ICFlash: Web-based Network Measurement Using Adobe AIR

CSE 534 Midterm Research Paper

Jian Xu[1], Benjamin X. Lin[1], Yang Sheng Fang[1]

[1]Department of Computer Science, Stony Brook University,

Stony Brook, NY 11794-4400

{jianxu1, xianlin, yafang}@cs.stonybrook.edu

http://www.ic.sunysb.edu/stu/xianlin/~cse534

ABSTRACT

In order to help ICLab[1] collect data for censorship related researches, we develop ICFlash, a network measurement application embedded on a webpage. In specific, ICFlash allows an end user to issue HTTP GET requests to a remote web server to get the HTTP responses and to execute DNS queries. We implement ICFlash by using Adobe AIR, after investigating the feasibility of HTML5, JavaScript, PHP, Adobe Flash and AIR, since AIR does not have the cross-domain restriction. We describe the implementation details, provide the user manual, and outline our next step.

Key Words: Cross-Domain, Adobe Flash, Adobe AIR, HTTP, DNS

1. INTRODUCTION

Currently, the ICLab[1] application requires an end user to download the Centinel Client and run it locally to collect enough information for research purposes. These complex configurations and installation steps may limit the number of users who are willing to participate in this censorship data collection program, because some users would be reluctant to download unknown software applications from the web.

We create an Adobe AIR application named ICFlash for web browsers, enabling end users to help ICLab collect HTTP and DNS data by simply typing URL links and clicking buttons. In order to implement these functions, ICFlash first issues an HTTP GET request (or DNS query) to the web server designated by the URL; then it sends the HTTP (or DNS) response to Centinel Servers.

In the rest of this paper, we firstly review the background (**Section 2**) technologies that may be used for ICFlash and briefly describe why we reach our decision of using Adobe AIR. Next, we introduce the design of our application (**Section 3**), followed by its implementation (**Section 4**). We present the prototype of ICFlash in **Section 5**, discuss the next step after this midterm report in **Section 6**, and conclude in **Section 7**.

2. BACKGROUND

In this section, we review several web technologies for their feasibility in ICFlash.

2.1 HTML5

HTML5 is a core technology markup language of the Internet used for structuring and presenting content for the World Wide Web. It is derived from HTML4, which was standardized in 1997. Comparing to HTML4, a great improvement of HTML5 is the support of the latest multimedia technology[2].

Some new features were designed to make it easy to include and handle multimedia and graphical content on the web without having to resort to proprietary plugins and APIs. Additionally, several technologies were applied in HTML5 to support specific features, like WebSocket API, to provide full-duplex communication channels over a single TCP connection[3]. So we can apply Websocket API to issue HTTP requests to servers.

While in this case in order to process successfully, servers that accept the WebSocket protocol requests have to support the protocol by upgrading. It is hardly practical in our project since we have to urge all the web servers to upgrade to support Websocket protocol.

Therefore, applying HTML5 with Websocket API is not practical in our project.

2.2 JavaScript

JavaScript is a language used on all modern web browsers. It is usually used in conjunction with HTML and CSS. It allows client-side scripts to interact with the user, control the browser, communicate asynchronously, and alter the document content that is displayed[4].

Due to security issues, the browser enforces cross-domain policy as well. This prevents a third-party website from accessing a user's authentication information on another website. The browser does not provide stream accesses (e.g. TCP, UDP, etc.) or DNS requests, either. Therefore, this is not suitable for our application.

2.3 PHP

PHP is a server-side scripting language designed for web development, in which case PHP generally runs on a web server (e.g. Apache, Nginx, etc.)

PHP code can be mixed with HTML code to construct a dynamic webpage. It is usually processed by a PHP

interpreter, which is usually implemented on web server side by a server native module or a Common Gateway Interface (CGI) executable. After the PHP code is interpreted and executed, the web server sends the resulting output to its client, usually in form of a part of the generated web page; for example, PHP code can generate a web page's HTML code, an image, or some other data[5].

