Composite Trust Scoring and the Evasion Surface

A defender-facing analysis of where a bot-detection stack's assurance actually comes from

Sudipto Chandra

Independent Researcher

Abstract: **Between 2022 and 2026, commodity bot mitigation shifted from binary, single-signal blocklisting toward *composite trust scoring*: a request is admitted only if it satisfies a conjunction of independently evaluated detection layers. Cloudflare, which fronts a large share of all sites that deploy bot-mitigation tooling, exemplifies this architecture with a stack we decompose into fifteen layers grouped into five domains — network and transport, client environment, behavioral, interactive challenge, and programmable-cryptographic controls. This paper offers a defender-oriented taxonomy of that stack. For each layer we characterize the signal it reads, the structural reason it is or is not evadable, and the residual hardness that remains after best-effort evasion. Our central observation is that evadability is not uniform: layers that read *artifacts a client emits* (TLS handshakes, HTTP/2 frame ordering, header order) are structurally weak because the artifact can be reproduced exactly, whereas layers that read *properties a client must continuously possess* (genuine per-zone behavioral history, a privately held cryptographic credential) resist forgery by construction. We argue that the long-run defensive trajectory follows from this asymmetry, and that newer mechanisms such as cryptographic agent attestation are a different kind of control altogether: forgery-resistant by construction. Their present weakness is not cryptographic but a matter of deployment, since current configurations fail open when no signature is presented.**

Index Terms: **bot detection, web security, browser fingerprinting, TLS fingerprinting, trust scoring, adversarial measurement, cryptographic attestation.**

I. Introduction

Automated traffic now accounts for so much of what reaches a public web server that running some form of bot mitigation has become routine rather than exceptional. A large fraction of it passes through a single content-delivery and security provider, which sits in front of a sizable share of all sites that deploy anti-automation tooling. The practical effect is that one vendor's design choices set the threat model that scrapers and defenders alike have to reason about. Looking closely at how that system decides to admit or reject a request tells us less about a particular company than about the prevailing state of the art.

The decisive architectural change of the last four years is the move away from any single dispositive signal. Early systems asked a binary question of one feature at a time: is this IP on a blocklist, is this `User-Agent` obviously a script? Such systems failed in two directions at once: they were trivially evaded by an adversary who corrected the single checked feature, and they were prone to false positives, blocking legitimate clients that happened to share an incriminating attribute. The contemporary design instead computes a *trust score* from many independent detectors evaluated in parallel, and gates the response on that aggregate. This is a defensively sound move, but it changes the character of the evasion problem in a way that, we argue, is widely misunderstood: it is not that any one layer became unbreakable, but that an adversary must now defeat *all active layers simultaneously and consistently*, including layers that interrogate properties no emitted artifact can synthesize.

This paper provides a structured, defender-facing account of that stack as it stands in 2026. We deliberately frame the work around *why* each layer is or is not evadable rather than around operational evasion recipes; our aim is to give those building and evaluating defenses a principled map of where their assurance actually comes from. Section II formalizes the composite-scoring threat model and our methodology. Section III presents the fifteen-layer taxonomy. Section IV analyzes the resulting evasion surface and isolates the structural property that separates weak layers from strong ones. Section V draws defensive implications, and Section VI states the limitations and ethics of the study.

Our contributions are: *(i)* a consolidated, five-domain taxonomy of a production bot-detection stack; *(ii)* an *emit-versus-possess* distinction that predicts evasion hardness more reliably than the conventional "network vs. browser" framing; and *(iii)* an argument that cryptographic agent attestation is the first layer in this lineage to achieve forgery-resistance by construction rather than by obscurity.

