第14回 IEEE広島支部学生シンポジウム論文集
2012/11/17-18 岡山県立大学

ネットワークサービス利用時における 個人情報保護を目的としたプログラム変換方法の検討

Program Conversion to Protect User's Information

工藤 邦晃 † 高橋 健一 † 川村 尚生 † 菅原 一孔 †
Kuniaki Kuto† Kenichi Takahashi† Takao Kawamura† Kazunori Sugahara†
† 鳥取大学 工学研究科 情報エレクトロニクス専攻

1 はじめに

現在，様々なサービスがインターネット上で提供されている．例えば，ショッピングサイトやホテルの予約などが挙げられる．これらのサービスの一部は，サービス提供者が個人情報の提出を要求する．しかし，ユーザは一旦渡した情報をサービス提供者にどのように利用されるか確かめることができない．また，情報漏洩事件 [1] や不正利用，フィッシングサイトによる被害 [2] なども多発している．その結果，個人情報の不正利用に苦しまなければならない．もし，利用者がサービス提供者に個人情報を渡さなければ，利用者はこれらの危険を回避することができる．しかし，このとき利用者はサービスを利用することができなくなる．そこで，利用者の個人情報をサービス提供者が悪用することから守る仕組みが必要となる．そこで，利用者は個人情報の処理方法を選択可能なプログラム変換モデルを提案する．

本モデルでは利用者に個人情報の処理方法を選択させ，サービス提供者にその処理方法を強制する．利用者の個人情報はサービス提供者が持つプログラムで処理される．そのため，本モデルでは利用者によって選択された処理方法をサービス提供者が持つプログラムにインストールする．サービス提供者は利用者が選択した処理方法がインストールされたプログラムを通して個人情報を処理する個人情報は利用者が選択した処理方法に従って処理されるため，利用者はサービス提供者による個人情報の悪用を妨げることができる．その結果，利用者は安心して自身の個人情報をサービス提供者に提供できる．

本稿では，本システムを実現するための最初のステップとして，利用者によって選択された処理方法をインストールするためにプログラムを変換する方法を検討する．

2 プログラム変換モデル

プログラム変換モデルでは，サービス提供者が持つプログラムにユーザが指定した処理方法をインストールする．このことを実現するためには，サービス提供者が持つプログラムとユーザが指定した処理方法をインストールするための情報が必要となる．そこで，利用ポリシと保護ポリシを定義する．

利用ポリシ
プログラムを変換するためには，サービス提供者が持つプログラムがどのような処理をどのように行うかを知る必要がある．そこで，サービス提供者が持つプログラムの処理方法を利用ポリシとして定義する．利用ポリシはサービス提供者がユーザに要求する個人情報ごとに準備され，ユーザは利用ポリシを得ることでそのプログラムが個人情報をどのように処理するかを知ることができる．

保護ポリシ
提案モデルにおいてユーザが安心だと思う処理方法を選択し，その処理方法をサービス提供者が持つプログラムにインストールする．そこで，処理方法の説明とそのインストール方法を保護ポリシとして定義する．処理方法の説明によってユーザは自分が安心できる処理方法 (保護ポリシ) を選択でき，インストール方法に従って，その処理方法をサービス提供者のプログラムにインストールできる．

プログラム変換は変換モジュールによって行われる．変換モジュールではサービス提供者のプログラムを保護ポリシと利用ポリシに従って変換することで，ユーザが指定した処理方法を反映したプログラムを生成する．また，予め定義された複数の保護ポリシを管理するための保護ポリシデータベースを定義する．ユーザは保護ポリシデータベースの中から，自分が安心だと考える個人情報の処理方法に対応する保護ポリシを選択し利用することができる．図 1 にプログラム変換モデルの概要を示す．

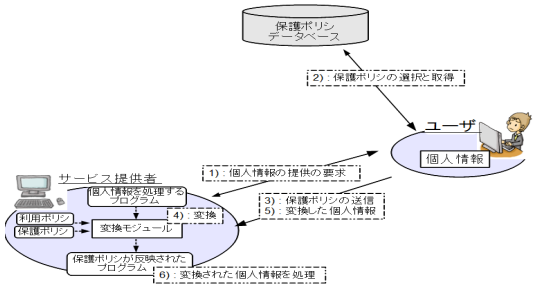

図 1: プログラム変換モデル

1. ユーザはサービス提供者にサービスを要求する．この時，サービスの利用条件としてサービス提供者は利用者に対して個人情報の提供を要求する．しかし，ユーザはサービス提供者を信頼できず，個人情報を与えることに不安に感じる．そこで，ユーザは自分で処理方法を決定したいことをサービス提供者に伝え，処理方法を決定するために利

