

Arc Technologies

React2Shell (CVE-2025-55182)

Zero Breaches, 631 Would-Be Intruders: Lessons From 61 Days of React2Shell Attacks

Prepared by: Mathew Ibbetson, Junior Security Analyst, Arc Technologies

Date: October 15, 2026

Observation window: August 13, 2026 to October 13, 2026 (62 days)

Data source: Arc Technologies web application intrusion detection, 297 investigated incidents

Contents

1. Overview
 2. Key Findings
 3. Vulnerability Background
 4. How the Exploit Works
 5. Activity Observed Against Arc
 6. Attack Patterns and Payload Families
 7. What Attackers Are After
 8. Impact Study: If Arc Had Been Vulnerable
 9. Why Arc Was Not Affected
 10. Lessons for Defenders
 11. Recommendations
- Appendix A. Indicators
- Appendix B. Detection Markers
- Appendix C. Glossary
- Appendix D. Sources

1. Overview

React2Shell (CVE-2025-55182) is an unauthenticated remote code execution flaw in React Server Components, rated CVSS 10.0 and disclosed on December 3, 2025. A single crafted HTTP POST can make a vulnerable Next.js or other RSC-based server run attacker-supplied JavaScript.

Arc Technologies is a Canadian managed hosting and cybersecurity provider serving non-profit, community and public-sector organizations. Between August 13 and October 13, 2026, our detection recorded 1,440 React2Shell exploitation requests against 46 hostnames belonging to 36 client projects. The requests came from 597 unique source IPs. Arc's hosted applications do not use React Server Components, so none of the payloads could execute.

Because nothing ran, it would be easy to file this activity as noise. This study does the opposite. We decoded every payload, identified eleven distinct attack families, and then asked what each request would have achieved had it reached a vulnerable application. The answer, set out in Section 8, is that every host we operate would have been compromised, most by more than a dozen separate attackers, and that the payloads aimed at us were built to steal exactly the credentials that unlock a modern Kubernetes and cloud hosting environment.

All client names, hostnames and internal identifiers in this report and its companion data have been replaced with neutral labels. Attacker infrastructure is reported in full.

2. Key Findings

Metric	Value
React2Shell requests recorded	1,440
Investigated incidents containing React2Shell	297
Days with activity	58 of 62
Unique source IPs	597
Source IPs seen in only one incident	568 (95.1%)
Source IPs in Google Cloud address ranges	481 (80.6%)
Hostnames targeted	46 hostnames across 36 client projects
Requests carrying an authenticated session	0
Distinct payload families	11
Separate attacker and host pairs that would have achieved code execution	631
Hosts that received host profiling or secret theft payloads	18 of 46 (39%)
Successful executions against Arc	0

- **Every host was found, quickly and repeatedly.** All 46 hostnames were probed, including preview and development environments. Once scanning reached steady state, a new attacker arrived at each host every 37 hours (median). A vulnerable application would not have stayed undiscovered for long.
- **A vulnerable host would have had many intruders, not one.** On average 13.7 separate attackers would have run code on each host. The most targeted host would have been compromised by 33 different parties.

- **The payloads were built to steal cloud and Kubernetes credentials.** One harvester reads the pod service account token and asks the Kubernetes API for every Secret and ConfigMap in the namespace. Others take every environment variable, .env file, AWS key, SSH key and git credential they can reach.
- **Victims are sorted for profit.** A hex-encoded loader profiles each host for cryptomining (it checks for AVX2 CPU support) and flags "high value targets" holding cloud keys, crypto wallets or transactional email keys.
- **Volume is falling but capability is rising.** Weekly requests peaked at 371 in early September and are now under 80, while the share of credential-theft payloads has grown. The last two days of the window carried 26 secret-harvesting requests.
- **Massive attacker Infrastructure.** 95.1% of source IPs were used once and never again, almost all rented cloud instances.
- **The financial stakes are real.** The average Canadian data breach cost CA\$6.98 million in 2025 (IBM). Personal information held by our clients would trigger federal or provincial breach notification duties.

3. Vulnerability Background

3.1 What React2Shell Is

React Server Components (RSC) let a browser call server-side functions ("Server Functions" or "Server Actions" in Next.js) by sending data in React's "Flight" serialization format. The server deserializes that data in `ReactFlightReplyServer.js` before any application code or authentication check runs.

The flaw is insecure deserialization. When reviving objects, React called `hasOwnProperty` on attacker-controlled objects and followed `$`-prefixed references through property paths without restricting them to the object's own properties. An attacker can walk from a chunk object up through `__proto__` and constructor to reach JavaScript's `Function` constructor, then have React call it with attacker-supplied source code. The fix calls `Object.prototype.hasOwnProperty` safely and blocks `__proto__` traversal.

- Pre-authentication: no login, session or valid action ID is needed. Next.js reaches the vulnerable code for any Next-Action header value.
- Default configurations are exposed: an app only needs to support RSC (for example, any Next.js App Router app), not define its own Server Functions.
- Code runs with the full privileges of the Node.js process, including its environment variables, file system, network access and any mounted cloud or Kubernetes credentials.
- React tracks the flaw as CVE-2025-55182. Next.js tracked its downstream exposure separately as CVE-2025-66478.

