# Protecting Intellectual Property in Flash Applications

## WHITE PAPER

Options for protecting Flash ActionScript source code from reverse engineering



By Mohammed Abu Rahmeh, VP of Research
15 July 2009

KINDISOFT

# Executive Summary

Proprietary intellectual property (IP) in the form of graphics and ActionScript source code resides in compiled Flash applications. Unlike compiled binary files, IP in Flash SWF files is easily accessible to unauthorized people armed with one of today's powerful decompiling and reverse engineering tools.

This paper looks briefly at several of the most common available methods of securing IP in Flash applications, outlines the pros and cons of each, and discusses why code obfuscation may be the best option for many Flash application developers. It provides a nutshell overview of how the code obfuscation works, explains briefly the quality factors one should understand in order to evaluate any obfuscation solution, and it concludes with a look at Kindisoft's obfuscation solution secureSWF.

## The Problem

ActionScript is compiled into byte-code that is contained within SWF files. Unlike binary executables, the byte-code contains enough information about the source code that makes it possible for over 22 tools available on the internet today to decompile readable and useful source code with a click of a button.

# Vulnerabilities in Flash Applications

The Internet has made it easy for proprietary computer applications to fall into the hands of those who would exploit them illegally, and easy for those people to achieve huge distributions that rob the rightful owner company of revenues to which it is entitled.

In addition to executable application itself, the source code is a highly valuable IP asset to the developer organization. In the first generation desktop world of compiled binary executable files, the source code was quite difficult to reproduce from the compiled distribution. It could be done, but it was not easy.

With the advent of programming languages and applications, and Flash applications in particular, a lot of information about the source code and its structural characteristics are also compiled into distributions. As Flash gained broad usage in gaming, video content delivery, and business applications, we have seen the advent of powerful decompilers that make it easy to extract everything from graphics and animation, all the way to ActionScript source code from the compiled SWF files of Flash applications.

In addition to these vulnerabilities, Flash applications are subject to first-generation issues such as unauthorized distribution, as well as Internet age issues such as execution on unauthorized web servers and domains.

Diagram 2-1 on the next page illustrates how the source code is passed from the developer to the attacker.

