

Developing Secure Rich Internet Applications (RIA) for the Enterprise

An Overview of Curl's Security Architecture

by David Kranz, CTO



This whitepaper describes the security architecture of the Curl RTE and Curl language as it affects all interested parties. It is intended to give a high-level overview of what the architecture is and how it can be used. More details are available in the security chapter of the Curl Developer's Guide, which is also supplied with the Curl RTE. This paper does not discuss more general security issues associated with securing servers, protecting data, etc. unless there is a specific Curl feature that is relevant. Although the Curl security architecture was designed with the needs of enterprise applications in mind, it has all the properties needed for secure Internet deployment as well.

Introduction

The Curl RTE is a mobile code platform: it allows a user to run code on his or her machine without necessarily knowing that such execution is happening. A mobile code platform must ensure that a mobile code application cannot modify or steal data that is accessible to the user, either on the user's machine or on the network to which it is connected, without the permission of both the user and the system administrator of the user's machine. This paper provides an overview of the security features in the Curl RTE and how they should be understood by the relevant parties: system administrators for client machines, system administrators of servers, developers, and end-users.

Curl security is based on the following principles:

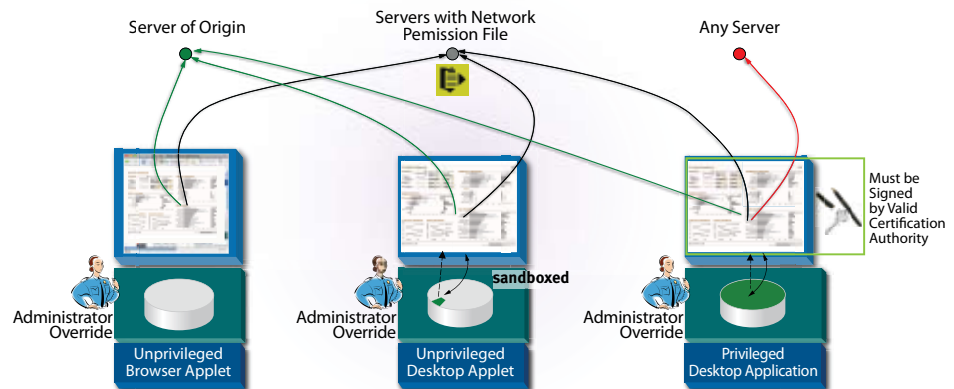
1. In general, end-users do not know how to make security decisions.
2. System administrators must have complete control over the security of client machines under their control.
3. A security model should be as simple as possible, with security taking precedence over functionality where necessary.
4. The security sandbox should have as much functionality as is possible without compromising security.

In order to keep security simple, the Curl RTE avoids fine-grain settings for resources that need to be protected. Except for a small number of cases, an applet runs either with full privilege or in a restricted sandbox. The goal is to make it unnecessary for applets to require privilege in order to meet their specifications.

Curl Security Architecture

The overall security architecture of the Curl RTE is straightforward. A Curl applet or application is loaded from the file system or the network as Curl language source code, which may be compressed and obfuscated. Since the Curl language is type-safe, the Curl compiler can translate it to native code and execute it with no chance of buffer overruns or similar behavior of unsafe pointers. Although byte codes are not used, the compiler performs the same function as the “verifier” does in mobile code systems based on byte codes such as the Java Virtual Machine. When an applet is loaded, if the system administrator has allowed it, the Curl RTE checks if the applet is signed with a valid signature or has been authorized by the user to run with privilege. If so, the applet runs with full privileges and otherwise runs in the secure sandbox.

In an unprivileged applet, access to resources such as the file system and network is controlled by the Curl RTE through the APIs that applets use to access those resources. The Curl RTE supports applets that run inside browsers as well as applications that run in their own application windows and may be activated from, for example on Windows, the Start Menu or a desktop shortcut. The security architecture is exactly the same in either case.



Curl deployment and security models

Security for End Users

For end-users, Curl security is mostly invisible, with very little security-related user-interface. There are no sandbox access-rules, and few security-related dialogs or windows. Also, nothing potentially unsafe is allowed until the user explicitly allows it.

