

著作権情報

© 2008 Adobe Systems Incorporated. All rights reserved.

AIR のホワイトペーパー

本マニュアルがエンドユーザ使用許諾契約を含むソフトウェアと共に提供される場合、本マニュアルおよびその中に記載されているソフトウェアは、エンドユーザ使用許諾契約にもついて提供されるものであり、当該エンドユーザ使用許諾契約の契約条件に従ってのみ使用または複製することが可能となるものです。当該エンドユーザ使用許諾契約により許可されている場合を除き、本マニュアルのいかなる部分といえども、**Adobe Systems Incorporated**（アドビ システムズ社）の書面による事前の許可なしに、電子的、機械的、録音、その他いかなる形式・手段であれ、複製、検索システムへの保存または伝送を行うことはできません。本マニュアルの内容は、エンドユーザ使用許諾契約を含むソフトウェアと共に提供されていない場合であっても、著作権法により保護されていることにご留意ください。

本マニュアルに記載される内容は、あくまでも参照用としてのみ使用されること、また、なんら予告なしに変更されることを条件として、提供されるものであり、従って、当該情報が、アドビ システムズ社による確約として解釈されてはなりません。アドビ システムズ社は、本マニュアルにおけるいかなる誤りまたは不正確な記述に対しても、いかなる義務や責任を負うものではありません。

新しいアートワークを創作するためにテンプレートとして取り込もうとする既存のアートワークまたは画像は、著作権法により保護されている可能性のあるものであることをご留意ください。当該アートワークまたは画像を新しいアートワークに許可なく取り込んだ場合、著作権者の権利を侵害することがあります。従って、著作権者から必要なすべての許可を必ず取得してください。

例として使用されている会社名は、実在の会社・組織を示すものではありません。

Adobe、Adobe ロゴ、Adobe Studio、ActionScript、After Effects、Dreamweaver、Flash、Flash Player、Flash Video および Soundbooth は、アドビ システムズ社の米国ならびに他の国における登録商標または商標です。

Apple is a trademark of AppleInc., registered in the United States and other countries. All other trademarks are the property of their respective owners.

This product includes software developed by the Apache Software Foundation (<http://www.apache.org/>). MPEG Layer-3 audio compression technology licensed by Fraunhofer IIS and Thomson Multimedia (<http://www.iis.fhg.de/amm/>). You cannot use the MP3 compressed audio within the Software for real time or live broadcasts. If you require an MP3 decoder for real time or live broadcasts, you are responsible for obtaining this MP3 technology license. Portions of this product contain code licensed from Nellymoser (www.nellymoser.com). Flash CS3 video is powered by On2 TrueMotion video technology. © 1992-2005 On2 Technologies, Inc. All Rights Reserved. <http://www.on2.com>. This product includes software developed by the OpenSymphony Group (<http://www.opensymphony.com/>)

Notice to U.S. Government End Users. The Software and Documentation are “Commercial Items,” as that term is defined at 48 C.F.R. §2.101, consisting of “Commercial Computer Software” and “Commercial Computer Software Documentation,” as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

目次

Adobe AIR 1.5 HTML のセキュリティ

RIA のセキュリティ問題	1
サンドボックスによるセキュリティの保護	2
セキュリティの向上	4
セキュリティ：共通の責任	6

Adobe AIR 1.5 HTML のセキュリティ

The Adobe® AIR™ HTML security model is based on developer feedback and an evolving understanding of the security issues inherent in developing HTML and Ajax applications. The development of Rich Internet Applications for AIR presents new challenges because AIR applications provide, unlike Web-based RIAs, access to the desktop. This means that AIR applications can read and write files, draw to the screen, communicate with the network, and so on.

Security is a key concern of Adobe, users, system administrators, and application developers. For this reason, Adobe AIR includes a set of security rules and controls to safeguard the user and application developer. This white paper presents the security considerations in developing HTML-based AIR applications.

