# **blackhat** EUROPE 2023

### DECEMBER 4-7 Excel London / UK



# **blackhat**®

# HODOR: Reducing Attack Surface on Node.js via System Call Limitation

Speakers: Wenya Wang, Xingwei Lin

Contributors: Wenya Wang, Xingwei Lin, Jingyi Wang, Wang Gao, Dawu Gu M









### Agenda

- Introduction
- **Previous work & Remaining challenges**
- HODOR: system call level protection system for Node.js applications
- **Evaluation**
- **Conclusion & Takeaways**

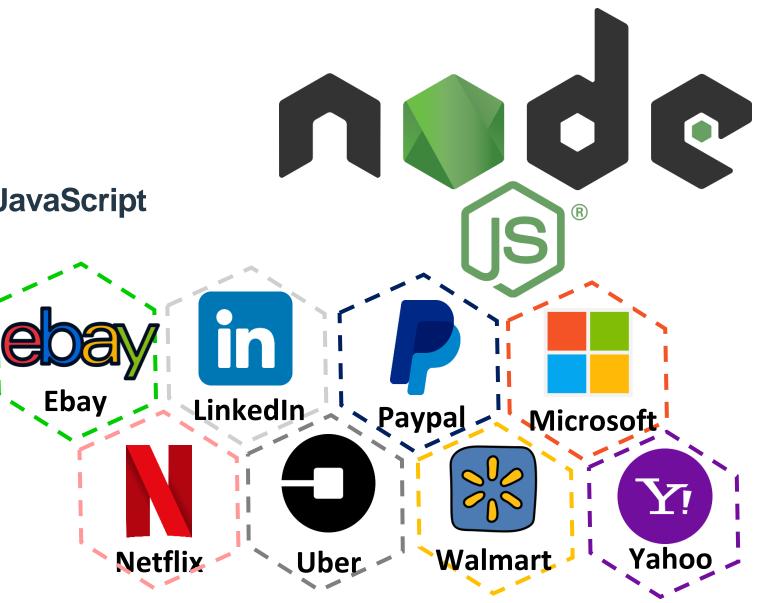




# Node.js

Node.js is an **open-source, cross-platform JavaScript runtime** environment.

- ✓ Asynchronous and Event-Driven
- ✓ Single-Threaded
- ✓ Cross-Platform
- ✓ NPM (Node Package Manager)
- ✓ JavaScript Everywhere





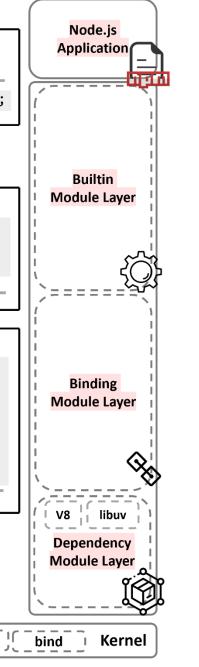


## Node.js architecture

- ✓ Node.js Applications (JS)
- ✓ Built-in Module Layer (JS)
- ✓ Binding Module Layer (C++)
- ✓ Dependency Module Layer (C)

```
→1 var child = require('child process');
   2 var c = child.exec(cmd, {*}, function(err) { '
         if (callback) callback(*);
   4 });
5 const { Process } = internalBinding('process_wrap');
   6 function exec(*) _  \leq 
   7 this._handle = new Process(); >
8 ChildProcess.prototype.spawn = function(options) {
         this._handle.spawn(options); -
   9
  10
      };
  11 module.exports = child process = {
 -12 exec,
  13 };
  14 static void Spawn(*) {
  15 uv_spawn(*); _
  16 }
  17 static void Initialize(*) {
        env->SetProtoMethod(*, "spawn", Spawn);
  18
  19 }
  20 NODE_MODULE_CONTEXT_AWARE_INTERNAL(
  -21
       process_wrap,
       node::ProcessWrap::Initialize
  22
  23)
  24 int uv_spawn(*) { ←
       uv__process_child_init(*);
  25
  26 }
  27 static void uv_process_child_init(*) {
       execvp(*);_ _
   28
  29 }
  Syscall1 ) Syscall2 ) fork ) exec ) write ( read ) bind Kernel
     5
```







# Motivation

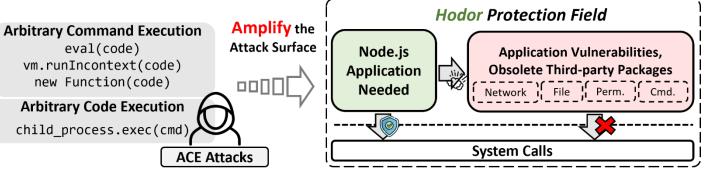
- $\succ$  NPM is a package manager with over 1 million packages  $\rightarrow$  The key to the success of Node.js
- ➤ 19.63% of packages in the NPM ecosystem depend on vulnerable packages, such as gadget chain attacks, inject-related attacks, and supply chain attacks.  $\rightarrow$  Most of them may lead to ACE attacks.

Arbitrary Command/Code Execution: the attackers can perform arbitrary dangerous critical operations

- mail `cat / etc /passwd`
- mail `nc –l –e /bin/bash 8001`
- mail `su root`

. . .

1 // main.js growl(message); \_ // ./lib/growl.js 10 11 12 function growl(msg, \*) { 13 args = [cmd.pkg]; 14 15 exec(args.join(''),...); 16 };





### var growl = require("growl"); var message = 'You have mail!'; exports = module.exports = growl; ← var exec = require('child process').exec cmd = { pkg: "notify-send" }; args.push(quote(msg)); ← Growl Application (v1.8.0)





- Introduction
- **Previous work & Remaining challenges**
- HODOR: system call level protection system for Node.js applications
- **Evaluation**
- **Conclusion & Takeaways**





# How to reduce the attack surface of ACE attacks for Node.js applications?





# How to reduce the attack surface of **ACE attacks for Node.js applications?**

### **Threat Model**

- ✓ Consider an attacker with <u>ACE ability</u>
- $\checkmark$  Not considered: preventing ACE, code vulnerabilities in binding layer/dependency layer, race condition, DOS attack, etc

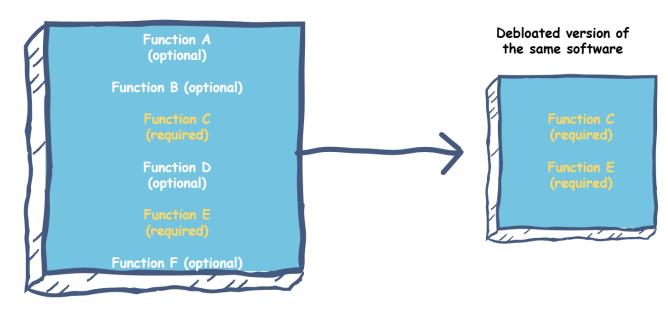




# **Existing Works: Software Debloating**

### • Use program analysis to cut the *useless code*

✓ (USENIX Sec'19) RAZOR: A Framework for Post-deployment Software Debloating ✓ (USENIX Sec'19) Less is More: Quantifying the Security Benefits of Debloating Web Applications ✓ (Usenix Sec'20) Slimium: Debloating the Chromium Browser with Feature Subsetting ✓ (RAID'20) Mininode: Reducing the Attack Surface of Node.js Application



Original version of a software





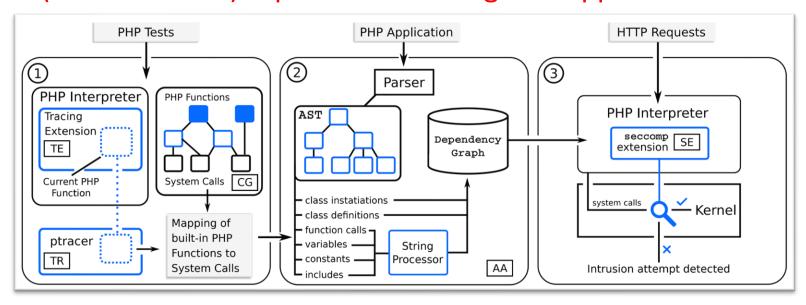
### What is software debloating? (educative.io)



