

# 2009年度 脆弱性を利用した 新たなる脅威の分析による調査



2010年7月

**IPA**<sup>®</sup>

独立行政法人 情報処理推進機構  
セキュリティセンター

本ページは白紙です

# 目次

1. はじめに.....	6
1.1. 背景.....	6
1.2. 調査の内容と目的.....	6
1.3. 用語集.....	7
2. 「脆弱性を利用した攻撃」の現状および動向に関する調査.....	9
2.1. 近年の脆弱性攻撃の代表例.....	9
2.2. 攻撃の背景.....	11
3. 「Adobe Reader/Acrobat の脆弱性を利用した攻撃」の分析.....	13
3.1. 攻撃メールの分析.....	13
3.1.1. メール内容（本文・ヘッダ・添付ファイル）.....	13
3.1.2. 実際のメール受信者の対応.....	14
3.1.3. 攻撃メールでの利用手口の分析.....	15
3.2. マルウェアの分析.....	17
3.2.1. 解析対象検体.....	17
3.2.2. マルウェア動作概要.....	19
3.2.3. マルウェア動作詳細.....	20
3.2.3.1. 脆弱性の利用（Exploit）.....	20
3.2.3.2. シェルコード.....	22
3.2.3.3. マルウェア 1-1（EXE ファイル）.....	22
3.2.3.4. マルウェア 1-2（DLL ファイル）.....	24
3.2.3.4.1. ServiceMain.....	25
3.2.3.5. 通信内容詳細.....	26
3.2.3.6. 本攻撃における脆弱性利用の環境依存性.....	27
4. 「Adobe Reader/Acrobat の脆弱性を利用した攻撃」で利用される脆弱性の分析.....	29

4.1.	脆弱性の概要	29
4.2.	攻撃手法	29
4.3.	攻撃状況	29
5.	「Microsoft Word の脆弱性を利用した攻撃」の分析	31
5.1.	攻撃メールの分析	31
5.1.1.	メール内容 (本文・ヘッダ・添付ファイル)	31
5.1.2.	実際のメール受信者の対応	32
5.1.3.	攻撃メールでの利用手口の分析	34
5.2.	マルウェアの分析	35
5.2.1.	解析対象検体	35
5.2.2.	マルウェア動作概要	36
5.2.3.	マルウェア動作詳細	37
5.2.3.1.	脆弱性の利用 (Exploit)	37
5.2.3.2.	シェルコード	38
5.2.3.3.	マルウェア 2-1 (EXE ファイル)	38
5.2.3.4.	マルウェア 2-2 (DLL ファイル)	40
5.2.3.4.1.	RundllInstallA	41
5.2.3.4.2.	InstallService	41
5.2.3.4.3.	ServiceMain	42
5.2.3.5.	通信内容詳細	43
5.2.3.6.	本攻撃における脆弱性利用の環境依存性	45
6.	「Microsoft Word の脆弱性を利用した攻撃」で利用される脆弱性の分析	46
6.1.	脆弱性の概要	46
6.2.	攻撃手法	46
6.3.	攻撃状況	46
7.	「脆弱性を利用した攻撃」への対策	48
7.1.	脆弱性/攻撃情報の入手と対応	48

7.2.	直接的な対策	48
7.3.	汎用的な対策	48
7.3.1.	Adobe Reader/Acrobat の脆弱性を利用した攻撃への汎用的な対策	48
7.3.2.	Microsoft Word の脆弱性を利用した攻撃への汎用的な対策	49
7.4.	ダメージコントロール	50
8.	まとめ	51
8.1.	2 ケースの攻撃の比較	51
8.2.	攻撃メールの特徴	52
8.3.	マルウェアの特徴	53
8.4.	利用された脆弱性の特徴	53
8.5.	攻撃タイミングの特徴	53
8.6.	今後に向けて	53

## 1. はじめに

### 1.1. 背景

独立行政法人情報処理推進機構（以下「IPA」という）では、安心して利用できる情報化基盤の構築・維持のため、情報セキュリティ対策の強化、整備を進めている。IPA セキュリティセンターでは脆弱性関連情報の取扱い制度を 2004 年 7 月から開始し、ソフトウェア製品やウェブサイトの脆弱性について適切な情報の流通および対策の促進を図ると共に、届出情報を基に注意喚起や啓発資料の作成を行い、コンピュータ不正アクセスなどによる被害発生抑制に努めている。近年、ソーシャルエンジニアリング（話題や盗み聞き・盗み見などを利用し、人間の心理・行動の隙を突くことで情報を不正に取得する手段の総称）やマルウェア（悪意あるプログラム。ユーザの望まない悪さをするプログラム）などの、情報セキュリティ上の新たな脅威による被害が発生している。

昨年度、IPA は「脆弱性を利用した新たな脅威の監視・分析による調査」を実施し、IPA 設置の窓口、「不審メール 110 番」に 2008 年度第 4 四半期に相談されたマルウェアなどの解析を行い、その攻撃手法や対策についてまとめた。調査の結果、この攻撃の裏側には、ソフトウェアの脆弱性を悪用する、悪意の PDF ファイルを自動的に生成するツールの存在を確認した。また、そういったツールを売買する違法ビジネスが存在することも推測された。さらに、2008 年 4 月に発生した、IPA をかたった「なりすましメール」による攻撃の詳細を明らかにすることで、脆弱性とソーシャル・エンジニアリングを利用した標的型攻撃への対策方法を示した。一方で、セキュリティ対策回避策が組み込まれたり、未知の脆弱性を狙ったりするなど、マルウェアもより攻撃が成功しやすくなるような新しい仕組みを取り入れていると考えられる。

脆弱性を狙った攻撃を行なう環境が整備されつつある一方で、ユーザ側の対策環境の整備や周知徹底は未だ不十分のままである。日々報告される新しい脅威への対策を検討するために、IPA では毎年調査を行っており、本年度も継続して、脆弱性を利用した新たな脅威の詳細な分析を行なっていくと共に、利用された脆弱性の解析や対策について検討することが求められる。本調査は、これらの活動を着実に実施するため、脆弱性を利用した新たな脅威に関し、分析・調査を行なうものである。

### 1.2. 調査の内容と目的

最新の「脆弱性を利用した攻撃」について調査を行い、その詳細を明らかにする。具体的には、ウェブ調査と IPA が提供する実際に発生・悪用されたダウンロードおよびマルウェアなどのプログラムの分析を行なう。

#### (1) 「脆弱性を利用した攻撃」の現状および動向に関する調査

脆弱性を利用した新たな攻撃手法が日々発見・報告される現状にある。そのため、ウェブ調査によって、「脆弱性を利用した攻撃」の現状および動向について調査を行なう。

## (2) 「脆弱性を利用した攻撃」で利用されるダウンローダおよびマルウェアの分析

IPA に届けられた検体について、動的分析および静的分析を実施し、その攻撃シーケンスを分析する。また、攻撃の過程においてシステム上に生成、実行されたダウンローダ、マルウェアの構造、挙動を分析する。これらダウンローダおよびマルウェアが外部のサーバと通信を行なう場合、その通信による当該検体の動作についても分析を行なう。

## (3) 「脆弱性を利用した攻撃」で利用される既知・未知のセキュリティ脆弱性の分析

(2)で分析した「脆弱性を利用した攻撃」で利用される脆弱性について、攻撃手法、攻撃安定性、環境依存性、技術的特徴などの観点から分析する。未知の脆弱性であることが判明した場合についても同様の観点から分析を行なう。

## (4) 「脆弱性を利用した攻撃」の検知および対策の検討

上記の結果を踏まえ、本調査で明らかになった「脆弱性を利用した攻撃」の検知・対策手法について検討し、具体的な対策の提言を行なう。

### 1.3. 用語集

本報告書で使用する用語の意味を以下に定義する。

用語	定義
Exploit コード	脆弱性を利用するためのプログラムコードまたはその仕掛けのこと。本報告書では、攻撃者が意図した目的の動作を行うプログラムコード(シェルコード)を実行させるために行う、脆弱性を利用しようとする仕掛けまたはそのためのプログラムコードのことを「Exploit コード」と呼ぶ。
コード実行	CPU が理解する機械語などのプログラムコードが CPU によって実行されること。
シェルコード	脆弱性を利用した後に実行されるプログラムコードのこと。本報告書では、脆弱性を利用しようとする仕掛けまたはそのためのプログラムコード(Exploit コード)によって実行される、攻撃者が意図した目的の動作を行うためのプログラムコードのことを「シェルコード」と呼ぶ。
ゼロデイ攻撃/ゼロデイ脆弱性	脆弱性を塞ぐためのセキュリティパッチが提供されていない状態でその脆弱性を利用して行う攻撃、またはその脆弱性のこと。
ソーシャルエンジニアリング	コンピュータを利用した技術的手段によらず、人間の心理的な癖につけこむなど「社会的」な手段によって秘密情報を入手しようとする手口のこと。
ホワイトボックス解析	解析対象の中身の構造を調べることによって解析対象について調べること。マルウェア解析においては、静的解析/動的解析にかかわらず、逆アセンブラ、デバッガなど

用語	定義
	で機械語などの実行コードを参照しながらその動作を解析する手法のことを指す。
ブラックボックス解析	解析対象の中身を知らずに必要な入力を与えた際に得られた出力のみを見て解析対象について調べること。マルウェア解析においては、実行ファイルの実行によって行われた挙動(ファイル/レジストリ/プロセスの作成/変更/削除や送受信した通信内容など)から実行コードの動作を解析する手法のことを指す。
逆アセンブラ	CPU が解釈可能な機械語をアセンブリ言語に変換するためのプログラムやアプリケーションのこと。マルウェア解析においては、マルウェアである実行プログラムファイル(EXE、DLL など)に含まれている機械語の実行コードが行う動作を調べるためなどに使用される。
デバッガ	実行プログラムのデバッグを支援するプログラムやツールのこと。マルウェア解析においては、マルウェアである実行プログラムファイル(EXE、DLL など)に含まれている機械語の実行コードの動作をステップ単位で実行させながら、実際の動作を確かめるためなどに使用される。
ガンブラー	不特定多数のウェブサイトと同じ種類のスクリプトを追加することによってマルウェアに感染させようとする手口のこと。送り込んだマルウェアを利用して盗んだ FTP アカウント情報をウェブ改ざんに悪用するため、ウェブ改ざんが連鎖的に広がることが多い。
バックドア	コンピュータに正規の方法ではなく不正に侵入する方法を提供するためのプログラム、またはその侵入口のこと。バックドアプログラムが能動的に C&C サーバへ接続して指令となるコマンドを受信しようとすることも多く、この場合は「ボット」と呼ばれることもある。攻撃者は C&C サーバを利用して任意のタイミングで任意のコマンドをバックドアプログラムに送り込むことで、バックドアプログラムが仕掛けられたコンピュータを自由に制御する。バックドアによる影響は、バックドアプログラムが持つ受信可能なコマンドの種類、および実際に受信したコマンドの種類に依存する。
C&C サーバ	Command and Control サーバ。攻撃者がバックドアプログラムに対して指令となるコマンドを送信し、バックドアプログラムが仕掛けられたコンピュータの動作を制御するために用いられる。

## 2. 「脆弱性を利用した攻撃」の現状および動向に関する調査

### 2.1. 近年の脆弱性攻撃の代表例

近年、継続的に発生している「脆弱性を利用した攻撃」の脅威の詳細、変化を明らかにするため、関係情報を公開しているセキュリティベンダーなどのウェブサイトなどから攻撃発生状況の調査を行なった。その結果として、脆弱性を利用した攻撃の手法として、1)メール感染型、2)ネットワーク感染型、3)ウェブ感染型、の大きく三つに分類し、それぞれの種類とその代表例について紹介する。

#### (1) メール感染型

メールに添付されたファイルを開くことによって脆弱性が利用されてマルウェアに感染する種類を、ここでは「メール感染型」として分類する。マルウェアが添付されたメールとしては、マルウェアが勝手に大量の宛先に対してマルウェアを添付したメールを送信する「マスメーリングワーム」が知られているが、近年はマルウェアを添付したメールを特定の宛先のみを送りつける手法、いわゆる「標的型メール攻撃」の存在が確認されている。この標的型メール攻撃では、添付ファイルが安全なファイルであると油断して開かせるソーシャルエンジニアリング的手法の一つとして、アプリケーションのデータファイルを添付する事例が多く確認されている。データファイルであってもアプリケーションの脆弱性を利用するよう細工することによって、攻撃者の任意のコードが実行され、結果として利用者の意図しない不正な動作が行われる被害に遭う。被害に遭った組織では攻撃を受けた事実の公表を好まないことが多いため、攻撃/被害状況を正確に掴むことは困難であり代表例を挙げるのが難しい。そこで本報告書では、IPA「不審メール 110 番」に相談のあったマルウェアをサンプルとして解析し、この実例からこの攻撃の特徴の調査と対策方法の検討を行う。