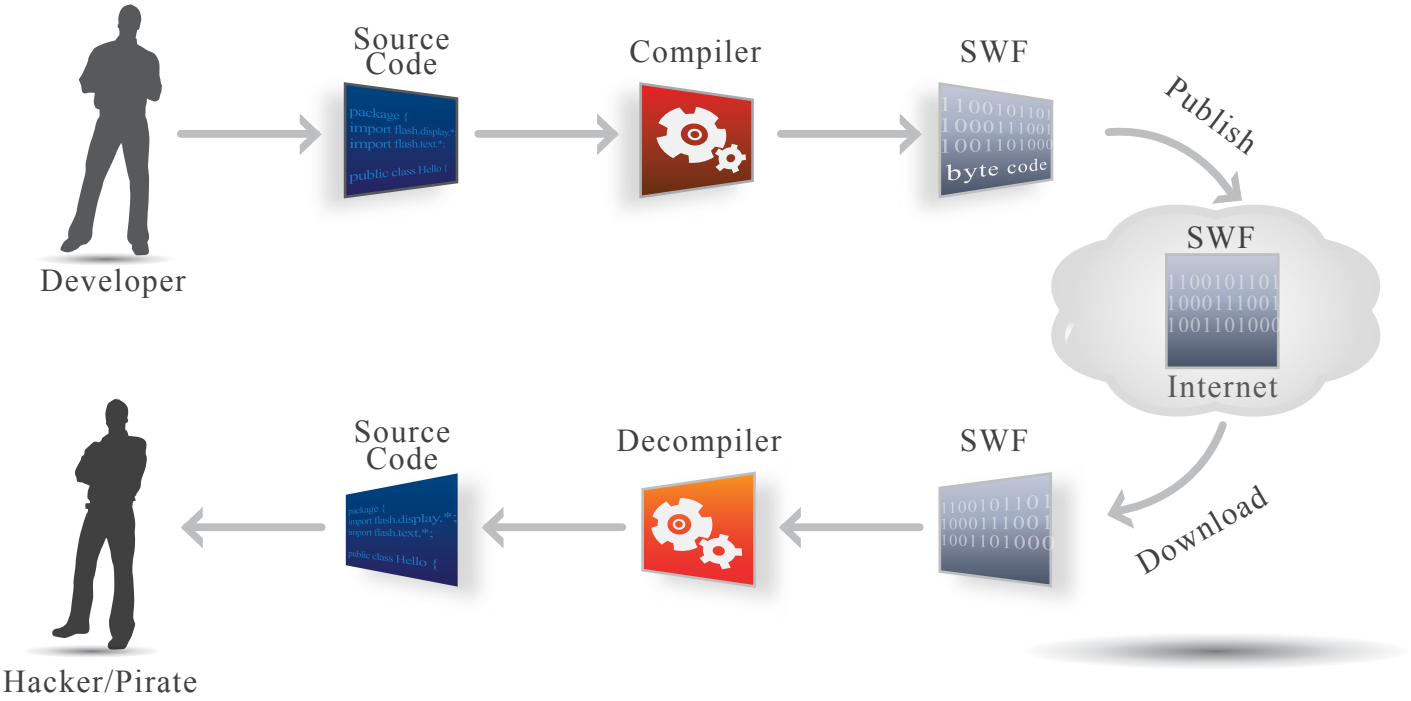



Diagram 2 · 1
Reverse Engineering Compiled ActionScript

# Approaches to Protecting Source Code

The three main approaches to securing Flash applications used in the industry today are:

- Server-side execution
- Application encryption
- Code obfuscation

There are pros and cons to each approach. Let's look briefly at these.

## Server-side Execution

One way to prevent reverse engineering is by keeping important components of the application running on the server-side and only delivering minimal functions to run on the client.

The server-side approach has the advantage of preventing unauthorized distribution of the application by taking advantage of all the security available on the network and the server. However, there are several disadvantages:

- It's not always technically possible.
- Limited distribution may not suit the business model.
- Execution is subject to interruptions of Internet service.
- The organization must bear the cost of the necessary network and server infrastructure.

Diagram 3 · 1 on the next page illustrates this approach.

