Reverse Engineering Windows Defender's JavaScript Engine

Alexei Bulazel
@0xAlexei

REcon Brussels 2018

About Me

- Security researcher at River Loop Security
- RPI / RPISEC 2015 graduate
- Longtime REcon attendee, first time presenter
- Prior work on AV emulator analysis - “AVLeak”

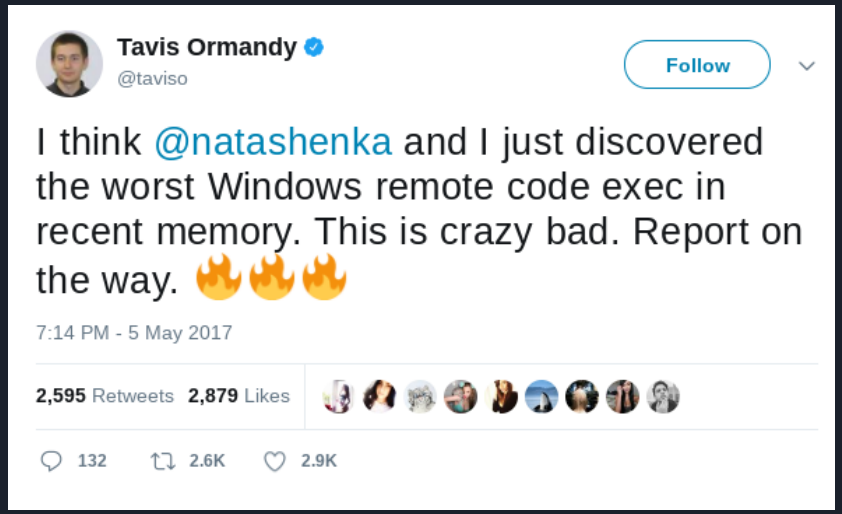
Twitter:
@0xAlexei

RPISEC

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Discussion
5. Conclusion

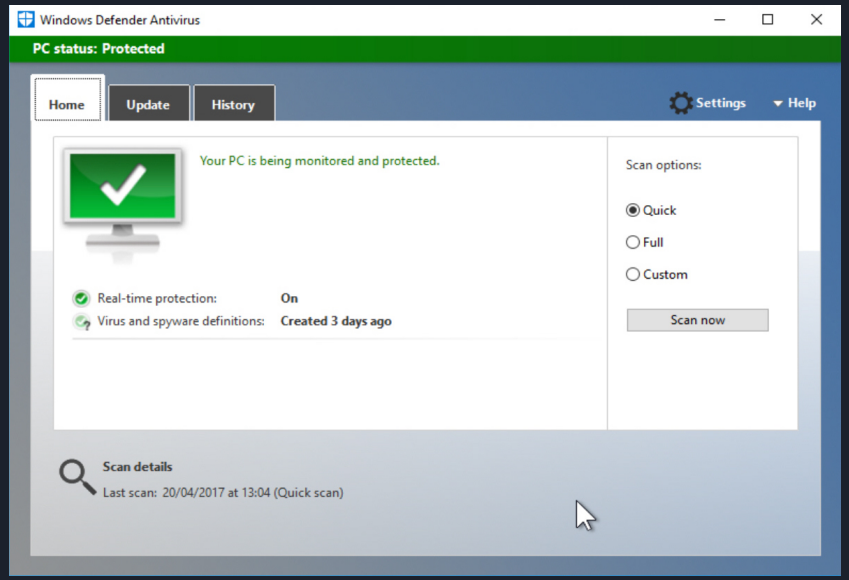
Motivation



- Tavis and Natalie at P0 dropped some awesome bugs
- Interest in JS engines, but I hadn't written JS since college
- I had reverse engineered AVs before, but never Defender
- This was a personal research project

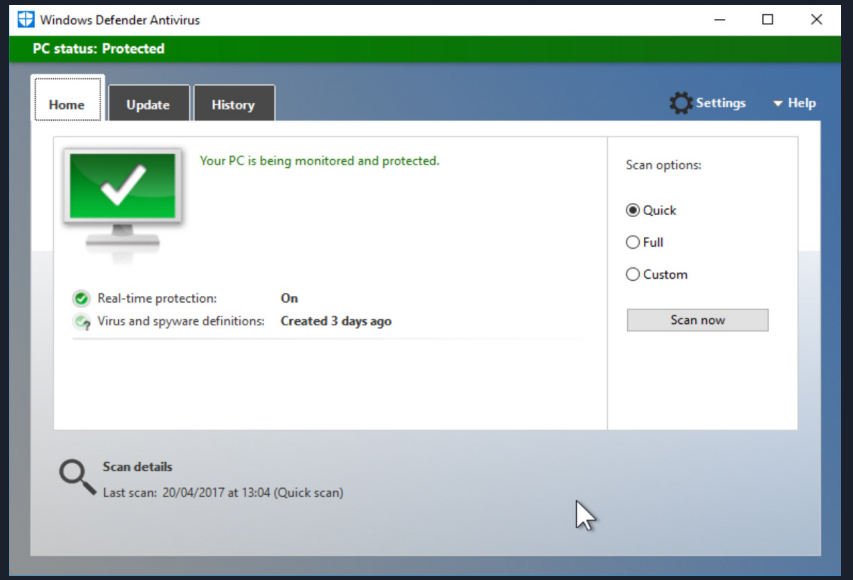
Windows Defender

- Microsoft's built-in antivirus software
  - Now the name seems to cover all mitigations / security controls built into Windows
- Runs as `NT AUTHORITY\SYSTEM`
  - Unsandboxed
- Built from many scanning subsystems stuck together



Windows Defender

- Microsoft's built-in antivirus software
  - Now the name seems to cover all mitigations / security controls built into Windows
- Runs as `NT AUTHORITY\SYSTEM`
  - Unsandboxed
- Built from many scanning subsystems stuck together



Focus: REing the JavaScript engine dedicated to scanning potentially malicious JavaScript, ~2% of Defender's code

JS Engines

Modern JS Engines

- Open source
- Highly complex
- Often integrated with browsers

JS Engines

Modern JS Engines

- Open source
- Highly complex
- Often integrated with browsers

Defender's JS Engine

- Binary
- Complex but tractable for RE from binary
- Standalone with some minor browser emulation

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Discussion
5. Conclusion

Binaries

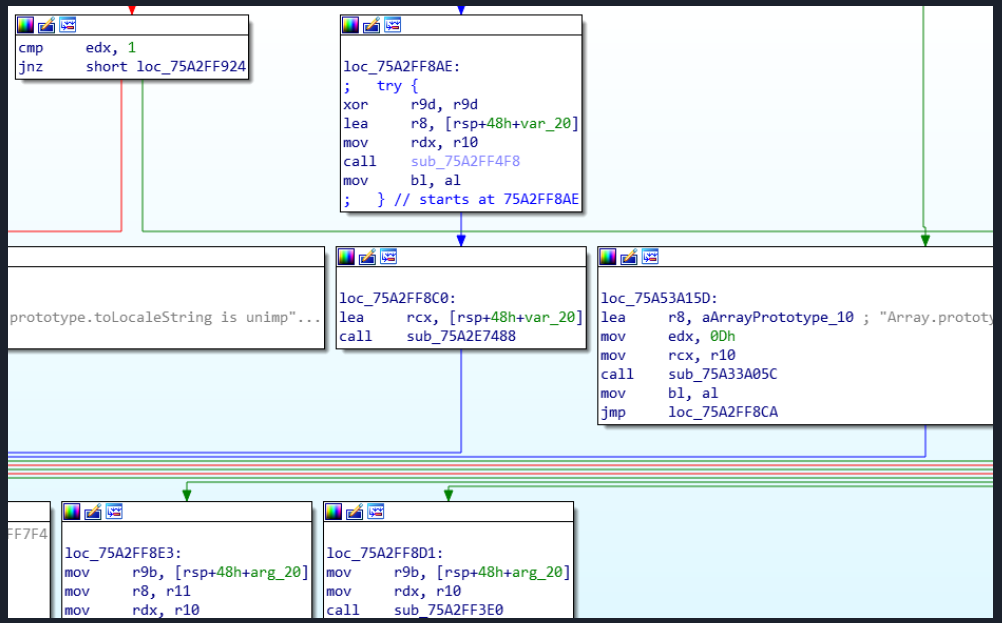
`mpam-fe.exe` released monthly:

- `mpengine.dll`
  “Microsoft Malware Protection Engine”
- `MPSigStub.exe`
  “Microsoft Malware Protection Signature Update Stub”

- `mpasbase.vdm`
- `mpasdlta.vdm`

- `mpavbase.vdm`
- `mpavdlta.vdm`

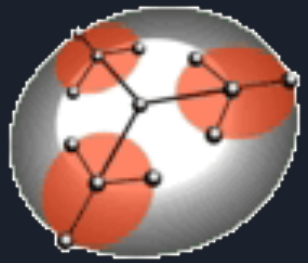
- 5/23 (P0 bugs fixed)
- 6/20
- 7/19
- 8/23
- 9/27
- 11/1
- 12/6 (UK NCSC bugs fixed)
- 1/18 (latest)



32 & 64-bit builds

Tools

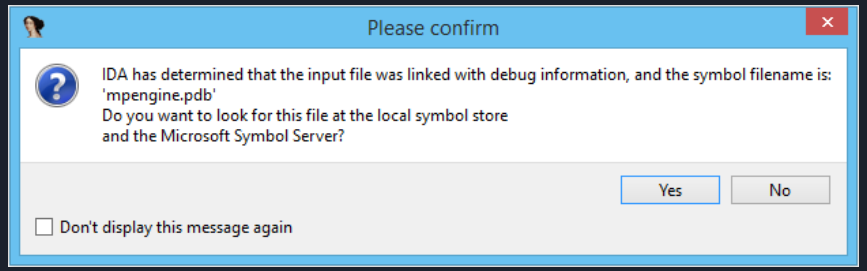
- Static reversing in IDA with PDBs
- BinDiff / Diaphora diffing patches
- Dynamic analysis with JS shell harness and WinDBG

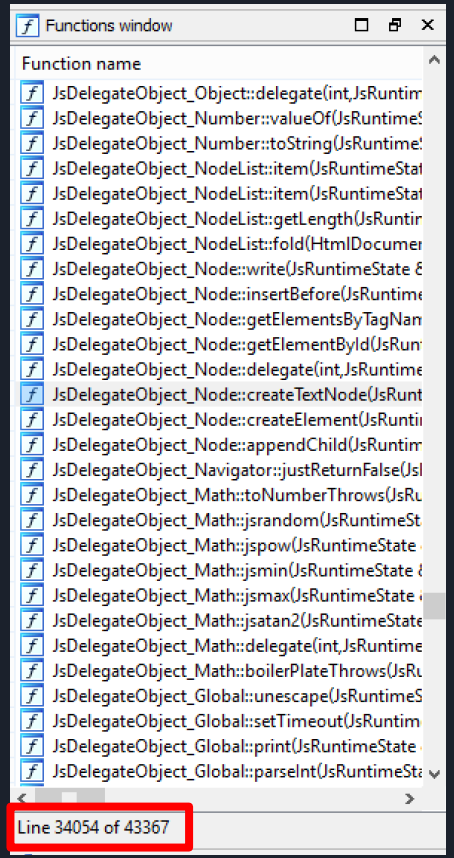


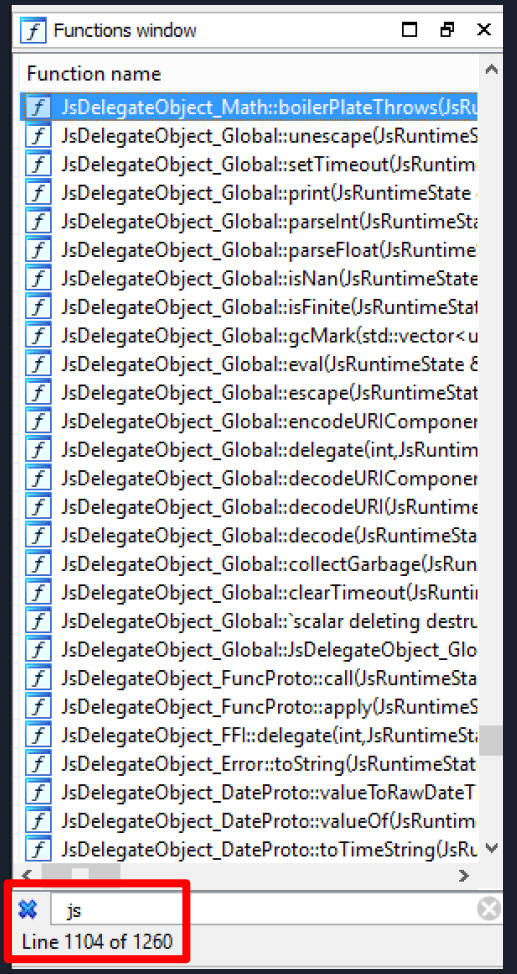
```
$ ./JsShell.exe
CONSTRUCTOR_CALL:       6EA109AE
DESTRUCTOR:             6EA21830
CONSTRUCTOR:            6EA21ACA
EVAL:                   6EA10875

mpscript> (function (){for(var i = 0; i < 3; i++){print(i + ": Hello from insid
e MpEngine.dll")}})()
print(): 0: Hello from inside MpEngine.dll
print(): 1: Hello from inside MpEngine.dll
print(): 2: Hello from inside MpEngine.dll
print(): undefined
Log():            <NA>:   0: execution took 239 ticks
Log():            <NA>:   0: final memory used 9KB
Log():            <NA>:   0: total of 0 GCs performed

Ended. Result code: 0
mpscript> _
```

Static Analysis - `mpengine.dll`

- 45,000+ functions total
- JS Engine is ~1,200 functions
- Microsoft publishes PDBs!







Shell

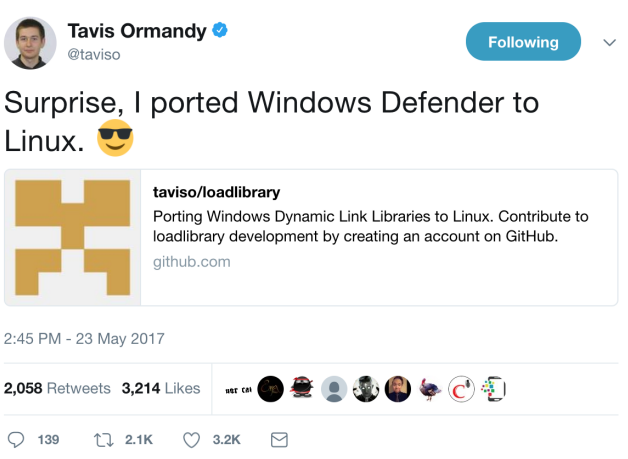
```
$ ./JsShell.exe
CONSTRUCTOR_CALL:       6EA109AE
DESTRUCTOR:             6EA21830
CONSTRUCTOR:            6EA21ACA
EVAL:                   6EA10875

mpscript> (function (){for(var i = 0; i < 3; i++){print(i + ": Hello from insid
e MpEngine.dll")}})()
print(): 0: Hello from inside MpEngine.dll
print(): 1: Hello from inside MpEngine.dll
print(): 2: Hello from inside MpEngine.dll
print(): undefined
Log():              <NA>:   0: execution took 239 ticks
Log():              <NA>:   0: final memory used 9KB
Log():              <NA>:   0: total of 0 GCs performed

Ended. Result code: 0
mpscript> _
```

Custom Loaders

Challenges:

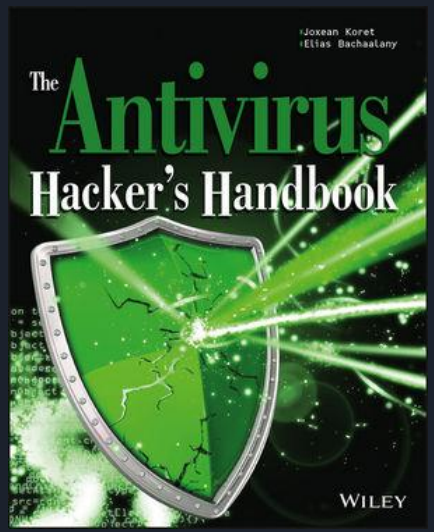
- Introspection
- Protected process
- System stability
- Scanning on demand
- Code reachability may be configuration / heuristics dependent

```
int __thiscall nscript::shouldRunJSEmulation(nscript *this)
{
  int result; // eax@3

  if ( this->m_bJSDisableEmulation )
  {
    result = 0;
  }
  else if ( this->m_bJSForceEmulation )
  {
    result = 1;
  }
  else
  {
    result = nscript::checkJsFeatures(this, 0) >= g_JSEmu_heurPointsThreshold;
  }
  return result;
}
```



Example: MPEngine Lockdown

- “Protected Processes” - Windows programs that you cannot debug with a usermode debugger, even if you have all privileges
- Attackers can load a signed vulnerable driver, run an exploit, get execution & deprotect the process - so … why?



“Repeated vs. single-round games in security”
Halvar Flake, BSides Zurich Keynote

Custom Loaders

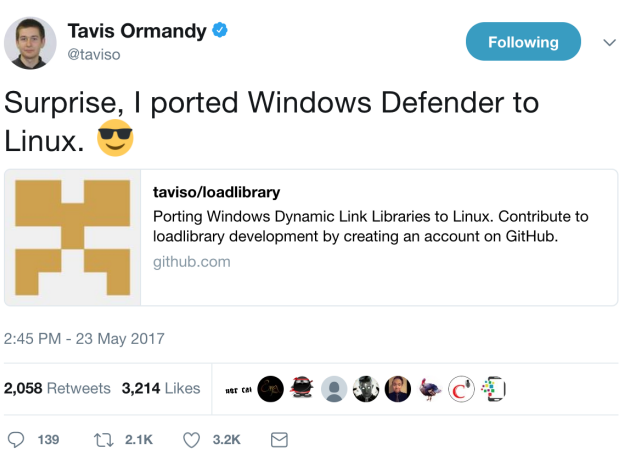
Challenges:

- Introspection
- Protected process
- System stability
- Scanning on demand
- Code reachability may be configuration / heuristics dependent

Solution:

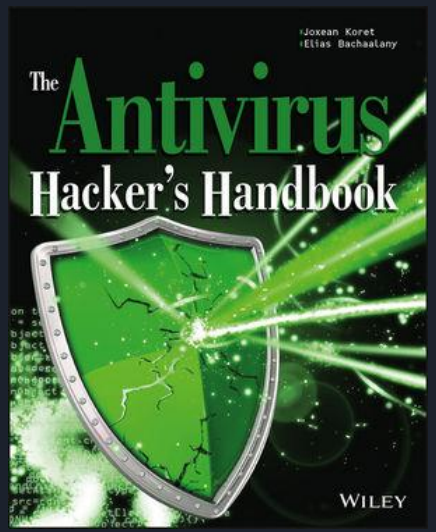
- Custom loader
- Call directly into functions that initiate scanning

```
int __thiscall nscript::shouldRunJSEmulation(nscript *this)
{
  int result; // eax@3

  if ( this->m_bJSDisableEmulation )
  {
    result = 0;
  }
  else if ( this->m_bJSForceEmulation )
  {
    result = 1;
  }
  else
  {
    result = nscript::checkJsFeatures(this, 0) >= g_JSEmu_heurPointsThreshold;
  }
  return result;
}
```



Example: MPEngine Lockdown

- "Protected Processes" - Windows programs that you cannot debug with a usermode debugger, even if you have all privileges
- Attackers can load a signed vulnerable driver, run an exploit, get execution & deprotect the process - so … why?



"Repeated vs. single-round games in security"
Halvar Flake, BSides Zurich Keynote

Loader and Shell

```
JsRuntimeState::triggerEvent(jsState, 0, "print", strCstr, strCstr_4, v8, v8)
```

- Collab with Rolf Rolles, based on a shell written in D released by `@TheWack0lian` on Twitter

- Use `LoadLibrary` on Windows
  - WinDbg works natively
- Patch constructor for `JsRuntimeState::JsRuntimeState()`
  - Provide a custom VTable implementing analysis callbacks
  - Print to `stdout` on "`print`" events
  - Log other events
- Directly call to start scan

```
JavaScriptInterpreter::eval(
    const char *input,
    unsigned int inputSize,
    JavaScriptInterpreter::Params *params)
```

```
mov     esi, [ebp+toStringTree.baseclass_0.vfptr]
push    ecx             ; monitor
lea     ecx, [ebp+jsState] ; this
push    dword ptr [esi+20h] ; domWrapper
push    dword ptr [esi+14h] ; regexpLimit
push    dword ptr [esi+18h] ; gcLimit
push    dword ptr [esi+10h] ; memLimit
push    dword ptr [esi+0Ch] ; exeLimit
call    ??0JsRuntimeState@@QAE@IIIIPAUHtmlDocumentProvider@@PAUJsEvaluationMonitor@@@Z
mov     byte ptr [ebp+var_4], 3
mov     ecx, [esi]
mov     al, cl
shr     al, 1
and     cl, 1
and     al, 1
mov     dl, cl          ; addBrowserRt
push    eax             ; addDomRt
lea     ecx, [ebp+jsState] ; jsState
call    ?declareGlobalProperties@@YA_NAAUJsRuntimeState@@_N1@Z ; declareGlobalPropertie
pop     ecx
test    al, al
jz      loc_5A5838CC
```

