仮想ネットワークの観測技術の研究開発

Towards Traceable Overlay Network over Virtualized Systems

安藤類央
ANDO Ruo

要旨

トレーサブルネットワーク技術の研究開発として、近年急激に普及が進んでいる仮想ネットワークの観測技術の研究開発を行った。具体的には、仮想ネットワークを構成する仮想マシンの観測、仮想ネットワークであるP2Pなどのオーバーレイネットワークの観測技術の研究開発を行った。観測技術のセキュリティ事案への適用は主として高粒度な侵入検知と広域な情報漏洩の追跡であり、仮想ネットワーク上で発生するセキュリティインシデントのうち、情報漏洩やマルウェア感染の事象の可観測化と可視化のためのシステムを産学官連携の上研究開発を行い、NICTのテストベッド上での実証実験を行った。

We have been researched the traceability of overlay network including P2P network and virtualized systems. For this purpose, we have developed monitoring system for P2P network and virtual machines running on hypervisor. As one of the design goals, we have aimed to enhance the system about fine-grained and large-scale monitoring for tracing information leaks and malware outbreaks. In this research, we have cooperated with software house and SIer concerning information security and evaluated our systems on testbed of NICT.

[キーワード]
仮想ネットワーク観測，仮想化技術，オーバーレイネットワーク，情報漏洩，テストベッド
Virtual network monitoring, Virtualization technologies, Overlay network, Information leaks, Testbed

1　まえがき

ネットワークの広帯域化とクラウドコンピューティング、P2Pネットワーク技術の普及に伴い、仮想マシンによって構成される仮想ネットワークを利用する機会が多くなっている。第2期中期計画では、トレーサブルネットワーク技術の研究開発として、侵入検知の精度と観測範囲の拡充を行った。また、そのため、トレーサブルネットワーク観測技術の精緻化、観測レンジの拡充として、仮想マシンの観測、P2Pネットワークの観測の2つを対象範囲として研究開発を行った。

現在急激に進んでいるクラウドコンピューティングの普及の原因の1つに、仮想化技術の発展がある。SaaS、PaaS、IaaSのサービス種別に関わらず、クラウドコンピューティングの定義の1つとして、仮想マシンの利用を提供するサービスとすることができる。そこから、クラウドコンピューティングなどの新形態のシステムのセキュア化には仮想マシンの観測が重要であるとの認識のもと、probing技術の研究開発を行った。

また、前期中期計画期の2006年前後から、P2Pネットワークなどのオーバーレイネットワーク上での情報漏洩が、もっとも重要なセキュリティインシデントの1つとして社会問題化するようになり、トレーサブルネットワークの研究の1つとして、同事象のトレース可能化技術の開発を行った。また、P2Pネットワークから得られる追跡のためのデータは大規模になるため、解析処理機能の強化を行った。

仮想化技術により構成されるシステム上の観測、侵入検知技術は一般に、virtual machine introspectionと呼ばれる。仮想マシン上で実装したIDS（Intrusion Detection System）は、監視対象とプローブが仮想化技術によって隔離されているという性質から、従来のIDSにはない機能用件を満たすことができる。従来のNIDSは攻撃者から見えにくいがデータの精度が落ちる一方で、

HIDSはデータの精度が上がるが、visibilityが増すということが指摘されており、VMMの特性を活かして、この相反する機能用件を止揚する設計が提案されている。IDSを構築する際の仮想マシンモニタの利点はIsolation（プローブがゲストOSから隔離されて配置される）、Inspection（カーネル空間でのイベントを含む詳細なデータを得ることができる）、Interposition（システムコールや割り込み、VMMへのIO要求などをフックして、その際に中間動作を入れることができる）の3点にあると想定される。そのため、この3点が侵入検知に特徴を発揮できるように研究開発を行った。

1台のコンピュータがサーバとクライアントを兼ねるP2Pでは、ユーザ同士の情報交換が簡単に行え、ネットワークトラフィックの急増などの変化にも耐性があるため、急速に普及が進んでいる。その反面、P2Pネットワークを介した著作権侵害や、P2P経由でのウィルス感染による情報漏洩が問題になっている。P2Pソフトウェアはフリーで公開される場合が多い反面、商用ソフトに比べ開発コミュニティの発足と発展の段階でバージョンアップや修正が行われなくなることが多い。また、諸般の事情で発展が止まる場合もある。そのため、新たな脆弱性やP2Pアプリケーションによって形成されるオーバーレイネットワークへの攻撃が判明しても、対応する修正やパッチなどが用意されないことがある。そのため、P2Pネットワークの観測は仮想ネットワークを構成するソフトウェアの解析を行い、プロトコルやデータ構造を抽出し、観測を行う必要がある。前期中期計画では、P2Pソフトウェアの解析により、広域仮想ネットワークを観測するシステムの研究開発を行った。

2　仮想化システムの観測技術の開発

近年のプロセッサ性能の急速な向上は、複数のOSを同時に稼動させる仮想化技術の実用性を高めた。仮想マシンモニタなどのシステムは、従来のデバッガやダンプツールと比較して、OSやプロセスの外部観測を容易にした。また、完全仮想化モードは、プロテクトモード以来のプロセッサ構造の革新と指摘されており、この仮想化技術が提供する外部観測性は、デバッグやソフトウェアのチェック、特にVMM（仮想マシンモニタ）に関してはセキュリティ機能の実装の観点から、注目されている。特にセキュリティインデントは発生するタイミングが予測不可能であり処理時間が見積もれないことから、観測防御対象から防御システムへ非同期に処理を依頼することが重要であり、仮想化技術を用いることでこのような処理が可能になる[1]。また、観測防御対象のOSが複数になる場合、これらを仮想化し1つの物理マシンに集約することで一元的にセキュリティポリシーの設定やアクセス制御を行うことが可能になる[2]。

加えて、ここ数年のデバッグ技術の発達は目覚しいものがあり、仮想化技術の発展とあわせて、より詳細な観測と、高粒度なロギングが可能になってきている。特に、Microsoft Windowsが提供するデバッギングAPI、ネットワークアプリケーション、そしてカーネルモジュールの開発のための環境も急速に整備されつつある。また、VMWareなどが提供しているAPIや、オープンソースの仮想マシンモニタの普及により、ホストOSのドライバと仮想マシンモニタ側のメモリ管理ユニットや割り込みハンドラが連携し、ゲストOSの情報をホストOS側に通達できることになった。これに伴い、仮想化技術を用いたデバッグやマルウェア解析機能の強化が研究されている[3]。

