Juicing V8: A Primary Account for the Memory Forensics of the V8 JavaScript Engine

By:
Enoch Wang (University of New Haven), Samuel Zurowski (University of New Haven), Orion Duffy (University of New Haven), Tyler Thomas (University of New Haven), and Ibrahim Baggili (University of New Haven)

From the proceedings of

The Digital Forensic Research Conference

DFRWS USA 2022

July 11-14, 2022

TECH THROUGH
THE LENS OF
SECURITY

Juicing V8: A Primary Account for Forensics of the V8 Javascript Engine

Enoch Wang
Samuel Zurowski
Orion Duffy
Tyler Thomas
Dr. Ibrahim Baggili

University of New Haven
CONNECTICUT INSTITUTE OF TECHNOLOGY
UNHcFREG

This material is primarily based upon work supported by Centers of Academic Excellence (CAE), National Security Agency (NSA) and Department of Defense (DoD) under grant H98230-20-1-032. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Security Agency or Department of Defense.

Support

AUTHOR INFORMATION

University of New Haven

Enoch Wang
B.S. Computer Engineering
M.S. Cybersecurity, Class 2022
ewang3@unh.newhaven.edu

Samurai Zurowski
B.S. Computer Science
M.S. Cybersecurity, Class 2023
szuro1@unh.newhaven.edu

Orion Duffy
B.S Computer Science, Class 2022
oduff1@unh.newhaven.edu

Tyler Thomas
B.S Computer Science
M.S. Cybersecurity, Class 2022
tthom10@unh.newhaven.edu

CORRESPONDING AUTHOR

Ibrahim (Abe) Baggili, Ph.D.

Ibrahim "Abe" Baggili is an internationally recognized expert in cyber security and digital forensics. He is the founder and director of the University of New Haven's Cyber Forensics Research and Education Group.

Dr. Baggili works regularly with law enforcement on digital forensics to help solve crimes, and he created the Artifact Genome Project, a digital information resource for law enforcement. His work has been supported by the National Security Agency and the MITRE Corporation, which funded a project entitled "Survey for Automation of Child Abuse Investigations."

Agenda

- Introduction
- Motivations and Goals
- Initial Research and Findings
- Methodology
- Discussion and Tool Evaluation
- Conclusion and Future Work

Introduction

- What is V8?
  - Google's open source interpreter that enables Javascript within Chrome and other software
  - Executes JavaScript but also manages it in a highly optimized manner
  - Handles call stack, memory heap, garbage collection, and objects
  - Written in C++
  - Constantly being optimized
- What uses V8?
  - Chrome, Every Chromium Browser, Discord, Node.js, etc

Motivations and Goals

Motivation

- Tyler's previous publication *Memory FORESHADOW* [1], explored the memory forensics of crypto wallets (Electrum) runs on V8
- 99% of web-browsers use Javascript [2]
- Assist investigators in obtaining more from live memory
- Ali-Gombe's *DroidScraper* [3], was able to leverage the mechanics of Garbage Collection (GC) in Android devices to extract objects

Goals

- Research and reverse engineer the V8 Javascript Engine
- Identify the data structures
- Find V8 objects in live memory
- Create a volatility tool capable of extracting V8 objects
- Evaluate the tool

[1] https://www.sciencedirect.com/science/article/pii/S2666281720302511
[2] https://ieeexplore.ieee.org/document/6189228
[3] https://www.usenix.org/conference/raid2019/presentation/ali-gombe

Initial Research and Findings

- Reverse Engineering V8
  - Reading the Source Code [1]
    - V8 Isolate
    - Object Maps
    - MetaMap
      - Root map that is referenced by all object maps
      - References it's own address in it's data structure
      - Identifiable by the magic bytes *FF 03 (20 - 40)*

