An empirical study of Android APK disributuion sites using headless browser with navigation scripting

Ruo Ando[1,a)]

Abstract: With rising popularization of Android application, Android has become attractive target for malware developers. More comprehensive view of infected APK distribution ecosystem should be helpful for malware analysis. In this paper we report an empirical study for Android APK distribution using headless browser. Navigation scripting with JavaScript enables more interactive web page crawling in order to fetch the results after dynamic web page loading. We present the implementation detail of navigation scripting for gathering information of unofficial APK sites. In experiment, we show statistical analysis of current situation of destination address of APK communications.

1. Introduction

1.1 Android security

Over the last decade, there have been rapid performance improvement of smartphones in both its computing power and connectivity which came nearly to the level available on small desktop computers. In addition to becoming pervasive and ubiquitous, computing resources of smartphones including storage and sensors are rapidly becoming rich featured. On the downside, the number of malwares targeting Android smartphones is drastically increasing especially, for large scale exploitation. The large market share of Android makes Android platform extremely attractive target for malware attacks. Especially, the openness of Android markets are unfortunately one of the aspects which makes it friendly for malware authors. With the openness of APK distribution environment, malware developers can freely upload their malicious applications which targets unsuspecting victims to install them. As a result, current smartphone platforms of Android unexpectedly involve large ecosystems of untrusted third party applications which are often integrated with SNS such as Facebook and twitter as well ac remote sensing devices. Among these apps, unsuspected users are required to provide passwords upon installation or personal information.

The fluidity of the markets also presents enormous security challenges. Rapidly developed and deployed applications [40], coarse permission systems [16], privacy-invading behaviors [14, 12, 21], malware ,and limited security models have led to exploitable phones and applications.

1.2 Static analysis

We present the first large scale analysis of about 800000 applications. In particular, we have extracted about 12000 URLs and expose IP adresses aud country code.

Performing a large-scale study by actually running APK has several major drawbacks. First of all, physically acquiring the trace logs of ten thousands of devices to study would be expensive.Moreover, some of them may be another research effort to operate outside the system for which they are designed. Unsurprisingly, static analysis performs better than dynamic analysis as it does not require running environmrnt.

2. Web scraping

Recently web scraping is widely adopted for researching security topics such as deep web exploration and malicious software distribution. Basically, web scraping is a procedure for processing the HTML of web page to retrieve data for converting the 縲 fetched web page to another format including XML on JSON. Often scraper Simulate a human browsing a web site with browser application. With a proper implementation of scraper, user can access a web page even with stateless page transition exactly as if a real browser would do. The Web Server recognize your access as human manipulation and the Send back the page including specific information you would like to extract.

2.1 AJAX

Trend in the last decade for pursuit of more interactive browser interaction have seen AJAX which stands for asynchronous JavaScript and XML. AJAX is a terminology for pointing a group of web development technologies for handling interaction of client side web applications. AJAX enables web client application to send query and retch the response from a server asynchronously without interfering with page appearance and its transition of current web page. AJAX does not always require XML. Instead, JSON is often used and the request do not always need to be asynchronous. AJAX is sort of comprehensive technology related to HTML and CSS which can be combined for marking up and arranging information. One important object AJAX handles

[1] Network Security Institute, National Institute of Information and Communications Technology 4-2-1 Nukui-Kitamachi, Koganei, Tokyo 184-8795, Japan
[a)] ruo@nict.go.jp

is DOM which is accessed by Java Script for dynamically displaying and enabling client to interact with presented document. JavaScript and the XML HttpRequest object provide a interface for interacting asynchronously between browser and server instead of reloading full page.