近年、ネットワークに接続されるWindows OSのアプリケーションは質、量ともに急激な増加傾向にある。その結果、仮想化技術の普及とも相俟って、ネットワークアプリケーション間の依存関係は複雑化しており、定性的なログ解析によるネットワーク上のクライアントの稼動状況の異常を把握する際の情報処理負荷が増加している。本論文では、仮想化技術を用いた仮想ネットワークアプリケーションの観測手法を提案した。

2.1　仮想マシンモニタの種類と修正方法

仮想マシンモニタは2つの種類に分類される。XEN[4]のようにハイパーバイザーを自前で構築するもの、KVM（kernel virtual machine）のようにホストOS（Linux）内部に構築する2つである。

仮想マシンモニタタイプ1は、ホストOSをハイパーバイザーとして用いるもので、このタイプにはKVMがある。この方式では、ホストOSのカーネル空間内にあるゲストOSの仮想メモリを、ユーザ空間あるいはカーネル空間に移動する。こ

の方式では、QEMUのインターフェイスを使う。KVMはLinuxカーネルに取り込まれているので、管理OSで最新機能が使えるが、専用のAPIが用意されておらず、まだ安定していない。そのため、提案システムの実装にはデバッグハンドラを用いた値の転送や、共有メモリを用いた文字列の転送などに適用した。

仮想マシンモニタタイプ2は、独自にブートローダとハイパーバイザーを用意するもので、このタイプには、XENがある。この場合、準仮想化と完全仮想化に分かれるが、本論文では、Windowsの完全仮想化を扱っているため、スナップショットの取得には、XENのインターフェイスではなく、QEMUのインターフェイスを用いることになる。KVMと同様に、VCPUのレジスタ経由でAPIの引数やASCIIコードなどを受け渡す方法がある。XENの特徴として、APIが整備されていることがある。そのため、APIを駆使することでメモリのスナップショットからWindowsの状態の解析を行うことやログ情報を転送することができる。また、ハードウェア割り込みをWindows OSに先んじて捕捉することが可能である。

仮想マシンタイプ1、2の双方の場合ともハードウェア、あるいは仮想化されたハードウェアへのアクセスや状態遷移を捕捉し、仮想化マシンの挙動を観測する手法が適用可能である。この手法はVirtual Machine Introspectionと呼ばれ、詳細は文献[5]で提案されている。

2.2 仮想Windows OSの修正方法

本論文での提案システムである統合システムモニタは、DLL Injectionやフィルタドライバ、システムコールテーブル書き換えなどによるAPIフックにより、Microsoftから提供されているモニタツールよりも高粒度なモニタを行うことを可能にするものである。図1は、統合システムモニタの概略を示したものであり、Windows OSで提供されているデバッグとフィルタリングのための機構を利用して、各種リソース（メモリ、ソケット、ファイル、レジストリ）のアクセスをインターセプトし、ログに記載する。また、分岐命令の実行やハードウェア割り込みなどのフックは、仮想マシン側で修正を行うことで対応する。

2.2.1 フィルタドライバによるカーネルAPIフック

Microsoft Windowsのフィルタドライバは、Windows XPから積極的に導入、活用が始まったソフトウェアモジュールである。フィルタドライバは、IOマネージャとカーネルドライバの間に位置して、ファンクションドライバ（既存のデバイスドライバ）の前後に既存のWindowsが提供する機能を利用して呼び出され、新しい機能の追加、修正、デバッグなどを行うソフトウェアである。フィルタドライバを用いることで、Native APIフックを行うことができる。Native API Hookは、カーネルモードでのイベント検出に用いられる。ユーザモードでのAPI発行は、Ntdll.DLL経由でカーネル空間に伝達される。Windowsでは、カーネルモードで動作するAPIを、Native APIといい、これらは、SystemServiceDescriptorTableという構造体によって、制御されている。そのため、Native API Hookを行うためには、SystemServiceDescriptorに修正を加える必要がある。

Modification 1は、InterlockExchangeを用いて、システムコールテーブルを書き換える方法である。Modification 2は、ドライバのロード時にシステムコールテーブルを書き換え、フック関数を通過する方法である。このシステムコールテーブル修正により、Windows OSのメモリアクセスを捕捉し、メモリ挙動の可視化などを行うことが可能になる[6]。

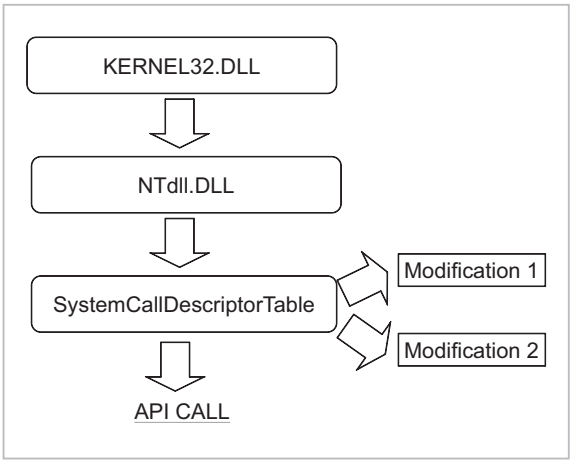


図1 カーネル空間でのデータ構造テーブルの変更によるAPI挙動の観測

2.2.2 フィルタマネージャによるAPIフック

開発環境や稼動する状況にも依存するが、通常のフィルタドライバを用いたファイルIOのフックは、安定動作しない場合がある。フィルタドライバは、Microsoftが新たに提供をはじめたファイルシステムフィルタドライバである。フィルタドライバは、大量に呼び出される割り込みやAPIに関して、順序などを適切に制御し、サードパーティの開発を簡素化、支援することを主眼に設計されている。図2は、フィルタマネージャの機能概要を示したものである。フィルタマネージャは、カーネルドライバ(ファンクションドライバ)とフィルタドライバの間を仲介し、フィルタドライバの実装を簡素化し、機能を安定にする。フィルタドライバは、Microsoftが提供するフィルタマネージャを介して実装され、軽量な中間処理を可能にする[7]。

