Attack the Glass (ATG): FAQs

Frequently Asked Questions for the Attack the Glass (ATG) Vulnerability

v2.4.7, September 14, 2026

TLP:CLEAR

Contact: Robert Beckett

Signal: ndiligence.99

Email: atg@nDiligence.com

About this Document

This document is the discipline anchor for anyone speaking about ATG. It states what nDiligence claims and what it does not, and it holds the line on three failure modes: overclaiming what is known, letting the finding be dismissed as a rename of something familiar, and turning the disclosure record into a story about government failure. Answers here are the answers every outlet gets. They do not change under deadline pressure.

The finding

What is Attack the Glass, in one line?

A client-side supply-chain vulnerability that lets whoever controls one component of a modern application change what a screen shows or what the application does, through a source the person and the application both already trust, with no visible difference and no forensic trail.

What is "the Glass"?

Any surface where a runtime renders for a human. A browser window is the common case, not the definition. A mobile app, a desktop application, a vehicle display, and an industrial operator console are the same thing under a different housing, and they are built the same way. The distinction matters for what an attacker can do: a browser page is mostly there to be read, so deception dominates there, while a control or appliance interface is a control layer, and interfering with what a function does becomes the wider and more consequential effect.

Is this a new vulnerability, or a rebranding of known supply-chain risk?

It is a new framing, and it is broader than the incidents people know.

Magecart and Polyfill.io are specific instances. ATG is the general property that makes them possible: modern screens are assembled at runtime from components granted trust by address rather than by content, and the manipulation can happen after every security check has passed. The contribution is naming the class, mapping where it operates across the full runtime landscape rather than the browser alone, and identifying the Integrity Blindspot, which is that nothing verifies at execution that a resource is what it should be, and nothing is recorded afterward that would let a substitution be reconstructed.

nDiligence does not claim to have discovered an unprecedented exploit. The parts are individually familiar. Client-side supply-chain security addresses third-party script risk. Magecart has skimmed data from the rendering layer for a decade. Cloaking, meaning serving one response to a scanner and another to a victim, is documented behavior: the Polyfill.io payload filtered on request headers, checked device type a second time in the browser, and gated further on referrer and local device time, sampling sessions to stay harder to find. What is new is one account that covers all of it and extends it past the browser.

What can an attacker actually do?

Five things, which the research calls the D5 effects. They are not a ramp and not a severity order; each is fully capable on its own, and nothing prevents an attacker combining them.

Deceive. False content is accepted as authoritative and acted on. The consumer can be a person whose decision forms on a false render, or an automated one: an alarm evaluation, a control loop, an agent, or a model. A machine does not judge whether its input is true, which is exactly why feeding it false data works.

Disrupt. An intended function is present and does not do what it is for. A password reset that completes and sends nothing. An alarm control that fires and reaches no one.

Degrade. Volume, speed, and scale are reduced. Everything works, and works worse.

Deny. Something is unreachable at the rendering surface. The control is gone, the route does not resolve, the returned set is truncated.

Destroy. Adversary-controlled code reaches through the rendering surface into a backend or connected system and damages it. Deny acts on the glass; Destroy goes through it.

Which effect it is and who receives it are separate questions. The range runs from every user of an application, through a country or language community, through a segment defined by behavior or demographics, down to one named individual on one device, while everyone outside the selection receives the genuine version.

The bound is worth stating in the same breath, because it is the honest limit. ATG grants an adversary nothing the application could not already do; it substitutes what the application is working from. The highest effect available is set by the authority of the session rather than by the attacker, so the same swap reaches a customer's belief on a checkout page and reaches the directory on an identity console. A read-only reference site affords almost none of the five. A database console affords all of them.

Is this about targeting individuals?

Not mainly, and the emphasis matters. Targeting one named person on one device is the precision extreme. It proves how fine the aim can be, and an operation that wanted reach would not use it.

The realistic use is segmentation. One position can serve different content to different groups at the same moment, from the same address: the real article to most readers, one alteration to readers in a particular country, a different alteration to readers whose prior behavior places them in a political or commercial segment, and the real article again to any requester that looks like a scanner, a fact-checker, or a newsroom. No group sees another group's version.