2.2 Testing framework and unit testing

Since Smalltalk was introduced in 1994 by Kent Beck, testing framework as a variant of object-oriented approach has been gradually evolved so that part of this approach is adopted and ported to variety of other environments and languages. In testing framework, an object as test case handle the execution of a single test. For simplicity, each test is isolated from the others. Then, it generates all its data before performing and destroys it when its execution is completed. As variations, there are enormous variations of automated tests such as eclipse groovy. Particularly, the Durandal Test Framework for unit testing is adopted by PhamtomJS and Jasmine. For performing the practice of testing a small piece of code, unit testing is adopted. With unit testing shipped, a smaller unit of code is isolated as a function or area of code. By doing this, we can predictably determine if the part of code behaves as expected. Unit testing is the practice of testing a smaller unit of code, which can be a function or area of code that we can isolate. This gives us the ability to determine if the function behaves as expected. These tests are independent of each other and can be executed individually. Each can verify for outputs based on the given inputs, determine if the process will cause errors, and also check if the process can handle exceptions. Using unit testing, we can catch code problems and trace them easily. There are more benefits of using unit testing, and it is probably one of the necessities in programming

3. System overview

In this section we show the detail of proprosed system. Our system is designed for gathering information and APK files from official and unofficial distribution sites. As we discussed before, for getting APK, browser should pass some interactions with distribution sites such as entering ID and clicking buttons. That is because we adopts headless browser technologies. On the other hand, scriptable API on this layer is written by Java Scripts in asynchronous manner with many callbacks which result in that scripting imposes a great burden ou progrommers about controlling concurrency. casperjs with navigation scripting is introduced to reduce the yiming complexy by automatically handles synchronization to some extent. Unfortunately, these two scripting methods are basically designed for single interaction with AJAX. Thus these cannot be fully adopted for the situation where several procesyes of Java Scripts are running concurrently. In this phase we need additional layer for handling synchronization (mainly for timing).

Proposed system is divided into four parts: Linux, phantomJS scriptable APIs, navigation scriptor with CasperJS and perl spripts. Figure shows interaction between phantomJS and CasperJS. The lower part of phantomJS takes managements in handling inputs to browser with event base loop. the upper part of casperJS provides friendly interface for coping with xpath selector and wait() module for keeping good synchrouization. Figure shows overview of proposed system. the first lowest layer of Linux kernel runs JS processes concurrently. phantomJS of second layer handles I/O manipulation between crawler and browser. CasperJS processes Xpath data and ifs completion of retrieving. Finally perl script manages concurrent execution and each timeout.

4. PhantomJS

PhantomJS is implemented based on Qt webkit aimed for providing a new solution for testing web applications with headless browser. Another feature of PhantomJS is a utility for dynamically capturing and rendering web pages as images which makes it possible to manipulate web pages into different forms in programmable style. PhantonJS also provide functions to get network level information such as response time about the page and site resources. PhantomJS adopts Qt Webkit for implementing core browser capability and also uses WebKit JavaScript engine for interpreting and evaluating script. PhantomJS has a thrust for capability of implementing anything and everything which a webkit-based browser such as chrome, safari and opera browser. PhantomJS is more than just a browser, providing programming interface of CSS selector, DOM manipulation, JSON, HTML5 Canvas, and SVG. As we discussed later, PhantomJS provides system related API performing file system I/ O, accessing system environment variables.

4.1 Headless Browser

In this paper we propose an application of headless browsers for automated control of a web page. Headless browser provides automation similar to web browsers which executing via a CUI or using network communication. Headless browser is usually adopted for inspecting web pages as they are able recognize and render HTML exactly like a browser would concerning page layout, colour, font selection and execution of JavaScript and AJAX. Particalarly, Java Script execution and AJAX are not available by other testing methods. Since 2009, Google has been applying headless browser for handling indexed content from websites that use AJAX.

4.2 meta rules

Qt metacall is core routine for headless browser manipulation. The Q OBJECT macro is defined in src/corelib/kernel/qobjectdefs.h. In that directly many functions implemented for fully support Qt meta object system.

```
1 int qt metacall(QMetaObject::Call call, int id,
2 void __arguments);
```

List above shows basic template of qt metacall. Qt uses qt metacall() to invoke the slots corresponding to the signal in the case that a signal is emitted. QMetaObject::Call which is first parameter is then set to QMetaObject::InvokeMetaMethod. For setting or getting properties of the meta-object, the qt metacall() function adopts other types of access methods. First argument of write system call is socket descriptor and second one is socket buffer.