用ポリシが必要であることを伝える．そこで，サービス提供者は利用ポリシをユーザに送信する．

2. ユーザは利用ポリシにより，個人情報を処理するプログラムでどのような個人情報がどう処理されているかを知ることができる．そこで，ユーザは保護ポリシデータベースにアクセスし，利用ポリシに定義された処理方法に合致する保護ポリシの一覧を得る．ユーザはその中から最も安心できる処理方法が定義された保護ポリシを選択・取得する．
3. ユーザは選択した保護ポリシをサービス提供者に送信する．
4. サービス提供者は変換モジュールに個人情報を処理するプログラムと保護ポリシ，利用ポリシを与えることで保護ポリシが反映されたプログラムを生成する．
5. ユーザは保護ポリシに従い守りたい個人情報を変換し，変換した個人情報をサービス提供者に送信する．
6. サービス提供者は，ユーザから送られてきた個人情報を変換されたプログラムで処理する．その後，サービスをユーザに提供する．

これにより，ユーザは個人情報を安全だと思う処理方法でサービス提供者に処理してもらうことが可能となる．

3 利用ポリシ

利用ポリシではユーザの個人情報をプログラム中でどのように利用するかが定義される．ユーザの個人情報はプログラム中の変数に格納され，その変数に対して何らかの処理が行われる．このため，ユーザが指定した処理方法をプログラムに反映させるためには，ユーザの個人情報が格納される変数と，その変数に対して適用する処理を知る必要がある．また，サービスはユーザとサービス提供者の間の通信 (ユーザからサービス提供者への個人情報の送信，サービス提供者からユーザへのサービス提供など) によって実現される．このため，ユーザの個人情報を守るためにユーザとサービス提供者間，また，それ以外の第 3 者との通信を制御することが必要となる．そこで，利用ポリシを図 2 のように定義した．

<INFORMATION> *value*
<OPERATION> *operation*
<SEND> *method*
<RECEIVE> *method*

図 2: 利用ポリシ

<INFORMATION> ではプログラム中でユーザの個人情報を利用する変数を定義する．<INFORMATION> で定義された変数を調べることで，個人情報を守るためにどの変数を変換対象とすればよいかを知ることができる．また，個人情報を処理するための方法は <OPERATION> で定義される．<OPERATION> によって，個人情報に対してどのような処理が適用されるか知ることができ，<OPERATION> を参考にユーザは保護ポリシを選択することができる．<SEND> と <RECEIVE> はデータを送受信するために利用されるメソッドが定義される．ここで定義されたメソッドを利用することで，ユーザとサービス提供者間の通信を制御することができる．

4 保護ポリシ

保護ポリシでは，その保護ポリシが実現する処理方法とそれを実現するためのプログラム変換ルールを定義する．そのため，保護ポリシがどの情報のどの処理に対して変換するのかを定義する必要がある．また，保護ポリシは利用者が選択するため，保護ポリシの説明の定義が必要となる．保護ポリシの構成を図 3 に示す．

<EXPLANATION> 説明文
<INFORMATION> *information*
<OPERATION> *operation*
<PROGRAMCONVERSIONRULES>

図 3: 保護ポリシ

保護ポリシは利用者によって選択される．そのため，保護ポリシに保護ポリシについての説明を定義すべきである．そこで，保護ポリシの説明を<EXPLANATION>で定義する．<EXPLANATION>は自然言語で書かれ，利用者は<EXPLANATION>を見ることによって，保護ポリシで定義された処理方法が安心できるものであるかを判断できる．<INFORMATION>と<OPERATION>は機械が理解できる形式で示す．<INFORMATION>は保護ポリシが対象とする情報を定義する．また，<OPERATION>は<INFORMATION>に定義された情報を対象とする操作を定義する．これらのタグは，利用者がプログラムに適用できる保護ポリシを絞り込むことを可能とする．<PROGRAMCONVERSIONRULES>はプログラムに処理方法を適用するために定義する．<PROGRAMCONVERSIONRULES>の内容のサブルールを表 1 に示す．

表 1: プログラム変換ルール

ルール名	ルールの説明
データ変換ルール	データを変換するルール
操作変換ルール	操作を変換するルール
送信制御ルール	変換前/後のデータの送信を制御するルール
受信制御ルール	変換前/後のデータの送信を受信するルール

本モデルは利用者が選択した処理方法によって個人情報を処理することを目的とする．そのため，個人情報を守るためにデータを変換するためのルールが必要となる．このようなルールを<DATACONVERSIONRULE>として定義する．これにより，利用者はサービス提供者から情報を守ることができる．しかし, 変換された情報はサービス提供者が持つプログラムで処理することができない．そのため，ある操作を変換された情報が処理可能な操作に変換するようなルールが必要となる．このようなルールを<OPERATIONCONVERSIONRULE>とし