#### (2) ネットワーク感染型

ネットワーク上の通信によって脆弱性を狙う攻撃コードが攻撃対象のコンピュータへ直接送り込まれ、狙われた脆弱性が存在する場合に攻撃対象のコンピュータ上で攻撃者の任意のコードを実行してしまい、これによってマルウェアに感染する種類を、ここでは「ネットワーク感染型」として分類する。CodeRed、Nimda、SQL Slammer などこれまでに大きな被害を及ぼしたネットワークワームがよく知られているが、現在もこのようなネットワーク感染型の被害は止んでいない。

2008 年末に確認された MS08-064 脆弱性を利用する Conficker (別名 Downadup) は複数の亜種が作成されると共に被害も拡大した。MS08-064 はリモートからのコード実行が可能な脆弱性であり、これを利用したワームがネットワーク越しに脆弱性を持つコンピュータを探し出し感染活動を行なうこと、および USB メモリ経由など複数の感染経路を持つことから、セキュリティパッチ未適用のコンピュータを多数持つ組織などで大きな被害が発生した。実際、多くのセキュリティベンダーからの 2009 年のマルウェア被害年間レポートの上位に Conficker が挙げられている。リモートからのコード実行が可能な脆弱性が大きな被害につながる可能性が高いことは、以前から知られていたにもかかわらずこのような実際の被害が及んだことは、利用者側での迅速なセキュリティパッチ適用の困難さ、およびウイルス対策ソフト使用の徹底の困難さを物語っている。このようにネットワーク感染型ワームは依然として大きな脅威であり、組織内ネットワークの通信経路上での不正通信パケットを検出するソリューションの普及促進といった、さらなる対策の必要性が

窺える。

### (3) ウェブ感染型

ウェブページ閲覧を契機として脆弱性が利用されマルウェアに感染する種類を、ここでは「ウェブ感染型」として分類する。ウェブページ閲覧によってマルウェア感染の動作を引き起こすには、ウェブページ内にその動作を引き起こすためのコードを何らかの方法で忍びこませる必要がある。そのための方法として攻撃者は従来から様々な手法を用いてきた。マルウェアによるローカル/ネットワークドライブ内の HTML ファイルなどの直接書き換え、ARP スプーフィングによる通信パケット内の HTML データへの直接挿入、SQL インジェクションによるデータベースの改ざん、などである。それぞれの手法の紹介は本報告書の主題ではないので割愛するが、ここでは最近最も大きな被害を及ぼしている「ガンブラー」について紹介する。

正規ウェブサイトを改ざんして不正スクリプトを埋め込み不正ウェブサイトへ誘導する手法、いわゆる「ガンブラー」は、2009年4月に発生してから、2009年10月、2009年12月と埋め込むスクリプトの種類や結果としてダウンロードされるマルウェアの種類を変えながらもその度々の大きな被害が問題になった。「ガンブラー」では、マルウェアによって主に FTP アカウント情報が収集された。ウェブサーバ上のコンテンツの更新には FTP を利用することが多い。そこで攻撃者は収集した FTP アカウント情報を悪用して FTP サーバへ侵入し、サーバ上の HTML ファイルや JS ファイルに攻撃用コードを埋め込んだのである。この攻撃用コードとは、実際には JavaScript や IFRAME タグを利用して別の攻撃用ウェブページへ誘導することのみを行うことが多い。そして、誘導先の攻撃用ウェブページでは、JavaScript を利用して複数の脆弱性を併用し、どれか一つでも脆弱性が存在すれば存在した脆弱性の種類に関わらず同じマルウェアをダウンロードするという手法が使用された。例えば 2009年12月末から被害が発生している「ガンブラー」（通称、Gumblar.8080）では、少なくとも以下の脆弱性が利用されていることを確認している（JVNDDB 番号順）。

- JVNDDB-2006-000166 (CVE-2006-0003)

「MDAC の RDS.Dataspace ActiveX コントロールにおける任意のコードを実行される脆弱性」

<http://jvndb.jvn.jp/ja/contents/2006/JVNDDB-2006-000166.html>

- JVNDDB-2008-001494 (CVE-2008-2463)

「Microsoft Office Snapshot Viewer ActiveX コントロールにおける任意のファイルをダウンロードされる脆弱性」

<http://jvndb.jvn.jp/ja/contents/2008/JVNDDB-2008-001494.html>

- JVNDDB-2008-001095 (CVE-2007-5659)

「Adobe Reader および Adobe Acrobat の JavaScript メソッドにおけるバッファオーバーフローの脆弱性」

<http://jvndb.jvn.jp/ja/contents/2008/JVNDDB-2008-001095.html>

- JVNDB-2009-001885 (CVE-2008-0015)  
「Microsoft Windows の MPEG2TuneRequest ActiveX コントロールにおけるバッファオーバーフローの脆弱性」  
<http://jvndb.jvn.jp/ja/contents/2009/JVNDB-2009-001885.html>
- JVNDB-2008-001914 (CVE-2008-2992)  
「Adobe Acrobat および Reader における util.printf JavaScript 関数を呼び出す PDF ファイル処理に関するバッファオーバーフローの脆弱性」  
<http://jvndb.jvn.jp/ja/contents/2008/JVNDB-2008-001914.html>
- JVNDB-2008-002152 (CVE-2008-5353)  
「Java Runtime Environment (JRE)における権限昇格の脆弱性(JRE 脆弱性)」  
<http://jvndb.jvn.jp/ja/contents/2008/JVNDB-2008-002152.html>
- JVNDB-2009-002451 (CVE-2009-4324)  
「Adobe Reader および Acrobat における解放済みメモリを使用する脆弱性」  
<http://jvndb.jvn.jp/ja/contents/2009/JVNDB-2009-002451.html>

マルウェアをばら撒く機会を窺っている攻撃者の観点では、ウェブサイトのウェブページ閲覧という多くの利用者が日常的に行う能動的な行為は、格好の標的となる。また、複数の脆弱性の併用は、より多くの利用者への感染を可能にする。OS やすべてのアプリケーションが最新の状態に保たれていればどのような既知の脆弱性が狙われようとも防ぐことが可能であるが、インストールしていることの認識度が低いアプリケーションや、最新バージョンへの自動更新機能がないアプリケーションの場合には、脆弱性が放置されている可能性が高くなる。また、ゼロデイ脆弱性が利用された場合には感染可能性が飛躍的に高まる。利用者としては、脆弱性を塞ぐという基本的対策の徹底が必要であることはもちろんだが、一つの対策では完全な対策にならない可能性があることを踏まえ、埋め込まれたスクリプトや結果的にインストールされるマルウェアの動作など全体を一連の流れとして捉えて、その流れのいずれかで流れを断ち切ることで実際の被害発生を防ぐという考え方への転換が求められている。

## 2.2.攻撃の背景

ソフトウェアの脆弱性を利用した攻撃を行なうには、攻撃者は脆弱性の存在とその利用方法を事前に明らかにしておく必要がある。しかし、新しい脆弱性を発見するには一般に高い技術力が必要となる。したがって、攻撃者自身が攻撃に利用する脆弱性を発見しているのではなく、他者が発見した脆弱性情報を何らかの方法で入手していると考えの方が妥当であろう。では攻撃者はどのようにして攻撃に利用可能な脆弱性情報を入手しているのだろうか。いくつかの手段が考えられるが、まず一つ目としては、前回の報告書で取り上げた「マルウェア売買のビジネスモデル」と同様に脆弱性情報が売買されていることが挙げられる。また、

同じく前回の報告書で触れられた「マルウェア作成ツール」が作成され、脆弱性に関する技術的知識の高くない者でも、高度な脆弱性を利用した攻撃が簡単に行なえるようになっている場合があると考えられる。

二つ目には、セキュリティ研究者によって調査され公開された脆弱性情報を、攻撃に悪用することが考えられる。この場合「マルウェア作成ツール」を利用するほど簡単ではないが、自ら新たな脆弱性の存在を調査して利用可能にするほどには、高い技術力は必要ない。日本国内においては脆弱性情報の受付期間であるIPAと調整機関であるJPCERT/CCによって、対策方法が整わない時点で情報が公開されないよう調整されているが、世界的には脆弱性の存在を実証するためのプログラム（Proof of Concept）をセキュリティ研究者が公開する場合があります、そのことが実際の攻撃を誘発している可能性が考えられる。このような状況が発生したと考えられるケースの例として、ゼロデイ攻撃が確認された時期がPoC公開後である脆弱性の例を以下に挙げる。

#### PoC 公開後にゼロデイ攻撃が行われた最近の主な脆弱性：

- JVNDB-2008-001894 (CVE-2008-4250)  
「Microsoft Windows の Server サービスにおける RPC リクエストに関する任意のコードを実行される脆弱性」  
<http://jvndb.jvn.jp/ja/contents/2008/JVNDB-2008-001894.html>
- JVNDB-2010-001058 (CVE-2010-0249)  
「Microsoft Internet Explorer において任意のコードが実行される脆弱性」  
<http://jvndb.jvn.jp/ja/contents/2010/JVNDB-2010-001058.html>
- JVNDB-2010-001171 (CVE-2010-0806)  
「Microsoft Internet Explorer における解放済みメモリを使用する脆弱性」  
<http://jvndb.jvn.jp/ja/contents/2010/JVNDB-2010-001171.html>

また、ゼロデイ攻撃であるかどうかにかかわらず、実際の攻撃で利用された攻撃コードが一部のセキュリティ研究者やセキュリティベンダーによって公開されることもある。このような攻撃コードの公開は対策方法の研究目的で行なわれることが多い。攻撃と対策は表裏の関係にあるので、対策のためにある程度の攻撃情報を出さざるを得ない場面は存在するが、負の側面としてこの情報に触れた第三者による類似の攻撃を誘発する危険性があることも認識しなくてはならない。したがって、セキュリティ研究者による調査やその成果の公表自体は推奨されるが、調査によって得られた情報が攻撃に悪用される可能性がある現実を踏まえ、公表の方法には細心の注意を払うことが望まれる。

### 3. 「Adobe Reader/Acrobatの脆弱性を利用した攻撃」の分析

本章では、IPA 設置の「不審メール 110 番」に相談のあった「Adobe Reader/Acrobat の脆弱性を利用した攻撃」における攻撃メールおよびマルウェアを分析し、以下の点を明らかにする。

- 攻撃メールの特徴
- 利用している脆弱性の特徴
- マルウェアの動作
- 通信内容

#### 3.1. 攻撃メールの分析

本節では、「Adobe Reader/Acrobat の脆弱性を利用した攻撃」に用いられた攻撃メールの内容および利用された手口を分析する。

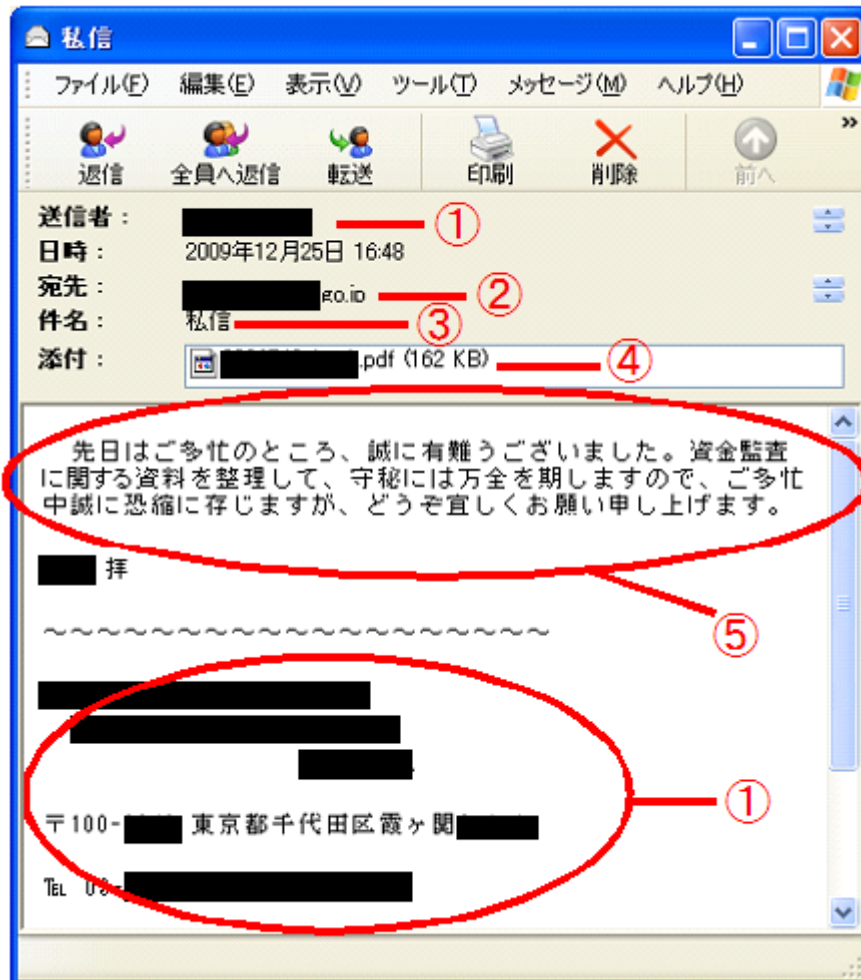
##### 3.1.1. メール内容（本文・ヘッダ・添付ファイル）

IPA 「不審メール 110 番」に相談のあったメールの本文およびヘッダの内容を以下に示す。

不審メールの構成：

項目	内容
差出人	・公的機関の实在の人物のメールアドレス(詐称)
宛先	・受信者が業務のため加入しているメーリングリスト
送信時間	・2009/12/15(金) 16:48
件名	・「私信」
本文	・「先日はご多忙のところ、誠に有難うございました。資金監査に関する資料を整理して、守秘には万全を期しますので、ご多忙中誠に恐縮に存じますが、どうぞ宜しくお願いします。」 ・实在の人物をかたった署名
メール形式	・HTML およびテキスト形式 (multipart/alternative)
添付ファイル	・マルウェア1が添付