# **Existing Works: System Call Limitation**

### Restrict the system calls that can be used by the application

√ (USENIX Sec'20) Temporal System Call Specialization for Attack Surface Reduction √ (RAID'20) Confine: Automated System Call Policy Generation for Container Attack Surface Reduction √ (RAID'20) sysfilter: Automated System Call Filtering for Commodity Software √ (PLDI'20) BlankIt Library Debloating Getting What You Want Instead of Cutting What You Don't √ (USENIX Sec'21) Saphire: Sandboxing PHP Applications with Tailored System Call Allowlists



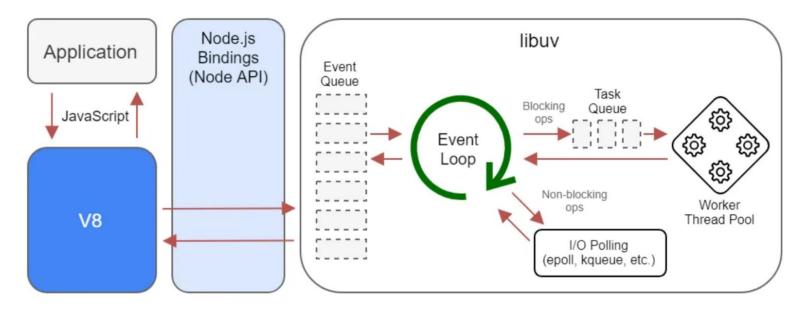




# **Remaining Challenges**

### **1. Cross-language mapping requirement**

- $\checkmark$  JS code layer & C/C++ code layer
- 2. Integration with Node.js framework
  - $\checkmark$  Node.js runs in a single process that creates two kinds of threads.



deeper-node-core-c96b4cec7aa4



### https://medium.com/preezma/node-js-event-loop-architecture-go-



# **Problem Formulation**

The number of all system calls provided by the system: 

 $S_{base} = |SYSCALL_{system}|$ 

The number of system calls in the whitelist: 

 $S_{app} = |SYSCALL_{main-thread}| \cup |SYSCALL_{thread-pool}|$ 

The degree of attack surface reduction in the system call level:

$$SR = \frac{S_{app}}{S_{base}}$$

Goal: minimize the attack surface in the system call level to prevent malicious critical operations, while not affecting the application's normal execution







- Introduction
- **Previous work & Remaining challenges**
- HODOR: system call level protection system for Node.js applications
- **Evaluation**
- **Conclusion & Takeaways**

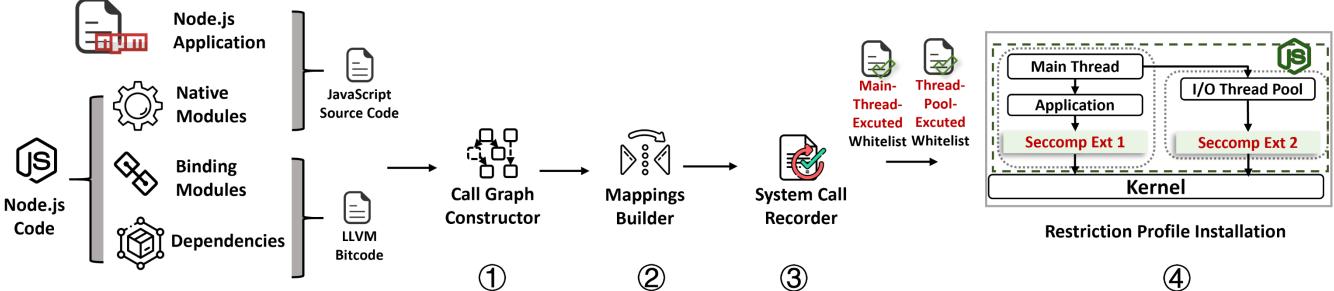






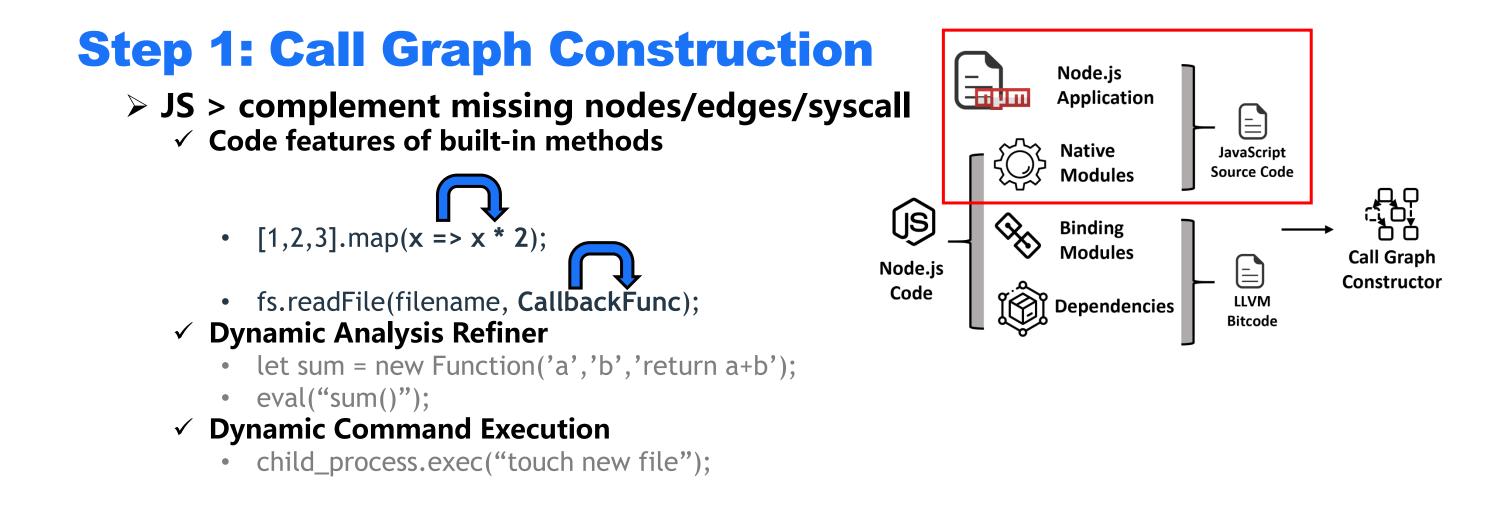
### **Our approach: Hodor**

### A lightweight runtime protection system.







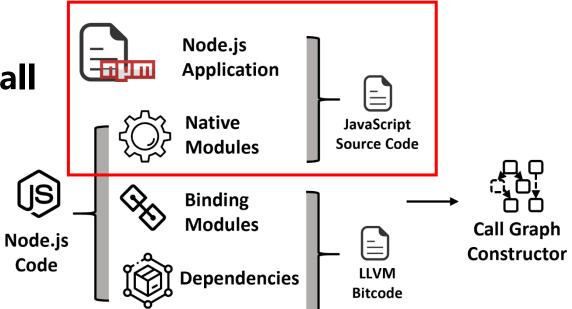






### > JS > complement missing nodes/edges/syscall

- ✓ Code features of built-in methods
  - [1,2,3].map(x => x \* 2);
  - fs.readFile(filename, CallbackFunc);
- ✓ Dynamic Analysis Refiner
  - let sum = new Function('a','b','return a+b');
  - eval("sum()");
- ✓ Dynamic Command Execution
  - child\_process.exec("touch new file");



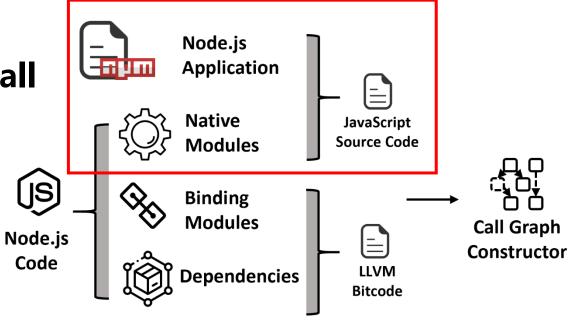






### > JS > complement missing nodes/edges/syscall

- ✓ Code features of built-in methods
  - [1,2,3].map(x => x \* 2);
  - fs.readFile(filename, callbackFunc);
- ✓ Dynamic Analysis Refiner
  - let sum = new Function('a', 'b', 'return a+b');
  - eval("sum()");
- ✓ Dynamic Command Execution
  - child\_process.exec("touch new file"); System Call Required





