ソフトウェア電子透かしに関する研究
～埋め込みツールの実現に関する検討～

2001MT101 問山 典章
指導教員　真野 芳久

1　はじめに

近年のインターネットの普及により、ソフトウェアの無断複製などの不正利用が行われ、著作権侵害が問題になっている。その対策として現在、ソフトウェアの著作権保護を目的としたソフトウェアプロテクションと総称される技術の研究が進められている。その中にソフトウェア透かしという技術があり、本研究ではソフトウェア透かしの研究環境を整えるために、種々の透かし埋め込み方法を容易に実装できる埋め込むツールの検討、構築を行う。

2　ソフトウェア透かしとは

ソフトウェア透かしはプログラムに予め透かし情報を埋め込んでおき、必要な時に埋め込んでおいた透かし情報を取り出す技術である。ソフトウェア透かしは、その埋め込みの形式から大きく静的透かしと動的透かしの二種類に分けて考えられる。

2.1　静的透かし

静的透かしとはプログラムを実行することなく、透かしを取り出すことができる透かしである。この場合、基本的に透かし情報はプログラムの仕様とは無関係な要素として埋め込まれる。

2.2　動的透かし

動的透かしとは、特定の入力に対してプログラムが実行された場合にのみ、透かしが作られ取り出せる透かし技術のことである。特定の入力に対するプログラムの実行があって、初めて透かしとしての形が構築されるので、透かしへの攻撃者に発見されにくい特徴がある。特に、動的に構築されたグラフの位相の中に透かし情報を埋め込む方法を動的グラフ構造透かしという。

3　ソフトウェア透かし埋め込みツール

埋め込みツールの利用者が期待する機能としては、埋め込んだ透かしの検証ができ、容易に透かしの埋め込み方法を追加できるということである。それに加え、ツールに求められる機能として、プログラムのどこに透かしを埋め込むかを決めるということが考えられる。よって、透かしを埋め込む場所を決定し、透かし処理部分をはめ込む形によって加え、容易に様々な透かし処理を実行、追加することができるような透かし埋め込みツールの検討を行う。本研究での埋め込みツール利用者は、埋め込み前のプログラム開発者を想定しているが、現状では、透かし処理部分の記述をする透かしの研究者となっている。埋め込み対象とする透かしの種類は、動的グラフ構造透かしを含む、動的透かしである。

3.1　ツール実行までの手順

用意するものとして、埋め込みツールの他に C 言語でソース状態の透かしを埋め込みたいプログラム (複数ファイルから成り立っているプログラムでも可)、はめ込み部分である透かし処理の記述をしている C 言語のソースプログラムが必要であり、埋め込み前のプログラムの制約として、必ずキーボードで数値の入力をすることを、透かし処理部分の制約として、透かし処理開始の関数とその引数は、決められた関数名と引数で開始し、透かし処理後に引数の値を変えておくということを付けている。

3.2　埋め込みツールの構成と処理の流れ

本研究で試作した埋め込みツールと埋め込み前後のプログラムの関係は図 1 のようになる。

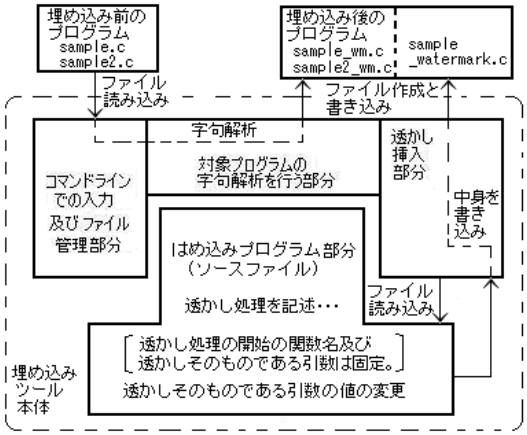

図 1　埋め込みツールと埋め込み前後のプログラムの関係

透かし処理を行うはめ込みプログラム部分は、データとして扱い、中身をそのままコピーするといった方法で扱う。記述することは、透かし情報の値を保持している引数に透かし処理を施す関数である。埋め込みツールとはめ込みプログラム部分の接点はファイル読み取りのみで、はめ込みプログラム部分の種類を増やすことが埋め込むことのできる透かしの種類を増すことに繋がるので追加が容易である。

内部処理の流れとして、埋め込み後のファイルの作成、そのファイルへ解析済のプログラムのコピーを行いながらの入力処理と入力に使用されている変数を探すための字句解析、解析の該当場所への受渡し関数の書き込み、透かしの記述をするファイルの作成、そこへ透かし処理