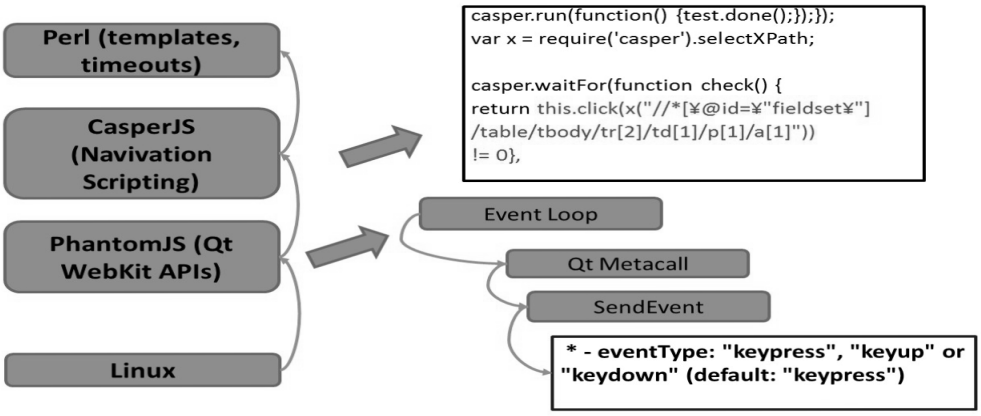

Fig. 1 Proposal system overview1: PhantomJS and CasperJS.

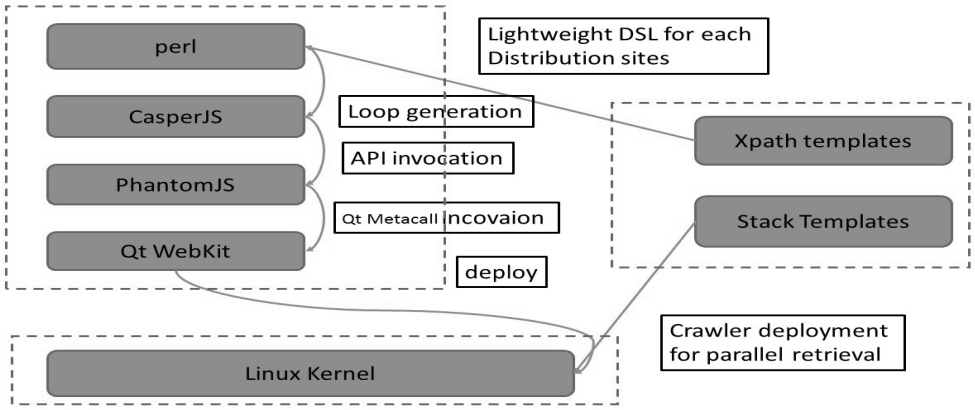

Fig. 2 Proposal system overview2.

4.3 SendEvent

The key feature of PhantomJS is the Webpage API for supporting sending events to the page. Events includes mouse events and keyboand events. For handling these events, PhantomJS sets sendEvent triggers directly to the target container. sendEvent refers the x and y coordinates which are passed where the event should be triggered. This makes it possible to simulate user behavior of performing the actual event, such as a click.

```
1 sendEvent(mouseEventType[, mouseX, mouseY, button='
      left'])
```

List above depicts a template of SendEvent routine. monseEventType such as mouseup, mousedown, mousemove, doubleclick and click is set in the first argument. The following arguments represent the mouse position for the event. The last argument of button Which is set to left in default specifies the button type to push. When mousemove type is set, there is no button pressed.

4.4 Module API

Custom modules are useful for writing custom objects and api sets. Module API makes it possible to create original modules and import it to an arbitrary point of actual implementation. implemented in the part ofThe built-in modules are webpage, system, fs (File System), and webserver.

4.4.1 WebPage API

WebPage API is core functionalities for accessing and manipulating web documents including DOM objects. Access, control and manipulation requires WebPage API. WebPage API makes easy for providing a useful interface to activate capturing of events like error reporting, navigation to other page and page reloading. In addition, web API provides capability of capturing

web pages and rendering it as images. Another important feature is document manipulation on the fly oud dynamic traversal of DOM objects. Web API is enormously useful for building an automated user interface for event triggering so as to click mouse, or post forms, and so on.