II. Background and Threat Model

A. From Blocklists to Composite Trust Scoring

Let a request r be observed through a set of detectors $\{d_1 \dots d_n\}$, each returning a real-valued anomaly contribution $a_i(r) \geq 0$ rather than a hard verdict. The system maps these to a scalar trust score $S(r) = f(a_1(r), \dots, a_n(r))$, decreasing in each a_i, and acts by thresholding against two cut-points $\tau_{block} < \tau_{chal}$: it serves the content when $S(r) \geq \tau_{chal}$, interposes a challenge when $\tau_{block} \leq S(r) < \tau_{chal}$, and blocks when $S(r) < \tau_{block}$. The detectors are evaluated *concurrently*, and the empirically decisive feature of f is that it is *non-compensatory*: strong scores on many axes do not buy back a single strongly anomalous one. The trust score behaves as though dominated by its worst component — approximately $S(r) \approx g(\min_i s_i(r))$ for per-detector trusts s_i — so the admission region is close to an *intersection* $\cap_i \{r : a_i(r) \leq \theta_i\}$ of

per-detector acceptance regions rather than their weighted average. We call this the **conjunctive-pass property**: from the adversary's perspective the layers compose as a logical conjunction, even though the implementation is a soft score. A purely additive, compensatory aggregator would not exhibit it, so the property is itself evidence about the shape of f in the operative region.

Two consequences follow directly. Write p_i for the probability that a fixed adversarial strategy passes detector d_i. Because admission requires passing *every* active detector, the overall evasion probability obeys $P_{evade} \le \min_i p_i$; were the detectors conditionally independent given the strategy, $P_{evade} = \prod_i p_i \le \min_i p_i$. Either way the system's evadability is upper-bounded by its single hardest layer. This is the formal content of two familiar observations. Correcting a previously-failing feature yields no improvement once some *other* layer is the binding constraint, because the bound is fixed by the new minimum and not the old one — hence the absence of any "silver bullet." Symmetrically, a defender gains disproportionate value from adding even one layer whose p_i is near zero, since it drives $\min_i p_i$, and with it P_{evade}, toward zero irrespective of the others.

The bound also locates *where* that value comes from. The emit-reading transport layers are strongly correlated: a client built to reproduce one browser's network stack tends to pass them together, so they do not multiply as independent factors but collapse into a single effective term. Adding another correlated emit-layer barely moves $\min_i p_i$, whereas adding an *independent* possess-reading layer multiplies the product down. A defender's leverage therefore lies not in the sheer number of detectors but in their statistical independence and in lowering the floor $\min_i p_i$ — precisely the quantity an emitted artifact cannot lower and a possessed property can.