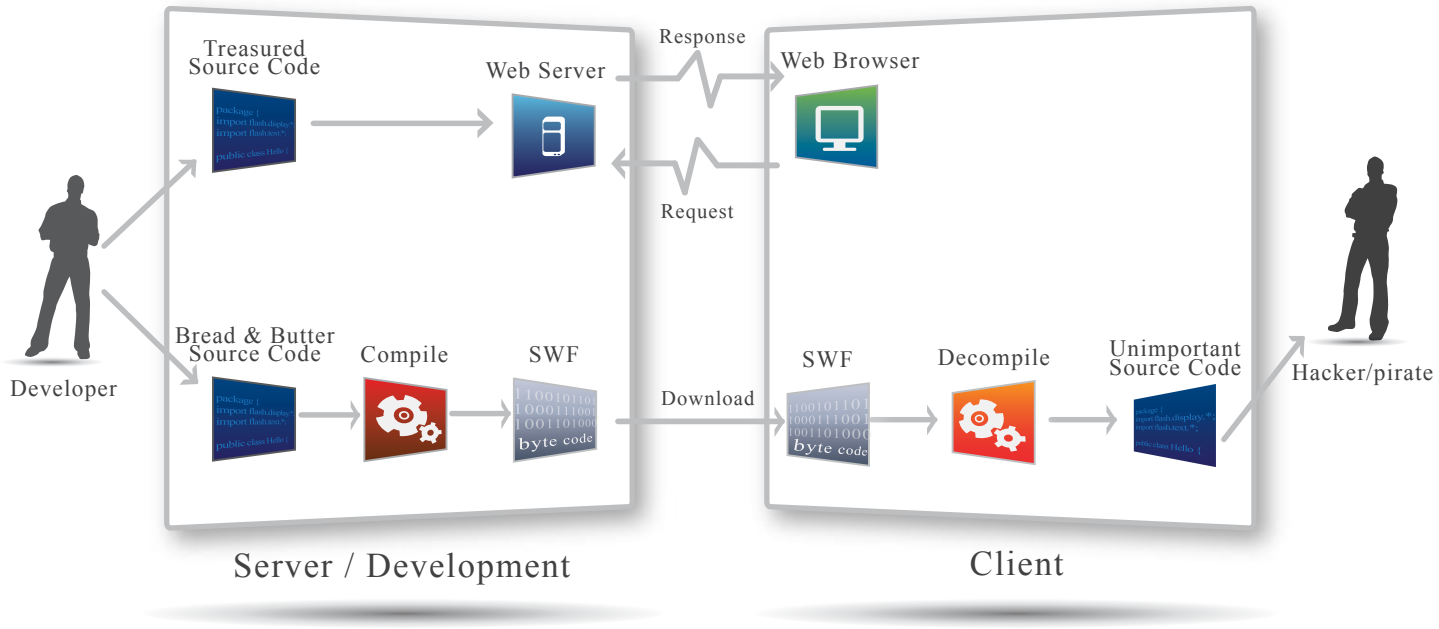



Diagram 3 - 1
Server-side Execution Approach

## Main Solutions

- Server-side execution
- Application encryption
- Code obfuscation

## Application Encryption

This approach applies encryption to the application as a whole, or to key components inside the application. Although actual encryption can make your code completely unreadable, it suffers from a classic encryption flaw; it needs to keep the decryption-key with the encrypted data. An automated tool could be created to decrypt the code. Once that happens the fully unencrypted code is in plain view. Disadvantages of this approach also include:

- Added complexity to the distribution process.
- Potential for performance impact due to overhead required by decryption processes.

Diagram 4 – 1 on the following page illustrates how code encryption works.

## Code Obfuscation

This approach came about with the wide adoption of cross-platform interpreted languages generating easily reverse engineered byte-code — Java™ for example. It is essentially a process that makes code unintelligible for humans without affecting its interpretation by computers.