4.4.2 System API

System API provides execution environment of PhantomJS scripts. Functionalities of system module are OS information, environment variables, command-line arguments, and process-related properties. DSL or more fine grained operation of PhantomJS apps require system API.

4.4.3 File System API

File system API provide functionality ranging from Accessing files, writing to text files to just reading a custom configuration file 窶付 hese. Basic file I/O such as read, write and delete files are performed by file system API.

5. CasperJS

With the rising popularity of phantomJS, several projects have been started for evolving phantomJS functionality. CasperJS is an open source extension of PhantomJS for improving tha scripting ability of PhantomJS 図擦 asperJS mainly focuses on tasks such as web scraping, testing, and DOM manipulation. Figure illustrates the difference between phantomJS and CasperJS 図燦 hile CasperJS depends on phantomJS, CasperJS provides a new set of API functions for step-by-step coding approach, which is known as navigation scripting.

5.1 XPath Selector

Usually headless browser uses CSS3 selector rather than Xpath selectors. while xpath selectors are less readable than CSS. However, XPath selector takes advantages in some points such as matching text contents or putting conditions on the DOM element 窶洒 ascendants or descendants. To use XPath selectors, we should load selectXPath utility and use re syntax as follows:

```
1 1 var x = require('␣casper'). selectXPath;
2 2 ... test.assertExists( x("//*[␣contains(␣text(),'Tokyo
      ␣(koganei)')]"), '
```

5.2 Callback mechanism - timing

Developing JavaScript requires building two pieces of code such as loading JSON and updating page content with it. Basically these two blocks are non-blocking which results in that the rest of the code keeps executing.

6. Experiment

In experiment, we prototyped CasperJS script which is generated by simple DSL (Domain Specific Language) mainly focusing on enumerating loop for apps list.

6.1 Apps list

Our prototype targeted a popular and relatively small APK distribution site of www.freewarelovers.com which stores 1197 applications. List below shows brief structure summary of the site. Once you click the category page, you will access applist page each of which item is linked download page. We deployed our prototype into Amazon VPC(virtual pri- vate cloud) with several Micro instances. We compiled our sys- tem on ubuntu12 LTS with Linux kernel 3.2.0. proposed system is hosted on Intel Xeon E5645 with 2.4 GHZ clock

```
1  this.click(x("//*[\@id=\"fieldset\"]/table/tbody/tr
2  [2]/td[1]/p[1]/a[1]")),215
3  3 this.click(x("//*[\@id=\"fieldset\"]/table/tbody/
       tr
4  [2]/td[1]/p[1]/a[2]")),255
5  4 this.click(x("//*[\@id=\"fieldset\"]/table/tbody/
       tr
6  [2]/td[1]/p[1]/a[3]")),62
7  5 this.click(x("//*[\@id=\"fieldset\"]/table/tbody/
       tr
8  [2]/td[1]/p[1]/a[4]")),175
9  6 this.click(x("//*[\@id=\"fieldset\"]/table/tbody/
       tr
10 [2]/td[1]/p[1]/a[5]")),91
```

6.2 Loop generation

We have implemented tiny DSL for generating cascade to crawl XPATH of apps list. List below shows XPATH representation for each APK item. Here counten1 is one to increment and counter2 is upper limit.

```
1  this.click(x("//*[\@id=\"fieldset\"]/table/tbody/tr
       [2]/td[1]/p[1]/a[$counter1]")),$counter2
```

```
1  for($counter=1;$counter<$item;$counter++) {
2  print "casper.waitFor(function␣check()␣{␣\n";
3  print "竦 returnfthis.click(x(";
4  print "\"";
5  print "//*[\@id=";
6  print "\\";
7  print "\"fieldset";
8  print "\\";
9  print "\"]";
10 print "/table/tbody/tr[1]/td[3]/table/tbody/tr[";
11
12 print $counter."]/td/p/b/a\"))␣!=0;␣\n";
13  print "},";
14
15 print "竦 functionfthen()␣{␣\n";
16 print "console.log($category␣+␣\",\"+
17 $counter␣+␣\",\"+this.
18 getCurrentUrl());\n";
19 print "},\n";
20 print "竦 functionftimeout()竦f{\n";
21 print "this.echo(\"timeout\");\n";
22 print "});\n";
```

The are two ways for sending mouse click event: casper.mouse.click() and casper.click(). casper.click() just handles a selector as a parameter. On the other hand, casper.mouse.click() processes a selector or (x,y) position. While casper.click() directly generates an event and allocate slot to the target event, casper.mouse.click() does not handle any element and just produces a mouse event at the given position.

