



Macromedia Flash® Player 7 Client-Side Security

December 2004

Macromedia Flash Platform Security Documentation

This white paper is part of a series of complementary documentation resources aimed at informing users about the Macromedia Flash Player security model and educating developers on the best practices and patterns for building secure applications for the Flash platform.

Macromedia's Commitment to Security

Macromedia takes security very seriously and continues to assess, plan, and develop means to ensure that Macromedia Flash technology protects users' security and privacy. This document discusses security and privacy concerns and how they are addressed by the Flash Player security model. The intended audience is IT professionals, managers, and decision-makers who may find this white paper helpful for making informed decisions about the technology choices available when striving to keep their environments secure and safe for users.

Macromedia has worked with, and continues to work with, numerous independent security research groups and security advocates in the industry—such as @stake, Agorics, TruSecure, and CERT—to address issues with our products proactively. Macromedia considers any issue related to security to be of the highest priority and addresses them as such. More information on our security policy, our process for addressing security issues, and our partners can be found on the Macromedia Security Topic Center at www.macromedia.com/devnet/security.

Copyright ©2004 Macromedia, Inc. All rights reserved.

The information contained in this document represents the current view of Macromedia on the issue discussed as of the date of publication. Because Macromedia must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Macromedia, and Macromedia cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for information purposes only. MACROMEDIA MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Macromedia may have patents, patent applications, trademark, copyright or other intellectual property rights covering the subject matter of this document. Except as expressly provided in any written license agreement from Macromedia, the furnishing of this document does not give you any license to these patents, trademarks, copyrights or other intellectual property.

Macromedia, the Macromedia logo, ColdFusion, Flex, and Flash are either trademarks or registered trademarks of Macromedia, Inc. in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Macromedia, Inc.
600 Townsend Street, Suite 500
San Francisco, CA 94103
415-252-2000

Contents

Client-Side Security	1
What Are We Attempting to Secure?.....	2
Case 1: Unauthorized Access to Host System Resources	2
Flash Player provides only limited access to specific resources.	2
Flash Player monitors memory usage and limits disk storage by default.....	3
Flash Player provides strictly controlled access to other Flash- based applications.	4
Case 2: Unauthorized Access to Data.....	5
Flash Player secures data stored on the client by Flash content and applications.....	5
Flash Player requires servers to allow requests explicitly from other domains.	5
When requested by the developer, Flash Player leverages the browser for securing client-server communications.	6
Case 3: Unauthorized Access to End-User Information That Should Remain Private	7
Flash Player does not collect information about users.	7
Users have control over Flash Player’s behavior when encountering decisions concerning privacy.	7
Flash Player settings adjusted in the browser are stored locally and never transmitted to any web server.....	8
Appendix A: Flash Player 7 Security Sandbox Model	9
Appendix B: Stand-alone Flash Player Security Model	12
Appendix C: Additional Resources	13
Security	13
Flash Player	13
Product Centers	13

Macromedia Flash Player is a software product created and distributed by Macromedia and designed to be a cross-platform presentation layer for content and applications. It is installed as an additional component to web browsers, such as Microsoft Internet Explorer, Mozilla Firefox, and Apple Safari. Flash content is delivered as a series of instructions in binary format to the player over web protocols in the precisely described SWF (.swf) file format. SWF files are typically hosted on a server and then downloaded to and displayed on the client when requested.

Most content consists primarily of binary ActionScript instructions. ActionScript itself is based on the ECMA-262 standard scripting language and features APIs designed for creating and manipulating client-side user interface elements and working with data. Flash Player provides a controlled set of constructs which are designed to be safe and powerful. The complete environment reference is available on the Macromedia LiveDocs site at <http://livedocs.macromedia.com>.

A general understanding of the Flash Player 7 architecture is a prerequisite for accurately interpreting the information in this white paper.

Client-Side Security

Macromedia Flash Player uses client-side technology that is found on more than half of a billion computers worldwide (*Quarterly Flash Player Penetration Study*, NPD Research: www.macromedia.com/software/player_census/flashplayer). The broad availability of Flash Player, its ease of deployment and management, and its sophisticated capabilities make it a compelling foundation for content focused on entertainment or marketing, as well as Internet or intranet applications.