Most end-users do not understand security principles or computer architecture and networking. Unfortunately, most end-users are also the system administrators for their own machines and networks. It is therefore important that users not be asked security questions that they do not know how to answer properly. Out-of-the-box behavior of the Curl RTE is very conservative with respect to security. By default, a Curl applet has no potentially unsafe access to either the host machine resources or the network. Access to servers on the network is controlled by the administrators of those servers.

Experience has shown that too many people say “yes” to security questions when they are not sure of all the implications. If security questions need to be asked, they should be asked in a way that users will likely give the right answer based on a general understanding of using a computer. For example, the Curl language has an API that allows an applet to write to a file chosen by the user. Since this feature is designed to look like an ordinary Save File dialog, the user will understand that they are granting permission to write to a file even if they are not thinking about mobile code security. Aside from a few cases like the one just described, the only security decision an end-user can make is whether to allow an applet to run with privilege. If allowed by the user’s system administrator, such privilege can be granted by using signed applets or settings in the Control Panel.

Security for System Administrators

There are two kinds of administrators that need to know about Curl security:

1. Administrators of servers that want to allow access to unprivileged Curl applets.
2. Administrators of client machines that have the Curl RTE installed.

In order to be useful, most applets coming from a server on the network will require some kind of network access. Server administrators can allow access to unprivileged Curl applets by putting a “curl-access” file on the server. This file specifies which applets will have access, and what kind of access, based on the domain from which the applet was loaded. Note that this mechanism is not a real server security mechanism in that it does not prevent access to the server by a privileged Curl applet, or by any other kind of application. Its purpose is to notify the Curl RTE that the server administrator was granting permission for access by mobile code. Complete details about how to create and deploy curl-access files can be found in the security chapter of the Curl Developer’s Guide.

Administrators of client machines may not trust their end-users to make even the limited security decisions allowed by the Curl RTE, in particular the decision to grant privilege to an applet. In order to exercise tighter control, an administrator may add a policy file to the installed Curl RTE that partially or totally restricts the ability to grant privilege or that restricts even unprivileged execution to applets served from some specified domains. In particular, the following features may be controlled:

1. Whether the user is allowed to grant privilege to applets by either accepting a signed applet or using the control panel to authorize an origin URL.
2. Whether execution should be restricted to a particular set of origin urls.
3. Whether automatic updates of the Curl RTE are allowed.

Security for Developers

Developers of client-server applications, such as web-based or rich Internet applications (RIA), need to make sure their applications are secure. Most of these security considerations involve protecting the servers from unauthorized access and protecting user data. But there are some issues for client mobile code as well. If an applet is going to run with privilege, then from a security standpoint it is no different than a regular native application. If the network infrastructure allows a server to be located and the server allows access from the client running the applet, access will be allowed. Privileged applets also have access to local files, the registry, etc. Obviously, developers of privileged applications need to be careful when accessing host or network resources.

Developers of unprivileged applets do not need to worry about their applets damaging the host machine or stealing data from the network. But they still have to worry about protecting user data. Sensitive data being sent to or from a server on an open network should always use https, whether the applet is privileged or not. Clients should also be authenticated using client certificates or user id/password. Using cookies for authentication has all of the same security issues as with other client-side technologies and must be thought about carefully. Uses of “evaluate” should also be scrutinized carefully. Unlike with AJAX, use of “evaluate” is not necessary to gain acceptable performance while parsing returned data. Next we will mention some of the APIs that are disallowed in the secure sandbox and those that allow partial access. Many more details can be found in the security chapter of the Curl Developer’s Guide.

System Restrictions

The Curl language places a number of restrictions on applet's access to general system resources. Unprivileged applets cannot

- Include ActiveX controls
- Spawn host processes
- Access environment variables
- Access system preferences
- Access Windows registry entries
- Use more than 1GB of memory

These restrictions prevent applets from circumventing the Curl language's other security restrictions by calling code that is outside of the Curl RTE's control. They also prevent applets from trying to destabilize the system by using up too many system resources.