3.2 Affected and Fixed Versions

Component	Affected	Minimum fix for RCE	Recommended (includes later DoS and source exposure fixes)
react-server-dom-webpack react-server-dom-parcel react-server-dom-turbopack	19.0.0, 19.1.0, 19.1.1, 19.2.0	19.0.1, 19.1.2, 19.2.1	19.0.4, 19.1.5, 19.2.4 or later
Next.js (App Router)	15.x, 16.x, 14.3.0-canary.77 and later canaries	15.0.5, 15.1.9, 15.2.6, 15.3.6, 15.4.8, 15.5.7, 16.0.7	14.2.35, 15.0.7, 15.1.11, 15.2.8, 15.3.8, 15.4.10, 15.5.9, 16.0.10 or the latest patch in the line
Other RSC frameworks	react-router, waku, @parcel/rsc, @vitejs/plugin-rsc, rwsdk	Per vendor	Per vendor
Not affected	Next.js 13.x and 14.x stable, Pages Router apps, Edge Runtime, React apps with no server		

There is no configuration workaround. Vendors state that WAF rules are a stopgap and that upgrading is required. Next.js advises rotating all application secrets for any app that was online and unpatched as of December 4, 2025.

3.3 Timeline

Date	Event
November 29, 2025	Lachlan Davidson reports the flaw through the Meta bug bounty program
December 3, 2025	Patches published to npm; CVE-2025-55182 and CVE-2025-66478 disclosed
December 4, 2025	AWS observes exploitation attempts within hours of disclosure. Shadowserver later counts about 77,664 exposed IPs
December 6, 2025	CISA adds CVE-2025-55182 to the Known Exploited Vulnerabilities catalogue
December 11, 2025	Follow-on CVEs disclosed: CVE-2025-55184 (DoS), CVE-2025-55183 (source code exposure), CVE-2025-67779 (complete DoS fix)
January 26, 2026	Further DoS issue disclosed (CVE-2026-23864)
August 13, 2026	First React2Shell request recorded against Arc-hosted applications
October 13, 2026	Last request in this dataset; activity is ongoing

3.4 Known Threat Activity

Public reporting attributes early exploitation to China-nexus groups including Earth Lamia, Jackpot Panda and UNC5174, with more than 30 organizations confirmed compromised in the first days. Post-exploitation tooling reported by Trend Micro and others includes Cobalt Strike and Sliver implants, the Snowlight dropper and Vshell backdoor, Mirai-variant botnets, XMRig and Kinsing cryptominers, Nezha and FRP remote access tools, and a "Secret-Hunter" Node.js payload that harvests cloud and repository credentials.

Ten months after disclosure, the activity we see is a long tail of automated scanning for unpatched servers, now mixed with more capable credential-theft payloads. A vulnerability does not stop being exploited when it stops being news.

4. How the Exploit Works

4.1 Anatomy of an Observed Request

The most common payload in our data (1,022 requests) is a multipart POST with three form fields. Decoded, it reads:

```
field 0: {
  "then": "$1: __proto__:then",
  "status": "resolved_model",
  "reason": -1,
  "value": "{\\"then\\":\\"$B1337\\"}",
  "_response": {
    "_prefix": "var res=process.mainModule.require('child_process')
      .execSync('echo $((41*271)) | base64 -w 0').toString().trim();
      throw Object.assign(new Error('NEXT_REDIRECT'),
        {digest:`NEXT_REDIRECT;push;/login?a=${res};307;`});",
    "_chunks": "$Q2",
    "_formData": { "get": "$1:constructor:constructor" }
  }
}
field 1: "$@@"
field 2: []
```

4.2 Step by Step

1. **Self-reference.** Field 1 (\$@@) is a raw reference back to chunk 0, giving the resolver a real internal chunk object to walk from.
2. **Fake chunk.** "then": "\$1: __proto__:then" borrows the real chunk prototype's then method, so React treats the forged object as a promise-like chunk. "status": "resolved_model" tells React it is ready to be parsed.
3. **Hijacked response object.** The forged _response replaces React's internal response state. Its _formData.get is set to \$1:constructor:constructor, which walks object to Object to Function.
4. **Code injection.** The blob reference \$B1337 makes React call _formData.get(_prefix + "1337"), which is now Function("<attacker code>1337").
5. **Execution.** The resulting function becomes the then method of a value React resolves as a promise, so the attacker's JavaScript runs inside the Node.js process.
6. **Output return.** The payload throws an error whose digest is NEXT_REDIRECT;push;/login?a=<output>;307;. Next.js treats this as a redirect and returns the attacker-chosen location, carrying the command output, in the HTTP response. No outbound callback is needed.

4.3 Request Shapes Observed

Scanners usually send each payload twice in quick succession: once as a plain three-field array (729 requests) and once as an object whose first key is 1_\$ACTION_ID_vercel (526 requests), which mimics the hidden field Next.js adds to progressively enhanced form actions. For the dominant canary family, 499 of 509 request pairs from the same IP arrived within one second of each other.

The canary value is chosen so success is unambiguous: 41*271 is 11111, and echo 11111 | base64 returns MTExMTEK (verified). All 1,440 requests targeted the site root and none carried a session, consistent with a flaw that needs neither a specific route nor authentication.

5. Activity Observed Against Arc

5.1 Dataset, Method and Anonymization

Every investigated incident involving React2Shell over 61 days was exported and parsed programmatically. Each request body was decoded through all of its encoding layers (JSON nesting, URL encoding, HTML entities, Unicode escapes, null-byte padding and non-Latin digits), and the JavaScript each payload would run was extracted and clustered by behaviour. Hex-encoded second stages were decoded and read in full..