As a result, content created for Flash Player can be found on servers across the web and on private business networks. With such wide software distribution, it is critical to understand the security model employed by Flash Player. This security model has been highly successful at guaranteeing a great degree of safety. Because it shields complexity from developers, it is also easy to implement.

The purpose of this white paper is to explain the client-side security model of Macromedia Flash Player. More specifically, it outlines how that model ensures security from malicious servers hosting Flash content which cannot necessarily be trusted. In other words, this paper answers the hypothetical question: “Are Flash Player users, and the information on their computers, secure from attackers?”

What Are We Attempting to Secure?

The first question to answer is what we are attempting to secure. There are three specific breaches of security on the client side, against which Flash Player specifically protects users when they view Flash content hosted on a web server:

- 1 Unauthorized access to host system resources.** This includes gaining control of applications, devices, or resources attached to the system for the purposes of disabling or denying/redirecting access to those resources (buffer overruns, denial of service attacks).
- 2 Unauthorized access to data.** This data could be on local disks, networked disks, or web servers that are communicated over the network or stored in memory by an application or process (password lists, address books, privileged documents, application code).
- 3 Unauthorized access to end-user information that should remain private.** This includes personal and financial data, among other information that might be on the end user’s machine. This also includes information about the end user’s security settings for Flash Player.

Case 1: Unauthorized Access to Host System Resources

Flash Player provides only limited access to specific resources.

A fundamental, yet subtle aspect of the Flash Player security model is the extensive list of what Flash content *cannot* do. The functionality available to developers is a small, controlled subset of the functionality that could *potentially* be exposed by virtue of the system architectures on which Flash Player is built. For example, Flash Player does not allow content to allocate its own memory, install software, or make changes to the system registry.

These deep-reaching capabilities drive the primary cause of concern about the security of client-side, web-delivered technologies like ActiveX. Content sent by these streamlined delivery mechanisms can potentially be unsafe by virtue of the scope of their instruction set. Digital signing technologies may accurately tell you the originator of such content but they cannot guarantee that the content itself is safe.

By contrast, Flash Player contains a controlled set of objects and operations that are predominantly exclusive constructs within the Flash execution environment. Even so, a few objects do correspond to system resources available on the client:

- Streams from cameras and microphones attached to the system can be read by Flash Player. Privacy settings give users control over this access, which is disabled by default. These settings can only be accessed on the client by the end user. For more information about access to settings, see the section “Users have control over Flash Player’s behavior when encountering decisions concerning privacy.”
- Text data can be written to the system Clipboard for use in implementing copy and paste operations.

Because the system functionality that can be accessed by Flash Player is limited, the risk of creating content which gains unauthorized access to the host system or resources attached to it is virtually nonexistent.

Flash Player monitors memory usage and limits disk storage by default.

By monitoring its usage of key system resources, Flash Player limits the potential of denial-of-service (DoS) attacks involving disk space and system memory—arguably two of the most critical resources on a computer.

Flash Player safeguards memory by ensuring that a movie cannot slowly consume increasing memory over an indefinite period of time. A movie which is “caught” in an infinite loop is detected and selected for termination by prompting the user of the situation. The resources used by the application are immediately released when it is terminated.

Disk space is conserved through default limits set by Flash Player. Each domain is initially limited to 100K of disk space. Content from that domain can proactively prompt the user to increase disk space allotment. If the application attempts to store data that exceeds the limit, Flash Player automatically raises the prompt. In either case, the disk space limit is maintained until the user gives explicit permission for an increase to the allotment of a particular domain.

Flash Player provides strictly controlled access to other Flash-based applications.

Flash Player does not provide a public interface by which a Flash-based application can communicate outside Flash Player to other non-Flash applications. Thus there is no possibility of a malicious Flash application gaining control over a non-Flash application.

In principle, Flash applications can communicate with one other. However, Flash Player employs the “sandbox” security model to segregate content that originates from different domains. Content can communicate freely within the sandbox and communication across the perimeter of the sandbox is securely guarded. This includes scenarios in which multiple Flash content items are executing within a single instance of Flash Player and in which communication is attempted between two discrete instances of Flash Player.

For example, imagine an HTML page that contains both a Flash application from www.onlinebanking.com and a Flash-based advertisement from www.advertisements.com. Although found in the same HTML page, each instance of Flash content would be executed in a separate sandbox within Flash Player, preventing the advertisement from accessing any data within the banking application.

