Proof of Concept (PoC) Report: <u>**JS-Beautify & JS-Deobfuscator Tool**</u>

1) Tool Name: JS-Beautify

Description:

JS-Beautify is a code formatting tool that reformats JavaScript, HTML, and CSS to make the code more readable and standardized according to user-defined or default formatting rules.

What Is This Tool About?

JS-Beautify is used to beautify (pretty-print) minified or messy source code. It helps developers and analysts by converting obfuscated or compact code into a structured format, which is easier to read and analyze.

Key Characteristics / Features:

- Supports JavaScript, HTML, and CSS
- CLI and web-based versions available
- Fully configurable formatting rules
- Works on Windows, macOS, and Linux
- Supports tab/space indentation
- Preserves line breaks and comments
- Supports beautifying embedded scripts
- Allows formatting from stdin or files
- Easily integratable with IDEs and CI pipelines
- Lightweight and open-source
- Can beautify entire folders
- Available as a Python and Node.js package
- Offline use supported
- Maintained by a large community
- Handles both ES5 and ES6 syntax

Types / Modules Available:

- JavaScript Beautifier
- HTML Beautifier
- CSS Beautifier
- Command Line Interface (CLI)
- VS Code / Sublime Plugin
- Web Interface / Online Tool

How Will This Tool Help

- Helps reverse engineer minified or obfuscated JavaScript
- Makes reading and debugging easier
- Useful in web app security assessments
- Helps during malware JS unpacking
- Aids source code reviews for vulnerabilities
- Improves collaboration with clean formatting

Summary:

1. Supports JS/HTML/CSS beautification
2. Cross-platform compatibility
3. CLI, web, and plugin-based options

4. Fully customizable formatting

5. Ideal for clean code or reverse engineering

6. Supports large files and folders

7. Works offline

8. Preserves code comments

9. Can integrate in CI pipelines

10. Handles ES5/ES6 and beyond

11. Supports stdin/stdout for pipelines

12. Used by devs, analysts, and auditors

13. Maintained open-source tool

14. Useful in malware analysis

15. Minimal setup and easy use

Time to Use / Best Case Scenarios:

- When analyzing obfuscated JS from web pages

- During malware or exploit unpacking

- For formatting legacy or badly written code

- While preparing code for audits

- Prior to source review or team collaboration

When to Use During Investigation:

- During web malware analysis

- Reverse engineering browser-based payloads

- Analyzing obfuscated phishing kits

- Reviewing client-side logic in security tests

- Format recovery from broken source files

Best Person to Use This Tool & Required Skills:

Best User: Web Security Analyst / Malware Analyst / Developer

Required Skills:

- Basic JavaScript/HTML/CSS knowledge

- Familiarity with terminal/CLI (for advanced use)

- Understanding of code minification and obfuscation

- Ability to identify logical structures in scripts

Flaws / Suggestions to Improve:

- Lacks syntax validation (beautifies even broken code)

- No inline code execution or debugging

- Limited support for modern JavaScript frameworks

- No built-in diff view or versioning

- Could benefit from real-time collaboration features

Good About the Tool:

- Free, open-source, and lightweight

- Fast and effective beautification

- Works across platforms and languages

- Supports plugins and automation

- Helps greatly with static analysis of web code

2) Tool Name: JS-Deobfuscator

Description:

JS-Deobfuscator Tool is a specialized utility designed to reverse JavaScript obfuscation. It helps uncover the original logic of scripts that have been deliberately modified to hide their true purpose, often used in malware or web-based exploits.

What Is This Tool About?
This tool is focused on analyzing and cleaning up obfuscated JavaScript code. It reveals hidden operations, replaces encoded values, and helps restore original source behavior. This makes analysis and debugging much easier, especially in the field of cybersecurity.

Key Characteristics / Features:

1. Supports static and dynamic deobfuscation
2. Works with most JavaScript obfuscation types
3. Inline string decoding
4. Call stack tracing
5. Built-in AST (Abstract Syntax Tree) parser
6. Command-line and GUI versions available
7. Logs every transformation for transparency
8. Supports plugin extensions for new techniques

9. Fast deobfuscation on large files
10. Supports ES5, ES6, and newer JS features
11. Pattern recognition for common obfuscators
12. Cross-platform support (Windows, Linux, macOS)
13. Works with browser-injected JS
14. Output comparison with original code
15. Helps identify suspicious function calls

Types / Modules Available:

- ✓ Static JS Analyzer
- ✓ Obfuscation Pattern Detector
- ✓ Live Deobfuscation Console
- ✓ String Decoder Engine
- ✓ Malicious Function Identifier
- ✓ AST Viewer & Editor

How Will This Tool Help?

-Helps understand the behavior of obfuscated JavaScript
-Used in malware analysis and threat detection
-Assists in identifying hidden payloads
-Supports researchers in web-based exploit analysis
-Helps in code transparency for audit and documentation

Summary:

- Helps reverse obfuscated JavaScript
- Supports common obfuscation patterns
- Works with CLI and GUI
- Handles inline encodings

- Detects hidden payload triggers
- Fast and lightweight
- Perfect for malware analysts and students
- No internet required to function
- Works with modern and legacy JS
- Export options available
- Helps in classroom analysis exercises
- Open-source and customizable
- Detailed logs of changes made
- Great for practical learning
- Easy to operate with basic JS knowledge

Time to Use / Best Case Scenarios:

- When reviewing suspicious JavaScript files
- During CTF challenges and malware labs
- While analyzing phishing kits
- When checking embedded JS in PDFs or HTML
- Before sharing code to ensure transparency

When to Use During Investigation:

- After detecting obfuscated scripts on websites
- During reverse engineering of drive-by downloads
- In malware research labs
- During browser extension analysis
- When analyzing JavaScript-heavy attack vectors

Best Person to Use This Tool & Required Skills:

Best User: Cybersecurity Students / Malware Analysts / Ethical Hackers

Required Skills:

- Basic understanding of JavaScript syntax
- Familiarity with obfuscation techniques

- Some knowledge of debugging tools
- Logical thinking for code analysis

Flaws/ Suggestions to Improve:

- May not detect newer or custom obfuscation methods
- GUI could be more modern and interactive
- Add support for browser plugin integration
- Improve AST documentation for learners
- Offer more tutorials for beginners

Good About the Tool:

- Tailored for security learners and professionals
- Lightweight and doesn’t need installation
- Great for JS malware lab work
- Transparent transformation logs
- Encourages hands-on reverse engineering practice