一時的セキュア空間作成による情報の二次流出防止

多田 政美　古市 実裕　新村 泰英

Preventing Secondary Data Leaks by Creating Temporary Secure Areas

Masami Tada, Sanehiro Furuichi and Yasuhide Niimura

依然として情報漏えいは後を絶たず社会的に大きな問題となっている．特に，近年，グローバル化の進展に伴い，海外へ業務委託する機会が増える傾向にあり，自社からの情報流出だけではなく，業務委託先からの情報流出にも対策を施すことが急務になっている．情報流出を防ぐためにさまざまな製品が流通しているが，業務委託先にそれらの製品の導入を強制することは難しく，委託先からの二次流出を防止する決定的な解法はほとんどない．本論文では，特定の製品を導入したりOSを再起動したりすることなく，アクセス制御ポリシーで指定したプロセス以外のすべてのプロセスがアクセスできないセキュアな領域（セキュア空間）を一時的に作り出し，その領域だけでデータの編集を許すことにより，情報の流出を防ぐ新たなシステムを提案する．Microsoft® Windows® 上で提案システムを実際に構築し，その有効性を示す．

Data leaks remain a serious, seemingly never-ending, social problem. Especially, in recent years, as a result of advancing globalization, there is a trend for increasing numbers of companies to outsource work overseas, and as a result it has become an urgent issue for enterprises to not only protect information internally, but also to prevent data leaks from their subcontractors. There are many commercially available security products to prevent data leaks, but in most cases enforcing the deployment of such software at subcontractors is impractical. As such, even now there are hardly any effective approaches to solving this issue. In this paper, we propose a new approach, involving the creation of a temporary secure area that does not require the installation of any software or rebooting of the operating system. In this approach, processes are unable to access any resources in the temporary secure area, apart from those processes that have been given permission under an access control policy, allowing users to edit the data only within that area. We have implemented this new system under Microsoft Windows, and the results of our evaluation demonstrate the effectiveness of our approach.

Key Words & Phrases：情報漏えい，二次流出，グローバル化，アクセス制御，動的注入
Data leak, Leaks from Subcontractors, Globalization, Access control, Dynamic injection

1. はじめに

近年，機密情報や個人情報の流出が大きな事件として報道されている．情報漏えいは企業に経済的な損失をもたらすと同時に，社会的信用の喪失に発展するなど，大きな被害となる場合が多い．情報漏えいの危険性は，企業活動のグローバル化にともないますます高まっており，発生した場合の被害もより甚大になると思われる．

提出日：2010年9月6日　再提出日：2010年12月2日

現在，多くの企業では，さまざまな取引先に業務を委託する．以前は，紙媒体で機密情報を提供することも多かったが，IT技術の浸透に伴い，現在ではほとんどの企業がデジタル・データでやり取りすることが一般的になった．その結果，膨大な量の機密情報を容易に持ち出すことが可能となり，情報漏えいに対する被害は増大する傾向にある．さらに，企業活動のグローバル化に伴い，国内の企業だけではなく，よりコストの安い海外の企業に対して業務委託することも多くなってきた．こうした海外企業の中には，セキュリティー意識の低いところもあり，業務を委託する企業では，社内での情報漏えい対策に加え，業務委託先での情報漏えいの問

題対策が急務になってきている．社内においては，シンクライアント［1］［2］によるシステム全体の再構築やリアルタイム監視・制御技術［3］［4］を応用した情報漏えい対策アプリケーション導入の義務化などが対策として挙げられるが，業務委託先が中小企業である場合，これらの対策を施すコストの負担が問題となることがある．また，対策アプリケーション導入の義務化は，既存の業務アプリケーションとの共存問題に加え，複数の企業から異なる対策アプリケーションを指定された場合，対策アプリケーション間の共存も問題となる可能性も高く現実的ではない．

本論文では，既存の IT 資産を有効に活用して効率的にクライアント PC のセキュリティーを強化する現実的な解法を示す．

2. 従来技術と問題点

セキュリティー強化のため，業務委託先企業に対してシステムの大規模な再構築や,指定したアプリケーション導入の義務化を要求することは，実際の現場では容易に受け入れられない．本章では，既存技術の課題を整理し，クライアント PC のセキュリティー強化に必要な現実的なアプローチを検討する．