メールを表示した様子(Outlook 2000 を使用):



メールの特徴は以下の通りです。

- ① メールを受信者が信頼することを狙った、官公庁をかたる署名
- ② 宛先は業務用メーリングリスト
- ③ メーリングリストの用途に合わない件名
- ④ 添付ファイルは PDF ファイル (実態はマルウェア)
- ⑤ 件名と結びつかない本文

### 3.1.2. 実際のメール受信者の対応

このメールを受信者から、メール受信時の状況について聞き取り調査を行った。聞き取り調査によって判明した受信時の状況を以下に示す。

#### (1) メールを見ての感想

- 差出人と同じ名字の職員がいるため、その人からのメールかと思った。
- 件名が「私信」となっているため、不信感を持った。

- 宛先アドレスが個人宛てではなく、メーリングリストであったことから、このアドレスに「私信」を送るのはおかしいと思った。
- 差出人に記憶はない。
- メール本文の「先日は・・・」という始まりに不信感を持つ。この時点で多分添付ファイルにウイルスが入っているだろうと思い、添付ファイルを開く意思はなかった。

## (2) メール受信後の対応

- 差出人の所属先が存在するかどうかを確認し、存在しないことが分かったので IPA へ報告した。
- 該当メールを読む可能性がある機関へ、添付ファイルを開くことなくメールを削除するように連絡した。

### 3.1.3. 攻撃メールでの利用手口の分析

マルウェアが添付されたメールとしては、マルウェアが勝手に大量の宛先に対してマルウェアを添付したメールを送信する「マスメーリングワーム」が知られているが、このメールに添付されているファイルには勝手に大量の宛先へメールを送信する機能は存在していない。このことから、このメールは攻撃者が任意の宛先に対して意図的に送信したメールであると考えられる。このようにマルウェアを意図的に添付して任意の宛先にのみ送信されるメールは「標的型攻撃メール」、このようなメールを送信してマルウェアに感染させようとする手口は「標的型メール攻撃」と呼べる。本節では、今回の「標的型メール攻撃」で使用された「標的型攻撃メール」の内容と詳しい手口について、以下の点から分析を加える。

- メール の 件名 と 本文 の 自然 さ
- メール の 差出人 と 受信者 の 自然 さ
- メール の 送信 方法
- 添付ファイルの種類およびファイル名と開きやすさ
- 添付ファイルを開いた後の目的動作の達成しやすさ
- 実行中マルウェアの存在の気付きやすさ

#### (1) メール の 件名 と 本文 の 自然 さ

標的型攻撃メールでは、件名や本文の内容や使用されている言語は、受信者に添付ファイルを開かせるための重要な要素である。例えば、英語で書かれたメールは、主に日本語のみを使用して生活する者にとってはその内容以前に英語であるというのみで通常届くはずのない不審なメールであると簡単に認識することができる。この標的型攻撃メールでは受信者が日本人であり差出人も日本人の名前になっている。また、メール本文と添付ファイルの件名、および添付ファイルを開いた後に作成されるダミーの正常な文章ファイルの内容がいずれも日本語で書かれている。しかし、このメールの受信者は差出人とメール本文の「先日は」という書き方に不審感を持ち、添付ファイルがマルウェアである可能性があるとの疑いを持ったという。送信先を誤って送信されたメールであると仮定してもメール内容に興味を持ち添付ファイルを開く可能性はあるが、今回のように不審感を持たれ攻撃が失敗する可能性も高く、攻撃成功率が低い手法が使われたと言える。

しかしこの手法では宛先を問わず攻撃できることから、同じ内容の攻撃が複数の宛先に対して行われた可能性が考えられる。

## (2) メール差出人と受信者の自然さ

標的型メール攻撃では、誰から送信されてきたメールであるかは、メール内容と併せて受信者がメールを信じて添付ファイルを開くか、あるいは不自然さを感じ不審感を持つかどうかに関わる重要な要素である。今回の標的型攻撃メールでは、メール受信者が差出人と面識がなく、さらにメールに書かれていた差出人の所属先に存在するか確認し、存在しないことが分かったことから不審なメールであると気付いている。

## (3) 添付ファイルの種類およびファイル名と開きやすさ

添付ファイルの種類は、どのようにして不正動作を引き起こすかに関わる重要な要素である。また、ファイル名はメール本文と同様に内容によっては不審感を持つ要因となるため重要な要素であると言える。一般に Windows OS のコンピュータ上で何かの動作をさせるには拡張子「EXE」などの実行ファイルが用いられるが、一般的な利用場面において実行ファイルを添付することは少ないため、メールシステム上でのブロックや検出が容易である。このため、メールに添付されていても違和感がない DOC ファイルなどのデータファイルを用いて、アプリケーションの脆弱性を利用して不正コードを実行させる手口が用いられることが多い。しかし、アプリケーションの脆弱性を用いた攻撃は標的となるコンピュータ上でそのアプリケーションがインストールされていることが攻撃成功の必須要件となる。このため、事前調査によって標的が利用していることが判明しているアプリケーションを用いるか、デファクトスタンダードのような一般に広く利用されているアプリケーションを用いることが多い。今回の標的型攻撃メールでは、添付ファイルは PDF ファイルである。拡張子は「.pdf」で二重拡張子や大量の空白文字の利用などによる拡張子の偽装は行われていない。脆弱性の対象アプリケーションである「Adobe Reader/Acrobat」がインストールされていなければ脆弱性の利用はできないが、Adobe Reader/Acrobat は一般に多く利用されているソフトウェアであることから、受信者が Adobe Reader/Acrobat を利用している可能性は高いと考えられる。

## (4) 添付ファイルを開いた後の目的動作の達成しやすさ

攻撃者は情報搾取などの明確な目的を持っており、添付ファイルを開かせることによってあらかじめ用意した実行コードを動作させることによって、目的達成を図っている。脆弱性を利用した攻撃では、標的となったコンピュータ上で利用された脆弱性が存在していることが攻撃成功の必須要件である。また、OS 種類などプログラムの動作に必要な要件を満たしていること、ウイルス対策ソフトなどによって添付ファイルやインストールする別のマルウェアの動作がブロックされないこと、そして動作にインターネット上のホストとの通信を必要とする場合には、通信が途中で阻害されずに正しく行えることが必須要件である。今回の添付ファイルでは APSA09-07 の脆弱性が利用されている。この脆弱性は今回の攻撃メールの受信日（2009年12月25日）時点では、脆弱性の存在や緊急対策方法は公開されていたもののセキュリティパッチは公開されておらず、いわゆるゼロデイ攻撃の状態であった。この状況では、メール受信者の環境では適切な対策がなされておらず脆弱性が存在している可能性が高いと考えられることから、脆弱性の利用に成功する確率は

高い。ただし、今回の添付ファイルを開いた場合に行われる動作ではその通信に一般的には使用されないポート番号（13522）が用いられることから、ファイアウォールなどで比較的ブロックされやすく攻撃者の攻撃目的が達成される可能性は低くなると考えられる。

#### (5) 実行中マルウェアの存在の気付きやすさ

添付ファイルを開かせることに成功しても、メッセージの表示やコンピュータの通常の動作への支障など目立つ動作があると、コンピュータ利用者がマルウェアの存在に気付き、攻撃者の目的が達成される前にマルウェアの削除などの対処が行われてしまうため、マルウェアが動作していることに気付かれにくくすることは攻撃者の観点からは重要な要素である。このため、マルウェアは開かれた不正なドキュメントファイルが開かれた不正動作を行う際にわざと正常なドキュメントファイルを作成して開くことで利用者が意図した動作が行われていると偽装したり、不正なドキュメントファイルの内容から不正部分を取り除き、一度開かれた不正ドキュメントファイルを調査しても不正なファイルであったことを分からないように細工したりすることがある。今回の標的型攻撃メールでは、ダミーの正常な PDF ファイルが作成されて開かれる。またその内容はメール本文の内容と整合した内容であるため、利用者はメールの添付ファイルを開いただけでは不正ファイルを開いたとは気付かない可能性がある。また、バックドアである DLL ファイルはネットワークサービスを動作させるために汎用的に利用されるプログラムである「svchost.exe」を利用して動作するため、不正なプロセスの存在を見つけるというアプローチによるマルウェアの存在調査手法では検出されにくい。さらに、バックドアである DLL ファイルをインストールする EXE ファイルは、DLL ファイルのインストール後に DLL ファイルのタイムスタンプを変更した上で自分自身のファイル（EXE ファイル）を削除するため、後からバックドアである DLL ファイルの存在に気付いても、その DLL ファイルがいつ何によって作成されたものなのかを調査することが困難である。

### 3.2. マルウェアの分析

本節では、「Adobe Reader/Acrobat の脆弱性を利用した攻撃」に用いられたマルウェアの動作を分析する。

#### 3.2.1. 解析対象検体

解析対象検体の概要を以下に示す。

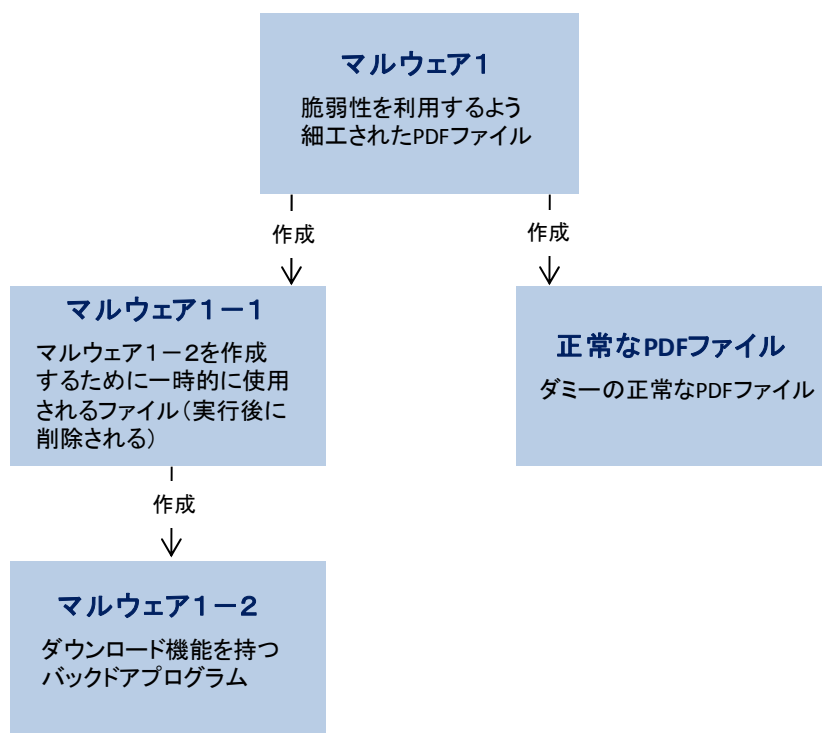
マルウェア	マルウェア1
概要	<ul style="list-style-type: none"><li>● 攻撃メールに添付されていたファイル</li><li>● APSA09-07 の脆弱性を利用するよう細工された PDF ファイル</li><li>● 脆弱性の利用に成功すると、マルウェア1-1 (EXE ファイル) とダミーの正常な PDF ファイルを作成する</li></ul>

解析対象検体の実行によって作成されるファイルの概要を以下に示す。

マルウェア	マルウェア1-1 (EXE ファイル)
概要	<ul style="list-style-type: none"> <li>埋め込まれたファイルデータを使用してマルウェア1-2 (DLL ファイル)を作成する</li> <li>作成したマルウェア1-2をサービスとして登録する</li> </ul>
ダミーファイル	正常な PDF ファイル
概要	<ul style="list-style-type: none"> <li>ダミーの正常な PDF ファイル</li> <li>メールに添付されていた PDF ファイルと同名 (ファイル名は可変ではなく固定)</li> </ul>
マルウェア	マルウェア1-2 (DLL ファイル)
概要	<ul style="list-style-type: none"> <li>バックドアプログラム</li> <li>Windows のサービスとして登録され動作する</li> </ul>