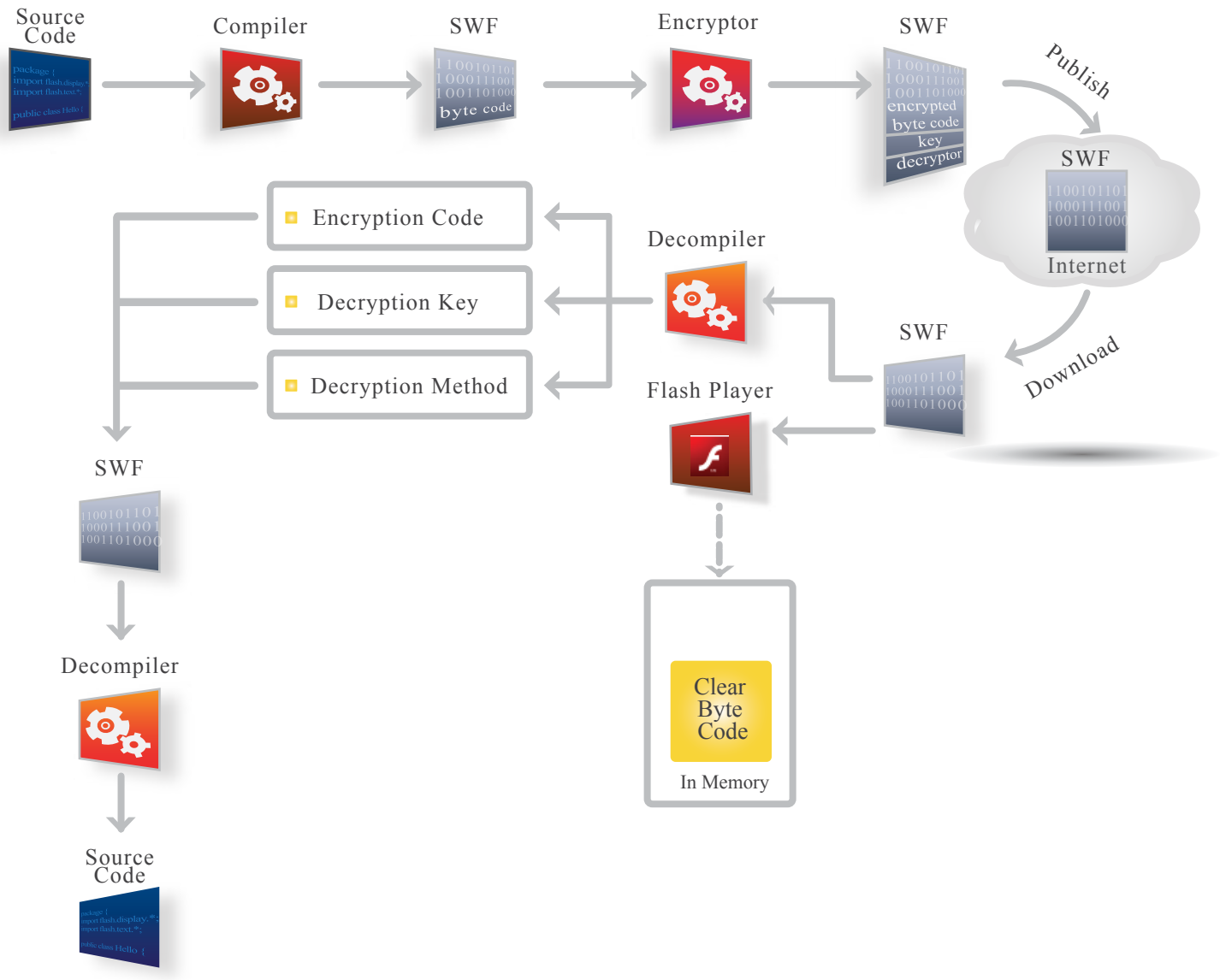



Diagram 4 - 1
Flash Applications Encryption, and Decryption

Code obfuscation raises the bar for reverse engineers and makes decompiling Flash applications just as hard as reverse engineering native executables, if not a bit harder. It delivers a level of security that is good enough in the vast majority of use cases, and several fairly compelling advantages:

- Does not affect distribution.
- No added complexity to the development process, and only one additional step in the packaging.
- No need to deliver decryption keys or devices to end users
- Negligible performance impact, as the computer still understands the obfuscated code
- Controlled costs through less complex development, ease of change to business models, and no need to maintain server-side infrastructure.

Obfuscating Flash applications can simply be done by running the final SWF files, that are ready to be deployed, into the obfuscator as the diagram shows below.

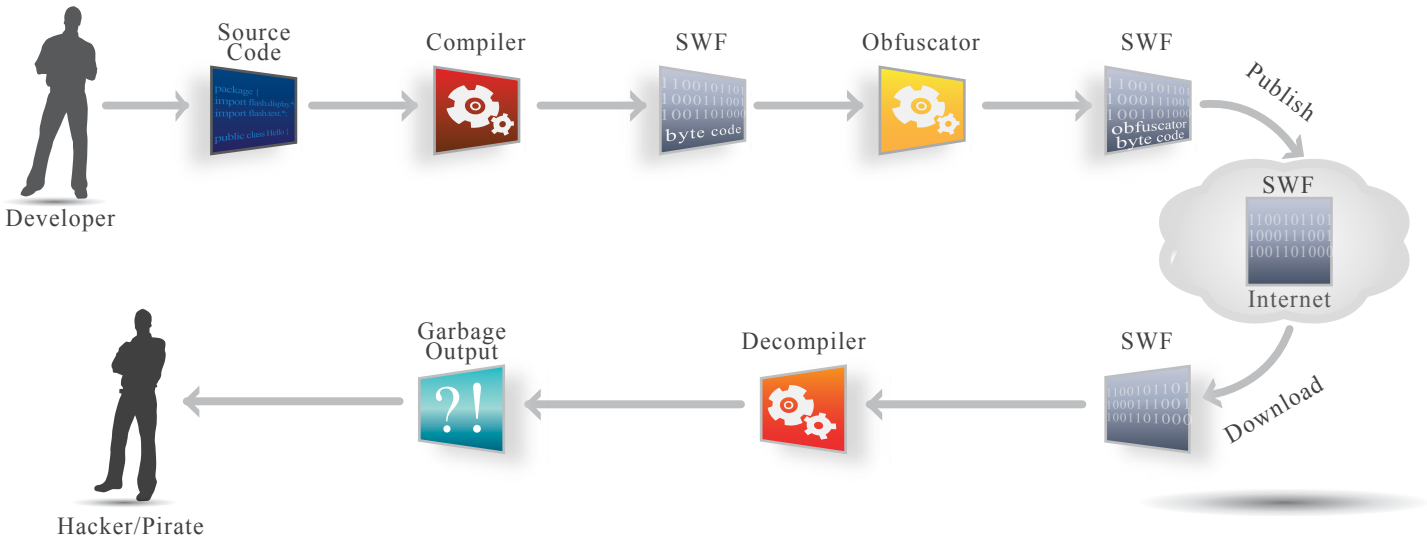



Diagram 5 - 1
Code Obfuscation

# The Case for Code Obfuscation

From the foregoing section, you can see that code obfuscation has the potential to deliver a good level of security for your Flash applications, plus simplicity, flexibility, and overall low total cost of ownership (TCO). You may well conclude that it's an option that has potential for your Flash development, in which case you will want to start learning more about it. So in this section we will give you a brief overview of the technology and answer some key questions many people have about it.

