

LLM-generated Information Stealer for Bug Bounty Hunting

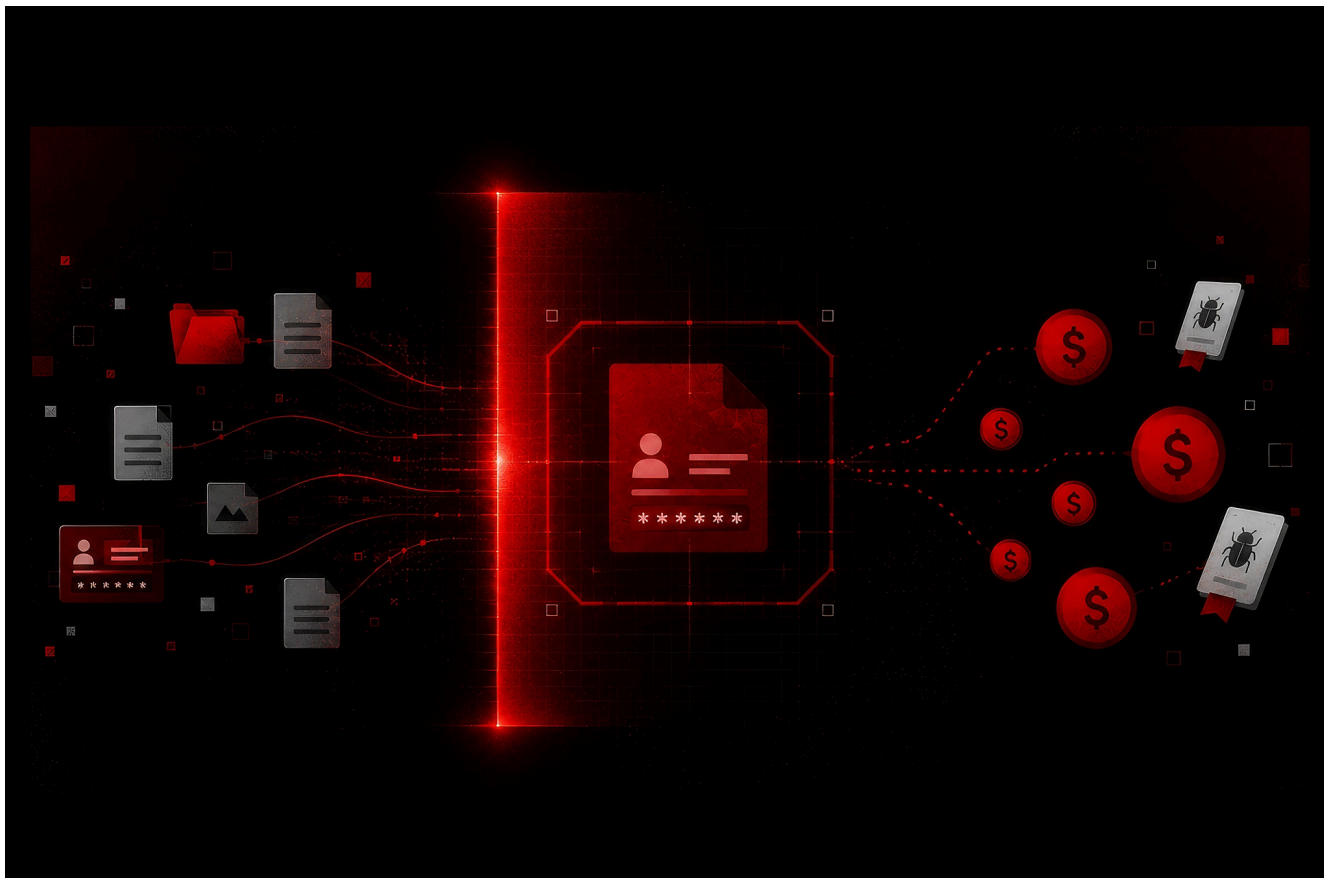
 crowdstrike.com/en-us/blog/phantomraven-llm-generated-information-stealer-for-bug-bounty-hunting

Maddie Stewart



PhantomRaven: An LLM-Generated Information Stealer Developed for

Bug Bounty Hunting



CrowdStrike Counter Adversary Operations identified a financially motivated threat actor who works as a bug bounty hunter and who developed and distributed the JavaScript (JS)-based information stealer *PhantomRaven* via npm, a platform on which developers can access open-source packages to build applications and software. The developer likely wrote the malware using a large language model (LLM), an assessment made with high confidence based on verbose comments, placeholder code, and statistical token-analysis patterns.

