自己消去プログラムによる情報漏洩防止システム

石川　達大*　小高 知宏*　黒岩 丈介*　白井 治彦**

Information leakage prevention system by self-clearing program

Tatsuhiro ISHIKAWA*,　Tomohiro ODAKA*, Jousuke KUROIWA* and Haruhiko SHIRAI**

(Received January 31, 2019)

In this paper, we proposed a new method to prevent information leakage which is not found in conventional information leakage prevention systems. Using the method of the new information leakage prevention system, the usefulness of the system was discussed.

In recent years, information systems have been used in various places such as personal computers and smart phones due to the spread of the Internet. However, even if companies and organizations take measures against information leakage, information leakage remains unchanged at present.

In this research, we propose new information leakage prevent method using the self-clearing program. To attach the self-clearing program to the confidential information itself, the self-clearing program can delete the confidential information if information leakage occurs.

We conducted an experiment on whether the information leakage prevention system by the self-clearing program actually operates, and examined the usefulness of whether it is possible to prevent information leakage in a new method.

Key words :*Confidential information*

1. はじめに

近年、情報システムはインターネットの普及によりパソコンやスマートフォン等様々なところで利用されている。[1] また、IOT[2] の進歩により、身の回りのあらゆるモノがインターネットにつながるようになった。例えば、家庭用ゲーム機や家電製品、カーナビ、医療の分野まで広がっている。そのため、情報システムは社会生活になくてはならないところまで浸透している現代になっている。

しかし、各企業や団体等で情報漏洩対策をしても情報漏洩が跡を絶たないのが現状である。なぜ、情報漏洩が跡を絶たないのか、それは新しい攻撃・ウイルス等が攻撃者によって考えられたり、組織内部の人間による情報漏洩が発生しているからと考えられる。攻撃者は、セキュリティ対策を固めてもそこから抜け穴を見つけ出し攻撃する。それを防止したとしても、また次の抜け穴を見つけ出し攻撃してくる。

そのため、セキュリティ対策を施してもその度に突破されてしまい、完璧な情報漏洩防止システムを作成することは難しいと考えられる。また、組織内部の人間による情報漏洩は、誤送信や管理ミス、置き忘れなどヒューマンエラーであるため、DLP[3] や注意喚起、閲覧できるサイトを制限して防ぐことしか出来ない。しかし、ヒューマンエラーであるためすべてをなくすことは出来ず、対策を講じるのが難しいと考えられる。

そこで、本研究での目的は、機密情報の漏洩をすべて防ぐことが出来ないという前提をもとに、機密情報が外部に出た場合に情報漏洩を防止するシステムの開発・実装を行う。

本研究での目的を果たすために、自己消去プログ

* 大学院工学研究科 知能システム工学専攻
** 工学部技術部
* Human and Artificial Intelligence Systems Course, Graduate School of Engineering
** Technical Division

ラムによる情報漏洩防止システムを開発した。[4] このシステムは、例えばハッキングなどによる情報漏洩や組織内部の人間による情報漏洩などで機密情報が許可されたパソコンから外部に出た際に、機密情報自体を消去することによって情報漏洩を防止するシステムである。

本論文では、2 章で自己消去プログラムによる情報漏洩防止システムの詳細について述べる。3 章で自己消去プログラムによる情報漏洩防止システムの動作実験の方法、動作例、結果について述べる。4 章では、自己消去プログラムによる情報漏洩防止システムの考察、そして情報漏洩対策における新しい手法の提案を述べる。

2. 自己消去プログラムによる情報漏洩防止システム

本章では、自己消去プログラムによる情報漏洩防止システムについて述べる。自己消去プログラムとは、自分自身を消去するプログラムである。この自分自身を消去するという性質を利用して、機密情報と合わせることにより機密情報自体を消去する。機密情報自体を消去することによって、情報漏洩を防止するシステムを作成し、実装する。

2.1 自己消去プログラムによる情報漏洩防止システムの構成方法

本説では、自己消去プログラムによる情報漏洩防止システムの構成について述べる。自己消去プログラムとは、自分自身を消去するプログラムである。この自分自身を消去するという性質を利用して、機密情報と合わせることにより機密情報自体を消去する。機密情報自体を消去することによって、情報漏洩を防止するシステムである。機密情報自体を消去するかどうかは、許可されたパソコンか許可されていないパソコンかで処理が分かれている。条件判定としては、パソコン固有の情報を用いる。パソコン固有の情報を用いることによって、機密情報を見ることを許可されたパソコンか許可されていないかを判断する。