According to the fact that PHP is a server-side scripting language and can only be executed on the server side that makes us have to execute our program remotely. Therefore, servers instead of browsers will send the DNS/HTTP requests that we embed in PHP programs. Results will reflect the accessibility in servers' perspective instead of users which is not we intend to measure.

Alternatively we can achieve our goal by deploying LAMP-like environment in client-side (e.g. LAMP[6], MAMP[7], WAMP[8]) where to let PHP scripts execute on, but this approach breaches our principle that do not install heavy packages and complex configurations on the client side.

As a consequence, PHP is not an ideal method for this project.

2.4 Adobe Flash

Adobe Flash is a common user-side presentation platform in modern web applications. Its common usage includes video player, animation, website navigation, and complex application (such as image editor). It supports cross-platforms running on Windows, Mac OS, and Linux. It not only has a wide number of platform supports, but also gains a huge user base.

Flash application is written in ActionScript with various front-end technologies. The source code is compiled into a Shockwave Flash file (i.e. SWF file).

SWF files can be embedded in web pages, and they must be running on a system with Adobe Flash installed.

Flash provides certain APIs that do not exist in browsers. These APIs include TCP streams and custom TCP requests, which are crucial to our application implementation.

However, Flash enforces certain security policies. One of such is the cross-domain policy[9]. The target server which the Flash application tries to access, must grant permission to the domain where the Flash application resides, by including a "crossdomain.xml" file on its own server. For example, if a Flash application on *www.a.com* wants to access *www.b.com*, *www.b.com* must include "crossdomain.xml" on its root that grants permission to *www.a.com*. This policy demonstrates a huge restriction on what we want to achieve, which is to send requests to any domain. And, it is impractical for us to ask web administrators to grant our application access to their web servers. Therefore, we opt not to use Flash.

2.5 Adobe AIR

Adobe AIR supports the full Adobe Flash API so that it gains all the advantages of Flash. In addition, AIR provides an API named *flash.net.dns* for DNS queries and other APIs support file system integration and accesses to connected devices. The AIR runtime supports installable applications on Windows, OS X and mobile operating systems like Android, iOS and BlackBerry Tablet OS[10].

Another advantage of Adobe AIR is allowing Adobe Flash and ActionScript code to construct applications that run outside a web browser, and behaves like a native application on supported platforms[10].

The most important point is that AIR does not have the cross-domain restriction. Consequently it allows us to send HTTP requests to arbitrary servers[10].

According to the investigations on HTML5, JavaScript, PHP, Flash and AIR technologies, we find that AIR is a suitable platform for our application because the other technologies have the cross-domain restriction and lack APIs accessing to low level streams.

We decide to implement ICFlash using AIR as it does not have those drawbacks mentioned above, even though it requires users to install the AIR runtime environment onto their local computers.

3. DESIGN

Figure 1 presents an overview of ICFlash's workflow.

Initially, ICFlash is downloaded from our web server, and installed on a user's local computer. Next, ICFlash gets instructions from the user to send an HTTP or DNS request. The destination server does not need to grant ICFlash any special permission to send a request. After that, ICFlash records the response from the server. If the request is an HTTP GET, the result includes HTTP response status code, response URL and the name-value pairs of the HTTP header. If the request is a DNS query, the result will contain one or more IP addresses for the requested domain name. In both cases ICFlash records the round-trip time. In case a response is not received, ICFlash will report an error to the user. Finally, ICFlash sends response data to ICLab Centinel Servers for further analysis.