2.2.3 DLL InjecionによるAPIフック

DLL Injectionとは、任意の関数をライブラリ化して特定のAPIが発行されたときに、実行させるもので、既存のアプリケーションに新しい機能を加えたいとき、ソフトウェアのデバッグの際に用いられる。開発の要請から、DLL(独自API)を他のプロセスに任意のタイミングで実行させるDLL Injectionという技術が用いられることがある。DLL Injectionの方法には、Microsoft Windowsが提供している機能を使うもの、デバッグ機構の機能を使うもの、リモートスレッドを使うもの、そしてモジュールのインポートセクションの変更によるものなどがある。本論文では、モジュールのインポートセクションの変更による方法により、対象ソフトウェアのデータ送受信関数をフックし、ここに適切な操作を入れることにより、ソフトウェアの挙動を追跡し、問題となるトラフィックが発生または到着する前に防止を行うことを可能にした。インポートテーブルの変更によるDLL注入を行った(図1)。インポートテーブル(インポートセクション)とは、セクションテーブルの中にあるデータ構造体(ヘッダ)で、プログラムが実行される前に必要なDLLのアドレスと、利用するDLL内のシンボルのアドレスのリストが格納されている。ここで、フックしたいDLLのアドレスを、任意のDLLへのアドレスに書き換えることで、ソフトウェアの動作の修正を行うことができる。この方法は、特定のCPUの仕様に依存しなく、またスレッドの同期の問題もないため、非常に柔軟な方法として用いられることが多い。処理としては、1) 注入したいDLLを用意する。2) 対象となるソフトウェアのインポートテーブルアドレスを取得する。3) フックしたい関数のアドレスを探す。4) 発見したアドレスを、注入したいDLL(関数)へのアドレスに書き換える。関数形は

```
void ReplaceIATTableInP2Psoftware
("kernel32.dll",
funcORG,
```

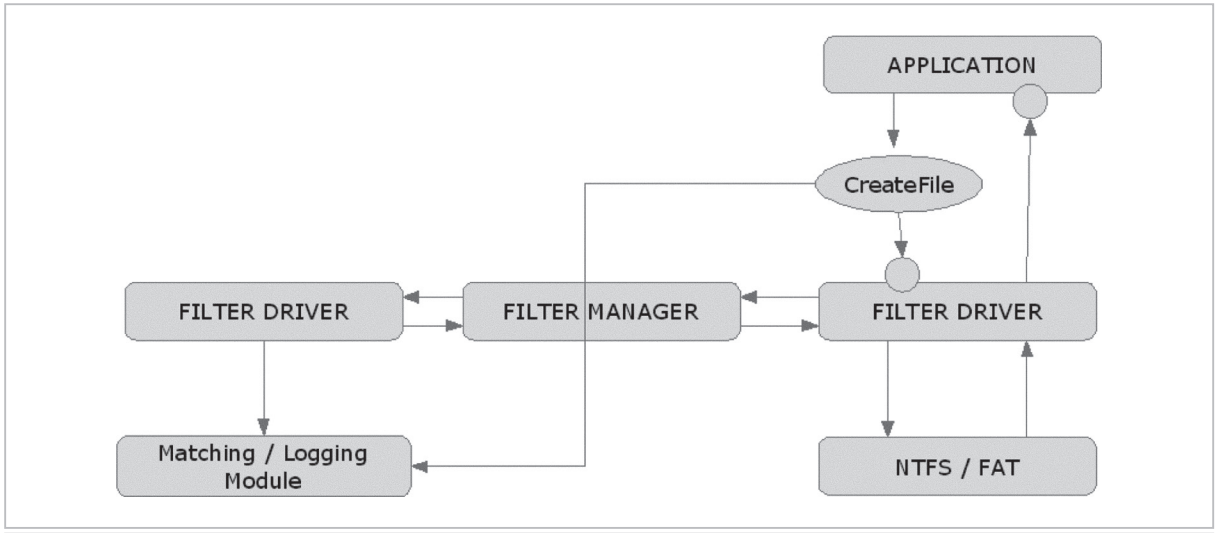


図2 フィルタドライバによるWindows OSファイルアクセスのロギング

```
funcINSERT",
moduleHandler);
```

対象とするプロセスの構造などにより、実装方法はいくつか存在するが、大枠は以上で示した通りである。

2.3 提案システムの出力

ここでは、上記で述べた仮想マシン観測システムの出力ログを自己組織化マップによるクラスタリングを行った結果の可視化について述べる。可視化処理の前の Windows OS のレジストリアクセスの捕捉や集計処理の詳細は文献[8]で提案されている。

2.3.1 取得データとアルゴリズム

ここでは、観測した仮想マシンの挙動データを可視化した結果について示す。仮想マシンの可視化に可観測化したレジストリアクセスログを用いた。下記にログの出力例を示す。

```
128799977931406250 ctfmon.exe(324) EnumKey 0x00000000
HKEY_LOCAL_MACHINE¥SOFTWARE¥Microsoft¥CTF¥TIP
{78CB5B0E-26ED-4FCC-854C-77E8F3D1AA80}
```

ここで、上列は８つの値で構成される。

1 システム時間
2 呼び出し元のプロセス情報
3 呼び出されたレジストリ関数名
4 関数の返り値
5 レジストリキー
6 レジストリバリュー
7 設定値
8 取得値

レジストリアクセスログを定量化する際には、レジストリキー、レジストリの関数名での出現単語の頻度をカウントし、行列を生成した。

さらに、本論文ではレジストリアクセスデータの可視化に自己組織化マップを用いた。自己組織化マップは、教師なし学習アルゴリズムの１つであり、K 平均法等と異なりクラスタ数を所与としないところに特徴がある。また、自己組織化マップの特長に、データ間のトポロジーを変えずに学習を行うことが可能という点がある。競争層を２次元に設置すると、多次元の入力データ間の位相関係を保持しつつ、可視化を行うことが可能である。

動作式は下記になり、その他のニューラルネットワークのアルゴリズムと同じくニューロン間の重みを、入力ベクトルと加重ベクトルの差分により修正する。

$$Wv(t+1)=Wv(t)+F(t)a(t)(D(t)-Wv(t))$$

ここで W はノード間の加重係数行列、D は入力ベクトル、a(t) は学習係数である。自己組織化マップでは、各ノードに対して BMU (best matching unit) を決定する。上式の F(t) は近傍半径であり、BMU からの距離によって変化する。学習過程を終了させる敷居値は、学習回数を用いる。自己組織化マップの可視化の詳細については以下で述べる。