Loader and Shell

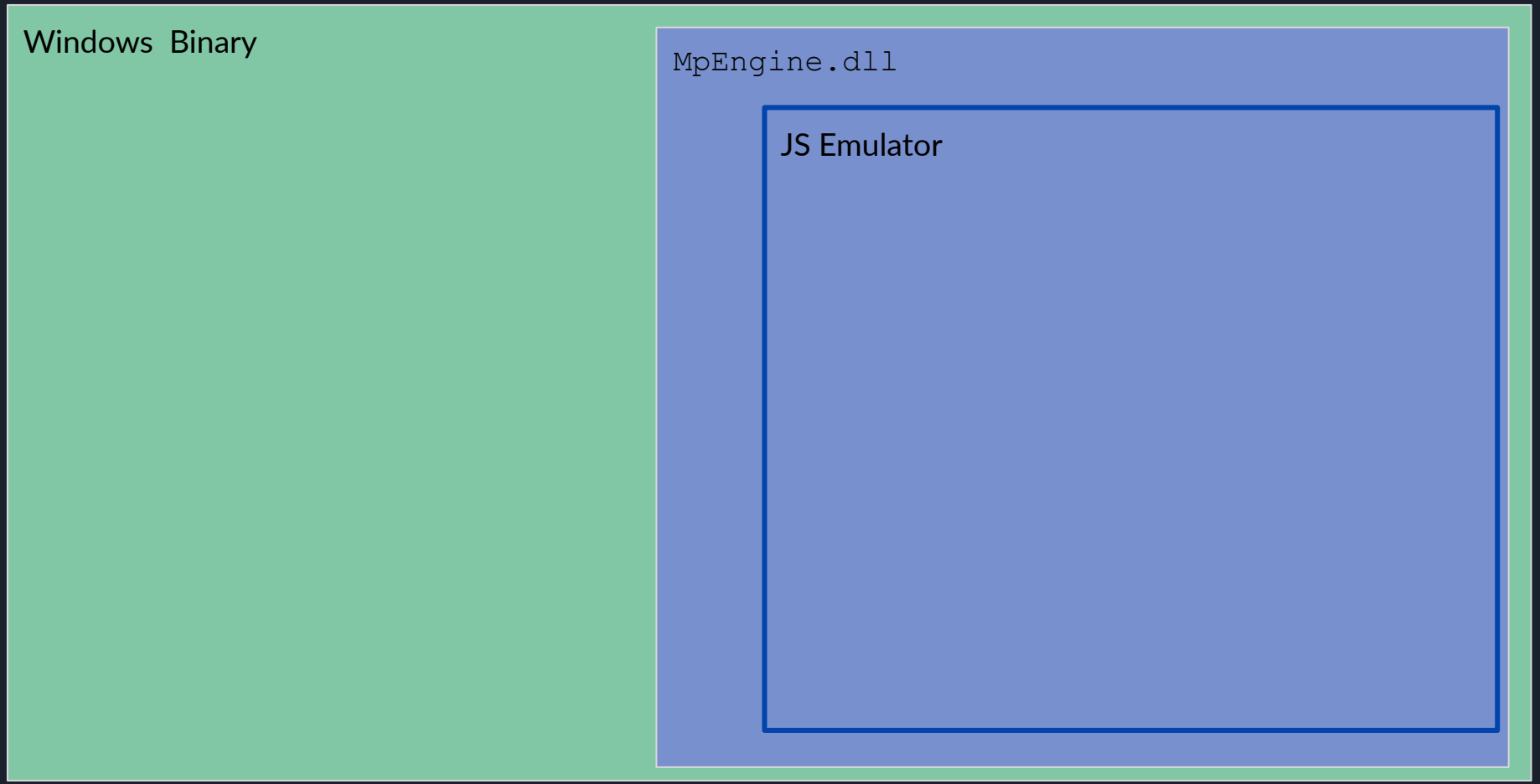
Windows Binary

Loader and Shell

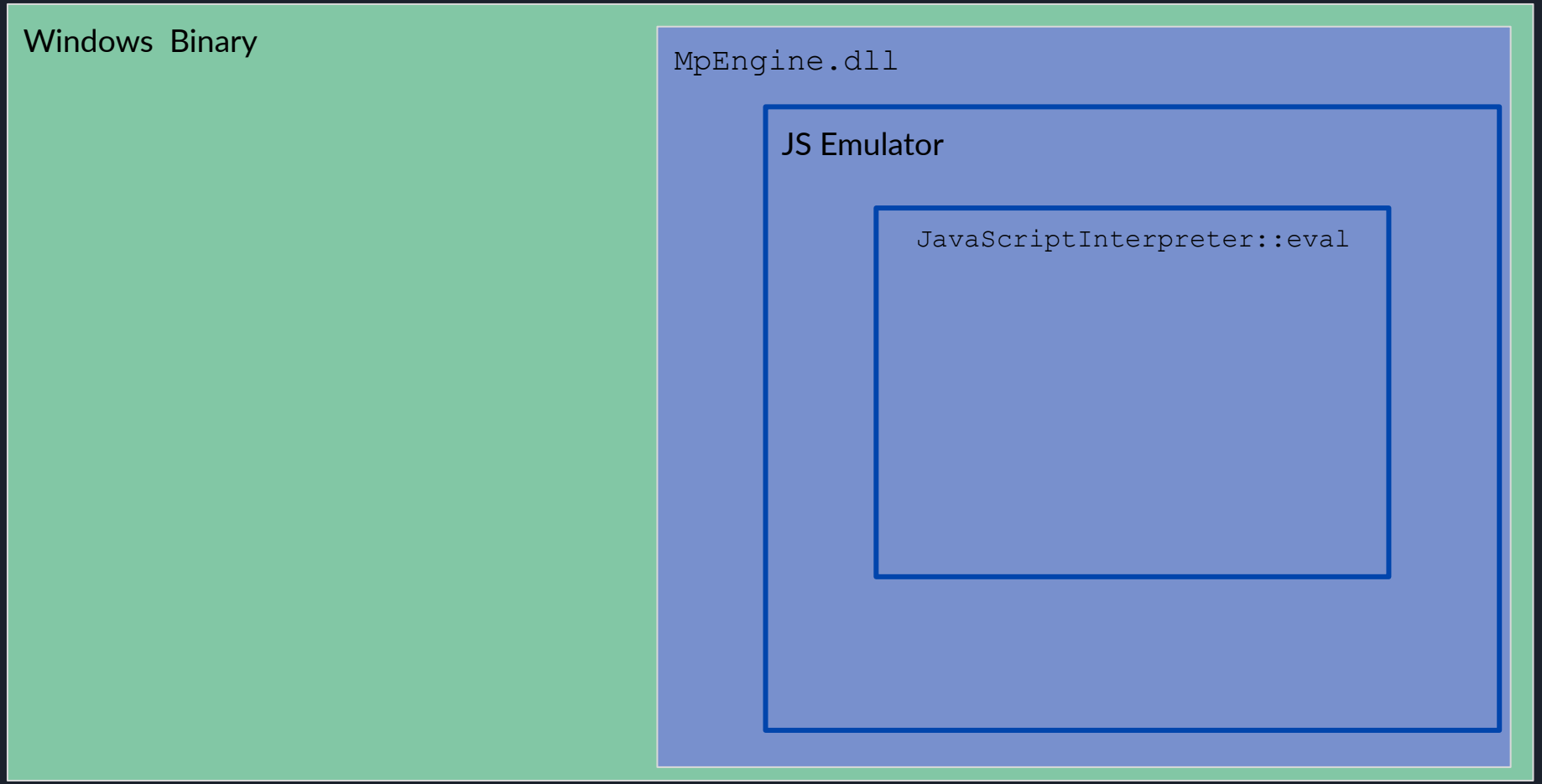
Windows Binary

MpEngine.dll

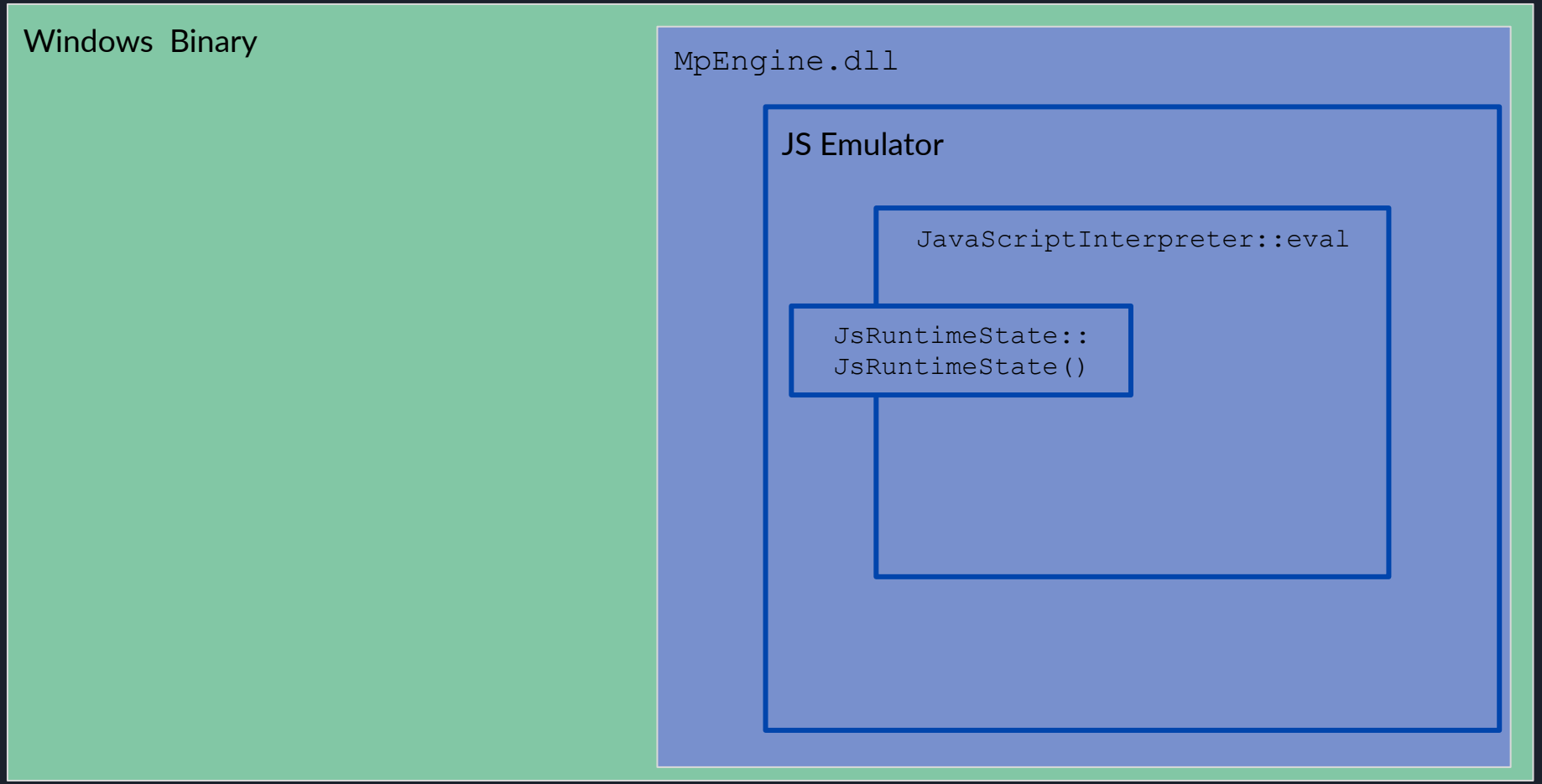
Loader and Shell



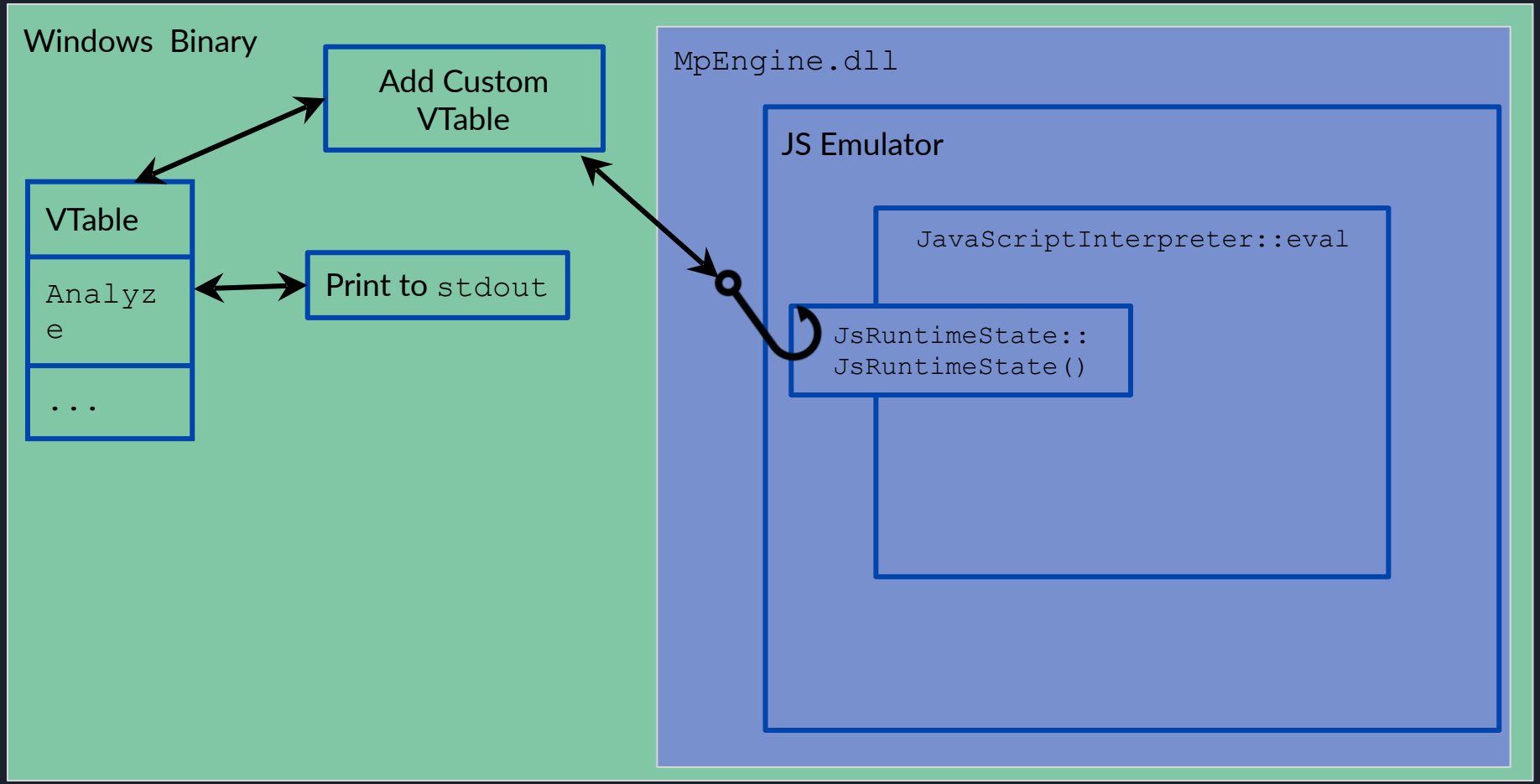
Loader and Shell



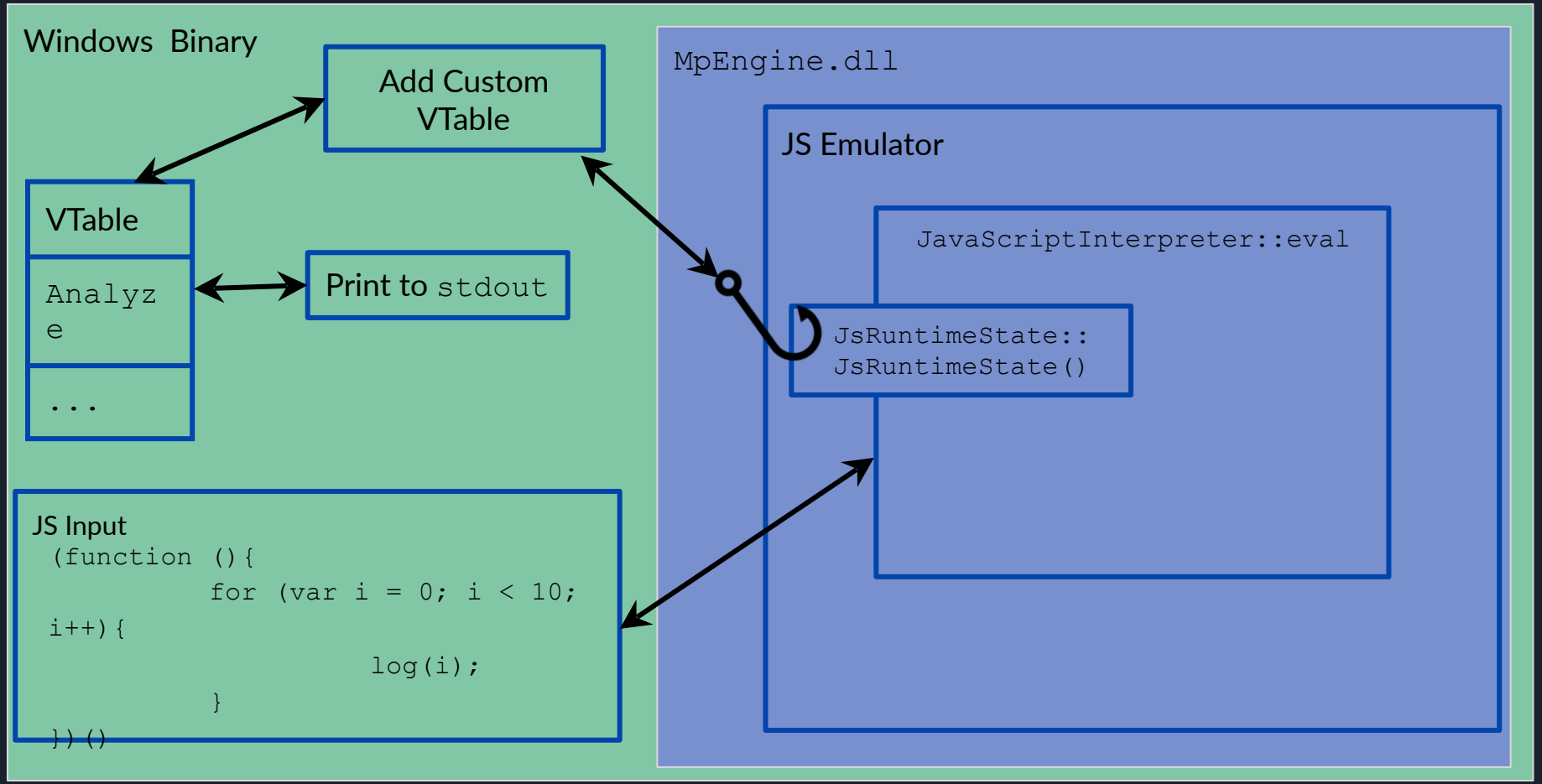
Loader and Shell



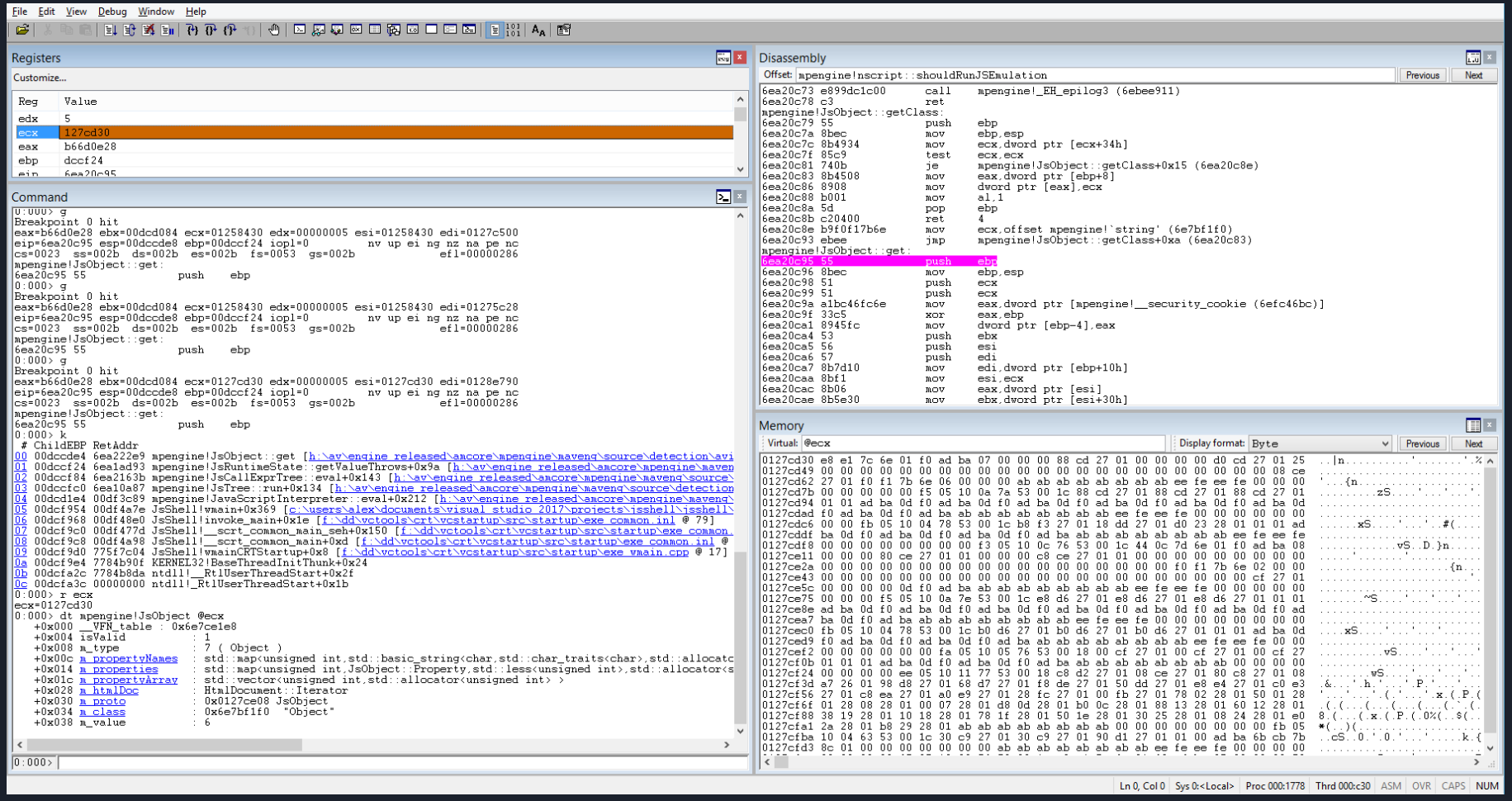
Loader and Shell



Loader and Shell



WinDbg


File Edit View Debug Window Help
Registers
Customize...
Reg Value
edx 5
ecx 127cd30
eax b66d0e28
ebp dccf24
Command
0:000> g
Breakpoint 0 hit
eax=b66d0e28 ebx=00dcd084 ecx=01258430 edx=00000005 esi=01258430 edi=0127c500
eip=6ea20c95 esp=00dccde8 ebp=00dccf24 iopl=0 nv up ei ng nz na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000286
mpengine!JsObject::get:
6ea20c95 55 push ebp
0:000> g
Breakpoint 0 hit
eax=b66d0e28 ebx=00dcd084 ecx=01258430 edx=00000005 esi=01258430 edi=01275c28
eip=6ea20c95 esp=00dccde8 ebp=00dccf24 iopl=0 nv up ei ng nz na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000286
mpengine!JsObject::get:
6ea20c95 55 push ebp
0:000> g
Breakpoint 0 hit
eax=b66d0e28 ebx=00dcd084 ecx=0127cd30 edx=00000005 esi=0127cd30 edi=0128e790
eip=6ea20c95 esp=00dccde8 ebp=00dccf24 iopl=0 nv up ei ng nz na pe nc
cs=0023 ss=002b ds=002b es=002b fs=0053 gs=002b efl=00000286
mpengine!JsObject::get:
6ea20c95 55 push ebp
0:000> k
ChildEBP RetAddr
00 00dccde4 6ea222e9 mpengine!JsObject::get [h:\av\engine released\amcore\mpengine\maveng\source\detection\avi
01 00dccf24 6ea1ad93 mpengine!JsRuntimeState::getValueThrows+0x9a [h:\av\engine released\amcore\mpengine\maven
02 00dccf84 6ea2163b mpengine!JsCallExprTree::eval+0x143 [h:\av\engine released\amcore\mpengine\maveng\source\
03 00dccfc0 6ea10a87 mpengine!JsTree::run+0x134 [h:\av\engine released\amcore\mpengine\maveng\source\detection
04 00dcd1e4 00df3c89 mpengine!JavaScriptInterpreter::eval+0x212 [h:\av\engine released\amcore\mpengine\maveng\
05 00dcf954 00df4a7e JsShell!wmain+0x369 [c:\users\alex\documents\visual studio 2017\projects\jsshell\jsshell\
06 00dcf968 00df48e0 JsShell!invoke_main+0x1e [f:\dd\vctools\crt\vcstartup\src\startup\exe_common.inl @ 79]
07 00dcf9c0 00df477d JsShell!__scrt_common_main_seh+0x150 [f:\dd\vctools\crt\vcstartup\src\startup\exe_common.
08 00dcf9c8 00df4a98 JsShell!__scrt_common_main+0xd [f:\dd\vctools\crt\vcstartup\src\startup\exe_common.inl @
09 00dcf9d0 775f7c04 JsShell!wmainCRTStartup+0x8 [f:\dd\vctools\crt\vcstartup\src\startup\exe_wmain.cpp @ 17]
0a 00dcf9e4 7784b90f KERNEL32!BaseThreadInitThunk+0x24
0b 00dcfa2c 7784b8da ntdll!__RtlUserThreadStart+0x2f
0c 00dcfa3c 00000000 ntdll!_RtlUserThreadStart+0x1b
0:000> r ecx
ecx=0127cd30
0:000> dt mpengine!JsObject @ecx
+0x000 __VFN_table : 0x6e7ce1e8
+0x004 isValid : 1
+0x008 m_type : 7 (Object)
+0x00c m_propertyNames : std::map<unsigned int,std::basic_string<char,std::char_traits<char>,std::allocatc
+0x014 m_properties : std::map<unsigned int,JsObject::Property,std::less<unsigned int>,std::allocator<s
+0x01c m_propertyArray : std::vector<unsigned int,std::allocator<unsigned int> >
+0x028 m_htmlDoc : HtmlDocument::Iterator
+0x030 m_proto : 0x0127ce08 JsObject
+0x034 m_class : 0x6e7bf1f0 "Object"
+0x038 m_value : 6
0:000>
Disassembly
Offset: mpengine!nscript::shouldRunJSEmulation
Previous Next
6ea20c73 e899dc1c00 call mpengine!_EH_epilog3 (6ebee911)
6ea20c78 c3 ret
mpengine!JsObject::getClass:
6ea20c79 55 push ebp
6ea20c7a 8bec mov ebp,esp
6ea20c7c 8b4934 mov ecx,dword ptr [ecx+34h]
6ea20c7f 85c9 test ecx,ecx
6ea20c81 740b je mpengine!JsObject::getClass+0x15 (6ea20c8e)
6ea20c83 8b4508 mov eax,dword ptr [ebp+8]
6ea20c86 8908 mov dword ptr [eax],ecx
6ea20c88 b001 mov al,1
6ea20c8a 5d pop ebp
6ea20c8b c20400 ret 4
6ea20c8e b9f0f17b6e mov ecx,offset mpengine!`string' (6e7bf1f0)
6ea20c93 ebee jmp mpengine!JsObject::getClass+0xa (6ea20c83)
mpengine!JsObject::get:
6ea20c95 55 push ebp
6ea20c96 8bec mov ebp,esp
6ea20c98 51 push ecx
6ea20c99 51 push ecx
6ea20c9a a1bc46fc6e mov eax,dword ptr [mpengine!__security_cookie (6efc46bc)]
6ea20c9f 33c5 xor eax,ebp
6ea20ca1 8945fc mov dword ptr [ebp-4],eax
6ea20ca4 53 push ebx
6ea20ca5 56 push esi
6ea20ca6 57 push edi
6ea20ca7 8b7d10 mov edi,dword ptr [ebp+10h]
6ea20caa 8bf1 mov esi,ecx
6ea20cac 8b06 mov eax,dword ptr [esi]
6ea20cae 8b5e30 mov ebx,dword ptr [esi+30h]
Memory
Virtual: @ecx
Display format: Byte
Previous Next
0127cd30 e8 e1 7c 6e 01 f0 ad ba 07 00 00 00 88 cd 27 01 00 00 00 00 d0 cd 27 01 25 ..|n..........'.....'.%
0127cd49 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 08 ce ....
0127cd62 27 01 f0 f1 7b 6e 06 00 00 00 ab ab ab ab ab ab ab ab ee fe ee fe 00 00 00 '...{n..........
0127cd7b 00 00 00 00 00 f5 05 10 0a 7a 53 00 1c 88 cd 27 01 88 cd 27 01 ....zS....'...'.
0127cd94 01 01 ad ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ba 0d ....
0127cdad f0 ad ba 0d f0 ad ba ab ab ab ab ab ab ab ee fe ee fe 00 00 00 00 00 00 00 ....
0127cdc6 00 00 fb 05 10 04 78 53 00 1c b8 f3 27 01 18 dd 27 01 d0 23 28 01 01 01 ad ......xS....'...'..#(....
0127cddf ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ba ab ab ab ab ab ab ee fe ee fe ....
0127cdf8 00 00 00 00 00 00 00 00 f3 05 10 0c 76 53 00 1c 44 0c 7d 6e 01 f0 ad ba 08 ....vS..D.}n....
0127ce11 00 00 00 80 ce 27 01 01 00 00 00 c8 ce 27 01 01 00 00 00 00 00 00 00 00 .....'.......'..........
0127ce2a 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 f0 f1 7b 6e 02 00 00 00 ....{n...
0127ce43 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 cf 27 01 ....'.
0127ce5c 00 00 00 00 0d f0 ad ba ab ab ab ab ab ab ab ee fe ee fe 00 00 00 00 00 ....
0127ce75 00 00 00 f5 05 10 0a 7e 53 00 1c e8 d6 27 01 e8 d6 27 01 e8 d6 27 01 01 01 ....~S....'...'...'...
0127ce8e ad ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ba 0d f0 ad ....
0127cea7 ba 0d f0 ad ba ab ab ab ab ab ab ab ee fe ee fe 00 00 00 00 00 00 00 00 ....
0127cec0 fb 05 10 04 78 53 00 1c b0 d6 27 01 b0 d6 27 01 b0 d6 27 01 01 01 ad ba 0d ....xS....'...'...'.....
0127ced9 f0 ad ba 0d f0 ad ba 0d f0 ad ba ab ab ab ab ab ab ab ee fe ee fe 00 00 00 ....
0127cef2 00 00 00 00 00 00 fa 05 10 05 76 53 00 18 00 cf 27 01 00 cf 27 ....vS....'...'.
0127cf0b 01 01 01 ad ba 0d f0 ad ba 0d f0 ad ba ab ab ab ab ab ab ab 00 00 00 00 ....
0127cf24 00 00 00 00 ee 05 10 11 77 53 00 18 c8 d2 27 01 08 ce 27 01 80 c8 27 01 08 ....wS....'...'...'...
0127cf3d a7 26 01 98 d0 27 01 68 d7 27 01 f8 de 27 01 e8 e4 27 01 c0 e3 .&...'.h.'...'.....'....
0127cf56 27 01 c8 ea 27 01 a0 e9 27 01 28 fc 27 01 00 fb 27 01 78 02 28 01 50 01 28 '...'...'.(.'....'.x.(.P.(
0127cf6f 01 28 08 28 01 00 07 28 01 d8 0d 28 01 b0 0c 28 01 88 13 28 01 60 12 28 01 .(.(...(...(...(...(.`.(.
0127cf88 38 19 28 01 10 18 28 01 78 1f 28 01 50 1e 28 01 30 25 28 01 08 24 28 01 e0 8.(...(.x.(.P.(.0%(..$(..
0127cfa1 2a 28 01 b8 29 28 01 ab ab ab ab ab ab ab 00 00 00 00 00 00 00 fb 05 *(..)(..........
0127cfba 10 04 63 53 00 1c 30 c9 27 01 30 c9 27 01 90 d1 27 01 01 00 ad ba 6b cb 7b ..cS..0.'.0.'.....'.....k.{
0127cfd3 8c 01 00 00 00 00 00 00 00 ab ab ab ab ab ab ab ab ee fe ee fe 00 00 00 00
Ln 0, Col 0 Sys 0:<Local> Proc 000:1778 Thrd 000:c30 ASM OVR CAPS NUM