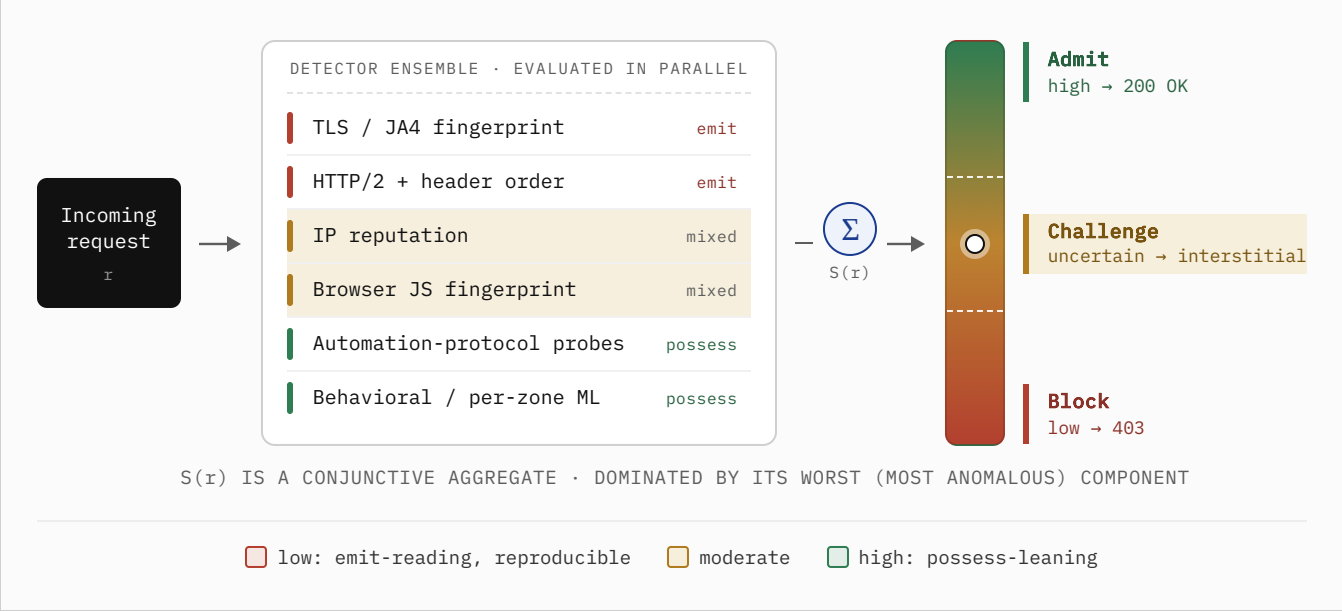

Fig. 1. The composite-scoring pipeline. Independent detectors are evaluated concurrently and reduced to a single trust score that selects among admit, challenge, and block. The conjunctive-pass property means an adversary must satisfy every active detector at once; a defender benefits most from adding a detector whose underlying property cannot be emitted by the client.

B. The Emit-Versus-Possess Distinction

We draw a distinction between two kinds of detector that, in our view, predicts evasion hardness better than the customary split between "network-level" and "browser-level" checks. A detector is **emit-reading** if it inspects an artifact the client transmits at connection or request time: the bytes of a TLS `ClientHello`, the ordering of HTTP/2 frames, the sequence of request headers. Such artifacts are, in principle, fully under adversary control: anything a genuine browser emits can be reproduced verbatim by a client engineered to do so, because the artifact carries no secret and no history. A detector is **possess-reading** if it interrogates a property the client must *continuously hold* and cannot fabricate on demand: accumulated, zone-specific behavioral history; a private key bound to a registered identity. These resist forgery not because the vendor hides how they work, but because reproducing them requires actually being the thing they test for.

C. Methodology and Scope

The taxonomy below synthesizes vendor documentation, the published fingerprinting literature, and the public behavior of widely available open-source tooling, cross-referenced against community measurement reports current to mid-2026. We restrict scope to one provider's stack to keep the analysis concrete. We describe each layer at the level of *mechanism and structural evadability*; we do not provide operational evasion code, working challenge solvers, or step-by-step circumvention procedures, both because they are unnecessary for the analytical claims and for the ethical reasons set out in Section VI. Tool

names appear only as they already appear in the public literature, and as objects of study rather than instructions.

III. The Detection Stack: A Layered Taxonomy

We group the fifteen layers into five domains, ordered roughly from the network edge inward toward the application and, finally, toward cryptographic identity. For each we state the operating mechanism, the signal read and, in defender's terms, what residual assurance it provides.

A. Network and Transport Signals

1) IP Reputation

Every source address is scored against global threat intelligence incorporating datacenter ASN membership, anonymity-network exit lists, residential-proxy pool reputation, and carrier-grade NAT classification.[1] The defensive strength here is asymmetric and economic rather than technical: datacenter ranges are cheaply and safely blockable because legitimate human traffic rarely originates there, whereas mobile carrier ranges are nearly unblockable precisely because a single CGNAT address multiplexes thousands of real subscribers, so any block inflicts collateral damage. Reputation therefore filters the cheapest infrastructure well and the most expensive infrastructure poorly. The layer is really a hybrid. An address is *emitted*, freely chosen by the client, but the reputation attached to it is a *possessed* property built up over time, something an adversary can rent but not fabricate. Its residual hardness is economic rather than structural, and is bounded only by how much the adversary is willing to pay for cleaner provenance.

2) TLS Fingerprinting (JA3 / JA4)

A TLS `ClientHello` exposes an ordered list of cipher suites, extensions, supported groups, and ALPN values whose exact composition is characteristic of the originating library and version. JA3 hashed a subset of these;[2] the now-standard JA4 family captures a richer, more evasion-resistant projection and is checked alongside its predecessor.[3] Generic HTTP libraries produce handshakes that resemble no mainstream browser and are separated from genuine clients deterministically, by a single fingerprint lookup. This is the canonical *emit-reading* layer: because the handshake carries no secret, a client engineered to reproduce a target browser's exact byte sequence becomes indistinguishable on this axis. Its defensive value is consequently bounded: it reliably catches naive clients and contributes nothing against a faithful impersonator.