For details on AIR security, see the the **Adobe AIR Security** white paper, posted at http://www.adobe.com/go/learn_air_security_wp_en.

RIA のセキュリティ問題

HTML アプリケーションが直面する主な脅威の 1 つは（デスクトップアプリケーションと Web アプリケーションのいずれの場合も）、結果としてクロスサイトスクリプティング（XSS）などの悪意のあるコードが実行されるインジェクション攻撃です。通常、コードは、URL 処理（javascript: やその他の危険なスキーム）、eval()、および innerHTML や outerHTML などの DOM エレメントへの外部 HTML コンテンツの割り当てなど、いくつかの共通する手段によって挿入されます。HTTP を通じて読み込み操作が行われた場合、信頼できるサービス（Google Maps など）に悪意のあるコードが挿入される可能性もあります。

開発者がアプリケーションは安全なコンテンツを利用していると想定しているときに、実際には悪意のあるコードがコンテンツに含まれていると、このような脆弱性が発生します。例えば、オンラインのブログからコメントを抽出して表示する HTML ベースの AIR アプリケーションで、ユーザが次のようなコメントを書き込むことが許可されているとします。

```
<a href="#" onclick="var f=new air.File('c:\\something.txt'); f.deleteFile();">I'm cool!</a>
```

ユーザがこのリンクをクリックすると、ファイルが削除される可能性があります。損害はさらに大きくなる可能性があり、セキュリティの脆弱性はブラウザベースの HTML アプリケーションや Ajax アプリケーションに共通しています。

JSON（JavaScript Object Notation）を通じて読み込まれた悪意のあるコードによる損害を、既存のドメインサンドボックスの制限によって緩和できるという考え方は正しくありません。frame や iframe などの小さい DOM サンドボックス境界内でさえ、すべてのデータが生成元に関係なくサンドボックス内で実行されます。最も小さいフレーム定義の DOM サンドボックス境界も、コンテンツが異なる生成元ドメインに配置されている場合にのみ、正常に機能します。さらに、一般的な設計および実装のパターンによって、このような危険な手法が助長されています。

従来の Ajax アプリケーションの場合、ブラウザ内で実行されるアプリケーションはサンドボックス化されるので、この手法はある程度認められてきました。ユーザのコンピュータから分離され、同じ生成元ポリシーによって制限されるので、ブラウザアプリケーションには固有の対称性があります。

機能が豊富な API にアクセスできるリッチインターネットアプリケーションには、特有の問題が伴います。単純なアプリケーションでも、コーディングが不適切である場合、インジェクション攻撃に対する脆弱性が含まれ、ユーザのコンピュータに多大な損害を与える可能性があります。RIA プラットフォームの場合、悪意のある攻撃による潜在的な損害は、ランタイムが提供する価値に直接比例します。RIA プラットフォーム内の API がより強力になり、機能が豊富になるほど、リスクも大きくなります。

サンドボックスによるセキュリティの保護

このようなリスクを緩和するために、AIR には、AIR アプリケーションの各内部ファイルと外部ファイルの権限を定義する包括的なセキュリティアーキテクチャが用意されています。権限はその生成元に従ってファイルに付与され、サンドボックスと呼ばれる論理的なセキュリティグループに割り当てられます。ランタイムでは、このサンドボックスを使用して、データへのアクセスや実行可能な操作を定義します。

- アプリケーションサンドボックス内のコードは AIR API にアクセスでき、限定的にページの読み込み後に動的にコードを生成できますが、`script` タグを通じてコンテンツを読み込むことはできません。
- その他のすべてのサンドボックス（非アプリケーションサンドボックスと呼ばれる）内のコードには、基本的に、ローカルの信頼された HTML に対する標準的なブラウザと同様の制限と動作が適用されます。