へ移行する際の処理の書き込みとはめ込みプログラムの中身を書き込み、終了という流れになる。

3.3 埋め込み後のプログラム本来の処理と透かし処理

プログラム本来の処理と透かし処理の接点の様子は、図 2 のようになる。実行処理として、まず、入力した値が関数に渡され、透かしを認識するための鍵となる入力かどうか判別し、もしその引数が透かしを認識するための鍵となる入力の場合、透かしを実行する関数へ埋め込む透かしの値を渡す。渡された場合、透かし埋め込み処理の実行をし、実行後に埋め込む透かしの値をそのまま残しておかないように透かしそのものの値を保持している変数の値を変更する。そして、プログラム本来の処理に戻るという流れになる。

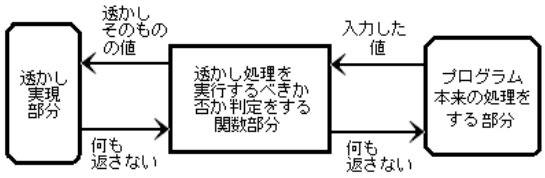

図 2　埋め込み後のプログラムの透かし実現部分との関係

3.4 透かし情報の確認

透かしの埋め込まれたプログラムを動作させる、その時、透かしを認識する際に必要な鍵となる入力をし、プログラム終了時に残った透かしが含まれているメモリを、認識ツール (本研究では対象外) を使用し、挿入した透かしの認識手順に則って埋め込まれた透かし情報を認識する。各ツールと利用者の関係は図 3 のようになり、認識ツールは本研究での埋め込みツールと互いに独立しており、埋め込んだ透かしの種類によって決まった認識方法で透かしを取り出すツールである。

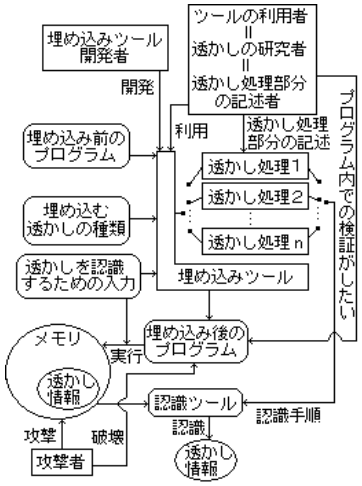

図 3　各ツールと利用者との関係

透かしを埋め込んだプログラムが盗用された際、埋め込みツール利用者が埋め込んだ自分が制作したということを証明できる透かし情報を取り出す手順を記録しておき、それを盗用された際にそのプログラムを記録した手順にしたがって透かし情報を取り出すという方法で自分が制作したプログラムであるということを証明する。

3.5 透かしの耐性と問題点

このツールでの埋め込み後のプログラムの耐性として、埋め込まれた透かしによって変わってしまうが、主に動的透かし特有の耐性があり、実行をしないと透かしを認識できない、鍵となる入力のときのみの透かしの実行ということが挙げられ、どれも透かし攻撃者に埋め込まれている透かしを見つかりにくくする効果がある。

本研究で試作したツールの問題点として、鍵として使える入力は最初の入力のみであること、透かし埋め込み場所が入力の直後に固定されている、コンパイル後のプログラムは埋め込み不可能であること、3.1 節での制約があること、透かしのソースを見ると透かしがそのまま代入されている部分があり、透かしが一目瞭然であること、逆コンパイルを行い、静的な解析をされたり、動的な解析で統計的に解析されてしまうと攻撃者に透かしの存在が発見されてしまう可能性があることが挙げられる。これらの問題点を解決するのに、より上位の解析を行ったり、他の言語で現存しているツールを参考にし、コンパイル済のプログラムでも埋め込むことが出来るようにするという方法や静的な解析については、透かし部分の分割や難読化を施すということが考えられる。

4 おわりに

本研究では透かし処理部分をはめ込みプログラムにすることによって、埋め込める透かしの種類を容易に追加できる埋め込みツールを試作した。問題点も多く、多くの制限があるため一般的なプログラムに埋め込むことができない。よって、これらの問題点を解決し、一般的なプログラムに埋め込むことができるようなシステムの構築が課題となり、将来的には埋め込みツールの利用者がプログラムの開発者となり、実用化できるようなシステムの構築が課題となる。

参考文献

[1] C.Collberg,C.Thomborson:Watermarking,Tamper-Proofing,and Obfuscation − Tools for Software Protection, IEEE Tr. on SE, Vol.28 No.8(2002.8).

[2] C.Collberg, C.Thomborson: Software Watermarking Models and Dynamic Embeddings, ACM POPL (1999).

[3] C.Collberg, C.Thomborson: Dynamic Graph-Based Software Watermarking, Uni. of Arizona TR04-08, 2004.

[4] SandMark: A Tool for the Study of Software Protection Algorithms,http://www.cgi.cs.arizona.edu/~sandmark/sandmark.html