Towards Revealing JavaScript Program Intents using Abstract Interpretation

Gregory BLANC & Youki KADOBAYASHI

Internet Engineering Laboratory (IPLab)
Nara Institute of Science and Technology, Japan

gregory@is.aist-nara.ac.jp

November 17th, 2010
6th Asian Internet Engineering Conference
Bangkok, Thailand
The Web is an Application of the Internet

- Web applications run on top of servers that are connected to the Internet
- Web applications communicate using the HTTP/S protocol
- Modern web applications make use of a bunch of programming languages and formats
- It is a popular used by many users via many terminals (desktop computer, laptops, mobile phones, smartphones, game consoles, etc.)
Web 2.0: a Paradigm Shift

...in the usage

- Web-as-a-Participation-platform
  - wikis, rich-feature blogs, content sharing platforms, social networks, etc.
- Web Services (WS)
  - mashups, aggregation portals, mobile services, software-as-a-service, managed services, cloud computing, etc.

...in the development

- intensive use of the AJAX framework
- use of APIs/plugins to extend the browser’s capabilities
- applications are pushed on the client-side
JS Malware and Related Threats

Definition

Attacks that abuse the web browsers capabilities to execute malicious script-code within the victims local execution context\(^a\)


strengths

- stealth
- transformation
- concealment

purposes

- phishing, drive-by download, heapspray, etc.
- information leakage, LAN exploits, etc.
- (XSS) worm propagation, (XSS) botnet, etc.
Defense issues

Defending is difficult:

- it resembles benign HTTP contents: uses widespread scripting language and legal Same-Origin Policy bypasses, exploits logic flaws, embodies many different purposes, is polymorphic
- it uses obfuscation among other techniques to conceal contents subjected to pattern-matching
- it is confusing since it relies on many layers of obfuscation and redirection
Deployment issues

Deploying is difficult:

- servers are excluded from the game since some attacks avoid reaching the server. Plus, it is impossible to secure every server in the world...
- browsers that include security plugins are not able to overcome costly problems such as obfuscation or legal injection of code in pivot attacks
- users are not always willing to let down Web 2.0 functionalities and therefore blocking JavaScript is not an option
From Intuition to Abstraction

Intuition
In this research, we propose a different approach to the description of script behaviors inspired by the stages carried out by malicious scripts upon execution:

- redirection and cloaking
- deobfuscation
- environment preparation
- exploitation

We believe we can capture the intent of carrying out these activities hidden within the analyzed script.

---

Detection and Analysis of Drive-by Download Attacks and Malicious JavaScript Code, M. Cova et al., WWW 2009

Abstraction
Malicious programs are characterized by the expression of the above activities and may be abstracted into such functional activities. At that level of abstraction, we believe most of malicious programs may share a common phylogenetic root.
We propose a client-side proxy that will provide abstraction of script-based malware into functional units to:

- alleviate the load on the user’s browser
- provide a sound and complete analysis
- maximize the code coverage of the analysis
- provide realtime analysis
The Web cluster on the leftmost represents a malicious script distribution site while the Web server mashes contents from different sources (including the malicious Web cluster) into pages served to the victim located in the private network at the rightmost.
Before

str = "qndy‘mh)(:gtobuhno!qndy‘mh)(!zw‘s!doeds!&enbtlldou/bsd‘udDmdldou)&nckdbu&(:doeds/rdu@uushctud)&he-&&doeds&(:doeds/rdu@uushctud)&bm‘rrhe-&&b&&m&&#rhe;C***E8**&7B447,74&**@2,00**&E1,892@,1**1B**&15G**B38**&D27&(usx!zw‘s!‘rp<!doeds/Bsd‘udNckdbu)&l*&#ry**&lm3&**/**&Y**#LM**&I**&UUQ&-&&(:"  // the above string has been abbreviated

str2 = "";

for (i = 0; i < str.length; i++)
{
    str2 = str2 + String.fromCharCode(str.charCodeAt(i) ^ 1);
};
After

\[ L_1 : \text{Create} (L_{Downloader}, \text{asq}) \rightarrow L_2 \]
\[ L_2 : \text{Create} (L_{ShellCommand}, \text{ass}) \rightarrow L_3 \]
\[ L_3 : \text{Create} (L_{FileSystemAccess}, \text{asst}) \rightarrow L_4 \]
\[ L_4 : \text{asst}.\text{Write} (\text{asq}) \rightarrow L_5 \]
\[ L_5 : \text{Create} (L_{FileObject}, \text{imya}, <\text{filepath}>) \rightarrow L_6 \]
\[ L_6 : \text{imya} := \text{Save} (\text{asst}) \rightarrow L_7 \]
\[ L_7 : \text{ass} = \text{Execute} (\text{imya}) \]
Decision Module