3) Post-Quantum Key Shares and Encrypted ClientHello

Recent browser builds advertise the standardized hybrid post-quantum key share `X25519MLKEM768` (classical X25519 combined with ML-KEM-768, the FIPS-203 successor to the earlier Kyber draft) in the handshake, and increasingly negotiate Encrypted ClientHello, which conceals the destination name.[4] Both are folded into the fingerprint. For the defender this raises the *currency* bar: a client that claims to be a current browser yet omits the post-quantum key share now contradicts itself, creating a hard anomaly. But this is a moving target, not a structural gain: it penalizes stale impersonation profiles and is neutralized the moment the adversary updates to a current target. It remains emit-reading.

4) HTTP/2 and HTTP/3 Fingerprinting

At the application-transport layer, the ordering and parameterization of HTTP/2 control frames (settings values, window updates, the relative position of the first headers frame) are version-specific, as is the pseudo-header ordering, and QUIC adds its own transport-parameter signature.[5] Merely enabling HTTP/2 in a generic client does not help: the frames are sent with the wrong values in the wrong order. Yet, like TLS, the protocol exchange is an emitted artifact; a client that reproduces the exact frame choreography of a target build is indistinguishable here.

5) Header-Order Fingerprinting

Browser engines emit request headers in a deterministic, version-stable order, and the presence and position of fetch-metadata headers is part of that signature. A client that supplies every correct header *value* but in a different order still presents an anomalous sequence.[5] This is perhaps the purest illustration of the emit-reading category: the information being tested is structural and secret-free, and is therefore fully reproducible. We note that all four transport-and-header layers tend to be solved *together* by any client built to reproduce a specific browser at the network stack, which is why they collapse into a single defensive proposition in practice.

B. Client-Environment Signals

1) Browser and JavaScript Fingerprinting

When script executes on a challenge or protected page, the environment is interrogated for canvas and WebGL rendering signatures, audio-stack characteristics, font enumeration consistent with the claimed platform, the automation flag on the navigator object, permission-query behavior, plugin presence, screen geometry, and the timing resolution of the script engine.[6] A particularly consequential probe induces real-time-communication connectivity checks that can reveal a host's true address even behind a proxy, unless that subsystem is explicitly disabled.[6] This domain is harder to satisfy than the transport layers because it tests a *large, coupled surface*: a value corrected in isolation often contradicts another. Yet each individual probe still reads an emitted property of a controllable environment, so a sufficiently faithful instrumented browser can, with enough effort, present a coherent profile. The right label is emit-reading, but high-dimensional.

2) Automation-Protocol (CDP) Detection

Mainstream automation frameworks drive a browser through a debugging protocol that leaves low-level residue (characteristic property descriptors, evaluation and injection side-effects, internal target markers) persisting beneath surface patches.[7] This is the first client-side layer that begins to behave like a *possess-*

reading check: the residue is a property of *how the client is being driven*, not a single artifact, and the most effective open tooling addresses it only by replacing the control channel wholesale rather than masking individual tells. The literature is explicit that for the hardest configurations no freely available tool fully eliminates this surface,[7] which positions automation-protocol detection as a genuine partial barrier rather than a solved problem.

C. Behavioral and Statistical Signals

Per-zone anomaly models, trained separately for each protected site, score request-timing regularity, navigation and referrer structure, interaction events, session and cookie age, concurrency per source and, critically, the *consistency of all of the above across a session window*.[8] A single request is uninformative; the signal lives in the trajectory. This is the first unambiguously *possess-reading* domain. The property under test (a plausible, zone-specific history of having behaved like a returning human) cannot be emitted at request time; it can only be *accumulated*, at real cost in time and consistency, and it is per-zone, so history laundered on one site does not transfer. Per-zone training also makes the layer a moving and non-portable target, which is exactly why evasion success here is reported as inconsistent rather than solved.