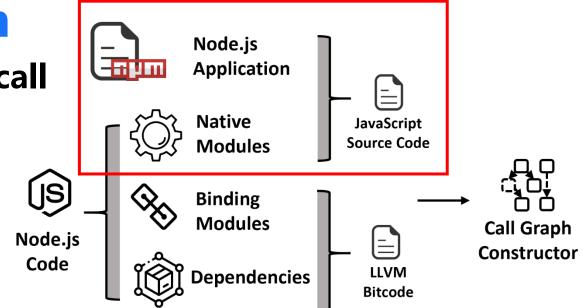


### > JS > complement missing nodes/edges/syscall

- ✓ Code features of built-in methods
  - [1,2,3].map(x => x \* 2);
  - fs.readFile(filename, callbackFunc);
- ✓ Dynamic Analysis Refiner
  - let sum = new Function('a', 'b', 'return a+b');
  - eval("sum()");
- ✓ Dynamic Command Execution
  - child\_process.exec("touch new file");

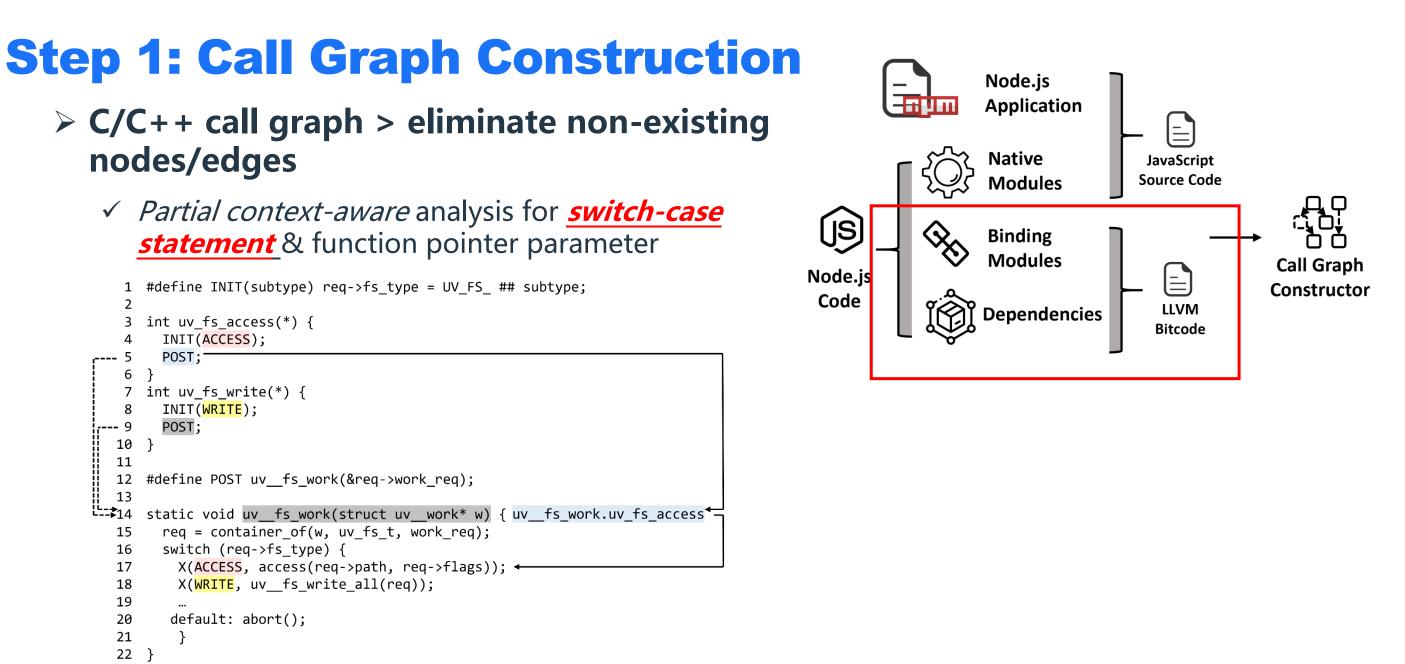
### Implementation

- Reimplement JAM and add in proposed optimizations
  - ISSTA'21 Modular call graph construction for security scanning of node.js applications
- ✓ Combine dynamic call graph tool *Nodeprof* and Linux *strace* utility







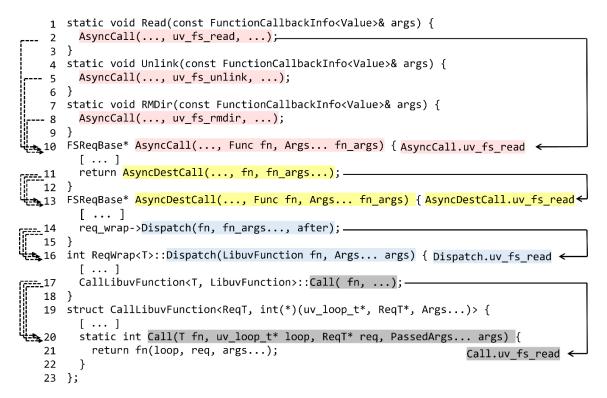






### > C/C++ call graph > eliminate non-existing nodes/edges

✓ *Partial context-aware* analysis for switch-case statement & *function pointer parameter* 



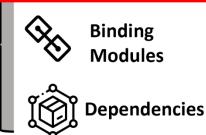
JS

Node.js

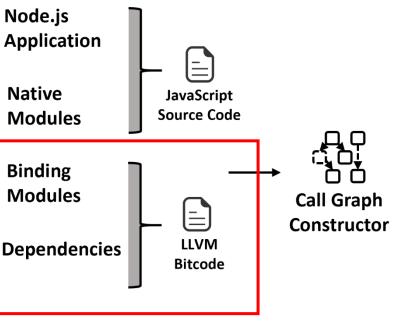
Code

Native **Modules** 

Node.js







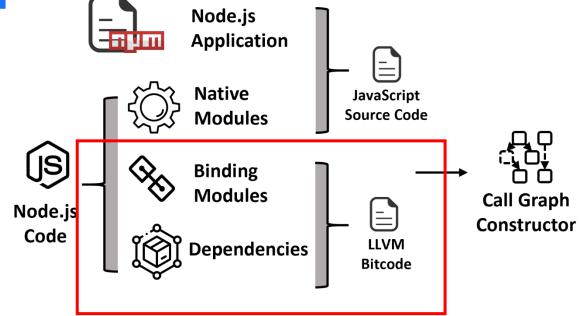


### C/C++ call graph > eliminate non-existing nodes/edges

✓ Partial context-aware analysis for switch-case statement & function pointer parameter

### > Implementation

✓ clang with wllvm > llvm link > SVF + +

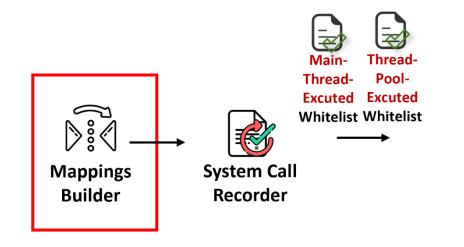






### **Step 2: Mapping Builder**

- > We build *call graph traversal* for call graphs of the Node.js application layer, Binding Module layer, and Dependency layer.
- > We build *LLVM Pass* for the Built-in Module layer.
- > We get mappings of different layers.