goal  decide whether a given script is malicious or not

method  trace JavaScript program objects or variables using data flow analysis in order to cluster instructions that are using these objects. These clusters are called functional units. We then extract an abstract functional model of this program which is the combination/sequence of functional units. This abstract model will help determine whether the program intends to carry out a benign or a malicious operation

requirements  provide a deobfuscated JavaScript program or an expression of the plain instructions that were once obfuscated
A functional unit is a set of instructions that expresses the same functional activity (e.g., download, storage, execution, etc.). To achieve such abstraction, we need a semantic understanding of the analyzed scripting language. We need to define a semantic for the targeted scripting language in order to classify native objects, functions to a finite set of functions.
Abstract Semantic Graph

Download Shell Application

Filesystem Access

VarStatement

AssignExpr

Name

asq

left

FunctionCall

right

DotAccessor

String

msxml2.XMLHTTP

String

CreateObject

left

FunctionCall

right

DotAccessor

String

GET

String

<URL>

False

Name

open

left

FunctionCall

right

DotAccessor

Name

send

left

FunctionCall

right

DotAccessor

Name

responseBody

VarStatement

AssignExpr

Name

ass

left

FunctionCall

right

DotStatement

VarStatement

AssignExpr

Name

imya

left

FunctionCall

right

Name

shellexecute

VarStatement

AssignExpr

Name

asst

left

FunctionCall

right

DotAccessor

String

adodb.stream

Name

CreateObject

left

OpEqual

DotAccessor

left

Number

right

right

Name

type

FunctionCall

DotAccessor

Name

Write

VarStatement

AssignExpr

String

..\svchost.exe

FunctionCall

DotAccessor

Name

SaveToFile

left

FunctionCall

right

Name

Close
Comparison of Abstract Functional Models

Decision is made by comparing the extracted abstract model of the analyzed script with known models.

Building corpora

- ontology
- machine learning

We can apply graph comparison or phylogeny-based methods

Minimal models

There is a need to survey corpora of both benign, malicious and obfuscated scripts to draw minimal models that allow distinguishing the different datasets
Deobfuscation Module

**Goal** provide a deobfuscated code that will ease the analysis of the JS instructions

**Method** deduce the results of the deciphering routine of the obfuscation scheme by using automated deduction. Without executing the code, the instructions are clustered by tracing the flow of variables caught into critical sinks. These instructions are then transformed to first-order logical clauses that will be fed to a theorem prover

**Requirements** provide enough code and context to detect and simulate the deciphering routine
Deobfuscation Challenges

Theorem provers are not readily usable for the purpose of deobfuscation. In this research, we wish to implement a rewriting system through demodulation. However, there are a few challenges:

- transition axioms: theorem provers do not understand JavaScript native objects
- termination property: rewriting systems need a condition to stop processing
- recursive processing: theorem provers do not handle loops
Prefetching (Content Aggregation) Module

**goal**  provide enough code to the deobfuscation module to conduct deciphering and reversing

**method**  prefetch all web page contents likely to contain malicious JavaScript and mash all the contents in a single web page to prevent anti-analysis tricks
We proposed a proxy-based solution...

- for realtime analysis of client-side JavaScript threats
- that alleviates the browser from any load or risk
- that does not put trust into the server side
- that offers a context by preserving realtime DOM and URL information as well as the HTML contents of the analyzed webpage
- that maximize the coverage of analyzed code by using static techniques

Advantages compared to other methods

- better code coverage
- lower false positive rate (deobfuscation)
- offers the opportunity to carry out online analysis
- safer
Thank you for your attention!

Any questions?

more information at
http://iplab.aist-nara.ac.jp/~gregory/
Browser Model Shift: Before
Sample reconnaissance code

```javascript
<script>
    function loaded(){// resource exists}
    function timed_out(){// resource does not exist}
    function err(){// requesting the resource errs}
    i = new Image(); i.onload = loaded;
    i.onerror = err;
    window.setTimeout(timed_out ,1000);
    i.src = "http://target.tld/path";
</script>
```

Real-life examples

- XSS worms: Samy, Yamanner, Justin.tv, Mikeey
- XSS botnets: BeEF, BrowserRider
- Insider attacks: reconnaissance, CSRF, information gathering
Web 2.0 Security Solutions

On the Server Side
- integrating security into development: SDL, code auditing
- detecting attacks by signature matching: IDS, IPS
- learning the application and controlling input: WAF

On the Client Side
- allowing only scripts which prevention is not harmful: sandbox
- following critical assets to prevent leakage: tainting
- monitoring the sequence of activities: control flow analysis
- detecting attacks by pattern/behavior matching: heuristics
- building benign web application models: machine learning

Client-server Collaboration
- enforcing server-side code constraints on the client-side: policies
- rewriting possibly malicious instructions: interposition
Obfuscation

**Definition**

The goal of program obfuscation is to make a program unintelligible while preserving its functionality\(^1\).