2.1 シンクライアント

PC からの情報漏えいを防止するために,あらゆるデータの処理をサーバー上で実行し，エンド・ユーザーの端末では画面表示と入力デバイス処理だけを行うシンクライアントが提案されている［1］［2］．

しかし，クライアント端末がネットワークに常時接続することを前提としており，現在においては一般的になってきたが，常時接続を維持するための導入コストや運用コストの問題は相変わらず解決されておらず，大企業であれば可能性もあるが，業務委託先の企業まで含めた再構築は難しい．

2.2 Enterprise DRM （Digital Rights Management）

データとセキュリティー･ポリシーをひも付けし，アプリケーション上での操作制限や閲覧制限を行う文書管理システムが存在する．代表的なものに，Adobe® LiveCycle®［5］や Microsoft Windows Rights Management Services［6］がある．アプリケーションプログラム自身が編集権限や印刷権限などのセキュリティー・ポリシーを解釈し，プログラムの操作を制限する．

しかし，アプリケーションが文書管理システムに対応している必要があり，現状では Microsoft Office や Adobe PDF などの一部の製品に限定されている．企業で利用する業務ソフトや作り込みのアプリケーションの多くは対応しておらず，システムを導入するためには，業務ソフトの全面的な置き換えや業務形態の大幅な見直しが必要になる．

また，文書の属性としてのセキュリティー・ポリシーが定義されているだけであり，複数プログラム間の相互連携などは考慮されていない．

2.3 セキュアOS

SELinux［7］や AppArmor［8］などのセキュア OS は，管理者の設定に基づき機密情報へのアクセス制御を厳密に管理する強制アクセス制御機構を備えている．

しかし，シンクライアントや Enterprise DRM の課題と同様に，既存の Windows ベースのアプリケーションや業務ソフトを全面的に置換する必要があり，操作性も大きく変わるため，一般業務で本格的に用いるには敷居が高い．

セキュリティー・ポリシーの設定作業が煩雑となりがちという課題もある．使いやすいポリシー・エディター［9］や少ない手間による情報漏えい防止機構［10］なども提案されているが，業務委託先の企業に強制することは難しい．

2.4 セキュリティー対策アプリケーション

市販のセキュリティー対策アプリケーションを導入することで，ファイル操作などをリアルタイムに監視して，機密データなどが複製されることを防いだり，複製したことを記録としてログに残したりすることもできる．また，外部メディアなどに機密データを暗号化して保存するアプリケーションも数多く存在する［11］［12］．

アプリケーションが数多く存在するため，発注者が指定したアプリケーションを業務委託先が導入して使うとしても，複数の発注者から複数のアプリケーションを指定された場合，アプリケーション間の競合の問題となり，実現が難しい．

2.5 従来技術の課題

従来の PC のセキュリティー対策手法は，社内のセキュリティーを守るためのものであり，かつ，既存の業務ソフトや IT 資産の有効活用が難しく，業務形態の大幅な変更やユーザーの操作性の大幅な低下を招くことが多い．さらにコスト面での問題もあり，大企業でさえその

導入にためらいをみせるなか，業務委託先として多い中小企業での導入は難しいと考えられる．

既存の業務形態やIT資産を維持したまま，ユーザーが容易に扱えるセキュリティー・ポリシーを順守させる新しい手法が求められる．

3. 一時的セキュア空間の提案

本論文では特定のアプリケーションをインストールすることなく，一時的に任意のPC上の特定フォルダーを，保護領域と呼ぶセキュア空間に仕立てる手法を提案する．セキュア空間とは，アクセス制御ポリシーで指定されたプロセスのみが限定的な操作だけを行える環境のことを指す．限定的な操作とはアクセス制御ポリシーで許された操作のみを許すことである．提案手法では，Win32 APIおよびCOMインターフェースを対象にしたBinary Interception［13］［14］をベースにしたリアルタイム監視・制御技術［3］［4］を採用し，動的な監視モジュールの注入とアクセス制御ポリシーによりセキュア空間の構築を行う．

一般的な手法では，監視用のアプリケーションをインストールし，AppInit_DLLs［15］に監視用のDLLを記述することで実現しているが，本提案手法では，ユーザーによるインストールの手間や再起動を省くために，API監視用モジュールの動的注入を行っている．さらに，プロセス起動系のAPIを監視することにより，セキュア空間を作成した後に起動されるプロセスに対してもアクセス制御ポリシーを適用させ，セキュア空間を保持することができる．