許可されたパソコンの場合の自己消去プログラムによる情報漏洩防止システムが、機密情報を編集できるまでの流れを以下に示す。

1. 自己消去プログラムの中に機密情報を埋め込む
2. 許可されたパソコンに機密情報が存在する
3. 許可されたパソコンでプログラムを実行する
4. 許可されたパソコンまたは許可されていないパソコンかを判断する
5. 自分のパソコンの場合、機密情報が開かれ編集することが出来る

許可されていないパソコンの場合の自己消去プログラムによる情報漏洩防止システムが機密情報自体を消去するまでの流れを以下に示す。

1. 自己消去プログラムの中に機密情報を埋め込む
2. 攻撃者に機密情報が抜き取られる、組織内部の人間による誤送信や管理ミス、紛失・置き忘れなどにより間違った相手に機密情報を送ってしまう
3. 許可されていないパソコンでプログラムを実行する
4. 許可されたパソコンまたは許可されていないパソコンかを判断する
5. 許可されていないパソコンの場合、機密情報自体を消去する

自己消去プログラムによる情報漏洩防止システムは上記のような流れのシステムである。許可されたパソコンと許可されていないパソコンで処理が分かれている。許可されたパソコンか許可されていないパソコンかを IP アドレスによって判断している。

IP アドレスとは、インターネット上に接続された機器が持つナンバーのことである。データをやり取りする際、ネットワーク上で通信相手を間違わないようにするために使われる。IP アドレスには種類やルールが存在する。IP アドレスは、ネットワーク上の機器を識別するために割り当てられているため、インターネット上での住所のような役割をしている。

自己消去プログラムがどのように機密情報自体を消去するのかのフローチャートを図 1 に示す。

まず、自己消去プログラムが実行されるかどうかで処理が分かれる。自己消去プログラムが実行されなかった場合、機密情報自体を見ることが出来ない為機密情報は守られている。自己消去プログラムが実行された場合、実行されたパソコンの IP アドレスを取得する。次に、取得した IP アドレスが許可されたパソコンのものかどうかで処理が分かれる。IP アドレスが許可されたパソコンのものだった場合、機密情報編集システムへ移行する。許可されていないパソコンの場合、自己消去処理を行い、機密情報自体を消去する流れに移行する。

2.2 機密情報編集システム

本説では、機密情報編集システムについて述べる。

知られたくない情報が書いていないソースファイルを用意する。ここで知られたくない情報は、機密情報の中身とする。まず、知られたくない情報が書いていないソースファイルをコンパイルする。コンパイルして出来た実行ファイルを実行する。次に、機密情報編集を許可されているかの条件判定に入り、次の処理へ進む。次の処理としては、gedit やメモ帳を新規に立ち上げて開く。そこで、機密情報を編集し閉じた場合、次の処理に移る。機密情報の中身を知られたくない情報が書いていないソースファイルの機密情報の配列の行を書き換える。この処理が終わった時点では、知られたくない情報、機密情報の中身が入った状態のソースファイルとなっている。次に、機密情報の中身が入った状態のソースファイルをコンパイルし、機密情報の中身が入った実行ファイルを作成する。これと同時に、機密情報のファイルを消去する。機密情報の中身が入った実行ファイルが作成された時に、機密情報のファイルを消去することによって前回編集した機密情報が実行ファイルの中に残っている状態になっている。次に、機密情報の中身が入った状態のソースファイルの機密情報が入った配列の行を消去する。そして、機密情報が空の状態の配列に書き換える。この処理を行うことによって、機密情報の配列の中身が空のソースファイル、元のソースファイルに戻すことが出来る。そのため、ソースファイルを抜き取られり、誤送信などで送ってしまった場合でも、機密情報が簡単に見ることができず、漏洩しない状態となっている。上記のような手順によって前回編集・保存した機