2.3.2 仮想マシン挙動の可視化

本論文では、提案手法の評価実験として、セキュリティインシデントの発生した仮想マシンと状態の類似した仮想マシンを自己組織化マップでの可視化を行った。

評価実験では、ウィルスの感染動作をインストール、通信（パケット送信）に分類し、それぞれ動作が類似するアプリケーションとデバイスドライバのインストール、P2P アプリケーションと WEB ブラウザの実行時のデータを用意し、分類識別を行った。処理を行った動作パターンは下記の通りである。

```
<Pattern Ⅰ>
input 1: installation of text editor
input 2: installation of device driver
input 3: installation and
execution of malware
<Pattern Ⅱ>
input 1: running P2P application
input 2: running Internet Explorer
input 3: installation and
execution of malware
<Pattern Ⅲ>
input 1: running P2P application
```

input 2: installation of device driver
input 3: installation and
execution of malware

それぞれの可視化結果を図3～5に示す。図3は、Pattern I、マルウェアによる感染と悪意のないソフトウェアのインストールを可視化識別したものである。マルウェアのインストールは図3の中央側、ソフトウェアのインストールはアプリケーションとデバイスドライバの2つであるが、これは図の右側に配置されている。図4は、Pattern II、マルウェアの感染実行と、P2PとWEBブラウザという2つのネットワークアプリケーションの実行を可視化識別したものである。

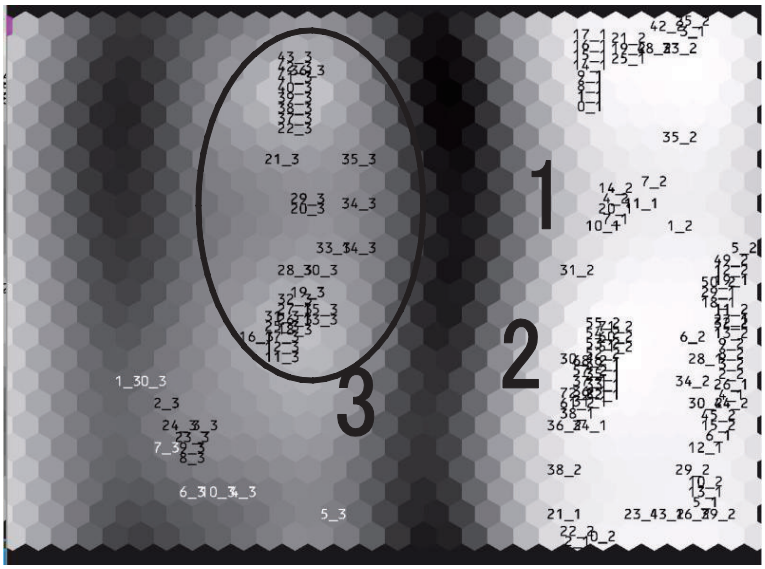


図3 Windows OS上の3状態（アプリケーションのインストール、デバイスドライバのインストール、マルウェアの感染実行）のクラスタリングと可視化の結果表示。マルウェアの感染実行は、図の中央左側（**3**）に分布している。

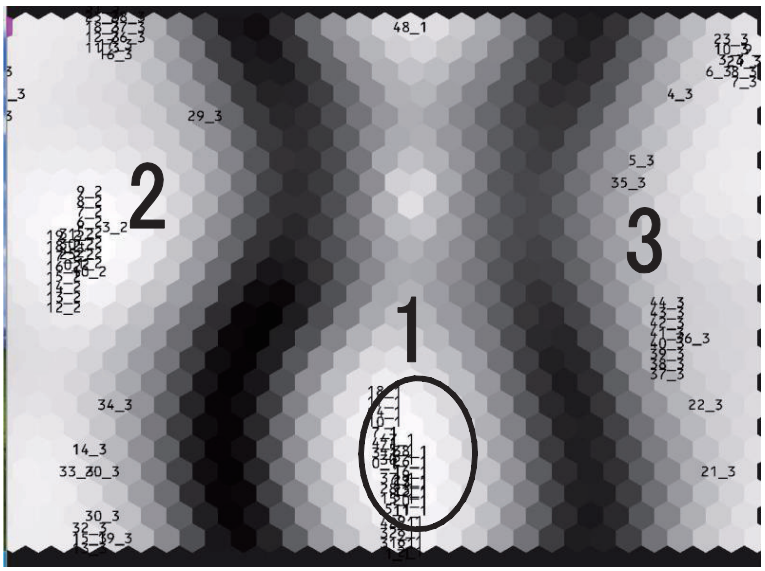


図4 Windows OS上の3状態（P2Pアプリケーションの実行、WEBブラウザ（IE）の実行、マルウェアの感染実行）のクラスタリングと可視化の結果表示。マルウェアの感染実行は図の右側（**3**）に分布し、他の2つの通信から識別されている。

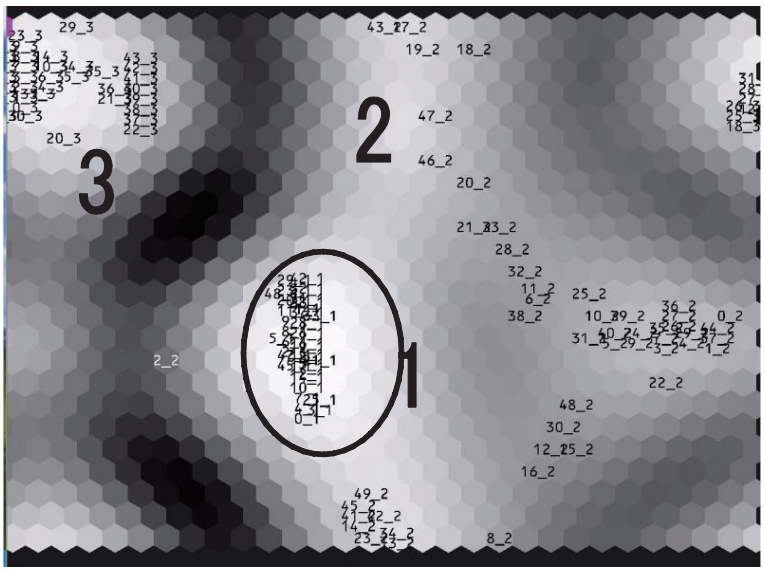


図５ Windows OS 上の３状態（P2P アプリケーションの実行、デバイスドライバのインストール、マルウェアの感染実行）のクラスタリングと可視化の結果表示。マルウェアの感染実行は図の左側（**3**）に分布している。