3.1 アクセス制御ポリシー

アクセス制御ポリシーは各プロセスのアクセスを規定するものである．任意のプロセスに対し，セキュア空間のアクセス権を付与することができる．例えば，Windowsのメモ帳（notepad.exe）に対して，セキュア空間に読み込みと書き込みを許可したと仮定する．この場合，セキュア空間以外の場所を同時に書き込み可能にすると，セキュア空間からファイルの持ち出しが可能になるため，セキュア空間から読み出せるプロセスに対しては，セキュア空間以外に書き込み許可を出してはいけない．

Windowsではクリップ・ボードを使ったデータのコピーが可能であるが，これに対しても，セキュア空間から読み出せるプロセスに対して，コピーを可能にしてはいけない．ただし，同じプロセス内に貼り付ける場合は許可したいことが多いため，実際には，ペースト時に，セキュア空間からのデータかどうかで許可と拒絶を判断する．

セキュア空間にあるデータを実際に印刷して持ち出すことや，仮想プリンター・デバイスを用いてPDFなどに変換することも考えられるため，印刷に対しても禁止する必要がある．

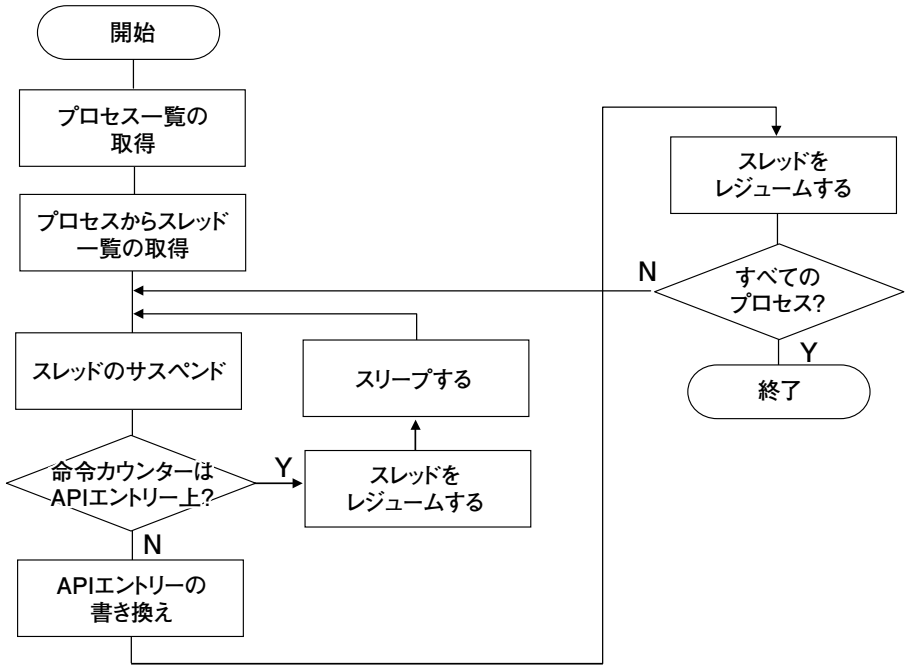

図1．動的監視モジュールの注入フロー

3.2 動的監視モジュールの注入

モジュールを注入する方法として，よく知られているのは，レジストリーの AppInit_DLLs に DLL 名を書いておくことにより，プロセスが起動されると同時に Windows が自動的にロードする仕組みである．その仕組みではプロセスが起動する前に AppInit_DLLs に DLL 名が書き込まれていなければならず，既に起動しているプロセスには注入を行うことができない．また，AppInit_DLLs を修正するためには管理者権限が必要であり，ユーザー権限では禁止されていることが一般的である．そのため，本提案手法では，AppInit_DLLs を用いずに，起動しているプロセスに対して動的に DLL を注入する方法を採用した．

図 1 にそのフローを示す．原理は，現在，動いているすべてのプロセスを列挙し，その 1 つ 1 つに対して，各プロセスで動いているすべてのスレッドをサスペンドし，その間に，監視する API のエントリーを書き換えるためのモジュールをそのプロセス上でロードすることによって行われる．

具体的には，最初に指定したプロセス空間にメモリーを割り当てる VirtualAllocEx 関数を用いて，ロードする DLL の名前を記述する場所を作成する．その後，指定したプロセスに対して別スレッドを作成する CreateRemoteThread 関数を用いて，別スレッドで動かす関数として LoadLibrary 関数のアドレスを，そして，その LoadLibrary に渡す引数として，先に割り当てたメモリーのポインターを渡すことで実現している．図 2 にその方法を示す．