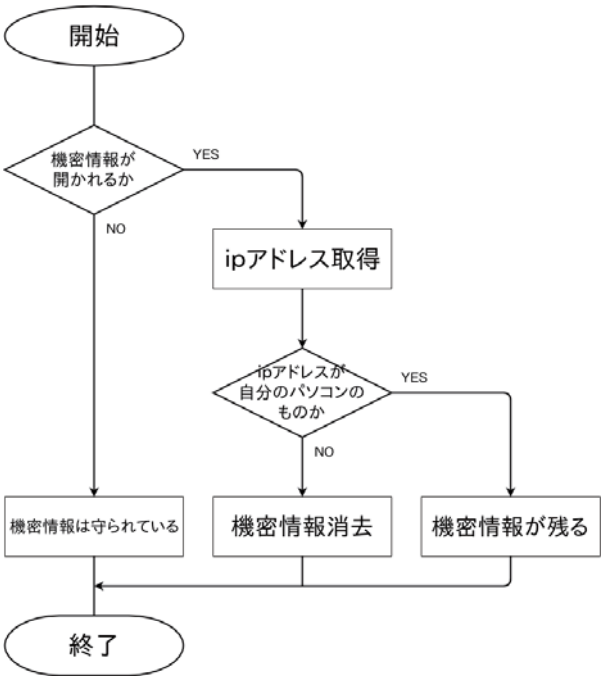

図 1 自己消去プログラムの流れ

2.3 java の自己消去プログラム

本説では、java 言語の自己消去プログラムについて述べる。

java 言語の自己消去プログラムは、実行環境に依存しない自己消去プログラムである。実行環境に依存しなければ、自己消去プログラムは確実に実行することができ、機密情報を見られる可能性を低くすることが出来ると考え、実行環境に依存しない java 言語での自己消去プログラムを作成した。

java の自己消去プログラムを作成する上で問題なのは、機密情報編集システムである。機密情報編集システムでは、OS のコマンドを用いて実行する。OS のコマンドは、OS ごとに違うため処理が OS ごとに分かれる。java の自己消去プログラムのフローチャートを図 2 に示す。

まず、自己消去プログラムが実行された場合、IP アドレス取得の処理に入る。取得した IP アドレスが許可されたパソコンのものかどうかで処理が分かれる。許可されていないパソコンの場合、機密情報消去の処

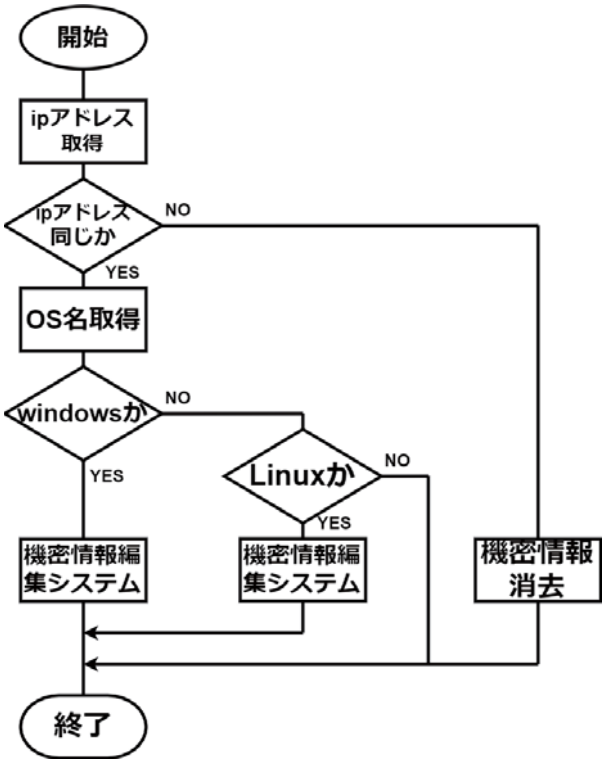

図 2 java の自己消去プログラムフローチャート

理に入る。機密情報消去の処理では、java 言語のプログラムで実行することが出来るため、OS に関係なく機密情報自体を消去することが出来る。IP アドレスが許可されたパソコンのものの場合、OS 名取得の処理に入る。OS 名を取得した次にその取得した OS 名が Windows かどうかで処理が分かれている。OS 名が Windows の場合、機密情報編集システムの処理に入る。取得した OS 名が Windows ではない場合、次に取得した OS 名が Linux かどうかで処理が分かれている。OS 名が Linux の場合も機密情報編集システムの処理に入る。OS 名が Window、Linux でない場合は、機密情報消去の処理に入る。