これらの３ケースはそれぞれ分離され、図４の３箇所に配置される結果になった。マルウェアの感染実行と通信系アプリケーションの実行を可視化識別したものである。通信系アプリケーションは P2P と WEB ブラウザであり、同２状態は図の中央から左側に、マルウェアの感染状態は右側に配置されている。図５は、Pattern Ⅲ、P2P アプリケーションの実行（**1**）、デバイスドライバソフトウェアのインストール（**2**）、マルウェアの感染実行（**3**）の３パターンを分類可視化したものである。**1** と **2** は図の中央右側に、マルウェアの感染実行は図の左側に分布する結果となっている。

３　オーバーレイネットワーク観測技術の開発

１台のマシンがサーバとクライアントを兼ねる P2P ネットワークによる情報漏洩が社会問題になっている。P2P ネットワークによるファイル流通は、ここ数年関連調査が行われるようになったが、海外も含めた広域なファイル伝播については詳細な調査や公開された情報が少ないのが現状である。P2P ソフトウェアの普及による社会的影響が大きくなっている現況では、ネットワークの状況の可視化（見える化）などによる影響の把握が重要である。P2P ソフトウェアは、フリーで公開／利用されるケースが多い反面、情報が公開させられないか、または入手が困難であるため、ネットワーク全体でどのようなファイルの流通が行われているか把握されない場合が多く、セキュリティインシデントレスポンスを難しくしている。広域に渡る P2P ネットワークは、観測データ量が膨大になるため、観測ノードを仮想化し集約度を上げ、かつ解析系のリソース消費が観測系の実行運用を妨害・錯乱しないように設計することが有効である。本論文では仮想化技術を用いた広域 P2P ネットワーク観測システムの大規模データ処理性能強化について述べる。仮想マシンモニタとホスト OS 側へ観測データを転送（VM introspection）する際に、仮想レジスタを用いたプロトコルを実装することで、仮想観測系でのファイル IO やメモリアクセスの発生頻度を削減することができる [1]。これにより、仮想観測系側では観測問題制約を充足し、リソースをネットワークモニタのために最大限に利用することが可能になる。

3.1　P2P ソフトウェアの構造と特徴

１台のコンピュータがサーバとクライアントを兼ねる P2P では、ユーザ同士の情報交換が簡単に行え、ネットワークトラフィックの急増などの変化に

も耐性があるため、急速に普及が進んでいる。その反面、P2P ネットワークを介した著作権侵害や、P2P 経由でのウィルス感染による情報漏洩が問題になっている。P2P ソフトウェアはフリーで公開される場合が多い反面、商用ソフトに比べ開発コミュニティの発足と発展の段階でバージョンアップや修正が行われなくなることが多い。また、諸般の事情で発展が止まる場合もある。そのため、新たな脆弱性や P2P アプリケーションによって形成されるオーバーレイネットワークへの攻撃が判明しても、対応する修正やパッチなどが用意されないことがある。

P2P ソフトウェアは、一般に 3 つの形態に分けられる。ノード同士が直接データやファイルを交換するのが P2P の特徴であるが、ネットワーク上のどのノードがどういうファイルを持っているかはサーバが管理するものを第 1 世代、純粋 P2P といわれているのが、サーバを必要とせず、完全にノード同士がデータ交換をするのが第 2 世代である。さらに、匿名性を増すためにキャッシュ機能を備えたものは第 3 世代の P2P ソフトウェアと言われる。

P2P ソフトウェアは、ノード管理、クエリ管理、キー管理、タスク管理の 4 つのモジュールで構成されることが代表的である。このうち P2P ソフトウェアの挙動を追跡するのに重要な情報はキーテーブル、ノードテーブル、送受信ファイルテーブルであるが、本論文では、仮想メモリ上に展開されているこれらのデータを用いて、パケットを送信時の API 呼び出しをフックし、キャプチャし、解析、復号とフィルタを行うシステムを構築した。フックを行う方法については **2** で詳解した。

P2P アプリケーションは、複数のユーザの利用を前提としている以上、プログラムを気軽に作りかえることはできない。また、バージョンアップや修正が保障されているわけではなく、諸般の事情で開発が止まってしまう事がある。このような特徴を持つソフトウェアを安全に使うためには、本論文で示すデバッグの技術を用いた修正を施すことが有効である。また、第 2 の重要な特徴に、P2P アプリケーションでは、構成されるネットワークのトポロジーを把握できないことが上げられる。これは、特定のプローブシステムを P2P ネットワーク上に設置すると、それが攻撃の対象になる可能性があり、ブロードキャスティングも同様の理由で行われないことが多いためである。そのため、サーバベースでのモニタを行うことは難しく、トポロジーやトラフィックフローを把握するためには各プローブでのプロセスメモリの情報を得る必要がある。このため、**2** で前述した DLL Injection などの技術を用いて P2P アプリケーションの修正を行い、トラフィックの解析を行う手法が提案されている [9]。

3.2 P2P ネットワークトラフィックの観測と処理

P2P ネットワークのトラフィックは、提携業務や WEB と比較しても変動が激しく、予測が困難である。また、短期間でトラフィックの急激な増加に対応し、ストレージやプロセッサの追加を行う必要がある。そのため、解析処理部は、仮想ノード外部のホスト OS 側に配置することが解決策の 1 つとしてあげられる。解析初期部をノード内部に配置する場合、ストレージやプロセッサを動的に追加するための機能については現在のところ拡充されていない。本論文では、仮想レジスタを用いて仮想 PROBE からホスト OS へログ情報の送信を行い、解析処理の拡充は物理マシンを直接制御するホスト OS 側で行う手法を提案する。

広域に渡る P2P ネットワークは、観測データ量が膨大になるため、観測ノードを仮想化し集約度を上げることが有効であるが、集約化により各 PROBE のトラフィックを削減する事は意図されておらず、また観測量の増大につれ解析処理にかかる負荷も増加するため、同一 PROBE に観測と解析の機能を配置すると、解析処理負荷が観測性能に影響を及ぼす可能性がある。

3.1 で述べた P2P トラフィック観測の条件から、本論文では観測系と解析系を仮想化技術によって分離する方法を提案する。これにより、観測系からの出力ログの増大や変動にあわせて、ホスト OS 側のストレージやプロセッサ性能を柔軟に変更することが可能になる。P2P ネットワークのような広域、大規模トラフィックデータ処理のために VM Introspection（VM 観測技術）を併用する手法の詳細は文献 [10] で提案されている。