Tavis Ormandy's `loadlibrary`

- PE loader for Linux
  - Shim out implementations for Windows API imported functions
  - Go through full initialization process
- `mpscript` tool exposes the JS engine
- Hook the `__rsignal` function that tells MpEngine to scan something
  - Scan a buffer, it gets detected as JS and subsequently analyzed
- Hook `_strtod` to get output
  - `function log(msg){ parseFloat('__log: ' + msg);}`

https://github.com/taviso/loadlibrary

Taviso's `loadlibrary` https://github.com/taviso/loadlibrary

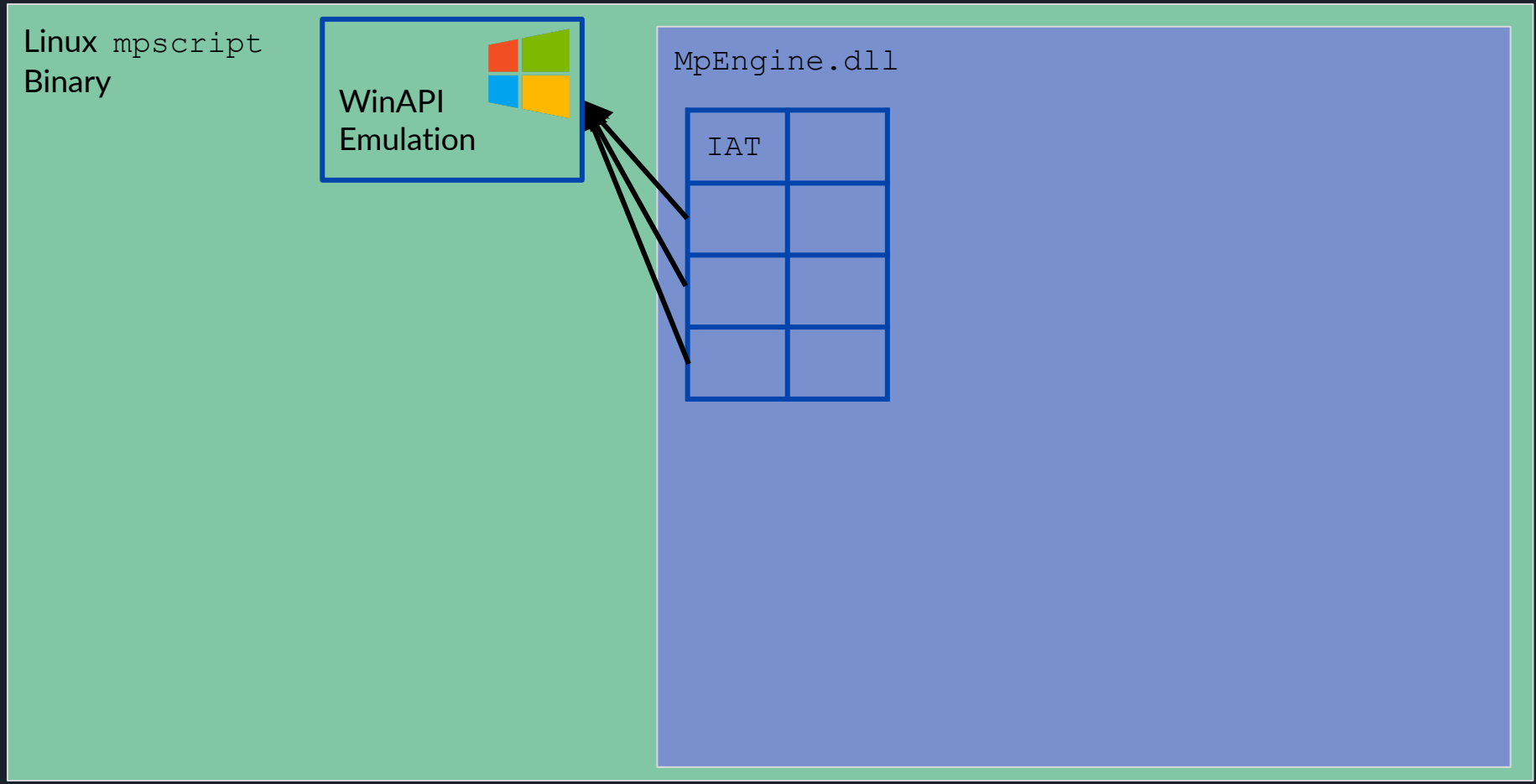
Linux `mpscript`
Binary

Taviso's `loadlibrary` https://github.com/taviso/loadlibrary

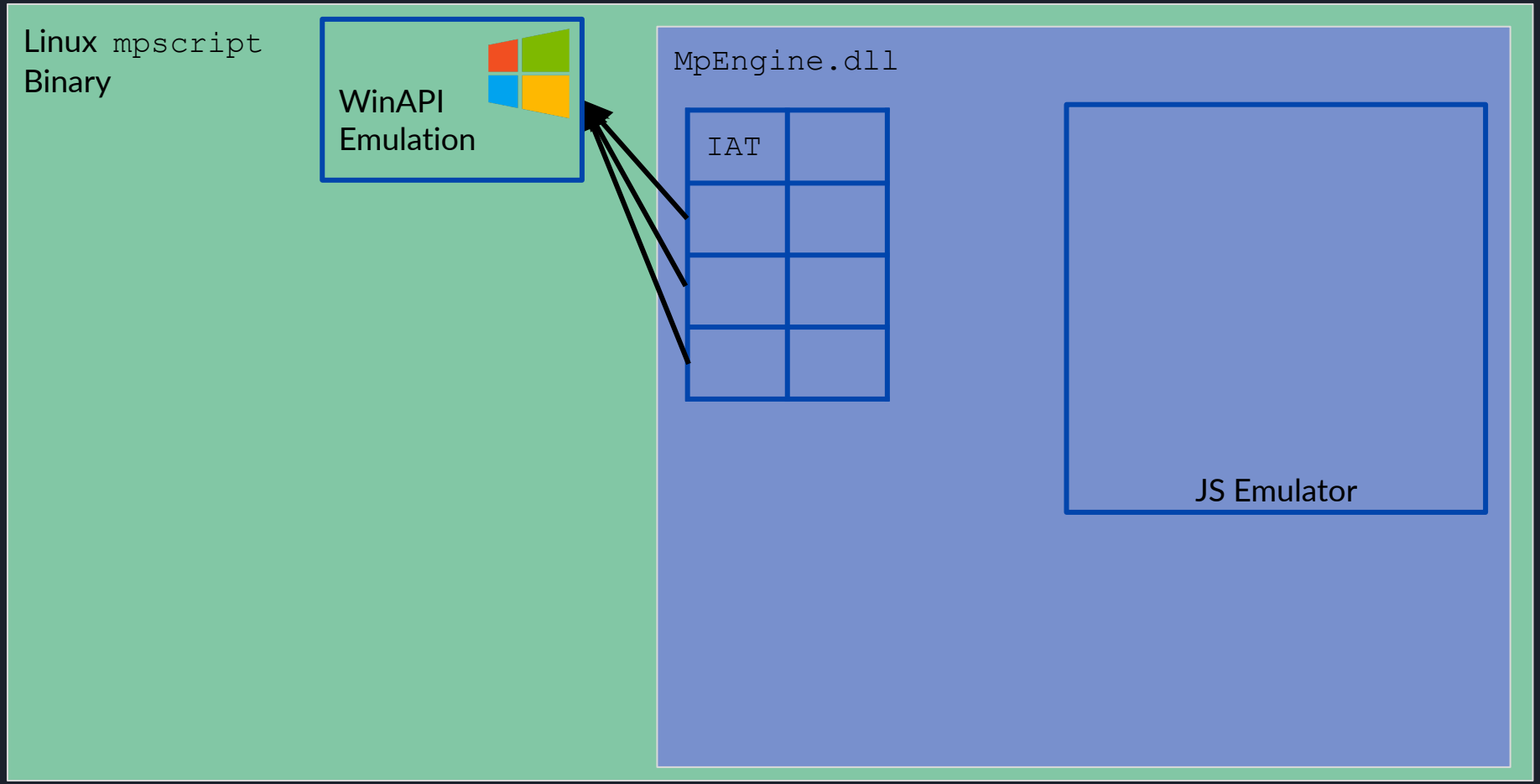
Linux `mpscript`
Binary

`MpEngine.dll`

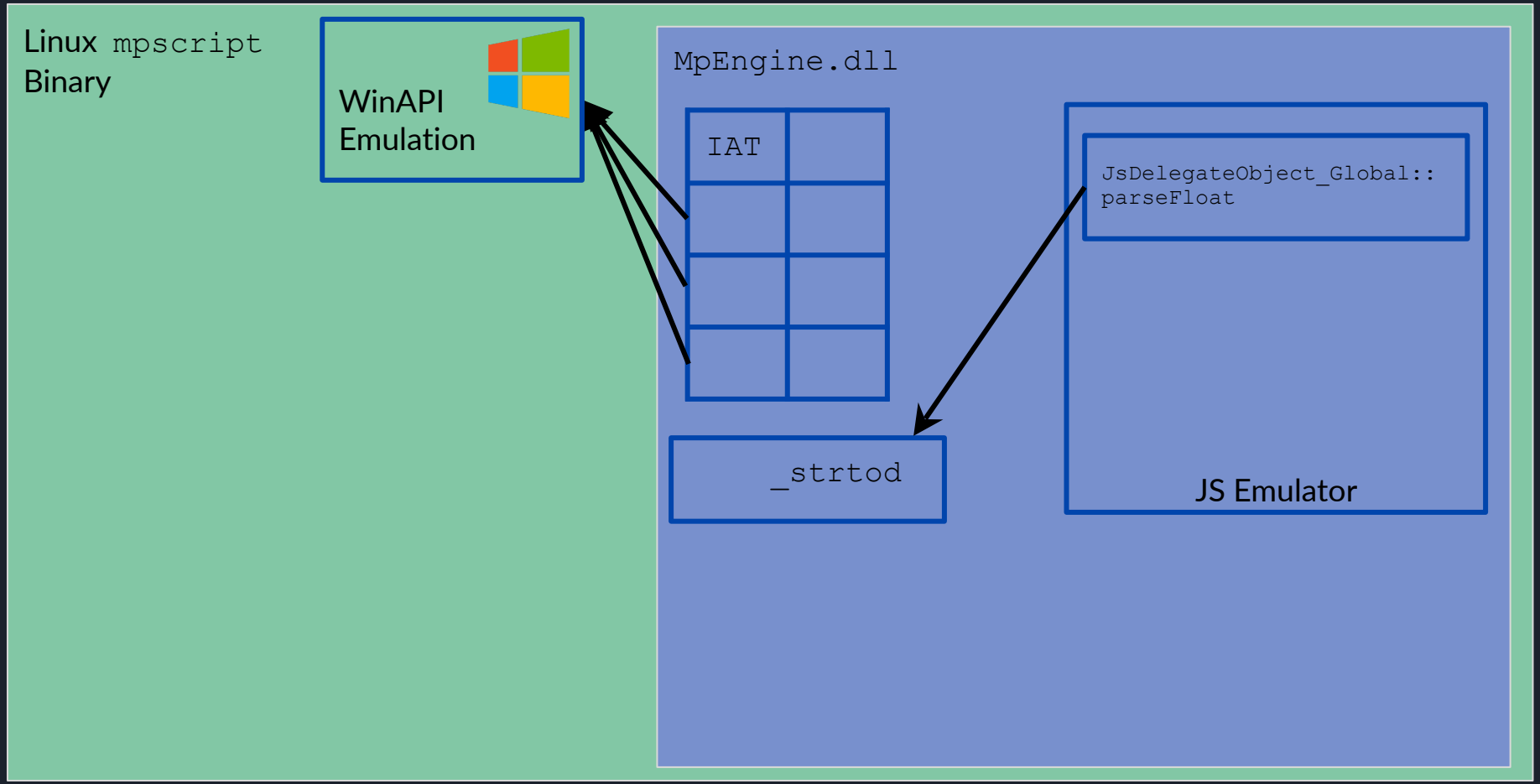
Taviso's `loadlibrary` https://github.com/taviso/loadlibrary



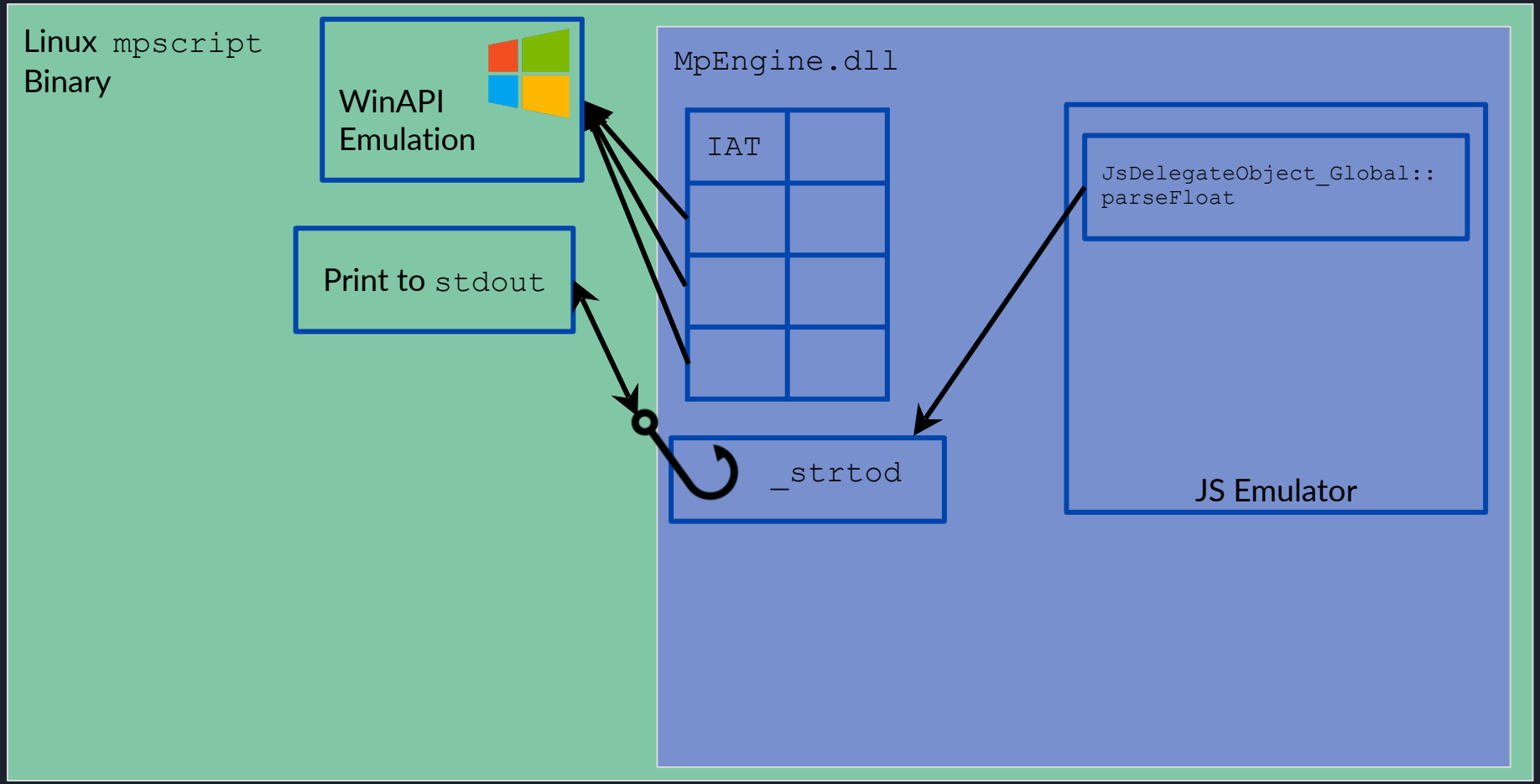
Taviso's `loadlibrary` https://github.com/taviso/loadlibrary



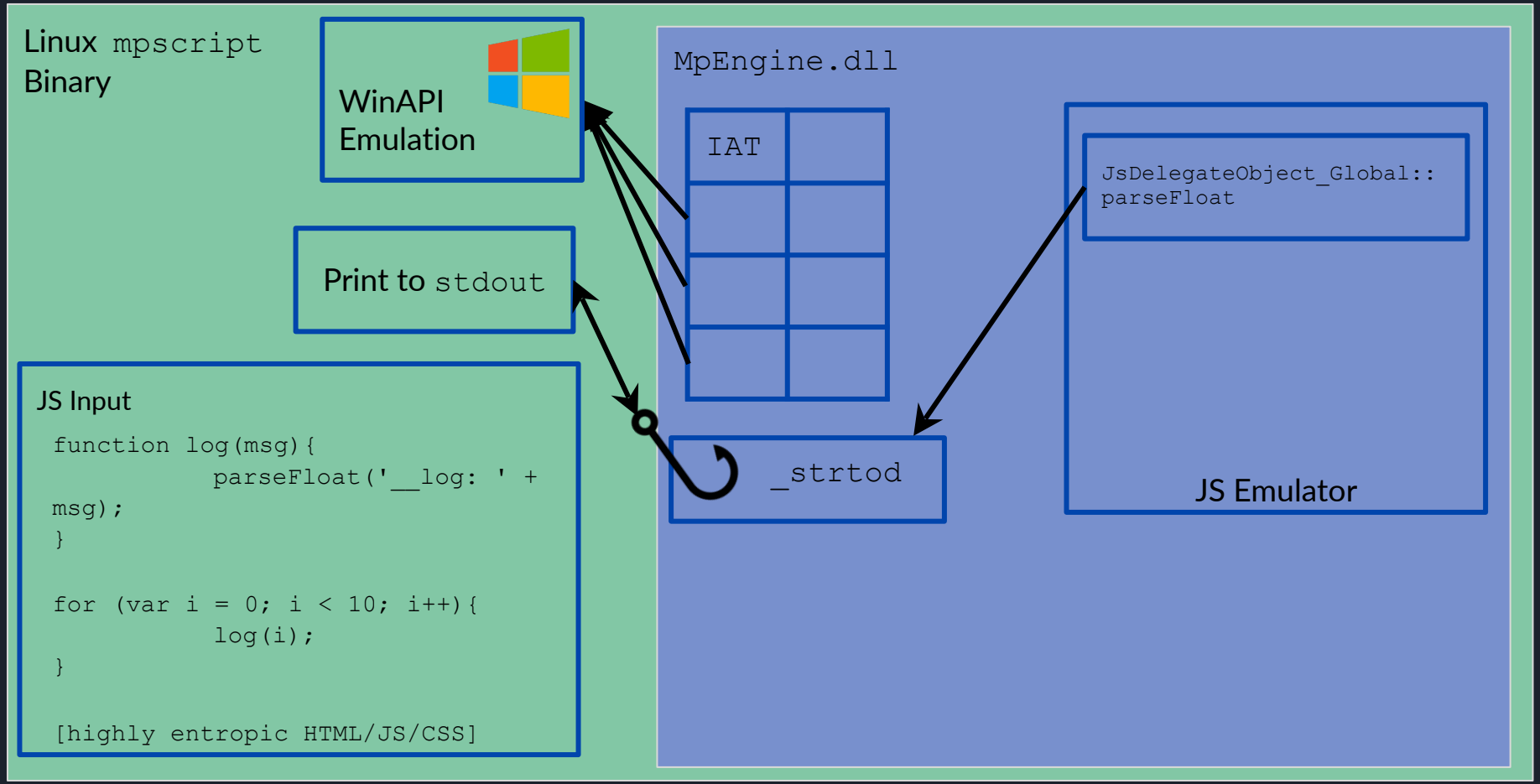
Taviso's `loadlibrary` https://github.com/taviso/loadlibrary



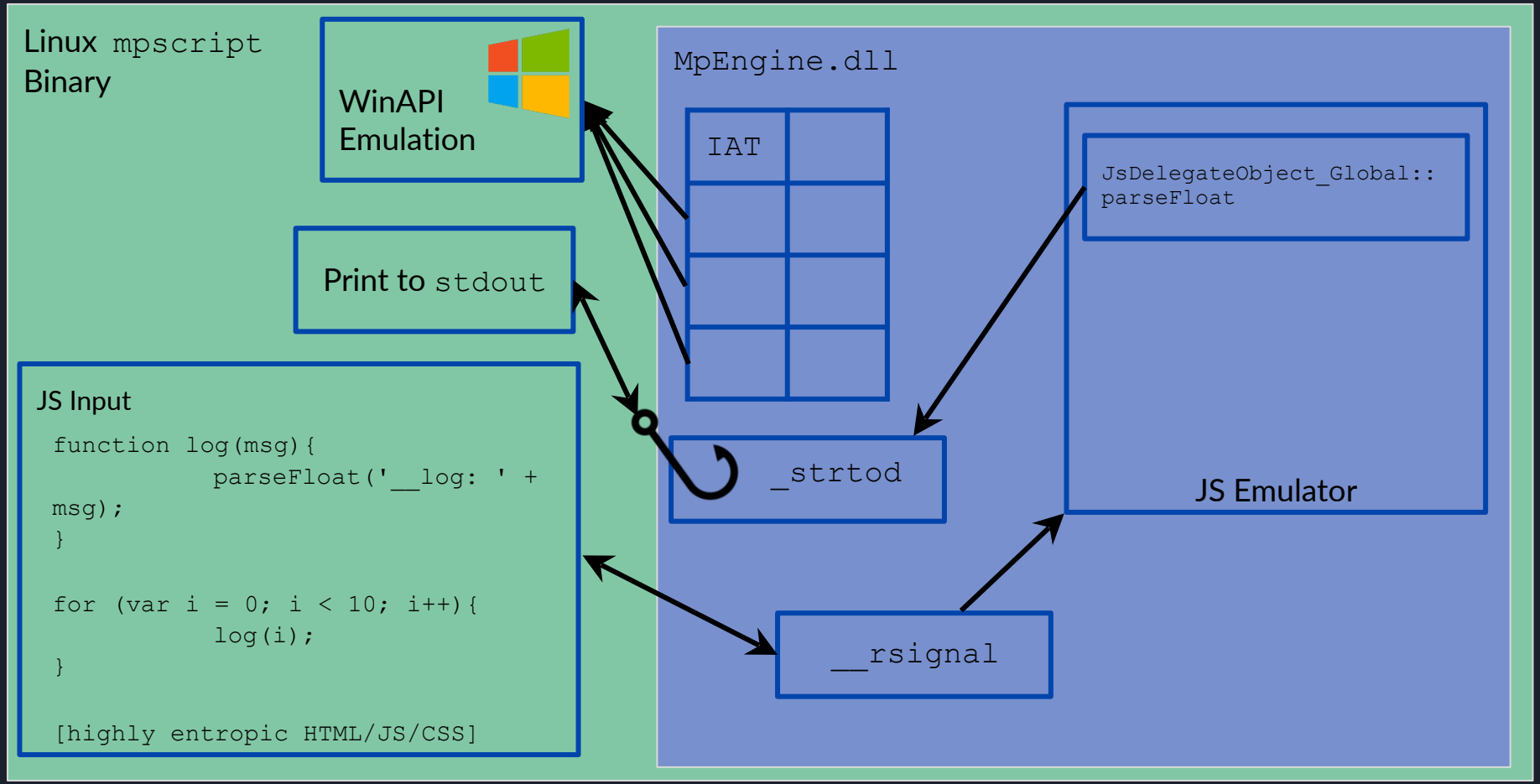
Taviso's `loadlibrary` https://github.com/taviso/loadlibrary



Taviso's loadlibrary https://github.com/taviso/loadlibrary



Taviso's loadlibrary https://github.com/taviso/loadlibrary



Our Shell vs `loadlibrary`

Engine	Our Shell	Taviso's `loadlibrary` https://github.com/taviso/loadlibrary
Platform / Debugger	Windows WinDbg with native PDBs	Linux GDB with custom symbol files
Initialization	None, just `LoadLibrary` the DLL	Full, requires corresponding VDM files
Scanning	Direct call into JS engine	Call main entry point for any AV scan
Output	Hooked VTable, using AV callbacks	Function hook for `_strtod`
Configuration	Relies on default emulation parameters set in engine	Get parameters from VDM files

Demo 1

```
(function() {
  var msg = "Hello REcon";
  for (var i = 0; i < 5; i++) {
    print(i + ":" + msg)
  }
})()
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
    a. JS Language
    b. Types
    c. Memory Management
    d. Fingerprinting
4. Vulnerability Discussion
5. Conclusion

JS Engine

- Proprietary, best I can tell no shared code with other engines
- Focus on JS language implementation, not full browser emulation
- Code is interpreted, not JITed
- Written in C++
- Single-threaded
- AV Emulator
  - Time & memory constrained
  - Analysis callbacks
- ~1200 functions

ECMAScript 3-ish Language Features

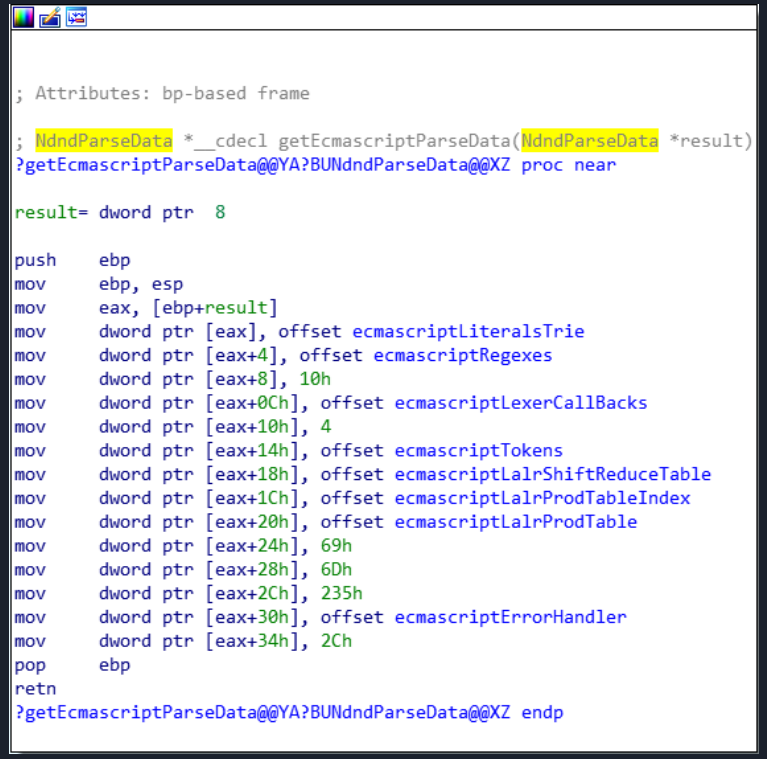
Implemented

- `if / else`
- `try / catch`
- `for-in`
- `switch` statements (broken! - no fallthrough)
- `var` declarations
- Regular expressions
- Error objects
- Function scoping
- Hoisting
- Object literals
- JS: timeouts

Not Implemented

- `for-of`
- Getter / setter methods
- Collections: `set`, `map`
- Classes
- Proxies
- Reflect
- Generators / `yield` statement
- `let` declarations
- Promises
- Typed arrays

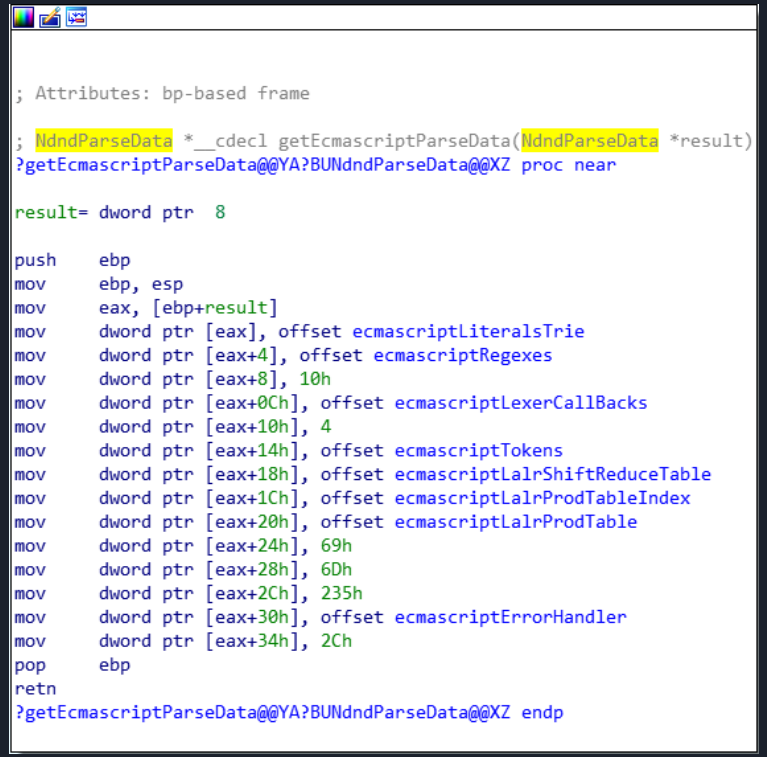
Parsing & Evaluation



```
; Attributes: bp-based frame

; NdndParseData *__cdecl getEcmascriptParseData(NdndParseData *result)
?getEcmascriptParseData@@YA?BUNdndParseData@@XZ proc near

result= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+result]
mov     dword ptr [eax], offset ecmascriptLiteralsTrie
mov     dword ptr [eax+4], offset ecmascriptRegexes
mov     dword ptr [eax+8], 10h
mov     dword ptr [eax+0Ch], offset ecmascriptLexerCallBacks
mov     dword ptr [eax+10h], 4
mov     dword ptr [eax+14h], offset ecmascriptTokens
mov     dword ptr [eax+18h], offset ecmascriptLalrShiftReduceTable
mov     dword ptr [eax+1Ch], offset ecmascriptLalrProdTableIndex
mov     dword ptr [eax+20h], offset ecmascriptLalrProdTable
mov     dword ptr [eax+24h], 69h
mov     dword ptr [eax+28h], 6Dh
mov     dword ptr [eax+2Ch], 235h
mov     dword ptr [eax+30h], offset ecmascriptErrorHandler
mov     dword ptr [eax+34h], 2Ch
pop     ebp
retn
?getEcmascriptParseData@@YA?BUNdndParseData@@XZ endp
```

`NdndParseData` struct is populated with
EMCAScript-specific parsing functions and parameters

Parsing & Evaluation



```
; Attributes: bp-based frame

; NdndParseData *__cdecl getEcmascriptParseData(NdndParseData *result)
?getEcmascriptParseData@@YA?BUNdndParseData@@XZ proc near

result= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+result]
mov     dword ptr [eax], offset ecmascriptLiteralsTrie
mov     dword ptr [eax+4], offset ecmascriptRegexes
mov     dword ptr [eax+8], 10h
mov     dword ptr [eax+0Ch], offset ecmascriptLexerCallBacks
mov     dword ptr [eax+10h], 4
mov     dword ptr [eax+14h], offset ecmascriptTokens
mov     dword ptr [eax+18h], offset ecmascriptLalrShiftReduceTable
mov     dword ptr [eax+1Ch], offset ecmascriptLalrProdTableIndex
mov     dword ptr [eax+20h], offset ecmascriptLalrProdTable
mov     dword ptr [eax+24h], 69h
mov     dword ptr [eax+28h], 6Dh
mov     dword ptr [eax+2Ch], 235h
mov     dword ptr [eax+30h], offset ecmascriptErrorHandler
mov     dword ptr [eax+34h], 2Ch
pop     ebp
retn
?getEcmascriptParseData@@YA?BUNdndParseData@@XZ endp
```

`NdndParseData` struct is populated with EMCAScript-specific parsing functions and parameters

After parsing - everything passes through `JsTree::run`

Parsing and `run` may be invoked multiple times during execution - user defined callbacks, timeouts, evals, etc...