監視モジュールはロードされると，Binary Interception 技術を用いて,そのプロセス内の API を監視する.そして，すべてのスレッドをサスペンドする．それらをすべてのプロセスに対して行ったら終了である．監視対象にはプロセスを起動する API も含まれているため，プロセスが起動したら，自動的に監視モジュールを注入することができる．これにより，現在動いているプロセスおよび，これから起動されるすべてのプロセスに監視モジュールを注入することができる．

4. 実装と評価

Windows XP を対象に提案手法を実際に実装して評価を行った．まず，実際のユース・ケースとして，業務委託先に機密デジタル・データを提供し，その中身を編集後に返してもらうことを考える．デジタル・データをそのまま渡すことはできない．よって，発注者はデータを暗号化しておく必要がある．今回は，より強固な漏えい防止のため，特定の PC のみでしか閲覧あるいは編集できないようにした．

業務委託先の PC 上で動作するプログラムは，以下の事柄を行うものである．

1. 起動と同時に PC 上にセキュア空間を作成する．
2. セキュア空間にデジタル・データを復号する．
3. ユーザーは指定されたアプリケーションによってデータを編集する．
4. アプリケーションの終了を待ってセキュア空間を破棄する．このとき，データは再び暗号化され，それ以外はすべて消去される

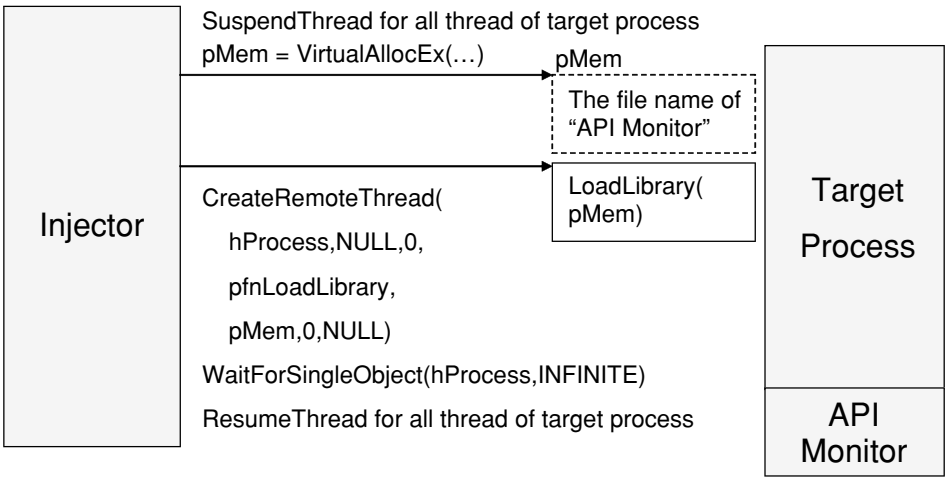

図 2. 動的監視モジュールの注入

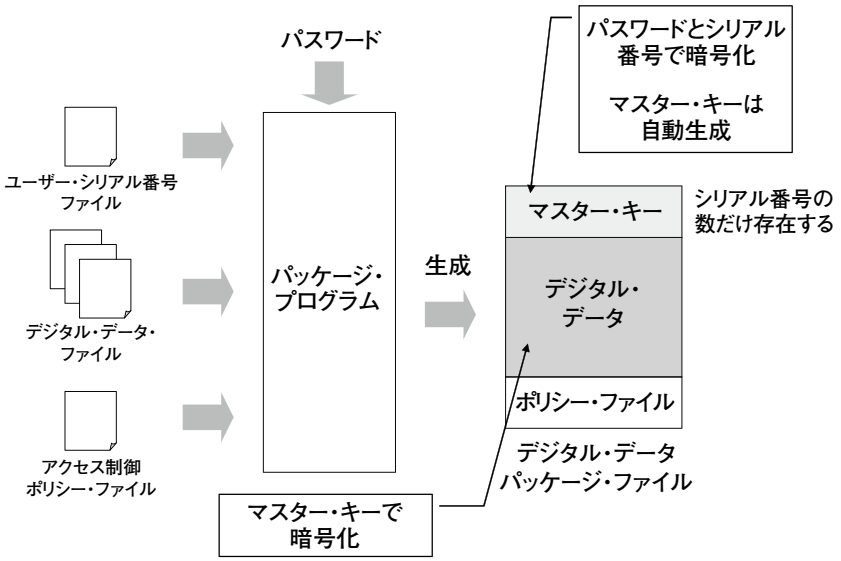

図 3. 暗号化パッケージ作成概念図

4.1 暗号化パッケージの作成

発注者側のパッケージ化プログラムの概念を図 3 に示す.