Client projects are labelled Client 01 to Client 36 (requests logged under a shared multi-client label are grouped separately) and hostnames Host 01 to Host 46, in descending order of request volume. Labels are consistent across this report and the companion workbook. Timestamps are Pacific time. Source IPs were attributed to providers using the published IP ranges of the major cloud platforms, and ASN details for the highest-volume non-cloud IPs were confirmed on AbuseIPDB.

5.2 Volume and Timeline

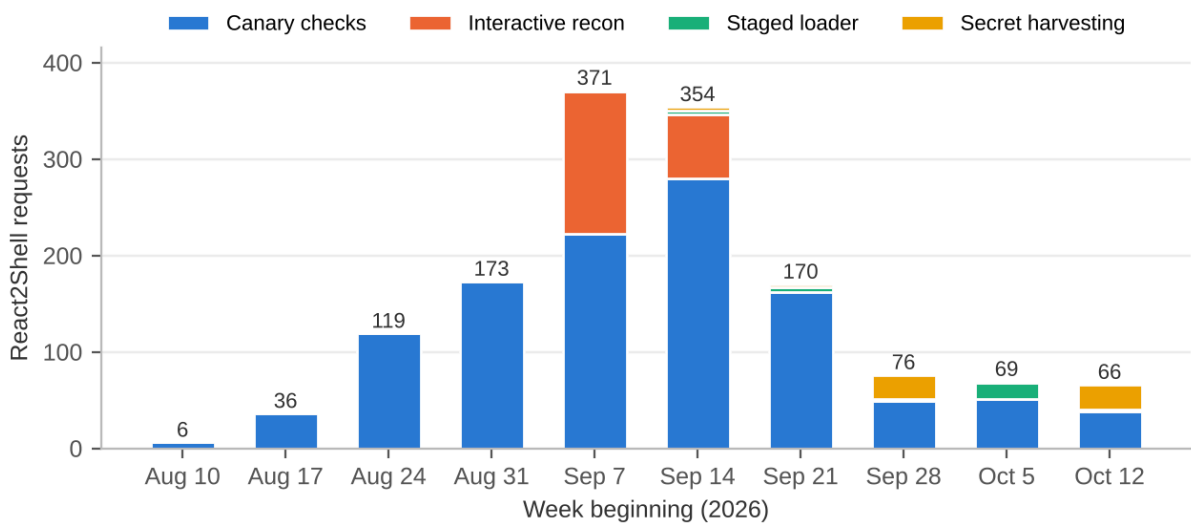


Figure 1. React2Shell requests per week by activity tier. The week of October 12 covers two days only.

Week beginning	Canary checks	Interactive recon	Staged loader	Secret harvesting	Total
August 10	6	0	0	0	6
August 17	36	0	0	0	36
August 24	119	0	0	0	119
August 31	173	0	0	0	173

Week beginning	Canary checks	Interactive recon	Staged loader	Secret harvesting	Total
September 7	222	148	1	0	371
September 14	280	66	4	4	354
September 21	162	0	5	3	170
September 28	49	0	2	25	76
October 5	51	0	17	1	69
October 12	38	0	2	26	66

- Activity was recorded on 58 of 62 days, averaging 23 requests and 4.8 new incidents per day.
- The busiest days were September 13 (149 requests), September 15 (114) and September 11 (77), driven by short, dense bursts from single IPs.
- Requests arrive around the clock. With the burst traffic removed, every hour of the day sees between 29 and 77 requests, which points to fully automated tooling.

5.3 Source Infrastructure

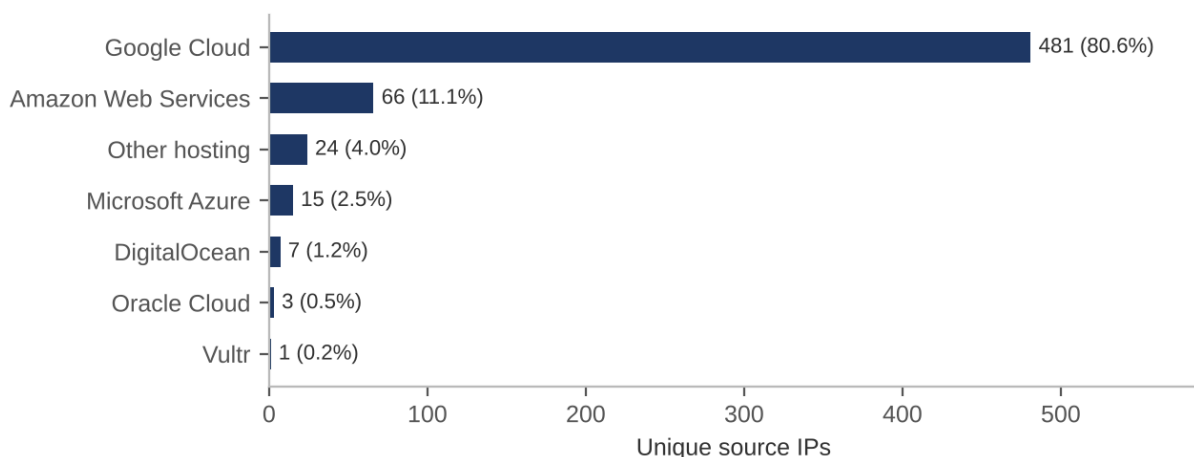


Figure 2. Unique React2Shell source IPs by hosting provider (597 IPs).

Attackers overwhelmingly rent short-lived cloud instances. 481 of the 597 IPs (80.6%) sit in Google Cloud ranges and a further 66 (11.1%) in AWS. 491 IPs sent exactly two requests and never returned, the signature of a scanner that cycles through fresh instances to avoid reputation blocking.