```
v5 = ProgramTree::Impl::astToProgTree(jsState->m_builder.m_pimpl._Myptr, &ast);
if ( v5 && JsTree::run(v5, jsState, 0) )
```

Parsing & Evaluation

```
; Attributes: bp-based frame

; NdndParseData *__cdecl getEcmascriptParseData(NdndParseData *result)
?getEcmascriptParseData@@YA?BUNdndParseData@@XZ proc near

result= dword ptr  8

push    ebp
mov     ebp, esp
mov     eax, [ebp+result]
mov     dword ptr [eax], offset ecmascriptLiteralsTrie
mov     dword ptr [eax+4], offset ecmascriptRegexes
mov     dword ptr [eax+8], 10h
mov     dword ptr [eax+0Ch], offset ecmascriptLexerCallBacks
mov     dword ptr [eax+10h], 4
mov     dword ptr [eax+14h], offset ecmascriptTokens
mov     dword ptr [eax+18h], offset ecmascriptLalrShiftReduceTable
mov     dword ptr [eax+1Ch], offset ecmascriptLalrProdTableIndex
mov     dword ptr [eax+20h], offset ecmascriptLalrProdTable
mov     dword ptr [eax+24h], 69h
mov     dword ptr [eax+28h], 6Dh
mov     dword ptr [eax+2Ch], 235h
mov     dword ptr [eax+30h], offset ecmascriptErrorHandler
mov     dword ptr [eax+34h], 2Ch
pop     ebp
retn
?getEcmascriptParseData@@YA?BUNdndParseData@@XZ endp
```

`NdndParseData` struct is populated with EMCAScript-specific parsing functions and parameters

After parsing - everything passes through `JsTree::run`

Parsing and `run` may be invoked multiple times during execution - user defined callbacks, timeouts, evals, etc...

```
v5 = ProgramTree::Impl::astToProgTree(jsState->m_builder.m_pimpl._Myptr, &ast);
if ( v5 && JsTree::run(v5, jsState, 0) )
```

Various `::eval` functions implement the interpretation of JS statements

- EvaluateSignature<ulong>::Eval(char const *)
- EvaluateSignature<ulong>::EvaluateSignature<ulong>(PEFileWriter *,PtrType const
- JavaScriptInterpreter::eval(char const *,uint,JavaScriptInterpreter::Params &)
- JsArgumentsTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsArrayLiteralTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsAssignExprTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsBinaryExprTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsBlockStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsBooleanLiteralTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsCallExprTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsCaseStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsCatchStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsConditionalExprTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsConstExprTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsControlFlowStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsDelegateObject_Global::eval(JsRuntimeState &,std::vector<uint,std::allocator<uin
- JsEmptyStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsExprStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsForInStatmentTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsForStatementTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsFuncExprTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsIdentifierTree::eval(JsTree::CoroutineState &,JsRuntimeState &)
- JsIfStmtTree::eval(JsTree::CoroutineState &,JsRuntimeState &)

Example Stack Trace During Callback

```
(function(){
  var x = new Array(1,2,3);
  x.toString = function(){
    return isNaN("foobar")
  };
  var y = new String(x);
})()
```

Example Stack Trace During Callback

Stack Trace (Upside Down) - breakpoint on `isNan`

Initial evaluation

`JavaScriptInterpreter::eval`

`JsTree::run`

`JsNewExprTree::eval`

```
(function(){
  var x = new Array(1,2,3);
  x.toString = function(){
    return isNaN("foobar")
  };
  var y = new String(x);
})()
```

Example Stack Trace During Callback

Stack Trace (Upside Down) - breakpoint on `isNan`

Initial evaluation

```
JavaScriptInterpreter::eval
JsTree::run
JsNewExprTree::eval
```

Allocating a new String

```
preInvokeFunctionThrows
JsConstructor_String::call
newStringObjectThrows
newStringObjectThrowsT<JsStringObject>
```

```
(function(){
  var x = new Array(1,2,3);
  x.toString = function(){
    return isNaN("foobar")
  };
  var y = new String(x);
})()
```

Example Stack Trace During Callback

Stack Trace (Upside Down) - breakpoint on `isNan`

Initial evaluation

```
JavaScriptInterpreter::eval
JsTree::run
JsNewExprTree::eval
```

Allocating a new String

```
preInvokeFunctionThrows
JsConstructor_String::call
newStringObjectThrows
newStringObjectThrowsT<JsStringObject>
```

Reentrance into JS state for `.toString`

```
JsTree::run
JsCallExprTree::eval
preInvokeFunctionThrows
JsFunctionProxyObject<JsDelegateObject_Global>::call
JsDelegateObject_Global::delegate
JsDelegateObject_Global::isNan
```

```
(function(){
  var x = new Array(1,2,3);
  x.toString = function(){
    return isNaN("foobar")
  };
  var y = new String(x);
})()
```

Regex Engine

- Regex engine accessible from JS
- Regex parsing and compiling system - big attack surface

```
mpscript> (function(){
    var str = "Hello REcon!";
    var i = str.search(/REcon/i);
    print(i)
})()
triggerEvent(): str_valueof
triggerEvent(): str_search
print(): 6
```

```
 # ChildEBP RetAddr
00 00ddcb18 7067abf3 mpengine!JsRegExpEngine::Compiler::compileOp [h:\av\
01 00ddcbc0 7067af98 mpengine!JsRegExpEngine::Compiler::compile+0x11f [h:
02 00ddcc10 706286fd mpengine!JsRegExpEngine::Compiler::operator()+0x5f [
03 00ddcc40 7069c1c9 mpengine!JsRegExpEngine::init+0x4d [h:\av\engine rel
04 00ddcd08 70626825 mpengine!regExpExecThrows+0x117 [h:\av\engine releas
05 00ddcd64 706a9212 mpengine!JsDelegateObject_StringProto::search+0x13a
06 00ddcd78 706a6b6c mpengine!JsDelegateObject_StringProto::delegate+0xd6
07 00ddcd94 706aa66d mpengine!JsFunctionProxyObject<JsDelegateObject_Stri
08 00ddcdc8 706aae0d mpengine!preInvokeFunctionThrows+0xf9 [h:\av\engine
09 00ddce28 706b163b mpengine!JsCallExprTree::eval+0x1bd [h:\av\engine re
0a 00ddce64 706a0a87 mpengine!JsTree::run+0x134 [h:\av\engine released\am
0b 00ddd088 00a53c89 mpengine!JavaScriptInterpreter::eval+0x212 [h:\av\en
0c 00ddf7f8 00a54a7e JsShell!wmain+0x369 [c:\users\alex\documents\visual
0d 00ddf80c 00a548e0 JsShell!invoke_main+0x1e [f:\dd\vctools\crt\vcstartu
0e 00ddf864 00a5477d JsShell!__scrt_common_main_seh+0x150 [f:\dd\vctools\
0f 00ddf86c 00a54a98 JsShell!__scrt_common_main+0xd [f:\dd\vctools\crt\vc
10 00ddf874 76377c04 JsShell!wmainCRTStartup+0x8 [f:\dd\vctools\crt\vcsta
11 00ddf888 77c3b90f KERNEL32!BaseThreadInitThunk+0x24
12 00ddf8d0 77c3b8da ntdll!__RtlUserThreadStart+0x2f
13 00ddf8e0 00000000 ntdll!_RtlUserThreadStart+0x1b
```

```
JsRegExpEngine::Compiler::NfaFragment::NfaFragment(JsRegExpEngine::Compi
JsRegExpEngine::Compiler::NfaFragment::NfaFragment(JsRegExpEngine::Compi
JsRegExpEngine::Compiler::NfaFragment::~NfaFragment(void)
JsRegExpEngine::Compiler::compile(std::vector<JsRegExpEngine::Compiler::Tok
JsRegExpEngine::Compiler::compileOp(std::stack<JsRegExpEngine::Compiler::N
JsRegExpEngine::Compiler::convertToRpn(std::vector<JsRegExpEngine::Compil
JsRegExpEngine::Compiler::isEscCharClass(char)
JsRegExpEngine::Compiler::operator()(std::basic_string<char,std::char_traits<c
JsRegExpEngine::Compiler::parseEscapedChar(std::basic_string<char,std::char_
JsRegExpEngine::Compiler::tokenise(std::basic_string<char,std::char_traits<cha
JsRegExpEngine::Compiler::tokeniseCharacterClass(std::basic_string<char,std::
JsRegExpEngine::Compiler::tokeniseCountedRepetition(std::basic_string<char,std::char_traits<char>,std::a
JsRegExpEngine::Compiler::tokeniseEscapedCharClass(std::basic_string<char,std::char_traits<char>,std::a
```

Antivirus Integration

- `NscriptJSMonitor` class implements callbacks used to monitor execution for antivirus purposes
  - `NscriptJSMonitor::analyse` receives info about specific script actions
- `JsRuntimeState::triggerEvent` is called to register events (function calls)

```
void __thiscall NscriptJSMonitor::NscriptJSMonitor(
{
  NscriptJSMonitor *v2; // esi
  void *v3; // eax
  SCAN_REPLY *v4; // [esp+0h] [ebp-20h]

  v2 = this;
  this->vfptr = &NscriptJSMonitor::`vftable';
  this->m_lastEventWasRefErrorCountdown = 0;
  std::basic_string<char,std::char_traits<char>,st
  v2->m_eLimitMet = 0;
  v2->m_write = 0;
  v2->m_eval = 0;
  v2->m_func = 0;
  v2->m_sr = sr;
  v3 = siga_init(4, v4);
  v2->m_BestResult = -1;
  v2->m_sigAttrHandle = v3;
  v2->m_BestResultIsSuspicious = 0;
```

```
v13 = siga_cksig(v7->m_sr, v9, &attr, &d, 0, 0);
if ( v13 == 1 )
{
  v24 = kpopobject(d.DetectionRecId);
  if ( !v24 || (v25 = v24[1], p_name = 0, get_threat_name(v25, &p_name, v37), (newname = p_name) == 0) )
    newname = "n/a";
  v26 = v7->m_BestResult;
  if ( v26 == -1 || v7->m_BestResultIsSuspicious )
    goto LABEL_76;
  v27 = kpopobject(v26);
  if ( !v27 || (v28 = v27[1], p_name = 0, get_threat_name(v28, &p_name, v37), (v29 = p_name) == 0) )
    v29 = "n/a";
  if ( IsBetterMatch(v7->m_sr, v29, newname) )
  {
LABEL_76:
    v7->m_BestResult = d.DetectionRecId;
    v7->m_BestResultIsSuspicious = 0;
```

```
JsRuntimeState::triggerEvent(jsState, 0, "date_setdate", 0, 0, v5, v5)
```

```
if ( v6 == 34 )
  result = JsRuntimeState::triggerEvent(jsState, 0, "bool_valueof", "false", 5u, "false", "false");
else
  result = JsRuntimeState::triggerEvent(jsState, 0, "bool_valueof", "true", 4u, "true", "true");
```

JsRuntimeState

Large struct that stores entire JS runtime state: variables, function arguments, global utility functions, timeouts, execution parameters, etc...

```
dt mpengine!JsRuntimeState
+0x000 __VFN_table : Ptr32
+0x004 m_callStack       : Ptr32 CallStack
+0x008 m_heap            : JsHeap
+0x070 m_exeCtxStack     :
std::vector<JsRuntimeState::ExecutionContext,std::
allocator<JsRuntimeState::ExecutionContext> >
+0x07c m_globalObj       : Ptr32 JsObject
+0x080 m_complType       :
JsRuntimeState::CompletionType
+0x084 m_complValue      : Uint4B
+0x088 m_complTarget     : Uint4B
+0x08c m_labelStack      : std::vector<unsigned
int,std::allocator<unsigned int> >
+0x098 m_conversionValue : Uint4B
+0x09c m_conversionValueType : JsValueType
+0x0a0 m_builtins        :
std::vector<JsRuntimeState::Builtin,std::allocator
<JsRuntimeState::Builtin> >
+0x0ac m_callerPropHash : Uint4B
+0x0b0 m_callerValue     : Uint4B
+0x0b4 m_callerDepth     : Uint4B
+0x0b8 m_exeLimit        : Uint4B
+0x0bc m_exeCounter      : Uint4B
+0x0c0 m_regexpLimit     : Uint4B
+0x0c4 m_runLevel        : Uint4B
+0x0c8 m_domWrapper      : Ptr32
HtmlDocumentProvider
+0x0cc m_emptyPageDom    : HtmlDocument
+0x0dc m_monitor         : Ptr32
JsEvaluationMonitor
+0x0e0 m_builder         : ProgramTree
+0x0e4 m_monitorHeartBeatIsHappy : Bool
+0x0e8 m_monitorHeartBeatDelayCount : Uint4B
+0x0ec m_shortEventBuf   : [80] Char
+0x13c m_logBuf          : Ptr32 Char
+0x140 m_timeOutCallBacks :
std::list<JsRuntimeState::TimeOutCallBack,std:
:allocator<JsRuntimeState::TimeOutCallBack> >
+0x148 m_nextCallBackId : Int4B
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
    a. JS Language
    b. Types
    c. Memory Management
    d. Fingerprinting
4. Vulnerability Discussion
5. Conclusion

dt mpengine!JsObject

```
+0x00 __VFN_table          : Ptr32
+0x04 isValid              : Bool
+0x08 m_type               : JsValueType
+0x0c m_propertyNames      :
    std::map<
        unsigned int,
                 std::basic_string<char,std::char_traits<char>,std::allocator<char> > const ,
                 std::less<unsigned int>,
                 std::allocator<std::pair<unsigned int const ,
         std::basic_string<char,std::char_traits<char>,std::allocator<char> > const > > >
+0x14 m_properties         :
    std::map<
        unsigned int,
        JsObject::Property,
        std::less<unsigned int>,
        std::allocator<std::pair<unsigned int const ,JsObject::Property> > >
+0x1c m_propertyArray      : std::vector<unsigned int,std::allocator<unsigned int> >
+0x28 m_htmlDoc            : HtmlDocument::Iterator
+0x30 m_proto              : Ptr32 JsObject
+0x34 m_class              : Ptr32 Char
+0x38 m_value              : Uint4B
```

Parent class for JS object types

Enum `JsValueType`

0. `Empty`
1. `Undefined`
2. `Null`
3. `Boolean`
4. `String`
5. `Number`
6. `Date` (present in PDBs, but not used, instead compared by class “Date”)
7. `Object`
8. `FunctionObject`
9. `RegExpObject`
a. `Reference`
b. `List`
c. `BadValueType`

dt mpengine!JsArrayObject

```
+0x00 __VFN_table              : Ptr32
+0x04 isValid                  : Bool
+0x08 m_type                   : JsValueType
+0x0c m_propertyNames          :
     std::map<
          unsigned int,
                std::basic_string<char,std::char_traits<char>,std::allocator<char> > const ,
                std::less<unsigned int>,
                std::allocator<std::pair<unsigned int const ,
           std::basic_string<char,std::char_traits<char>,std::allocator<char> > const > > >
+0x14 m_properties             :
     std::map<
          unsigned int,
          JsObject::Property,
          std::less<unsigned int>,
          std::allocator<std::pair<unsigned int const ,JsObject::Property> > >
+0x1c m_propertyArray          : std::vector<unsigned int,std::allocator<unsigned int> >
+0x28 m_htmlDoc               : HtmlDocument::Iterator
+0x30 m_proto                  : Ptr32 JsObject
+0x34 m_class                  : Ptr32 Char
+0x38 m_value                  : Uint4B
+0x3c m_lengthPropHash         : Uint4B
```

dt mpengine!JsRegExpObject

```
+0x00 __VFN_table             : Ptr32
+0x04 isValid                 : Bool
+0x08 m_type                  : JsValueType
+0x0c m_propertyNames         :
     std::map<
          unsigned int,
                std::basic_string<char,std::char_traits<char>,std::allocator<char> > const ,
                std::less<unsigned int>,
                std::allocator<std::pair<unsigned int const ,
           std::basic_string<char,std::char_traits<char>,std::allocator<char> > const > > >
+0x14 m_properties            :
     std::map<
          unsigned int,
          JsObject::Property,
          std::less<unsigned int>,
          std::allocator<std::pair<unsigned int const ,JsObject::Property> > >
+0x1c m_propertyArray         : std::vector<unsigned int,std::allocator<unsigned int> >
+0x28 m_htmlDoc               : HtmlDocument::Iterator
+0x30 m_proto                 : Ptr32 JsObject
+0x34 m_class                 : Ptr32 Char
+0x38 m_value                 : Uint4B
+0x3c m_patStr                :
std::basic_string<char,std::char_traits<char>,std::allocator<char> >
+0x54 m_flags                 : Uint4B
```

dt mpengine!JsObject

```
+0x00 __VFN_table    : Ptr32
+0x04 isValid        : Bool
+0x08 m_type         : JsValueType
+0x0c m_propertyNames : std::map<...>
+0x14 m_properties   : std::map<...>
+0x28 m_htmlDoc      : HtmlDocument::...
+0x30 m_proto        : Ptr32 JsObject
+0x34 m_class        : Ptr32 Char
+0x38 m_value        : Uint4B
```

"Date"

dt mpengine!JsDate

```
+0x00 __VFN_table: Ptr32
+0x04 isValid    : Bool
+0x08 m_type     : JsValueType
+0x10 m_time     : Int8B
```

```
bool __stdcall JsDelegateObject_DateProto::getThisValueThrows(JsRuntimeState *jsState, const
{
  const char *classStr; // [esp+0h] [ebp-8h]
  JsObject *thisObj; // [esp+4h] [ebp-4h] MAPDST

  thisObj = 0;
  if ( !JsRuntimeState::getThisPtr(jsState, &thisObj) )
    return 0;
  classStr = 0;
  if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "Date", 5u) )
    return JsRuntimeState::throwNativeError(jsState, BltinTypeError, notDateErrMsg);
  *date = valueToPtr<JsDate>(thisObj->m_value);
  return 1;
}
```

String Primitive Types

- `JsBufString`
  - `char *` buffer
- `JsSubString`
  - Reference to some slice of another string
- `JsRefString`
  - Reference to a `.data` section string
- `JsConcatString`
  - Tree of concatenated string elements

```
dt mpengine!JsConcatString
+0x00 __VFN_table: Ptr32
+0x04 isValid     : Bool
+0x08 m_type      : JsValueType
+0x0c m_numBytes  : Uint4B
+0x10 m_lhs       : Uint4B
+0x14 m_rhs       : Uint4B
+0x18 m_refDepth  : Uint4B
```

```
dt mpengine!JsBufString
+0x00 __VFN_table: Ptr32
+0x04 isValid     : Bool
+0x08 m_type      : JsValueType
+0x0c m_numBytes  : Uint4B
+0x10 m_str       : Ptr32 Char
```

```
dt mpengine!JsSubString
+0x00 __VFN_table: Ptr32
+0x04 isValid     : Bool
+0x08 m_type      : JsValueType
+0x0c m_numBytes  : Uint4B
+0x10 m_parent    : Uint4B
+0x14 m_offset    : Uint4B
+0x18 m_refDepth  : Uint4B
```

```
dt mpengine!JsRefString
+0x00 __VFN_table: Ptr32
+0x04 isValid     : Bool
+0x08 m_type      : JsValueType
+0x0c m_numBytes  : Uint4B
+0x10 m_str       : Ptr32 Char
```

Method Dispatch

Each JS object class has a `delegate` function that dispatches calls to object methods

```
char __thiscall JsDelegateObject_Boolean::delegate(JsDelegateObject_Boolean *this,
{
  if ( !method )
    return JsDelegateObject_Boolean::toString(this, jsState, this, isConstructor);
  if ( method == 1 )
    return JsDelegateObject_Boolean::valueOf(jsState, this, isConstructor);
  return 0;
}
```

```
bool __thiscall JsDelegateObject_StringProto::delegate(J
{
  bool result; // al

  switch ( method )
  {
    case 0:
    case 1:
      result = JsDelegateObject_StringProto::valueOf(this, jsState, this, isConstructor);
      break;
    case 2:
      result = JsDelegateObject_StringProto::charAt(this, jsState, argsList, isConstructo
      break;
    case 3:
      result = JsDelegateObject_StringProto::charCodeAt(this, jsState, argsList, isConstr
      break;
    case 4:
      result = JsDelegateObject_StringProto::concat(this, jsState, argsList, isConstructo
      break;
    case 5:
      result = JsDelegateObject_StringProto::indexOf(jsState, argsList, isConstructor);
      break;
    case 6:
      result = JsDelegateObject_StringProto::lastIndexOf(this, jsState, argsList, isConst
      break;
    case 7:
      result = JsRuntimeState::throwNativeError(
                 jsState,
                 BltinTypeError,
                 "String.prototype.localeCompare is unimplemented");
      break;
```

```
char __thiscall JsDelegateObject_Node::delegate(JsDelegateObject_Node
{
  switch ( method )
  {
    case 0:
      return JsDelegateObject_Node::appendChild(this, jsState, argsLis
    case 1:
      return JsDelegateObject_Node::insertBefore(this, jsState, argsLi
    case 2:
      return JsDelegateObject_Node::getElementsByTagName(this, jsState
    case 3:
      return JsDelegateObject_Node::createElement(jsState, argsList, i
    case 4:
      return JsDelegateObject_Node::createTextNode(this, jsState, args
    case 5:
      return JsDelegateObject_Node::getElementById(this, jsState, args
    case 6:
      return JsDelegateObject_Node::write(this, jsState, argsList, isC
  }
  return 0;
}
```

Unimplemented Methods

Object.prototype:

- toLocaleString
- propertyIsEnumerable
- isPrototypeOf

Array.prototype:

- toLocaleString
- unshift
- concat (for sparse arrays)
- sort

Number.prototype:

- toLocaleString
- toPrecision
- toFixed
- toExponential

String.prototype.localeCompare

encodeURI

```
JsRuntimeState::throwNativeError(
  jsState,
  BltinTypeError,
  "Array.prototype.concat() sparse arrays not supported.");
```

```
case 9:
  result = JsRuntimeState::throwNativeError(jsState, BltinTypeError, "Array.prototype.sort is unimplemented");
  break;
case 10:
  result = JsDelegateObject_ArrayProto::splice(this, jsState, argsList, isConstructor);
  break;
case 11:
  result = JsRuntimeState::throwNativeError(jsState, BltinTypeError, "Array.prototype.unshift is unimplemented");
  break;
```

Demo 2

```
(function() {
  var i = 0;
  switch (i) {
    case 0:
      print("0");
    case 1:
      print("1");
  }
})()
```

```
(function() {
  var x = [3, 2, 1];
  x.sort()
})()
```

```
(function() {
  String(parseFloat([1, 2, 3].join())).toString()
})()
```

Type Checking

Explicit Checking

- `m_type`
- `m_class`

Type Checking

Explicit Checking

- `m_type`
- `m_class`

```
if ( getValueType(v5) == 9 )
{
  v7 = valueToPtr<JsRegExpObject>(v5);
  regExpObj = v7;
}
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "Date", 5u) )
  return JsRuntimeState::throwNativeError(jsState, BltinTypeError, notDateErrMsg);
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "String", 7u) )
```

```
if ( thisPtr && JsObject::getClass(thisPtr, &classStr) && !memcmp(classStr, "Number", 7u) )
```

Type Checking

Casting

```
if ( !JsTree::run(&v4->m_toStringTree.vfptr, jsState, 1) )
  return 0;
```

```
if ( JsDelegateObject_StringProto::toStringThrows(&newBufStr, jsState, v4, &newBufStr) )
```

