May to 71,737 in June; that figure is dominated by our own scanner ramp and is **not** a measured growth rate, but the underlying reality it gestures at is real: attack surface changes daily. A developer ships a new subdomain, a team stands up a vector database for a retrieval pipeline, a contractor commits a config file, a load balancer is reconfigured. Each of those is an exposure event, and each happens on the timescale of a sprint, not a fiscal year.

> **An exposure that is cheap for an adversary to find is, by the same token, cheap for the defender to find first. The entire discipline reduces to one rule: find it first.**

A point-in-time assessment — a penetration test, an annual audit, a compliance attestation — produces a photograph. The internet operates as a film. The 2,050 AI services we observed *confirmed active and open* were a snapshot of a constantly-churning population; some have since been secured, and others have since appeared. A control that is verified once and re-verified twelve months later is unverified for the 364 days in between. The discipline of exposure management is the discipline of shrinking that blind window from a year to a day — and, for the highest-severity classes such as a newly-published CISA-KEV vulnerability on an internet-facing host, to hours.

This is not a call to buy a particular product. It is a call to assign an owner, a budget line, and a recurring cadence to a question that most organizations currently answer only when prompted by a customer questionnaire or a breach: *what can the whole internet see about us, right now?*

The find → protect model

Every category in this report decomposes into the same two-stage problem, and the order is not negotiable. You cannot protect an asset you have not found. The recurring failure mode behind the Capital One breach in 2019, the Equifax breach in 2017, and the wave of unauthenticated-database incidents that the 2025 disclosure of an open DeepSeek datastore typified, was not the absence of a control catalogue. It was that a specific asset — a misconfigured WAF path, an unpatched internet-facing application, an open database — was not on anyone's map at the moment it mattered. The control existed in policy; the asset did not exist in inventory. That gap, between the assets you govern and the assets you actually have, is where exposure lives.

We frame the path forward as a closed loop with two halves:

1. **Find** — continuous, outside-in discovery of every asset, service, and secret attributable to the organization, performed the way an attacker performs it: passively, from public data, with no reliance on an internal asset register that is itself probably stale. This is the half most programs skip, and it is the half that determines whether the rest is theatre. Of the 6,519 organizations we observed with at least one host carrying a known vulnerability, the overwhelming majority

were almost certainly running a vulnerability-management program. The program did not fail at remediation; it failed at enumeration — it acted only on the assets it knew about.

2. **Protect** — for each found exposure, the decision and the action: authenticate it, patch it, take it offline, rotate the leaked credential, reclaim the dangling DNS record, or accept the risk explicitly and on the record. The protect half is well-understood and well-served by existing tooling; Chapters 11 and 13 of this report set out the tactical and programmatic specifics. What this chapter adds is the insistence that protect is downstream of find, and is worthless without it.

The loop must close and then repeat. Reclaiming one of the 134 hijackable subdomains is a remediation; building the process that catches the 135th the day its CNAME starts dangling is exposure management. Rotating one of the roughly 2,600 AWS credentials we observed in exposed web files is hygiene; instrumenting commit-time and surface-time detection so the next leaked key is caught in minutes rather than discovered by a stranger is the discipline.

The find → protect loop: continuous outside-in discovery feeds prioritized protective action, which feeds re-verification — a cycle measured in hours and days, not quarters.

Continuous external monitoring: see yourself as the internet sees you

The single highest-leverage investment available to most organizations is to monitor their own external attack surface continuously, from the same vantage point an adversary uses. Every figure in this report was produced that way — passively, from Certificate Transparency logs, public internet-scan data, public source repositories, and public DNS, without ever authenticating or exploiting. Nothing in our methodology is exotic or proprietary to us; it is the standard reconnaissance toolkit, and it is available to your adversaries at negligible cost. The asymmetry today is that adversaries run this reconnaissance against you continuously and you run it against yourself rarely, if ever. Continuous external monitoring erases that asymmetry.