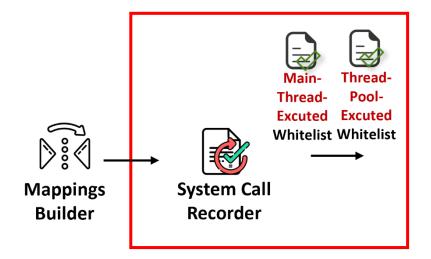
Algorithm 1: Mapping generation Data: Call graphs of builtin modules *cq.builtin*, call graphs of binding modules and dependencies *cq\_bottom*, call graphs of libc *cq.libc*, LLVM IR of binding modules ir bind **Result:** Output mapping dict *M* 1 M.builtin  $\leftarrow$  {}; /\* Mappings of builtin modules \*/ **2** forall  $cq.Module \in cq\_builtin$  do **forall** method  $\in$  module.exports **do** 3 *M.builtin.module.method*  $\leftarrow$  {}; 4 5 Callers *C* invoked by the method by traversing *cq.Module*; 6 forall  $c \in C$  do 7 if c == internalBinding then *M.builtin.module.method*  $\leftarrow$  (*module, method*); 8 9 M.binding  $\leftarrow$  {}; /\* Mappings of binding modules \*/ 10 forall *ir*. Module  $\in$  *ir* bind do forall method\_bind  $\in$  ir.Module do 11 *M.binding.module.method*  $\leftarrow$  *func*; 12 13  $M.depend \leftarrow \{\};$ /\* Mappings of dependencies \*/ 14 forall func  $\in$  M.bindings.module.method do  $M_depend.module.method \leftarrow \{\};$ 15 Callers *C* invoked by the function by traversing *cq.bottom*; 16 17 forall  $c \in C$  do if c == libc then 18  $M.depend.module.method \leftarrow libc;$ 19 20 forall  $libc \in M.depend.module.method$  do *M.depend.module.method*  $\leftarrow$  {}; 21 22 Callers C invoked by the function by traversing cq.libc; 23 forall  $c \in C$  do if  $c \in syscall$  then 24  $M.depend.module.method.libc \leftarrow$  syscall; 25 26 return M.builtin, M.binding, M.depend;





### **Step 3: System Call Recorder**

- Based on mappings, we calculate the <u>system call</u> <u>whitelists</u> for the Node.js application.
- We <u>divide</u> the system call list into the system call list of main thread and the system call list of the thread pool.



Algorithm	2:	Whitelist	ger
-----------	----	-----------	-----

D	ata: Call graph of Node.js Application
R	esult: Output whitelist W
1 W	$l.main \leftarrow \{\};$
2 W	$l.pool \leftarrow \{\};$
3 C	allers C invoked application by trave
4 fo	rall $c \in C$ do
5	if $c \in M$ .native.c then
6	forall $b \in M.native.c$ do
7	if $f \in M.bindings$ .
8	forall $sys \in M$
9	$ $ if $b \in bu$
10	wl.
11	else
12	wl.
13 re	turn W



### neration

on cg\_app, mapping sets M

ersing cg\_app;

### ,

s.b then M.depend.f do uiltin\_threadpool then '.pool ← sys;

.main  $\leftarrow$  sys;



### **Step 4: Hodor Installation**

### Seccomp Implementation

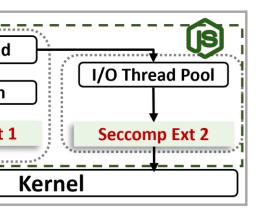
- ✓ For *thread pool required applications*, we *first* install the filter for the thread pool thread and *then* install the filter for the main thread to prevent the thread pool thread from inheriting the main thread filter.
- ✓ For thread pool dis-required applications, we *only* load the main thread filter.

### > Read/write Permission Restrictions.

- Read and write system calls are widely used by Node.js engine.
- *Chroot* mechanism and Switch the ownership.

	Main Threa
Main- Thread-	
Thread- Pool-	Application
Excuted Excuted	<b> </b>
Whitelist Whitelist	Seccomp Ext
$\longrightarrow$	





### **Restriction Profile Installation**





- Introduction
- **Previous work & Remaining challenges**
- HODOR: system call level protection system for Node.js applications
- **Evaluation**
- **Conclusion & Takeaways**





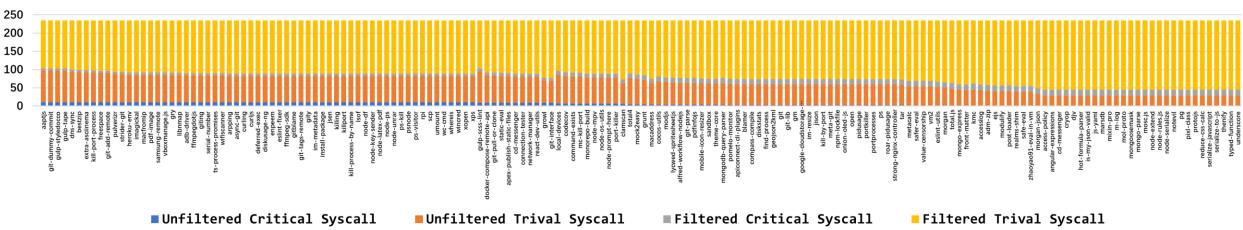
### **Evaluation**

### > Dataset

- ✓ 169 packages suffered from ACE attacks
- ✓ Three large-scale real-world applications (koa, express and json-server).
- ✓ Node.js core tests and 4 well-known web frameworks (koa, fastify, express, and connect).

### > Total Result

✓ HODOR can reduce the attack surface of Node.js applications to <u>19.42%</u>.







### **Evaluation - Call Graph Construction and Resulting Protection**

- ✓ The optimization of JS call graph construction helps identify hidden required system calls for 23.21% packages.
- ✓ The optimization of C/C++ call graph construction further reduces the system call permissions by 71.02%.
- ✓ HODOR reduces the attack surface for the main thread to **19.20%**, for the thread pool thread to 7.73%, while not affecting the application's normal operation.

Туре	# of Package	Node.js	w/ Musl	Hodor				
туре	$\pi$ of 1 ackage	# of CS	# of TS	# of CS	# of TS			
Arb Comm Exec	110	1,161	10,636	910	7,617			
Arb Code Exec	58	243	3,352	110	2,294			
Total	168	1,404	13,988	1,020	9,911			
Туре	# of Package	# of MT	# of TP	# of MT	# of TP			
Arb Comm Exec	110	11,797	2,958	8,444	1,212			
Arb Code Exec	58	3,595	1,218	2,362	496			
Total	168	15,392	4,176	10,806	1,708			

CS: Critical syscalls invocation; TS: Trivial syscalls invocation; MT: Main Thread system calls invocation; TP: Thread Pool system calls invocation;