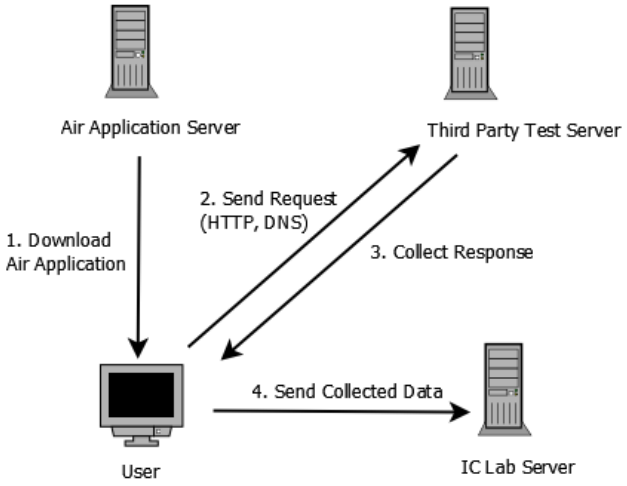

Figure 1. Interaction between ICFlash and other systems

4. IMPLEMENTATION

4.1 Library

In order to implement the function of sending HTTP requests, we use the *flash.net* API[11] from the Flash library. We first build a *URLRequest* object, which allows us to construct an HTTP request with GET method and "text/html" content type. Then by using an instance of *URLLoader*, our application sends an HTTP request to a remote server. An event handler is used to asynchronously process the response. We extract the data and then present it.

4.2 Embedding Air into a Web Page

We compile our code into an AIR installer and then deploy it to our current server. We use a collection of sample html files named "Badge" from the official Adobe site[12]. Then we replace the AIR file with ours, and modify the AIR version and the absolute URL to our AIR file.

Then we upload the whole folder containing "Badge" files to our server, allowing end users to test our application.

5. PROTOTYPE

5.1 User Interface

The user interface of ICFlash looks like *Figure 2*.

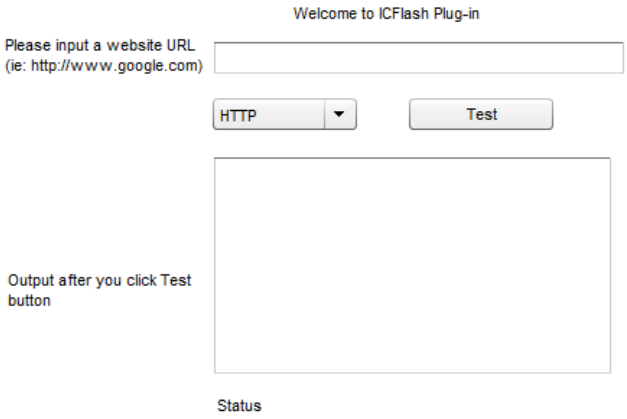

Figure 2. The user interface of ICFlash

5.2 User Manual

Step 1: The user needs to visit the webpage with ICFlash embedded, as shown in *Figure 3*.

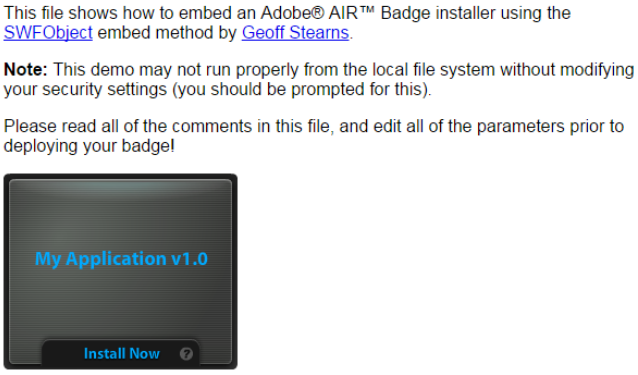

Figure 3. The initial webpage to start ICFlash installation

Step 2: The user clicks the "Install Now" button. ICFlash asks the user whether or not he wants to open and run it as shown in *Figure 4*.