上記のような流れで java の自己消去プログラムは動作する。ip アドレスが許可されたもので、自己消去プログラムが実行されたパソコンが Windows、Linux の場合は、機密情報編集システムの処理に入り機密情報を編集することが出来る。

ip アドレスが許可されていないものまたは、ip アドレスが許可されていても自己消去プログラムが実行されたパソコンが Windows、Linux でない場合は、機密情報を消去する。機密情報自体を消去することによって情報漏洩を防止するシステムである。

2.3.1 Linux での実現

本説では、OS が Linux の場合の java での自己消去プログラムについて述べる。java の自己消去プログラムを実行した際、機密情報編集システムが OS ごとによって処理が分かれている。OS ごとに処理が分かれているため、それぞれの処理をフォルダに分けた。OS が Linux での処理があるプログラムをまとめたフォルダが Linux フォルダである。Linux フォルダの内容を図 3 に示す。

OS が Linux である場合のそれぞれのプログラムについて述べる。

gedit.java は、Linux で機密情報を編集するためのプログラムである。Linux では、機密情報を編集する場合、gedit を用いる。そのため、gedit.java では、gedit を起動して編集・保存して閉じるまでの処理が書かれている。

sed1.java、sed2.java、sed3.java は、Linux の sed コマンドを用いるためのプログラムが書かれている。sed の s コマンドを用いることによって、文字列を全置換したり、行単位で抽出したり、削除したり、様々なテキスト処理が出来る。このコマンドを用いることによって、ソースファイルの sample.java の中身を置換することによって、機密情報編集システムを構成している。

sed1.java では、gedit で編集し終わった機密情報の中身を取得し、ソースファイルの sample.java の機密情報を格納する配列の行を書き換えている。

sed2.java、sed3.java では、機密情報が簡単に見られないようにするためのプログラムである。sed1.java によって、ソースファイルの sample.java の機密情報を格納する配列の行を書き換えているため、gedi t で編集し終わった機密情報の中身がそのままソースファイルの sample.java に残っている。そのため、ソースファイルの sample.java を抜き取られてしまった場合、簡単に機密情報を見ることができ、すぐに機密情報が漏洩してしまう。そこで、sed2.java により、ソースファイルの sample.java の機密情報が格納されている行を消去し、sed3.java によって機密情報が格納されてない元の空欄の行を挿入する。上記のような処理をすることによって、sed1.java によってソースファイルの sample.java の機密情報を格納する配列の行を書き換えて、sed2.java、sed3.java によってソースファイルの機密情報を格納する配列が空欄の元の行に戻す処理を行う。この 2 つの処理を行うことによって、ソースファイルの sample.java が抜き取られてしまった場合でも、機密情報が簡単に見れてしまうことを防いでいる。

compairu.java は、2 回目のコンパイルを行うプログラムである。sed1.java によって、gedit で編集し終わった機密情報を取得するが sed2.java、sed3.java によって元のソースファイルに戻してしまう。そのため、機密