Source IP	Provider / ASN	Requests	Activity
45.148.10.13	TECHOFF SRV LIMITED, AS48090, NL (AbuseIPDB 100%)	118	September 13, 05:32 to 05:34. Junk-padded recon against 4 hosts
45.148.10.15	Same /24 as above	38	September 14, 04:00 to 04:02. Same payload, 1 host
20.81.188.20	Microsoft Azure, AS8075	30	September 11. Encoding-bypass test run against 1 host

Source IP	Provider / ASN	Requests	Activity
193.47.62.167	BESTDC LIMITED, AS216014, NL (AbuseIPDB 100%)	25	October 12. Secret-file sweep and full Kubernetes harvester
193.47.62.168	Adjacent to above	25	October 1. Same secret-file sweep and harvester
45.148.10.12	Same /24 as 45.148.10.13	24	September 15, 04:22 to 04:24. Same payload, 1 host
34.171.54.233	Google Cloud	12	September 1 to 3. Arithmetic canary across 4 hosts
45.138.12.89	Other hosting	7	October 5. Arithmetic canary with null-byte padding
93.123.109.166	TECHOFF SRV LIMITED, AS48090, NL (AbuseIPDB 100%)	4	September 18. Same recon payload as the 45.148.10.x bursts

5.4 Targeting

Host type	Hosts	Requests	Share
Arc-managed client dashboards	20	629	43.7%
Dashboards on client-owned domains	13	446	31.0%
Client public websites	5	182	12.6%
Preview environments	7	175	12.2%
Development environment	1	8	0.6%

Attackers do not choose targets by technology. Every hostname received the same payloads whether or not it runs React, and preview and development hostnames were found as readily as production ones. Public certificate transparency logs are a likely discovery source for these subdomains.

6. Attack Patterns and Payload Families

Decoding every request produced eleven distinct payload families, which group into four tiers of intent.

Family	Tier	Requests	IPs	Incidents	First seen
F1 Arithmetic canary (41*271)	Canary	1,080	517	261	August 13
F3 Static eval canary	Canary	52	35	31	September 15
F9 Marker echo canary	Canary	4	3	3	September 22
F2 id / hostname / pwd recon	Recon	173	4	4	September 13
F0 Junk-padded recon, body not captured	Recon	11	1	3	September 13
F4 Tagged recon (C3R marker)	Recon	30	1	1	September 11
F8a Staged host profiler (hex)	Staged	27	27	14	September 13
F8b Staged secret walker (hex)	Staged	4	1	1	September 27
F5 Environment dump	Secrets	9	7	9	September 14

Family	Tier	Requests	IPs	Incidents	First seen
F6 Targeted secret file read	Secrets	47	2	4	October 1
F7 Credential and Kubernetes secret harvester	Secrets	3	2	2	October 1

6.1 Canary Checks

Canary payloads prove code execution without doing damage, so the scanner can record the host as vulnerable for later use or resale.

- **F1 Arithmetic canary (1,080 requests, 75.0%).** Runs `echo $((41*271)) | base64 -w 0` and returns `MTE XMTEK`. Four variants were seen, including one with a `module.constructor._load` fallback for environments where `process.mainModule` is unavailable. This is the payload used by widely shared public scanning templates.
- **F3 Static eval canary (52 requests).** Returns the fixed string `vULN_cHeCk_sUcCeSs_69420` without running a shell command, testing whether JavaScript evaluation works even where `child_process` is blocked.
- **F9 Marker echo canary (4 requests).** Echoes unique markers such as `VULN_1790083574_8931` and `RECONX_R2S_OK` so responses can be matched to specific requests.

6.2 Interactive Reconnaissance

- **F2 id / hostname / pwd (173 requests).** Runs `id; hostname; pwd` through `/bin/sh` and returns the result through the redirect digest or between `<<RCE_OUT>>` and `<<RCE_END>>` markers. Every request was prefixed with exactly 4,096 bytes of padding. All four source IPs belong to TECHOFF SRV LIMITED (AS48090). Four bursts on September 13, 14, 15 and 18 each lasted under two minutes; one reached 118 requests in 105 seconds.
- **F0 Junk-padded recon (11 requests).** Same actor and bursts as F2, with the payload also Unicode-escaped. The padded body was large enough that the exploit code itself was not captured.
- **F4 Tagged recon (30 requests, one Azure IP).** Runs `echo C3R<random hex>; pwd; id; whoami; ls -d / /tmp` with a unique tag per request, cycling through seven encodings in one session: plain JSON, Unicode escapes, double URL encoding, HTML entities, Arabic-Indic digits, `{IFS}` in place of spaces, and an extra empty form key. This is a WAF-bypass harness working out which encoding gets through.

6.3 Secret Harvesting

- **F5 Environment dump (9 requests).** Runs `env` or `printenv` and returns every environment variable, which in a typical Node.js deployment includes database passwords, API keys and cloud credentials.
- **F6 Targeted secret file read (47 requests).** Uses `fs.readFileSync` directly, avoiding a shell, to read `.env` variants, `wp-config.php`, `../.aws/credentials`, `.ssh/id_rsa`, `/root/.ssh/id_rsa` and `../.git-credentials`, one file per request. The same 12-file sweep ran on October 1 and October 12 from two adjacent IPs on BESTDC LIMITED (AS216014).
- **F7 Credential and Kubernetes harvester (3 requests).** In one request it collects environment variables and `.env` files; searches `/var/www`, `/home`, `/root`, `/app` and `/srv` for credential files; reads AWS credentials, SSH keys, git credentials, `.netrc` and MySQL and PostgreSQL credential files; reads Docker secrets; and, if a Kubernetes service account token is mounted, calls

`https://kubernetes.default.svc` to list and decode every Secret and ConfigMap in the namespace.