The selection rules are the ones any content-personalization system already uses: source address and inferred geography, device characteristics, time and session timing, authenticated identity, prior behavior, cohort membership, and experiment buckets. A segment is no harder to target than a person, because the machinery was built to target segments.

That is why this matters at population scale. Reporting it as a way to fool one individual describes the narrowest thing it does. A single target is still not the trivial case. What one recipient can decide sets the ceiling as surely as how many recipients there are. A head of government, a bank's chief executive, and a commander directing a force each read from one screen, and an operation that reaches only that screen has reached everything it needed. Reporting that treats one-person targeting as the small case gets it as wrong as reporting that treats it as the headline.

If the user trusts the source, isn't that on them?

The user never made the choice. The application author decides which third parties the device will trust and encodes that decision in what it ships; the device carries it out. The person at the screen is not shown the list, is not asked to agree to it, cannot inspect what any entry on it returns, and cannot decline one without losing the application.

nDiligence calls this Conscripted Trust: the user does not choose whom to trust, the application chooses, and the choice runs on the user's device.

We state the limit precisely, because a reviewer will reach for an ad blocker otherwise. Extensions, script blockers, enterprise policy, and DNS filtering all exist and all work. What none of them can do is inspect what a permitted provider actually returned, or accept a provider on conditions. The client can refuse the relationship. It cannot condition it.

Who is affected?

Anyone who publishes to a screen or relies on one. News, finance, healthcare, government, public safety, and critical infrastructure, across browsers, mobile applications, desktop shells, appliances, IoT, vehicle systems, operational-technology and control systems, and AR and VR. Because closed and classified networks are built from the same upstream components, network isolation reduces the size of the supply chain rather than removing it. Those environments are exposed differently rather than exempt, and the difference is real: the supply chain inside a controlled network is smaller and better governed than the open internet. We hold this as reasoned rather than established, because no incident in the public record documents an ATG-class compromise inside an isolated environment and we do not imply one.

What we know, and what we do not

Are you saying this is being actively exploited right now?

We cannot say for certain, and that limit is central to why it matters.

The attack is built to leave nothing behind: no malicious file to scan, no fraudulent website to block, and no record in the logs. With the tools in standard use there is no reliable way to confirm that a campaign is running or to rule one out. The absence of visible evidence is not a sign that nothing is happening. It is a direct result of how the attack works.

We are careful about what we assert, and we use four registers consistently. The mechanism is **established**, and so are the documented incidents that have used it. The severity scaling with the screen is **reasoned**, meaning it follows from stated premises rather than from a documented case. That capable actors are positioned to use this is **presumed**, meaning we hold it as a working assumption for operational reasons and mark it as one. The true scale of any operational use today is **unknown**, and cannot be measured with the instrumentation available.

We are not claiming, in this public material, that a specific operation is underway.

What we will say is that this is not a theoretical exercise. We are raising it now because our threat-intelligence work gives us specific reason for concern: it points to capable actors positioned in the supply chain at the point where such a campaign would run. Those findings, including any named entities, are handled through restricted channels and are not part of this public release. A demonstrated capability, no reliable way to detect its use, and actors positioned where they would need to be positioned is the combination that leads us to treat this as urgent rather than hypothetical.

If it leaves no trace, how would anyone ever know?

Today they would not, and that is the finding rather than a caveat to it.

Runtime behavioral monitoring would change it: watching how content behaves as the screen is drawn, rather than checking resources before they arrive. So would telemetry at population scale, which is what lets a response served to one person be compared against what everyone else received. So would visibility reaching past the browser into the other runtimes, where today there is almost none. None of the three ships as a default in any deployed runtime. That is an engineering problem, not a mystery. Naming what is missing is more useful than restating the unknown.

Isn't "absence of evidence is not evidence of absence" just an unfalsifiable claim?

It would be, if we used it to assert that something is happening. We do not.

The distinction we hold is between two statements that sound similar and are not: "we have not detected this" and "this is not happening." Our claim is the first. We say the mechanism is real and demonstrated, that detection of its use is not currently possible, and that we therefore cannot tell you either way. The falsifiable part is the mechanism, and we invite you to test it. The unknown part is marked unknown and carries no weight in what we claim.