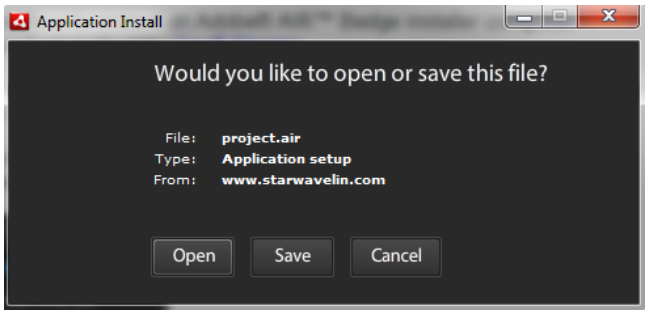

Figure 4. Installation Prompt

Step 3: The user will see the warning message in *Figure 5*. By clicking "Install" button, he confirms his decision of installing.

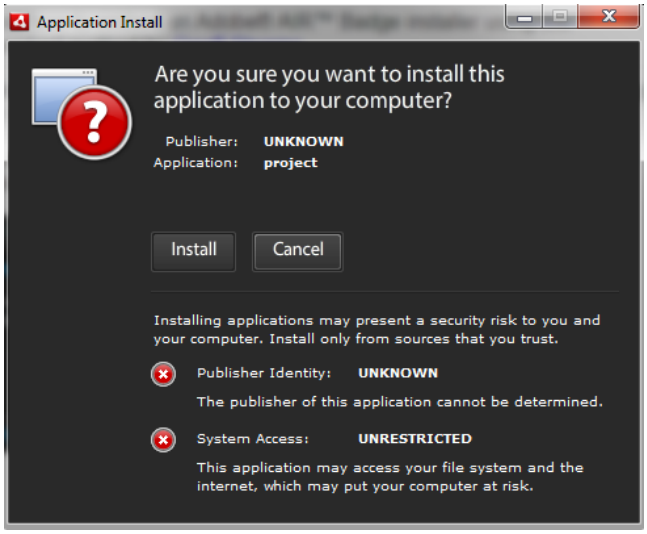

Figure 5. Request uer confirmation

Step 4: Then ICFlash asks the location where he wants to install it as shown in *Figure 6*.

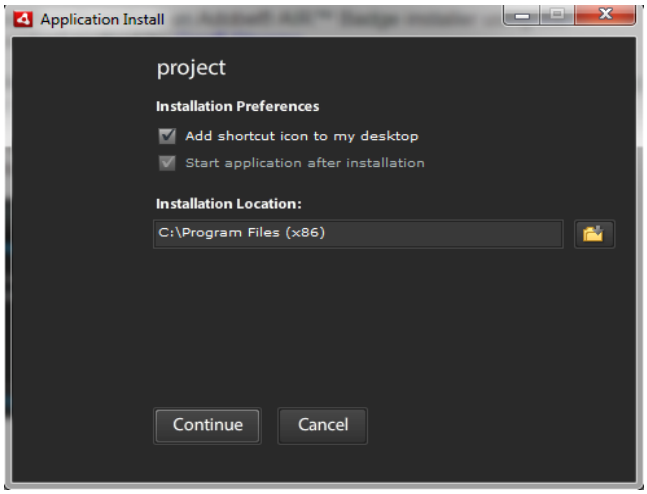

Figure 6. Installation Path

Step 5: By clicking the "Continue" button, the installation of ICFlash will start.

After the installation is finished, the interface of ICFlash will pop up, as shown in *Figure 2*.

In the text input line next to the label "Please input a website URL", the user can input any site URL without the protocol prefix (e.g. https://). Then, he needs to use the dropdown menu to select either HTTP or DNS feature for testing. By clicking the "Test" button, the results will be displayed in the large output textbox.

5.3 Result

5.3.1 Result from an HTTP GET Functionality

After the user inputs a correct website URL, selects HTTP in the dropdown menu, and then clicks "Test" button, he will see the HTTP status code, header and body in the output textbox, and the round-trip time (RTT) in the status line at the bottom, as shown *Figure 7*.