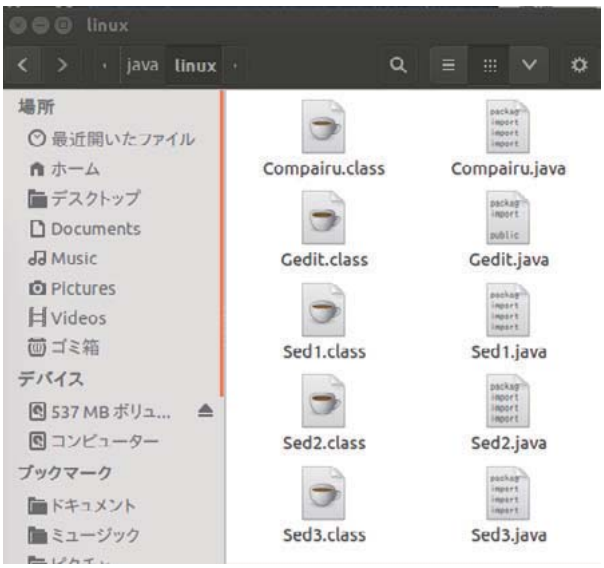

図 3 Linux フォルダの中身

情報がどこにも残らないので自己消去プログラムの2回目の実行で機密情報が新しくなり、前回編集した内容が見れなくなってしまう。そこで、compairu.java によって2回目のコンパイルを行うことによって、2回目に実行した場合、前回の機密情報を見れるようなプログラムになっている。2回目のコンパイルのタイミングは sed1.java によって gedit から機密情報を取得し配列に格納した後にコンパイルを行う。このタイミングでコンパイルすることによって、機密情報が格納された配列が書かれているソースファイルの sample.java ごとコンパイルするため、実行ファイルの中に機密情報が残る形になっている。その後に sed2.java、sed3.java によってソースファイルの sample.java に書かれている機密情報を格納する行を消去することによって簡単に機密情報を見ることを出来なくし、タイミングよくコンパイルすることによって前回の機密情報が2回目の実行の時に見れる形になっている。

OS が Linux の場合は上記のような内容で、プログラムが動作することによって自己消去プログラムを実行している。

2.3.2 Windows での実現

本説では、OS が Windows の場合の java での自己消去プログラムについて述べる。java の自己消去プログラムを実行した際、機密情報編集システムが OS ごとによって処理がわかれている。OS ごとに処理が分かれているため、それぞれの処理をフォルダに分けた。OS が Windows での処理があるプログラムをまとめたフォルダが Windows フォルダである。その内容を図 4 に示す。

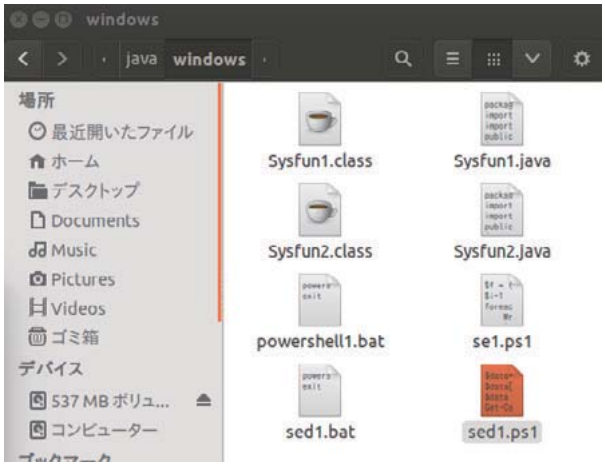

図 4 Windows フォルダの中身

OS が Windows である場合のそれぞれのプログラムについて述べる。

sysfun1.java は、Windows で機密情報を編集するためのプログラムである。Windows では、機密情報を編集する場合、notepad・メモ帳を用いる。そのため、notepad を起動して編集・保存して閉じるまでの処理が書かれている。

powershell1.bat、sed1.bat、se1.ps1、sed1.ps1 は、機密情報を書き換える処理である。Linux の場合、sed コマンドがあるため、ファイルの中の文字列を書き換えることが容易にできた。しかし、Windows では sed コマンドのような容易に文字列を書き換えるコマンドが無いため、powershell を用いて機密情報の中身を書き換えている。

powershell とは、マイクロソフトが開発した拡張可能なコマンドラインインターフェースシェル及びスクリプト言語である。powershell は Windows7 から有効になった機能であり、コマンドプロンプトよりもさらに強力で、スクリプトを書く為の言語として導入されている。powershell を用いることによって、コマンドプロンプトより様々な処理が出来るようになり、ファイルの中身を書き換えることが出来る為、powershell を用いている。

powershell1.bat、sed1.bat では、スクリプトファイル se1.ps1、sed1.ps1 を別端末で実行させるためのファイルである。se1.ps1、sed1.ps1 は、powershell でしか実行することが出来ないため、あらかじめ powershell に入って se1.ps1、sed1.ps1 を実行しなくてはいけない。しかし、すべて自動で自己消去プログラムを実行するために powershell1.bat によってコマンドプロンプトから se1.ps1、sed1.ps1 を実行させている。それぞれのスクリプトファイルを実行した後に新しい端末を閉じるための処理が書かれている。

se1.ps1 は、機密情報の中身を配列に格納する処理が書かれている。se1.ps1 では機密情報のテキストファイルから機密情報を取得し、機密情報を格納する配列の行のみ書き換えてそれを別のファイル sample1.java に保存する。次に sample1.java のファイルの中身全部を Sample.java に上書きを行い、sample1.java を消去する処理を行っている。se1.ps1 は Linux での sed1.java と同じような処理を行っている。

sed1.ps1 は機密情報を簡単に見れないようにするための処理が書かれている。Linux と同様に Windows でも機密情報の中身を取得しソースファイルの sample.java に書き換えている。そのため、ソースファイルの sample.java を抜き取られてしまった場合、簡単に機密情報を見ることができ、すぐに機密情報が漏