各ファイルの関連を以下に示す。

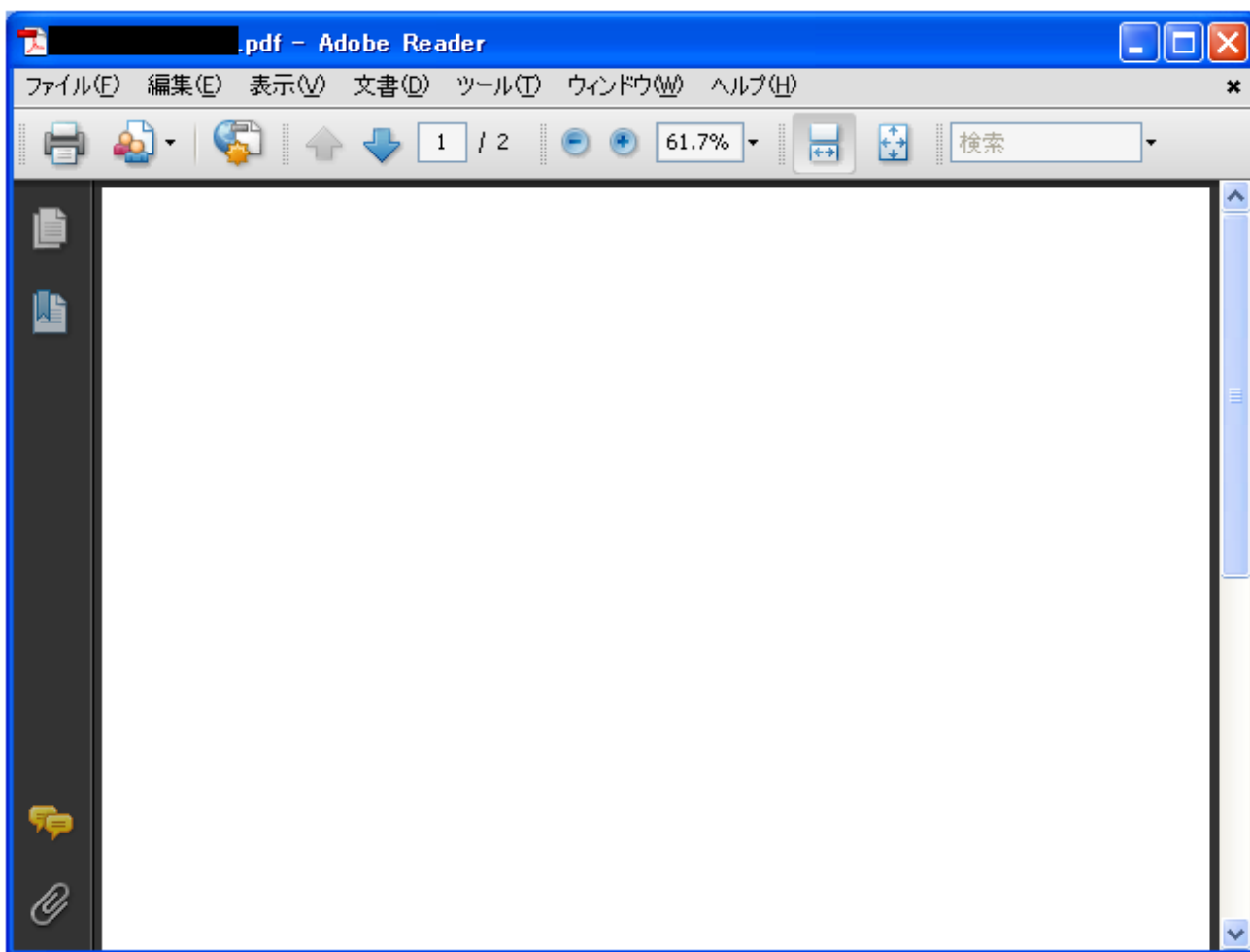
ファイル関連図：



### 3.2.2. マルウェア動作概要

攻撃メールに添付されていたファイル「マルウェア 1」は、Adobe Acrobat/Reader 9.2.0 以前のバージョンの脆弱性を利用し、Windows プラットフォームで任意のコードを実行するよう細工された PDF ファイルである。この脆弱性および影響を受ける OS 種類の詳細は、4 章で述べる。Windows プラットフォーム上で添付されている不正 PDF ファイルが開かれると、その不正 PDF ファイルを開いたアプリケーション (Adobe Reader/Acrobat など) に狙われた脆弱性が存在する場合、その脆弱性が利用されてシェルコードが実行される。脆弱性が存在しない場合にはシェルコードは実行されない。この脆弱性が修正されている Adobe Reader バージョン 9.3 で開いた場合には、以下のような何も書かれていない真白な PDF が表示され、エラーメッセージなどは表示されない。

この脆弱性が修正されている Adobe Reader バージョン 9.3 で開いた場合の画面例：



脆弱性の利用に成功しシェルコードが実行されると、シェルコードでは添付されていた不正な PDF ファイルに埋め込まれているデータを利用して、別の不正ファイル「マルウェア 1-2」を Windows のテンポラリフォルダ (Windows の一時ファイル保存場所) 内に作成して実行する。さらに、正常な PDF ファイルも

Windows のテンポラリフォルダ内に作成し、実行中プロセス（Adobe Reader/Acrobat など）の実行ファイルを利用して開く。

ここで作成される正常な PDF ファイルは添付ファイルと同名のファイルであり、正常な PDF ファイルのコンテンツが含まれている。このように正常な PDF ファイルを開くのは、添付ファイルを開いたユーザーに意図した動作が行われていると偽装するためであると考えられる。なお、この正常な PDF ファイルは、ウェブサイト上で一般に公開されているファイルであることを確認している。

マルウェア 1-1 は実行されると自身のファイル内に埋め込まれているデータを利用して別の不正ファイルであるマルウェア 1-2 を作成し、サービスとして登録する。また、処理の最後で自身のファイルを削除する。自身のファイルを削除するのは、感染後に感染経路の調査を困難にするためであると考えられる。

マルウェア 1-2 は実行されると外部のサーバへ通信してコマンドの受信を待ち受ける。この動作はいわゆるバックドアであるが、バックドアとしての機能はファイルのダウンロードと実行のみである（バックドア機能の詳細は 3.2.4.章で述べる）。攻撃者はこのダウンロード機能を利用して秘密情報の入手など攻撃者の攻撃動機を満たすための別のマルウェアを送り込むと考えられるが、本解析時点では接続先のサーバ（C&Cサーバ）への接続は既にできない状態であったため、攻撃者の本来の意図は不明である。このように作成されるファイルだけでは本来の意図を見えないようにすること、および接続可能な時間などを制限することは、攻撃の全貌が明らかになることを防ぐために攻撃者が意図的に行っている可能性が考えられる。

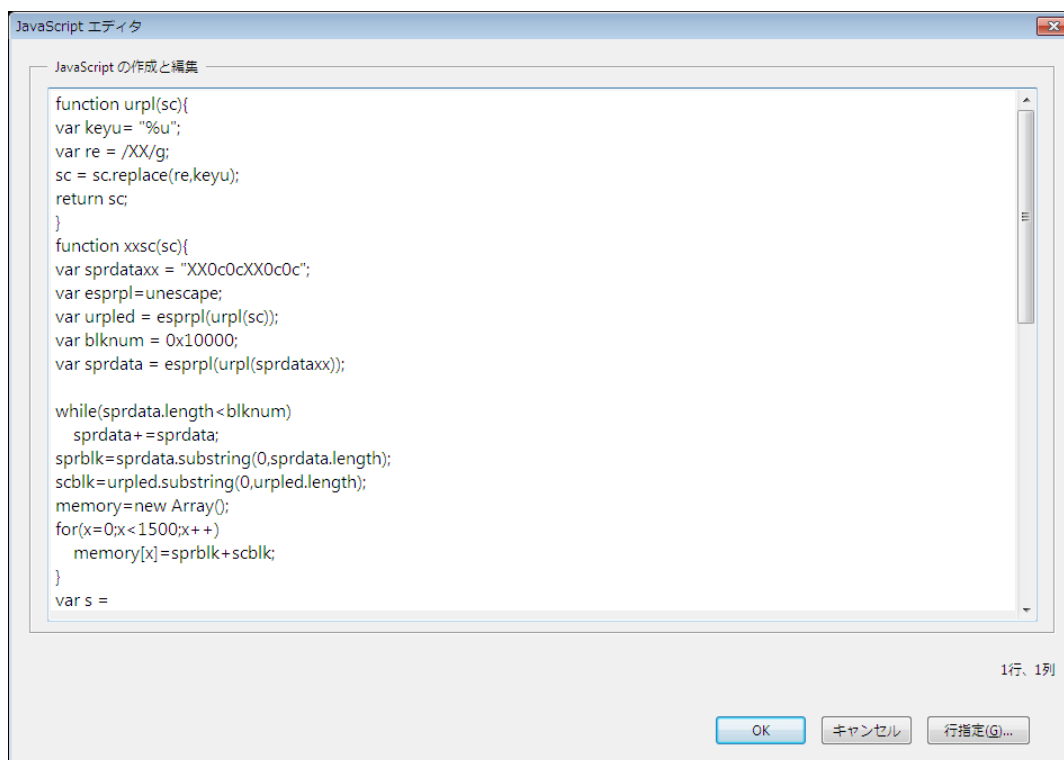
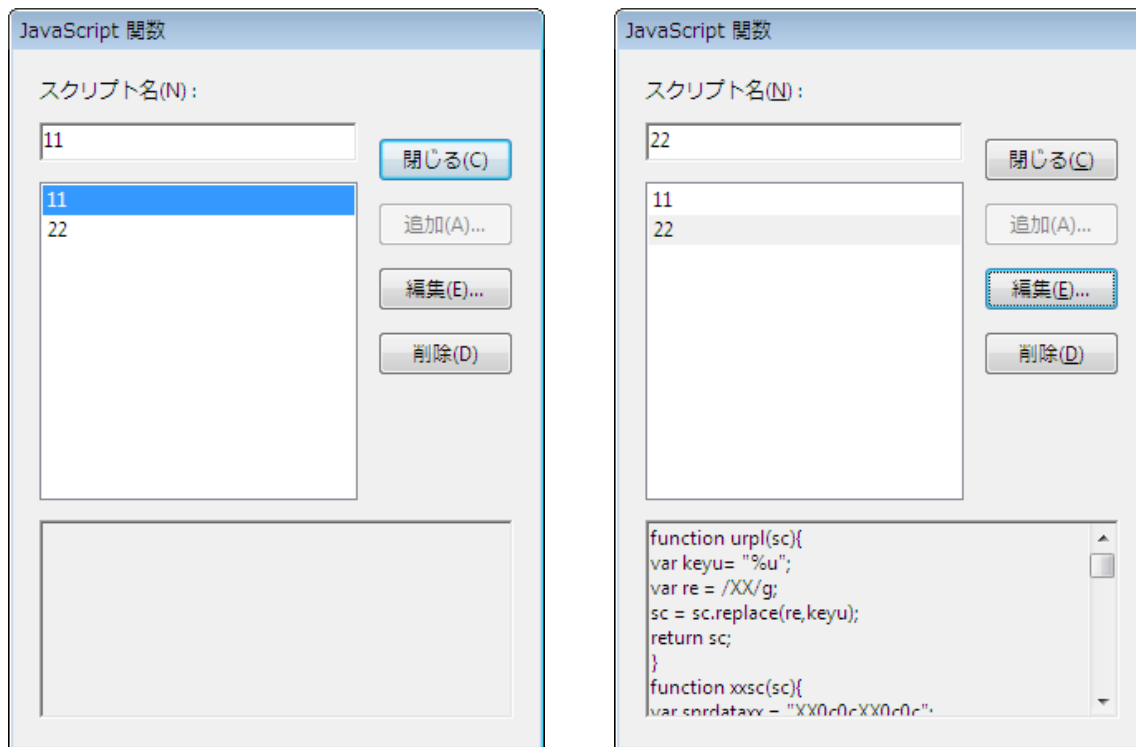
これらのファイルの実行コードの特徴としては、コードに解析を困難にするような工夫がほとんど施されていないことが挙げられる。シェルコードでも近年は API 呼び出し手法を工夫してデバッガ利用時にどの API が呼び出されているか単純には分からないようにするなど巧妙化が進んだものが存在しているが、今回のシェルコードは 2 段階に分かれてはいるもののどちらも比較的単純な作りであり解析は困難ではない。また、マルウェア 1-1 は UPX 形式で圧縮されており、マルウェア 1-2 ではプログラム内で使用する文字列が一部暗号化されているが、どちらも簡単に解析可能なレベルであり、ホワイトボックス解析の大きな妨げにはならない。

### 3.2.3. マルウェア動作詳細

#### 3.2.3.1. 脆弱性の利用 (Exploit)

マルウェア 1 には「文章レベル (Document Level) の JavaScript」が埋め込まれており、Adobe Reader/Acrobat によって開かれると埋め込まれた JavaScript が自動的に実行される（参考：[http://livedocs.adobe.com/acrobat\\_sdk/9/Acrobat9\\_HTMLHelp/wwhelp/wwhimpl/js/html/wwhelp.htm?href=JS\\_Dev\\_Overview.71.1.html&accessible=true](http://livedocs.adobe.com/acrobat_sdk/9/Acrobat9_HTMLHelp/wwhelp/wwhimpl/js/html/wwhelp.htm?href=JS_Dev_Overview.71.1.html&accessible=true)）。この JavaScript 自動実行は Adobe Reader/Acrobat が保有している機能であり、脆弱性利用などの不正な動作ではない。Adobe Acrobat では埋め込まれている JavaScript の内容を表示/編集することが可能である。