Developers can use the System.security API to allow Flash applications from different domains to talk to one other; however, each SWF must explicitly consent to these relaxed privileges by naming specific domains whose applications are to receive special privileges.

More detail on the Flash Player security sandbox is available in Appendix A.

Case 2: Unauthorized Access to Data

Flash Player secures data stored on the client by Flash content and applications.

Flash Player provides a mechanism for reading and writing data to a specific location on the client for use in Flash content. This mechanism is akin to browser cookies, except that these persistent “local shared objects” store a larger amount of data than cookies and do so in a binary format on disk. Local shared objects allow content authors to save application state, remember user preferences, or store frequently used data.

Flash Player ensures that non-Flash applications cannot access this data by obfuscating the path to the data on disk so that only Flash Player content can access it. Flash Player accomplishes this by randomly hashing the file path to prevent applications from performing known-location attacks.

In addition, Flash content is able to access stored data only in the domain from which it is hosted; cross-domain access to local shared objects is not allowed. These restrictions are applied as part of the Flash Player security sandbox model. For more information on the security sandbox model, see Appendix A.

Although they are somewhat protected, shared objects, like browser cookies, are not recommended as a means of storing sensitive data.

Flash Player requires servers to allow requests explicitly from other domains.

Flash Player secures access to server data from web-based Flash content by requiring explicit permission for the requesting domain.

You can refer to the domain serving the Flash content as the *accessor domain* and the domain hosting the data that the Flash content requests as the *provider domain*. The user viewing the Flash content over a web connection may be in any other third domain.

Flash Player requires that the provider domain explicitly grant permission to access data to the accessor domain. If no permission is explicitly granted, the load fails. These permissions are specified by a *policy file*, a text file located on the server of the provider domain that explicitly lists the domains which have permission to access data on that server, in locations at or below its own location. Policy files support various levels of granularity in specifying access control:

- Everyone

- A set of subdomains like *.foo.com
- An exact domain
- A set of related URLs like http://foo.com/special/stuff/*
- An exact URL

This aspect of the security model, as well as the required format of policy files, is described in detail in Deneb Meketa's article, "Security Changes in Macromedia Flash Player 7," in the Flash Developer Center at

www.macromedia.com/devnet/mx/flash/articles/fplayer_security.html.

When requested by the developer, Flash Player leverages the browser for securing client-server communications.

Flash Player leverages SSL-capable browsers to communicate by secure HTTP (HTTPS) protocols, protecting data transmission with encryption when employed by the developer.

When Flash content is loaded into Flash Player (and, thus, the browser) with the HTTPS syntax, the communications between SSL-capable clients and servers that are managed by an SSL-capable browser during the session will be two-way encrypted.

There are two cases in which the browser's facilities for encryption cannot be used for safeguarding data communications because the browser is not the middleman for communication:

- Flash content running outside the browser, for example in the stand-alone projector or in Flash Player embedded within another application
- Flash Player communicating directly with XML Socket servers (the browser does not negotiate these communications)

In these cases, you must use alternate facilities for encrypting communications.

Case 3: Unauthorized Access to End-User Information That Should Remain Private

“Privacy” and “security” are essentially the same. Because private information is necessarily sensitive, Flash Player does not make a distinction between the two. Although we call it out here specifically to allay concerns, the same security model which protects secured information protects end-user privacy too. This means that we refrain from collecting information about a user and sending that information to a server without the user’s permission.

Flash Player does not collect information about users.

Flash Player does not collect information about users or send any user information to macromedia.com or any other server.

The general policy of Flash Player when requiring communication with macromedia.com is a “pull” mechanism. This means that Flash Player retrieves the necessary information from the server and performs the operations on the client. This mechanism is employed for the auto-notification and settings management features of Flash Player. For more information on the macromedia.com privacy policy, visit the Macromedia Privacy & Security page at www.macromedia.com/go/privacy.

Users have control over Flash Player’s behavior when encountering decisions concerning privacy.

The Flash Player Settings User Interface and the Settings Manager provide interfaces for users to fine-tune settings for better control over their privacy and security. These options mimic the settings found in the browser’s options and are enhanced to reflect the advanced capabilities of Flash Players. These settings are for preferences regarding the following:

- Local storage of data using the local shared objects mechanism
- Access to cameras and microphones connected to the system
- Notification of updates to Flash Player

Flash Player allows network administrators to control the settings for Flash Player on corporate networks to conform to a sitewide security policy. Using a configuration file that they can install either manually or with a distributed installation process, network administrators can specify default settings to propagate to all installations of Flash Player:

- To access the settings user interface, right-click any Flash content area and choose Settings from the context menu.
- For more precise control over your Flash Player settings, read more about the Settings Manager at www.macromedia.com/go/settingsmanager.