アプリケーションサンドボックス

アプリケーションサンドボックスの主な機能は、AIR API と信頼されたディレクトリ内のファイルとの間でのやり取りを有効にすることです。非アプリケーションサンドボックス内のコードからの分離は、最初のページの読み込みの後で JavaScript パーサを無効にして `eval()` などのコードが例外をスローするようにすることで維持されます。`innerHTML` の設定は許可されていますが、これに含まれるスクリプトはすべて無視されます。これによって、外部で読み込まれたデータに AIR API を直接公開することなく、幅広い機能を提供します。

開発者は、`app:/` URL スキームを使用して、アプリケーションサンドボックス内の信頼されるディレクトリを参照します。`frame` や `iframe` の場合、アプリケーションディレクトリ内のファイルでも、AIR の `sandboxRoot` および `documentRoot` 属性を使用して `frame` や `iframe` に読み込まれると、非アプリケーションサンドボックスに配置されることがあります。AIR インストーラに含まれるすべてのファイルは自動的に `app:/` にインストールされるので、アプリケーションサンドボックスの一部になります。アプリケーションを実行すると、これらのファイルには、ローカルファイルシステムの操作を含め、AIR アプリケーションのすべての権限のセットが付与されます。`app:/` の外部のコンテンツはアプリケーションサンドボックスの一部にはならず、ブラウザ環境で処理する場合と同様に処理されます。

AIR API は、進化したブラウザのセキュリティモデルと、デスクトップと同様のセキュリティモデルとを安全に橋渡しするために、セキュリティで保護された手段を提供します。AIR サンドボックスブリッジモデルによって、標準的なブラウザベースのアプリケーションに比べて、幅広い機能を、低いリスクで有効にすることができます。