マルウェア 1 には「11」および「22」の二つの JavaScript が存在している。これらを Adobe Acrobat 9.1 によって表示した様子を以下に示す。スクリプト「22」が脆弱性を利用するための JavaScript であり、スクリプト「11」は中身の無い（スクリプトコードが存在しない）。



埋め込まれている JavaScript は、Acrobat API で使用される Doc.Media オブジェクトの newPlayer メソッドに起因する「Adobe Reader および Acrobat における解放済みメモリを使用する脆弱性」(JVND-2009-002451(CVE-2009-4324)) を利用して任意のシェルコードを実行するための Exploit コードになっている。Doc.Media オブジェクトの newPlayer メソッドは、新しい MediaPlayer オブジェクトを作成するために使用されるメソッドである。

参考 : JavaScript for Acrobat API Reference

[http://livedocs.adobe.com/acrobat\\_sdk/9/Acrobat9\\_HTMLHelp/wwhelp/wwhimpl/js/html/wwhelp.htm?href=JS\\_API\\_AcroJS.88.1.html#1555776&accessible=true](http://livedocs.adobe.com/acrobat_sdk/9/Acrobat9_HTMLHelp/wwhelp/wwhimpl/js/html/wwhelp.htm?href=JS_API_AcroJS.88.1.html#1555776&accessible=true))

JavaScript が実行されると、実行中の Adobe Reader/Acrobat のバージョンが 8 以上であるかを調べ、バージョンが 8 未満の場合は何も行わずに処理を終了する。バージョンが 8 以上の場合は、難読化されたシェルコードを元に戻した後、JavaScript 内で確保した 1500 のメモリ領域 (ヒープメモリ) にシェルコードをコピーする。このように大量のヒープメモリにシェルコードをコピーすることでシェルコードの実行安定性を高める手法は、一般に「ヒープスプレー」と呼ばれている。このメモリ領域に先のヒープスプレーによってシェルコードが存在していることで、シェルコードが実行されることになる。何らかの理由でこのメモリ領域にシェルコードが存在していない場合には、実行中プログラムは例外エラーを引き起こして異常終了する可能性が高い。なお、どのメモリ領域が使用されるかは OS に依存するが、1500 という大量のメモリ領域にコピーされることでこのメモリ領域にはほぼ確実にシェルコードが配置されると考えられる。

### 3.2.3.2. シェルコード

シェルコードは、細工された PDF ファイルであるマルウェア 1 が脆弱性の利用に成功した場合に実行される実行コードである。このシェルコードは全体が二つの段階に分かれている。一つ目の段階では、暗号化されている二つ目のシェルコードの復号と実行のみを行い、ファイル作成や実行などの処理はすべて二つ目の段階に含まれている。このように二つの段階に分かれているのは、脆弱性を利用してシェルコードを実行する JavaScript コード内のシェルコードのデータを取り出しただけでは、シェルコードの実質的な処理が分からないようにするためであると考えられる。

### 3.2.3.3. マルウェア 1-1 (EXE ファイル)

マルウェア 1-1 は、細工された PDF ファイルであるマルウェア 1 が脆弱性の利用に成功した場合に作成および実行するファイルであり、マルウェア 1-2 を作成しサービスとして登録するために一時的に使用される。

マルウェア 1-1 が行う動作を、処理順序に沿って以下に記載する。

- 1) UPX 形式によって圧縮されているコードを展開し、展開後のコードの実行を開始する。

2) CreateServiceA API を使用して次の名前のサービスを作成する。

サービス名 : 「NTRIP」  
バイナリファイル名 : %SystemRoot%\System32\svchost.exe -k netsvcs  
サービス開始時期 : 「SERVICE\_AUTO\_START」  
サービスタイプ : 「SERVICE\_WIN32\_SHARE\_PROCESS」

3) 2 で作成したサービス内に次の名前のキーを作成と値を作成する。

```
キー :  
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTRIP\Parameters  
  
値 :  
ServiceDll = "<システムフォルダ>\<マルウェア 1-2>"
```

4) 次のレジストリ値に自身が作成したサービスを追加する。これによって Windows 起動時に Svchost.dll によってマルウェア 1-2 が実行されるようになる。

```
キー :  
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SvcHost\  
  
値 :  
netsvcs = "<既存の設定値>NTRIP"
```

5) 自身に埋め込まれているデータを利用して次のファイルを作成する。

ファイル名 : <システムフォルダ>\<マルウェア 1-2>

6) SetFileTime API を使用して 5 で作成したファイルに次のファイルと同じ作成日時に変更する。これは、作成したファイルがファイルの作成日時によって容易に検出されることを防ぐためであると考えられる。

ファイル名 : <システムフォルダ>\<Kernel\*.dll (Windows の既定の状態では Kernel32.dll が該当)

7) 環境変数「COMSPEC」から取得したコマンドプロンプトのシェルプログラムに対するファイルパスを利用し、CreateProcessA API を使用して次のプロセスを作成する。また、作成したプロセスによるファイル削除を確実にこなうため、SetPriorityClass API および SetThreadPriority API を使用して自身のプロセスには「REALTIME\_PRIORITY\_CLASS」および「THREAD\_PRIORITY\_TIME\_CRITICAL」を、作成したプロセスには「IDLE\_PRIORITY\_CLASS」および「THREAD\_PRIORITY\_IDLE」を設定する。これによって自身のプロセスの実行が終了した後、自身のファイルを削除するプロセスが実行

される。

作成するプロセスのコマンドライン： <シェルプログラム> /c del <自身のファイル> > nul 例：  
C:\WINDOWS\system32\cmd.exe /c del C:\DOCUMENTS~1\ADMINI~1\Temp\マルウェア 1 - 1  
> nul

### 3.2.3.4. マルウェア 1 - 2 (DLL ファイル)

マルウェア 1 - 2 はマルウェア 1 - 1 が作成し、Windows 起動時に自動実行されるようサービスとして登録するファイルであり、攻撃者のバックドアとして利用される。

マルウェア 1 - 2 には以下の API がエクスポートされている。

ServiceMain：バックドア活動を行う。

マルウェア 1 - 2 のエクスポート API：



エクスポート API が行う動作を、処理順序に沿って以下に記載する。

### 3.2.3.4.1. ServiceMain

1) プログラム内で暗号化されている以下の文字列を排他的論理和 (XOR) によって復号する。

暗号化文字列	復号キー	復号後文字列
psst	0x41	1225
(&#2#,'1!-&'1/;\$5 71	0x42	japanesecode.myfw.us
rpvqq	0x43	13522
".4	0x44	fcjp

2) connect API を使用して C&C サーバへ TCP プロトコルで接続を試みる。

接続先サーバ： japanesecode.myfw.us

接続先ポート： 13522

なお、接続先サーバの FQDN の IP アドレスおよび IP アドレス割当国情報は以下の通りであった (2010 年 3 月 24 日時点で確認)。

ホスト名 : japanesecode.myfw.us  
 IP アドレス : 140.113.87.112  
 IP アドレス割当国 : 台湾 (tw)  
 ドメイン登録日 : Mon Aug 24 20:05:47 GMT 2009  
 ドメイン登録期限 : Mon Aug 23 23:59:59 GMT 2010  
 最終更新日 : Mon Aug 24 20:17:33 GMT 2009

GeoIP 情報 (情報元 : <http://maxmind.com>) :

Hostname	Country Code	Country Name	Region	Region Name	City	Postal Code	Latitude	Longitude	ISP	Organization	Metro Code	Area Code
140.113.87.112	TW	Taiwan	4	T'ai-wan	Hsinchu		24.8047	120.9714	National Chiao Tung	National Chiao Tung		

3) 接続に失敗すると、Sleep API を使用して 3 秒間待機した後に接続処理を繰り返す。

接続に成功すると、C&C サーバから以下のコマンドを受け取り実行する可能性がある。

コマンド	内容
upload	C&C サーバからファイルをシステムフォルダ内にダウンロードして実行する。
exit	現在の接続を終了した後、再接続する。
Uninstall	接続を終了し自身の処理を終了する。

### 3.2.3.5. 通信内容詳細

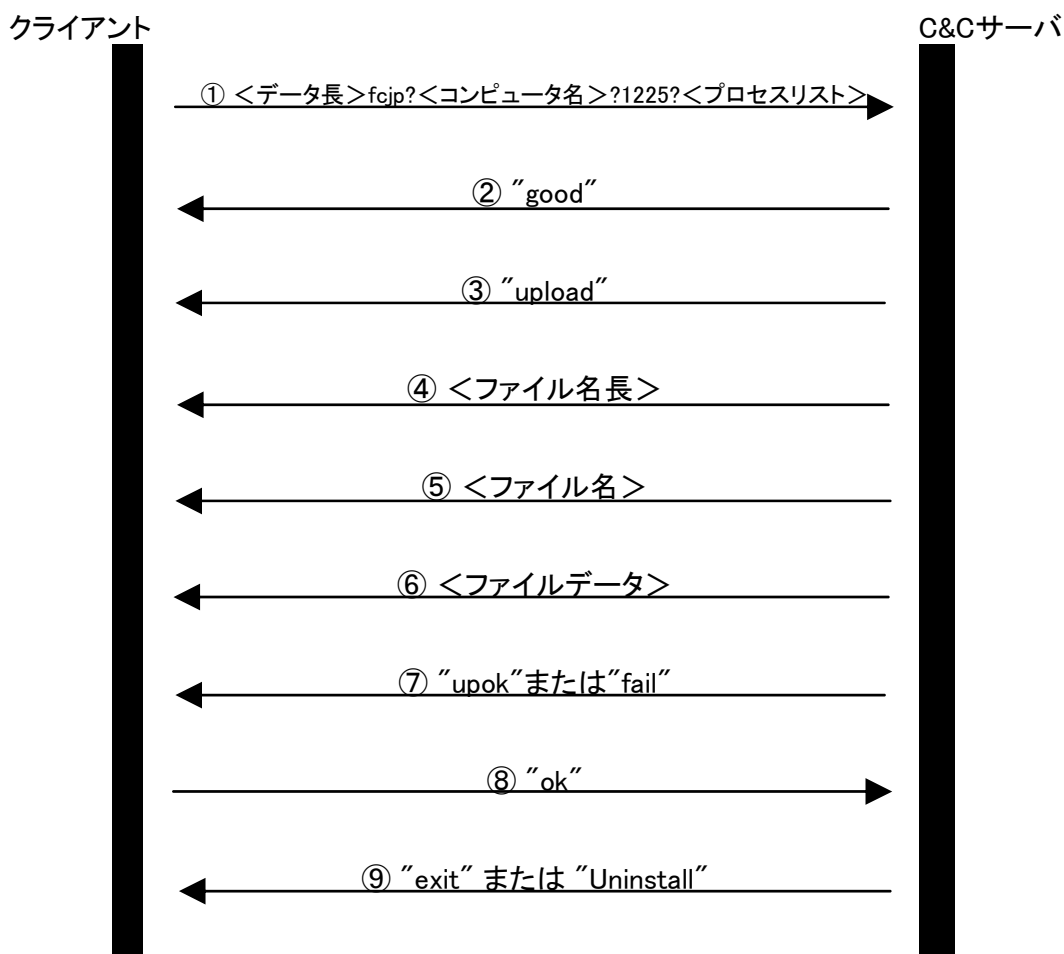
マルウェア 1-2 はリモートの C&C サーバに接続し、受信したコマンドに対応した処理を行なう機能を有している。本解析時点（2010 年 3 月 24 日）では接続先の C&C サーバへの接続に失敗する（名前解決は行えるが TCP 接続が確立しない）ため、C&C サーバから実際にどのようなコマンドが送られてくるかは不明であるが、逆アセンブラを使用した静的コード解析、およびデバッガと動作検証用に用意したダミーサーバを使用した動的コード解析の結果から、マルウェア 1-2 が C&C サーバとの通信のために送受信するデータの内容を調査した。その結果判明した通信用コマンドと通信シーケンスを以下に示す。

#### 通信用コマンド：

シーケンス	向き	内容	意味
①	クライアント→C&C サーバ	<データ長>fcjp?<コンピュータ名>? <固定値"1225">?<プロセスリスト>	C&C サーバへの接続開始
②	C&C サーバ→クライアント	"good"	接続成功
③	C&C サーバ→クライアント	"upload"	クライアントへのファイルのアップロード
④	C&C サーバ→クライアント	<ファイル名長>	アップロードするファイル名の長さ(4バイト)
⑤	C&C サーバ→クライアント	<ファイル名>	アップロードするファイル名
⑥	C&C サーバ→クライアント	<ファイルデータ>	アップロードするファイルのデータ
⑦	C&C サーバ→クライアント	"upok"または"fail"	C&C サーバ側アップロード成功または失敗 失敗時はアップロードしたファイルを削除
⑧	クライアント→C&C サーバ	"ok"	クライアント側アップロード成功(ファイル実行)
⑨	C&C サーバ→クライアント	"exit"または"Uninstall"	接続終了(再接続)またはクライアント処理終了