D. Interactive Challenges and Scoring Products

When the aggregate is uncertain, the system interposes an active challenge. A managed JavaScript interstitial verifies correct execution and environment quality before issuing a short-lived clearance credential bound to the issuing address;[9] a successor interactive widget evaluates environment authenticity and interaction history before emitting a token, optionally escalating to a visible task.[10] Tiered products range from free heuristic modes through a manual "challenge everyone" posture to a paid machine-learning score that composes *all* preceding layers into one number.[1] The defensive significance of the address-bound clearance credential is underappreciated. By cryptographically tying the proof-of-solve to the address that produced it, the system turns a reusable token into one whose continued use forces address stability. This couples the challenge layer to the IP-reputation and behavioral layers and denies the adversary cheap address rotation. The composed paid score is the practical high-water mark of the emit-readable stack; the literature describes outcomes against a well-tuned configuration as inconsistent even for a maximal evasion stack.

E. Programmable and Cryptographic Controls

1) Edge-Programmable Detection

Operators can deploy arbitrary first-party logic at the edge: bespoke rate limits, honeypot endpoints reachable only by clients that fabricate links, custom challenge-response, and integration with first-party analytics.[11] By construction there is no generic characterization of this layer; its strength is that it is *non-portable and unknown a priori*, forcing per-site analysis and defeating tooling that relies on the standard stack being standard.

2) Labeled-Crawler Blocking

A one-click control blocks self-identifying automated agents by their declared identity.[1] Analytically this is a courtesy layer: it governs *honest* clients that announce themselves and is, by design, irrelevant to any client presenting as a browser. We include it for completeness as the limiting case of a purely declaration-based control.

3) Cryptographic Agent Attestation

The most consequential recent addition lets a site require connecting agents to present a cryptographically signed credential proving membership in a registered, authorized set.[12] Concretely, in the scheme proposed in May 2025, an agent signs selected components of each HTTP request — at minimum the `@authority` it is addressing, together with a short `created`/`expires` validity window that bounds replay — under a private Ed25519 key whose public counterpart is published at a well-known directory, following the HTTP Message Signatures standard (RFC 9421); the edge fetches the published key and validates the signature.[12] This is the first layer in the entire lineage that is *possess-reading by construction at the protocol level*: admission depends on holding a private key, and no fidelity of TLS, HTTP/2, header, fingerprint, or behavioral emulation substitutes for the secret. There is no forge-the-artifact path, because the artifact is a signature over the request rather than a reproducible client byte string.

One qualification is essential to an accurate reading, however. In current production form the control is deployed as a positive *identification* signal rather than a hard gate: a request bearing a valid signature is treated as an authenticated agent and is exempted from bot challenges, whereas a request bearing *no* signature is not thereby blocked but simply falls back to the rest of the stack. The cryptographic layer is therefore unforgeable yet, as presently configured, *fails open*; it becomes a true admission gate only where a site elects to mandate a valid signature. Adoption has nonetheless moved quickly along the participation axis: the scheme was folded into the provider's verified-bot program in mid-2025, several major agent operators now sign their traffic by default, and parallel adoptions by a competing CDN and by major payment networks followed through late 2025.

4) Identity-Gated Access

Finally, full authentication gates (login via an identity provider before any content is served) are not anti-bot controls at all but access controls; "bypassing" them without credentials is simply unauthorized access and is out of scope as a security-research object.

IV. Analysis: The Structure of the Evasion Surface

Collecting the layers by our classification yields the central result of the paper. Table I arranges all fifteen by domain and an-

notates each with its emit/possess character and the residual hardness reported in the current literature, and Fig. 2 plots the same fifteen along the emit-possess axis, where residual hardness rises by tier from left to right. The pattern that emerges is hard to miss, and we do not think it is accidental.

TABLE I

The Fifteen-Layer Stack Classified by the Emit-Versus-Possess Distinction

Layer	Domain	Reads	Residual hardness
IP reputation	Net / transport	hybrid (paid)	Economic
TLS / JA3 / JA4	Net / transport	emit	Low
Post-quantum TLS / ECH	Net / transport	emit	Low (currency)
HTTP/2 & HTTP/3 frames	Net / transport	emit	Low
Header order	Net / transport	emit	Low
Browser JS fingerprint	Client env.	emit (coupled)	Moderate
Automation-protocol (CDP)	Client env.	emit→possess	Partial barrier
Behavioral / per-zone ML	Behavioral	possess	High, non-portable
Managed challenge	Challenge / scoring	emit + bound	Moderate
Interactive widget (CAPTCHA)	Challenge / scoring	emit + bound	Moderate
Tiered heuristic modes	Challenge / scoring	emit	Low
Composed ML score	Challenge / scoring	emit + possess	High when tuned
Edge-programmable logic	Prog. / crypto	arbitrary	Per-site
Labeled-crawler block	Prog. / crypto	declaration	N/A (honest)
Cryptographic attestation	Prog. / crypto	possess	Unforgeable