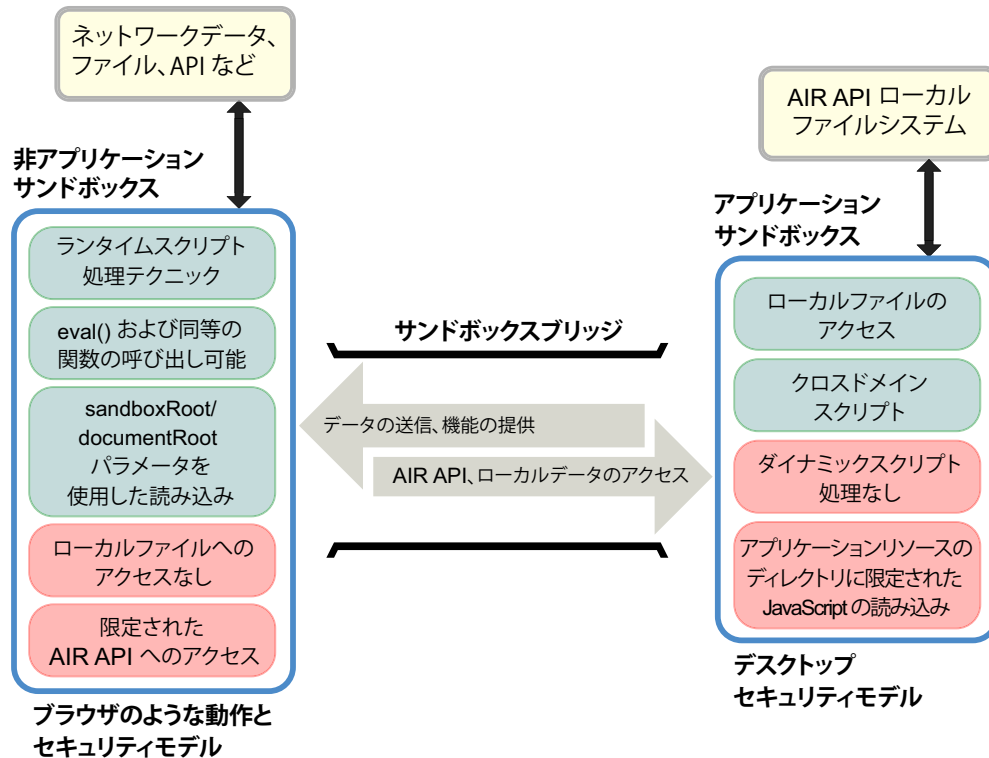
非アプリケーションサンドボックス

非アプリケーションサンドボックスの主な機能は、開発者によって明示的に許可されている場合を除き、ローカルファイルおよび AIR API へのアクセスを制限することです。ネットワークやインターネットから読み込まれたコンテンツは、自動的にこのサンドボックスに割り当てられます。非アプリケーションサンドボックスでは、`eval()` 関数を使用したり、`innerHTML` の割り当てによってコードを実行したりすることができます。非アプリケーションサンドボックス内のコードには同一生成元ポリシーが適用されるので、デフォルトでは、XMLHttpRequest を通じてクロスドメインデータを読み込むことはできません。ただし、非アプリケーションサンドボックス内のコードは、直接 AIR API を実行できません。

このサンドボックスのいくつかの制限によって、ブラウザモデルとは若干動作が異なりますが、開発者はアプリケーションサンドボックスで実行されるコードを作成することをお勧めします。

サンドボックスブリッジ

最も単純な場合、ネットワークコンテンツを使用する HTML ベースの AIR アプリケーションは、ルートユーザインターフェイス (UI) ファイル、ダウンロードしたデータやファイル、および (アプリケーションサンドボックス内の) アプリケーションファイルで構成されます。ただし、外部コンテンツや一部の UI ファイルが非アプリケーションサンドボックスで動作し、アプリケーションファイルがアプリケーションサンドボックスで動作する場合、非アプリケーションサンドボックスのコンテンツと、アプリケーションサンドボックスのコンテンツの相互作用は以下のようになります。



AIR ソリューションはサンドボックスブリッジ API であり、これによって異なるサンドボックス間での間接的な通信が有効になります。開発者は AIR API を呼び出す関数を作成し、サンドボックスブリッジでその「仲介」関数を公開できます。これらのブリッジ関数は、基本的にブリッジ上に配置され、非アプリケーションサンドボックスから呼び出されるまで待機します。

このアーキテクチャによって、2つのゴールが達成されます。1つは、非アプリケーションサンドボックスのコンテンツが AIR API に直接アクセスしないので、アプリケーションコンテンツが未知数の潜在的に悪意のある外部データから保護されることです。もう1つは、開発者がブリッジメカニズムを通じて公開することを明示的に選択した API のみが公開されることです。

サンドボックスブリッジはセキュリティを保証しませんが、開発者は AIR API を信頼されないコンテンツに公開する際のリスクを軽減するための鍵を握っています。クライアントサーバモデルと同様に、サーバはクライアントを信頼するべきではなく、クライアントはサードパーティから読み込まれたデータを信頼するべきではありません。同様に、開発者は、サーバを危険にさらす可能性があり、オペレーティングシステムへのアクセスを可能にするので、サーバでのコードの読み込みを使用したり、信頼したりしないようにする必要があります。したがって、ブラウザと AIR サンドボックスブリッジのいずれのクライアントサーバモデルの場合も、writeFile() や readFile() などの基本的なシステム API を常に攻撃から保護する必要があります。

Adobe AIR セキュリティは、Web アプリケーション開発コミュニティを念頭に置いて設計されています。アドビ システムズ社では、セキュリティのリスクを軽減するための標準および手法をサポートしています。さらに、OpenAjax Alliance などの組織が、次のようなベストプラクティスの公開を支援しています。

- 信頼されないコンテンツをフレームまたは `iframe` に分離します。同一生成元ポリシーを利用することによって、攻撃者は DOM ツリー全体にアクセスすることが困難になります。
- 異なるドメインからフレームに読み込まれたデータには、固有の JavaScript 実行コンテキストと DOM ツリーを指定します。
- コードを動的に生成および実行しないでください。
- 信頼されるソースからのものではない HTML コンテンツを挿入しないようにします。
- セキュリティで保護された JSON を使用します。