Concretely, the outside-in view is the only view that reliably surfaces the exposures that hurt most, because each of them is invisible from the inside:

Exposure class	Why internal tooling misses it	What outside-in monitoring catches	Observed in this report
Open, unauthenticated data stores	The database is "inside" a VPC that is, in fact, reachable from the internet; internal scanners authenticate and never test the anonymous path.	The store answering on its public interface with no credential challenge.	6,607 open DBs · 1,723 orgs · 120 countries
Known-exploited vulnerabilities on internet-facing software	Asset inventory is incomplete, so the vulnerable host is never scanned.	The banner, version, and the KEV/EPSS correlation, from the outside.	21,299 hosts on actively-exploited CISA-KEV vulns
Leaked credentials in public repositories	The repository is personal, or a fork, or a contractor's — outside the corporate SCM the security team watches.	The secret, the provider, and the public repo it sits in.	619 secrets · 478 repos
Secrets in exposed web files	The `.env` or `.git` directory is served by a misconfiguration no internal review anticipated.	The reachable file and the credential type within it.	750 hosts · ≈2,600 AWS keys
Subdomain takeover	DNS and the cloud service it pointed to are managed by different teams; neither sees the dangling record.	The CNAME pointing at an unclaimed service.	134 confirmed vulnerable of 7,952 observed
Shadow AI infrastructure	The team that stood up the proxy or vector store did not tell security it existed.	The service, its category, and whether it answers without authentication.	82,416 discovered · 2,050 confirmed active & open

A clarification this report has insisted on throughout, and repeats here because it is the heart of doing this honestly: discovery is not exposure. Of the 82,416 AI services in our footprint, 36,991 were auth observed or not determined *(corrected 2026-09-27)* and 42,593 resolved in DNS without serving anything reachable *(partial: these two sum to*

79,584 of the 82,416; the other two liveness states are the 2,050 confirmed open and the 782 unreachable; corrected 2026-09-30). Finding a service on your perimeter is the beginning of an investigation, not the end of one. The value of continuous external monitoring is not that it generates alarm; it is that it converts an unknown surface into a triaged, prioritized, evidence-backed worklist — separating the 2,050 things that are actually open from the 80,000 that merely exist. A monitoring practice that cannot make that distinction produces noise, and noise is abandoned.

Prioritize by exploitability and toxic combination, not by raw count

Outside-in discovery will, for any organization of meaningful size, return more findings than any team can action at once. The discipline is therefore as much about triage as enumeration. Two principles separate a program that reduces risk from one that merely produces tickets.

First, prioritize by evidence of exploitation, not by severity score alone. Of the 340,552 CVEs in our corpus, 103,468 are rated High and 33,409 Critical — a backlog no organization can clear, and most of which will never be weaponized. But 1,621 are on the CISA-KEV list, 326 of those are tied to ransomware operations, and 4,265 carry a high EPSS probability of near-term exploitation. Those are not 1,621 of 340,552 equal items; they are the ones for which the question "will this be attacked" has already been answered in the affirmative. The same logic applies host-side: a host on one of the 50 KEV vulnerabilities we observed in the wild outranks a host with a theoretically-higher-CVSS issue that no one is exploiting. Exploitability is the sort key. Severity is a tiebreaker.

Second, hunt for toxic combinations, not isolated findings. A leaked AWS key is a finding. A leaked AWS key plus an open S3-adjacent data store plus a public host on a known-exploited vulnerability is a breach path — and breach paths are what attackers actually traverse. The Capital One and Equifax incidents were not single-control failures; they were chains, in which one modest exposure granted the foothold that made the next one reachable. This is why the discipline cannot stop at a flat list of findings. It must model how findings connect — how an exposed credential reaches a database, how a vulnerable edge device reaches an internal segment, how a hijacked subdomain reaches a user's session. Mapping those relationships is the difference between a vulnerability list and an attack graph, and the latter is what tells you which of ten thousand findings to fix on Monday.

What "good" looks like in twelve months

An organization that has internalized the path forward exhibits a recognizable and auditable set of behaviors. None of them is exotic; all of them are continuous.

- **It knows its own external surface as well as an attacker does** — every domain, subdomain, internet-facing service, and exposed credential attributable to it, discovered continuously from outside-in, not reconstructed annually from an internal register it already suspects is incomplete.
- **It measures its blind window in hours, not months** — the time between an exposure event (a new dangling CNAME, a freshly-published KEV on a live host, a just-committed secret) and the organization becoming aware of it. This is the single most predictive metric of whether an organization appears in a report like this one, and it should be tracked as a board-level number.
- **It prioritizes by exploitability and breach-path, not by count** — KEV, ransomware-linkage, and high-EPSS findings are escalated automatically; toxic combinations are surfaced as paths, not buried as unrelated rows.
- **It has eliminated the cheap, default, catastrophic exposures entirely** — no open databases, no secrets in public files or repos, no dangling DNS, no unpatched internet-facing KEV vulnerabilities. These are table stakes, and every one of them appears, in volume, in this report.
- **It governs AI infrastructure the way it governs every other service** — model proxies, vector stores, notebooks, and MCP endpoints are inventoried, authenticated, and monitored, rather than stood up outside security's line of sight. The AI surface is the newest and fastest-moving category in this report, and it is the one most likely to be governed by no one.
- **It re-verifies continuously** — closing a finding triggers re-checking, and the loop runs again. Remediation without re-verification is a hope, not a control.