Threat Actor Profile and Attribution

In November 2025, the threat actor deploying *PhantomRaven* contacted a potential victim organization and claimed to have identified a compromised device. The threat actor attributed the compromise to a dependency-confusion attack that used malicious npm packages to deploy *PhantomRaven*. The email address used to contact the potential victim included the string JPD as part of the username.

The *PhantomRaven* operator is a self-proclaimed bug bounty hunter who has been active since November 2022. According to their public X profile, the threat actor has collected bounties from at least nine entities across the technology, retail, and hospitality sectors using

reputable bug bounty submission platforms, including Bugcrowd, Intigriti, YesWeHack, HackenProof, and HackerOne. To date, CrowdStrike Counter Adversary Operations has not observed *PhantomRaven* logs for sale on log shops, further indicating that *PhantomRaven*'s operator likely uses the information stealer solely to identify bug bounty opportunities.

CrowdStrike Falcon® Complete managed detection and response (MDR) responded to and remediated multiple incidents involving *PhantomRaven*. Falcon Complete identified two npm packages containing *PhantomRaven* scripts: *transform-jsbi-to-bigint*, published by npm user *jpdhellonpm1*, and *sort-imports-es6-autofix*, published by npm user *jpd15*. The threat actor discussed in this post likely operates both monikers; this assessment is made with high confidence, as both *PhantomRaven* packages contain files with the threat actor's name and initials in the description and author fields, respectively, and both usernames include the string JPD from the initial contact email (Figures 1 and 2).

Figure 1. *jpdhellonpm1*'s npm account profile

Figure 2. *jpd15*'s npm account profile

The usernames *jpdhellonpm1* and *jpd15* are similar to usernames that industry sources associate with *PhantomRaven* deployments using malicious npm packages. These usernames include *jpd12*, *jpd13*, *npmhell*, *npmpackagejpd*, *npmtestdharsh*, *jpdhackerone11*, and *packagedharsh*.¹ Two of these monikers contain characters that appear in the threat actor's name, X account username, and email address, the latter of which has been used to contact potential victims. Moreover, the username *jpdhackerone11* references HackerOne, a platform that the threat actor claims they use to manage their bug submissions and payments.

npm profiles associated with *jpdhellonpm1* and *jpd15* have published packages containing files that include remote dynamic dependency (RDD) links to the domain `npm[.]jpartifacts[.]com`. November 2025 *PhantomRaven* incidents leveraged this domain for command and control (C2).

In August 2025, the threat actor claimed to have discovered a remote code execution (RCE) vulnerability via a malicious npm package they published. The threat actor explained that they had compromised the target machine and executed their `preinstall` script, which purportedly allowed them to achieve RCE. While CrowdStrike Counter Adversary Operations could not verify this claim, a November 2025 *PhantomRaven* campaign exhibited similar techniques.

In December 2025, CrowdStrike Counter Adversary Operations identified the threat actor's GitHub account. In February 2025, the threat actor used this account to submit a GitHub issue on PyPI, asking why their code failed to upload to PyPI. A member of the PyPI organization

replied that PyPI does not allow projects with names that are similar to existing or deleted projects (a policy consistent with preventing dependency-confusion attacks) and further accused the threat actor of attempting to build an information stealer.