発注者は業務委託先の PC 固有情報, 例えば PC のシリアル番号を入手する. シリアル番号, デジタル・データ, アクセス制御ポリシーをパッケージ化プログラムに渡すことにより, デジタル・データはランダムに生成されたキーによって暗号化され, そのキーは与えられたパスワードとシリアル番号によって暗号化され, 1 つのファイルにパッケージ化される. 暗号化ロジックは実用性を考え, AES（Advanced Encryption Standard）[16], キー長は 128bit を採用した. パッケージ化されたファイルの中にはほかに有効期限などを入れることも可能であり, さらに, それらの改ざん防止のためにデータを暗号化したキーを用いて計算された HMAC（Keyed-Hashing for Message Authentication code）[17] もパッケージ内に同梱される.

4.2 セキュア空間の作成

図 4 にセキュア空間の概念図を示す. その作成手順を以下に示す.

① 実行プログラムを起動する.
② PC 固有情報, 例えば PC のシリアル番号を取得する. シリアルが一致しない場合, 終了する.
③ 暗号化されたデジタル・データを復号するためのパスワードを取得する. パスワードが一致しない場合は終了する.
④ API 監視モジュールを展開する.
⑤ API 監視モジュールをすべてのプロセスに対して注入を行い, 監視を始める. この時点で保護領域（セキュア空間）が作成される.
⑥ アクセス制御ポリシーとデジタル・データを復号して保護領域に展開する.
⑦ ポリシーを API 監視モジュールに適用する.
⑧ ポリシーに記述されたプロセスは保護領域のデータに対してアクセス可能になる.

セキュア空間作成フローを図 5 に示す. このように作成されたセキュア空間内でアクセス制御ポリシーによって指定されたプロセスのみが閲覧および修正が可能になる. 修正後は保護領域に保存が可能である.

また, 指定されたプロセスはセキュア空間以外からも読み込み可能である. ただし, セキュア空間が作られている間は, 書き込みは保護領域内に限られる. クリップ・ボードに関しても, 指定されたプロセス間に限り双方向が有効である. それ以外は, 指定されたプロセスからのデータのペーストは行えないが, その逆は可能であり, ユーザーの利便性を損なうことなく, 機密データを守ることが可能である.