# What is Code Obfuscation?

In a nutshell, code obfuscation is the process of transforming code into a form that is unintelligible to human readers while preserving the functionality and structure for computers.

Obfuscated executable or byte-code, when reverse engineered or decompiled, yields output which is extremely difficult for humans to grasp or understand, and therefore extremely difficult to modify or manipulate. The key aspect of securing a code with obfuscation is the paradox of *security through obscurity*.

# How Does Code Obfuscation Work?

There are many techniques for obfuscation, and it has become a very active research topic, especially as applied with Java byte-code. The same concepts and techniques are applicable to Flash applications as well. Methods range from simple transformations to complex structural and control-flow manipulation.

Broadly speaking, obfuscation methods can be categorized this way:

**1. Lexical transformation:**

Identifiers, variables, and structure names are scrambled/renamed to reduce the readability of the code for humans (but readability for computers is not affected).

**2. Control transformation:**

The application control flow, computations, as well as the aggregated functionality are all processed and modified in such as way as to break the high level language, obscuring the logic of code statements and computations for the human reader. Examples of control transformation include the use of resilient opaque predicates, statement reordering and merging unrelated components. Bear in mind that such transformations do not affect the functionality of the application.

**3. Data abstraction transformation:**

The relations that define the building blocks of the code design are altered, reducing the clarity of the overall design. Examples include adjusting inheritance relationships, restructuring data arrays, and encrypting literal strings.

The following diagram illustrates the three main obfuscation techniques categories:

## Effectiveness

Unlike other code protection techniques, there is no concept of code decryption with code obfuscation. Protected code blocks are always in the executable state, and they are executed as transformed obfuscated code. The original code is completely absent.