Explicit Checking

- `m_type`
- `m_class`

```
if ( getValueType(v5) == 9 )
{
  v7 = valueToPtr<JsRegExpObject>(v5);
  regExpObj = v7;
}
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "Date", 5u) )
  return JsRuntimeState::throwNativeError(jsState, BltinTypeError, notDateErrMsg);
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "String", 7u) )
```

```
if ( thisPtr && JsObject::getClass(thisPtr, &classStr) && !memcmp(classStr, "Number", 7u) )
```

Type Checking

Casting

```
if ( !JsTree::run(&v4->m_toStringTree.vfptr, jsState, 1) )
  return 0;
```

```
if ( JsDelegateObject_StringProto::toStringThrows(&newBufStr, jsState, v4, &newBufStr) )
```

Explicit Checking

- `m_type`
- `m_class`

```
if ( getValueType(v5) == 9 )
{
  v7 = valueToPtr<JsRegExpObject>(v5);
  regExpObj = v7;
}
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "Date", 5u) )
  return JsRuntimeState::throwNativeError(jsState, BltinTypeError, notDateErrMsg);
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "String", 7u) )
```

```
if ( thisPtr && JsObject::getClass(thisPtr, &classStr) && !memcmp(classStr, "Number", 7u) )
```

```
v4 = JsObject::genPropHash("length", 0);
v5 = thisObj;
if ( JsObject::get(thisObj, jsState, v4, &lengthValue)
  && JsRuntimeState::toUint32Throws(jsState, lengthValue, &length) )
```

Duck Typing

Type Checking

```
if ( !JsTree::run(&v4->m_toStringTree.vfptr, jsState, 1) )
  return 0;
```

Casting

```
if ( JsDelegateObject_StringProto::toStringThrows(&newBufStr, jsState, v4, &newBufStr) )
```

Explicit Checking

- m_type
- m_class

```
if ( getValueType(v5) == 9 )
{
  v7 = valueToPtr<JsRegExpObject>(v5);
  regExpObj = v7;
}
```

Redefinition?

```
(function x(){
      var x = new Array();
      x.foo = (new Date()).getMonth;
      x.foo();
})()
triggerEvent(): err_typeerror
triggerEvent(): error_tostring
Log(): <NA>: 0: uncaught exception: TypeError:
Date.prototype.getMonth() must be called only for Dates
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "Date", 5u) )
  return JsRuntimeState::throwNativeError(jsState, BltinTypeError, notDateErrMsg);
```

```
if ( !thisObj || !JsObject::getClass(thisObj, &classStr) || memcmp(classStr, "String", 7u) )
```

```
if ( thisPtr && JsObject::getClass(thisPtr, &classStr) && !memcmp(classStr, "Number", 7u) )
```

```
v4 = JsObject::genPropHash("length", 0);
v5 = thisObj;
if ( JsObject::get(thisObj, jsState, v4, &lengthValue)
  && JsRuntimeState::toUint32Throws(jsState, lengthValue, &length) )
```

Duck Typing

DOM Emulation

- Objects may have `m_html` pointer to `HtmlDocument::Iterator` for iteration over `std::vector` of `HtmlDocument::Impl::Node` objects

- "`document`" object declared globally to allow HTML interaction

- Minimal implementation allows creation and manipulation of document elements

```
dt mpengine!HtmlDocument::Impl::Node
+0x00 type : HtmlDocument::NodeType
+0x04 name : std::pair<char const *,unsigned int>
+0x0c data : std::pair<char const *,unsigned int>
+0x14 html : std::pair<char const *,unsigned int>
+0x1c parent : Uint4B
+0x20 nextSibling : Uint4B
+0x24 firstKid : Uint4B
+0x28 lastKid : Uint4B
```

```
v66 = JsObject::genPropHash("document", 0);
if ( JsRuntimeState::declare(v3, v66, v6) )
```

```
JsDelegateObject_Navigator::justReturnFalse(JsRuntimeState &,std::vector<uint,
JsDelegateObject_Node::appendChild(JsRuntimeState &,std::vector<uint,std::allo
JsDelegateObject_Node::createElement(JsRuntimeState &,std::vector<uint,std::a
JsDelegateObject_Node::createTextNode(JsRuntimeState &,std::vector<uint,std:
JsDelegateObject_Node::delegate(int,JsRuntimeState &,std::vector<uint,std::allo
JsDelegateObject_Node::getElementById(JsRuntimeState &,std::vector<uint,std:
JsDelegateObject_Node::getElementsByTagName(JsRuntimeState &,std::vector<u
JsDelegateObject_Node::insertBefore(JsRuntimeState &,std::vector<uint,std::allo
JsDelegateObject_Node::write(JsRuntimeState &,std::vector<uint,std::allocator<
JsDelegateObject_NodeList::fold(HtmlDocument::Iterator &&,bool,std::function<b
JsDelegateObject_NodeList::getLength(JsRuntimeState &,JsNodeListObject *,uint
JsDelegateObject_NodeList::item(JsRuntimeState &,JsNodeListObject *,uint,uint &
JsDelegateObject_NodeList::item(JsRuntimeState &,std::vector<uint,std::allocato
```

Object Properties

- JavaScript objects are associative arrays mapping property name → value
  - Backed by `std::_Tree`
- Strings and sparse array numeric properties are hashed to index into map
- Not maintained for primitive non-object types

```
char __thiscall JsObject::getLocal(JsObject *this, JsRuntimeState *__formal, unsigned int
{
  unsigned int retrievedValue; // ecx
  unsigned int propNum; // [esp+4h] [ebp-4h]

  propNum = 0;
  if ( JsObject::propIsNumeric(propHash, &propNum)
    && propNum < (this->m_propertyArray._Mylast - this->m_propertyArray._Myfirst) >> 2 )
  {
    retrievedValue = this->m_propertyArray._Myfirst[propNum];
    if ( retrievedValue == 2 )
      return 0;
setReturnValue:
    *value = retrievedValue;
    return 1;
  }
  std::_Tree<std::_Tmap_traits<unsigned int,VfsFileData::Stat,std::less<unsigned int>,std
    &this->m_properties,
    &propNum,
    &propHash);
  if ( propNum != this->m_properties._Myhead )
  {
    retrievedValue = *(propNum + 20);
    goto setReturnValue;
  }
  return 0;
}
```

dt mpengine!JsObject

```
+0x00 __VFN_table          : Ptr32
+0x04 isValid              : Bool
+0x08 m_type               : JsValueType
+0x0c m_propertyNames      :
    std::map<
        unsigned int,
                 std::basic_string<char,std::char_traits<char>,std::allocator<char> > const ,
                 std::less<unsigned int>,
                 std::allocator<std::pair<unsigned int const ,
         std::basic_string<char,std::char_traits<char>,std::allocator<char> > const > > >
+0x14 m_properties         :
    std::map<
        unsigned int,
        JsObject::Property,
        std::less<unsigned int>,
        std::allocator<std::pair<unsigned int const ,JsObject::Property> > >
+0x1c m_propertyArray      : std::vector<unsigned int,std::allocator<unsigned int> >
+0x28 m_htmlDoc            : HtmlDocument::Iterator
+0x30 m_proto              : Ptr32 JsObject
+0x34 m_class              : Ptr32 Char
+0x38 m_value              : Uint4B
```

Parent class for JS object types

Sparse vs. Dense Array Storage

+0x14 m_properties

- “Sparse”
- `std::map` of hash(property) → value
- Stores string values and nonsequential numeric properties
- Numbers converted to string and hashed
- hash(“foo”) → “bar”
- hash(“12”) → “abc”

+0x0c m_propertyNames

- `std::map` of hash(property) → name
- Stores property names
- hash(“foo”) → “foo”

+0x1c m_propertyArray

- “Dense”
- `std::vector` of values
- Stores sequentially numbered properties from 0 to end value
- `[0,1,2]`

```
var x = [0,1,2];
x.foo = “bar”;
x[12] = “abc”;
```

```
char __thiscall JsArrayObject::IsSparseThrows(JsArrayObject *this, JsRuntimeState *jsState, bool *bSparse)
{
  JsArrayObject *v3; // edi
  unsigned int uLen; // [esp+8h] [ebp-8h]
  unsigned int lengthValue; // [esp+Ch] [ebp-4h]

  uLen = 0;
  v3 = this;
  lengthValue = 6;
  if ( !JsObject::get(&this->vfptr, jsState, this->m_lengthPropHash, &lengthValue)
    || !JsRuntimeState::toUint32Throws(jsState, lengthValue, &uLen) )
  {
    return 0;
  }
  if ( jsState->m_complType != 4 )
    *bSparse = ((v3->m_propertyArray._Mylast - v3->m_propertyArray._Myfirst) >> 2) + 1 < uLen;
  return 1;
}
```

Property Hashing

```
lengthPropHash = JsObject::genPropHash("length", 0);
if ( JsObject::get(thisObj, jsState, lengthPropHash, &lengthValue)
  && JsRuntimeState::toUint32Throws(jsState, lengthValue, &pushIndex) )
```

Property Hashing

```
lengthPropHash = JsObject::genPropHash("length", 0);
if ( JsObject::get(thisObj, jsState, lengthPropHash, &lengthValue)
  && JsRuntimeState::toUint32Throws(jsState, lengthValue, &pushIndex) )
```

```
unsigned int __fastcall JsObject::genPropHash(const char *name, unsigned int nameSize)
{
  unsigned int nameSizeLocal; // ecx@3 MAPDST
  unsigned int result; // eax@5
  unsigned int *outNum; // [sp+0h] [bp-18h]@0
  unsigned int number; // [sp+8h] [bp-10h]@2
  StringAdapter str; // [sp+Ch] [bp-Ch]@4

  if ( name )
  {
    number = 0;
    if ( nameSize )
      nameSizeLocal = nameSize;
    else
      nameSizeLocal = strlen(name);
    str.vfptr = &CStringAdapter::`vftable';
    if ( strToInt<unsigned int>(&str, nameSizeLocal, &number, outNum) )
      result = JsObject::genPropHash(number);
    else
      result = propHashStr(name, nameSize);
  }
  else
  {
    result = -1;
  }
  return result;
}
```

Property Hashing

```
lengthPropHash = JsObject::genPropHash("length", 0);
if ( JsObject::get(thisObj, jsState, lengthPropHash, &lengthValue)
  && JsRuntimeState::toUint32Throws(jsState, lengthValue, &pushIndex) )
```

```
unsigned int __fastcall JsObject::genPropHash(const char *name, unsigned int nameSize)
{
  unsigned int nameSizeLocal; // ecx@3 MAPDST
  unsigned int result; // eax@5
  unsigned int *outNum; // [sp+0h] [bp-18h]@0
  unsigned int number; // [sp+8h] [bp-10h]@2
  StringAdapter str; // [sp+Ch] [bp-Ch]@4

  if ( name )
  {
    number = 0;
    if ( nameSize )
      nameSizeLocal = nameSize;
    else
      nameSizeLocal = strlen(name);
    str.vfptr = &CStringAdapter::`vftable';
    if ( strToInt<unsigned int>(&str, nameSizeLocal, &number, outNum) )
      result = JsObject::genPropHash(number);
    else
      result = propHashStr(name, nameSize);
  }
  else
  {
    result = -1;
  }
  return result;
}
```

```
unsigned int __fastcall propHashStr(const char *str, unsigned int len)
{
  unsigned int idx; // esi@3
  int currentHash; // eax@4
  unsigned int pcchLength; // [sp+8h] [bp-4h]@8

  if ( !len )
    len = strlen(str);
  idx = 0;
  if ( !len )
  {
    pcchLength = 0;
    if ( StringCchLengthA(str, 0x7FFFFFFFu, &pcchLength) < 0 )
      return idx | 0x80000000;
    len = pcchLength;
  }
  currentHash = 0x1505;
  if ( len )
  {
    do
      currentHash = str[idx++] + 65599 * currentHash;
    while ( idx < len );
  }
  idx = currentHash;
  return idx | 0x80000000;
}
```

Brute Forcing a Hash Collision

```
unsigned int __fastcall propHashStr(const char *str, unsigned int len)
{
  unsigned int idx; // esi@3
  int currentHash; // eax@4
  unsigned int pcchLength; // [sp+8h] [bp-4h]@8

  if ( !len )
    len = strlen(str);
  idx = 0;
  if ( !len )
  {
    pcchLength = 0;
    if ( StringCchLengthA(str, 0x7FFFFFFFu, &pcchLength) < 0 )
      return idx | 0x80000000;
    len = pcchLength;
  }
  currentHash = 0x1505;
  if ( len )
  {
    do
      currentHash = str[idx++] + 65599 * currentHash;
    while ( idx < len );
  }
  idx = currentHash;
  return idx | 0x80000000;
}
```

```
import itertools

def mphash(v2):
    v7 = 5381;
    v6 = 0
    while v6 < len(v2):
        v7 = ((65599 * v7) + ord(v2[v6])) & 0xFFFFFFFF;
        v6+=1;

    v7 |= 0x80000000;
    return v7

lenhash = mphash("length") #0x8c7bcb6b

print lenhash, hex(lenhash)

chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

for i in range(1,10):
    for x in itertools.product(chars, repeat=i):
        if mphash("".join(x)) == lenhash:
            print "".join(x)
```

Brute Forcing a Hash Collision

```
unsigned int __fastcall propHashStr(const char *str, unsigned int len)
{
  unsigned int idx; // esi@3
  int currentHash; // eax@4
  unsigned int pcchLength; // [sp+8h] [bp-4h]@8

  if ( !len )
    len = strlen(str);
  idx = 0;
  if ( !len )
  {
    pcchLength = 0;
    if ( StringCchLengthA(str, 0x7FFFFFFFu, &pcchLength) < 0 )
      return idx | 0x80000000;
    len = pcchLength;
  }
  currentHash = 0x1505;
  if ( len )
  {
    do
      currentHash = str[idx++] + 65599 * currentHash;
    while ( idx < len );
  }
  idx = currentHash;
  return idx | 0x80000000;
}
```

```
import itertools

def mphash(v2):
    v7 = 5381;
    v6 = 0
    while v6 < len(v2):
        v7 = ((65599 * v7) + ord(v2[v6])) & 0xFFFFFFFF;
        v6+=1;

    v7 |= 0x80000000;
    return v7

lenhash = mphash("length") #0x8c7bcb6b

print lenhash, hex(lenhash)

chars = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"

for i in range(1,10):
    for x in itertools.product(chars, repeat=i):
        if mphash("".join(x)) == lenhash:
            print "".join(x)
```

hash(“length”) == hash(“fyW093”)

JsArrayObject::putUpdateLengthThrows

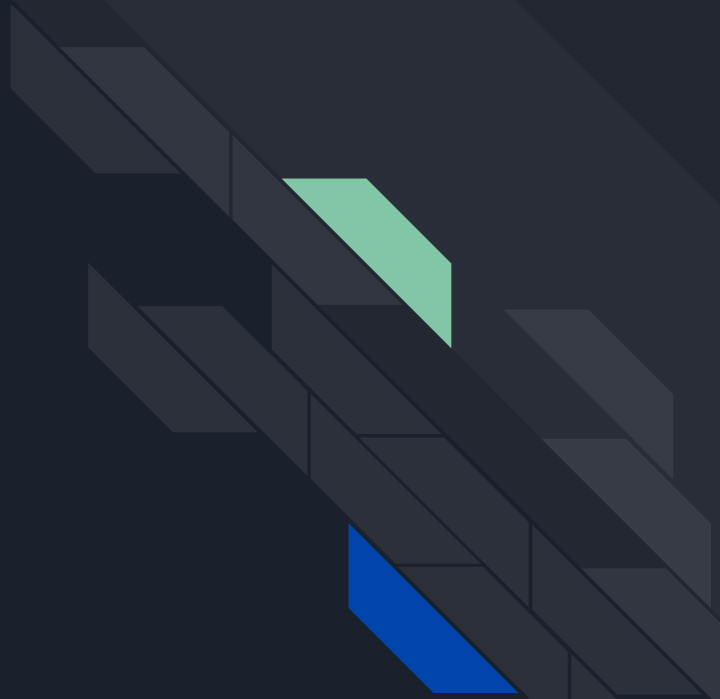
- Fires on any update of Array object properties, checks for update of hash("`length`")

- Manages arrays when "`length`" property is changed
  - Erases elements from the array if indexed above the new length

```
char __thiscall JsArrayObject::putUpdateLengthThrows(JsArrayObject *this, JsRuntimeState *jsState, unsigned int propHash, unsigned int value, bool propIsNum, unsigned int propNumber)
{
  JsArrayObject *v6; // edi@1
  unsigned int v7; // eax@1
  unsigned int v8; // eax@4
  unsigned int v10; // ebx@13
  unsigned int **v11; // edi@13
  unsigned int v12; // esi@13
  std::_Tree<std::_Tmap_traits<unsigned int,std::basic_string<char,std::char_traits<char>,std::allocator<char> > const ,std::less<unsigned int>,std::allocator<std::pair<unsigned int co
  unsigned int v14; // esi@15
  unsigned int lengthValue; // [sp+10h] [bp-10h]@3
  unsigned int propNum; // [sp+14h] [bp-Ch]@11
  unsigned int length; // [sp+18h] [bp-8h]@1
  std::_Tree_iterator<std::_Tree_val<std::_Tree_simple_types<std::pair<unsigned int const ,JsObject::Property> > > > result; // [sp+1Ch] [bp-4h]@24

  v6 = this;
  length = this;
  v7 = this->m_lengthPropHash;
  if ( propHash == v7 )
  {
    propNum = 0;
    if ( JsRuntimeState::toUint32Throws(jsState, value, &propNum) )
    {
      if ( jsState->m_complType != 4 )
      {
        v10 = propNum;
        v11 = &v6->m_properties;
        v12 = **v11;
        lengthValue = **v11;
        while ( v12 != *v11 )
        {
          propNum = 0;
          if ( JsObject::propIsNumeric(*(v12 + 16), &propNum) && propNum >= v10 )
          {
            std::_Tree_unchecked_const_iterator<std::_Tree_val<std::_Tree_simple_types<char const *>>,std::_Iterator_base0>::operator++(&lengthValue);
            std::_Tree<std::_Tmap_traits<unsigned int,JsObject::Property,std::less<unsigned int>,std::allocator<std::pair<unsigned int const,JsObject::Property>>,0>>::erase(
              v11,
              &result,
              v12);
          }
          else
          {
            std::_Tree_unchecked_const_iterator<std::_Tree_val<std::_Tree_simple_types<char const *>>,std::_Iterator_base0>::operator++(&lengthValue);
          }
          v12 = lengthValue;
        }
        v13 = (length + 12);
        v14 = **(length + 12);
        lengthValue = **(length + 12);
        while ( v14 != v13->_Myhead )
        {
          propNum = 0;
          if ( JsObject::propIsNumeric((v14 + 20), &propNum) && propNum >= v10 )
          {
            std::_Tree_unchecked_const_iterator<std::_Tree_val<std::_Tree_simple_types<std::pair<unsigned int const,ThreadManager::Impl::ThreadInfo>>>,std::_Iterator_base0>::operator++
```

Demo 3

```
(function() {
  var x = [1, 2, 3];
  print(x.length);  // 3
  print(x);         // "1,2,3"

  x.fyW093 = 5;
  print(x.fyW093);  // 5
  print(x.length);  // 5
  print(x);         // "1,2,3,,"

  x.fyW093 = 1;
  print(x);         // "1"
  print(x.length)   // 1
})()
```



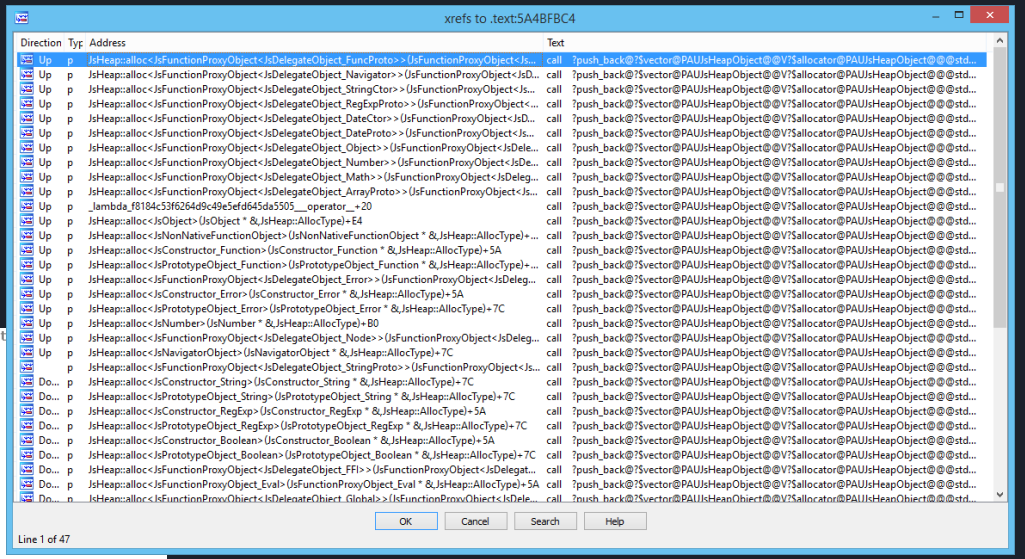
Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
    a. JS Language
    b. Types
    c. Memory Management
    d. Fingerprinting
4. Vulnerability Discussion
5. Conclusion

Memory Management

- Memory safe types - `std::vector` and `std::map`
- Homogeneous `Array` sizes (no typed arrays)
- No optimizations for object property get / set, fallback on `std::` library operations
- Raw arrays used for `JSBufString` objects, but few chances for controlled allocation sizes

Heap Management




```
char __thiscall JsHeap::alloc<JsFunctionProxyObject<JsDelegateObject_StringProto>>(JsHeap *this, JsFunct
{
  JsHeap *v3; // edi@1
  unsigned int v4; // eax@2
  JsFunctionProxyObject<JsDelegateObject_StringProto> *v5; // eax@5
  JsFunctionProxyObject<JsDelegateObject_StringProto> *v6; // ebx@5
  JsFunctionProxyObject<JsDelegateObject_StringProto> *v7; // eax@6
  JsFunctionProxyObject<JsDelegateObject_StringProto> *v9; // eax@11
  JsHeapObject *v10; // eax@12
  JsFunctionProxyObject<JsDelegateObject_StringProto> *v11; // ecx@12
  JsFunctionProxyObject<JsDelegateObject_StringProto> *v12; // esi@14
  std::default_delete<BlockDecryptor> *v13; // ecx@15
  BlockDecryptor *v14; // edi@17
  std::default_delete<BlockDecryptor> *v15; // ecx@18
  void *(__thiscall *v16)(JsHeapObject *, unsigned int); // edi@21
  unsigned int offBenchPtr; // [sp+Ch] [bp-24h]@15
  std::unique_ptr<JsFunctionProxyObject<JsDelegateObject_StringProto>,std::default_delete<JsFunctionProxyObject<JsDelegateO
  JsHeap::alloc::__l10::<lambda_2cc29d6b834e0eec69ceed780bf269eb> storePtrInDynamicHeap; // [sp+18h] [bp-18h]@7
  std::unique_ptr<JsHeapObject,std::default_delete<JsHeapObject> > v20; // [sp+1Ch] [bp-14h]@12
  int v21; // [sp+2Ch] [bp-4h]@5

  v3 = this;
  if ( this->m_haveHitLimit || (v4 = this->m_allocLimit, this->m_allocSize > v4) || v4 - this->m_allocSize < 0x58 )
  {
    this->m_haveHitLimit = 1;
  }
  else
  {
    if ( allocType == Permanent )
    {
      v5 = operator new(0x58u);
      newPtr._Myptr = v5;
      v6 = 0;
      v21 = 0;
      if ( v5 )
      {
        JsFunctionProxyObject<JsDelegateObject_StringProto>::JsFunctionProxyObject<JsDelegateObject_StringProto>(v5);
        v6 = v7;
      }
      newPtr._Myptr = v6;
      v21 = 1;
      std::vector<JsHeapObject *,std::allocator<JsHeapObject *>>::push_back(&v3->m_staticHeap, &storePtrInDynamicHeap);
      v3->m_allocSize += 88;
```

Allocations stored in `std::vector<JsHeapObject*>`

Garbage Collection

```
char __thiscall JsTree::run(JsTree *this, JsRuntimeState *jsState, bool blockGC)
```

Garbage Collection

```
char __thiscall JsTree::run(JsTree *this, JsRuntimeState *jsState, bool blockGC)
```

Garbage Collection

```
char __thiscall JsTree::run(JsTree *this, JsRuntimeState *jsState, bool blockGC)
```

Mark-and-Sweep Garbage Collection

```
if ( !blockGC )
{
  rootSeta = &jsState->m_callStack->m_workingStack.m_values;
  if ( JsHeap::garbageCollectStart(&jsState->m_heap) )
  {
    JsHeap::garbageCollectMark(&jsState->m_heap, rootSeta);
    jsState->m_heap.m_markedSize += JsHeap::gcMarkChildValue(&jsState->m_heap, jsState->m_complValue);
    jsState->m_heap.m_markedSize += JsHeap::gcMarkChildValue(&jsState->m_heap, jsState->m_conversionValue);
    jsState->m_heap.m_markedSize += JsHeap::gcMarkChildObject(&jsState->m_heap, &jsState->m_globalObj->vfptr);
    for ( i = jsState->m_exeCtxStack._Myfirst; ; ++i )
    {
      v26 = &jsState->m_heap;
      if ( i == jsState->m_exeCtxStack._Mylast )
        break;
      jsState->m_heap.m_markedSize += JsHeap::gcMarkChildObject(v26, &i->m_varObj->vfptr);
      jsState->m_heap.m_markedSize += JsHeap::gcMarkChildObject(&jsState->m_heap, &i->m_this->vfptr);
    }
    v8 = JsHeap::garbageCollectSweep(v26);
    goto LABEL_7;
  }
}
```

Garbage Collection

```
char __thiscall JsTree::run(JsTree *this, JsRuntimeState *jsState, bool blockGC)
```

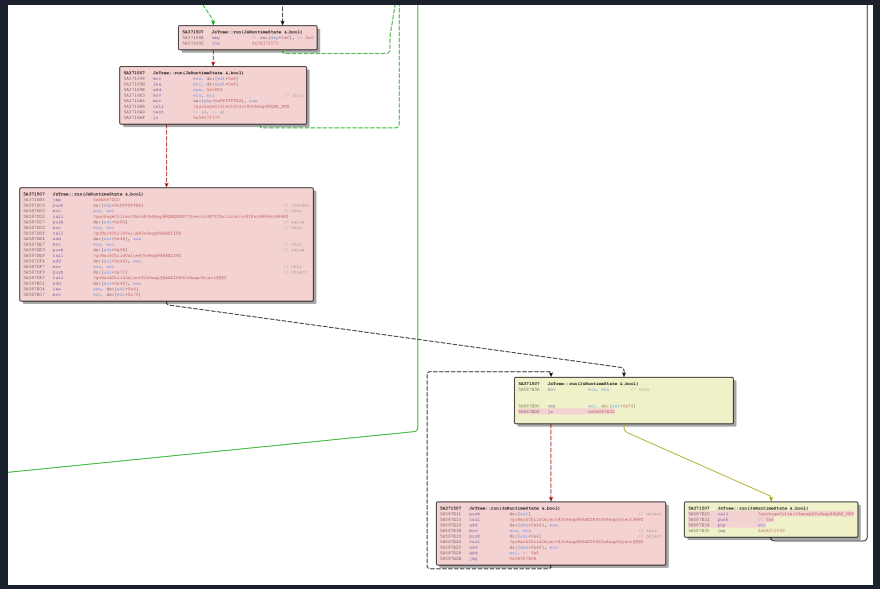
Mark-and-Sweep Garbage Collection

```
if ( !blockGC )
{
  rootSeta = &jsState->m_callStack->m_workingStack.m_values;
  if ( JsHeap::garbageCollectStart(&jsState->m_heap) )
  {
    JsHeap::garbageCollectMark(&jsState->m_heap, rootSeta);
    jsState->m_heap.m_markedSize += JsHeap::gcMarkChildValue(&jsState->m_heap, jsState->m_complValue);
    jsState->m_heap.m_markedSize += JsHeap::gcMarkChildValue(&jsState->m_heap, jsState->m_conversionValue);
    jsState->m_heap.m_markedSize += JsHeap::gcMarkChildObject(&jsState->m_heap, &jsState->m_globalObj->vfptr);
    for ( i = jsState->m_exeCtxStack._Myfirst; ; ++i )
    {
      v26 = &jsState->m_heap;
      if ( i == jsState->m_exeCtxStack._Mylast )
        break;
      jsState->m_heap.m_markedSize += JsHeap::gcMarkChildObject(v26, &i->m_varObj->vfptr);
      jsState->m_heap.m_markedSize += JsHeap::gcMarkChildObject(&jsState->m_heap, &i->m_this->vfptr);
    }
    v8 = JsHeap::garbageCollectSweep(v26);
    goto LABEL_7;
  }
}
```

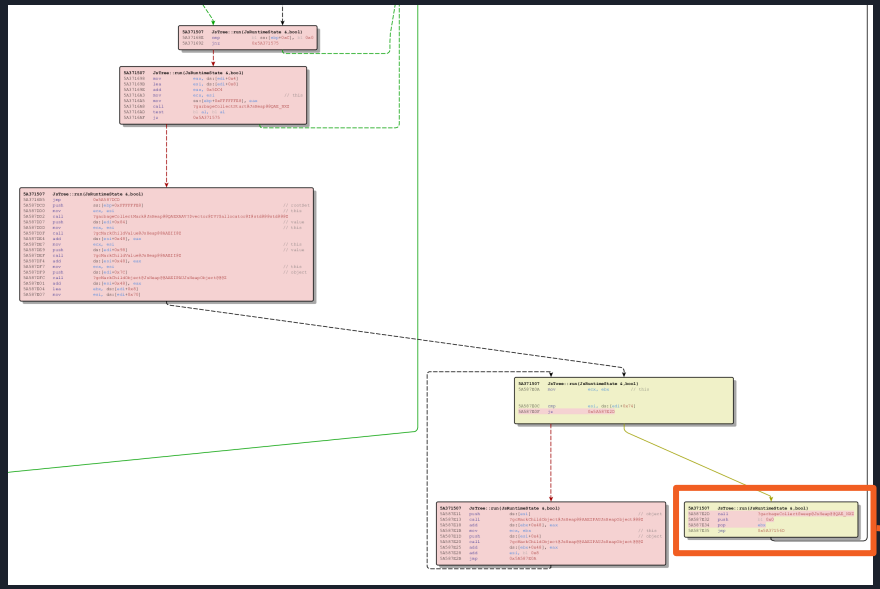
Project Zero beat me to it - 2 Days Later...