If someone builds the runtime monitoring described above and finds nothing, that is a real result and we will say so.

The technical objections

These are the questions a security specialist asks, and the honest answers are more useful than confident ones.

Isn't this just cross-site scripting with a new name?

No. Cross-site scripting exploits a defect in a specific application, one that the application's owner can fix. ATG requires no defect anywhere. Every component behaves exactly as designed. The application correctly fetches from an address it was configured to trust, and the runtime correctly executes what that address returned. There is nothing to patch in the application, because the application is not broken.

Doesn't HTTPS solve this?

No, and the reason is the center of the whole finding. Transport security establishes that a response genuinely came from the claimed origin and was not altered in transit. It establishes nothing about what that origin chose to send. If the origin is under adversary control, or has changed hands, transport security faithfully protects the delivery of the altered content.

Doesn't Subresource Integrity solve this?

It closes part of it, and it is worth deploying.

SRI validates a fetched resource against a hash before the runtime executes it. Three things limit it. Its declarative form covers `<script>` and `<link>` elements only. Anything an already-executing script subsequently fetches is verified only if a developer passes integrity metadata on that specific call, which nothing requires and nothing audits. And a hash cannot cover a resource whose content legitimately varies per request, which is a large share of what modern applications load.

The deeper limit is the two-stage form. A script that is entirely legitimate and verifiably unmodified can carry an attack through the data it fetches while running. The file really is clean. The weapon is in the data, not the code.

We want to be precise about one thing, because it is often overstated on our behalf. Nothing in the SRI specification says that verification "ends at load and never resumes." That phrase is our characterization of where the check sits in the algorithm. What the specification actually does is define integrity checking as a step performed on a response during fetching, and define no mechanism for re-verifying that resource once the response has been consumed.

Doesn't Content-Security-Policy solve this?

It helps, and it is the most useful control available in the browser today. It also has a boundary.

In its ordinary deployed form, CSP is an origin allowlist: it controls which locations a document may load from. An allowlist approves the sender, not the package. If an allowed origin returns altered content, CSP has done its job and the altered content runs.

The exception is worth stating rather than hiding, because a reviewer will find it. CSP Level 3 allows hash source expressions to match external scripts where the script element also carries matching integrity metadata. That is genuine content pinning. It applies only to script and style elements, requires SRI metadata alongside it, and operates at load time.

Doesn't server-side rendering solve this?

It reduces the surface, which is a real benefit, and it does not eliminate it. A server-rendered page still loads third-party scripts, analytics, advertising, payment components, and identity providers into the client, and those still execute with the page's permissions. What server-side rendering removes is the client's role in assembling the primary content. What remains is everything else on the page.

What about signature-based Subresource Integrity?

We would rather name it than have you find it.

Signature-based SRI is a live draft in the W3C Web Platform Incubator Community Group, with implementation intent in Chromium, and it is aimed squarely at the case classic SRI cannot handle: resources whose content legitimately varies. Instead of pinning content, it proves provenance, so a script executes only if it is signed with a known key. It is real work and it is worth supporting.

It does not answer ATG. Provenance at load is still a load-time property. A correctly signed script that fetches adversary-controlled data while it runs passes a signature check exactly as it passes a hash check. The gap the two-stage form uses is not about establishing who sent the resource.

Zero Trust is supposed to assume breach. Doesn't it cover this?

Zero Trust does what it was designed to do, and the rendering layer is outside that design.

NIST's Zero Trust guidance is about access: authenticating and authorizing a subject's access to a resource, dynamically, per session, and, in its own words, "before access is allowed." It states seven tenets and is guidance rather than a standard with a conformance

checklist. Every one of those tenets concerns access decisions or the posture of enterprise-owned assets. None concerns the integrity of content after that content has been delivered to a client and rendered.

The tenet people raise is the one about monitoring "the integrity and security posture of all owned and associated assets." Its object is assets the enterprise owns, in service of the access decision. A third-party script rendering in a member of the public's browser is neither.

We put this carefully: the guidance does not extend to the rendering layer. It does not declare the rendering layer out of scope, and we do not say it does.