て定義する．<OPERATIONCONVERSIONRULE>によって変換された操作は<DATACONVERSIONRULE>によって変換された情報を処理することを可能とする．また，ネットワークサービスでの利用を考えると，元の情報および変換した情報を送受信するものである．元の情報の送信は制御され，変換された情報はサービス提供者へ送られる必要がある．送受信の情報の順序はインターネットのプロトコル (特に，セキュリティ関連) では重要となる．このようなルールを <SENDCONTROLRULE> と <RECEIVECONTROLRULE> に定義する．これらのルールはサービス提供者と利用間のデータフローを制御する．利用者は保護ポリシに定義された方法をこれらのルールに従ってプログラムにインストールことができる．

5 プログラム変換方法

本システムでは，利用者によって選択された個人情報の処理方法は利用者が選択した保護ポリシと利用ポリシに従ってプログラムにインストールされる．インストールはプログラム変換モジュールによって行われる．変換モジュールの流れを図 4 に示す．

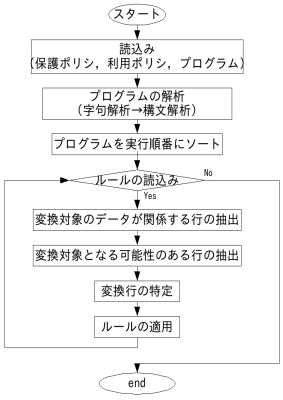

図 4: 変換モジュールの流れ

変換プログラムは保護ポリシ，利用ポリシ，サービス提供者が個人情報を扱うプログラムを引数として読込む．保護ポリシを読込みことで，変換対象となる情報，操作，プログラムの変換方法 (プログラム変換ルール) がわかる．また，利用ポリシを読込むことで，保護ポリシが変換対象とする情報が個人情報を扱うプログラムでどの変数に格納されて処理されているかや情報の送受信方法を知ることができる．利用ポリシで定義された変数を見ることでプログラム中のどの変数に守りたい情報が格納されているか知ることができる．この変数がデータ変換の対象となる．しかし，プログラム上で変数は複数の箇所で利用されている．そのため，どこでデータを変換すればいいかを決定する必要がある．そこで，適用箇所を特定するためにプログラムを解析する．守りたい情報は利用ポリシに定義された変数に格納されている．このため，字句解析により変数が関係する行を特定する．また，プログラム中で変数に格納されたデータは代入やメソッドに引数として渡すなどの処理が行われる．そのため，データの流れを追跡する必要がある．そこで，分解した字句を解析することで，変数が何を示すか，また，各行が何を示すかを意味付けする必要がある．そこで，構文解析を行う．これにより，字句解析された字句や文をプログラム上で意味づけできる．これらの解析を行うことでデータの流れを追跡する．

次に，プログラムをソートする．これにより，データの流れが実際にどう行われているかを知ることができる．また，実行状態のデータの流れがわかるため，変換後に正常に動作するかも確かめることができる．他にも，複数のクラスフィイルからなるプログラムを 1 つのファイルにまとめることで，データの追跡がしやすくなる．

その後，ルールの適用箇所の候補を抽出する．ソートされたプログラムからデータが関係する行や格納される変数が関係している行をすべて抽出することで，適用箇所の候補を抽出することができる．そこから，変換対象となる可能性のある行を抽出する．そのため，適用すると正常に動作しない行やデータが何らかの処理が行われた後の行などの変換しても意味のないところを削除する．変換対象となる行が複数ある場合は順番付けや絞込みルールを定義し，適用箇所を 1 つに絞り込む．絞り込んだ行にプログラム変換ルールを適用することで変換プログラムを生成する．

上記の処理を保護ポリシに書かれたルールがすべて適用されるまで繰り返す．ルールをすべて適用することにより保護ポリシが反映されたプログラムを生成することができる．その後，仮想上で生成されたプログラムをテスト実行させる．正しく実行されれば，変換を終了する．しかし，実行されなかった場合は，実行時のエラーに関わったルールの適用対象の箇所が複数存在するか調べて，複数存在する場合は，適用箇所を変えてルールを適用し，再びテスト実行させる．すべてエラーの場合または複数存在しない場合は，変換エラーを返す．

5.1 ポリシの情報の結びつけ

プログラムの変換の最初のステップでは保護ポリシと利用ポリシを解釈する．保護ポリシの<INFORMATION>には保護対象となる個人情報が定義されている．一方，利用ポリシの<INFORMATION>にはプログラム中で個人情報が格納される変数が定義されている．そのため，変換モジュールはこれらの<INFORMATION>に従って利用者の個人情報とプログラム中の変数を結びつける．ここで結びつけられた変数が <DATACONVERSIONRULE> による変換対象となる．これによって変換モジュールは変換する変数と変換する方法を結びつけることが可能となる．