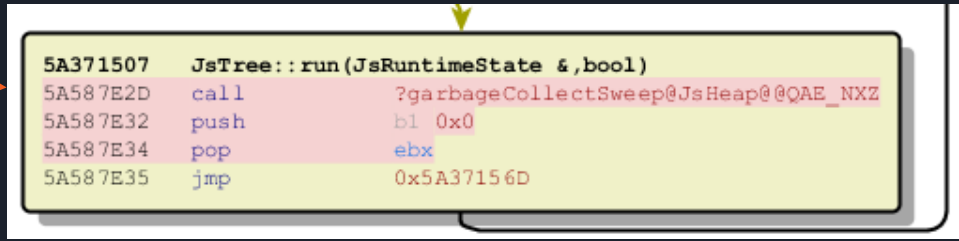
ianbeer@google.com	Windows MsMpEng remotely exploitable UaF due to design issue in GC engine CCProjectZeroMembers
lokihardt@google.com	MsMpEng: UAF via saved callers CCProjectZeroMembers

BinDiffing

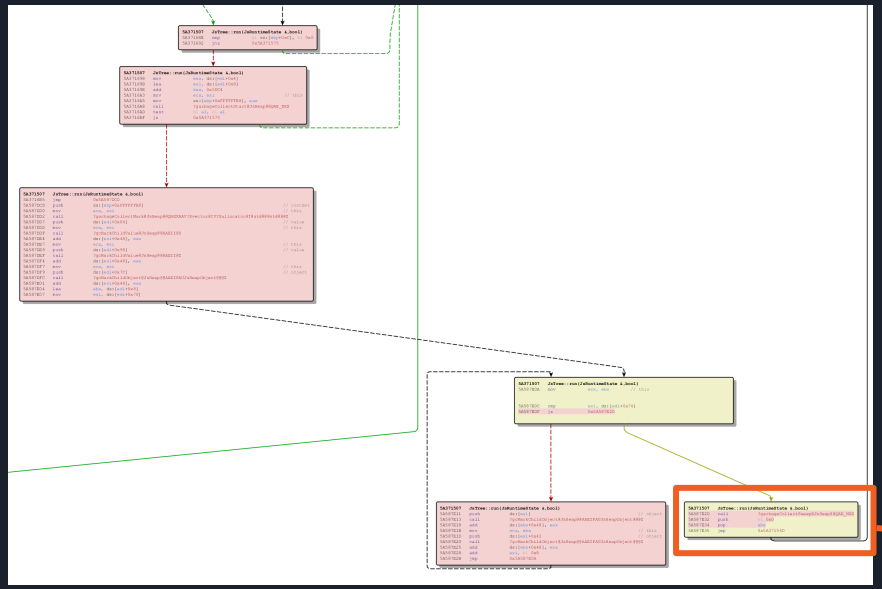


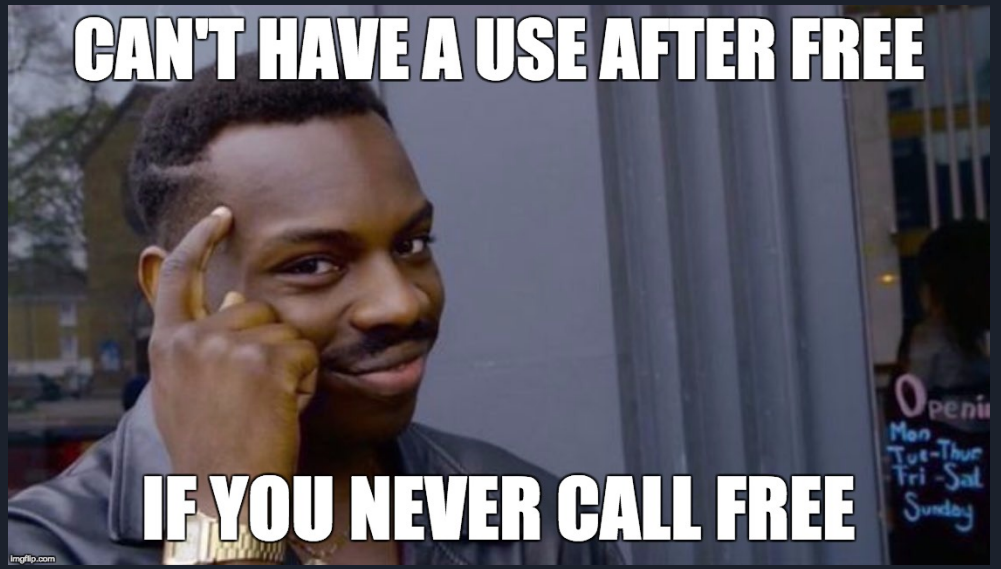
BinDiffing




5A371507 JsTree::run(JsRuntimeState &,bool)
5A587E2D call ?garbageCollectSweep@JsHeap@@QAE_NXZ
5A587E32 push bl 0x0
5A587E34 pop ebx
5A587E35 jmp 0x5A37156D

BinDiffing






```
5A371507    JsTree::run(JsRuntimeState &,bool)
5A587E2D    call    ?garbageCollectSweep@JsHeap@@QAE_NXZ
5A587E32    push    bl 0x0
5A587E34    pop     ebx
5A587E35    jmp     0x5A37156D
```

Heap Teardown After Emulation

`JsRuntimeState::~JsRuntimeState` destructor tears down the JS heap with `JsHeap::~JsHeap`

```
void __thiscall JsHeap::~JsHeap(JsHeap *this)
{
  char *v2; // edi@1
  void *const *staticHeapFirst; // ebx@1
  std::vector<JsHeapObject *,std::allocator<JsHeapObject *> > *v4;

  this->vfptr = &JsHeap::`vftable';
  v2 = &this->m_dynamicHeap;
  lambda_ae1ca4fc1b395ad924f2a00ccf21ba68_::operator()(this);
  staticHeapFirst = &this->m_staticHeap._Myfirst;
  lambda_ae1ca4fc1b395ad924f2a00ccf21ba68_::operator()(v4);
  JsBench::freeAtIdx(&this->m_shortLifeBench, 1u);
  JsBench::freeAtIdx(&this->m_shortLifeBench, 0);
  std::vector<forwarded_export_resolution_data_t,std::allocator<fo
  if ( this->m_staticHeap._Myfirst )
  {
    free(*staticHeapFirst);
    *staticHeapFirst = 0;
    this->m_staticHeap._Mylast = 0;
    this->m_staticHeap._Myend = 0;
  }
```

```
void __stdcall lambda_ae1ca4fc1b395ad924f2a00ccf21ba68_::operator(
{
  JsStringObject **i; // esi@1
  JsStringObject *v3; // edi@3
  JsStringObject *(__thiscall *v4)(JsStringObject *, unsigned int)
  JsStringObject *(__thiscall *v5)(JsStringObject *, unsigned int)

  for ( i = heapa->_Myfirst; i != heapa->_Mylast; ++i )
  {
    v3 = *i;
    if ( *i )
    {
      v4 = v3->vfptr->__vecDelDtor;
      v5 = v4;
      if ( v4 == JsObject::`vector deleting destructor' )
      {
        JsObject::`vector deleting destructor'(v3, 1u);
      }
      else
      {
        __guard_check_icall_fptr(v4);
        v5(v3, 1u);
      }
    }
  }
}
```

JsBufString char Array Backing

```
numBytes = JsString::numBytes(encStrValue);
*decStrValue = 18;
if ( !numBytes )
  return 1;
decStrPtr._Myptr = operator new(numBytes);
```

```
JsString::initByReceipt(jsState, &decStrPtr, numBytes_1, decStrValue);
```

```
char __usercall JsString::initByReceipt@<al>(JsRuntimeState *jsState@<ecx>, std::uniqu
{
  std::unique_ptr<char [0],std::default_delete<char [0]> > *v4; // esi
  JsBufString *v6; // edi
  JsBufString *newStr; // [esp+8h] [ebp-8h]

  v4 = bufPtr;
  if ( numBytes < 3 )
  {
    *str = initImmStr(bufPtr->_Myptr, numBytes);
    return 1;
  }
  newStr = 0;
  if ( JsHeap::alloc<JsBufString>(&jsState->m_heap, &newStr, jsState) )
  {
    v6 = newStr;
    if ( (newStr->vfptr[3].__vecDelDtor)(newStr, v4, numBytes) )//  JsBufString::recv
    {
      *str = v6;
      return 1;
    }
  }
  return 0;
}
```

- `char` arrays used for storage backing `JsBufString` - not `std::vector`
- Arrays allocated then assigned to `JsBufString` with `initByReceipt`
- Only allocated in a few places, done safely

Bug - `escape()` is broken

Bug - `escape()` is broken

```
escape("%")
```

Bug - `escape()` is broken

`escape("%")`

```
numBytesOrig = JsString::numBytes(str);
allocatedBufLenX3 = 3 * numBytesOrig;
escStrBuf._Myptr = operator new(3 * numBytesOrig);
```

Buffer size = 3 * numBytes

3 = 3*1

Bug - `escape()` is broken

`escape("%")`

```
numBytesOrig = JsString::numBytes(str);
allocatedBufLenX3 = 3 * numBytesOrig;
escStrBuf._Myptr = operator new(3 * numBytesOrig);
```

Buffer size = 3 * numBytes
3 = 3*1

```
if ( numBytesOrig )
{
  idxNext1Chr = 1;
  idxNext3Chr = 3;
  idxNext5Chr = 5;
  while ( v9 % 100 || JsRuntimeState::exeTick(jsState, 1u) )
  {
    currentChar = 0;
    utf8Decode(str, &escStrByteOffs, &currentChar);
```

"%" should escape to 3 characters

```
> escape("%")
<· "%25"
```

Bug - `escape()` is broken

`escape("%")`

```
numBytesOrig = JsString::numBytes(str);
allocatedBufLenX3 = 3 * numBytesOrig;
escStrBuf._Myptr = operator new(3 * numBytesOrig);
```

Buffer size = 3 * numBytes
3 = 3*1

```
if ( numBytesOrig )
{
  idxNext1Chr = 1;
  idxNext3Chr = 3;
  idxNext5Chr = 5;
  while ( v9 % 100 || JsRuntimeState::exeTick(jsState, 1u) )
  {
    currentChar = 0;
    utf8Decode(str, &escStrByteOffs, &currentChar);
```

"%" should escape to 3 characters

```
> escape("%")
< "%25"
```

```
else
{
  if ( idxNext3Chr >= allocatedBufLenX3 )
    goto END_FREE;
  widthCurrentChar = 3;
  escStrBuf._Myptr[numBytesEsc] = '%';
  escStrBuf._Myptr[numBytesEsc + 1] = hexChrs[v10 >> 4];
  escStrBuf._Myptr[numBytesEsc + 2] = hexChrs[v10 & 0xF];
}
```

3 >= 3, function returns early to avoid a buffer overflow

Bug - `escape()` is broken

`escape("%")`

```
numBytesOrig = JsString::numBytes(str);
allocatedBufLenX3 = 3 * numBytesOrig;
escStrBuf._Myptr = operator new(3 * numBytesOrig);
```

Buffer size = 3 * numBytes
3 = 3*1

```
if ( numBytesOrig )
{
  idxNext1Chr = 1;
  idxNext3Chr = 3;
  idxNext5Chr = 5;
  while ( v9 % 100 || JsRuntimeState::exeTick(jsState, 1u) )
  {
    currentChar = 0;
    utf8Decode(str, &escStrByteOffs, &currentChar);
```

"%" should escape to 3 characters

```
> escape("%")
< "%25"
```

```
else
{
  if ( idxNext3Chr >= allocatedBufLenX3 )
    goto END_FREE;
  widthCurrentChar = 3;
  escStrBuf._Myptr[numBytesEsc] = '%';
  escStrBuf._Myptr[numBytesEsc + 1] = hexChrs[v10 >> 4];
  escStrBuf._Myptr[numBytesEsc + 2] = hexChrs[v10 & 0xF];
}
```

3 >= 3, function returns early to avoid a buffer overflow

`escape()`ing any single character element fails - similar problems in other escape related functions

Demo 4

```
(function() {
  print(escape("%a"));
  print(escape("%"));
})()
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
    a. JS Language
    b. Types
    c. Memory Management
    d. Fingerprinting
4. Vulnerability Discussion
5. Conclusion

Fingerprinting

Unique traits that identify the Defender JS engine

```
function (){
  if (DetectDefender())
  {
    return;
  }
  else
  {
    MaliciousCode();
  }
}
```

Fingerprinting

Unique traits that identify the Defender JS engine

```
function (){
  if (DetectDefender())
  {
    return;
  }
  else
  {
    MaliciousCode();
  }
}
```

```
function (){
  if (DetectDefender())
  {
    ExploitDefender();
  }
  else
  {
    ...
  }
}
```

Hardcoded Values

```
mpscript>(function(){
                    print(navigator.userAgent);
          })()
print(): Mozilla/5.0 (compatible; MSIE 6.0; Windows NT 5.1)
```

```
> log(document.referrer)
http://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web
```

```
mpscript> (function (){
             var x = new Date();
             print(x);
           })()
triggerEvent(): date_tostring
print(): Mon 19 Mar 2012 01:29:10 UTC
```

Instantiating Non-Objects

```
mpscript> (function (){
            var x = new isFinite();
            print(x);
          })()
JavaScriptLog(): [object Object]
```

```
mpscript> (function (){
            var x = new eval();
            print(x);
          })()
JavaScriptLog(): [object Object]
```

```
(function (){ var x = new print(); print(x) })()
JavaScriptLog(): [object Object]

(function (){ var x = new isNaN (); print(x) })()
JavaScriptLog(): [object Object]
```

```
if ( isConstructor )
  return JsRuntimeState::throwNativeError(jsState, BltinTypeError,
```

```
> var x = new isFinite()
▶Uncaught TypeError: isFinite is not a constructor
    at <anonymous>:1:9
```

Typos

```
char __stdcall JsDelegateObject_Global::collectGarbage(JsRuntimeState *jsState, std::vector<unsigned int,std::all
{
  char result; // al@2

  if ( isConstructor )
  {
    result = JsRuntimeState::throwNativeError(jsState, BltinTypeError, "collectGarbage() is not a constructor");
  }
  else
  {
    jsState->m_complValue = 6;
    jsState->m_complType = 3;
    jsState->m_complTarget = 0;
    result = JsRuntimeState::triggerEvent(jsState, 0, "collectgarbage", 0, 0, jsState, jsState);
  }
  return result;
}
```

```
mpscript> (function (){
            try{
              var x = new CollectGarbage();
            }
            catch(e){
              print(e);
            }
          })()

JavaScriptLog():
TypeError: collectGarbage() is not a constructor
```

(`collectGarbage()` throws an exception, undefined)

Elements Are Functions

```
mpscript> (function x(){
              var x = document.createElement("p");
              print(typeof(x));
          })()
print(): function
```

```
> (function x(){var x = document.createElement("p"); console.log(typeof(x)) })()
  object
```

Multiple Error Handlers

```
(function x(){
     var x = new Object();
     x.foo = (new String()).valueOf;
     x.foo()
})()
triggerEvent(): err_typeerror
triggerEvent(): error_tostring
Log(): uncaught exception: TypeError: String.prototype.toString and String.prototype.valueOf
must be called only for Strings


(function x(){
     var x = new navigator.javaEnabled();
})()
JavaScriptLog(): TypeError: Navigator.javaEnabled() and Navigator.taintEnabled() are not a
constructors
```

BUG, should never happen

```
mpscript> (function x(){
            var num = new Number(1);
            var node = document.createTextNode("node");
            var elem = document.createElement("element");
            num.appendChild = elem.appendChild;
            num.appendChild(node);
          })()

triggerEvent(): err_typeerror
triggerEvent(): error_tostring
Log(): uncaught exception: TypeError: node.insertBefore()
                'this' object must be DOM Object (BUG, should never happen)
```

Also works for `node.appendChild()`

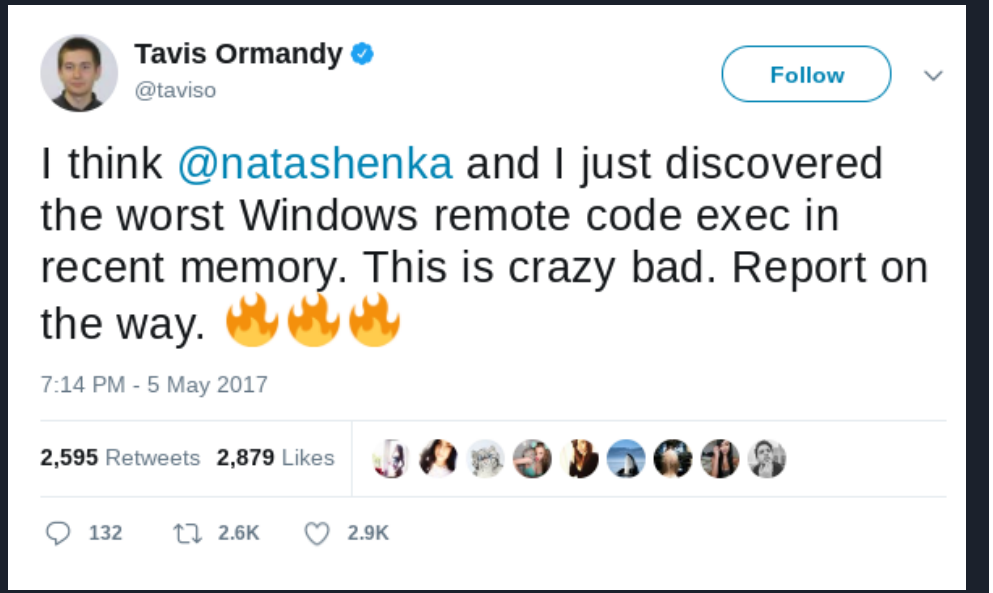
Other

```
(function(){
    print("A") print("B") // no semicolon seperator
})()
print(): A
print(): B

(function (){
    var myFunction = function namedFunction(){};
    print(myFunction.name);
})()
print(): undefined

(function x(){
    var x = new Array();
    x.getTimezoneOffset = (new Date()).getTimezoneOffset;
    print(x.getTimezoneOffset())
})()
JavaScriptLog(): 0
```

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Discussion
5. Conclusion

May 2017




```
(new Error()).toString.call({message: 0x41414141 >> 1})
```

Understanding P0's Vulnerability

```
char __stdcall JsDelegateObject_Error::toString(JsRuntimeState *jsState, std::vector<unsigned int,std::allocator<unsig
{
  int v3; // edx@2
  JsObject *JsRunTimeStateThis; // ebx@3
  JsHeapObjectVtbl *v5; // eax@4
  unsigned int (__thiscall *v6)(JsHeapObject *, std::vector<unsigned int,std::allocator<unsigned int> > *); // esi@4
  int v7; // ST00_4@4
  char success; // al@5
  char shortStrEventSuccess; // al@7
  JsEvaluationMonitor::EventCode eventCode; // ST08_4@11 MAPDST
  unsigned int propHash; // [sp+0h] [bp-10h]@2
  unsigned int messageObj; // [sp+4h] [bp-Ch]@2 MAPDST
  JsObject *thisObj; // [sp+8h] [bp-8h]@2

  if ( isConstructor )
  {
    shortStrEventSuccess = JsRuntimeState::throwNativeError(
                             jsState,
                             BltinTypeError,
                             "Error.prototype.toString() is not a constructor");
  }
  else
  {
    thisObj = 0;
    propHash = JsObject::genPropHash("message", 0);
    messageObj = 6;
    if ( JsRuntimeState::getThisPtr(jsState, &thisObj)
      && (JsRunTimeStateThis = thisObj) != 0
      && (v5 = thisObj->vfptr,
          thisObj = 0,
          v6 = v5[2].gcMark,
          v7 = v3,
          __guard_check_icall_fptr(v5[2].gcMark),
          (v6)(JsRunTimeStateThis, v7, &thisObj)) )
    {
      success = JsObject::get(JsRunTimeStateThis, jsState, propHash, &messageObj);// propHash=hash("message")
    }
    else
    {
      success = JsString::initByRef(jsState, "undefined", 9u, &messageObj);
    }
    if ( success )
    {
      jsState->m_complTarget = 0;
      jsState->m_complValue = messageObj;
      jsState->m_complType = 3;
      shortStrEventSuccess = JsRuntimeState::triggerShortStrEvent(jsState, eventCode, "error_tostring", messageObj);
    }
    else
    {
      shortStrEventSuccess = 0;
    }
  }
  return shortStrEventSuccess;
}
```

`JsDelegateObject_Error::toString`

Initial validation

Understanding P0's Vulnerability

```
char __stdcall JsDelegateObject_Error::toString(JsRuntimeState *jsState, std::vector<unsigned int,std::allocator<unsig
{
  int v3; // edx@2
  JsObject *JsRunTimeStateThis; // ebx@3
  JsHeapObjectVtbl *v5; // eax@4
  unsigned int (__thiscall *v6)(JsHeapObject *, std::vector<unsigned int,std::allocator<unsigned int> > *); // esi@4
  int v7; // ST00_4@4
  char success; // al@5
  char shortStrEventSuccess; // al@7
  JsEvaluationMonitor::EventCode eventCode; // ST08_4@11 MAPDST
  unsigned int propHash; // [sp+0h] [bp-10h]@2
  unsigned int messageObj; // [sp+4h] [bp-Ch]@2 MAPDST
  JsObject *thisObj; // [sp+8h] [bp-8h]@2

  if ( isConstructor )
  {
    shortStrEventSuccess = JsRuntimeState::throwNativeError(
                             jsState,
                             BltinTypeError,
                             "Error.prototype.toString() is not a constructor");
  }
  else
  {
    thisObj = 0;
    propHash = JsObject::genPropHash("message", 0);
    messageObj = 6;
    if ( JsRuntimeState::getThisPtr(jsState, &thisObj)
      && (JsRunTimeStateThis = thisObj) != 0
      && (v5 = thisObj->vfptr,
          thisObj = 0,
          v6 = v5[2].gcMark,
          v7 = v3,
          __guard_check_icall_fptr(v5[2].gcMark),
          (v6)(JsRunTimeStateThis, v7, &thisObj)) )
    {
      success = JsObject::get(JsRunTimeStateThis, jsState, propHash, &messageObj);// propHash=hash("message")
    }
    else
    {
      success = JsString::initByRef(jsState, "undefined", 9u, &messageObj);
    }
    if ( success )
    {
      jsState->m_complTarget = 0;
      jsState->m_complValue = messageObj;
      jsState->m_complType = 3;
      shortStrEventSuccess = JsRuntimeState::triggerShortStrEvent(jsState, eventCode, "error_tostring", messageObj);
    }
    else
    {
      shortStrEventSuccess = 0;
    }
  }
  return shortStrEventSuccess;
}
```

`JsDelegateObject_Error::toString`

Initial validation

Get `this.message`

Understanding P0's Vulnerability

```
char __stdcall JsDelegateObject_Error::toString(JsRuntimeState *jsState, std::vector<unsigned int,std::allocator<unsig
{
  int v3; // edx@2
  JsObject *JsRunTimeStateThis; // ebx@3
  JsHeapObjectVtbl *v5; // eax@4
  unsigned int (__thiscall *v6)(JsHeapObject *, std::vector<unsigned int,std::allocator<unsigned int> > *); // esi@4
  int v7; // ST00_4@4
  char success; // al@5
  char shortStrEventSuccess; // al@7
  JsEvaluationMonitor::EventCode eventCode; // ST08_4@11 MAPDST
  unsigned int propHash; // [sp+0h] [bp-10h]@2
  unsigned int messageObj; // [sp+4h] [bp-Ch]@2 MAPDST
  JsObject *thisObj; // [sp+8h] [bp-8h]@2

  if ( isConstructor )
  {
    shortStrEventSuccess = JsRuntimeState::throwNativeError(
                             jsState,
                             BltinTypeError,
                             "Error.prototype.toString() is not a constructor");
  }
  else
  {
    thisObj = 0;
    propHash = JsObject::genPropHash("message", 0);
    messageObj = 6;
    if ( JsRuntimeState::getThisPtr(jsState, &thisObj)
      && (JsRunTimeStateThis = thisObj) != 0
      && (v5 = thisObj->vfptr,
          thisObj = 0,
          v6 = v5[2].gcMark,
          v7 = v3,
          __guard_check_icall_fptr(v5[2].gcMark),
          (v6)(JsRunTimeStateThis, v7, &thisObj)) )
    {
      success = JsObject::get(JsRunTimeStateThis, jsState, propHash, &messageObj);// propHash=hash("message")
    }
    else
    {
      success = JsString::initByRef(jsState, "undefined", 9u, &messageObj);
    }
    if ( success )
    {
      jsState->m_complTarget = 0;
      jsState->m_complValue = messageObj;
      jsState->m_complType = 3;
      shortStrEventSuccess = JsRuntimeState::triggerShortStrEvent(jsState, eventCode, "error_tostring", messageObj);
    }
    else
    {
      shortStrEventSuccess = 0;
    }
  }
  return shortStrEventSuccess;
}
```

`JsDelegateObject_Error::toString`

- Initial validation
- Get `this.message`
- Pass `this.message` to function expecting `JsString`

Understanding P0's Vulnerability