6.3 Handling timeout

Handling timeout requires process monitoring entity outside scripting framework. GNU coreutils and posix signal are provided, however, these are not available for headless browser scripting framework. The solid solution for handling timeout is using the the ALRM signal.

```
1  for($counter=1;$counter<$item;$counter++){
2  $TIMEOUT = 10;
3  eval {
4  local $SIGfALRMg = sub fdieg;
```

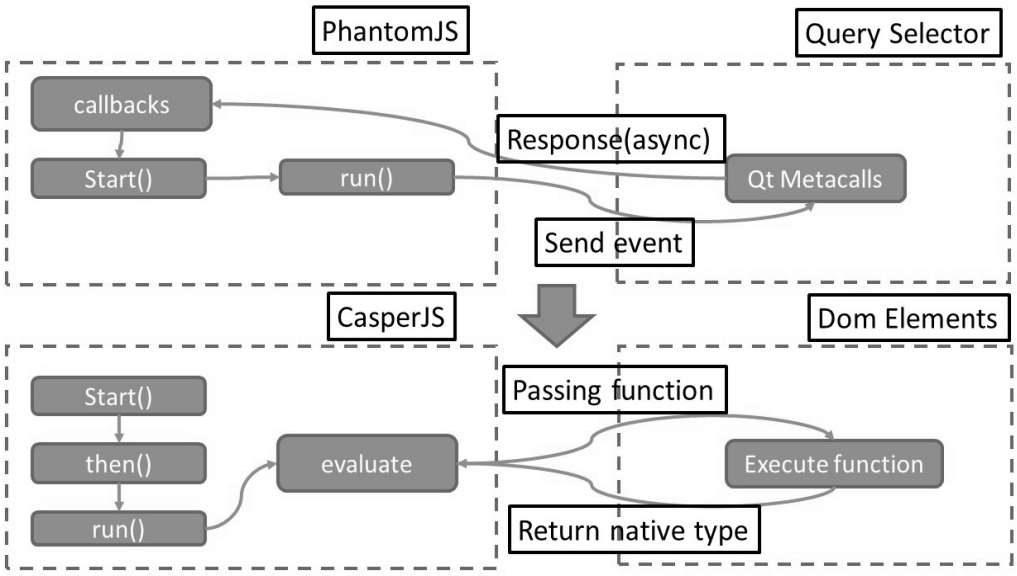

Fig. 3 CasperJS architecture.

```
5  8 alarm($TIMEOUT);
6
7  $str = "/home/ubuntu/casperjs/bin/竦竦
8  casperjsftestf"
9
10 $pid = fork;
11 if ($pid == 0) {
12 exec($str);
13 }
14 else f
15 wait;
16 }
17  my $timeleft = alarm(0);
18 }
19
20 if ($@) f{
21 # timeoit
22 kill(SIGKILL, $pid);
23 print "\nERROR:␣TIMEOUT\n";
24 elsef
25 print "Hello!␣$name";
```

In experiment, we have retrieved most of APKs from the site with reasonable computing time.

6.4 Results

In this section we show brief summary of mesurement results. Figure shows depicts the frequency over monitored over 100000 APKs. As is often with other distributions concerning malware, it seems that our plots are corresponding to zips distribution.

These two tables show the number of occurrence ot URL and coutries. in this paper we simply adopts static analysis, therefore we cannot expose root-cause of URL ranking. However, many anomaly behaviors have been found in the long-tail part of destination shown in Figure ☒撒 able shows country ranking of destination. It is turned out that Russia and China appear more freauently than expected before the measurement.