AIR には、開発者がこれらのベストプラクティスに従い、HTML ベースのアプリケーションのセキュリティを強化できるようにするメカニズムが用意されています。詳しくは、<http://www.openajax.org/whitepapers/Ajax%20and%20Mashup%20Security.html> を参照してください。

開発者は、明示的にサンドボックスの橋渡しをする必要があります。

サンドボックスの権限のまとめ

機能	アプリケーションサンドボックス	その他の（非アプリケーション）サンドボックス
AIR API に直接アクセスできるか。	はい	いいえ
ブリッジを通じて AIR API を使用するアプリケーションサンドボックスの関数にアクセスできるか。	N/A	はい
<code><script src='http://www.example.com /some_code.js'></code> などのリモートスクリプトを読み込むことができるか。	いいえ	はい
デフォルトで、クロスドメイン要求（XHR）を実行できるか。	はい	いいえ
load イベントの後にストリングをコードとして動的に読み込むことをサポートしているか（eval() 関数、setTimeout('string', millis)、javascript: URL、innerHTML を通じて挿入される onclick='myClick()' などのエレメントの属性ハンドラなど）。	いいえ	はい
Ajax フレームワークが変更なしに機能するか。	一部のフレームワーク	はい

セキュリティの向上

AIR セキュリティモデルは、開発者の制御下でないデータに対してシステム API を公開することに関連するリスクを軽減しながら、Ajax モデルとの後方互換性を実現することを目指しています。ブラウザのクライアントサーバモデルでは、ブラウザがプレゼンテーションの大半および動的なサードパーティデータの読み込みの一部を処理しています。サーバは、機密性の高い処理の多くを実行し、記憶領域や暗号化などの OS に似た機能を提供します。間接的に公開された関数を通じてやり取りする 2 つのサンドボックスを用意することによって、Adobe AIR では攻撃者がエンドユーザのコンピュータやデータにアクセスする可能性を低くしています。

Ajax フレームワークの利用

非アプリケーションサンドボックスは事実上 HTML ブラウザサンドボックスなので、Ajax 開発者は既存のアプリケーションおよびパターンを比較的容易に AIR に移植できることがわかるはずです。例えば、Web から既存のクロスワードゲームを入手して、そっくりそのまま非アプリケーションサンドボックスのフレームに挿入できます。

非アプリケーションサンドボックスで実行されている Ajax フレームワークは、Web ブラウザで実行される場合とまったく同じように実行されますが、フレームワークのバージョンによっては、eval() やクロスドメインのコード読み込みを利用する場合に、アプリケーションサンドボックスでは完全には機能しない可能性があります。これは、主に、特定のフレームワークが eval() やこれに類似する動作にどの程度依存しているか、および開発者がフレームワークのどの部分を使用しているかによって決まります。

これらの制限で回避されないのは、JSON オブジェクトリテラルを含む eval() の使用です。これによってアプリケーションコンテンツで、JSON JavaScript ライブラリを使用できます。ただし、アプリケーションサンドボックスコンテンツで、オーバーロードされた JSON コード（イベントハンドラを含む）を使用することは制限されます。その他の Ajax フレームワークおよび JavaScript コードライブラリの場合は、フレームワークまたはライブラリ内のコードが動的に生成されたコードに関するこれらの制限内で機能しているかどうかを確認します。制限に従っていない場合は、フレームワークまたはライブラリを使用するコンテンツを非アプリケーションセキュリティサンドボックスに含める必要があります。アドビ システムズ社では、既知の Ajax フレームワークによる AIR アプリケーションサンドボックスのサポートを示したドキュメントを用意しています。

http://www.adobe.com/go/airappsandboxframeworks_jp

アプリケーションの移行

単純なクロスワードゲームを使用して、saveGame() および loadGame() などの関数を実装するには、どうすればよいでしょうか。そのためには AIR API にアクセスする必要があり、関連 API をアプリケーションサンドボックスに実装します。アプリケーションサンドボックス内の関数およびデータは、サンドボックスブリッジを通じて非アプリケーションサンドボックスに公開されます。このプロセスには以下の手順が含まれます。