As evidence of their accusations, the PyPI organization member provided a link to the threat actor's main project, which was subsequently removed from the platform. However, multiple associated Python files currently remain accessible. Analysis of these files reveals they contain code for an information stealer similar to *PhantomRaven*. However, unlike previously identified *PhantomRaven* samples that leverage JS and npm, this tool uses Python and PyPI. One analyzed Python file contains metadata, including the email address `jpdtester01@gmail[.]com`; industry sources associate this username with *PhantomRaven*.²

Infection Vector

The threat actor distributes the malware via typosquatted npm packages that contain minimal, non-malicious code; typically, a simple `Hello, world!` script. However, the packages also specify a dependency via an HTTP URL rather than a standard npm package reference. At installation, npm fetches this remote dependency from attacker-controlled infrastructure; the returned package is the *PhantomRaven* payload.

The fetched malicious package includes a `preinstall` script³ that automatically executes during installation. In June 2026, npm released a new version of its package manager to prevent `preinstall` scripts from executing in packages when included as a dependency unless the developer explicitly allows this activity.⁴

In npm version 12 or later, if a developer attempts to install a dependency package with a `preinstall` script, they receive a warning message indicating that the script has been blocked and will not automatically execute. Only after the developer explicitly approves the `preinstall` script does it execute. Figure 3 shows what developers see when the `preinstall` script is blocked and presented for the developer's inspection.

Developer's Project Run via `npm install` Command

added 1 package, and audited 3 packages in 148ms

```
found 0 vulnerabilities
npm warn install-scripts 1 package had install scripts blocked because they are not
covered by allowScripts:
npm warn install-scripts   [developer-project-package]@1.0.0 (preinstall: node
index.js)
npm warn install-scripts
npm warn install-scripts Run `npm install-scripts ls` to review, or `npm install-
scripts approve <pkg>` to allow.
```

Malicious Project Run via npm install-scripts ls Command

```
1 package has install scripts blocked because they are not covered by allowScripts:  
  [typosquatted/malicious-package]@1.0.0 (preinstall: node index.js)
```

Run ``npm install-scripts approve <pkg>`` to allow, or ``npm install-scripts deny <pkg>`` to deny.

Figure 3. Installation messages displayed to developer in npm version 12 or later

Technical Overview

PhantomRaven is a simple JS information stealer that exfiltrates system information and continuous integration/continuous deployment (CI/CD)-related environment variables, likely in an attempt to collect account credentials. The code was almost certainly LLM-generated, and the author's technical sophistication is likely low.

The malware collects the following information (Table 1).

Category	Information Collected
System information	Operating system (OS)
	Architecture
	Hostname
	Local and external IP addresses
Process and runtime details	Current directory
	Current process ID
	NodeJS version
	Command-line arguments
	Environment variables
User ID	Username and email addresses from Git/npm configurations
Timestamp	Current date and time

CI/CD environment variables for GitHub Actions, GitLab CI, Jenkins, and CircleCI

The CI/CD variables include the following:

- BUILD_TAG
- BUILD_URL
- CI_NAME
- CI_PROJECT_ID
- CI_PROJECT_NAME
- CI_PROJECT_PATH
- CI_SERVER_NAME
- CI_SERVER_VERSION
- CI_SERVER
- CI
- CIRCLE_BUILD_URL
- CIRCLE_PROJECT_REPONAME
- CIRCLE_USERNAME
- CIRCLECI
- GITHUB_ACTION
- GITHUB_ACTIONS
- GITHUB_ACTOR
- GITHUB_REPOSITORY
- GITHUB_RUN_ID
- GITHUB_WORKFLOW
- GITLAB_CI
- HOME
- JENKINS_URL
- JOB_NAME
- LOGNAME
- npm_config_registry
- npm_package_name
- npm_package_version
- PATH
- USER

Table 1. Information collected by PhantomRaven

To determine the current external IP address of the system on which the malware is executing, *PhantomRaven* contacts `https[://api64[.]ipify[.]org?format=json`. After this request has concluded (regardless of success), the malware exfiltrates all collected information via HTTP GET and POST requests.