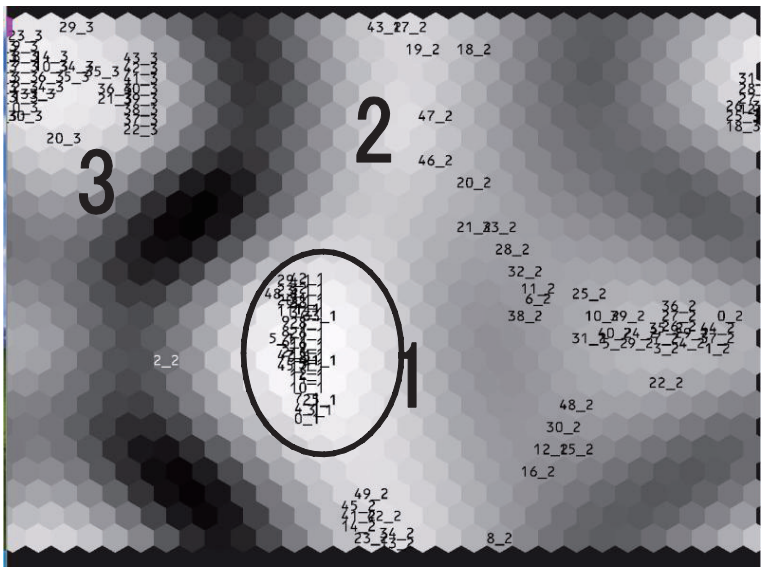


図５ Windows OS 上の３状態（P2P アプリケーションの実行、デバイスドライバのインストール、マルウェアの感染実行）のクラスタリングと可視化の結果表示。マルウェアの感染実行は図の左側（**3**）に分布している。

これらの３ケースはそれぞれ分離され、図４の３箇所に配置される結果になった。マルウェアの感染実行と通信系アプリケーションの実行を可視化識別したものである。通信系アプリケーションは P2P と WEB ブラウザであり、同２状態は図の中央から左側に、マルウェアの感染状態は右側に配置されている。図５は、Pattern Ⅲ、P2P アプリケーションの実行（**1**）、デバイスドライバソフトウェアのインストール（**2**）、マルウェアの感染実行（**3**）の３パターンを分類可視化したものである。**1** と **2** は図の中央右側に、マルウェアの感染実行は図の左側に分布する結果となっている。

３　オーバーレイネットワーク観測技術の開発

１台のマシンがサーバとクライアントを兼ねる P2P ネットワークによる情報漏洩が社会問題になっている。P2P ネットワークによるファイル流通は、ここ数年関連調査が行われるようになったが、海外も含めた広域なファイル伝播については詳細な調査や公開された情報が少ないのが現状である。P2P ソフトウェアの普及による社会的影響が大きくなっている現況では、ネットワークの状況の可視化（見える化）などによる影響の把握が重要である。P2P ソフトウェアは、フリーで公開／利用されるケースが多い反面、情報が公開させられないか、または入手が困難であるため、ネットワーク全体でどのようなファイルの流通が行われているか把握されない場合が多く、セキュリティインシデントレスポンスを難しくしている。広域に渡る P2P ネットワークは、観測データ量が膨大になるため、観測ノードを仮想化し集約度を上げ、かつ解析系のリソース消費が観測系の実行運用を妨害・錯乱しないように設計することが有効である。本論文では仮想化技術を用いた広域 P2P ネットワーク観測システムの大規模データ処理性能強化について述べる。仮想マシンモニタとホスト OS 側へ観測データを転送（VM introspection）する際に、仮想レジスタを用いたプロトコルを実装することで、仮想観測系でのファイル IO やメモリアクセスの発生頻度を削減することができる[1]。これにより、仮想観測系側では観測問題制約を充足し、リソースをネットワークモニタのために最大限に利用することが可能になる。

3.1　P2P ソフトウェアの構造と特徴

１台のコンピュータがサーバとクライアントを兼ねる P2P では、ユーザ同士の情報交換が簡単に行え、ネットワークトラフィックの急増などの変化に

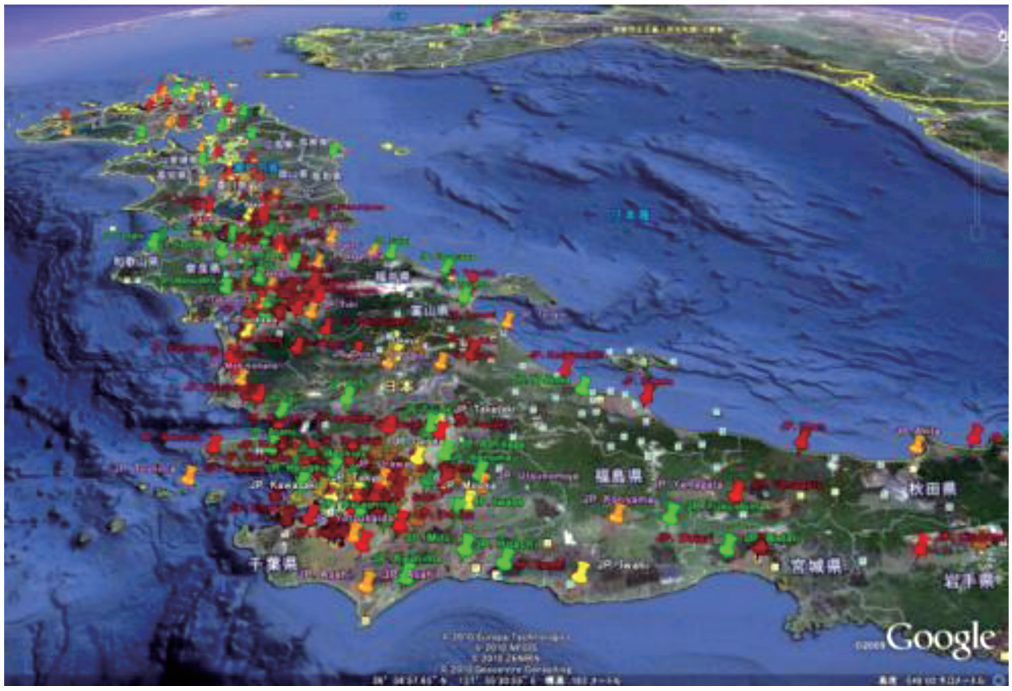


図7 日本のP2Pネットワークの可視化例

ノード規模（スーパーノード、中規模、小規模）によって色分けして分類プロットしている。

図8 小金井市周辺のP2Pネットワークノードの可視化例

分布状況などを目視することができないため、可視化システムでは拡大縮小により個別地域のノード分布を調べることができるようになっている。

3.4.2　スーパーノードの検出とノードの可視化

一般にP2Pネットワークには、ファイル分布情報が集約されるノードがある。P2Pネットワーク上の漏洩ファイル拡散の観測と防止には、これらの情報集約ノードを検出することが重要である。図9は、Winnyネットワーク上に存在するスーパーノードと呼ばれるノードと、その他のノードの階層構造を示したものである。漏洩したファイルがスーパーノードに格納されると、拡散スピードが急激に上昇するため、P2P観測システムでは観測したノードのデータから、スーパーノードを検出するアルゴリズムを備えている。