- 1 元のルートコンテンツファイルの名前を someName.htm に変更します。
- 2 myRoot.htm という名前の新しいファイルを作成し、someName.htm を指す IFRAME または FRAME を含めます。
- 3 AIR API を呼び出す関数を作成します。API を直接呼び出さない方が安全です。
- 4 アプリケーションサンドボックスの機能を非アプリケーションサンドボックスに公開する parentSandboxBridge を作成します。

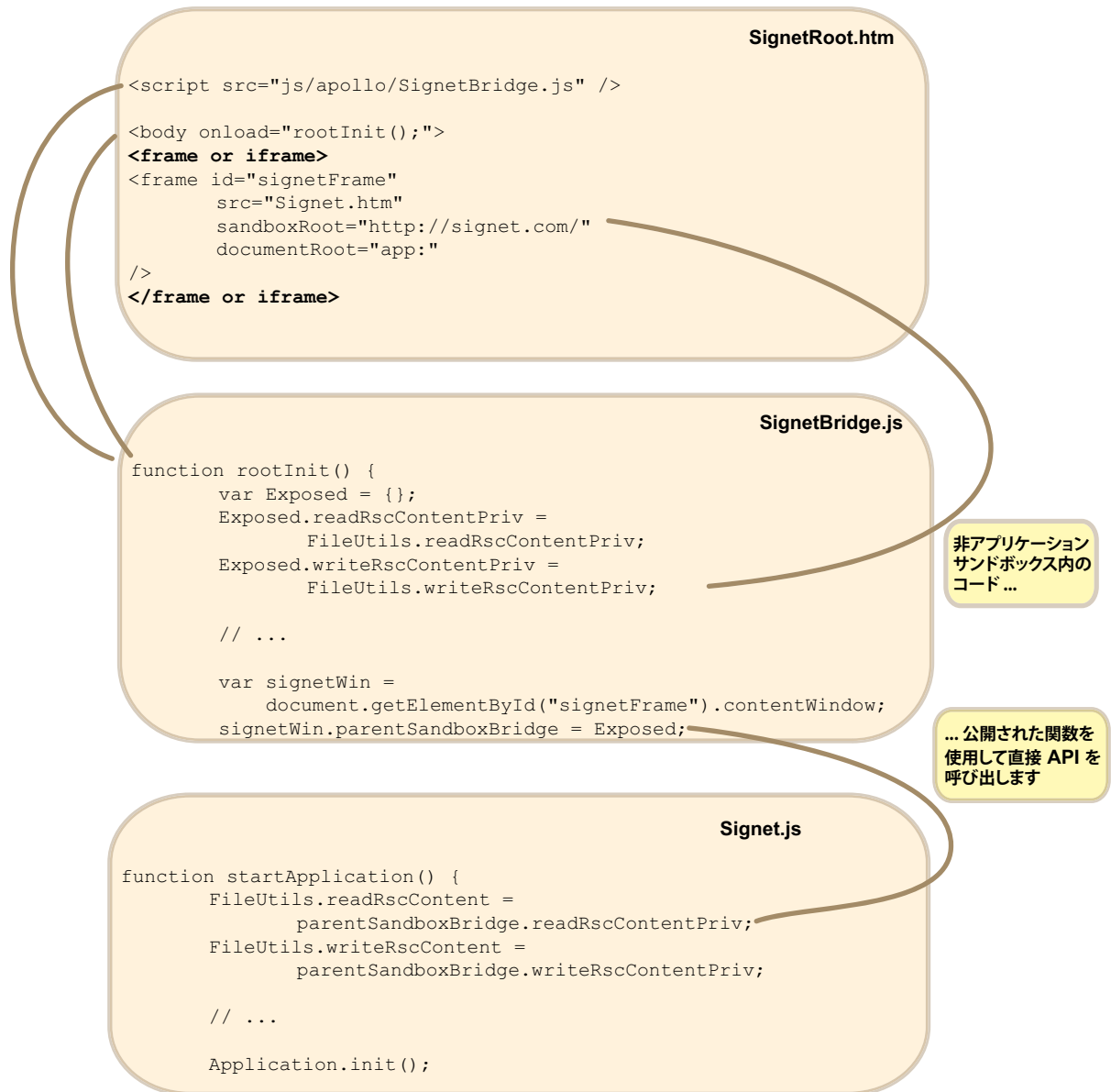
2つのサンドボックス間の通信は当然双方向です。parentSandboxBridge と childSandboxBridge によって、ブリッジでの双方向のやり取りが可能になります。これによって、開発者はアプリケーションとアプリケーション環境の間でのコンテンツフローをより細かく制御できます。