The GET request encodes the data as a query string parameter while the POST request serializes the data as JSON before placing it in the POST body. For the POST method, *PhantomRaven* uses the very minimal user-agent string Mozilla/5.0 (Windows NT 10.0; Win64; x64). The user agent is valid but misses many common elements, including browser type and version.

PhantomRaven's code also contains an incomplete fallback WebSocket exfiltration method. However, the code's hardcoded exfiltration URL is `wss[: ]//yourserver[.]com/socket` — highly likely a placeholder and not malicious infrastructure.

LLM-Generated Code Characteristics

PhantomRaven's code was almost certainly LLM-generated. This assessment is made with high confidence based on statistical token-analysis patterns as well as verbose comments and placeholder code, which indicate the text is highly consistent with a generated LLM token stream. LLM generation explains several unusual design choices, including exfiltrating data via both HTTP POST and GET requests.

Figure 4 shows a heavily abridged rendering of the source code's topmost structure and associated comments. In this code snippet, the marker `/.../` indicates an omission; all comments starting with `//` are present in *PhantomRaven*'s code as shown. The code includes a comment before every global variable and function definition, despite obvious redundancy between the comment's explanation and the defined symbol's name.

```
const os = require("os");
const https = require("https");
const fs = require("fs");
const path = require("path");

// Function to detect user email from various sources
function detectUserEmail() {
  /*...*/
}

// Collect CI/CD and Environment Information
const ciEnvVars = {
  /*...*/
};

// Collect System Information
const systemInfo = {
  /*...*/
};

// Fetch public IP dynamically
https.get(/*...*/, (res) => {
```

```

    /*...*/
}).on("error", () => sendData(systemInfo));

// List of endpoints
const endpoints = {
    /*...*/
};

// Get available endpoint
function getEndpoint(type = 'log') {
    /*...*/
}

// Convert system info to query string
function buildQueryParams(data) {
    /*...*/
}

// New function to lookup systems by email
async function lookupByEmail(email) {
    /*...*/
}

// Send Data (GET and POST)
async function sendData(data) {
    /*...*/
}

// WebSocket Backup (if HTTP requests fail)
async function sendViaWebSocket(data) {
    /*...*/
}

```

Figure 4. Verbose PhantomRaven code comments

Assessment

This threat actor's behavior represents an expansion of known eCrime activities. Most criminal actors that CrowdStrike Counter Adversary Operations tracks rent commodity tools or operate their own proprietary malware; however, this threat actor has likely developed their proprietary *PhantomRaven* to compromise company assets and then used these compromises as leverage to claim rewards from reputable disclosure programs.

eCrime threat actors will likely continue integrating AI-generated tooling into their operations, as these tools reduce technical barriers to participating in eCrime activity and accelerate tool creation. This assessment is made with moderate confidence based on the observed *PhantomRaven* deployment, as well as deployment of multiple likely AI-generated tools by

several big game hunting (BGH) adversaries, including a TRAVELING SPIDER /INC affiliate and PUNK SPIDER, who deployed AI-generated PowerShell (PS) scripts for operations throughout late 2025 and early 2026.

Recommendations

These recommendations can be implemented to help protect against the activity described in this report:

- Consider implementing a private npm registry (where package installations can be controlled, blocked, and monitored) rather than using the official npm registry
- Configure npm to use `--ignore-scripts` by default, and selectively enable script execution only for trusted packages⁵
- Update npm to the latest version to restrict `preinstall` and `postinstall` script execution by default
- Educate users on dependency-confusion attacks that exploit npm’s dependency resolution to deliver malicious packages
- Use `npm audit` for known vulnerabilities and problems

Table 2 details the tactics and techniques described in this blog post.