Image: bold in the set of the se		CVE	I		th Masl Libc	Ho		Node.js wi	ft Masl Libe	He	tor		Sof CL-1	
	Attack Type	CVE	Package Name	# of CS	# of TS	4 of CS	# of TS	# of MT	A of TP	# of MT	# of TP	# of CL		% of CL-2
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2021-23363 CVE-2021-23360	kill-by-port killport:	10	93 99	3	57 72	103	0	60 83	0	6 237	0.01%	85.71% 95.83%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2021-23386 CVE-2018-13797	kill-process-by-name macaddress	11 10	99 93	5	72 58	110	0	83 63	0	6 52	75.00% 36.36%	75.00% 36.36%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2022-25973 CVE-2021-23377	me-kill-port onion-oled-js	10	95 93	3	74 57	105	87	80 60	33	405	34,40%	72.72% 82.35%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2018-3757	pdf-image:	10	93 99	ů.	75	110	87	85	34	194	25.19%	86.15%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2017-1000220 CVE-2021-23379	pidesage: portkiller:	10	93 93	3	57 57	103	0	60 60	0	68 12	62.96% 88.23%	62.96% 88.23%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2021-23359 CVE-2021-23348	port-killer portprocesses:	10 10	95 93	7	72 57	105	0	79 60	0	8	100.00% 88.57%	100.00% 88.57%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2018-16460 CVE-2021-23355	ps ps-kill	10 11	93 99	3	57 72	103	0	60 83	0	28 3	75.00% 12.50%	75.00%
	Arbitrary Command Injection Arbitrary Command Injection	CVE-2021-23374 CVE-2021-23380	ps-visitor rour-pidusage:	11	99 93	3	72 57	110	0	83 60	0	29 62	14,795- 50,81%	76.68%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Command Injection Arbitrary Command Injection	CVE-DUR. TT	sansung-remote sep		99	11	72	110	0	83	0	14	87,50%	87.50% 83.31%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Command Injection Argument Injection	CVE-2021-23399 CVE-2022-24437	wincred git-gull-or-clone	11 10	99 95	11	72	110	87	83 82	0	10	76.92%	76.92%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7636	adh-driver alfred-workflow-molejs	11 10	99 93	4	73 60	110	87 87	83 62	33 34	66 729	39.75% 2.18%	91.30% 86.95%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2018-16462 CVE-2020-7633	apex-publish-static-files apiconnect-cli-plugins	10 10	95 93	9 3	73 57	105	0	82 60	0	8 19,230	7.76% 8.83%	7.76%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2021-3190 CVE-2020-7730	async-git # bestzip	11	99 99	11	72 82	110	87	83 92	35	59 4,204	69.41% 12.60%	74.19% 89.04%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2029-10807 CVE-2020-7795 CVE-2020-7795	cd-messenger	10	95	2	72	103	0	81	0	334	36.77%	\$3.33% \$3.33%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2009-15123	coons-utils codecov	10	93 95	4	64 76	103	87	66 81	34 37	95 1,895	9.17% 8.53%	9.17% 8.53%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7635 CVE-2020-7781 TAB	compass-compile connection-tester	10 10	93 95	3	57 72	103	0	60 81	0	202 43	13.96% 68.25%	95.65% 67.74%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2019-10789 CVE-2020-28425	curling curlis	11	99 99	11	72 72	110	0	83 83	0	52 75	85.24% 66.37%	85.24% 78.12%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-26408	diskstats]	10	93	3	57	103	ő	60	0	55	94.82%	\$7.75% 94.82%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7606 CVE-2013-10301	docker-compose-remote-api		99	9	75	110	87	83	35	1,143	10.99%	88.88% 87.80%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2821-26275 CVE-2821-23376	eslint-fixer: ffrmegdetis	11	99 99	ii ii	72 73	110	0	83 84	0	5,629	10.41%	66.667% 36.95%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection		ffmpeg-sdk 1 find-process	11 10	99 93	3	72 57	110	0	83 60	0	18 110	75.00% 5.56%	75.00% 42.96%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2828-28429	freespace: goojsan2kml	11 10	101 93	3	80 57	112 103	0	91 60	0	27 6	44.26% 100.00%	59.09% 100.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7630 CVE-2020-28434 CVE-2020-2764	git-ant-remote githkane	1	101 99		80 72 87	112	0	83 00	0	13 9 200	05.00% 18.00% 7.6***	65.00% 75.00% 88.32%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2019-10802 CVE-2022-1440	giting git-interface	11	99 93	11	73 60	110	87	83 69	34	197	20.02%	64.56%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	/ CVE-2020-28436	git-tags-remote: google-cloudstorage-commands:	11 10	99 93	3	72 57	110	0	83 60	0	236 13	32.14% 52.00%	100.00% 52.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2017-16042 CVE-2020-36650	grunt 877	10 11	93 99	9	61 74	103	0 87	70 83	0 34	35 190	45.45% 59.00%	45.45% 88.23%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7601 CVE-2020-7607	gulp-scss-lint: gulp-styledocon	11	99	9 11	86 86	110	87	89 91	42	5,065	9,60%	40.54%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection	CVE-2020-28437 CVE-2019-10788	heroku-env	ii i	99 99	ii	75	110	87	85	34	25	33.33%	42.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2019-10787 CVE-2020-7629	im-resize install-package:	10	93 99	3	57 72	103	0	60 83	0	85 42	19.63% 93.33%	85.71% 93.33%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-8178 CVE-2021-23381	jisen killing†	10 11	93 99	3 11	57 72	103 110	0	60 83	0	1,916 38	9.01% 9.76%	61.24% 90.24%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2013-15609 CVE-2013-16461	kill-port-process libernap	11	101 99	11	81 74	112	87	91 83	33 34	299 3,069	20.45%	56.75% 80.93%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2019-10783	local-devices loof	10	93 99	ů.	79 72 61	103	87 0 87	83 63	57 0 35	37 54	37,37% 91,30% 14,02%	55.73% 91.30%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7786 CVE-2020-25434	macfromip monorepo-build	11	99 99 95	11	75	110	87	85 79	35	39	45.88%	45.88%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2019-10786 CVE-2019-15597	network-manager node-df	10	95 90	9 11	72 72	105	0	81 83	0	93 84	89.42% 1.71%	93.00% 91.30%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2820-7627 CVE-2820-28433	node-key-sender: node-latex-pdf	11	99 99	11	72 72	110	0	83 83	0	93 9	77.50% 60.00%	77.50% 60.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7632 CVE-2020-7602	node-prompt-here	10 10	95 95	7	72 72	105	0	79	0	36	14.63% 6.65%	14.63% 57.14%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2828-7788	node-unrar node-unrar	11	99	11	72 72 57	110	0	83 83	0	17	75.51% 8.21% 7.01%	75.51% 100.00% 70.96%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2021-23375 CVE-2020-7604	penode	11	99 99	Ĥ	72 77	110	87	83 87	0	63 614	6.12%	35.13%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2021-24033 CVE-2013-10796	react-dev-utils rpi	10 11	95 99	9 11	72 72	105 110	0	81 83	0	6,425 28	11.77% 65.11%	22.32% 65.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2019-10804	serial-number strider-git	11	99 99	11	73 77	110	87 87	83 87	33 36	45	57.69% 24.82%	57.69% 42.40%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7621 CVE-2020-28432	strong-nginx-controller theme-core:	10	93 93	3	57	103	87	60 60	33	17,994	6.37%	49.70% 81.35%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-7628	unount showness is	ii i	99	ii	72	110	0	83	0	965	8,20%	71.73%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection Command Injection	CVE-2020-28431 CVE-2020-15362	weemd	ii ii	99 99	ii ii	72 73	110	0 87	83 83	0 33	31 215	0.44%	82.50% 83.3.3%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Command Injection	CVE-2020-28447	sopen;									2	90.00%	90.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution	CVE-2428-36378	amptis arpping:	ii ii	101 99	11	87 72	112	0	98 83	0	457 98	13.00% 27.68%	97.05% 78.40%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution	CVE-2020-11079 CVE-2021-23632	dau-sync git	11	93	3	83 57	112	87	93 60	35	344 648	27.80% 15.72%	97.2.2% 15.89%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution	1	git-nb; git-parse	10	93	4	60	103	87	62	34	704	2.96%	39.51%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution	1	imagickal) meta-git	11	99 93	11	75 57	110	87	85	35	1,183	13.13% 8.32%	83.16% 89.02%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution	CVE-2828-7628	node-as-utils1 pomelo-monitor	10 10	95 93	7	72 58	105	0	79 61	0	172 90	47.77% 86.53%	47,77% 86.53%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Execution	CVE-2828-7729	grunt			5						6,747	7.69%	47.13%
Arbitrary Command Execution 1,161 10,636 900 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Execution Arbitrary Code Execution	CVE-2828-7777	is-my-json-valid jsm	<u>.</u>	28	1	28 72	29 110	0	29 83	0	3,560	9.92% 34.56%	93.84% 95.15%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Execution Arbitrary Code Execution	CVE-2020-7675 CVE-2017-16082 CVE-2020-7640	pg pg		28		28	29	0	29	0	260	82.75% 12.26%	27.54%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Execution Arbitrary Code Execution	CVE-2022-0748	post-loader serialize-to-js	6	83 28		40 28	89 29	0	41 29	0	900	8.64% 5.10%	69.23% 86.33%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Execution Arbitrary Code Execution	CVE-2021-23389 CVE-2021-23390	total.js total4	6 10	83 94	2 5	46 72	89 104	87 87	44 75	35 38	2,072	6.59% 6.79%	6.55% 6.79%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Execution Arbitrary Code Injection	CVE-2017-1001064	typed-function kmc	6	28 83		28 45	29 89	0	29	0	397 2,267	74.62%	74.62% 55.81%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Injection Arbitrary Code Injection Arbitrary Code Injection	1	marson mixin-pro m-los	i .	28		28 28 28	29	0	29	0	78	82.10% 0.88%	82.10%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Injection Arbitrary Code Injection	1	mobile-icon-resizer mock2easy	10	93 93	4	59 70	103	87 87	61 73	35 35	54	33.75%	65.51%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Injection Arbitrary Code Injection	1	modijs modulity	10 6	93 83	4	62 40	103	87 0	64 41	34 0	167 1,133	2.11% 1.97%	6.77% 80.70%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Injection Arbitrary Code Injection	1	mongoosemask		28		28	29	0	22	0	50	19.92%	67.56%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary Code Injection Arbitrary File Overwrite	CVE-2020-7660 CVE-2021-22803	serialize-javascript	i	28		28	29	0 87	29	0	71	5.10%	93.42%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Arbitrary File Write Code Execution	CVE-2018-1002204 CVE-2017-5941	adm-zip) mode-serialize	6	83 28		41 28	89 29	0	42 29	0	516 52	26.81% 92.85%	26.84% 92.85%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2022-25760 CVE-2020-7674	accessing access-policy	6	83 28		43 28	89 29	0	44 29	0	95 81	41.48% 0.26%	83.33% 96.42%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2020-7675	ed-messenger		28		28	29	0	29	0	374	48.01%	4K.01% 90.69%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2019-15687 CVE-2021-23639	eslint-utils front-matter	6	83	2	50 42	89	87 87	48 36	35	436	8.92%	80.89%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2020-6836	hot-formula-parser js-yumi	1	28 28		28 28	29 29	0	29 29	0	1,650	6.25% 19.62%	89.94% 70.66%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2022-21122 CVE-2019-5413	metacale morgan	7	90 83	2	52 48	97 89	0 87	54 46	0 35	59 2,135	54.12% 11.69%	86.95% 81.48%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2022-25921 CVE-2020-7672 CVE-2020-7672	morgan-json mosc js	1	83 28		32	89 29	0	29	0	261 47 0.10	0.93%	94,115 88,675 91,595
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Code Injection Code Injection	CVE-2016-10548 CVE-2020-7677	reduce-css-cale thenify		28 28		28 28	29 29	0	29	ő	110	\$7.27% 56.81%	87.27% 82.14%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Prototype Pullution Prototype Pollution	CVE-2020-7743 CVE-2021-23594	math js realmo-shim	6	28 83		28 38	29 89	0	29 39	0	2,650 1,118	9.83% 3.59%	8.52% 78.18%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution		djv mongodh-query-parser	7	28 90	1	28 60	29 97	0 87	29 59	35	300 2,688	33.44% 5.32%	55.35% 28.49%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Remote Code Execution Remote Code Execution	CVE-2019-10758 CVE-2020-24391	mongo-express mongo-parse	1	28		43 28	29	0	42 29 50	0	342	83.82% 9,44%	23.58% 83.82% 89.5mm
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Sandbox Breakout Sandbox Breakout	CVE-2019-10769	safer-eval sandbox	7	90 93	2 4	52 58	97 103	0 87	54 60	0 33	119	35.31% 48.88%	80.76% 48.88%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Sandbox Breakout Sandbox Breakout	1	static-eval value-censorship	7	99 90	9	74 52	110 97	87 0	82 54	34 0	2,891 3,089	7.69% 8.43%	75.51% 95.00%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Sandbox Bypass Sandbox Escape Sandbox Escape	CVE-2019-10761 CVE-2020-7710 CVE-2020-7710	vm2 safe-eval	6	90 83	2	52 38	97 89	0	54 39 10	0	11	0.09% 95.15%	53.33% 95.15%
Arbitrary Command Execution 1,161 10,636 910 7,617 11,797 2,958 8,444 1,212 936 35.61% 69.14%	Template Injection Template Injection	CVE-2022-29078 CVE-2021-23358	ejs underscore	6	83 28		40 28	89 29	0	41 29	0	32 239	5.22% 48.38%	87.33% 48.38%
Arbitrary Cole Execution   243   3,352   110   2,294   3,595   1,218   2,362   496   1,178   30,83%   71,03%	Ar	bitrary Command Execut	lion		10,636		7,617	11,797		8,444		9.36	35.61%	69.14%
		Arbitrary Code Execution	•	243	3,352	110	2,294	3,595	1,218	2,362	-256	1,178	30.83%	71.03%