実際の使用例

Signet (del.icio.us ユーザ用のブックマークマネージャ) は、次のように AIR セキュリティモデルを実装した HTML ベースの AIR アプリケーションです。

- 1 最初に、id が signetFrame のフレームと利用可能なドメインを識別するために必要な 3つの属性を含む、SignetRoot.htm が作成されます。
- 2 SignetRoot.htm で、onload が、公開されるすべての関数を含む SignetBridge.js を呼び出します。
- 3 SignetBridge.js では、parentSandboxBridge を通じて signetFrame に関数が公開されます。
- 4 Signet.js で、公開された関数が直接 AIR API を呼び出します。例えば、writeRscContentPriv は writeRscContent を呼び出します。

この手法は比較的単純ですが、効果は絶大です。2つのサンドボックス用にリモートドメインとローカルドメインが定義され、`onload` の処理中にブリッジが呼び出され、ブリッジ API を通じて、仲介関数によって API が間接的に実行されます。



セキュリティ：共通の責任

最終的に、RIA のセキュリティは、アプリケーションのライフサイクルに関係するすべての当事者が支えることとなります。エンドユーザは、開発者がベストプラクティスと標準に従っていることを信じています。

推奨される開発手法に加えて、AIR アプリケーションのセキュリティは、ユーザのコンピュータにインストールするときに始まっています。

- 最初のインストールワークフローは変更できません。

- アプリケーションは、すべての Web ブラウザおよびオペレーティングシステムで共通した、合理化され、一貫性のあるインストールプロセスを使用します。
- インストールは、インストールされたアプリケーションでは操作できないランタイムによって管理されます。
- ユーザがコードの作成元を検証し、アプリケーションのアクセス権限を確認できるように、インストーラファイルは電子署名されている必要があります。
- ランタイムのメモリ管理によって、バッファオーバーフローやメモリ破損などの脆弱性が最小限に抑えられます。

セキュリティの問題に対応することにより、最終的には開発者の負担が増加します。ただし、AIR セキュリティモデルでは、セキュリティで保護されたコードを作成および保守するための複雑な作業を軽減することを目指しています。

ベストプラクティスの一覧については、AIR 開発者向けのドキュメントを参照してください。例えば、外部ファイルの使用を必要なものだけに制限する方法や、特定の操作にネットワークソースのデータを使用しない方法などが示されています。

Adobe Systems Incorporated 345 Park Avenue, San Jose, CA 95110-2704 USA www.adobe.com Adobe、Adobe ロゴ、Adobe AIR、ActionScript、Flash、および Flex は、アドビ システムズ社の米国ならびに他の国における登録商標または商標です。Windows は、米国 Microsoft Corporation の米国およびその他の国における登録商標または商標です。Mac OS は、米国およびその他の国々で登録された Apple Inc. の商標です。その他すべての商標は、それぞれの権利帰属者の物です。©2008 Adobe Systems Incorporated. All rights reserved. 11/08