#### 通信シーケンス：

※一つのコマンドの処理が終了した場合には、次のコマンドの受信を待ち受ける。通信途中で通信に失敗した場合は、3 秒間待機した後に接続処理を繰り返す。



### 3.2.3.6. 本攻撃における脆弱性利用の環境依存性

攻撃メールに添付されていたファイル「マルウェア1」による脆弱性の利用が成功するには、このファイルを開くコンピュータ環境において以下の条件がすべて整っている必要がある。

- ・ 脆弱性の存在する Adobe Reader/Acrobat がインストールされている
- ・ Adobe Reader/Acrobat の設定によって JavaScript 自体もしくは Doc.media.newPlayer メソッドの使用が制限されていない
- ・ Windows OS が利用されている
- ・ Windows OS においてデータ実行防止機能（DEP）が利用されていない

#### 1) 脆弱性の存在する Adobe Reader/Acrobat がインストールされている

マルウェア1は、Adobe Acrobat/Reader の「Multimedia.api」内の Doc.media オブジェクトの newPlayer メソッドが持つ脆弱性を利用してシェルコードを実行する。したがって、この脆弱性を持つ Adobe Acrobat/Reader がインストールされていない場合には脆弱性の利用に失敗する。

#### 2) Adobe Reader/Acrobat の設定によって JavaScript 自体および Doc.media.newPlayer メソッドの使用

が制限されていない

マルウェア 1 は、Doc.media オブジェクトの newPlayer メソッドは JavaScript から利用可能であるため、JavaScript の実行を必要とする。したがって、Adobe Reader/Acrobat の設定によって JavaScript 自体および Doc.media.newPlayer メソッドの使用が制限されている場合には実行に失敗する。

### 3) Windows OS が利用されている

脆弱性の利用に成功した場合に実行されるシェルコードでは、Windows API を使用している。このため、Windows OS 環境以外では実行に失敗する。

### 4) Windows OS においてデータ実行防止機能 (DEP) が利用されていない

マルウェア 1 がシェルコードを展開するヒープメモリ上のデータは実行可能なメモリ領域として設定されていない。したがって、DEP が有効な場合にはシェルコードが実行される際に実行が防止される。

なお、Adobe Reader/Acrobat 9.2 には利用される脆弱性が存在しているが、SetProcessDEPPolicy API によるプロセスレベルで DEP が利用されているため、この API が利用可能な Windows XP SP3 および Windows Vista SP1 以降の環境では DEP 機能が明示的に有効にされていない場合でもシェルコードの実行に失敗し異常終了する。JavaScript および Doc.media.newPlayer メソッドの使用が制限されておらず、かつ DEP がユーザによって有効にされていない場合のシェルコード実行可否について、Windows OS 種類ごとにまとめた表を以下に示す (○：実行不可、×：実行可能)。

OS 種類	Adobe Reader/Acrobat 9.1	Adobe Reader/Acrobat 9.2	Adobe Reader/Acrobat 9.3
Windows XP SP2	×	×	○
Windows XP SP3	×	○	○
Windows Vista	×	×	○
Windows Vista SP1	×	○	○
Windows 7	×	○	○

## 4. 「Adobe Reader/Acrobatの脆弱性を利用した攻撃」で利用される脆弱性の分析

### 4.1. 脆弱性の概要

攻撃メールに添付されていたファイルは、Adobe Acrobat/Reader 9.2.0 以前のバージョンのファイル「Multimedia.api」内の Doc.media オブジェクトの newPlayer メソッドが持つ脆弱性を利用してシェルコードを実行する。この脆弱性は、現在では既知の脆弱性であり、次の脆弱性識別番号で確認されている。

JVNDB-2009-002451 (CVE-2009-4324)

「Adobe Reader および Acrobat における解放済みメモリを使用する脆弱性」

<http://jvndb.jvn.jp/ja/contents/2009/JVNDB-2009-002451.html>

この脆弱性の概要、影響を受けるソフトウェアとバージョンおよび解決方法については、アプリケーションの開発元であるアドビシステムズ社から脆弱性識別番号「APSA09-07」として 2009 年 12 月 15 日に公開されている。

<http://www.adobe.com/jp/support/security/advisories/apsa09-07.html>

### 4.2. 攻撃手法

この脆弱性は、利用者が脆弱性を持つアプリケーション（Adobe Reader/Acrobat）で細工された PDF ファイルを開くことによって利用される。攻撃者が利用者に細工した PDF ファイルを開かせる方法としては、以下の方法などが考えられる。

- 1) メールに添付して利用者へ送りつける
- 2) メールに PDF ファイルを指す URL を書いて利用者へ送りつける
- 3) SEO によって利用者に PDF ファイルを指す URL をクリックさせる
- 4) 元々公開されている正規 PDF ファイルを不正 PDF ファイルに置き換えておき利用者が開くのを待つ
- 5) 正規ウェブページを改ざんしてウェブページ閲覧時に不正 PDF ファイルを自動的に開かせる

### 4.3. 攻撃状況

この脆弱性は、ウェブサイト上で公開されているセキュリティ研究者からの情報によれば、少なくとも 2009 年 11 月 30 日から行われていることが報告されている。

この脆弱性のセキュリティパッチは、2010 年 1 月 13 日にアドビシステムズ社から APSB10-02 として公開されている。

<http://www.adobe.com/jp/support/security/bulletins/apsb10-02.html>

アドビシステムズ社からセキュリティパッチが公開される以前から攻撃が行なわれていたことから、この攻撃はいわゆるゼロデイ攻撃であったと言える。また、セキュリティパッチ公開前の段階で、実行可能な Exploit コードが複数のウェブサイトで開催されており、悪用可能な状態であった。

セキュリティ研究者からは、様々な言語の様々な内容の攻撃メールが存在していることが公開されている。また、日本での被害が話題になったいわゆるガンブラー（通称、Gumblar.8080）でも、セキュリティパッチ未公開（ゼロデイ）の時点でこの脆弱性が利用されていることが確認されている。

参考：<http://www.jpCERT.or.jp/at/2010/at100001.txt>

## 5. 「Microsoft Wordの脆弱性を利用した攻撃」の分析

本章では、IPA 設置の「不審メール 110 番」に相談のあった「Microsoft Word の脆弱性を利用した攻撃」における攻撃メールおよびマルウェアを分析し、以下の点を明らかにする。

- 攻撃メールの特徴
- 利用している脆弱性の特徴
- マルウェアの動作
- 通信内容

### 5.1. 攻撃メールの分析

本節では、「Microsoft Word の脆弱性を利用した攻撃」に用いられた攻撃メールの内容および利用された手口を分析する。

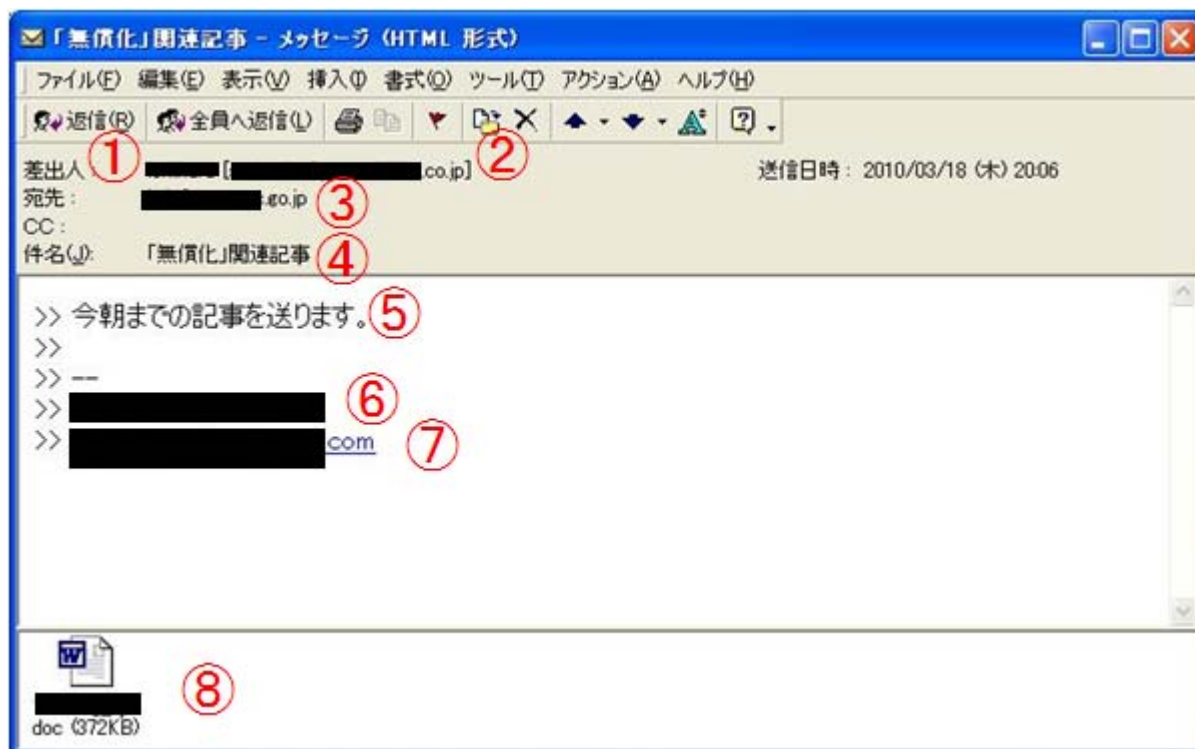
#### 5.1.1. メール内容（本文・ヘッダ・添付ファイル）

IPA 「不審メール 110 番」に相談のあったメールの本文およびヘッダの内容を以下に示す。

不審メールの構成：

項目	内容
差出人	・実在の組織のドメインを使用したメールアドレス ・実在の人物によって実際に使用されているメールアドレスかどうかは不明
宛先	・受信者が業務のため加入しているメーリングリスト
送信時間	・20010/03/18(木) 20:06
件名	・「「無償化」関連記事」
本文	・「今朝までの記事を送ります。」 ・実在の組織をかたった署名
メール形式	・HTML およびテキスト形式 (multipart/alternative)
添付ファイル	・マルウェア2が添付

メールを表示した様子(Outlook 2000 を使用):



メールの特徴は以下の通りです。

- ① 受信者に心当たりのない個人名
- ② フリーメールのアドレス
- ③ 受信者が業務用で加入しているメーリングリスト
- ④ 当時話題になっていたニュースを連想させる件名
- ⑤ 受信者に添付ファイルを開くよう仕向ける文章
- ⑥ 実在の団体名
- ⑦ フリーメールのアドレス
- ⑧ 本文と関連がありそうな名前の DOC ファイル。実態はマルウェア。

### 5.1.2. 実際のメール受信者の対応

このメールの受信者から、メール受信時の状況について聞き取り調査を行った。聞き取り調査によって判明した受信時の状況を以下に示す。

#### (1) メールを見ての感想

- 最初はただのスパムかと思った。
- 件名と本文は当時ニュースで盛んに取りざたされた内容だったが、このメーリングリストに送られてくるには不自然だった。

- メールに気付いた職員が口々に「変なメールが来ている」と言い出した。
- 差出人として表示されている名前とメールアドレスの名前が異なっており不自然だった。

(2) メール受信後の対応

- 不審メールとして IPA へ報告した。
- 該当メールを読む可能性がある部署へ、添付ファイルを開くことなくメールを削除するように連絡した。

### 5.1.3. 攻撃メールでの利用手口の分析

この標的型攻撃メールに対して、本報告書 3.1.3.章で取り上げた一つ目の標的型攻撃メールと同じ点から分析を加える。

#### (1)メールの件名と本文の自然さ

この標的型攻撃メールでは受信者が日本人であり差出人も日本人の名前になっている。また、メール本文と添付ファイルの件名、および添付ファイルを開いた後に作成されるダミーの正常な文章ファイルの内容がいずれも日本語で書かれている。さらに文章ファイルの内容がメール送信日時頃に日本国内において新聞などのニュースで話題になっていた内容であることから、明らかに日本に詳しい者による日本を意識した攻撃であると考えられる。しかし、このメール受信者は、自身に送られてくるメールの内容としては不自然であるとして、不審なメールであることにすぐに気付いている。標的型メール攻撃では、攻撃者は受信者を選定できることから、受信者にとって違和感のない内容を吟味してから送信することが可能であるが、今回の標的型攻撃メールでは社会的に話題になった句の内容を利用している。社会的な話題を利用した攻撃は、不特定多数に送信するスパムメールで多く用いられる手法であることから、今回の攻撃型メールは特定の標的のみに特化した攻撃ではなく、多数の宛先に対して同じ内容で攻撃するなど、攻撃先をそれほど吟味せずに行っている可能性が考えられる。

#### (2)メールの差出人と受信者の自然さ

この標的型攻撃メールではメールヘッダ上の差出人の名前が表示上の名前とメールアドレスに書かれている名前が異なっていることから、受信者が容易に不審なメールであることに気付いている。