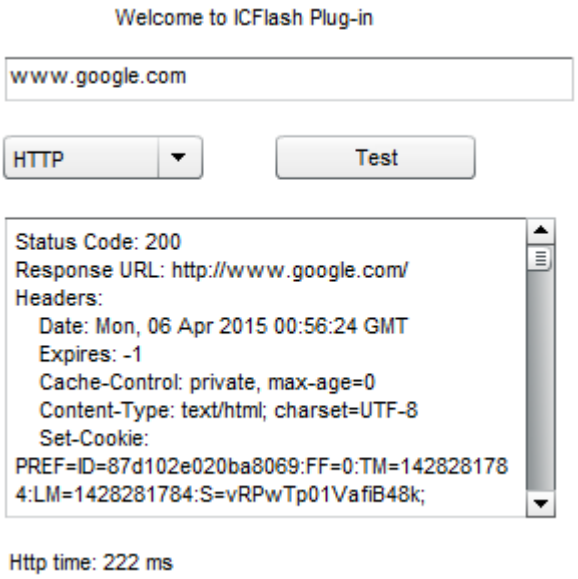

Figure 7. Result from an HTTP GET functionality

5.3.2 Result from a Nonexistent URL Input

This example shows the case when the user inputs a nonexistent URL and clicks "Test" button. He will not see any output in the output box but error information in the status line at the bottom, as shown in *Figure 8*.

Figure 8. Result fron an nonexistent URL input

6. NEXT STEP

Till now, we have overcome the cross-domain restriction and completed the HTTP GET module in ICFlash.

In our next project phase, we plan to complete the DNS query module and finish the features supporting ICFlash connecting to Centinel Servers for their analysis[13]. We will support ICFLash to send results (e.g. DNS query results) acquired to an ICLab server (i.e. Centinel Server) [13].

7. CONCLUSION

In order to get HTTP/DNS information from users' perspective via browsers, we implement a web-based application named ICFLash. In this paper we first investigate the feasibility of several technologies such as HTML5, JavaScript, PHP and Adobe Flash.

Then we eliminate those technologies, as they are not able to satisfy our requirements. For instance, we cannot get user-perspective information from servers executing PHP; since not all servers support WebSocket protocol, it is impossible to deploy a general application which sends WebSocket requests; because JavaScript and Flash are subjected to the cross-domain policy, they are impractical for us to apply, either.

Hence, we decide to use Adobe AIR technology due to its convenient deployment and capability of satisfying our project requirements, for instance, to acquire user-perspective accessibility to web servers.

Next, we describe several key steps on how to realize the project, for example, how to send HTTP GET requests by using Adobe AIR.

We also demonstrate the user manual of how to install and use the ICFlash step by step.

Finally we show our next phase of implementing DNS query feature and the interaction between ICFlash and Centinel Servers.

8. REFERENCES

[1] The Internet Censorship Lab: http://www.internetcensorshiplab.com/

[2] http://en.wikipedia.org/wiki/HTML5

[3] http://en.wikipedia.org/wiki/WebSocket

[4] http://en.wikipedia.org/wiki/JavaScript

[5] http://en.wikipedia.org/wiki/PHP

[6] http://en.wikipedia.org/wiki/LAMP_(software_bundle)

[7] http://en.wikipedia.org/wiki/MAMP

[8] http://en.wikipedia.org/wiki/LAMP_(software_bundle)#WAMP

[9] http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS5b3ccc516d4fbf351e63e3d118a9b90204-7e3f.html

[10] https://en.wikipedia.org/wiki/Adobe_AIR

[11] http://help.adobe.com/en_US/FlashPlatform/reference/actionscript/3/flash/net/package-detail.html

[12] Adobe *Getting started with the custom install badge.* https://www.adobe.com/devnet/air/articles/badge_for_air.html

[13] Discussion of our project scope with Abbas Razzaghpanah, a PhD student participating in the Internet Censorship Lab. Friday, April 03, 2015