Flash Player settings adjusted in the browser are stored locally and never transmitted to any web server.

As previously mentioned, users can adjust Flash Player settings in their browser. Although it appears that these settings are configured within the context of a web page, in reality Flash Player retrieves the settings locally and only displays them in the apparent context of the web page they are viewing. At no time are the user's settings at risk of being snooped or hijacked.

In the case of the settings user interface ("settings UI")—the small context-sensitive dialog boxes which appear superimposed over context—the player executes the settings application in a visually overlaid, yet functionally sandboxed manner.

The originating application which invokes the dialog box is unable to script against it, thus protecting the user's choice from snooping or spoofing. The originating application is additionally blocked from executing code, further ensuring that it is unable to peer into the context of the settings UI. The settings UI is then able to prompt the user for a security-related decision and securely record the choice for the duration of the session (until the Flash Player instance is terminated).

The Settings Manager—the series of web pages on the Macromedia support site presenting a more detailed interface than the settings UI—is allowed unique access to the location where settings are stored. Macromedia hosts the Settings Manager application and grants it unique privileges associated with the "home domain" when it is downloaded to the client and executed in Flash Player. These privileges cannot be granted to any other domain.

The end user's settings are not exposed to content or malicious users from any domain other than the local machine's zone and macromedia.com.

Appendix A: Flash Player 7 Security Sandbox Model

Sandboxes restrict access to scripting, content, and data and are created on a per-domain basis.

The Flash Player security model is based on the concept of a security “sandbox.” Everything within the perimeter of the sandbox is securely contained. This includes memory allocation and local data storage (shared objects) used by a particular domain and its content. Each sandbox is isolated from the operating system, file system, network, other applications, and other sandbox instances (see Figure 1).

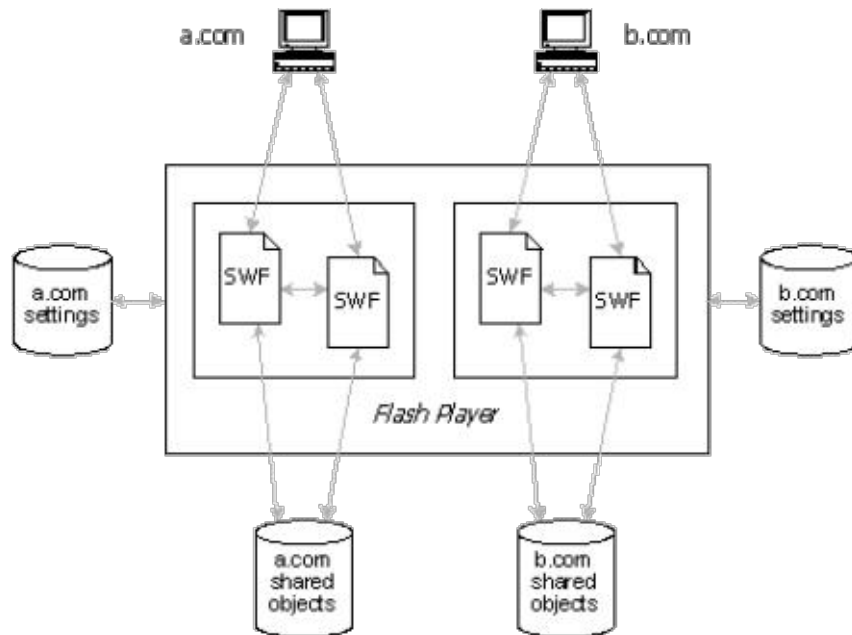


Figure 1: *Flash Player sandbox architecture*

The sandbox model is also employed by web browsers when working with DHTML. However, the Flash Player sandbox is more restrictive than the browser sandbox because the capabilities of Flash Player have no corresponding browser counterparts.

A sandbox is created for each domain which serves SWF content to Flash Player. Additional content and data can be loaded into a particular sandbox, provided that it is loaded from the precise domain as the original content around which the sandbox was created. Content or data from other domains cannot, by default, cross the perimeter of the sandbox.