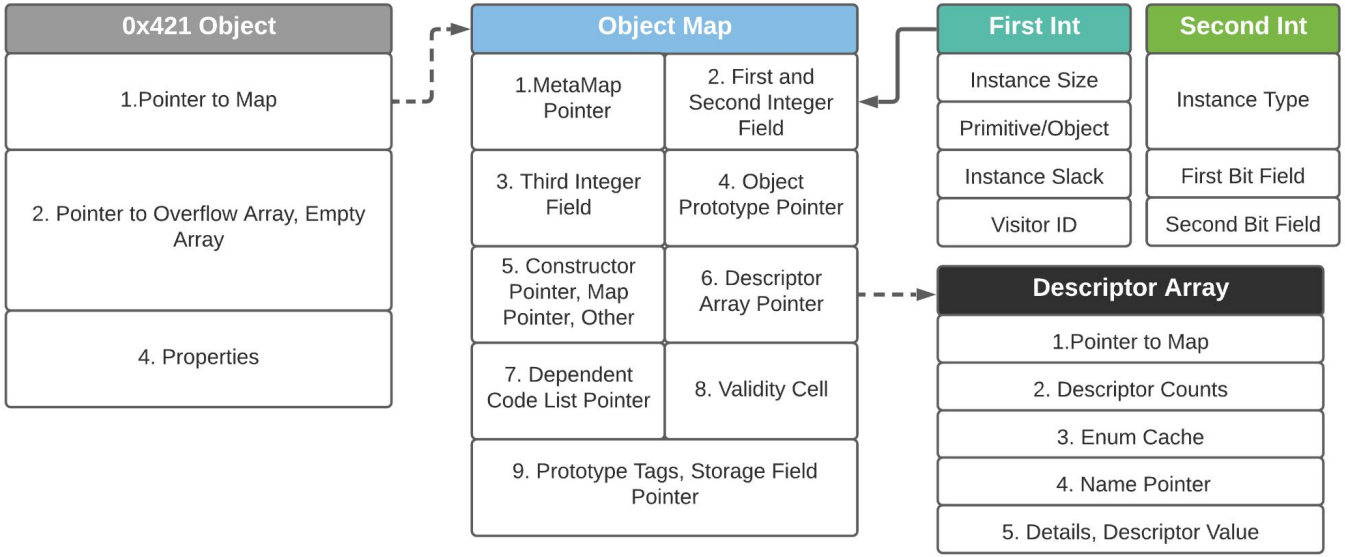
[1] https://github.com/v8/v8

Initial Research and Findings

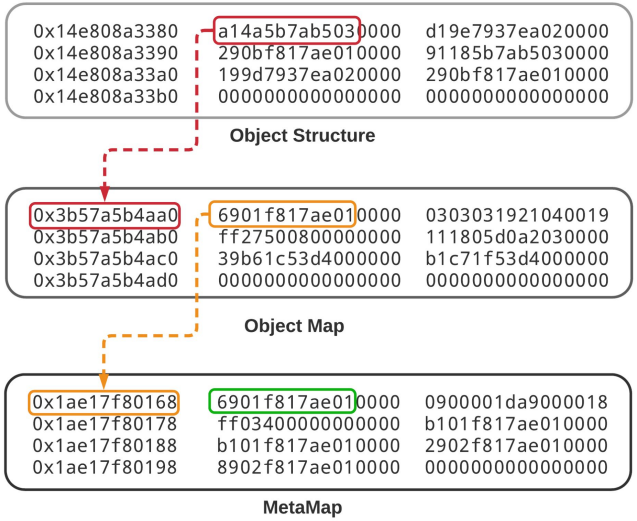
- Reverse Engineering V8
  - llnode [1]
    - Node.js debugging tool for developers
    - Breaks down high-level data types into low-level primitives
    - *compile-link-execute*
    - Similar to NSA's *Ghidra*
    - Run a live instance of V8 and observe data structures
      - Validate our findings from reading the source code
  - Cheat Engine / Volatility