Residual hardness reflects the assurance that *remains* against a best-effort, well-resourced adversary, not the difficulty of defeating a naive client.

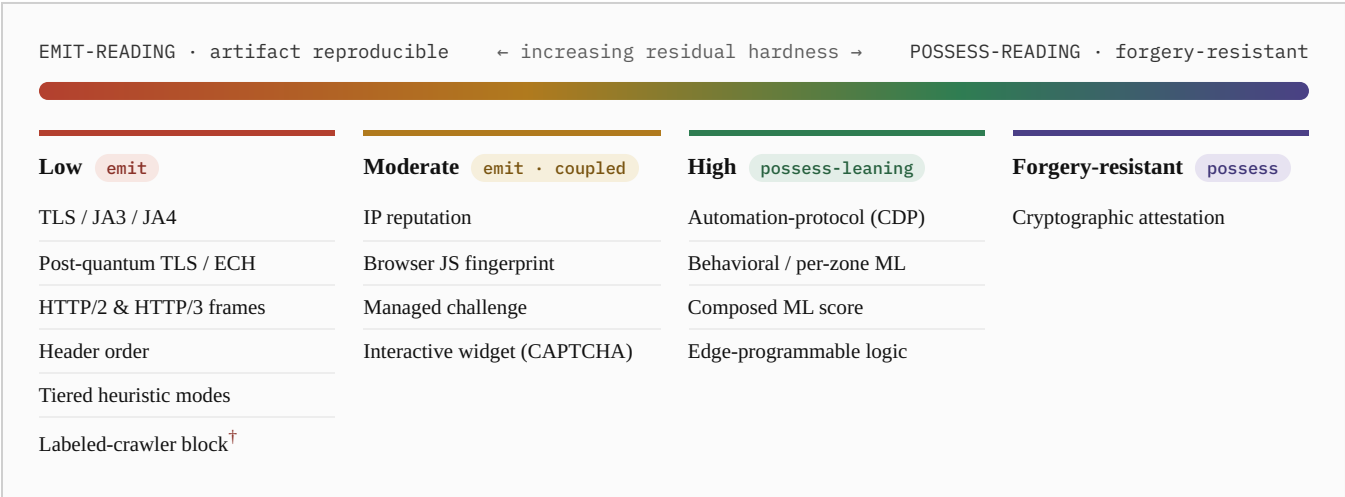

Fig. 2. The fifteen layers placed on the emit-possess axis. Residual hardness rises by tier from left to right: emit-reading artifacts are reproducible and therefore weak; possess-reading properties (genuine per-zone behavioral history and a privately held key) resist forgery by construction. †The labeled-crawler block governs only honest, self-identifying clients and is not applicable to browser-presenting traffic.

A. Why the Transport Layers Are Structurally Weak

The TLS, post-quantum, HTTP/2-3, and header-order layers share a single fatal property for the defender: each reads a secret-free artifact that the client controls in full. There is nothing to compute that a genuine browser would refuse to compute, no private input, no accumulated state, only a byte sequence to be reproduced. Consequently their defensive value is real but bounded and one-time: they raise the floor by eliminating clients that have not bothered to reproduce the artifact, and they contribute essentially nothing against a client that has. That they are routinely solved *as a bundle* by a single faithful network-stack implementation confirms the point; they are not four barriers but one, and a shallow one.