6.4 Staged Loader

The F8 families wrap a second-stage script in hex so that no recognizable keywords appear in the request. A small loader imports `child_process` and `zlib` dynamically, builds `Buffer` from character codes to avoid the literal word, decodes and evaluates the hex, then compresses and hex-encodes the result into an error named `RCE_RES`.

- **F8a Host profiler (27 requests, 27 Google Cloud IPs, each used once).** Reports platform, user, kernel, CPU model and cores, RAM within container limits, IP addresses, writable paths, whether the host is bare metal, a container or Lambda, and every environment variable. It checks for **AVX2 CPU support**, which matters for efficient Monero mining, and sets a "high value target" flag on variables matching `AWS_`, `PRIVATE_KEY`, `MNEMONIC`, `WALLET`, `SEED_PHRASE`, `SENDGRID`, `MAILGUN`, `POSTMARK` or `SES_`, a writable Docker socket, a Kubernetes token or AWS credentials on disk.
- **F8b Secret walker (4 requests, one Vultr IP).** Walks the working directory, `/app`, `/srv`, `/var/www` and home up to six levels deep, collecting `.env` files, private keys, `.npmrc`, `.netrc`, `.pgpass`, `.my.cnf`, `next.config.js`, `docker-compose.yml`, `serverless.yml`, `terraform.tfvars` and `.tfstate`.

6.5 Evasion Techniques

Technique	Requests	Purpose
4,096-byte junk padding field	184	Push the payload past inspection windows
Unicode-escaped JSON keys and values	13	Hide keywords such as <code>__proto__</code> and constructor
Hex-encoded second stage with compressed output	31	Remove recognizable code from request and response
Double URL encoding	1	Bypass single-decode inspection
HTML entity encoding	1	Bypass JSON-aware inspection
Arabic-Indic digits for 0 to 9	1	Defeat patterns that expect ASCII digits after \$
Null bytes between characters	1	Break byte-level signatures
<code>\$(IFS)</code> in place of spaces	1	Bypass command-injection rules that look for spaces
Pretty-printed JSON with line breaks	1	Break single-line pattern matching
Shell-free file reads via <code>fs.readFileSync</code>	47	Avoid command-injection signatures

6.6 Escalation Over Time

First seen	Family	What changed
August 13	F1 Arithmetic canary	Commodity scanning begins
September 11	F4 Tagged recon	An operator tests WAF bypass encodings
September 13	F2 and F0 junk-padded recon	Padding used to evade inspection
September 13	F8a Staged profiler	Hosts profiled for mining and resale
September 14	F5 Environment dump	First direct attempt to steal secrets

First seen	Family	What changed
September 27	F8b Staged secret walker	Recursive credential search
October 1	F6 and F7 harvesters	Cloud, SSH, git and Kubernetes secret theft

7. What Attackers Are After

1. **A list of vulnerable servers**, to exploit later or sell as access.
2. **Host suitability for mining**: CPU model, AVX2 support, usable cores and RAM.
3. **Cloud credentials**: AWS keys from environment variables and `~/.aws/credentials`.
4. **Application secrets**: `.env` files, `wp-config.php`, `next.config.js`, Terraform variables and state.
5. **Source and deployment access**: git credentials, `.npmrc` tokens, `.netrc`, SSH private keys.
6. **Database credentials**: `.pgpass`, `.my.cnf`, `/etc/mysql/debian.cnf`.
7. **Container and cluster secrets**: Docker secrets, the Kubernetes service account token, and every Secret and ConfigMap that token can read.
8. **Assets with resale value**: cryptocurrency wallets and seed phrases, and transactional email keys usable for spam and phishing.

8. Impact Study: If Arc Had Been Vulnerable

8.1 Approach

This section is a counterfactual built entirely from our own traffic. We assume that each of the 46 targeted hosts had been running a vulnerable Next.js App Router application, and we treat every request in the dataset as if it had succeeded. We then read what the decoded payload would have done and returned. Nothing here is hypothetical attacker behaviour: every action described is code that was sent to us.

8.2 Discovery Is Fast and Certain

- All 46 hosts were probed. None was overlooked, including 7 preview environments and 1 development environment.
- In steady state, a new attacker reached each host every 37 hours (median). 90% of gaps between new attackers on the same host were under 7.1 days.
- 31 of 46 hosts were visited by 10 or more separate attackers within the window.

Put plainly: a vulnerable application deployed on our platform would most likely have been found and exploited within two days, and almost certainly within a week. Public reporting shows the same pattern at disclosure, when AWS saw exploitation attempts within hours.