Tooling supports each of these behaviors and we build for them, but the behaviors are the point and they are vendor-neutral. An organization can buy nothing from anyone and still adopt the discipline; it cannot buy the discipline from anyone if it declines to adopt it. The deciding variable is organizational, not technical: whether someone owns the outside-in question and answers it every day.

Start where the leverage is highest

Exposure management can feel boundless, and that boundlessness is the most common reason it never starts. It should not. The data in this report points to a

concrete, ordered first move that any organization can make this quarter, sequenced so that the cheapest catastrophic exposures go first:

1. **Enumerate your external surface once, from the outside.** Before any remediation, get the map. You will almost certainly find assets you did not know you owned — that is the expected outcome, not a failure. Begin with a free external surface scan; the goal of the first pass is inventory, not action.
2. **Close the open data stores.** An unauthenticated, internet-reachable database is the highest-severity, lowest-effort exposure in this report and the one most directly responsible for the largest historical incidents. Authenticate or remove every one. Reference your own surface against the exposed-databases dataset.
3. **Rotate every leaked credential and reclaim every dangling DNS record.** Both are minutes of work per finding and both convert a wide-open door into a closed one. Cross-reference leaked credentials, exposed keys in web files, and subdomain takeover.
4. **Patch the known-exploited, internet-facing vulnerabilities first.** Not the largest CVE backlog — the exploited subset. Triage your perimeter against KEV exposure and the live CVE intelligence feed, sorted by EPSS and ransomware-linkage.
5. **Inventory and authenticate your AI infrastructure.** Treat every proxy, vector store, notebook, and MCP endpoint as a service that requires the same governance as a production API. Begin from the Shadow AI radar view of what is publicly discoverable.
6. **Then make all of the above continuous.** The first pass is a project. The repeat is the discipline. Wire the loop so that the next exposure event surfaces on its own, and map your findings as paths — against your attack graph and your compliance posture — rather than as a flat list.

Each of these is achievable this quarter. Together, over a year, they move an organization out of the population this report describes and into the one it aspires to.

A standing measurement of the open internet

This is the inaugural edition of the State of Internet Exposure, and it is deliberately framed as a baseline rather than a trend. The observation window is approximately five weeks, from 29 May to 28 June 2026 *(window corrected 2026-09-29)*; the figures are a first photograph of a moving subject. We have been explicit throughout about what that

photograph does and does not support — that discovery is not exposure, that an open store is detected but never read, that "millions of records" is a claim we will not make because we do not read contents, and that attribution is made only where ownership is DNS-provable *(method superseded 2026-09-29)*. Those constraints are not caveats bolted on at the end; they are what make the numbers in this report ones a CISO can carry into a board meeting without a footnote of doubt.

We intend to publish this report every year. The value of a baseline is realized only when there is a second measurement to set against it — when next year's edition can say, with the same passive methodology and the same conservative classification, whether the open-database count fell, whether organizations closed the gap between a KEV publication and a perimeter patch, whether the AI attack surface was brought under governance or grew further outside it. A single year is a snapshot. A series is a trend, and a trend is accountability — the kind the security industry has long had for incident counts and breach costs, and has lacked for the raw, observable, pre-incident exposure that precedes every one of them. That is the gap this series exists to close.

The path forward, in the end, is not complicated, and it does not depend on any single vendor. It is to look at yourself the way the internet already looks at you — continuously, from the outside, without flinching — and to fix what you find before someone else finds it first. Everything in this report was visible to anyone who chose to look. The organizations that thrive in the years ahead will be the ones that choose to look at themselves, first, every day.

See your own exposure now, from the same vantage point that produced every figure in this report: run the free EchelonGraph Surface Scanner.