#### (3)添付ファイルの種類およびファイル名と開きやすさ

この標的型攻撃メールでは、添付ファイルは Microsoft Word のドキュメントファイル（以下、DOC ファイル）である。拡張子は「.doc」で二重拡張子や大量の空白文字の利用など拡張子の偽装は行われていない。DOC ファイルはアプリケーションである Microsoft Word がインストールされていなければ利用できないが、Microsoft Word は一般に多く利用されているソフトウェアであることから、受信者が Microsoft Word を利用している可能性は高いと考えられる。

#### (4)添付ファイルを開いた後の目的動作の達成しやすさ

今回の添付ファイルでは MS06-027 の脆弱性が利用されている。この脆弱性は今回の攻撃メールの受信日と比較して約 4 年前に存在が確認された脆弱性であり、現時点ではセキュリティパッチの適用やこの脆弱性が存在しない新しいバージョンの Microsoft Office が利用されている可能性も高いことから攻撃成功率は低いと言える。Microsoft Word の脆弱性は MS06-027 が公開された後も複数の脆弱性が発見されており、攻撃者がなぜこのような古い脆弱性を使ったのかは不明だが、マルウェア作成ツールなどにより攻撃用のファイルが作成しやすい状況であった可能性が推測できる。また、今回の添付ファイルを開いた場合に行われる動作ではその通信に一般的には使用されないポート番号（13281）が用いられることから、ファイアウォールなどで比較的ブロックされやすく攻撃者の攻撃目的が達成される可能性は低くなると考えられる。

## (5)実行中マルウェアの存在の気付きやすさ

この標的型攻撃メールでは、ダミーの正常なドキュメントファイルが作成されて開かれる。またその内容はメール本文の内容と整合した内容であるため、利用者はメールの添付ファイルを開いただけではすぐには不正ファイルを開いたとは気付かない可能性がある。また、バックドアである DLL ファイルはネットワークサービスを動作させるために汎用的に利用されるプログラムである「svchost.exe」を利用して動作するため、不正なプロセスの存在を見つけるというアプローチによるマルウェアの存在調査手法では検出されにくい。さらに、バックドアである DLL ファイルをインストールする EXE ファイルは DLL ファイルのインストール後に DLL ファイルのタイムスタンプを変更した上で自分自身 (EXE ファイル) を削除するため、後からバックドアである DLL ファイルの存在に気付いても、その DLL ファイルが何によっていつ作成されたものなのかを調査することが困難である。

## 5.2.マルウェアの分析

本節では、「Microsoft Word の脆弱性を利用した攻撃」に用いられたマルウェアの動作を分析する。

### 5.2.1. 解析対象検体

解析対象検体の概要を以下に示す。

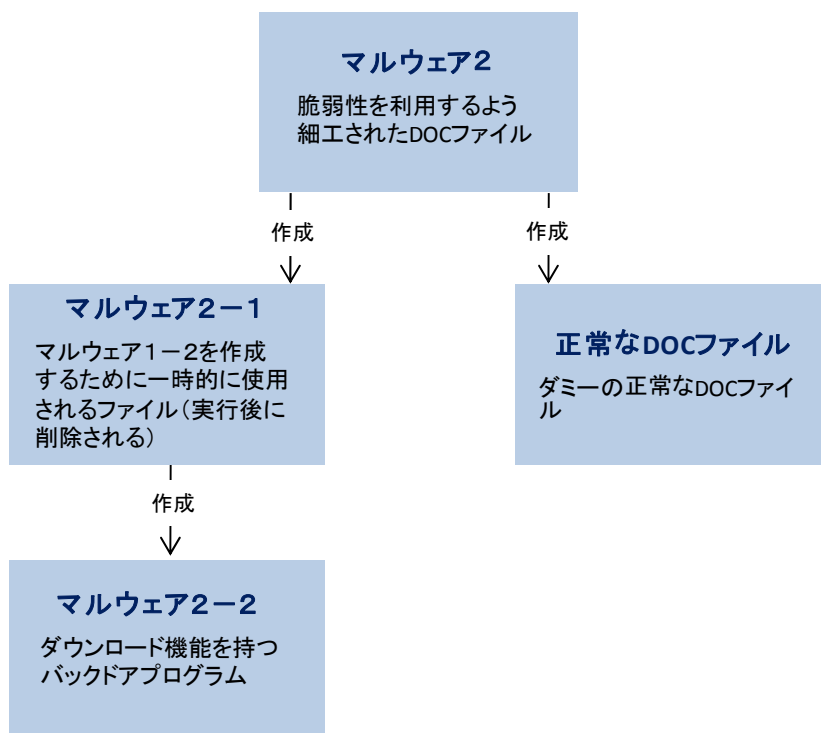
マルウェア	マルウェア2
概要	<ul style="list-style-type: none"><li>攻撃メールに添付されていたファイル</li><li>MS06-027 の脆弱性を利用するよう細工された DOC ファイル</li><li>脆弱性の利用に成功すると、マルウェア2-1とダミーの正常な DOC ファイルを作成</li></ul>

解析対象検体の実行によって作成されるファイルの概要を以下に示す。

マルウェア	マルウェア2-1 (EXE ファイル)
概要	<ul style="list-style-type: none"><li>埋め込まれたファイルデータを使用してマルウェア2-2を作成</li><li>作成したマルウェア2-2をサービスとして登録</li></ul>
ダミーファイル	正常な DOC ファイル
概要	<ul style="list-style-type: none"><li>ダミーの正常な DOC ファイル</li><li>メールに添付されていた DOC ファイルと同名 (ファイル名はオリジナルファイル名に依存して可変)</li></ul>
マルウェア	マルウェア2-2 (DLL ファイル)
特記事項	<ul style="list-style-type: none"><li>バックドアプログラム</li><li>Windows のサービスとして登録され動作する</li></ul>

各ファイルの関連を以下に示す。

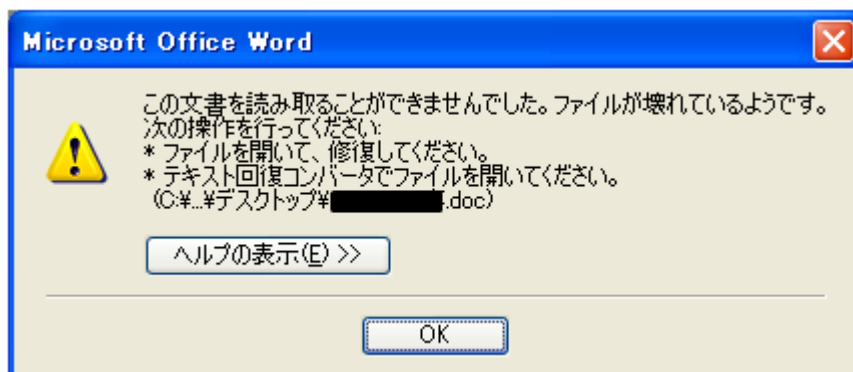
ファイル関連図：



### 5.2.2. マルウェア動作概要

攻撃メールに添付されていたファイルは、Microsoft Word の脆弱性を利用し、Windows プラットフォームで任意のコードを実行するよう細工された DOC ファイルである。この脆弱性および影響を受ける OS 種類の詳細は、6 章で述べる。Windows プラットフォーム上でこの不正 DOC ファイルが開かれると、その不正 DOC ファイルを開いたアプリケーション (Microsoft Word) に狙われた脆弱性が存在する場合、その脆弱性が利用されてシェルコードが実行される。脆弱性が存在しない場合にはシェルコードは実行されない。この脆弱性が修正されているバージョンの Microsoft Word 2003 で開いた場合には、以下のようなエラーメッセージが表示される。

この脆弱性用のセキュリティパッチが適用された Microsoft Word 2003 で開いた場合の画面例：



脆弱性の利用に成功しシェルコードが実行されると、シェルコードでは不正 DOC ファイルに埋め込まれているデータを利用して、別の不正ファイル「マルウェア 2-1」を Windows のテンポラリフォルダ（Windows の一時ファイル保存場所）内に作成して実行する。さらに、正常な DOC ファイルも Windows のテンポラリフォルダ内に作成し、winword.exe（Microsoft Word の実行ファイル名）を利用して作成した正常な DOC ファイルを開く。

ここで作成される DOC ファイルのファイル名は可変であり、最初に開かれた不正な DOC ファイルのファイル名と同じ名前で作成される。作成される DOC ファイルには、正常な DOC ファイルのコンテンツが含まれている。また、最初に開いた不正 DOC ファイルは作成した正常な DOC ファイルの内容で上書きされ、無害化される。このように最初に開かれた不正な DOC ファイルを正常な DOC ファイルの内容で上書きするのは、感染経路の調査を困難にするためであると考えられる。

マルウェア 2-1 は実行されると自身のファイル内に埋め込まれているデータを利用して別の不正ファイルであるマルウェア 2-2 を作成し、サービスとして登録する。また、処理の最後で自身のファイルを削除する。自身のファイルを削除するのは、感染経路の調査を困難にするためであると考えられる。

マルウェア 2-2 は実行されると外部のサーバへ通信してコマンドの受信を待ち受ける。この動作はいわゆるバックドアであるが、バックドアとしての機能はファイルのダウンロードと実行のみである（バックドア機能の詳細は 5.3. 章で述べる）。攻撃者はこのダウンロード機能を利用して秘密情報の入手など攻撃者の攻撃動機を満たすための別のマルウェアを送り込むと考えられるが、本解析時点では接続先のサーバ（C&C サーバ）への接続は既にできない状態であったため、攻撃者の本来の意図は不明である。このように作成されるファイルだけでは本来の意図を見えないようにすること、および接続可能な時間などを制限することは、攻撃の全貌が明らかになることを防ぐために攻撃者が意図的に行っている可能性が考えられる。

これらの実行コードの特徴としては、3 章で解析したマルウェアと比較してシェルコードがさらに多段になり複雑化していることが上げられる。シェルコードの多段化は、動的解析の妨げにはほとんどならないが、静的解析をより困難にする効果があり、攻撃者が解析を困難にするために工夫しようとしている様子が窺える。一方、マルウェア 2-1 とマルウェア 2-2 は 3 章で解析したマルウェアと動作が酷似しており、それらと同じく解析を困難にする技術はあまり使われていないが、ファンクションの構造は変化しており、バイナリレベルではなくソースコードレベルでプログラムに変更が加えられていると考えられる。

### 5.2.3. マルウェア動作詳細

#### 5.2.3.1. 脆弱性の利用 (Exploit)

マルウェア 2 にはスマートタグオブジェクト内のポインタに不正なポインタが指定されている。

脆弱性が存在するバージョンの Microsoft Word でマルウェア 2 を開くと、このポインタが利用されてこのポインタで指定されたアドレスに位置するテーブルが利用されて関数呼び出しが行われる。これにより、テ

ープル上で指定されたアドレス位置にシェルコードが置かれている場合には、そのシェルコードが実行される。

WINWORD.EXE 内のこのオブジェクトの処理における関数呼び出しにおいて、ファイル内に予め用意したアドレスが呼び出される。なお、これらの処理においてデータ実行防止機能 (DEP) を迂回するなど工夫は特に行われていない。

### 5.2.3.2. シェルコード

シェルコードは、細工された DOC ファイルであるマルウェア 2 が脆弱性の利用に成功した場合に実行される実行コードである。このシェルコードは全体が五つの段階に分かれている。

第一段階では、細工された DOC ファイルであるマルウェア 2 が読み込まれているメモリ中から、メインのシェルコードと呼べる第二段階以降のシェルコードのデータが読み込まれている部分を探し出してその実行を開始する。

第二段階では、第三段階以降のシェルコード実行の準備として、細工された DOC ファイルであるマルウェア 2 を操作できるようにこのファイルのファイルハンドルを取得してメモリ上にマッピングする。また、暗号化されている第三段階のシェルコードと埋め込まれている実行ファイルのデータを復号する。

第三段階では、暗号化されている第四段階のシェルコードの復号とその実行の開始のみを行う。

第四段階では、埋め込まれている暗号化された実行ファイルを復号して、テンポラリフォルダ内にマルウェア 2-1 として作成して実行する。また、埋め込まれている暗号化された正常な DOC ファイルを復号して、テンポラリフォルダ内に最初に開かれた DOC ファイルと同じファイル名で保存し、winword.exe (Microsoft Word の実行ファイル名) で開く。さらに、暗号化されている第五段階のシェルコードを復号してその実行を開始する。

第五段階では、作成した正常な DOC ファイルのデータで最初に開かれた DOC ファイルの先頭部分を上書きする。これによって最初に開かれた細工された DOC ファイルは無害化される。

このようにシェルコードが複数の段階に分かれているのは、細工された DOC ファイル内のシェルコードのデータを取り出し静的解析を行っただけでは、シェルコードの実質的な処理が分からないようにするためであると考えられる。

### 5.2.3.3. マルウェア 2-1 (EXE ファイル)

マルウェア 2-1 は、細工された DOS ファイルであるマルウェア 2 が脆弱性の利用に成功した場合に作成および実行するファイルであり、マルウェア 2-2 を作成しサービスとして登録するために一時的に使用される。

マルウェア 2-2 が行う動作を、処理順序に沿って以下に記載する。

- 1) GetSystemDirectoryA API を使用して取得して Windows システムフォルダ内に、自身のファイルのソース内のデータを使用して、次の名前のファイルを作成する。

ファイル名 : マルウェア 2-2 のファイル名

リソースタイプ : 256 (0x100)  
 リソース名 : 100 (0x64)  
 言語 : 0804 (中国語)  
 リソースサイズ : 38,400 (0x9600) バイト

リソース内のデータ :

アドレス	タイプ	ネーム	言語	サイズ (decimal)	インフォメーション
00404060	100	64	0804 中国語 (中国)	00009600 (38400.)	