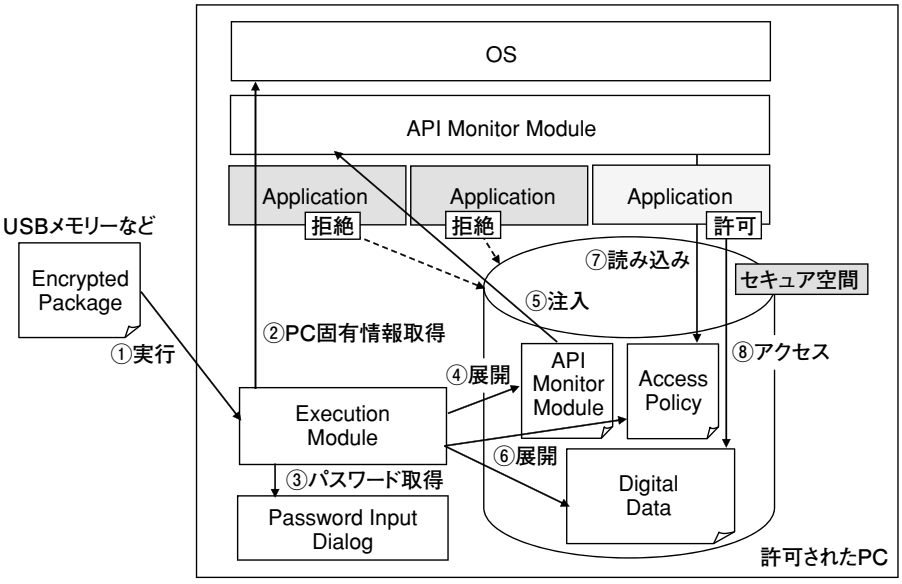

図 4. セキュア空間概念図

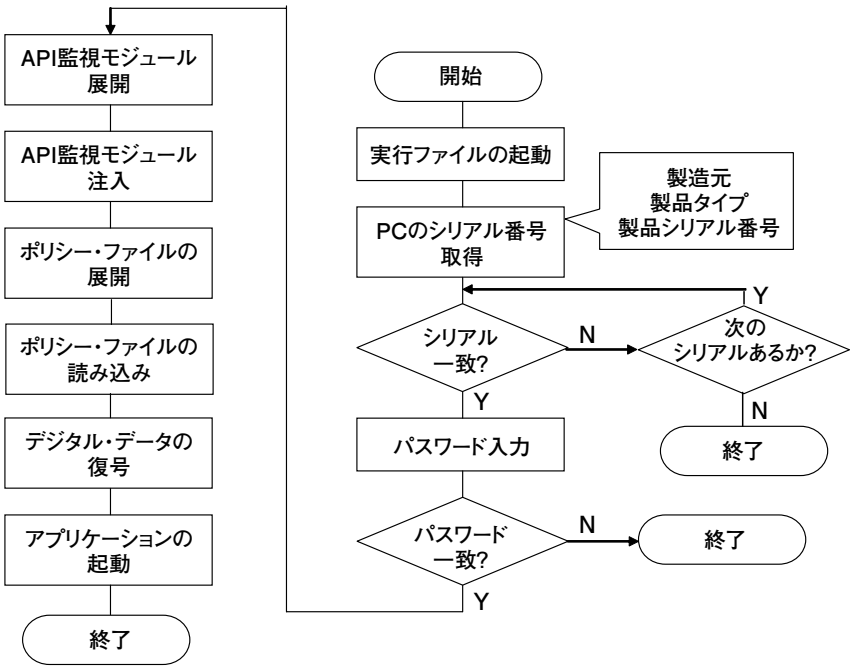

図 5. セキュア空間作成フロー

4.3 セキュア空間の破棄

セキュア空間の破棄は，実行プログラムが終了することによって行われる．ただし，指定されたプロセス，つまり保護領域にアクセス可能なプロセスがすべて終了するまではセキュア空間は保たれたままになる．

セキュア空間が破棄されるとき，その保護領域に展開されたデータは復号時と同じキーにより暗号化されて再パッケージされる．さらに，クリップ・ボードのデータも破棄され，すべてのファイル・データも消され，実行前の状態に戻る．再パッケージされたデータを，発注者に渡すことにより，編集後のデータを発注者は受け取ることができる．

4.4 性能評価テスト

セキュア空間を作っている間，すべてのプロセスに対して API 監視モジュールが注入され，API を監視していることから，Windows 全体のパフォーマンス低下が懸念される．そこで，ベンチマーク・テスト，BAPCO SYSmark 2004 SE［18］で性能評価を行った．オフィス業務やコンテンツ作成業務の生産性を評価するため，PC ベンダーやソフトウェア・ベンダーを中心にして作成された業界標準のベンチマークであり，複数のアプリケーションを同時に使用したときの個々のタスク応答時間を評価指標とする．従来のベンチマークが，CPU 演算，メモリー，ディスク I/O など特定のデバイスの局所的な性能評価を目的とするものに対し，アプリケーションの体感性能を総合的に評価することを目的としており，提案手法を評価するベンチマークとして適切であると考える．

表 1 にテスト環境の仕様を示す．**表 2** に提案した手法を実行中とそうでない場合のスコアおよび性能低下の割合を示す．すべてのプロセスを監視対象下においたとしても，平均で 2%，最大でも 4% の性能低下しか認められず，実用的には影響がないことを確認できた．

さらに，パッケージを作成する時間を比較した．ランダムに作成した 100MB のファイルのパッケージ化の測定結果を**表 3** に示す．

コピーはコマンド・プロンプト内のコピー・コマンドで単純にコピーしたときの測定値である．パッケージ化に要する時間は単なるファイル・コピーに比べ，3.6% の増加にとどまっており，パッケージ作成においても十分に実用的である．

表 1. テスト環境

OS	Microsoft Windows XP SP3
CPU	Intel® Core 2 Duo T5500 1.66GHz
RAM	1024MB
Video	Mobile Intel 945GM Express
Display	1400 × 1050 × 32bpp