また，保護ポリシの <OPERATION> には変換対象とする操作が定義されている．一方，利用ポリシの <OPERATION> にはプログラム中での操作を行

うためのメソッド名が定義されている．そのため，<INFORMATION>と同様に，<OPERATION> に従って操作とプログラム中のメソッド名を結びつける．これによって，操作と操作を変換するルールを結びつける．同様に，利用ポリシの <SEND> と保護ポリシの <SENDCONTROLRULE>，利用ポリシの <RECEIVE> と保護ポリシの <RECEIVECONTROLRULE> を結びつける．

5.2 プログラムの解析

Eclipse に導入されている ASTParser[3] を字句解析と構文解析のために用いる．ASTParser はプログラムを字句を解析し，構文木を生成する．このように，変換モジュールは行を (もっとも小さい役に立つ原始単位の) 字句に分割することができ, これらを文法上の構造に分類する．これにより，変換モジュールはそれぞれの字句の意味づけとこれらの関連付けが知ることができる．これは変換モジュールが利用ポリシに定義された情報と処理を見つけることを可能になり，プログラムで利用者の個人情報を追跡することを助ける．

5.3 プログラムをソート

構文解析の結果を使い，変換モジュールは複数の Java ファイルを 1 つのファイルに結合し，メソッド間で利用者の個人情報を利用するためにつかわれている異なる変数名を同一のものに統一する．そして，利用者の個人情報が影響する行だけを抜き出す．例えば，図 5 のサンプルプログラムから password が影響する行を抜き出すと図 6 のようにある．

```
1.  public class sample{
2.    public static void main(String[] args){
3.      String id;
4.      String password;
5.      Input input = new Input();
6.      Test test = new Test();
7.
8.      id = input.getID();
9.      password = input.getPassword();
10.     test.test(id, password);
11.   }
12. }
```

```
1.  public class Test{
2.    public void test(String data1, String data2){
3.      String sampleID = "abc";
4.      String samplePassword = "1234";
5.
6.      if(sampleID.equals(data1)){
7.        if(samplePassword.equals(data2)){
8.          System.out.println("OK");
9.        }
10.     }
11.   }
12. }
```

図 5: サンプルプログラム

ここでは，利用者が守りたい情報をパスワード (password) としている．Sample クラス (図 6 上) 中で password が Test.text メソッドの引数として渡されており，Test.test 中では password が data2 という変数に格納されている．このため，Sample.java の 2，7 行目の data2 という変数を password という変数に置き換える．その結果，データが影響する行は Sample.java の 4，9，10 行目と Test.java の 3，8 行目となる．さらに，samplePasswprd は Test.java の 7 行目で data2 と一緒に処理されている．そのため，samplePassword が関係する行も抜き出す．その結果，図 6 のように抽出できる．その後，変換モジュールはプログラム変換ルールが適用候補箇所を抽出する．例えば，Sample.java の 4，10 行目と Test.java の 2，4 行目のような宣言文やメソッドの呼び出し，samplePassword の利用や値の変化がない行を削除する．その結果，Sample.java の 9 行目と Test.java の 7 行目にプログラム変換ルールの適用箇所が絞り込まれる．最終的に，プログラムの Sample.java の 9 行目か Test.java の 7 行目の password が関係する行をプログラム変換ルールに従って変換する．

```
Sample.java:4.   String password;
Sample.java:9.   password = input.getPassword();
Sample.java:10.  test.test(id, password);
Test.java:2.   public void test(String data1, String password){
Test.java:4.   String samplePassword = "1234";
Test.java:7.   if(samplePassword.equals(password)){
```

図 6: password が関係する行

変換モジュールはすべてのプログラム変換ルールの適用が終わるまでこれらの処理を繰り返す．すべてのプログラム変換ルールを適用する保護ポリシに定義された処理方法がインストールされる．

6 おわりに

本研究では，プログラム変換モデルを実現する課題の一つであるプログラムの変換を実現するためにプログラム変換方法と保護ポリシ，利用ポリシ，プログラム変換ルールについてを検討した．その結果，プログラム変換の流れを示し，プログラム変換には字句解析や構文解析，プログラム上でのデータの遷移を追跡する必要があることを示した．また，保護ポリシと利用ポリシの定義を示した．今後の課題として，プログラム変換ルール，絞込み時のルールの定義を検討する必要があることが挙げられる．

謝辞

本研究は科学研究費補助金（23700098）の支援を受けて行なった．

参考文献

[1] NPO 日本ネットワークセキュリティ協会, 2009 年情報セキュリティインシデントに関する調査報告, http://www.jnsa.org/result/incident/2009.html

[2] フィッシング対策協議会，月次報告書，http://www.antiphishing.jp/report/monthly/

[3] Eclipse の ASTParser を試す, http://www.ibm.com/developerworks/jp/opensource/library/os-ast/