[1] https://github.com/nodejs/llnode

V8 Data Structures



V8 Live Memory Layout

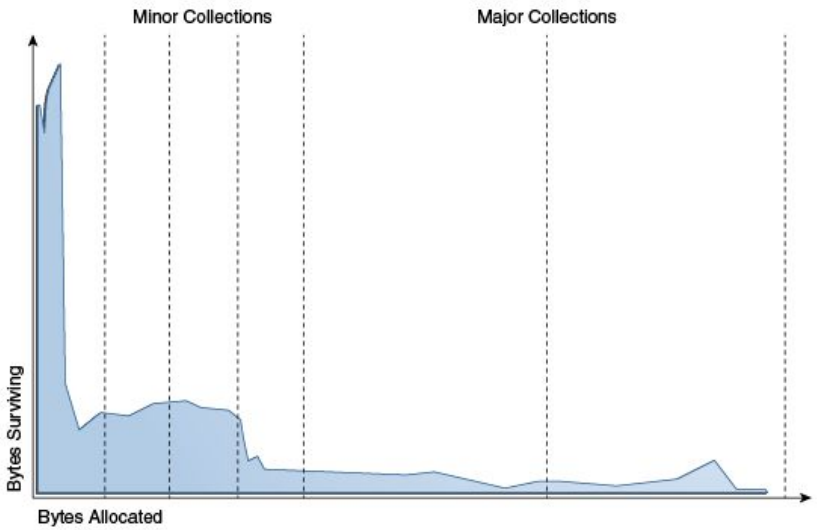


Initial Research and Findings

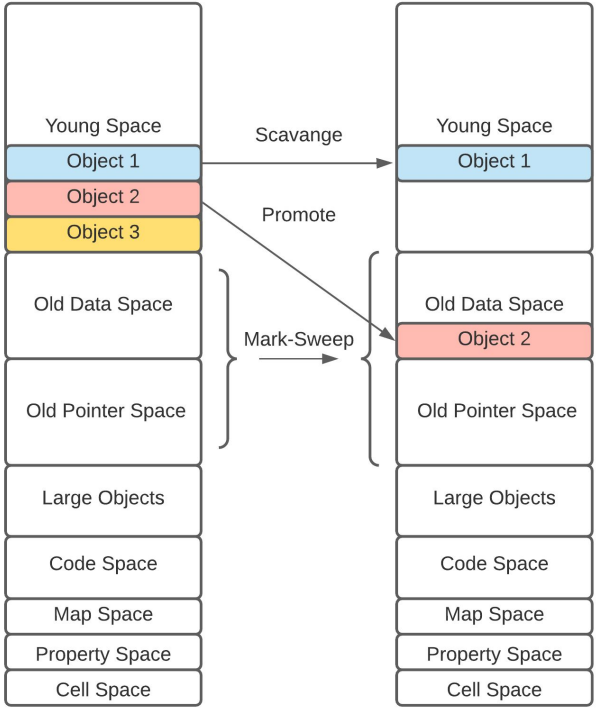
- Reverse Engineering V8
  - Garbage Collection in V8 [1]
    - Concurrent / Parallel (Not Stop-The-World)
      - “Weak Generational Hypothesis”
      - Indeterminate
    - Young Generation
    - Old Generation
    - Root Object Tracing (Transitive Closure)

[1] https://v8.dev/blog/trash-talk

V8 Garbage Collection



Typical Distribution for Lifetimes of Objects [1]



[1] https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/generations.html

Methodology

i. Created in Volatility2
   1. Find MetaMap using Magic Bytes
   2. Find Object Maps by finding every reference to MetaMap
   3. Find Objects by finding every reference to every Object Map!

Algorithm 1 Algorithm for V8MapScan

```
rule ← "FF 03 (20 | 40)"
y ← yara.compile(rule)
for process ∈ processList do
    if process ≠ V8_PROCESS then
        continue
    end if
    m ← y.match(process)
    if m ≠ V8_MetaMap then                ▷ Locate MetaMap
        continue
    end if
    yref ← yara.compile(m)                ▷ References to MetaMap
    mapRefs ← yref.match(process)
    for objMap ∈ mapRefs do
        V8Obj ← parseObj(objMap)          ▷ Extract objects
    end for
end for
```

It works!

```
v8@ubuntu: ~/volatility
File  Edit  View  Search  Terminal  Help
v8@ubuntu:~/volatility$ clear

v8@ubuntu:~/volatility$ python vol.py --plugins='/home/v8/volatility/contrib/plugins' -f '/home/v8/volatility/Windows 10 x64-a6dcc835.vmem'
Volatility Foundation Volatility Framework 2.6.1
Scanning node.exe pid: 1652
Opcode address: 0x7ff7e296d137L
Relative offset of first ptr: 0x2eb11a9
Address of first ptr: 0x7ff7e581e2e0L
Begin walking pointers
Current pointer: 0x7ff7e581e2e0L
Current pointer: 0x19adbf59750
Current pointer: 0x19adbf573b0
Current pointer: 0x19adbf57c70
Isolate address:
0x19adbf64ac0
META MAP
<volatility.plugins.objectscan.V8Map object at 0x7fac653d3710>
0x6b8d00988L
0x75
[['Property 0', 'unknown', '0x28afc0de259'], ['Property 1', 'unknown', '0x28afc0de271'], ['Property 2', 'unknown', '0x28afc0de289'], ['Prope
 ['Property 4', 'unknown', '0x28afc0de2b9'], ['Property 5', 'unknown', '0x28afc0de2d1'], ['Property 6', 'unknown', '0x28afc0de2e9'], ['Prope
 ['Property 8', 'unknown', '0x28afc0de319'], ['Property 9', 'unknown', '0x51b0c19cc1'], ['Property 10', 'unknown', '0x51b0c0aa01'], ['Proper
0471'], ['Property 12', 'unknown object', '0x35604800471'], ['Property 13', 'unknown', '0x51b0c19ca9'], ['Property 14', 'unknown object', '0
nown object', '0x35604800471'], ['Property 16', 'unknown object', '0x35604800471'], ['Property 17', 'unknown object', '0x35604800471'], ['Pr
04800471'], ['Property 19', 'unknown object', '0x35604800471'], ['Property 20', 'unknown object', '0x35604800471'], ['Property 21', 'unknown
ty 22', 'unknown object', '0x35604800471'], ['Property 23', 'unknown object', '0x35604800471'], ['Property 24', 'unknown object', '0x3560480
ect', '0x35604800471'], ['Property 26', 'unknown object', '0x35604800471'], ['Property 27', 'unknown object', '0x35604800471'], ['Property 2
'], ['Property 29', 'unknown object', '0x35604800471'], ['Property 30', 'unknown object', '0x35604800471'], ['Property 31', 'unknown object'
'unknown object', '0x35604800471'], ['Property 33', 'unknown object', '0x35604800471'], ['Property 34', 'unknown object', '0x35604800471'],
```