洩してしまう。sed1.ps1 では、sample.java の機密情報が格納されている配列の行を消去し、機密情報が格納されてない元の空欄の行を挿入している。sed1.ps1 は Linux での sed2.java、sed3.java と同じような処理が書かれている。

sysfun2.java は Linux の場合と同様に 2 回目のコンパイルを行うためのプログラムが書かれている。Linux と Windows の場合でコンパイルの仕方が違うため、Windows ではオプションをつけてコンパイルしている。Windows では-encoding UTF-8 というオプションをつけることによってファイル処理における文字化けがなくなるようになっている。Windows でのコンパイルのタイミングは、se1.ps1 によって機密情報の中身を取得しソースファイルの Sample.java を書き換えた後に 2 回目のコンパイルを行う。そして、sed1.ps1 によって、ソースファイルの Sample.java の機密情報が格納される配列の行を機密情報が格納されていない元の空欄の行を挿入している。

OS が Windows の場合は上記のような内容で、プログラムが動作することによって自己消去プログラムを実行している。

3. 実験結果

本章では、2 章で述べた java の自己消去プログラムが実際に動作するかの 2 つの動作実験の方法について述べる。次に、動作実験によって得られた動作例、結果について述べる。

3.1 実験方法

java の自己消去プログラムでは、環境に依存しない自己消去プログラムの作成を目的とした。動作実験 1 として、java の自己消去プログラムを許可されたパソコンで、OS が Windows、Linux の場合、実際に動作し機密情報が表示され編集できるのかの実験を行う。次に 2 回目以降を実行し、前回編集・保存した機密情報の内容が保存されていて機密情報を表示できるのかの確認を行う。動作実験 2 として、自己消去プログラムによる情報漏洩防止システムを USB やメールなどを用いて、OS が Windows、Linux 以外のパソコン・許可されていないパソコンへ移す。OS が Windows、Linux 以外の場合と許可されていないパソコンの場合、実際に動作し機密情報自体が消去されるかの実験を行う。

3.2 動作例

本説では、java の自己消去プログラムの動作例・結果について述べる。java の自己消去プログラムの動作例を図 5 に示す。

図 5 から図 7 では、OS が Windows であり、IP アドレスが許可されたパソコンでの画面である。

図 5 では、IP アドレスが許可されたパソコンのもので、OS が Windows である場合にメモ帳で機密情報 kimitu.txt を開いている画面である。まず Sample.java をコンパイルしている。OS が Windows である場合、文字コードが異なる為オプションにて文字化けしないようにしている。許可されたパソコンの IP アドレスは、192.168.56.1 である。そして、取得した IP アドレスは 192.168.56.1 である。設定された IP アドレスと取得した IP アドレスが同じである為、次に OS 名を取得する処理に移っている。取得した OS 名が Windows であるため、OS が Windows の場合の処理に移っている。OS が Windows である場合の機密情報編集システムが実行されメモ帳によって機密情報と仮定した kimitu.txt が開かれるようになっている。メモ帳によって kimitu.txt が立ち上がっている間は、kimitu.txt は Windows フォルダ内に表示されている。図 5 の動作時にメモ帳の kimitu.txt の中に「機密情報」という文字列を書き込み保存している。

図 6 は、メモ帳を閉じた後の画面である。メモ帳を閉じた後に前回編集した内容を保存するために機密情報編集システムの処理を行っている。機密情報編集システムの中で Windows フォルダ内の kimitu.txt を消去する処理に移っている。kimitu.txt を消去することによって機密情報が残らない形になっている。

図 7 は、2 回目の実行の画面である。2 回目の実行の際も ip アドレスが許可されたパソコンのものか、OS 名が Windows であるかを条件判定している。2 回目の実行の際、機密情報編集システムによって前回編集・保存した機密情報がメモ帳を用いて kimitu.txt 内