Tactic	Technique	Observable
Reconnaissance	T1016.001 - System Network Configuration Discovery: Internet Connection Discovery	<i>PhantomRaven</i> contacts <code>https[:]//api64[.]ipify[.]org</code> to determine the infected system’s external IP address
Resource Development	T1583.001 - Acquire Infrastructure: Domains	The threat actor registered and operated multiple domains for C2 infrastructure, including <code>packages[.]storeartifact[.]com</code> , <code>registry[.]storageartifact[.]com</code> , and <code>npm[.]jpartifacts[.]com</code>
	T1587.001 - Develop Capabilities: Malware	The threat actor developed <i>PhantomRaven</i> , likely using an LLM to generate the JS code

Initial Access	T1195.001 - Supply Chain Compromise: Compromise Software Dependencies and Development Tools	The threat actor published typosquatted npm packages that fetched malicious dependencies via HTTP URLs during installation, exploiting the npm supply chain
Execution	T1059.007 - Command and Scripting Interpreter: JavaScript	<i>PhantomRaven</i> executes as JS code within the NodeJS runtime environment
	T1072 - Software Deployment Tools	<i>PhantomRaven</i> leverages npm's preinstall script functionality to automatically execute during package installation
Defense Evasion	T1027.009 - Obfuscated Files or Information: Embedded Payloads	The threat actor embedded malicious payloads behind HTTP URL dependencies that are not displayed in npm's web interface
	T1036.005 - Masquerading: Match Legitimate Name or Location	Typosquatted package names mimic legitimate development tools and libraries (e.g., transform-jsbi-to-bigint and sort-imports-es6-autofix)
Credential Access	T1552.001 - Unsecured Credentials: Credentials In Files	<i>PhantomRaven</i> searches Git and npm configuration files for email addresses and credentials
	T1552.007 - Unsecured Credentials: Container API	The malware collects CI/CD-related environment variables from GitHub Actions, GitLab CI, Jenkins, and CircleCI that may contain authentication tokens and API keys

Discovery	T1082 - System Information Discovery	<i>PhantomRaven</i> collects OS type, architecture, hostname, NodeJS version, process ID, and current working directory
	T1083 - File and Directory Discovery	<i>PhantomRaven</i> searches for and reads <code>package.json</code> files, Git configuration files, and npm configuration files
	T1614.001 - System Location Discovery: System Language Discovery	<i>PhantomRaven</i> collects system location and time zone information
Collection	T1005 - Data from Local System	<i>PhantomRaven</i> collects system information, environment variables, and configuration files from the infected system
	T1119 - Automated Collection	All data collection and exfiltration are performed automatically upon malware execution, without user interaction
Command and Control	T1071.001 - Application Layer Protocol: Web Protocols	<i>PhantomRaven</i> uses HTTP GET and POST requests to communicate with C2 infrastructure
	T1104 - Multi-Stage Channels	The malware includes both HTTP-based exfiltration and an unfinished WebSocket fallback method for redundant C2 communication
Exfiltration	T1041 - Exfiltration Over C2 Channel	Collected data is exfiltrated to the threat actor's C2 servers via HTTP GET and POST requests

Table 2. PhantomRaven tactics and techniques aligned with the MITRE ATT&CK® framework

Indicators of Compromise (IOCs)

IOCs	Description
------	-------------

packages[.]storeartifact[.]com	C2 domains
registry[.]storageartifact[.]com	
packages[.]storageartifact[.]com	
npm[.]jpartifacts[.]com	
54.173.15[.]59	C2 IP address
c31831d47fcbf52ff1f4e61838611916a4276d005a564e69946d5dac04235eed	<i>PhantomRaven</i>
95a7dcc6de46826b22c43bee7fc550f3b5e2e6cbc5f33b0c241faf523641cf63	SHA256
db3fe46df0a65fe9f8c99d2e11126a032a72e9814e354ce017448ce088a01e02	hashes

Table 3. PhantomRaven IOCs

Additional Resources

- Learn more about [CrowdStrike Counter Adversary Operations](#) threat intelligence and threat hunting.
- Tune in to the [Adversary Universe podcast](#), where CrowdStrike reveals the threat actors behind the latest cyberattacks.
- Dive into more topics like this with [Fal.Con Digital](#), featuring keynote livestreams and 100+ sessions available on demand.



CrowdStrike 2026 Global Threat Report

AI threats have reached a critical turning point. Access the definitive look at the cyber threat landscape.