B. Why Behavioral and Cryptographic Layers Are Strong

The behavioral domain inverts every one of these properties. Its input is not an artifact but a *trajectory*; it is stateful, costly to accumulate, per-zone and therefore non-portable, and continuously re-evaluated for consistency. An adversary cannot emit a history; they can only invest in producing one, at a cost that scales with the realism demanded and that must be re-paid per site. This is why, unlike the emit-readable layers, the literature describes behavioral evasion as *calibration-dependent and inconsistent* rather than solved. Cryptographic attestation takes the same structural idea to its endpoint: it tests possession of a secret, and possession of a secret is exactly the thing that cannot be forged from observation of legitimate traffic.

> CENTRAL CLAIM
>
> Evasion hardness tracks the emit-versus-possess axis far more tightly than the network-versus-browser axis. Every layer that reads an emitted artifact is, in the limit, reproducible; every layer that reads a continuously possessed property (genuine per-zone history, a private key) resists forgery by construction. Durable defense comes from moving assurance onto the possess side of that line.

C. The Two Hard Limits

Two layers admit no evasion even in principle. Identity-gated access is not anti-bot at all; it is authentication, and circumventing it is unauthorized access rather than evasion. Cryptographic agent attestation is the genuinely novel case: it is the first anti-automation control in this lineage whose security rests on a private key rather than on the difficulty of imitation. It has two present limitations, and both are sociotechnical rather than cryptographic. Adoption is opt-in and still narrow, and current deployments fail open: an unsigned request is handed back to the emit-readable stack rather than refused outright. Neither limitation touches the underlying claim. Wherever a valid signature is genuinely required, no amount of imitation substitutes for the key. And the very existence of such a layer shows that everything analyzed above was always a contest of imitation fidelity — a contest that ends the moment a real secret enters the picture.

V. Defensive Implications

For practitioners building or evaluating defenses, the taxonomy yields concrete guidance that follows directly from the emit-versus-possess distinction.

Do not over-value transport fidelity. Investment in ever-finer TLS and HTTP/2 fingerprint discrimination yields rapidly diminishing returns, because a determined adversary reproduces the artifact exactly and the discriminator's contribution to the aggregate collapses to its false-positive cost. These layers are best understood as a cheap baseline filter, not a source of durable assurance.

Concentrate assurance on possess-reading layers. Per-zone behavioral modeling and, where appropriate, cryptographic attestation are the only layers analyzed here whose strength does not erode under faithful imitation. The address-binding of solve credentials is an instructive intermediate design: it manufactures a possess-like property (continued control of one address) out of an otherwise reusable token, and in doing so couples three layers together so they must be defeated jointly rather than serially.

Exploit non-portability deliberately. Per-zone models and edge-programmable logic derive their strength from being unknown and non-transferable. A defender should treat unpredictability as a first-class design goal: detectors that differ per site, rotate, or depend on first-party context impose per-target adaptation cost that no general tool amortizes.

Treat the conjunctive-pass property as leverage. Because the aggregate is dominated by its worst component, a single well-placed possess-reading layer raises the cost of the entire interaction. The defender's marginal return is highest not from hardening an already-strong emit layer but from introducing a new axis the adversary cannot emit at all.

The trajectory implied by this analysis is unambiguous. As imitation tooling continues to close the emit-readable gap, durable bot management migrates toward properties that cannot be emitted: verified behavioral continuity and, ultimately, cryptographic agent identity. The latter reframes the problem from “is this client imitating a browser well enough?” to “is this agent a registered party holding a valid key?”, a question to which imitation is simply not an answer.

VI. Limitations and Ethics

Limitations. Our account is necessarily a snapshot of a system that updates continuously; specific currency thresholds (such as which browser generation establishes the expected post-quantum signature) drift on a timescale of months. We characterize layers by mechanism and structural evadability rather than by measured admission rates, since the latter are zone- and time-specific and not reproducible across observers. The per-zone nature of the behavioral and programmable layers means any

single measurement generalizes poorly, a limitation we regard as itself a finding.

Ethics and scope. This work is framed for defenders and reviewers and is deliberately constructed to inform protection rather than to enable circumvention. We describe what each layer reads and why it is or is not evadable; we do not supply operational evasion software, working challenge solvers, credential-forgery techniques, or step-by-step circumvention procedures. We emphasize that circumventing a technical access-control measure can carry legal exposure under computer-misuse statutes in multiple jurisdictions independent of the public visibility of the target content, that terms-of-service and data-protection obligations attach to automated collection, and that where an authorized interface exists it is the correct channel. The analytical value of mapping an evasion surface (understanding where a defense's assurance genuinely comes from) does not require, and we do not provide, the means to exploit it.