**Techniques\(^2\)**

- simples techniques: string splitting and escaping, plain encryption, browser exploitation tricks, script packing
- monoalphabetic permutated encryption
- context dependent obfuscation (anti-deobfuscation tricks)

---

\(^1\) *On the (Im)possibility of Obfuscating Programs*, B. Barak et al., 21\(^{st}\) Annual IACR Crypto Conference, 2001

Obfuscated program: Dean Edwards packer’s example

eval(function(p,a,c,k,e,d) { e=function(c) {
return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?String.fromCharCode(c+29):c.toString(36))}; if('').replace(/\^/,String) { while(e(c--)d[e(c)]=k[c]||e(c); k=[function(e) {
return d[e]}; e=function() { return '\w+' }; c=1}; while(c--) if(k[c]) p=p.replace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]); return p}('6 b="D://C.u-B.A/z.8" ; y{6 4= a.x("w.v","" ); 4.t("s",b,0); 4.r(); 3.q=1; 3.7(); 3.p(4.o); 5= "..\\n.m"; 3.1(5,2); 3.k(); 6 9=a.j("i.h",""); 9["g"]("f.8","/c "+5,""","7",0)} d(e){}) ,40,40,
| |\|as\xml|path|***|open|exe|****|ado\url||****| |cmd|ShellExecute|Application |Shell|createobject|close|savetofile|pif |svchost|*************|write|type|****| GET|****| |XMLHTTP|Microsoft|************* |try|*|*|*|*|http.split(''|'),0,{}))
## Deobfuscation

### Techniques

<table>
<thead>
<tr>
<th>Category</th>
<th>Functions/Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>objects</td>
<td>the HTML document, the current URL</td>
</tr>
<tr>
<td>critical sinks</td>
<td>eval(), document.write()</td>
</tr>
<tr>
<td>dangerous functions</td>
<td>encode(), unescape(), fromCharCode(), charCodeAt(), + (concatenation)</td>
</tr>
<tr>
<td>dangerous events</td>
<td>onLoad(), onUnload(), onSubmit()</td>
</tr>
<tr>
<td>methodology</td>
<td>strip JavaScript code off from any HTML tags, then replace critical sinks with print() and execute. Repeat until the code is deobfuscated</td>
</tr>
</tbody>
</table>

---

Prefetching Linked Contents from a Web Page

```html
<script src="b.com/s.js"></script>
<div onclick="click.js"></div>
<img src="javascript:x.js"/>
```
We want to rewrite a given script with a simplified set of instructions that can be deducted from the original set of instructions.

Ex: given the following set of instructions:

```javascript
nuc="";
b="Str"+nuc+"ing"+nuc+"."+nuc+"from"+nuc+"Ch"+nuc+"arCo"+nuc+"de"+nuc+"(a"+nuc+")";
eval(b);
```

we want to deduce the following instruction:

```javascript
eval("String.fromCharCode(a"));
```
SpyProxy: Execution-based Detection of Malicious Web Content

**reference**  A. Moshchuk et al. @ USENIX Security 2007

**summary**  The authors designed a proxy solution that dynamically execute JS candidates and forward the URL or not upon execution results.

Detection and Analysis of Drive-by Download Attacks and Malicious JavaScript Code

**reference**  M. Cova, C. Kruegel and G. Vigna @ WWW 2010

**summary**  The authors combine anomaly detection, to detect malicious JS code on features modeled on the steps of a malicious exploitation, with emulation to dynamically execute the obfuscated code and to draw the necessary features using instrumentation.
Analyzing Obfuscated JavaScript Code

**Obfuscated Malicious JavaScript Detection using Classification Techniques**

**reference** P. Likarish, E. Jung, and I. Jo @ MALWARE 2009

**summary** The authors trained a classifier on a corpus of benign scripts using 65 features relevant to obfuscated scripts

**Detection for JavaScript Obfuscation Attacks in Web Pages through String Pattern Analysis**

**reference** Y. Choi et al. @ FGIT 2009

**summary** The authors use static data flow analysis to extract string candidates and decide on whether the string are obfuscated using 3 metrics

**Malicious Web Content Detection by Machine Learning**

**reference** Y. Hou et al. in Expert Systems with Applications #37, 2010

**summary** The authors used a mix of 171 features to judge on the nature of a DHTML page while being resilient to obfuscation
Online Deobfuscators

- JSunpack (iDefense): http://jsunpack.jeek.org/dec/go
- The Ultimate Deobfuscator (WebSense)

These solutions are not practical in the case of a realtime defense system.