### **Evaluation - Exploit Mitigation**

- We construct different advanced attack payloads to simulate various dangerous behaviors of attackers, where a variety of critical system calls can be invoked.
- ✓ HODOR could effectively mitigate the execution of **73.59%** exploits.

Critical Sys	scall	JavaScript	C*
Cmd Execution	exec	child_process.exec	execve
Cind Execution	fork	child_process.fork	fork
Permission	setgit	process.setuid	setuid
rermission	setuid	process.setgid	setgid
	bind	net.connect	bind
Network	connect	dgram.createSocket.bind	connect
	listen	server.listen	listen

 Table 5: Vulnerability payloads.

\*: We compile the C code into binary and execute the binary as a shell.

			MIR														
			C	md	Perm	nission	Net			I.	Cmd Permission			ission	Net		
Package Name	Initial Attack	Initial Attack	Exec	Fork	Setgid	Setuid	Connect	Listen	Bind	Initial Attack	Exec	Fork	Setgid	Setuid	Connect	Listen	B
accesslog	Print xSS	×	~	~	~	~	~	~	~	1			1				
access-policy	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
adm-zip	Path traversal	~	~	~	~	~	~	~	~				1				
ngular-expressions	Write file angular-expressions-success	~	~	~	~	~	~	~	~				1				
cd-messenger	Print JHU	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
cryop	Print defconrussia	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
djv	touch HACKED	~	~	~	~	~	~	~	~	0.000			1				
ejs	Write file ejs-success	~	~	~	~	~	~	~	~				1				
eslint-utils	Write file eslint-utils-success	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
front-matter	Print 1	×	2	~	-	~	~	~	~								
grunt	Returns Date.now	2	×	-	2		-	-	-	~	~	~	~	~	~	~	
	Write file test	-	2	<b>^</b>	-		2	-	-	-	-	2	-		2	2	
ot-formula-parser		-		-				2	2		~	~			~	~	
is-my-json-valid	Execute cat /etc/passwd		~	~	~	-	~						/				
jsen	Write file malicious	~	×	×	×	×	×	×	×				/				
js-yaml	Returns Date.now	~	~	~	~	~	~	~	~				1				
kmc	Write file kmc-success	~	~	~	~	~	~	~	~	~	V	~	~	~	~	~	
marsdb	Write file marsdb-success	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
metacalc	Print process	×	~	~	~	~	~	~	~				1				
mixin-pro	Print hacked	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
m-log	Print injected	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
mock2easy	Write mock2easy-success	×	~	-	~	-	×	×	×	~		~	~	~	~	~	
modis	Write modis-success.txt	×	×	×			×	×	×	~	~	~	~		~	~	
modulify	Print hacked	2	2	2			2	2	2	-	-	2	2			-	
		×	2	-	2		-	-	-	-							
mol-proto	Write file mol-proto-success	2	2		-	2										~	
ngodb-query-parser	touch test-file			~			×	×	×				1				
mongo-express	exec calculator	~	~	~	~	~	~	~	~				/				
mongoosemask	Print "my evil code was run"	×	~	~	~	~	~	~	~				/				
mongo-parse	Write file hacked	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
morgan	Write file mongui-success	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
morgan-json	Print GLOBAL CTF HIT	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
moscis	Write file Song	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
node-extend	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
node-extend	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
node-rules.js	Print 123	×	~	~	~	-	~	-	~	~	~	~	~	~		~	
node-serialize	Execute Is	x	-	-	-	2	-	-	-	-	2	-	~	-		-	
node-serialize	Print pwned	2	2	-	-		-	-	-	-	-	-	-	-		-	
											-						
Pg	Print process.env	×	~	~	~	~	~	~	~	~		~	~	~	~	~	
pixl-class	Print 123	×	~	~	~	-	~	~	~	~	~	~	~	~	~	~	
post-loader	Print rce	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
protojs	Write file protojs-success	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
realms-shim	Messed with Object.toString	~	~	~	~	~	~	~	~	~	~	~	~	~	~	V	
reduce-css-calc	Read /etc/passwd	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
safe-eval	Return proces	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
safer-eval	Print id	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
sandbox	Print process.pid	×	×	×	~	~	~	~	~	~	×	×	~	~	~	~	
erialize-javascript	Print 1	×	2	2	-	-	-	-	-	~	~	2	~	2	~	~	
serialize-to-js	Execute Is	x	2	-	2	-	-	-	-	-	-	2	-	-	-	-	
static-eval	Print hacked	2	×	-	-		2	-	-	-	-	-	~		-	-	
			2						2							2	
tar	Overwrite file	~					~	~		~	~			~		~	
thenify	Write file Song	~	~	~	~	-	~	~	~				1				
total.js	Touch HACKED	×	~	~	~	~	×	×	×				1				
total4	Touch HACKED	~	×	×	~	~	×	×	×				1				
typed-function	Execute whoami	~	~	~	~	~	~	~	~				1				
underscore	touch HELLO	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
value-censorship	Access the Function constructor	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
vm2	return process.env	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	
	return process.env	×	~				~	~	~	~	~	~	~	~	~	~	
aoyao91-eval-in-ym																	