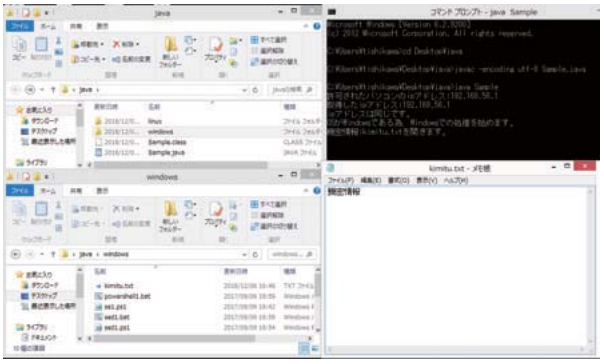

図 5 機密情報を開いた画面

に表示されている。機密情報編集システムによって2回目以降前回編集・保存した機密情報が残る形になっている。

図5から図7のようにOSがWindowsであり、IPアドレスが許可されたパソコンの場合、実際に動作し機密情報を編集することが可能であり、2回目以降の実行の際も前回編集・保存した機密情報を表示することが出来た。

IPアドレスが許可されたパソコンでOSがLinuxの場合でも同様の結果を表示することが出来た。

次に、javaの自己消去プログラムが許可されていないパソコンで実行された場合を示す。

図8、9は許可されていないパソコンでjavaの自己消去プログラムを実行した図である。ipアドレスが許可されていないパソコンのため機密情報自体を消去している。USBなどを用いてjavaの自己消去プログラムを移動し実行している。

図8では、許可されていないパソコン上でソースファイルSample.javaをコンパイルしている画面である。端末の名前が違うため、許可されていないパソコンと分かる。

図9は、許可されていないパソコン上でjavaの自己消去プログラムを実行している結果画面である。javaの自己消去プログラムを実行したパソコンが許可されいないパソコンのため機密情報自体を消去している図である。許可されたパソコンのIPアドレスとして設定した値は192.168.111.10である。次に取得したIPアドレスの値は192.168.111.140であり、許可されていないパソコンである。設定されたパソコンのIPアドレスと取得したIPアドレスで相違があるため機密情報自体を消去する処理に移っている。javaのフォルダ内にはソースファイルSample.java、実行ファイルSample.class、windowフォルダ、Linuxフォルダが存在している。2つのファイルはそのまま消去し、2つのフォルダはフォルダ内のファイルごと消去している。図40からわかるようにjavaの自己消去ポログラムが実行された後のjavaフォルダ内には、すべてのファイル、フォルダが消去されていることがわかる。図のように機密情報自体を消去することで機密情報の中身を見ることが出来ないため、情報漏洩を防止している。

図8、9のようにOSに関係なく、IPアドレスが許可されていないパソコンの場合、機密情報自体を消

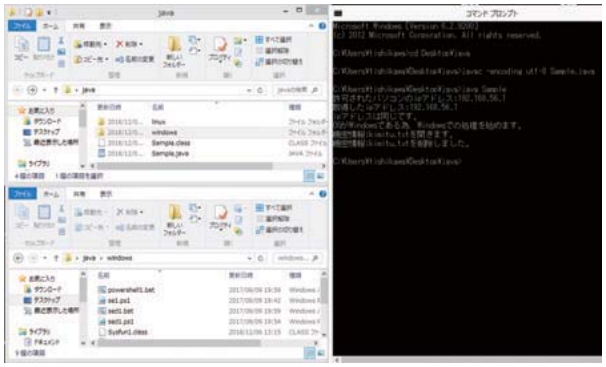

図6 機密情報を閉じた画面

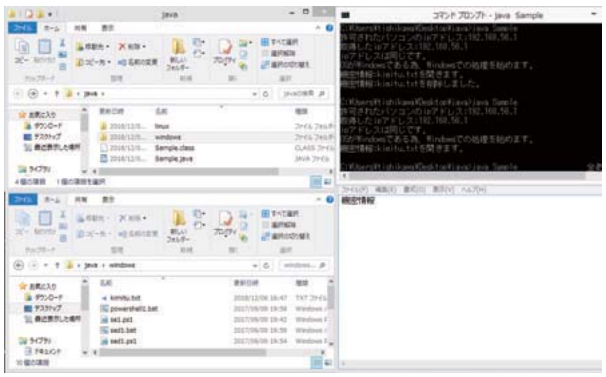

図7 機密情報を開いた画面 (2回目)