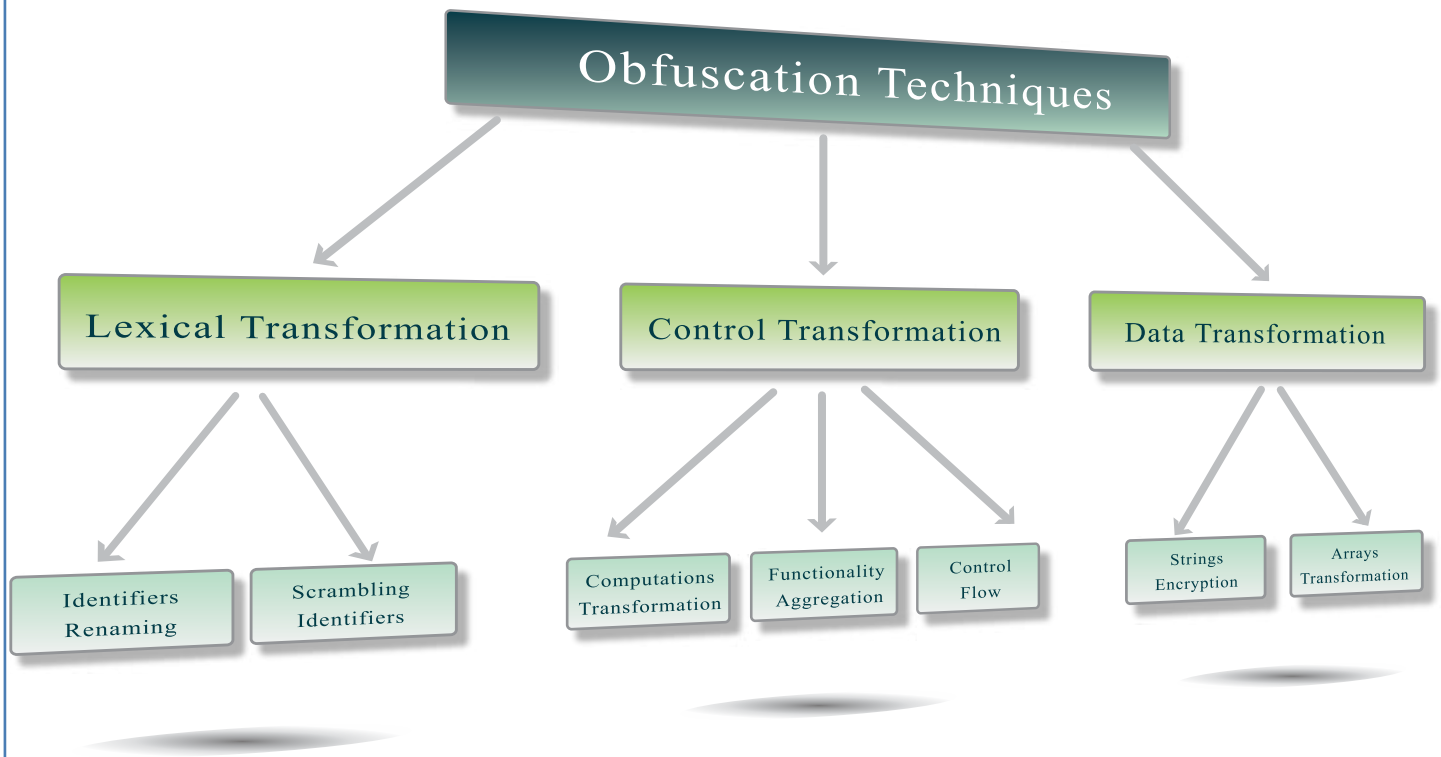



Diagram 6 - 1
Obfuscation Techniques Categories

# Obfuscation Concepts You Should Know

When evaluating or discussing tools and solutions for code obfuscation, it will be useful to understand some terminology related to the security level, or quality of the obfuscated code. The following terms describe some fundamental metrics.

- **Potency:**
  Refers level of obscurity an obfuscation method had added to some code.

- **Resilience:**
  Is the resistance of obfuscated code to being de-obfuscated – that is, to being inspected and understood by unauthorized people.

- **Stealth:**
  Is the visibility of obfuscated code within the source code as a whole.

- **Cost:**
  Is the runtime computing resources overhead added by the obfuscation.

# Other Security Issues to Consider

Code obfuscation is just one piece of the security puzzle. As we have seen, it protects your proprietary ActionScript source code from being easily read and used by unauthorized people. But concentration on this issue should not obscure other important issues such as:

- Illegal/unauthorized distribution of the application.

- Running the application online from unauthorized servers/domains.

- Capture of the running SWF file into local storage for further manipulation.

Any tool or solution you adopt should provide at least some level of support for these important security considerations.

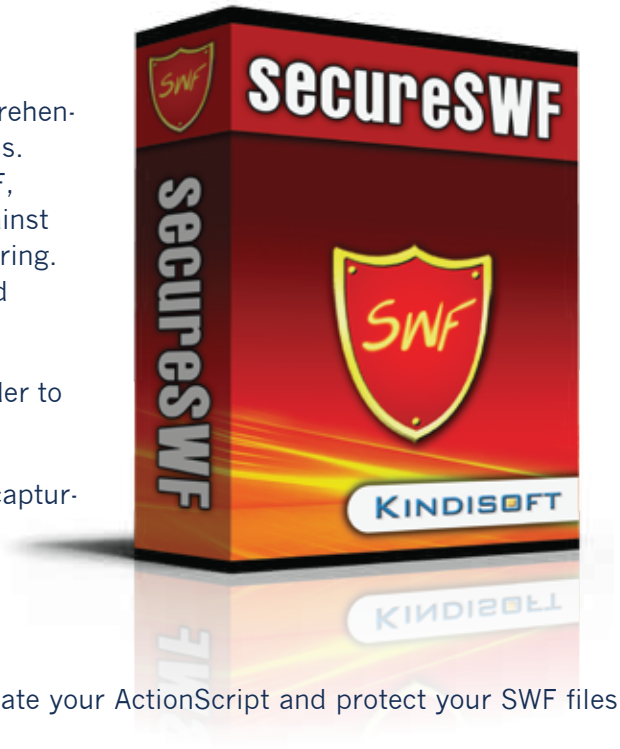
# Conclusion

Flash applications are known for their vulnerability to decompilation, an issue that has sent Flash developers scurrying for ways to protect their intellectual property. Different generic methods can be employed to overcome these issues, most viable of which is code obfuscation.