### Table 7: Exploit execution for packages with arbitrary code execution attacks.



Table 6: Exploit execution for packages with arbitrary command execution attacks.

		HODOR Cmd Permission Net										
ickage Name	Initial Attack	Initial Attack	Exec	nd Fork	Setgid	Setuid	Connect	Listen	Bind			
nmand-exists	Write command-exists	~	×	×	Jeegau ✔	~	×	~	~			
nmand-exists all-by-port	Write success Write success	~	×	×	2	2	~	2	2			
killport rocess-by-name	Write success	~	â			2	5	5	~			
nacaddress nc-kill-port	Write /tmp/poof Write newFile.txt	~	×	×	~	~	<u> </u>	~	~			
	Write success	~	× × ×	* * * *	2	2	2	2	ž			
open pdf-image	Write /tmp/tada	×.	×	×	~	~	~	~	~			
pdf-image pdfinfois	Write /tmp/hacked Write a	~	x	×	2	ž	5	5	ž			
pdfinfojs pidusage	Write a Execute /usr/local/bin/python	~	×	×	~	~	~	~	~			
portkiller port-killer	Write success Write success	~	×	×	2	2	ž	5	2			
ortprocesses	Write success Write success.txt	~	×	×	~	~	~	~	~			
ps ps-kill ps-visitor	Write success.txt Write success	~	×	×	~	~	2	2	~			
ps-visitor	Write success	-	×	Ŷ	2	2	2	2	2			
ar-pidusage		~	×	×	~	<u>,</u>			~			
scp	Write /tmp/malicious; nc localhost 4444;	-	×	* * * * * *	2	Ŷ	2	2	~			
scp whereis	Write /tmp/tada	~	×	×	~	×.	~	~	~			
wincred pull-or-clone adb-driver workflow-nodejs ublish-static-files	Write success Write /tmp/pwn3 Write a	-	×	x	2	2	×	2	2			
adb-driver	Write a	~	×	×	~	~	×.	~	~			
workflow-nodejs ublish-static-files	Write hacked Write anex-nublich-static-files	~	×	×	2	2	ž	2	2			
anect-cli-plugins	Write a Write hacked Write apex-publish-static-files Write Song Write HACKED #	~	x	x	~	~	2	~	~			
asyne.gif	Write HACKED #	~	×	×	×	×	×	×	×			
bestzip blamer	Write bestzip Write vulnerable	~	×	×	2	~	2	2	2			
-messenger clamscan	Write JHU	~	×	×	~	~	×	~				
cramscan cocos-utils	Write create.txt Write hacked	~	×	×	~	~	ž	2	2			
codecov	Write codecov	~	×	×	~	~	×	~	~			
nection-tester	Write JHU Write 111 Write JHU	~	×	×	ž	2	×	ź	2			
curling	Write JHU	~	×	×	2	~	÷	~	1			
curljs ferred-exec	Write JHU Write IHU	~	×	×	2	~	2	2	~			
diskstats	Write JHU Write HACKED Write Song	~	x	x	2	~	2	2	~			
iskusage-ng	Write Song		××	××	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\	\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\						
ompose-remote-api enpeem	Write vulnerable.txt Write create.txt	~	×	×	2	2	ž	ž	ž			
enpeem eslint-fixer Impegdotjs Tmpeg-sdk nd-process	Write eslint-fixer Write success	×.	x	x	×	×	2	×.	~			
impegdotjs Impeg.edk	Write success	~	×	×	2	2	5	2	2			
nd-process	Write success Write /tmp/semicolon_file	~	×	×	~	~	~	~	~			
freespace eojson2kml -add-remote	Write /tmp/semicolon_file Write JHU	~	×	×	~	~	×		~			
-add-remote	Write Song	2	×	×	2	2	×	2	2			
githlame	Write JHU	×.	×	×	~	×.	~	×.	~			
ummy-commit giting	Write git-dummy-commit Write create.txt	~	×	×	2	2	ž	2	2			
it-interface -tags-remote	Write /tmp/pwned	~	×	×	~	~	~	~	~			
-tags-remote idstorage-commands	Write /tmp/command-injection.test Write JHU	~	×	* * * * *	2	2	2	2	2			
growl	Write aaaa Write HACKED	-	x	Ŷ	2	2	2	2	2			
	Write HACKED	~	×	×	~	~	<u> </u>	<u>.</u>	٢			
gry ulp-scss-lint p-styledocco gulp-tape eroku-env n-metadata im-resize tall-package jison källing	Write create.txt Write Vulnerable	~	×	×	2	2	x	x	â			
gulp-tape	Write JHU.txt Write JHU	~	×	×	×	×	×	×	×			
n-metadata	Write JHU Write im-metadata	~	×	×	×	×	×	×	×			
im-resize		~	×	×	~	~	~	~	~			
tall-package	Write Song Write pwned	~	×	* *	2	2	2	2	2			
killing	Write success	~	×	x	2	~	2	2	~			
-port-process libnmap	Write kill-port-process Write success.txt		×	* * * * *	~	<u>,</u>	×		~			
cal-devices	Makek directory attacker Write create.txt	-		â	2	2	2	2	2			
	Write create.txt	~	×	×	×	×	~	×.	~			
d-spritesheetjs nacfromip	Write 111233 # Write JHU2	~	×	×	2	×	5	5	2			
sor sd-spritesheetjs nacfromip morepo-build sork-manager	Write JHU	~	×	×	~	~	×	~	~			
ork-manager node-df	Write create.txt Write HACKED	~	×	* * *	ž	2	×	ž	5			
node-df le-key-sender		~	×	x	2	2	2	2	~			
de-latex-pdf	Write JHU Write JHU	×.	×	×	×	×	×	×	×			
-prompt-here	Write JHU Write create.txt	~	×	×	2	2	×	2	2			
node-ps	Write create.txt Write JHU	~	×	×			~		1			
-prompt-here node-ps ode-unrar pm-lockfile	Write node-unrar Write rce	~	×	* * * * * *	ž	ž	ž	ž	2			
psnode pulverizr	Write success	~	×	×	~	V	~	~	~			
	Write Song Write react-dev-utils	~	×	×	2	~	ý	2	2			
rpi rial-number	Write vulnerable.txt Write create.txt	~	, x	x	2	-	2	2	-			
rial-number	Write create.txt	~	×	×	×	×	č	č	ć			
strider-git nginx-controller heme-core	Write HACKED; Write Song	~	×	* * *	2	2	÷	2	Ŷ			
heme-core	Write JHU Write JHU	~	×	×	~	~	~	~	~			
ocess-promises umount oxmanage.js	Write JHU Write Song	~	×	×	×	×	2	2	2			
oxmanage.js	Write Song Write HACKED	~	×	××	~	~	2	~	~			
we-end	Write IHU		×	×	~	~						
wifiscanner	Write /tmp/exploit.txt Write JHU	×.	×	×	×.		×.	×	~			
	Write HACKED	2	×	×	2	2	2	2	2			
aaptjs	Write HACKED Write HACKED	~	× × ×	×	×	×	×	×	x			
xps aaptjs arpping dns-sync git	Write HACKED Write pwned	*****	×	×	```````````````````````````````	``````````````````	、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、、	``````````````````````	````			
git		×	×	* * * * * * *	2	-	Ŷ	2	-			
git-lib	Write HACKED; Write HACKED	~	×	×				~	~			
git-parse gity	Write HACKED Write HACKED Write HACKED	2	x	x	2	2	2	2	2			
imagickal	Write HACKED	×.	×	×	~	~		~	~			
meta-git node-os-utils	Write HACKED Write DUMMY_FILE	****	×	×	2	2	×	2	2			
gm	Write Song Write gm	×	×	×	~	~	~	~	~			