Network Access Restrictions

The following networking APIs are restricted:

- Applets cannot access network urls unless an appropriate curl-access file is present or a dialog was used to get the user's permission
- Applets can open sockets only to the server that served them
- Applets cannot accept network connections

Allowing applets unrestricted access to servers could cause problems when applets are run within a local network that is protected by a firewall. If an applet is run on a computer within the firewall, the applet could have access to information stored on internal servers that is not supposed to be made public. It would be possible for a malicious applet to access a sensitive file from an internal server and transmit it outside of the local network.

Curl applets can ask the user to select a file to access on a server by using the "choose-location" API. This API opens a Choose Location dialog that prompts the user to enter the URL of a file he or she wants the applet to access. Since the user selects the file the applet can access, he or she decides what information is safe for the applet to access.

Applets can "silently" access files (without having to ask the user) on servers that have granted the applet access via a curl-access.txt file. Servers that do not contain a curl-access.txt cannot be accessed (except for images, as explained below). Servers that do contain a curl-access.txt can be accessed by applets if the directives contained in the file grant the applet access. IP address "pinning" is used to avoid DNS spoofing attacks. The Curl RTE will also honor crossdomain.xml files, which are the Flash-clone of Curl's curl-access.txt files.

The only exception to the restriction on silent access to servers is the "image" API that is used to display an image file. This API is similar to the IMG tag in HTML in that it will cause an image to be displayed, but the applet cannot access the actual bits of the data contained within the image file. Therefore, applets can use the image API to display images from any server, even if the sites do not have a curl-access.txt file in the same way that an HTML page can load images from any site, not just those from the same origin.

File Access Restrictions

The following restrictions are placed on local file access from unprivileged applets:

- Applets can only read from or write to files in the local file system that the user selects
- Applets cannot list directories in the file system, or use methods or procedures that return a path within the file system
- Applets can use persistent data to store information locally without asking the user
- Only 20 network connections or files can be open at a time

Applets are not allowed to access the local file system without permission. This restriction prevents them from stealing data or corrupting system files. An applet can access a file by raising the “choose-file” or “choose-multiple-files” dialogs to ask the user for a file to access. The applet is able to access a file or files that the user selects. The type of access (read, alter, or create a new file) is based on a parameter passed to the choose procedure. The dialog that appears to the user clearly states the type of access the applet will have to the file.

If an applet needs to save small amounts of data, such as user preferences, history, or state information, it can use persistent data rather than asking the user for permission to write a file. Persistent data lets an applet silently store small amounts of information on the user’s system that are accessible to only that applet.

Occasionally Connected Computing

The Curl platform supports the concept of Occasionally Connected Computing (OCC). This means that an applet can be deployed from a server for offline use. Applets that may run offline generally need more storage than is provided by persistent data. When an end-user gives permission to install an applet for offline use, they are also giving permission to store a larger amount of data on the local machine. An applet that has been installed for offline use retains the network “identity” of the server from which it was loaded. That identity is used to determine whether an applet will run with privilege and whether access to a server will be allowed through the curl-access mechanism. This means that when developing an applet for online or offline use, the developer does not have to worry about different security rules. They just have to take care that the applet does not assume it is always connected to the network.

Applets that are not being deployed for offline use may also receive permission to access more local storage by calling the “request-local-data-permission” API. This API will raise a dialog asking the user for permission to use more local storage.

Summary

The security architecture of the Curl RTE is very secure and easy to understand. It is designed to allow a great deal of application functionality while operating under those constraints. It can be used easily by system administrators, developers and end-users. We believe it compares favorably to other mobile code security architectures.



*Curl, Incorporated • 1 Cambridge Center, 10th Floor • Cambridge, MA 02142-1612
tel: 617-761-1200 • fax: 617-761-1203 • info@curl.com*

Curl and the Curl logo are registered trademarks of Sumisho Computer Systems Corporation in the United States. All other trademarks are the property of their respective owners. © 2008 Curl, Incorporated. All rights reserved.