表 2. BAPCO SYSmark 2004 SE のスコア

Test Category	非実行	実行	性能低下
Internet Content Creation			
3D Creation	237	235	1%
2D Creation	298	296	1%
Web Publication	224	219	2%
Overall	251	248	1%
Office Productivity			
Communication	93	89	4%
Document Creation	199	196	2%
Data Analysis	138	137	1%
Overall	137	134	2%
Total	**185**	**182**	**2%**

表 3. パッケージングの性能評価

	パッケージ	コピー	増加率
時間（sec）	15.96	15.40	3.6%

5. セキュア空間の安全性

提案手法によって作成されたセキュア空間にファイルを一時的に復号することから，幾つかのセキュリティー的な脅威が考えられる．以下にそれらを列挙するとともに，その対応策を記述する．

5.1 別ユーザーによるログイン

起動された監視モジュールは同一セッション内での監視のみが有効である．そのため，別ユーザーで同時にログインした場合，その別ユーザーのセッションでは監視が無効になる．その対応策として，Encrypting File System（EFS）［19］で暗号化したフォルダーに復号することで，別ユーザーからデータを保護することが考えられる．

5.2 強制終了

監視モジュールをタスク・マネージャーなどから強制的に終了することにより，保護領域のデータを残したままセキュア空間を破棄されることが考えられる．その対応策として，監視モジュールの終了を別モジュールで監視することにより，強制終了時にデータを削除させることが考えられる．あるいは，強制終了のAPIを監視することで，監視モジュールの強制終了自体を無効にさせることも技術的に可能である．

5.3 異常終了

電源オフなどの異常終了により，強制終了と同様に保護領域のデータを残したままセキュア空間を破棄されることが考えられる．これは，再起動時に保護領域に残っているデータを削除するためのプログラムを登録することで，対応することができると考えられる．さらに安全性を高めたい場合には，RAMディスクなどの揮発性メモリーに復号することが考えられる．

5.4 アクセス制御ポリシーの改ざん

セキュア空間の作成を可能にするためのポリシーはパッケージの中に存在する．そのポリシーを改ざんすることによって，保護領域からファイルの持ち出しを行うことが考えられる．しかし，パッケージにHMACによる署名をつけることにより，復号時にその署名を確認することができ，もし，パッケージが改ざんされていると判断された場合は，復号できないようにすることができる．

6. まとめ

本論文では，実行モジュールを実行した時点で，API監視モジュールを動的にすべてのプロセスに注入し，すべてのプロセスから読み込み禁止領域であるセキュア空間を作り出し，そこに暗号化されたデジタル・データを復号して展開し，ポリシーで指定されたプロセスに対してのみ読み込みや書き込みを許すことにより，データの漏えいを防ぐ手法を提案した．これは，既存のOSやアプリケーションを修正することなく，また，特定のアプリケーションをインストールすることなく，セキュア空間を作成することができるため，既存の業務形態を変えずに情報漏えいを防ぐことができる．

本手法の有効性を実証するために，実際にシステムを構築し，実際の業務環境に展開した．その結果，既存のWindows XP環境において，OSやアプリケーションの修正なしに，セキュア空間の構築に成功し，アクセス制御ポリシーで指定されたプロセスのみが復号されたデータにアクセスできることを確認し，ファイル操作やクリップ・ボード経由でのデータの複製ができないことを実証した．性能テストでは，業界標準のベンチマークを用いて，ユーザーの作業効率に影響を与えないことを示した．発注者によるパッケージ作成においても単なるファイル・コピーと比較し実用的であることを示した．さらに，セキュア空間に対して考えうるセキュリティー的脅威を列挙し，その対策を示した．

この手法はインストールが必要なく，また再起動も必要ないため，既存のソフトウェアやハードウェア資産を有効活用したまま容易に導入できると考えられる．よって，業務委託先の企業にデジタル・データを渡し修正依頼を行う場合，本手法で提示したように，デジタル・データを暗号化し，その実行モジュール（セキュア空間を構築し，デジタル・データを復号するモジュール）と一緒に渡すことにより，データ流出を防ぎながら作業の委託を行うことが可能になる．

今後の展望として，Windows XP以外のプラットフォームへの適用の検討を進める．

参考文献

[1] 新井利明,溝口幸信:“モバイルセキュリティを強化したシンクライアントソリューション,” 情報処理,Vol.47, No.10, pp.1127-1136 (2000).