Put another way, a sandbox is created when Flash Player loads content and determines that it is not part of an existing sandbox. This occurs by default when no other content is being executed. The created sandbox includes the following:

- Original Flash content loaded from a domain
- Additional Flash content or data loaded from the same domain
- User actions directed at the movie
- Local shared objects created by content from the same domain
- Communications with Flash content from the same domain running in other instances of the player
- Limited configuration information about the computer on which the Flash content is running, such as the platform, operating system, and display capabilities

Content from the same domain can share a sandbox, provided that authentication is successful.

Domain authentication is the basis for determining whether content should be placed in a specific sandbox. This authentication is performed by validating a series of requirements to determine whether a matching originating domain is found.

Content from two different servers is placed in the same sandbox only if all of the following requirements are true:

- The domain names consist of at least three tokens, where a token is one part of a domain name delineated by a period
- The server names each have the same number of tokens
- All tokens are the same except for differences in case

If a domain name is expressed in only one or two tokens (for example, “http://foo/” or “http://macromedia.com/”), the domain is in a sandbox by itself. If a domain is identified by its IP address (for example, “10.55.21.218”), its content must also play alone.

Table 1 lists examples of the results of exact domain matching. Italicized comparisons evaluate to distinct domains, each with a separate sandbox.

Table 1: *Exact Domain Matching Results*

Domain 1	Domain 2	Match?
widgets.com	widgets.com	Yes
secure.widgets.com	secure.widgets.com	Yes
<i>widgets.com</i>	<i>apps.widgets.com</i>	No
<i>doodads.com</i>	<i>widgets.com</i>	No
<i>widgets.doodads.com</i>	<i>widgets.com</i>	No

The protocol by which content is loaded also factors into the evaluation of domain matching. Content loaded using a protocol other than HTTPS cannot access content loaded by HTTPS, even from the same domain. This rule is observed in DHTML and Java. The converse is not true; HTTPS content may access content loaded using other protocols from the same domain.

Sandboxes control communication between Flash content and the browser host.

The security sandbox restricts the extent to which Flash Player communicates with other browser client-side scripting technologies. The security restrictions are governed by the domain-based authentication model, whereby the HTML page hosting the client-side script is assigned a sandbox based on its originating domain, which may or may not be the same as that of the Flash content.

If the two are assigned a separate sandbox, the set of API calls available to the client-side script will be strictly limited to read-only operations of the Level0 (root) MovieClip properties of Flash content. These properties include the movie size, frame rate, rotation, host domain, and a number of other properties which are fully documented in the ActionScript API reference for the MovieClip object. See Mike Chambers' white paper, "Macromedia Flash MX Security" at www.macromedia.com/devnet/mx/flash/whitepapers/security.pdf (PDF, 498K).

Appendix B: Stand-alone Flash Player Security Model

In Flash Player 7, the security model for Flash content run from the local file system is slightly different than Flash content viewed on the web. This is because of the different needs of content created for use locally, such as CD-ROMs, kiosks, and similar applications. In general, this security is more lax than that of the web player.

The stand-alone security sandbox allows greater inter-content communication and access to local data.

Sandbox restrictions do not apply to Flash movies and projectors loaded from the local file system, with the following exception: A sandbox from a Flash movie that is not a local file may not access a local file's sandbox. For example, if movie1.swf on a user's local disk loads movie2.swf from an HTTP server, the following is true:

- movie1.swf and movie2.swf are granted separate sandboxes
- movie1.swf may access the contents of movie2.swf
- movie2.swf is barred from accessing the contents of movie1.swf

Appendix C: Additional Resources

Security

Macromedia Flash Player Security Policy Information

www.macromedia.com/software/flashplayer/security

Security Topic Center

www.macromedia.com/devnet/security

Security Zone (Security Bulletins)

www.macromedia.com/devnet/security/security_zone

Flash Player

Flash Player Developer Center

www.macromedia.com/devnet/flashplayer

Flash Player Support Center

www.macromedia.com/support/flashplayer

Product Centers

Macromedia Flex

www.macromedia.com/software/flex

Macromedia Flash MX 2004

www.macromedia.com/software/flash

Macromedia Flash Communication Server

www.macromedia.com/software/flashcom

Macromedia ColdFusion MX

www.macromedia.com/software/coldfusion