Table 1 URL ranking

url	occurrence
apps.opera.com	26400
www.w3.org	20622
ns.adobe.com	13843
purl.org	13153
www.iec.ch	10243
blackberry.apps.opera.com	2454
ios.apps.opera.com	2452
vk.com	2205
iptc.org	1006
www.google.com	871
html5shim.googlecode.com	785
code.google.com	503
www.youtube.com	394
vkontakte.ru	368

Table 2 country ranking

country	occurrence
US	60107
EU	43585
CH	20017
RU	5231
AT	1100
NL	958
DE	820
CA	584
GB	398
FR	236
JP	214
SG	202
AP	174
CN	153
KR	115

7. Conclusion

We have presented the empirical study of Android APK disributuion sites using headless browser with navigation scripting. With rising popularization of Android application, Android has become attractive target for malware deveplopers. The thrust

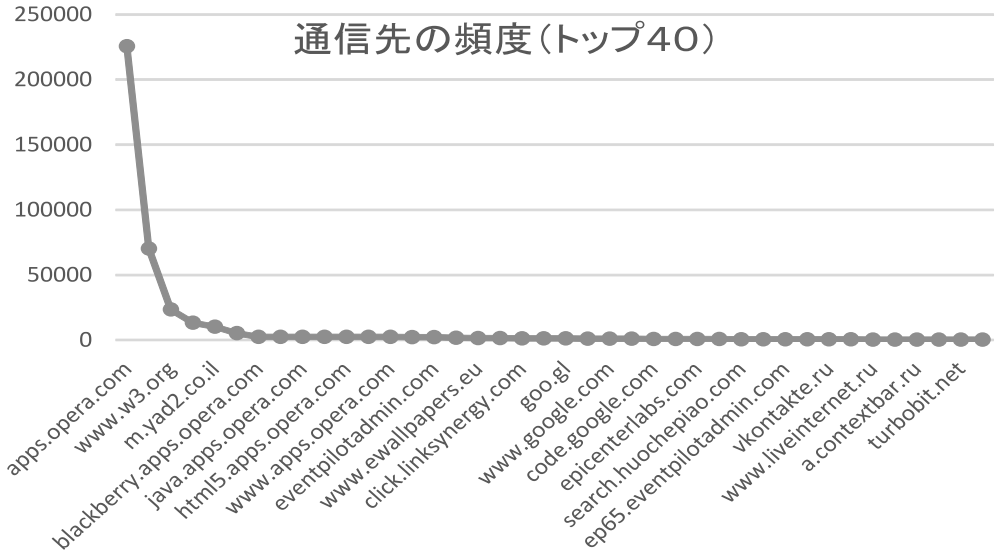

Fig. 4 URL ranking of measurement results.

ofNavigation scripting with JavaScript enables more interactive web page crawling in order to fetch the results after dynamic web page loading. We present the implementation detail of navigation scripting for gathering information of unofficial APK sites. In experiment, we show statistical analysis of current situation of destination address of APK communications.

References

[1] Lok-Kwong Yan and Heng Yin. DroidScope: Seamlessly reconstructing os and dalvik semantic views for dynamic android malware analysis. In Proceedings of the 21st USENIX Security Symposium, August 2012.

[2] Landon P. Cox, Peter Gilbert, Geoffrey Lawler, Valentin Pistol, Ali Razeen, BiWu, Sai Cheemalapati: SpanDex: Secure Password Tracking for Android. USENIX Security 2014: 481-494

[3] SmallTalk http://www.smalltalk.org/versions/ANSIStandardSmalltalk.html

[4] Durandal testing http://durandaljs.com/documentation/Testing-With-PhantomJS-And-Jasmine.html

[5] Andrei Costin, Jonas Zaddach, Aurlien Francillon, Davide Balzarotti: A Large-Scale Analysis of the Security of Embedded Firmwares. USENIX Security 2014: 95-110

[6] William Enck, Damien Octeau, Patrick McDaniel, Swarat Chaudhuri: A Study of Android Application Security. USENIX Security Symposium 2011