Aren't defenses outside the browser better than you say?

Some are, and we have corrected our own language on this.

An earlier formulation of our work implied that non-browser runtimes lack the browser's defensive primitives, and named desktop application shells built on web technology, Electron among them, along with mobile WebViews. That is wrong for both. Electron's own security documentation recommends Content-Security-Policy and documents how to deliver it. Mobile WebViews render documents using the same engines as their parent browsers and inherit CSP enforcement.

The accurate version is scoped by the specification itself. CSP attaches to a document. Runtimes that render no document, such as Node.js and React Native's JavaScript engines, have nothing for it to attach to. Operational-technology displays, vehicle infotainment, and appliance interfaces vary by implementation, and where they embed a browser engine the primitives are available whether or not anyone configured them.

What survives that correction is the load-bearing part. Availability is not deployment, deployment is uneven, and none of these controls reaches past the moment of load.

Why doesn't this have a CVE?

Because a CVE identifies an instance of a weakness in a product, and ATG is not in a product.

The CVE program's own rules define a vulnerability as weaknesses in a product, require an assigning authority with scope over that product, and assign identifiers per independently fixable vulnerability. An architectural pattern spanning thousands of vendors' products has no single product, no authority whose scope covers it, and no single fix.

The right MITRE construct for a cross-product weakness class is CWE, which enumerates classes of weakness, rather than CVE, which enumerates instances in products. ATG is

CWE-shaped and has been measured with a CVE-shaped instrument. We state this as reasoning from the published rules, not as a rule that says so, because no such rule exists.

Where does it fit in MITRE ATT&CK?

It touches several techniques and is fully described by none. Supply Chain Compromise and its sub-techniques cover manipulation "prior to receipt by a final consumer," which is pre-delivery by definition. Drive-by Compromise covers script that arrives malicious at load. Content Injection covers substitution in the network path. Web Portal Capture covers modification of a server-hosted page.

As of our check on August 24, 2026, the catalog contains no technique describing an adversary altering what a client renders after that content has passed integrity verification.

We are careful about what that means. ATT&CK is a catalog of observed adversary behavior and has never claimed to be exhaustive. "No technique exists for this" is a true statement about the catalog on a given date. It is not proof that the behavior is unobserved, and we do not present it as one.

Response and defense

Is there a patch? What should organizations do?

There is no single patch, because ATG is a property of how software is assembled rather than a flaw in one product. The constructive steps are three. Inventory transitive dependencies, meaning your providers' providers, which is the lesson the Cyber Safety Review Board drew from Log4j when it reported that, of the organizations it spoke to that were using software bills of materials, none reported having used them to identify vulnerable deployments. Apply the runtime-integrity controls your runtime supports, meaning Subresource Integrity, a strict Content-Security-Policy, and self-hosting of critical assets, and verify they are configured rather than merely available. And extend assurance thinking past the moment of load to what executes afterward, including the runtime data that clean, verified code fetches while running.

If there is no fix, why publish?

Because awareness is the mitigation available today, and it is not a small one. Publishers, platform owners, and defenders cannot manage an exposure they have never heard of. Polyfill.io is the precedent: the research went public on June 25, 2024, a major CDN began rewriting links to a safe mirror the next day, and the domain was gone by June 27. Inventory decisions, procurement questions, and architectural choices all change once someone knows to ask about the rendering layer.

Isn't publishing this a roadmap for attackers?

The mechanism is not secret and never was. It rests on documented, standard behavior of the web platform, and it has been used in the wild. The people who would run this already understand it. The people who cannot currently see it are the defenders, and they are the ones the publication is for. The materials contain no exploit code. We name actors only where public attribution already exists in the record. Our restricted attribution work stays restricted.

What do you actually want to happen?

Three things. That platform and application owners look at the rendering layer as an attack surface and inventory what their applications load at runtime. That the engineering work on runtime behavioral monitoring gets funded and built, because it does not exist as a default anywhere today. And that procurement and standards frameworks start asking about the layer that none of them currently reaches.

The disclosure record

Did you tell the government? What did they do?