一般にpure P2Pネットワーク上では、情報、ファイル、クエリが集中する大規模なスーパーノードが形成される。スーパーノードは多数のノードとリンクしているため、スーパーノードはネットワークに関する情報が集約しているだけでなく、スーパーノードに格納されたファイルは急速に拡散してしまうケースがある。そのため、本情報漏洩観測追跡システムでは、ネットワーク上に点在するスーパーノードを検出し、ネットワークの流通状況を迅速に把握するだけでなく、情報漏洩の抑止力を高めている。

図10は、都内に存在するスーパーノードを可視化したものである。放射図の中心から、ノード規模の大きさに合わせて、赤、緑、オレンジのノードが多数リンクしていることを示している。このスーパーノードに流通情報が集約されており、情報漏洩を迅速に観測するためには、スーパーノードを検出、観測することが重要である。

4　まとめ

ネットワークの広帯域化とクラウドコンピューティング、P2Pネットワーク技術の普及に伴い、仮想マシンによって構成される仮想ネットワークを利用する機会が多くなっている。第2期中期計画では、トレーサブルネットワーク技術の研究開発として、浸入検知の精度と観測範囲の拡充を行った。そのため、前期中期計画では、トレーサブルネットワーク観測技術の精緻化、観測レンジの拡充として、仮想マシンの観測、P2Pネットワークの観測の2つを対象範囲として研究開発を行った。

現在急激に進んでいるクラウドコンピューティングの普及の原因の1つに、仮想化技術の発展がある。SaaS、PaaS、IaaSのサービス種別に関わらず、クラウドコンピューティングの定義の1つとして、仮想マシンの利用を提供するサービスとすることができる。そこから、クラウドコンピューティングなどの新形態のシステムのセキュア化には仮想マシンの観測が重要であるとの認識のもと、probing技術の研究開発を行った。