In this paper we have talked about how the Internet has created a need for Flash developers to protect their ActionScript source code from decompilation or reverse engineering and reuse by unauthorized parties. We have explored the pros and cons of server-side execution, encryption, and code obfuscation as approaches to protecting SWF files before publication. We have seen that code obfuscation provides significant security benefits without the disadvantages (or with a lesser degree of them) of the other approaches. We've taken a brief look at what code obfuscation is and the basics of how it works, and talked briefly about some issues that go along with code obfuscation. So what's left?

How about a powerful, flexible code obfuscation solution that not only protects your source code, but helps you address other security issues?

# Introducing secureSWF




secureSWF, developed by Kindisoft LLC, is a specialized, comprehensive tool that provides excellent protection for Flash applications. Code obfuscation is just one of the many features of secureSWF, including specific features for protecting Flash applications against other security threats besides decompiling and reverse engineering. For example, secureSWF helps prevent illegal redistribution and off-line execution through features like:

• **Encrypted Domain Locking** which enables the copyright holder to limit the domains from which the application can be executed

• **Encrypted Loader Creator** which prevents the possibility of capturing the SWF file into local storage for further manipulation.

# secureSWF Key Features

secureSWF has the following set of features to help you obfuscate your ActionScript and protect your SWF files against Flash decompilers in the most convenient way:

- **All ActionScript Versions on Every Platform:**
  secureSWF supports ActionScript v1, v2 and v3. And is available for Windows, Mac OS X, and Linux.

- **Identifiers Renaming:**
  secureSWF renames just about every identifier (including classes, symbol instances names, and even frame labels) in your ActionScript into shorter meaningless names that include unprinted characters.

- **Smart Identifiers Selection:**
  secureSWF automatically determines which identifiers are safe to rename and which are not making code obfuscation easier than ever.

- **Stops Flash Decompilers:**
  secureSWF stops all Flash decompilers using the following advanced mechanisms:
  - Control flow obfuscation.
  - Dynamic code wrapping.
  - Statement-level randomization.

- **String Encryption:**
  secureSWF helps you protect your Flash application from variety of security threats by providing literal strings encryption.

- **Code Optimization:**
  secureSWF provides a number of code optimization techniques that you can use to produce smaller files and even faster code.

- **Access Limitation:**
  secureSWF limits access to your published SWF files through:
  - Encrypted domain locking.
  - Encrypted loader creator.

- **Build Integration:**
  secureSWF has XML project configuration files, a command-line interface, and an Ant script task definition to help you integrate obfuscation into your build process.

# Kindisoft's Clients

secureSWF has been proven in real-world applications by real-world companies to deliver high quality obfuscation in terms of potency, resilience, stealth, and low cost. Here are just a few of them:


"Thanks to Kindisoft's secureSWF we can confidently launch our Flash applications. We strongly recommend Kindisoft to everyone interested in protecting their Flash applications. Thanks for the great product and support."

**Sérgio Bessa**
*Director of Development at Ideavity Lda.*
*www.mingleworld.com*

"As developers of high quality and cutting-edge technology we were constantly searching for a proper way to protect our intellectual property. SecureSWF protects our work in no time, it offers us the most secure and advanced obfuscating measures on the market. We've tested the most advanced Flash decompilers on our SecureSWF-protected swf files; none of them was able to view any valuable information and many of them just crashed. We have been using SecureSWF from the beginning, appreciating its reliability and we can recommend it without exceptions. SecureSWF supports the most advanced Flash features, including Flash remoting, seamlessly. String encryption makes content hiding possible and secure, resulting in the circumstance that you will not able to tell where the preloader loads our real flash content from."

**Harald Wild**
*Founder & CEO at Wild Technology.*
*www.wild-technology.de*

# About Kindisoft

Kindisoft was established in 2005, to provide its clients with a variety of products and services to help them better protect their Rich Internet Applications. In a world were intellectual property is a major concern for software providers, and theft only requires a few mouse clicks and a bad intention, creative people need to focus on their innovation instead of worrying about their ideas being hijacked. Our products provide developers and artists with an easy-to-use and effective way to strengthen security and protect the intellectual property of their media rich applications.

**www.kindisoft.com**


### Headquarter

P.O.Box 36 iPARK · Kindisoft
Amman · 11941
Jordan
Tel: +962-6-533-5152

### North America

324 New Brooklyn Road
Berlin NJ 08009
USA
Tel: +1-(609)-678-0325