We followed responsible disclosure. U.S. Government briefings began in April 2024, before we said anything publicly, and extending to national and multinational cyber authorities through 2026. Each briefed body received formal notice of this publication in August 2026, thirty-five days ahead of it.

This is not a story about any agency's response, and we do not characterize how any of them handled the briefings. It is a structural problem with no single owner and no single patch, which is exactly why the public and defenders need to understand it.

If you want the notification dates for your own record, ask and we will provide them as fact and leave the interpretation to you.

Why did it take two years?

Because that is what responsible disclosure looks like when there is no vendor to patch and no single owner to fix it.

The ordinary disclosure clock assumes someone can ship a fix. Here there is no such party. What we could do instead was take it to the bodies whose job is systemic risk, give them the time to work it, and then publish when it became clear that the remaining useful step was telling the people at risk. That judgment is ours and we own it.

You reference a specific actor or provider in some material. Who is it?

We make no original attribution, which is a narrower and more accurate statement than "we name no one." Publicly documented operations are cited by name, and any actor we name is named as the public record already names them: Funnull in the Polyfill.io case, sanctioned by the US Treasury in 2025, and the Chinese state as the Citizen Lab attributed the Great Cannon in 2015. What we will not do is name a party as responsible for an operation that has not already been publicly reported. Our own attribution work is handled separately, under restricted handling and independent legal and technical review, and is not part of this release, which concerns the vulnerability only. We will not confirm specifics outside the appropriate channels. The ATG intelligence annexes and intelligence notes are separate products and remain restricted.

The claim and the language

Isn't calling this "weapons of mass deception" overstating it?

That phrase describes a specific upper-bound scenario: ATG paired with generative AI to tailor deception per target at scale. We use it to name the stakes, not to claim the scenario is occurring.

The everyday version, that you do not control your own front page, is demonstrable today and is the better place to start. If a reporter wants one framing, we would rather it be that one.

How was this researched, and can it be verified independently?

Yes, and we would rather you did.

One hundred percent of the research is nDiligence-internal, self-funded, and open-source, drawn from publicly available information. No government, vendor, platform, or investor funded it or reviewed it before publication. A provenance and methodology appendix maps every material claim to its public source, files each claim against one of the four registers, and states plainly what we could not verify and where we narrowed our own claims during review. The appendix names two exceptions. Our reading of where capable actors sit rests on restricted work you cannot check; it is filed as presumed, and no other claim depends on it. The record of whom we briefed and when rests on correspondence we hold rather than a public source, and we will give you those dates as fact. Take it to any expert you choose. We will not ask who you consulted.

The core architectural claim can be checked in about ten minutes without our help. Open the developer tools in any browser, load a major news, banking, or government site, and read the network tab. Count the distinct third-party origins the page fetches executable code from, then check how many of those script tags carry an integrity attribute.

Can you demonstrate it?

Yes. nDiligence will walk any reporter, or any expert a reporter engages, through the mechanism on request, using benign content on infrastructure we control.

The demonstration is that the same address returns different content to different people at the same moment. A fact-checker examining the source sees the genuine version. The targeted individual, at the identical address, at the identical moment, sees the altered one. Neither can see the other's version, and the publisher's own systems show nothing unusual, because the publisher's servers were never touched.

We will not demonstrate against any live third-party property, and the published materials contain no exploit code. The point is to show that selective delivery is real and undetectable from outside, not to hand anyone a working attack.

Are you selling something?

nDiligence is a diligence, security, and threat intelligence firm, and it does have commercial interests in this field. The ATG research was self-funded, and no vendor named in these materials has a commercial relationship with us. We do offer a platform, SCVue, that our team built for the ATG research and later surfaced; it can reveal ATG-class exposure, it has a free community edition, and its paid tiers cover real costs such as storage, data enrichment, and processing. SCVue predates this disclosure and is not being launched alongside it, and the defensive direction described in the technical materials is an engineering direction rather than a product we are selling. Weigh the findings on the sources, which is why the provenance appendix exists.

Where can I get the materials?

The technical whitepaper, the technical and government primers, the executive summary, this FAQ, and the provenance and methodology appendix are at **atg.nDiligence.com**.