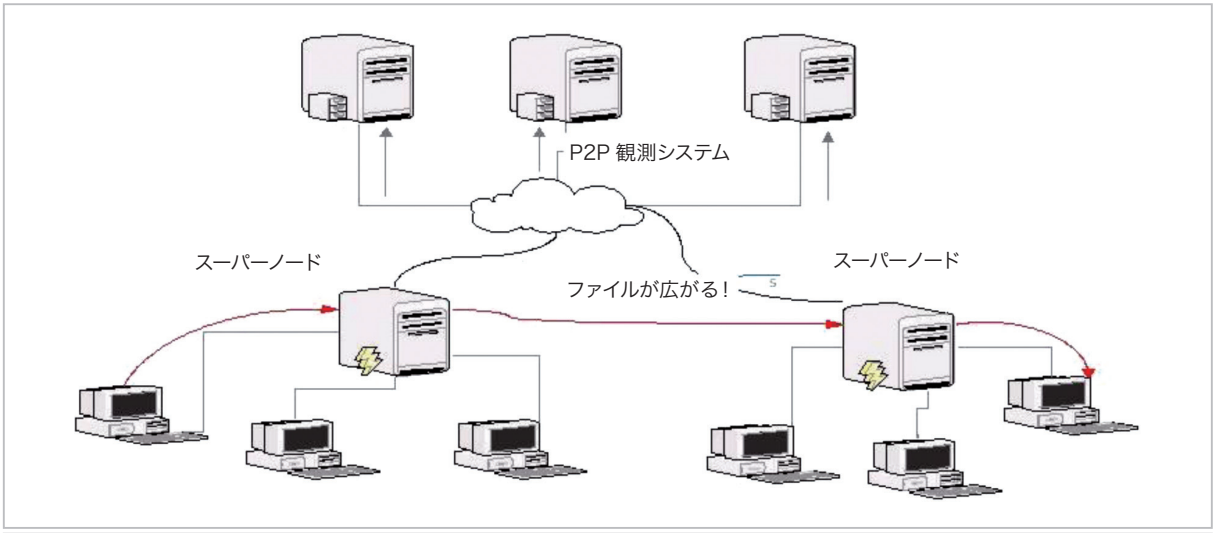


図9　スーパーノードを通じたファイル拡散

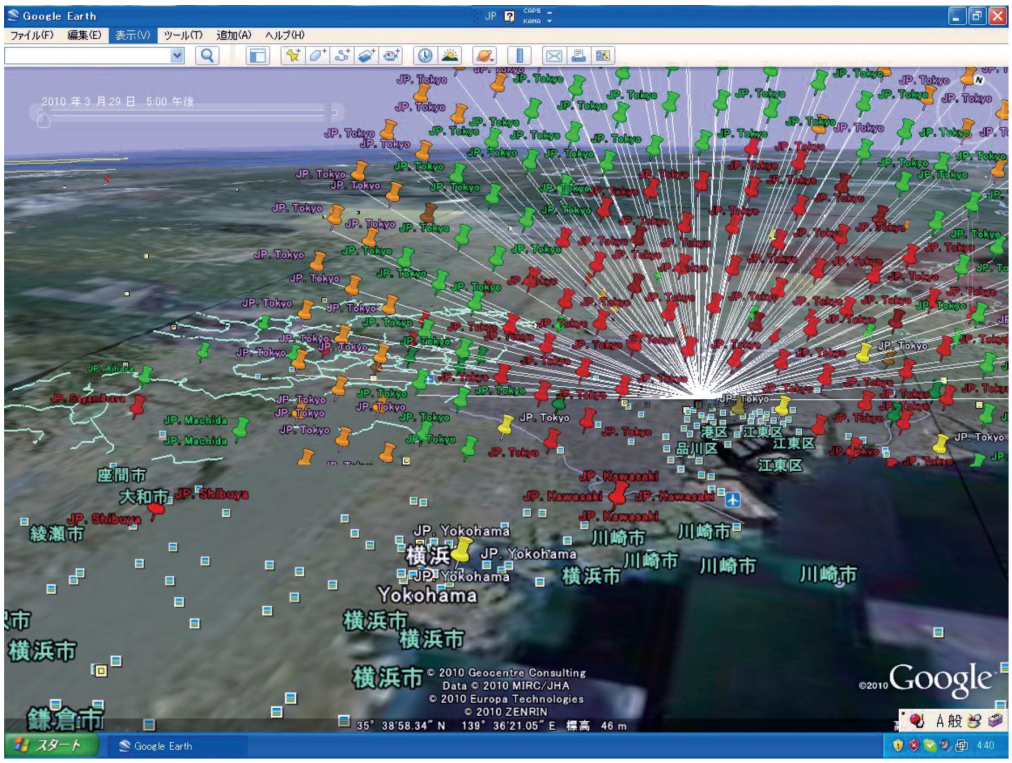


図 10　都内に存在するスーパーノードの可視化

また、前期中期計画期の 2006 年前後から、P2P ネットワークなどのオーバーレイネットワーク上での情報漏洩が、もっとも重要なセキュリティインシデントの 1 つとして社会問題化するようになり、トレーサブルネットワークの研究の 1 つとして、同事象のトレース可能化技術の開発を行った。また、P2P ネットワークから得られる追跡のためのデータは大規模になるため、解析処理機能の強化を行った。

仮想化技術により構成されるシステム上の観測、侵入検知技術は一般に、virtual machine introspection と呼ばれる。仮想マシン上で実装した IDS (Intrusion Detection System) は、監視対象とプローブが仮想化技術によって隔離されているという性質から、従来の IDS にはない機能用件を満たすことができる。従来の NIDS は攻撃者から見えにくいがデータの精度が落ちる一方で、HIDS はデータの精度が上がるが、visibility が増すということが指摘されており、VMM の特性を活かして、この相反する機能用件を止揚する設計が提案されている。IDS を構築する際の仮想マシンモニタの利点は Isolation (プローブがゲスト OS から隔離されて配置される)、Inspection (カーネル空間でのイベントを含む詳細なデータを得ることができる)、Interposition (システムコールや割り込み、VMM への IO 要求などをフックして、その際に中間動作を入れることができる) の 3 点にあると想定される。そのため、この 3 点が侵入検知に特徴を発揮できるように研究開発を行った。

これらの観測システムの研究開発に適用されたプロトコル解析技術、仮想化技術は、今期中期計画のセキュリティアーキテクチャ研究室でのクラウドコンピューティング環境の安全性分析、外部観測、そしてテストベッドでの再現検証の研究にて利用される。

参考文献

1 Ruo Ando, Youki Kadobayashi, and Youichi Shinoda,“Asynchronous Pseudo Physical Memory Snapshot and Forensics on Paravirtualized VMM Using Split Kernel Module,”The 10th International Conference on Information Security and Cryptology (ICISC 2007), Seoul, Korea, November 29–30, 2007.

2 Anh-Quynh Nguyen, Ruo Ando, and Yoshiyasu Takefuji,“Centralized Security Policy Support for Virtual Machine,”LISA 2006, pp. 79–87.

3 安藤類央，門林雄基，篠田陽一，“KVM を用いた完全仮想化上のインシデント駆動型チェックポインタの実装,” 情報処理学会，コンピュータセキュリティシンポジウム 2007.

4 Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Timothy L. Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield,“Xen and the art of virtualization,”SOSP 2003, pp. 164–177.

5 Tal Garfinkel and Mendel Rosenblu,“A Virtual Machine Introspection Based Architecture for Intrusion Detection,”In the Internet Society’s 2003 Symposium on Network and Distributed System Security, NDSS 2003, pp. 191–206.

6 安藤類央，Nguyen Anh Quynh，須崎有康，“VM Introspection を用いた Windows OS のメモリ上の異常挙動の可視化 ,” 第 112 回システムソフトウェアとオペレーティング・システム研究発表会，Aug. 5, 2009.

7 安藤類央，井上宣子，須崎有康，“Windows OS 上でのフィルタドライバを用いたセキュアアクセス制御機構の構築 ,” 情報処理学会，コンピュータセキュリティシンポジウム 2009，Oct. 2009.

8 Ruo Ando, Kazushi Takahashi, and Kuniyasu Suzaki,“A SOM based malware visualization system using resource access filter of virtual machine,”The 2011 International Conference on Computers, Communications, Control and Automation (CCCA 2011), Hong Kong, China, February 20–21, 2011.

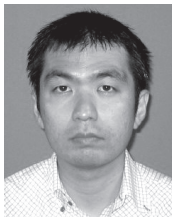
9 安藤類央，門林雄基，篠田陽一，“仮想レジスタ通信を用いた広域 P2P ネットワーク観測データ処理機能の強化 ,” 情報処理学会，コンピュータセキュリティシンポジウム 2010，Oct. 2010.

10 安藤類央，外山英夫，門林雄基，“DLL Injection による P2P ソフトウェアの情報漏洩の追跡と防止 ,” 情報処理学会，コンピュータセキュリティ研究会，第 36 回研究報告，March 1, 2007.

11 Ruo Ando,Youki Kadobayashi, and Yoichi Shinoda,“Blink: Large-scale P2P Network Monitoring and Visualization System Using VM introspection,”NCM 2010: 6th International Conference on Networked Computing and Advanced Information Management, Seoul, Korea, August 16–18, 2010.

12 安藤類央，外山英夫，門林雄基，篠田陽一，“Google Earth とノード内部観測を用いた P2P ノード特性と地理的分布の可視化 ,” 情報処理学会，第 143 回マルチメディア通信と分散処理研究会（DPS）.

（平成 23 年 6 月 15 日 採録）



安藤類央（あんどう るお）
ネットワークセキュリティ研究所
セキュリティアーキテクチャ研究室主
任研究員　博士（工学）
ネットワークセキュリティ

独立行政法人情報通信研究機構発行の技術情報誌「情報通信研究機構季報」Vol.57 Nos.3/4 2011年9･12月号の記事を、著者及び情報通信研究機構の承諾を得て掲載しています。内部を一部加筆修正しています。