8.3 Not One Intruder, Many

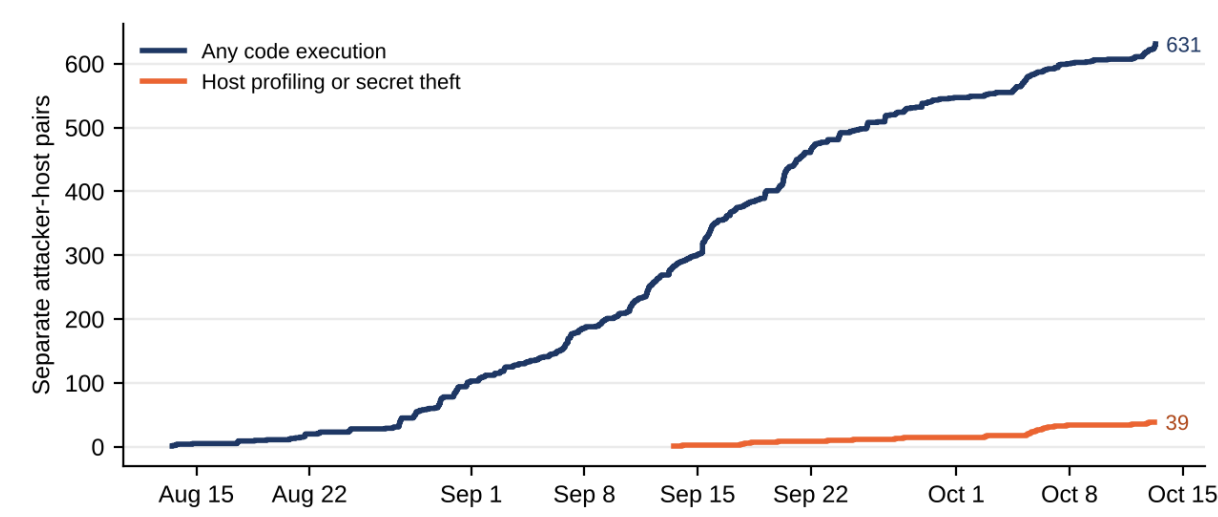


Figure 3. Cumulative separate attacker and host pairs that would have achieved code execution on a vulnerable host (blue), and those that would have profiled the host or attempted secret theft (orange).

Measure	Value
Separate attacker and host pairs that would have run code	631
Average separate attackers per host	13.7 (median 13, minimum 3, maximum 33)
Hosts receiving host profiling or secret theft payloads	18 of 46 (39%)
Attacker and host pairs attempting profiling or secret theft	39
Client projects with at least one host that would have been compromised	36 of 36

Incident response for a single intrusion is hard. Here, a single vulnerable host would have been compromised independently by an average of 14 unrelated parties, each with its own access, motives and tooling. Removing one would not have removed the others, and any one of them could have sold access to a ransomware operator.

8.4 Case Study A: An Arc-Managed Client Dashboard (Host 07)

Host 07 is a typical client dashboard on Arc-managed infrastructure. Over 47 days it received 49 React2Shell requests from 14 separate attackers. Had it been vulnerable:

Date	Source	Payload	What the attacker would have obtained
August 26	Oracle Cloud	F1 canary	Proof of code execution. Host added to a vulnerable list
August 28 to September 17	7 separate AWS and Google Cloud IPs	F1 canary	7 more parties confirm access. 8 independent footholds
October 3	Other hosting	F9 marker canary	A ninth party confirms access
October 5 and 7	2 Google Cloud IPs	F1 canary	Two more parties confirm access. 11 independent footholds

Date	Source	Payload	What the attacker would have obtained
October 6 and 7	2 more Google Cloud IPs	F8a staged profiler	Full host profile: container status, CPU and AVX2 support, memory, IP addresses, writable paths and every environment variable. If a service account token is mounted, as Kubernetes does by default, the host is flagged "high value"
October 12, 18:54:28	BESTDC LIMITED, AS216014	F7 full harvester	All environment variables, every .env file, AWS credentials, SSH keys, git and database credentials, Docker secrets, and, through the service account token, every Secret and ConfigMap in the namespace
October 12, 18:54:31	Same IP, 3 seconds later	F6 12-file sweep	A second pass at .env variants, AWS credentials, SSH keys and git credentials using shell-free reads

The progression matters. For six weeks the host would have looked quiet, touched only by "harmless" canary checks. Those checks were the reconnaissance. When the harvester arrived, would have taken the entire secret store in three seconds.

8.5 Case Study B: A Client Public Website (Host 02)

Host 02 is a client's main public website and the second most targeted host. It received 86 requests from 33 separate attackers, starting on August 13, the first day of the window. Had it been vulnerable:

Date	Source	Payload	What the attacker would have obtained
August 13 onward	24 separate IPs	F1 and F3 canaries	24 independent confirmations of code execution
September 11	Microsoft Azure	F4 bypass harness, 30 requests	User, hostname and working directory, plus a map of which payload encodings pass the site's defences
September 13, September 17, October 11	3 Google Cloud IPs	F8a staged profiler	Three separate host profiles with all environment variables
September 23, September 27, October 5, October 12	3 AWS IPs and 1 Azure IP	F5 environment dump	Four separate parties each take every environment variable: database credentials, API keys, email keys and any cloud keys
September 27	Vultr	F8b secret walker	Every .env file, private key, .npmrc, .netrc and database credential file within six directory levels, plus deployment configuration

By the end of the window, four unrelated groups would each have held a full copy of this client's application secrets. Rotating credentials after the first theft would not have been enough, because the same flaw remained open to the next group.