Sample Output of Plugin

Methodology

- Identify Different Objects
- Add Tool Functions
  - Object Properties
  - Object Data
  - Number of Objects
- Add support for different versions of Node.js
- Test support for other applications (Discord)

```
*************** INSTANCE TYPES FOR STRING ***************
INTERNALIZED_STRING_TYPE                                  : 0x00 0000000000000000     0
ONE_BYTE_INTERNALIZED_STRING_TYPE                         : 0x08 0000000000001000     8
EXTERNAL_INTERNALIZED_STRING_TYPE                         : 0x02 0000000000000010     2
EXTERNAL_ONE_BYTE_INTERNALIZED_STRING_TYPE                : 0x0A 0000000000001010    10
UNCACHED_EXTERNAL_INTERNALIZED_STRING_TYPE                : 0x12 0000000000010010    18
UNCACHED_EXTERNAL_ONE_BYTE_INTERNALIZED_STRING_TYPE       : 0x1A 0000000000011010    26
STRING_TYPE                                               : 0x20 0000000000100000    32
ONE_BYTE_STRING_TYPE                                      : 0x28 0000000000101000    40
CONS_STRING_TYPE                                          : 0x21 0000000000100001    33
CONS_ONE_BYTE_STRING_TYPE                                 : 0x29 0000000000101001    41
SLICED_STRING_TYPE                                        : 0x23 0000000000100011    35
SLICED_ONE_BYTE_STRING_TYPE                               : 0x2B 0000000000101011    43
EXTERNAL_STRING_TYPE                                      : 0x22 0000000000100010    34
EXTERNAL_ONE_BYTE_STRING_TYPE                             : 0x2A 0000000000101010    42
UNCACHED_EXTERNAL_STRING_TYPE                             : 0x32 0000000000110010    50
UNCACHED_EXTERNAL_ONE_BYTE_STRING_TYPE                    : 0x3A 0000000000111010    58
THIN_STRING_TYPE                                          : 0x25 0000000000100101    37
THIN_ONE_BYTE_STRING_TYPE                                 : 0x2D 0000000000101101    45
FIRST_NON_STRING_TYPE                                     : 0x40 0000000001000000    64
```