### **Evaluation - Comparison with Other Techniques**

✓ HODOR can defend against a wider spectrum of attacks (additionally covering) arbitrary command execution) with less runtime overhead.

Table 7: Exploit execution for packages with arbitrary code execution attacks.

					HOD	OR			2	1			MI	2			
D. J N	F. 101 A. 444 A.		Cmd		Perm	nission		Net		1	Cmd		Permission			Net	
Package Name	Initial Attack	Initial Attack	Exec	Fork	Setgid	Setuid	Connect	Listen	Bind	Initial Attack	Exec	Fork	Setgid	Setuid	Connect	Listen	Bind
accesslog	Print xSS	×	~	~	~	~	~	~	~	1			1				_
access-policy	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
adm-zip	Path traversal	~	~	~	~	~	~	~	~				1				
angular-expressions	Write file angular-expressions-success	~	~	~	~	~	~	~	~				1				
cd-messenger	Print JHU	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
cryop	Print defconrussia	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
djv	touch HACKED	~	~	~	~	~	~	~	~	- CONS.			1				
eis	Write file eis-success	~	~	~	~	V	~	~	~				1				
eslint-utils	Write file eslint-utils-success	~	~	~	~	~	~	~	~	~	V	~	~	V	~	~	~
front-matter	Print 1	×	~	~	~	~	~	~	~				1				
grunt	Returns Date.now	~	×	×	~	~	~	~	~	~	~	~	~	V	~	~	~
hot-formula-parser	Write file test	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
is-my-json-valid	Execute cat /etc/passwd	~	~	~	~	~	~	~	~				1				
jsen	Write file malicious	~	×	×	×	×	×	×	×				1				
js-yaml	Returns Date.now	~	~	~	~	~	~	~	~				1				
kmc	Write file kmc-success	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
marsdb	Write file marsdb-success	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
metacalc	Print process	×	~	~	~	~	-	~	~	1.5			• /				
mixin-pro	Print hacked	×	-	-	~	-	-	~	-	~	-	1			~	~	~
m-log	Print injected	×	-	~	~	~	-	-	-	-	-				-	-	
mock2easy	Write mock2easy-success	x	2	2	~	2		×	×	-	2				2		
modjs	Write modis-success.txt	Ŷ	×	×	~	2	0	×	x	-	-				-		
modulify	Print hacked	2	2	2	~	-	2	2	2	-					-		
		×	-	2	-	2		-	2								
mol-proto	Write file mol-proto-success touch test-file	2	-		~	-		×	×								
mongodb-query-parser		5	-		-	2	-	2	2								
mongo-express	exec calculator	×	-	~	~	2							1				
mongoosemask	Print "my evil code was run"	×		~			-	~	~	202	1.20		/			100	1121
mongo-parse	Write file hacked	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
morgan	Write file mongui-success	×	~	~	~	~	~	~	~	~	~	~	~	~	~	-	~
morgan-json	Print GLOBAL CTF HIT	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
mosc.js	Write file Song	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
node-extend	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
node-extend	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
node-rules.js	Print 123	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
node-serialize	Execute Is	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
notevil	Print pwned	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
pg	Print process.env	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
pixl-class	Print 123	×	~	~	V	~	~	~	~	~	~	~	~	~	~	~	~
post-loader	Print rce	~	~	~	V	~	~	~	~	~	~	~	~	~	~	~	~
protojs	Write file protojs-success	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
realms-shim	Messed with Object.toString	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
reduce-css-calc	Read /etc/passwd	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
safe-eval	Return proces	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
safer-eval	Print id	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
sandbox	Print process.pid	×	×	×	~	V	~	~	~	~	×	×	~	~	~	~	~
serialize-javascript	Print 1	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
serialize-to-js	Execute Is	×	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
static-eval	Print hacked	~	×	×	~	~	~	~	~	~	~	~	~	~	~	~	~
tar	Overwrite file	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~	~
thenify	Write file Song	~	~	~	~	~	~	~	~		-		1	-	-		-
total.is	Touch HACKED	×	-	2	~	-	×	×	×				1				
total4	Touch HACKED	2	×	×	~	~	×	×	×				1				
typed-function	Execute whoami	2	2	2	~	2	~	2	2				1				
underscore	touch HELLO	×	2	2	~	2	~	2	-	~	~	~	~	~	~	~	~
value-censorship	Access the Function constructor	2	-	2	2	2	-	-	2	-	-	-	~	-	2	~	2
varue-censor ship vm2	return process.env	×	-	2	~	-	~	-	~	-	-	~	~	~	2	~	~
zhaoyao91-eval-in-ym	return process.env	x	-	2	~	-	-	2	2	-	-	-	2	-	2	2	-
mobile-icon-resizer	Print hacked	×	×		-	2	-	2	2	-	-	-	-	2	-	2	-
mobile-teon-resizer	i int nackeu	^	•	<b>^</b>													

¥: Exploits are executed; ✔: Exploits are blocked;





### **Evaluation - Runtime Overhead**

✓ The runtime overhead of HODOR is **0.61%** for **Node.js core tests**, **2.80%** for the web framework, and 0.39% for all 168 packages.

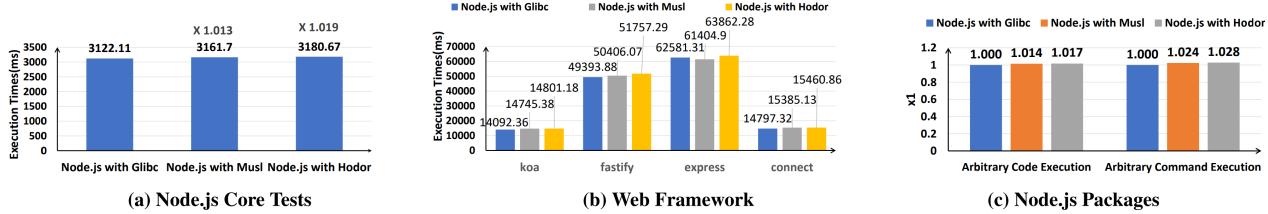


Figure 8: Runtime overhead of Node.js core tests, web framework and applications under the protection of HODOR (RQ4).







- Introduction
- **Previous work & Remaining challenges**
- HODOR: system call level protection system for Node.js applications
- **Evaluation**
- **Conclusion & Takeaways**





# **Conclusion & Takeaways**

1. Attendees will learn a new call graph building methods for JavaScript code and C/C++ code.

2. Attendees will gain knowledge of a novel protection mechanism for Node.js applications, focusing on thread-level and system call-level security.

Attendees will develop an understanding of the hazards associated with vulnerabilities 3. in the Node.js application ecosystem, with a particular emphasis on system call-level vulnerabilities.



# **blackhat** EUROPE 2023

# Thanks & Questions?

Wenya Wang, Xingwei Lin 🈏 @xwlin\_roy