VII. Conclusion

Modern bot mitigation is a composite, conjunctive system, and its security is spread unevenly across its layers in a way the usual network-versus-browser framing tends to hide. Sorting detectors by whether they read an emitted artifact or a possessed property predicts evasion hardness better than that framing does. The transport-fingerprint layers that attract most practitioner attention are the structurally shallow ones; what durable assurance the stack has lives in its behavioral and, more recently, cryptographic layers, which test for properties that cannot be synthesized by watching legitimate traffic. Cryptographic agent attestation is the inflection point, the place where a defender stops competing on how convincingly a client imitates a browser and starts relying on a secret instead. For defenders the lesson is to spend the assurance budget on the possess side of that line. For the field, it suggests that bot management is drifting, slowly, away from fingerprinting and toward verifiable identity.

References

[1] Cloudflare, "*Cloudflare Bots documentation: Bot Management, Bot Fight Mode, and Super Bot Fight Mode*," Cloudflare Developer Docs. [Online]. Available: https://developers.cloudflare.com/bots/ (accessed 2026).

[2] J. B. Althouse, J. Atkinson, and J. Atkins, "*TLS fingerprinting with JA3 and JA3S*," Salesforce Engineering Blog, 2017. [Online]. Available: https://github.com/salesforce/ja3 (BSD 3-Clause).

[3] J. Althouse and FoxIO, "*JA4+ network fingerprinting (JA4/S/H/L/X/SSH)*," FoxIO, 2023. [Online]. Available: https://github.com/FoxIO-LLC/ja4.

[4] D. Stebila, S. Fluhrer, and S. Gueron, "*Hybrid key exchange in TLS 1.3*," IETF Internet-Draft `draft-ietf-tls-hybrid-design`; and Chrome Security Team, Google, "Advancing to the standardized post-quantum key share `X25519MLKEM768` (TLS group `0x11EC`), shipped by default in Chrome 131 (Nov. 2024) and superseding the earlier `X25519Kyber768Draft00` experiment," 2024. ML-KEM standardized as NIST FIPS 203. Encrypted ClientHello: IETF `draft-ietf-tls-esni`.

[5] Akamai Technologies, "*HTTP/2 client fingerprinting*," Akamai SIRT technical report, 2020; and FoxIO, "JA4H: HTTP request fingerprinting," 2023 (see [3]).

[6] P. Eckersley, "*How unique is your web browser?*" in *Proc. 10th Privacy Enhancing Technologies Symp. (PETS)*, LNCS 6205, pp. 1–18, 2010; with subsequent canvas, WebGL, AudioContext, and WebRTC fingerprinting literature, 2012–2025.

[7] Google, "*Chrome DevTools Protocol*." [Online]. Available: https://chromedevtools.github.io/devtools-protocol/; with community analyses of automation-protocol detection, 2022–2026.

[8] Cloudflare, "*Bot management scoring with machine learning and per-zone behavioral modeling*," Cloudflare Blog and Learning Center, 2020–2025.

[9] Cloudflare, "*Cloudflare Challenges (Managed Challenge); the `cf_clearance` and `__cf_bm` cookies*," Cloudflare Developer Docs, 2022–2026.

[10] Cloudflare, "*Cloudflare Turnstile: a user-friendly, privacy-preserving alternative to CAPTCHA*," Cloudflare Blog and Docs, 2022.

[11] Cloudflare, "*Cloudflare Workers: programmable edge runtime*," Cloudflare Developer Docs, 2018–2026.

[12] Cloudflare, "*Forget IPs: using cryptography to verify bot and agent traffic (Web Bot Auth)*," Cloudflare Blog, May 2025; and "*Message Signatures are now part of our Verified Bots Program*," Cloudflare Blog, 2025. Built on HTTP Message Signatures, IETF RFC 9421 (2024), with Ed25519 keys and `draft-meunier-web-bot-auth-architecture-02`; reference implementation: https://github.com/cloudflareresearch/web-bot-auth.