```
bool __thiscall JsRuntimeState::triggerShortStrEvent(JsRuntimeState *this
{
  JsRuntimeState *v4; // edi@1
  unsigned int v5; // ebx@2
  unsigned int v6; // esi@4
  int v7; // eax@7
  JsEvaluationMonitor *v8; // edi@8
  int v9; // ST08_4@8
  bool (__thiscall *v10)(JsEvaluationMonitor *, JsEvaluationMonitor::Even
  bool result; // al@8

  v4 = this;
  if ( this->m_monitor )
  {
    v5 = JsString::numBytes(strAttr0StrVal);
    if ( v5 > 80 )
      v5 = 80;
    v6 = 0;
    if ( !v5 )
      goto LABEL_14;
    do
    {
      v4->m_shortEventBuf[v6] = JsString::getByte(strAttr0StrVal, v6, 0);
      ++v6;
    }
    while ( v6 < v5 );
    if ( v5 )
      v7 = v4->m_shortEventBuf;
    else
LABEL_14:
      v7 = 0;
    v8 = v4->m_monitor;
    v9 = v7;
    v10 = v8->vfptr->analyse;
    __guard_check_icall_fptr(v8->vfptr->analyse);
    result = v10(v8, 0, subEvent, v9, v5, 0, 0);
  }
  else
  {
    result = 1;
  }
  return result;
}
```

`JsRuntimeState::triggerShortStrEvent` is an AV monitoring callback

`JsString::numBytes` type confusion

Treat unvalidated input as `JsString`

Understanding P0's Vulnerability

```
bool __thiscall JsRuntimeState::triggerShortStrEvent(JsRuntimeState *this
{
  JsRuntimeState *v4; // edi@1
  unsigned int v5; // ebx@2
  unsigned int v6; // esi@4
  int v7; // eax@7
  JsEvaluationMonitor *v8; // edi@8
  int v9; // ST08_4@8
  bool (__thiscall *v10)(JsEvaluationMonitor *, JsEvaluationMonitor::Even
  bool result; // al@8

  v4 = this;
  if ( this->m_monitor )
  {
    v5 = JsString::numBytes(strAttr0StrVal);
    if ( v5 > 80 )
      v5 = 80;
    v6 = 0;
    if ( !v5 )
      goto LABEL_14;
    do
    {
      v4->m_shortEventBuf[v6] = JsString::getByte(strAttr0StrVal, v6, 0);
      ++v6;
    }
    while ( v6 < v5 );
    if ( v5 )
      v7 = v4->m_shortEventBuf;
    else
LABEL_14:
      v7 = 0;
    v8 = v4->m_monitor;
    v9 = v7;
    v10 = v8->vfptr->analyse;
    __guard_check_icall_fptr(v8->vfptr->analyse);
    result = v10(v8, 0, subEvent, v9, v5, 0, 0);
  }
  else
  {
    result = 1;
  }
  return result;
}
```

`JsRuntimeState::triggerShortStrEvent` is an AV monitoring callback

`JsString::numBytes` type confusion

Treat unvalidated input as `JsString`

```
00 00f0ce50 6ea21c72 mpengine!JsString::numBytes+0x1f [h:\av\engine released\amcore\mpengine\m
01 00f0ce68 6ea10796 mpengine!JsRuntimeState::triggerShortStrEvent+0x24 [h:\av\engine released
02 00f0ce94 6ea106cc mpengine!JsDelegateObject_Error::toString+0xb8 [h:\av\engine released\amc
03 (Inline) -------- mpengine!JsDelegateObject_Error::delegate+0x11 [h:\av\engine released\amc
04 00f0ceac 6ea1a66d mpengine!JsFunctionProxyObject<JsDelegateObject_Error>::call+0x1c [h:\av\
05 00f0cee4 6ed55394 mpengine!preInvokeFunctionThrows+0xf9 [h:\av\engine released\amcore\mpeng
06 00f0cf44 6ec8f9f8 mpengine!JsDelegateObject_FuncProto::call+0x1c8 [h:\av\engine released\am
07 (Inline) -------- mpengine!JsDelegateObject_FuncProto::delegate+0x28960d [h:\av\engine rele
08 00f0cf5c 6ea1a66d mpengine!JsFunctionProxyObject<JsDelegateObject_FuncProto>::call+0x289618
09 00f0cf94 6ea1ae0d mpengine!preInvokeFunctionThrows+0xf9 [h:\av\engine released\amcore\mpeng
0a 00f0cff4 6ea2163b mpengine!JsCallExprTree::eval+0x1bd [h:\av\engine released\amcore\mpengin
0b 00f0d030 6ea10a87 mpengine!JsTree::run+0x134 [h:\av\engine released\amcore\mpengine\maveng\
0c 00f0d254 00df3c89 mpengine!JavaScriptInterpreter::eval+0x212 [h:\av\engine released\amcore\
0d 00f0f9c4 00df4a7e JsShell!wmain+0x369 [c:\users\alex\documents\visual studio 2017\projects\
0e 00f0f9d8 00df48e0 JsShell!invoke_main+0x1e [f:\dd\vctools\crt\vcstartup\src\startup\exe_com
0f 00f0fa30 00df477d JsShell!__scrt_common_main_seh+0x150 [f:\dd\vctools\crt\vcstartup\src\sta
10 00f0fa38 00df4a98 JsShell!__scrt_common_main+0xd [f:\dd\vctools\crt\vcstartup\src\startup\e
11 00f0fa40 775f7c04 JsShell!wmainCRTStartup+0x8 [f:\dd\vctools\crt\vcstartup\src\startup\exe
12 00f0fa54 7784b90f KERNEL32!BaseThreadInitThunk+0x24
13 00f0fa9c 7784b8da ntdll!__RtlUserThreadStart+0x2f
14 00f0faac 00000000 ntdll!_RtlUserThreadStart+0x1b
```

Patch & Discussion

```
if ( this->m_monitor )
{
  v5 = JsString::numBytes(strAttr0StrVal);
```

```
if ( *(this + 224) )
{
  if ( getValueType(a4) == 4 )
  {
    v5 = JsString::numBytes(v10, v12);
```

Patched by adding explicit type checking that message type is `String`

Patch & Discussion

xrefs to .text:5A10EEF8

Type	Address	Text	
o	JsDelegateObject_Error::toString(JsRuntimeState &,std::vector<uint,std::alloca...	mov	ecx, offset aMessage; "message"
o	newErrorObjectT<JsObject>(JsRuntimeState &,uint,JsObject *,JsRuntimeState:...	push	offset aMessage ; "message"

```
if ( this->m_monitor )
{
  v5 = JsString::numBytes(strAttr0StrVal);
```

"`message`" only used during initialization and `toString`

Patched by adding explicit type checking that message type is `String`

```
if ( *(this + 224) )
{
  if ( getValueType(a4) == 4 )
  {
    v5 = JsString::numBytes(v10, v12);
```

Attack Surface Reduction

Language:

- Complex ECMAScript features avoided
  - eg: `Array.prototype.sort`

Attack Surface Reduction

Language:

- Complex ECMAScript features avoided
  - eg: `Array.prototype.sort`

Data Structures:

- Few controlled allocations
  - Integer overflow checked
- One array implementation
- Capped array lengths
- `std::vector` backing JS arrays

Attack Surface Reduction

Language:

- Complex ECMAScript features avoided
  - eg: `Array.prototype.sort`

Data Structures:

- Few controlled allocations
  - Integer overflow checked
- One array implementation
- Capped array lengths
- `std::vector` backing JS arrays

Implementation:

- Callbacks into JS runtime avoided
- Single threaded
- Little DOM implementation
- Extensive type checking (other than P0's bug)
- No JIT
- No GC

Attack Surface Reduction

Language:

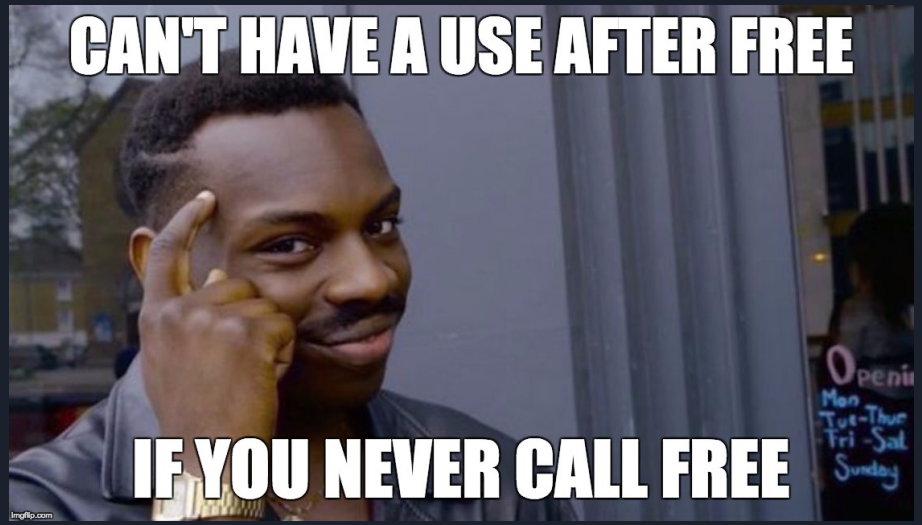
- Complex ECMAScript features avoided
  - eg: `Array.prototype.sort`

Data Structures:

- Few controlled allocations
  - Integer overflow checked
- One array implementation
- Capped array lengths
- `std::vector` backing JS arrays

Implementation:

- Callbacks into JS runtime avoided
- Single threaded
- Little DOM implementation
- Extensive type checking (other than P0's bug)
- No JIT
- No GC



Overall:

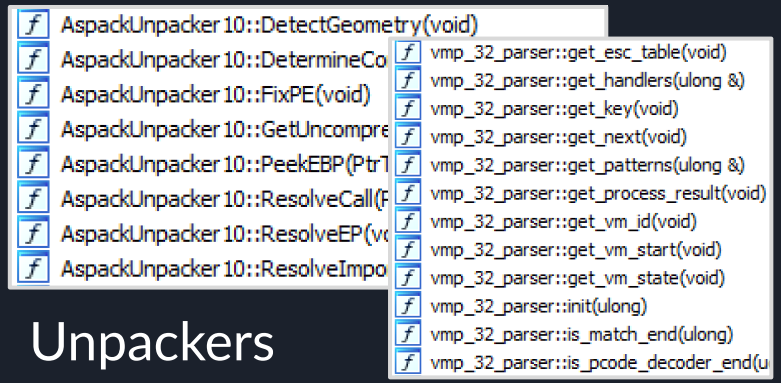
- Simplicity and ease of implementation
- Take advantage of being inside an AV
- Break the runtime in the interest of security
- Soon to be sandboxed...

Outline

1. Introduction
2. Tooling & Process
3. Reverse Engineering
4. Vulnerability Discussion
5. Conclusion

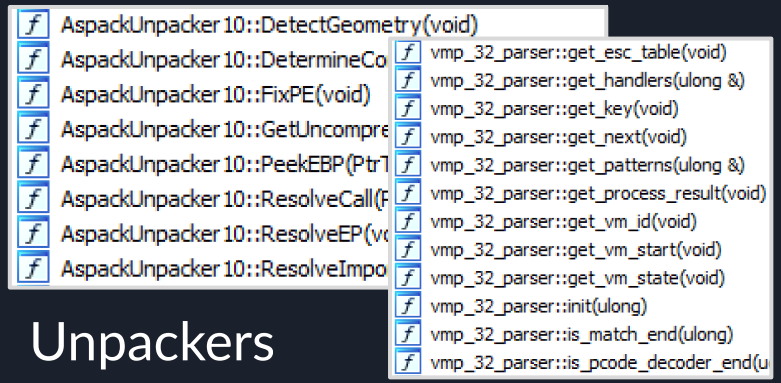
The Remaining 98%
43,000+ Functions

The Remaining 98%
43,000+ Functions

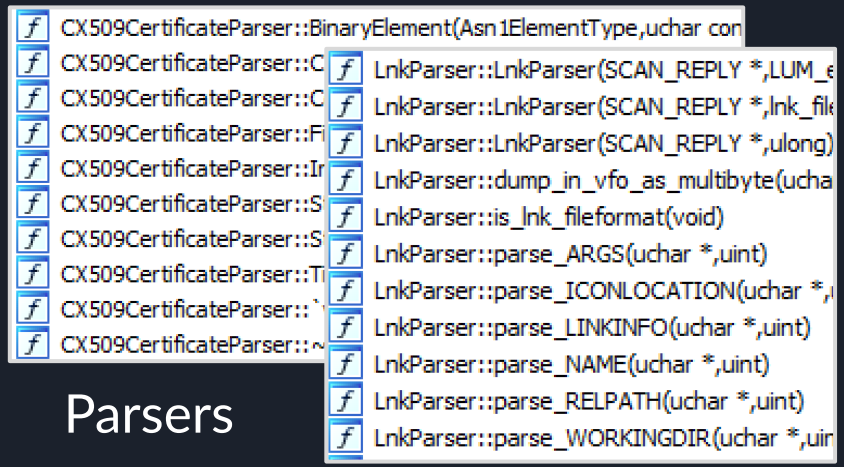


Unpackers

The Remaining 98%
43,000+ Functions



Unpackers



Parsers

The Remaining 98%
43,000+ Functions

AspackUnpacker10::DetectGeometry(void)
AspackUnpacker10::DetermineCo
AspackUnpacker10::FixPE(void)
AspackUnpacker10::GetUncompre
AspackUnpacker10::PeekEBP(Ptr
AspackUnpacker10::ResolveCall(
AspackUnpacker10::ResolveEP(v
AspackUnpacker10::ResolveImpo

vmp_32_parser::get_esc_table(void)
vmp_32_parser::get_handlers(ulong &)
vmp_32_parser::get_key(void)
vmp_32_parser::get_next(void)
vmp_32_parser::get_patterns(ulong &)
vmp_32_parser::get_process_result(void)
vmp_32_parser::get_vm_id(void)
vmp_32_parser::get_vm_start(void)
vmp_32_parser::get_vm_state(void)
vmp_32_parser::init(ulong)
vmp_32_parser::is_match_end(ulong)
vmp_32_parser::is_pcode_decoder_end(u

Unpackers

CX509CertificateParser::BinaryElement(Asn1ElementType,uchar con
CX509CertificateParser::C
CX509CertificateParser::C
CX509CertificateParser::Fi
CX509CertificateParser::Ir
CX509CertificateParser::S
CX509CertificateParser::S
CX509CertificateParser::Ti
CX509CertificateParser::`
CX509CertificateParser::~

LnkParser::LnkParser(SCAN_REPLY *,LUM_e
LnkParser::LnkParser(SCAN_REPLY *,lnk_file
LnkParser::LnkParser(SCAN_REPLY *,ulong)
LnkParser::dump_in_vfo_as_multibyte(ucha
LnkParser::is_lnk_fileformat(void)
LnkParser::parse_ARGS(uchar *,uint)
LnkParser::parse_ICONLOCATION(uchar *,
LnkParser::parse_LINKINFO(uchar *,uint)
LnkParser::parse_NAME(uchar *,uint)
LnkParser::parse_RELPATH(uchar *,uint)
LnkParser::parse_WORKINGDIR(uchar *,uin

Parsers

Windows Emulator

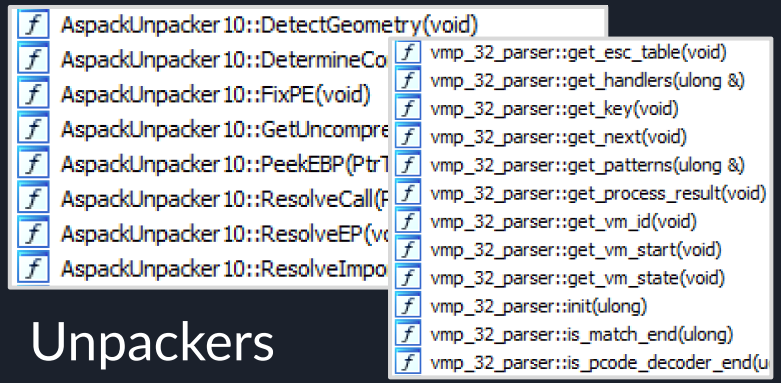
- x86, x64, & ARM
  - Lifted to IL for emulation
- WinAPI & NT Kernel emulation

ADVAPI32_DLL_RegCreateKeyExW(pe_vars_t *)
ADVAPI32_DLL_RegDeleteKeyW(pe_vars_t *)
ADVAPI32_DLL_RegDeleteValueW(pe_vars_t *)
ADVAPI32_DLL_RegEnumKeyExW(pe_vars_t *)
ADVAPI32_DLL_RegEnumValueW(pe_vars_t *)
ADVAPI32_DLL_RegOpenKeyExW(pe_vars_t *)
ADVAPI32_DLL_RegQueryInfoKeyW(pe_vars_t *)
ADVAPI32_DLL_RegQueryValueExW(pe_vars_t *)
ADVAPI32_DLL_RegSetValueExW(pe_vars_t *)

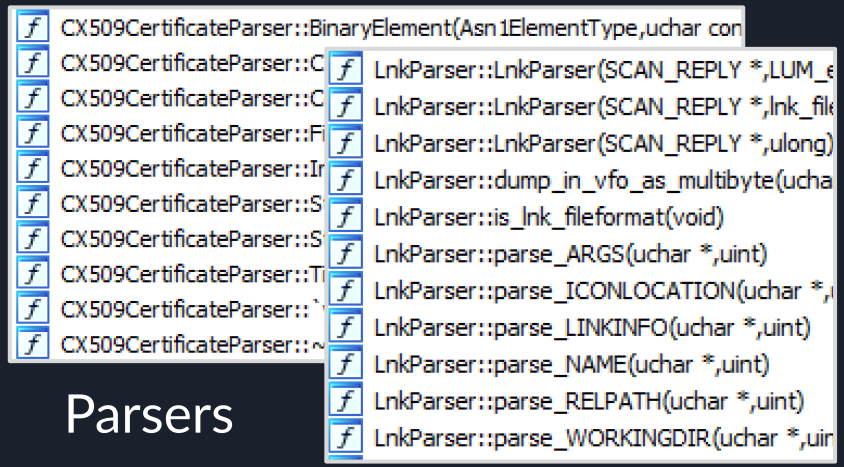
```
void __thiscall ARM_IL_emulator::eIL_shr32f(ARM_IL_emul
{
  char v2; // cl
  unsigned int v3; // edi
  unsigned int v4; // esi
  unsigned __int16 v5; // dx
  unsigned int v6; // edi
  ARM_IL_emulator *v7; // ecx
  ARM_IL_emulator *v8; // [esp+Ch] [ebp-4h]

  v8 = this;
  v2 = **(pQ + 2) & 0x1F;
  v3 = **(pQ + 1);
  v4 = **(pQ + 1) >> v2;
  **pQ = v4;
  v5 = ((v4 == 0) << 6) | (v4 >> 24) & 0x80;
```

```
void __cdecl KERNEL32_DLL_VirtualFree(pe_vars_t *v)
{
  DT_context *v1; // ecx
  signed int v2; // edi
  _int64 v3; // rax
  / al
  _int64 v5; // rax
  ; // rcx
  _int64 v7; // ST08_8
  s v8; // [esp+10h] [ebp-3Ch]
  TypeCallback Callback; // [esp+1Ch] [ebp-30h]
  *v10; // [esp+20h] [ebp-2Ch]
  <3> v11; // [esp+24h] [ebp-28h]
  / [esp+48h] [ebp-4h]

  <3>::Parameters<3>(&v11, v);
  pDTc;
  v8.m_vticks = 32;
  v8.m_init_vticks = &v->vticks32;
  v8.m_pC = v1;
  v2 = 0;
  v12 = 0;
  if ( (!(v11.m_Arg[2].val16 & 0x8000) || !(v11.m_Arg[2].val16 & 0x4000)
    && (!(v11.m_Arg[2].val16 & 0x8000) || !v11.m_Arg[1].val32) )
  {
    if ( v11.m_Arg[2].val16 & 0x4000 )
    {
      v5 = AlignDownToPageSize(v11.m_Arg[0].val64);
      HIDWORD(v6) = HIDWORD(v5);
      LODWORD(v6) = (v6 + 4095) & 0xFFFFF000;
      HIDWORD(v5) = (v5 + v6) >> 32;
      LODWORD(v6) = v5 + v6;
      if ( HIDWORD(v6) >= HIDWORD(v5) && (HIDWORD(v6) > HIDWORD(v5) || v5
```

The Remaining 98%
43,000+ Functions



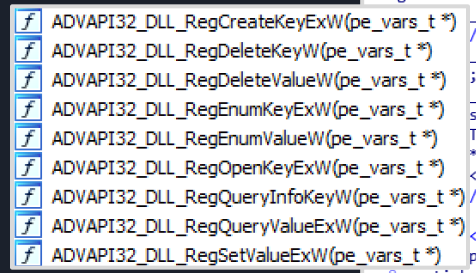
Unpackers



Parsers

Windows Emulator

- x86, x64, & ARM
  - Lifted to IL for emulation
- WinAPI & NT Kernel emulation



Tip: the Lua engine is for signatures - don't waste your time trying to do VR like I did

```
void __thiscall ARM_IL_emulator::eIL_shr32f(ARM_IL_emul
{
  char v2; // cl
  unsigned int v3; // edi
  unsigned int v4; // esi
  unsigned __int16 v5; // dx
  unsigned int v6; // edi
  ARM_IL_emulator *v7; // ecx
  ARM_IL_emulator *v8; // [esp+Ch] [ebp-4h]

  v8 = this;
  v2 = **(pQ + 2) & 0x1F;
  v3 = **(pQ + 1);
  v4 = **(pQ + 1) >> v2;
  **pQ = v4;
  v5 = ((v4 == 0) << 6) | (v4 >> 24) & 0x80;
```

```
void __cdecl KERNEL32_DLL_VirtualFree(pe_vars_t *v)
{
  DT_context *v1; // ecx
  signed int v2; // edi
  _int64 v3; // rax
  / al
  _int64 v5; // rax
  ; // rcx
  _int64 v7; // ST08_8
  s v8; // [esp+10h] [ebp-3Ch]
  TypeCallback Callback; // [esp+1Ch] [ebp-30h]
  *v10; // [esp+20h] [ebp-2Ch]
  <3> v11; // [esp+24h] [ebp-28h]
  / [esp+48h] [ebp-4h]

  <3>::Parameters<3>(&v11, v);
  pDTc;
  v8.m_vticks = 32;
  v8.m_init_vticks = &v->vticks32;
  v8.m_pC = v1;
  v2 = 0;
  v12 = 0;
  if ( (!(v11.m_Arg[2].val16 & 0x8000) || !(v11.m_Arg[2].val16 & 0x4000)
    && (!(v11.m_Arg[2].val16 & 0x8000) || !v11.m_Arg[1].val32) )
  {
    if ( v11.m_Arg[2].val16 & 0x4000 )
    {
      v5 = AlignDownToPageSize(v11.m_Arg[0].val64);
      HIDWORD(v6) = HIDWORD(v5);
      LODWORD(v6) = (v6 + 4095) & 0xFFFFF000;
      HIDWORD(v5) = (v5 + v6) >> 32;
      LODWORD(v6) = v5 + v6;
      if ( HIDWORD(v6) >= HIDWORD(v5) && (HIDWORD(v6) > HIDWORD(v5) || v5
```

Conclusion

- Defender is a great target for reverse engineering - much easier than other AVs
- This is just 2% of MpEngine.dll - and just the highlights of my JS research
  - Hope to talk about the Windows x86/x64/ARM binary emulator soon…
- Building custom tools is necessary for this sort of research
- Perception of vulnerability vs. reality
- Sandboxing will help security

Twitter: `@0xAlexei`

Alexei Bulazel

Thank You:

- Rolf Rolles - shell collaborator
- Tavis Ormandy & Natalie Silvanovich / P0 - exposing the engine and answering a some questions
- `@thewack0lian` - initial shell
- Joxean Koret - OG AV hacker
- REcon team

Backup Slides

Foreign Function Interface

- FFI handling functions seem like a good target for VR
- Not user reachable - never declared in `declareGlobalProperties`
- `addForeignSupport` called during engine initialization, but did not observe FFI functions ever actually being added
- Purpose / reachability remains unclear

```
char __thiscall JsDelegateObject_FFI::delegate(JsDelegateObject_FFI *this, int method,
{
  JsDelegateObject_FFI *v5; // edi
  std::_Tree_node<std::pair<unsigned int const ,std::function<JavaScriptInterpreter::FI
  signed int v7; // ebx
  signed int v8; // edi
  signed int v9; // edx
  std::pair<std::unique_ptr<char [0],std::default_delete<char [0]> >,unsigned int> *v10
  int v11; // ecx
  JavaScriptInterpreter::FFI_TypedValue v12; // ST08_8
  char v13; // bl
  int v15; // esi
  std::pair<std::unique_ptr<char [0],std::default_delete<char [0]> >,unsigned int> arg
  JavaScriptInterpreter::FFI_TypedValue ffiRetVal; // [esp+18h] [ebp-28h]
  JavaScriptInterpreter::FFI_TypedValue ffiArg; // [esp+20h] [ebp-20h]
  std::_Tree_iterator<std::_Tree_val<std::_Tree_simple_types<std::pair<unsigned int co
  unsigned int _Keyval; // [esp+2Ch] [ebp-14h]
  int v21; // [esp+3Ch] [ebp-4h]
```