Instance Types and Type Encodings

Discussion and Evaluations

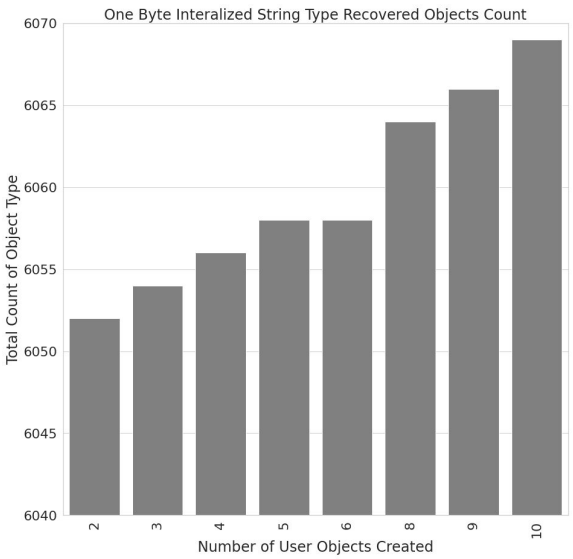
- Compare to Chrome's *heapdump* tool

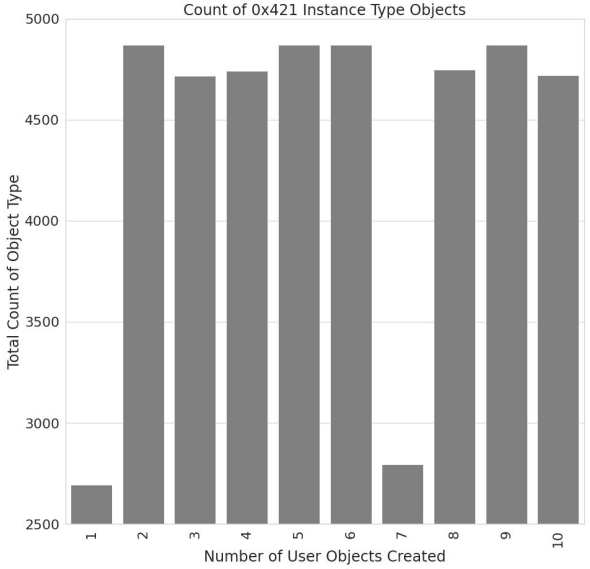
Instances Type	Heap Count	Plugin Count	Discovered Percent
All String Types	8180	8127	99.3%
Internalized String	5583	5527	98.9%
Array	1677	2699	>100%
ArrayBuffer	23	28	>100%
Global	2	1	50%
User Objects	10	10	100%
Symbols	111	154	>100%

Table comparing objects recovered

Discussion and Evaluations

- Evaluate the effect of increasing the number of objects
- Observe the effect of GC





Discussion and Evaluations

- Case study on *MyMonero* Crypto Wallet

```
NODE_UNIQUE_ID
_setupWorker
node-miner
miner
pool.minexmr.com
4BCwVXDPfEmCsfZU2LZyA2HRwdSvGuwxHaGHMxYc16ZBJunT6XhzwZDW7dEo87z6WmgnmAFyEjo5tfwWRxyzfvtqBTePMsD
password123
C:\Users\enoch\Desktop\node_modules\pmx\lib\utilsdebug
debugC:\Users\enoch\Desktop\node_modules\puppeteer\lib\node_modulesC:
\Users\enoch\Desktop\node_modules\puppeteer\node_modulesC:\Users\enoch\Desktop\node_modulesC:
\Users\enoch\node_modulesC:\Users\node_modulesC:\node_modulesC:\Users\enoch\.node_modulesC:
\Users\enoch\.node_librariesC:\Program Files\nodejs\lib\node
C:\Users\enoch\Desktop\node_modules\puppeteer\libws
wsC:\Users\enoch\Desktop\node_modules\puppeteer\lib\node_modulesC:
\Users\enoch\Desktop\node_modules\puppeteer\node_modulesC:\Users\enoch\Desktop\node_modulesC:
\Users\enoch\node_modulesC:\Users\node_modulesC:\node_modulesC:\Users\enoch\.node_modulesC:
\Users\enoch\.node_librariesC:\Program Files\nodejs\lib\node
C:\Users\enoch\Desktop\node_modules\ws./lib/WebSocket
./lib/WebSocketC:\Users\enoch\Desktop\node_modules\ws
C:\Users\enoch\Desktop\node_modules\ws\lib
C:\Users\enoch\Desktop\node_modules\ws\lib\WebSocket.js
C:\Users\enoch\Desktop\node_modules\send\node_modules
C:\Users\enoch\Desktop\node_modules\ipaddr.js
```

Address and Pool strings extracted from Wallet

Conclusion and Future Work

- Mapped out and verified the layout of V8 data structures
- It's possible to extract V8 objects through carving data structures
- The V8MapScan plugin works on all supported versions of Node.js
- Achieving support for Chrome will require a way to identify the core V8 isolate from multiple processes and sub-processes
- Achieving support for Discord will require converting the plugin to 32-bit
- Creating a Volatility3 version using Intermediate Symbol Format (ISF) files

Credits

Thanks for listening!

We hope you learned something! Feel free to reach out with questions or improvement ideas.

Additionally, we'd like to give a special thanks to...

Mathew Piscitelli for creating and testing the Discord version of the plugin.

Hailey Johnson for managing the project, testing the plugin, and keeping the teach on schedule

Contacts

Enoch Wang
enochsurge@gmail.com

Samurai Zurowski
szuro1@unh.newhaven.edu

Orion Duffy
oduff1@unh.newhaven.edu

Tyler Thomas
tthom10@unh.newhaven.edu

Ibrahim Baggili, Ph.D
ibrahim@gmail.com

Tool
https://github.com/unhcfreg/V8-Memory-Forensics-Plugins