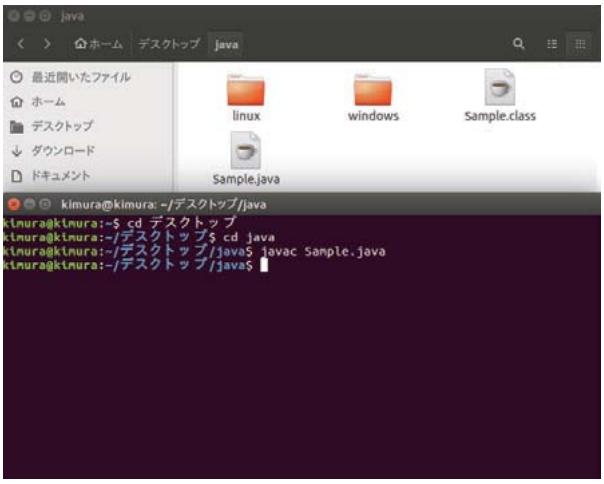

図8 消去前の画面

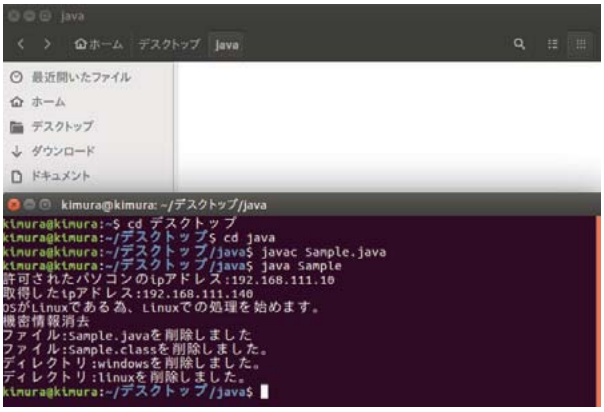

図9 消去後の画面

去することが出来た。機密情報自体を消去することによって情報漏洩を防止する結果となった。

4. 考察

本章では、java 言語を用いた自己消去プログラムによる情報漏洩防止システムについて考察を行う。

java 言語を用いることによって自己消去プログラムによる情報漏洩防止システムが許可されたパソコンかつ OS が Windows と Linux の 2 つの場合に機密情報を編集することができ、許可されていないパソコン又は OS が Windows と Linux ではない場合に機密情報自体を消去することが出来た。

機密情報編集システムをそれぞれの OS ごとで処理を分けることによって自己消去プログラムの中に機密情報を格納することが出来た為、機密情報自体を消去することが出来る。

本研究では、従来の情報漏洩防止システムにはない、機密情報が外部に出た後にも情報漏洩を防止することが出来るシステムを開発することによって、今まで情報漏洩をしてきた原因にも対処することが出来ると考える。

しかし、本手法では自己消去プログラムによる情報漏洩防止システム自体が守られていないため、逆コンパイルやコードを書き換えられた場合、機密情報が漏洩してしまう。そのため、自己消去プログラムによる情報漏洩防止システム自体を守るためにコードを書き換えづらくすることや、従来の情報漏洩防止システムと組み合わせることによって、様々な情報漏洩のケースに対応することが出来ると考える。

以上のことから、本研究で開発した自己消去プログラムによる情報漏洩防止システムは従来の情報漏洩防止システムの問題点を改善でき、情報漏洩を防止することが出来る新しい手法であるため有用性があると考える。

5. 参考文献

[1]https://iphone-mania.jp/news-201799/

[2]http://www.soumu.go.jp/johotsusintokei /whitepaper/ja/h29/html/nc133100.html.

[3]Kevin Alejandro Roundy, El Segundo,CA (US); Anand Kashyap, Pune (IN):SYSTEMS AND METHODS FOR DETECTING INFORMATION LEAKAGE BY AN ORGANIZATIONAL INSIDER,US 9,652,597 B2,May 16, 2017

[4] 石川 達大 , 小高 知宏 , 黒岩 丈介 , 諏訪 いずみ , 白井 治彦 . 自己消去プログラムによる情報漏洩防止システム . 平成 28 年度電気関係北陸支部連合大会 2016.