```
v8 = JsObject::genPropHash("Function", 0);
if ( !JsRuntimeState::declare(v3, v8, v7) || !createObjectConstructor(v3, &inf) )
  goto LABEL_106;
v9 = inf;
if ( !inf )
  v9 = 10;
v10 = v9;
v11 = JsObject::genPropHash("Object", 0);
if ( !JsRuntimeState::declare(v3, v11, v10) || !createArrayConstructorAndPrototype(v3, &inf, &nav) )
  goto LABEL_106;
v12 = inf;
if ( !inf )
  v12 = 10;
v13 = v12;
v14 = JsObject::genPropHash("Array", 0);
if ( !JsRuntimeState::declare(v3, v14, v13) || !createStringConstructorAndPrototype(v3, &inf, &nav) )
  goto LABEL_106;
v15 = inf;
if ( !inf )
  v15 = 10;
v16 = v15;
v17 = JsObject::genPropHash("String", 0);
if ( !JsRuntimeState::declare(v3, v17, v16) || !createBooleanConstructorAndPrototype(v3, &inf, &nav) )
  goto LABEL_106;
```

Timers

- Inspired by looking a ways of getting execution during `JsString::initByVector` - maybe fire a timed function?
- Single threaded architecture

Timers

- Inspired by looking a ways of getting execution during `JsString::initByVector` - maybe fire a timed function?
- Single threaded architecture

- `setTimeout()` - set a function to be called in some number of ms, returns ID number
- `clearTimeout()` - delete a timeout by ID number

Timers

- Inspired by looking a ways of getting execution during `JsString::initByVector` - maybe fire a timed function?
- Single threaded architecture

- `setTimeout()` - set a function to be called in some number of ms, returns ID number
- `clearTimeout()` - delete a timeout by ID number

- Idea: maybe timer callbacks will get serviced during execution?
  - Use one to change the vector while it is being copied

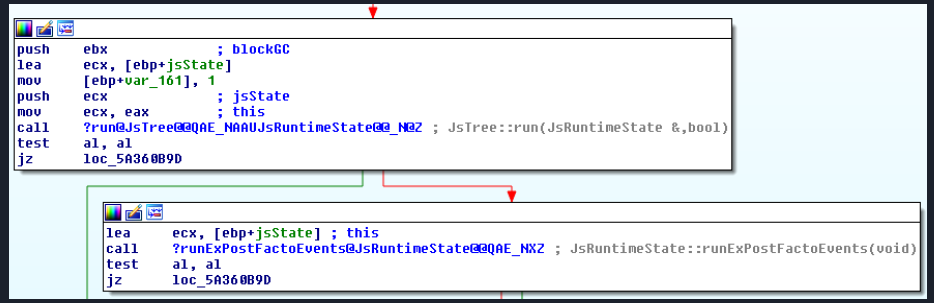
Analyzing `setTimeout()`

- Callbacks stored in a doubly linked list
  - *O(1)* insertion / removal
  - Ordered by timeout time
- Callbacks numbered sequentially from 0 → 100
- Max 101 callbacks ever

```
nextCallBackid = this->m_nextCallBackId;
if ( nextCallBackid > 100 )
{
  result = 0;
}
else
{
  *id = nextCallBackid;
  ifNoCallbacksExist = this->m_timeOutCallBacks._Mysize == 0;
  this->m_nextCallBackId = nextCallBackid + 1;
```

Timeout Dispatch

- Dispatched after `JsTree::run` in `JsRuntimeState::runExPostFactoEvents`
- Actual timeout times are not respected, but timeouts fire in order



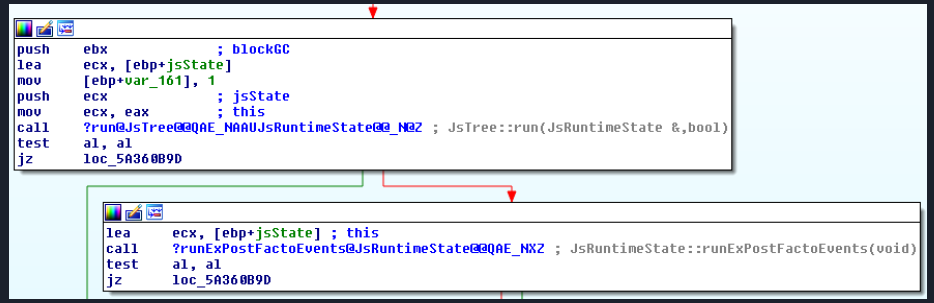
```
push    ebx             ; blockGC
lea     ecx, [ebp+jsState]
mov     [ebp+var_161], 1
push    ecx             ; jsState
mov     ecx, eax        ; this
call    ?run@JsTree@@QAE_NAAUJsRuntimeState@@_N@Z ; JsTree::run(JsRuntimeState &,bool)
test    al, al
jz      loc_5A360B9D
```

```
lea     ecx, [ebp+jsState] ; this
call    ?runExPostFactoEvents@JsRuntimeState@@QAE_NXZ ; JsRuntimeState::runExPostFactoEvents(void)
test    al, al
jz      loc_5A360B9D
```

```
while ( this->m_timeOutCallBacks._Mysize )// loop callbacks
{
  body = this->m_timeOutCallBacks._Myhead->_Next->_Myval.expr;
  callBackVal = body;
  std::list<std::pair<FileStateKey const,unsigned int>,std::allocator<std::pair<FileStateKey const,unsigned int>>>::erase(
    &this->m_timeOutCallBacks,
    &result,
    this->m_timeOutCallBacks._Myhead->_Next);
  funcObjectPInvoke = 0;
  v15 = 0;
  argsList = 0;
  v12 = 0;
  v13 = 0;
  v19 = 0;
  if ( getValueType(body) == 4 )        //  if string
  {
    funcObjectPInvoke = funcObj;
    v15 = funcObj;
    std::vector<unsigned int,std::allocator<unsigned int>>::push_back(&argsList, &callBackVal);
  }
  else if ( getValueType(body) == 8 )   // if function
  {
    funcObjectPInvoke = valueToPtr<JsFunctionObject>(body);
    v15 = funcObjectPInvoke;
  }
  body = 0;
  if ( !preInvokeFunctionThrows(this, funcObjectPInvoke, globalObj, &argsList, 0, &body) )
  {

    std::vector<Forwarded_export_resolution_data_t,std::allocator<Forwarded_export_resolution_data_t>>::_Tidy(&argsList);
    return 0;
  }
  if ( !JsRuntimeState::exceptionThrown(this) )
  {
    if ( body )
    {
      v7 = body->vfptr->declare;
      __guard_check_icall_fptr(body->vfptr->declare);
      if ( !v7(body, this, 0) || !JsTree::run(body, this, 0) )// here we actually run the function
        goto retn;
      funcObjectPInvoke = v15;
    }
```

Timeout Dispatch

- Dispatched after `JsTree::run` in `JsRuntimeState::runExPostFactoEvents`
- Actual timeout times are not respected, but timeouts fire in order



```
push    ebx             ; blockGC
lea     ecx, [ebp+jsState]
mov     [ebp+var_161], 1
push    ecx             ; jsState
mov     ecx, eax        ; this
call    ?run@JsTree@@QAE_NAAUJsRuntimeState@@_N@Z ; JsTree::run(JsRuntimeState &,bool)
test    al, al
jz      loc_5A360B9D
```

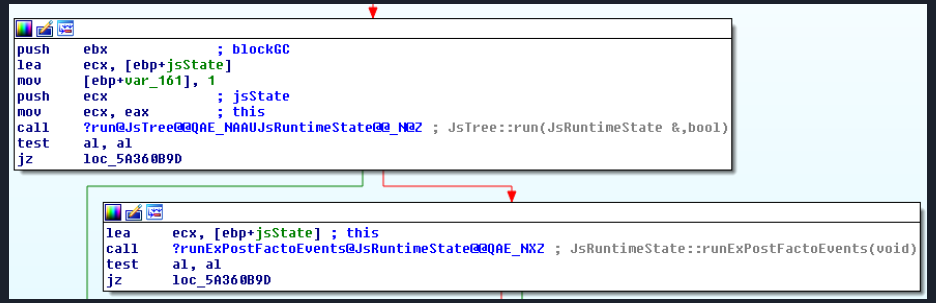
```
lea     ecx, [ebp+jsState] ; this
call    ?runExPostFactoEvents@JsRuntimeState@@QAE_NXZ ; JsRuntimeState::runExPostFactoEvents(void)
test    al, al
jz      loc_5A360B9D
```

```
while ( this->m_timeOutCallBacks._Mysize )// loop callbacks
{
  body = this->m_timeOutCallBacks._Myhead->_Next->_Myval.expr;
  callBackVal = body;
  std::list<std::pair<FileStateKey const,unsigned int>,std::allocator<std::pair<FileStateKey const,unsigned int>>>::erase(
    &this->m_timeOutCallBacks,
    &result,
    this->m_timeOutCallBacks._Myhead->_Next);
  funcObjectPInvoke = 0;
  v15 = 0;
  argsList = 0;
  v12 = 0;
  v13 = 0;
  v19 = 0;
  if ( getValueType(body) == 4 )        //  if string
  {
    funcObjectPInvoke = funcObj;
    v15 = funcObj;
    std::vector<unsigned int,std::allocator<unsigned int>>::push_back(&argsList, &callBackVal);
  }
  else if ( getValueType(body) == 8 )   // if function
  {
    funcObjectPInvoke = valueToPtr<JsFunctionObject>(body);
    v15 = funcObjectPInvoke;
  }
  body = 0;
  if ( !preInvokeFunctionThrows(this, funcObjectPInvoke, globalObj, &argsList, 0, &body) )
  {

    std::vector<Forwarded_export_resolution_data_t,std::allocator<Forwarded_export_resolution_data_t>>::_Tidy(&argsList);
    return 0;
  }
  if ( !JsRuntimeState::exceptionThrown(this) )
  {
    if ( body )
    {
      v7 = body->vfptr->declare;
      __guard_check_icall_fptr(body->vfptr->declare);
      if ( !v7(body, this, 0) || !JsTree::run(body, this, 0) )// here we actually run the function
        goto retn;
      funcObjectPInvoke = v15;
    }
```

Pop the first list entry

Timeout Dispatch

- Dispatched after `JsTree::run` in `JsRuntimeState::runExPostFactoEvents`
- Actual timeout times are not respected, but timeouts fire in order



```
push    ebx             ; blockGC
lea     ecx, [ebp+jsState]
mov     [ebp+var_161], 1
push    ecx             ; jsState
mov     ecx, eax        ; this
call    ?run@JsTree@@QAE_NAAUJsRuntimeState@@_N@Z ; JsTree::run(JsRuntimeState &,bool)
test    al, al
jz      loc_5A360B9D
```

```
lea     ecx, [ebp+jsState] ; this
call    ?runExPostFactoEvents@JsRuntimeState@@QAE_NXZ ; JsRuntimeState::runExPostFactoEvents(void)
test    al, al
jz      loc_5A360B9D
```

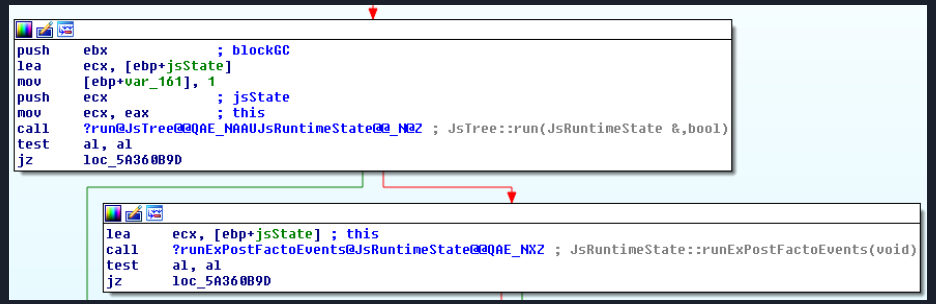
```
while ( this->m_timeOutCallBacks._Mysize )// loop callbacks
{
  body = this->m_timeOutCallBacks._Myhead->_Next->_Myval.expr;
  callBackVal = body;
  std::list<std::pair<FileStateKey const,unsigned int>,std::allocator<std::pair<FileStateKey const,unsigned int>>>::erase(
    &this->m_timeOutCallBacks,
    &result,
    this->m_timeOutCallBacks._Myhead->_Next);
  funcObjectPInvoke = 0;
  v15 = 0;
  argsList = 0;
  v12 = 0;
  v13 = 0;
  v19 = 0;
  if ( getValueType(body) == 4 )        //  if string
  {
    funcObjectPInvoke = funcObj;
    v15 = funcObj;
    std::vector<unsigned int,std::allocator<unsigned int>>::push_back(&argsList, &callBackVal);
  }
  else if ( getValueType(body) == 8 )   // if function
  {
    funcObjectPInvoke = valueToPtr<JsFunctionObject>(body);
    v15 = funcObjectPInvoke;
  }
  body = 0;
  if ( !preInvokeFunctionThrows(this, funcObjectPInvoke, globalObj, &argsList, 0, &body) )
  {

    std::vector<Forwarded_export_resolution_data_t,std::allocator<Forwarded_export_resolution_data_t>>::_Tidy(&argsList);
    return 0;
  }
  if ( !JsRuntimeState::exceptionThrown(this) )
  {
    if ( body )
    {
      v7 = body->vfptr->declare;
      __guard_check_icall_fptr(body->vfptr->declare);
      if ( !v7(body, this, 0) || !JsTree::run(body, this, 0) )// here we actually run the function
        goto retn;
      funcObjectPInvoke = v15;
    }
```

Pop the first list entry

Handling specific to callback being a string or a function

Timeout Dispatch

- Dispatched after `JsTree::run` in `JsRuntimeState::runExPostFactoEvents`
- Actual timeout times are not respected, but timeouts fire in order



```
push    ebx             ; blockGC
lea     ecx, [ebp+jsState]
mov     [ebp+var_161], 1
push    ecx             ; jsState
mov     ecx, eax        ; this
call    ?run@JsTree@@QAE_NAAUJsRuntimeState@@_N@Z ; JsTree::run(JsRuntimeState &,bool)
test    al, al
jz      loc_5A360B9D
```

```
lea     ecx, [ebp+jsState] ; this
call    ?runExPostFactoEvents@JsRuntimeState@@QAE_NXZ ; JsRuntimeState::runExPostFactoEvents(void)
test    al, al
jz      loc_5A360B9D
```

```
while ( this->m_timeOutCallBacks._Mysize )// loop callbacks
{
  body = this->m_timeOutCallBacks._Myhead->_Next->_Myval.expr;
  callBackVal = body;
  std::list<std::pair<FileStateKey const,unsigned int>,std::allocator<std::pair<FileStateKey const,unsigned int>>>::erase(
    &this->m_timeOutCallBacks,
    &result,
    this->m_timeOutCallBacks._Myhead->_Next);
  funcObjectPInvoke = 0;
  v15 = 0;
  argsList = 0;
  v12 = 0;
  v13 = 0;
  v19 = 0;
  if ( getValueType(body) == 4 )        //  if string
  {
    funcObjectPInvoke = funcObj;
    v15 = funcObj;
    std::vector<unsigned int,std::allocator<unsigned int>>::push_back(&argsList, &callBackVal);
  }
  else if ( getValueType(body) == 8 )   // if function
  {
    funcObjectPInvoke = valueToPtr<JsFunctionObject>(body);
    v15 = funcObjectPInvoke;
  }
  body = 0;
  if ( !preInvokeFunctionThrows(this, funcObjectPInvoke, globalObj, &argsList, 0, &body) )
  {

    std::vector<Forwarded_export_resolution_data_t,std::allocator<Forwarded_export_resolution_data_t>>::_Tidy(&argsList);
    return 0;
  }
  if ( !JsRuntimeState::exceptionThrown(this) )
  {
    if ( body )
    {
      v7 = body->vfptr->declare;
      __guard_check_icall_fptr(body->vfptr->declare);
      if ( !v7(body, this, 0) || !JsTree::run(body, this, 0) )// here we actually run the function
        goto retn;
      funcObjectPInvoke = v15;
    }
```

Pop the first list entry

Handling specific to callback being a string or a function

Actually run the function

Vulnerability Ideas

- Firing during `JsString::initByVector` - not going to work

Vulnerability Ideas

- Firing during `JsString::initByVector` - not going to work
- Another idea: UAF if we can free callbacks with `clearTimeout()` during traversal

Vulnerability Ideas

- Firing during `JsString::initByVector` - not going to work

- Another idea: UAF if we can free callbacks with `clearTimeout()` during traversal
    - Single-threaded

Vulnerability Ideas

- Firing during `JsString::initByVector` - not going to work

- Another idea: UAF if we can free callbacks with `clearTimeout()` during traversal
  - Single-threaded
  - Loop only maintains a pointer to the head, not to individual elements

Vulnerability Ideas

- Firing during `JsString::initByVector` - not going to work

- Another idea: UAF if we can free callbacks with `clearTimeout()` during traversal
  - Single-threaded
  - Loop only maintains a pointer to the head, not to individual elements

```
body = this->m_timeOutCallBacks._Myhead->_Next->_Myval.expr;
callBackVal = body;
std::list<std::pair<FileStateKey const,unsigned int>,std::allocator<std::pair<FileStateKey const,unsigned int>>>::erase(
  &this->m_timeOutCallBacks,
  &result,
  this->m_timeOutCallBacks._Myhead->_Next);
```

JsString::initByVector

Creates a new string from an array of strings

```
mpscript> (function(){
            var x = new Array(1,2,"A","B");
            var y = new String(x);
            print(y);
          })()
triggerEvent(): array_join
triggerEvent(): str_valueof
print(): 1,2,A,B
print(): undefined
Log():              <NA>:   0: execution took 68 ticks
Log():              <NA>:   0: final memory used 9KB
Log():              <NA>:   0: total of 0 GCs performed

Ended. Result code: 0
```

JsString::initByVector

```
  first = arrStrings->_Myfirst;
  curentCopyIdx = 0;
  totalLength = 0;
  while ( first != arrStrings->_Mylast )
  {
    currentArrayString = *first;
    if ( *first != 18 && getValueType(*first) == 4 )
    {
      numBytesCurrent = JsString::numBytes(currentArrayString);
      totalLength += numBytesCurrent;
      if ( totalLength < numBytesCurrent )
        return 0;
    }
    ++first;
  }
  if ( !totalLength )
  {
    *str = 18;
    return 1;
  }
  rawBuf = 0;
  if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
  {
    newBuf._Myptr = rawBuf;
    v19 = 0;
    for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
    {
      currentElem = *i;
      if ( *i != 18 && getValueType(*i) == 4 )
      {
        numBytes = JsString::numBytes(currentElem);
        if ( numBytes )
        {
          if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
            goto LABEL_22;
          curentCopyIdx += numBytes;
          if ( curentCopyIdx < numBytes )
            goto LABEL_22;
        }
      }
    }
    if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
    {
LABEL_22:
      if ( newBuf._Myptr )
        free(newBuf._Myptr);
      return 0;
    }
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 1;
  }
  return 0;
}
```

JsString::initByVector

```
  first = arrStrings->_Myfirst;
  curentCopyIdx = 0;
  totalLength = 0;
  while ( first != arrStrings->_Mylast )
  {
    currentArrayString = *first;
    if ( *first != 18 && getValueType(*first) == 4 )
    {
      numBytesCurrent = JsString::numBytes(currentArrayString);
      totalLength += numBytesCurrent;
      if ( totalLength < numBytesCurrent )
        return 0;
    }
    ++first;
  }
  if ( !totalLength )
  {
    *str = 18;
    return 1;
  }
  rawBuf = 0;
  if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
  {
    newBuf._Myptr = rawBuf;
    v19 = 0;
    for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
    {
      currentElem = *i;
      if ( *i != 18 && getValueType(*i) == 4 )
      {
        numBytes = JsString::numBytes(currentElem);
        if ( numBytes )
        {
          if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
            goto LABEL_22;
          curentCopyIdx += numBytes;
          if ( curentCopyIdx < numBytes )
            goto LABEL_22;
        }
      }
    }
    if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
    {
LABEL_22:
      if ( newBuf._Myptr )
        free(newBuf._Myptr);
      return 0;
    }
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 1;
  }
  return 0;
}
```

Sum the byte lengths of each vector element

JsString::initByVector

```
first = arrStrings->_Myfirst;
curentCopyIdx = 0;
totalLength = 0;
while ( first != arrStrings->_Mylast )
{
  currentArrayString = *first;
  if ( *first != 18 && getValueType(*first) == 4 )
  {
    numBytesCurrent = JsString::numBytes(currentArrayString);
    totalLength += numBytesCurrent;
    if ( totalLength < numBytesCurrent )
      return 0;
  }
  ++first;
}
if ( !totalLength )
{
  *str = 18;
  return 1;
}
rawBuf = 0;
if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
{
  newBuf._Myptr = rawBuf;
  v19 = 0;
  for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
  {
    currentElem = *i;
    if ( *i != 18 && getValueType(*i) == 4 )
    {
      numBytes = JsString::numBytes(currentElem);
      if ( numBytes )
      {
        if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
          goto LABEL_22;
        curentCopyIdx += numBytes;
        if ( curentCopyIdx < numBytes )
          goto LABEL_22;
      }
    }
  }
  if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
  {
LABEL_22:
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 0;
  }
  if ( newBuf._Myptr )
    free(newBuf._Myptr);
  return 1;
}
return 0;
}
```

Sum the byte lengths of each vector element

allocate a buffer of size of the sum of lengths

JsString::initByVector

```
first = arrStrings->_Myfirst;
curentCopyIdx = 0;
totalLength = 0;
while ( first != arrStrings->_Mylast )
{
  currentArrayString = *first;
  if ( *first != 18 && getValueType(*first) == 4 )
  {
    numBytesCurrent = JsString::numBytes(currentArrayString);
    totalLength += numBytesCurrent;
    if ( totalLength < numBytesCurrent )
      return 0;
  }
  ++first;
}
if ( !totalLength )
{
  *str = 18;
  return 1;
}
rawBuf = 0;
if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
{
  newBuf._Myptr = rawBuf;
  v19 = 0;
  for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
  {
    currentElem = *i;
    if ( *i != 18 && getValueType(*i) == 4 )
    {
      numBytes = JsString::numBytes(currentElem);
      if ( numBytes )
      {
        if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
          goto LABEL_22;
        curentCopyIdx += numBytes;
        if ( curentCopyIdx < numBytes )
          goto LABEL_22;
      }
    }
  }
  if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
  {
LABEL_22:
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 0;
  }
  if ( newBuf._Myptr )
    free(newBuf._Myptr);
  return 1;
}
return 0;
}
```

Sum the byte lengths of each vector element

allocate a buffer of size of the sum of lengths

Copy each vector element into the allocated buffer

JsString::initByVector

```
first = arrStrings->_Myfirst;
curentCopyIdx = 0;
totalLength = 0;
while ( first != arrStrings->_Mylast )
{
  currentArrayString = *first;
  if ( *first != 18 && getValueType(*first) == 4 )
  {
    numBytesCurrent = JsString::numBytes(currentArrayString);
    totalLength += numBytesCurrent;
    if ( totalLength < numBytesCurrent )
      return 0;
  }
  ++first;
}
if ( !totalLength )
{
  *str = 18;
  return 1;
}
rawBuf = 0;
if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
{
  newBuf._Myptr = rawBuf;
  v19 = 0;
  for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
  {
    currentElem = *i;
    if ( *i != 18 && getValueType(*i) == 4 )
    {
      numBytes = JsString::numBytes(currentElem);
      if ( numBytes )
      {
        if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
          goto LABEL_22;
        curentCopyIdx += numBytes;
        if ( curentCopyIdx < numBytes )
          goto LABEL_22;
      }
    }
  }
  if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
  {
LABEL_22:
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 0;
  }
  if ( newBuf._Myptr )
    free(newBuf._Myptr);
  return 1;
}
return 0;
}
```

Sum the byte lengths of each vector element

allocate a buffer of size of the sum of lengths

Copy each vector element into the allocated buffer

TOCTOU?

JsString::initByVector

`numBytes` does not callback into JS

`unsigned int` overflow checking

```
first = arrStrings->_Myfirst;
curentCopyIdx = 0;
totalLength = 0;
while ( first != arrStrings->_Mylast )
{
  currentArrayString = *first;
  if ( *first != 18 && getValueType(*first) == 4 )
  {
    numBytesCurrent = JsString::numBytes(currentArrayString);
    totalLength += numBytesCurrent;
    if ( totalLength < numBytesCurrent )
      return 0;
  }
  ++first;
}
if ( !totalLength )
{
  *str = 18;
  return 1;
}
rawBuf = 0;
if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
{
  newBuf._Myptr = rawBuf;
  v19 = 0;
  for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
  {
    currentElem = *i;
    if ( *i != 18 && getValueType(*i) == 4 )
    {
      numBytes = JsString::numBytes(currentElem);
      if ( numBytes )
      {
        if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
          goto LABEL_22;
        curentCopyIdx += numBytes;
        if ( curentCopyIdx < numBytes )
          goto LABEL_22;
      }
    }
  }
  if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
  {
LABEL_22:
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 0;
  }
  if ( newBuf._Myptr )
    free(newBuf._Myptr);
  return 1;
}
return 0;
}
```

Sum the byte lengths of each vector element

allocate a buffer of size of the sum of lengths

Copy each vector element into the allocated buffer

TOCTOU?

JsString::initByVector

`numBytes` does not callback into JS

`unsigned int` overflow checking

`getValueType` does not callback into JS

```
first = arrStrings->_Myfirst;
curentCopyIdx = 0;
totalLength = 0;
while ( first != arrStrings->_Mylast )
{
  currentArrayString = *first;
  if ( *first != 18 && getValueType(*first) == 4 )
  {
    numBytesCurrent = JsString::numBytes(currentArrayString);
    totalLength += numBytesCurrent;
    if ( totalLength < numBytesCurrent )
      return 0;
  }
  ++first;
}
if ( !totalLength )
{
  *str = 18;
  return 1;
}
rawBuf = 0;
if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
{
  newBuf._Myptr = rawBuf;
  v19 = 0;
  for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
  {
    currentElem = *i;
    if ( *i != 18 && getValueType(*i) == 4 )
    {
      numBytes = JsString::numBytes(currentElem);
      if ( numBytes )
      {
        if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
          goto LABEL_22;
        curentCopyIdx += numBytes;
        if ( curentCopyIdx < numBytes )
          goto LABEL_22;
      }
    }
  }
  if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
  {
LABEL_22:
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 0;
  }
  if ( newBuf._Myptr )
    free(newBuf._Myptr);
  return 1;
}
return 0;
}
```

Sum the byte lengths of each vector element

allocate a buffer of size of the sum of lengths

Copy each vector element into the allocated buffer

TOCTOU?

JsString::initByVector

`numBytes` does not callback into JS

`unsigned int` overflow checking

`getValueType` does not callback into JS

VT call, but no VT functions callback into JS

```
Js[Buf,
   Concat,
   Ref,
   Sub]String::
    localCopyToBuffer
```

```
first = arrStrings->_Myfirst;
curentCopyIdx = 0;
totalLength = 0;
while ( first != arrStrings->_Mylast )
{
  currentArrayString = *first;
  if ( *first != 18 && getValueType(*first) == 4 )
  {
    numBytesCurrent = JsString::numBytes(currentArrayString);
    totalLength += numBytesCurrent;
    if ( totalLength < numBytesCurrent )
      return 0;
  }
  ++first;
}
if ( !totalLength )
{
  *str = 18;
  return 1;
}
rawBuf = 0;
if ( JsHeap::alloc<char>(&jsState->m_heap, totalLength, &rawBuf) )
{
  newBuf._Myptr = rawBuf;
  v19 = 0;
  for ( i = arrStrings->_Myfirst; i != arrStrings->_Mylast; ++i )
  {
    currentElem = *i;
    if ( *i != 18 && getValueType(*i) == 4 )
    {
      numBytes = JsString::numBytes(currentElem);
      if ( numBytes )
      {
        if ( !copyToBuffer(currentElem, 0, numBytes, &rawBuf[curentCopyIdx]) )
          goto LABEL_22;
        curentCopyIdx += numBytes;
        if ( curentCopyIdx < numBytes )
          goto LABEL_22;
      }
    }
  }
  if ( !JsString::initByReceipt(jsState, &newBuf, totalLength, str) )
  {
LABEL_22:
    if ( newBuf._Myptr )
      free(newBuf._Myptr);
    return 0;
  }
  if ( newBuf._Myptr )
    free(newBuf._Myptr);
  return 1;
}
return 0;
}
```

Sum the byte lengths of each vector element

allocate a buffer of size of the sum of lengths

Copy each vector element into the allocated buffer

TOCTOU?