[2] Jason Nieh, S.Jae Yang, and Naomi Novik : “A comparison of thin-client computing architectures,” Technical Report CUCS-022-00, Columbia University (2000.11).

[3] 古市実裕, 三品拓也,大谷佑:“PCにおけるプログラムのリアルタイム監視・制御技術の開発と情報セキュリティシステムへの応用,” SCIS2006 (2006).

[4] 古市実裕,多田政美,池部敦巳:“PCセキュリティ強化のためのCOMの透過的ポリシー強制手法,” SCIS2007 (2007).

[5] Adobe. “Adobe LiveCycle Enterprise Suite2 (ES2),” http://www.adobe.com/products/livecycle/

[6] Microsoft. “Techical overview of Windows Rights Management Service for Windows Server 2003,” (2003.11).

[7] Peter Loscocco and Stphen Smallley : “Integrating flexible support for security policies into the Linux operationg system,” In Proceeding of the FREENIX Track: 2001 USENIX Annual Technical Conference (2001).

[8] Mick Bauer : “Paranoid penguin: an introduction to Novell AppArmor,” Linux Journal, Vol.2006, No.148, p.13 (2006.08).

[9] 中村雄一,鮫島吉喜 : “Security-Enhanced Linuxのアクセス制御ポリシ設定の簡略化,” SCIS2003, pp.831-836 (2003).

[10] 田端利宏,箱守聡,大橋慶,植村晋一郎,横山和俊,谷口秀夫:“機密情報の拡散追跡機能による情報漏えいの防止機構,” 情報処理学会論文誌, Vol.50, No.9, pp.2088-2102 (2009).

[11] 日立ソフト:秘文, http://hitachisoft.jp/products/hibun/

[12] VERDASYS: Digital Guaridan, http://www.verdasys.com/data_loss_prevention.php

[13] Galen Hunt and Doug Brubacher: “Detours: Binary Interception of Win32 Functions,” Proceedings of the 3rd USENIX Windows NT Symposium, pp.135-144 (1999).

[14] Galen C. Hunt and Michael L. Scott: “Intercepting and Instrumenting COM Applications,” Proceedings of the 5th Conference on Object-Oriented Technologies and Systems (COOTS '99), pp.45-56 (1999).

[15] Microsoft. “Working with the AppInit_DLLs registry value,” http://support.microsoft.com/kb/197571/en-us

[16] Joan Daemen andVincent Rijmen: “AES Proposal: Rijndae,” AES Algorithm Submission (1999.09.03).

[17] Hugo Krawczyk, Mihir Bellare, and Ran Canetti: “HMAC: Keyed-Hashing for Message Authentication,” Network Working Group, RFC2104 (1997).

[18] BAPCO. “SYSmark 2004 SE : Product Information,” http://www.bapco.com/products/sysmark2004se/

[19] Microsoft. “Encrypting File System,” http://technet.microsoft.com/en-us/library/cc782901(WS.10).aspx

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Tivoli 開発
マネージング・コンサルタント

多田 政美 Masami Tada

[プロフィール]
1989 年, 日本 IBM 入社. インプット・メソッドやオペレーティング・システムの開発に従事する. 現在, ソフトウェア開発研究所にて, クライアント・セキュリティーに関するサービスや開発に従事. 情報処理学会会員 .

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Tivoli 開発
アドバイザリー・ソフトウェア・エンジニア

古市 実裕 Sanehiro Furuichi

[プロフィール]
1996 年, 日本 IBM 入社. 東京基礎研究所にて, モバイル・コンピューティング, システム・ソフトウェア, 情報セキュリティーなどの研究に従事. 2007 年 12 月よりソフトウェア開発研究所にて, セキュリティーや Smarter Planet 関連の製品開発に従事. 情報処理学会会員.

日本アイ・ビー・エム株式会社
ソフトウェア開発研究所
Lotus ソフトウェアサービス
アドバイザリー・プロジェクトマネージャー

新村 泰英 Yasuhide Niimura

[プロフィール]
1987 年, 日本 IBM 入社. 現在, ソフトウェア開発研究所にて, クライアント・セキュリティーに関するソリューション開発および提供のプロジェクト・マネジメントに従事. プロジェクトマネジメント学会会員 .