8.6 Blast Radius in a Managed Kubernetes Environment

A managed hosting provider concentrates many clients on shared infrastructure. That is efficient, and it also means one vulnerable application can be a path to more than its own data. The payloads we received map onto that path layer by layer:

Layer	How the observed payloads reach it	Consequence
Application	F5, F6, F7, F8a, F8b read environment variables and .env files	Database passwords, API keys, session signing secrets and email keys. Attackers can impersonate users and forge sessions
Client database	Credentials taken above	Read, alter or delete client records, including personal information
Container	F7 reads Docker secrets and mounted files; F8a maps writable paths	Staging area for miners, backdoors and further tooling
Kubernetes namespace	F7 uses the mounted service account token to list Secrets and ConfigMaps	Credentials and TLS keys for every service sharing the namespace
Cluster and other tenants	Any service account with broader read rights	Movement to other namespaces and, in a multi-tenant cluster, to other clients
Cloud account	F6 and F7 read AWS credential files; F8a flags AWS_ variables	Access to storage, backups, database snapshots and compute, limited only by the IAM policy
Source code and supply chain	F6, F7, F8b take git credentials and .npmrc tokens	Malicious commits or package releases that outlive the original intrusion
Email reputation	F8a flags SendGrid, Mailgun, Postmark and SES keys	Phishing sent from a client's trusted domain to its own community

8.7 People and Data at Risk

Our clients are non-profit, community and public-sector organizations. Their dashboards typically hold names, contact details, membership, volunteer and program participant records, and several serve communities that are more vulnerable to harm from exposure. For these organizations a breach is not only a technical event: it can damage the trust of the people they exist to serve, and their capacity to absorb the cost of a breach is limited.

8.8 Legal, Regulatory and Financial Consequences

Obligation or cost	What it would mean
PIPEDA (private sector, federal)	Breaches posing a real risk of significant harm must be reported to the Office of the Privacy Commissioner, affected individuals must be notified as soon as feasible, and records of every breach kept for two years. Knowingly failing to report or notify is an offence
BC FIPPA section 36.3 (public bodies)	Since February 1, 2023, public bodies must notify affected individuals and the Information and Privacy Commissioner without unreasonable delay when a breach could reasonably be expected to cause significant harm
Contractual duties	Hosting and service agreements typically require prompt notice to the client, who then carries the statutory duty
Average cost of a breach in Canada	CA\$6.98 million in 2025, up 10.4% from CA\$6.32 million in 2024 (IBM Cost of a Data Breach Report 2025)
Time to detect	IBM reports a mean time to identify of 162 days for organizations without extensive security AI and automation. Secrets stolen in October could be in use well into the following year
Direct costs not captured above	Cloud bills from cryptomining, credential rotation across every affected client, forensic investigation, rebuilding clusters, and staff time

Averages over-state costs for small organizations and under-state the reputational harm to a provider whose business is security. For a managed security provider, a breach caused by an unpatched dependency would undermine the service itself.

8.9 How Attackers Would Profit

- **Cryptomining:** the F8a profiler checks AVX2 support and real CPU and memory limits, the inputs needed to decide whether a host is worth mining on.
- **Access brokering:** canary results and "high value target" flags are exactly what initial access brokers sell to ransomware operators.
- **Cloud abuse:** stolen AWS keys fund compute, storage theft and further attacks at the victim's expense.
- **Spam and phishing:** transactional email keys let attackers send mail that passes the client's own SPF and DKIM checks.
- **Theft of crypto assets:** wallet, mnemonic and seed phrase variables are flagged for immediate use.

8.10 Worst Credible Outcome

Combining only behaviours present in our data: a single client deploys a Next.js application that has not been patched. Within two days a cloud scanner confirms code execution. Over the following weeks a dozen more parties do the same, one profiles the container and flags it as high value because of its mounted service account token, and a harvester takes every secret in the namespace in three seconds. Database credentials expose the client's member records. AWS keys reach backups. Git credentials allow a backdoor to be committed to the application itself. A miner consumes the node, and access is sold on. The first sign may be a cloud bill or a notification from a third party, months later.

None of this happened to Arc. All of it was attempted.

9. Why Arc Was Not Affected

Arc's hosted applications are built on a stack with no Node.js runtime and no React Server Components. The forged Flight chunks were never parsed and the attacker code never ran. Our detection flagged and recorded every attempt, and analysts reviewed all 297 incidents.

The decisive control was the absence of the vulnerable component, not a signature or a block list. That is a strong position but not a permanent one: a single new client application, internal tool or third-party component built on Next.js would change the picture immediately. The appropriate response to this study is therefore not reassurance but readiness: continuous inventory of what runs on the platform, rapid patching, and limiting what any one compromised workload can reach.

10. Lessons for Defenders

- **Treat canary traffic as reconnaissance, not noise.** The "harmless" arithmetic checks are how vulnerable hosts are listed for the harvesters that follow.
- **Detect payloads, not sources.** With 95% of IPs used once, reputation blocking would have stopped a handful of operators and none of the main scanner.
- **Normalize before matching.** Attackers padded, escaped, double-encoded and swapped digit sets. Signatures applied to raw bodies miss these variants, and scoring tuned on the common payload can under-rate the rarer, more dangerous ones.

- **Mind logging limits.** Padding is used to push payloads past inspection and logging windows. Capture enough of large bodies to see what follows the padding.
- **Assume every subdomain is known.** Preview and development environments were targeted as heavily as production.
- **Design for the day a workload is compromised.** The harvesters target the defaults of container platforms: mounted service account tokens, environment variables and credential files on disk.

11. Recommendations

11.1 Organizations Running React or Next.js

1. Inventory every application and dependency that includes next or react-server-dom-*.

```
npm ls next react-server-dom-webpack react-server-dom-turbopack react-server-dom-parcel
find / -path "*/node_modules/react-server-dom-*/package.json" -exec grep -H '"version"' {} \;
2>/dev/null
```
2. Upgrade to the recommended versions in Section 3.2 or later. There is no workaround.
3. If an application was exposed while unpatched, rotate every secret it could read and review its logs for the markers in Appendix B.

11.2 Hosting Providers and Platform Teams

1. Search clusters for Node.js and Next.js workloads, including sidecars and internal tools
2. Set automountServiceAccountToken: false on workloads that do not call the Kubernetes API, and confirm no application service account can list Secrets.
3. Restrict pod egress so a compromised workload cannot reach the Kubernetes API, the instance metadata service or the internet unless required.
4. Enforce IMDSv2 with a hop limit of 1 on cloud nodes so containers cannot read node credentials.
5. Prefer short-lived, scoped cloud roles over long-lived access keys, and keep secrets out of plaintext files where practical.
6. Add a framework and version check to client onboarding so new technology does not arrive unreviewed.

11.3 Detection Engineering

1. Normalize request bodies before matching: URL-decode repeatedly, decode HTML entities and Unicode escapes, strip null bytes, and map non-ASCII digits to ASCII.
2. Alert at high severity on any marker in Appendix B regardless of padding or encoding.
3. Log enough of oversized request bodies to see past padding, and capture the Next-Action header.
4. Auto-classify confirmed inert probes on hosts without RSC so analyst time goes to the rarer harvester and staged families.

Appendix A. Indicators

A.1 Canary and Exfiltration Strings

Indicator	Family	Meaning
MTExmTEK	F1	base64 of 11111, proof of execution
vUIN_cHeCk_sUcCeSs_69420	F3	Static eval canary
C3R<8 hex>	F4	Per-request recon tag
VULN_<timestamp>_<n>, TEST_<timestamp>, RECONX_R2S_OK	F9	Marker echo canaries
<<RCE_OUT>> ... <<RCE_END>>	F2	Output delimiters in digest
RCE_RES, JS_EXEC_ERR, JS_ERR	F8	Staged loader result and error names
FILESEP	F7	Harvester output separator
NEXT_REDIRECT;push;/login?a=, /x?out=, /x?d=	Several	Redirect digest output channels

A.2 Source Infrastructure

Indicator	Notes
45.148.10.12, 45.148.10.13, 45.148.10.15, 93.123.109.166	TECHOFF SRV LIMITED (AS48090), junk-padded recon bursts
193.47.62.167, 193.47.62.168	BESTDC LIMITED (AS216014), secret file sweep and Kubernetes harvester
20.81.188.20	Microsoft Azure, encoding-bypass testing
173.199.127.30	Vultr, F8b staged secret walker
All 597 IPs	Source IPs tab of the companion workbook

Appendix B. Detection Markers

Apply after normalization. Any one in a POST body is a strong indicator; two or more is near-certain.

```
\$\d+:__proto__:then  
"status"\s*:\s*"resolved_model"  
\$\d+:constructor:constructor  
"_formData"\s*:\s*\{\s*"get"  
"_prefix"\s*:  
"\$@\d+"  
"\$B\d+"  
\$ACTION_ID_  
NEXT_REDIRECT;  
process\.mainModule\.require|module\.constructor\._load|process\.getBuiltinModule
```

Appendix C. Glossary

Term	Meaning
React Server Components (RSC)	React feature that renders components on the server and lets the browser call server-side functions

Term	Meaning
Flight protocol	The serialization format RSC uses to pass data between browser and server
Server Action / Server Function	A server-side function the browser can invoke directly; the entry point React2Shell abuses
Insecure deserialization	Turning attacker-supplied data back into objects in a way that lets the attacker control program behaviour
Canary	A harmless command whose output proves code execution, used by scanners to confirm a target
Service account token	A credential Kubernetes mounts into pods by default so they can call the Kubernetes API
IMDS	Instance metadata service; the cloud endpoint from which a server can obtain its own credentials
ASN	Autonomous System Number; identifies the network operator that owns an IP range
TLP:CLEAR	Traffic Light Protocol label for information that may be shared publicly

Appendix D. Sources

- React: Critical Security Vulnerability in React Server Components, react.dev/blog/2025/12/03/critical-security-vulnerability-in-react-server-components
- React: Denial of Service and Source Code Exposure in React Server Components, react.dev/blog/2025/12/11/denial-of-service-and-source-code-exposure-in-react-server-components
- Next.js: Security Advisory CVE-2025-66478, nextjs.org/blog/CVE-2025-66478
- Next.js: Security Update December 11, 2025, nextjs.org/blog/security-update-2025-12-11
- Vercel: Summary of CVE-2025-55182, vercel.com/changelog/summary-of-CVE-2025-55182
- AWS Security Blog: China-nexus cyber threat groups rapidly exploit React2Shell vulnerability
- Trend Micro: CVE-2025-55182 Analysis, PoC and In-the-Wild Exploitation
- Miggo: React2Shell Technical Breakdown
- BleepingComputer: React2Shell flaw exploited to breach 30 orgs, 77k IP addresses vulnerable
- UpGuard: Critical React2Shell Flaw Added to CISA KEV
- Office of the Privacy Commissioner of Canada: What you need to know about mandatory reporting of breaches of security safeguards
- Government of British Columbia: FOIPPA Policy and Procedures Manual, Privacy Breach Notifications
- IBM Canada: Cost of a Data Breach Report 2025 press release, July 30, 2025
- AbuseIPDB records for 45.148.10.13, 93.123.109.166, 193.47.62.167 and 20.81.188.20
- Cloud provider IP ranges: github.com/lord-alfred/ipranges
- Arc Technologies web application intrusion detection, 297 investigated incidents