Resource at 00404060 - SVOHOS~1.rsrc 00404060..0040D65F					
00404060	4D 5A 90 00	03 00 00 00	04 00 00 00	FF FF 00 00	MZ・L...J... . . .
00404070	B8 00 00 00	00 00 00 00	40 00 00 00	00 00 00 00	ク.....@.....
00404080	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
00404090	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....E8 00 00 00
004040A0	0E 1F BA 0E	00 B4 09 CD	21 B8 01 4C	CD 21 54 68	0000.I.^!kL^!Th
004040B0	69 73 20 70	72 6F 67 72	61 6D 20 63	61 6E 6E 6F	is program canno
004040C0	74 20 62 65	20 72 75 6E	20 69 6E 20	44 4F 53 20	t be run in DOS
004040D0	6D 6F 64 65	2E 0D 0D 0A	24 00 00 00	00 00 00 00	mode....\$. . . . .
004040E0	AD A7 63 BB	E9 C6 0D E8	E9 C6 0D E8	E9 C6 0D E8	урсщ.щ.щ.щ.щ.
004040F0	92 DA 01 E8	E8 C6 0D E8	6A DA 03 E8	FC C6 0D E8	婚r幸.鏢i鑿. . .
00404100	01 D9 07 E8	B0 C6 0D E8	8B D9 1E E8	E0 C6 0D E8	ル・佳.閨ll鞆. . .
00404110	E9 C6 0C E8	86 C6 0D E8	01 D9 06 E8	E2 C6 0D E8	鯛.閨.・ル-鞆. . .
00404120	01 D9 09 E8	E8 C6 0D E8	52 69 63 68	E9 C6 0D E8	ル.幸.鏢 ich鯛. . .
00404130	00 00 00 00	00 00 00 00	00 00 00 00	00 00 00 00	.....
00404140	00 00 00 00	00 00 00 00	50 45 00 00	4C 01 04 00	.....PE..Lr.
00404150	ED FE 96 4B	00 00 00 00	00 00 00 00	00 00 00 00	・訪.....・! . . .
00404160	0B 01 06 00	00 5C 00 00	00 56 00 00	00 00 00 00	♫r..¥...V.....
00404170	F5 28 00 00	00 10 00 00	00 70 00 00	00 00 00 10	・...+...p...+ . . .
00404180	00 10 00 00	00 02 00 00	04 00 00 00	00 00 00 00	.+...r..J.....
00404190	04 00 00 00	00 00 00 00	00 E0 00 00	00 04 00 00	J..... . . .J.. . . .

- 2) SetFileTime API を使用して 1 で作成したファイルに次のファイルと同じ作成日時に変更する。これは、作成したファイルがファイルの作成日時によって容易に検出されることを防ぐためであると考えられる。  
 ファイル名 : <システムフォルダ>¥Kernel\*.dll (Windows の既定の状態では Kernel32.dll が該当)
- 3) Sleep API を使用して 2.999 秒間待機する。これは、1 で作成したファイルの保存が確実に行われるのを待機するためであると考えられる。
- 4) ShellExecuteA API を使用して次のコマンドを実行する。これによって、1 で作成したファイルマルウェア 2-2 の RundllInstall API が実行され、マルウェア 2-2 が指定したパラメータ名としてサービスに登録される。

```
cmd.exe /c rundll32 ntsapi.dll,RundllInstall IPRIP
```

- 5) Sleep API を使用して 1.999 秒間待機する。これは、1 で作成したファイルの保存が確実に行われるのを待機するためであると考えられる。
- 6) ShellExecuteA API を使用して次のコマンドを実行する。これによって、4 で登録したサービスを開始する。

```
cmd.exe /c sc start IPRIP
```

- 7) Sleep API を使用して 0.999 秒間待機する。これは、6 で行ったサービスの開始が確実に行われるのを待機するためであると考えられる。
- 8) 環境変数「COMSPEC」から取得したコマンドプロンプトのシェルプログラムに対するファイルパスを利用し、CreateProcessA API を使用して次のプロセスを作成する。また、作成したプロセスによるファイル削除を確実にこなうため、SetPriorityClass API および SetThreadPriority API を使用して実行中の自身のプロセスには「REALTIME\_PRIORITY\_CLASS」および「THREAD\_PRIORITY\_TIME\_CRITICAL」を、作成したプロセスには「IDLE\_PRIORITY\_CLASS」および「THREAD\_PRIORITY\_IDLE」を設定する。これによって自身のプロセスの実行が終了した後、自身のファイルを削除するプロセスが実行される。  
作成するプロセスのコマンドライン： <シェルプログラム> /c del <自身のファイル> > nul  
例： C:\WINDOWS\system32\cmd.exe /c del C:\DOCUMENT~1\ADMINI~1\Temp\マルウェア 2-1 > nul
- 9) ExitProcess API を使用して実行中の自身のプロセスの処理を終了する。

#### 5.2.3.4. マルウェア 2-2 (DLL ファイル)

マルウェア 2-2 はマルウェア 2-1 が作成するサービスとして動作するファイルであり、攻撃者のバックドアとして利用される。

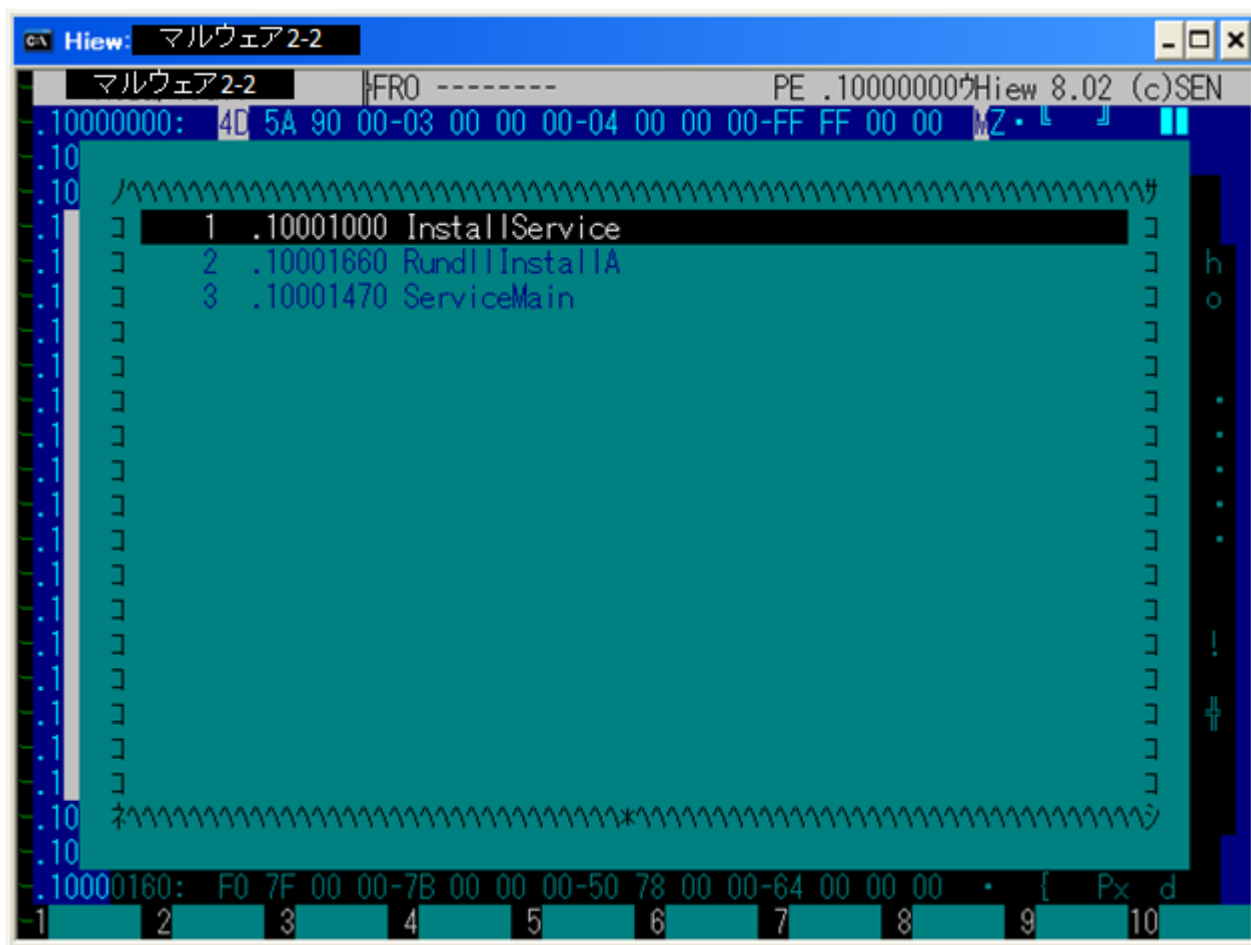
マルウェア 2-2 は以下の API をエクスポートしている。

InstallService： パラメータで指定された名前で自身のファイルをサービスとして登録する。

RundllInstallA： InstallService API を呼び出す。

ServiceMain： バックドア活動を行う。

マルウェア 2-2 のエクスポート API :



各 API が行う動作を、処理順序に沿って以下に記載する。

#### 5.2.3.4.1. RundllInstallA

- 1) 指定されたパラメータを使用して自身の InstallService API の実行を開始する。

#### 5.2.3.4.2. InstallService

- 1) 次のレジストリ値にパラメータで指定された名前が存在するかを確認する。

場所 :

HKEY\_LOCAL\_MACHINE¥SOFTWARE¥Microsoft¥Windows NT¥CurrentVersion¥SvcHost¥

値 :

Netsvcs

2) 1 で存在しなかった場合には、自身の処理を終了する。存在した場合は以下の処理を続行する。

3) CreateServiceA API を使用して次のサービスを作成する。

サービス名 : パラメータで指定された名前  
バイナリファイル名 : %SystemRoot%\System32\svchost.exe -k netsvcs  
サービス開始時期 : 「SERVICE\_AUTO\_START」  
サービスタイプ : 「SERVICE\_WIN32\_SHARE\_PROCESS」

※同じサービス名のサービスが既に存在している場合にはサービスの作成に失敗し、自身の処理を終了する。当該サービス名は既定では存在していない。

4) 次のレジストリキーおよび値を作成する。

```
キー :  
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTRIP\Parameters  
  
値 :  
ServiceDll = "<システムフォルダ>\マルウェア 2 - 2"
```

#### 5.2.3.4.3. ServiceMain

マルウェア 2 - 2 の ServiceMain API は、自身がサービスとして起動された場合に実行される API である。

1) プログラム内で暗号化されている以下の文字列を排他的論理和 (XOR) によって復号する。

暗号化文字列	復号キー	復号後文字列
{.}	1 バイトごとに異なった既定の値	2010
#UH*",4![]-"i*3.CvB#	1 バイトごとに異なった既定の値	japanesecode.myfw.us
x.s}	1 バイトごとに異なった既定の値	13281
/WR;	1 バイトごとに異なった既定の値	fcjp

2) connect API を使用して C&C サーバへ TCP プロトコルで接続を試みる。

接続先サーバ : japanesecode.myfw.us

接続先ポート : 13281

なお、接続先サーバの FQDN の IP アドレスおよび IP アドレス割当国情報は以下の通りであった（2010年4月29日時点で確認）。

ホスト名 : japanesecode.myfw.us  
 IP アドレス : 127.0.0.1  
 IP アドレス割当国 : -  
 ドメイン登録日 : Mon Aug 24 20:05:47 GMT 2009  
 ドメイン登録期限 : Mon Aug 23 23:59:59 GMT 2010  
 最終更新日 : Mon Aug 24 20:17:33 GMT 2009

- 3) 接続に失敗すると、Sleep API を使用して 3 秒間待機した後に接続処理を繰り返す。  
 接続に成功すると、C&C サーバから以下のコマンドを受け取り実行する可能性がある。

コマンド	内容
upload	C&C サーバからファイルをシステムフォルダ内にダウンロードして実行する。
exit	現在の接続を終了した後、再接続する。

#### 5.2.3.5. 通信内容詳細

マルウェア 2-2 はリモートの C&C サーバに接続し、受信したコマンドに対応した処理を行なう機能を有している。本解析時点（2010年4月29日）では接続先の C&C サーバへの接続に失敗する（名前解決で「127.0.0.1」が返される）ため、C&C サーバから実際にどのようなコマンドが送られてくるかは不明であるが、逆アセンブラを使用した静的コード解析、およびデバッガと動作検証用に用意したダミーサーバを使用した動的コード解析の結果から、マルウェア 2-2 が C&C サーバとの通信のために送受信するデータの内容を調査した。その結果判明した通信用コマンドと通信シーケンスを以下に示す。

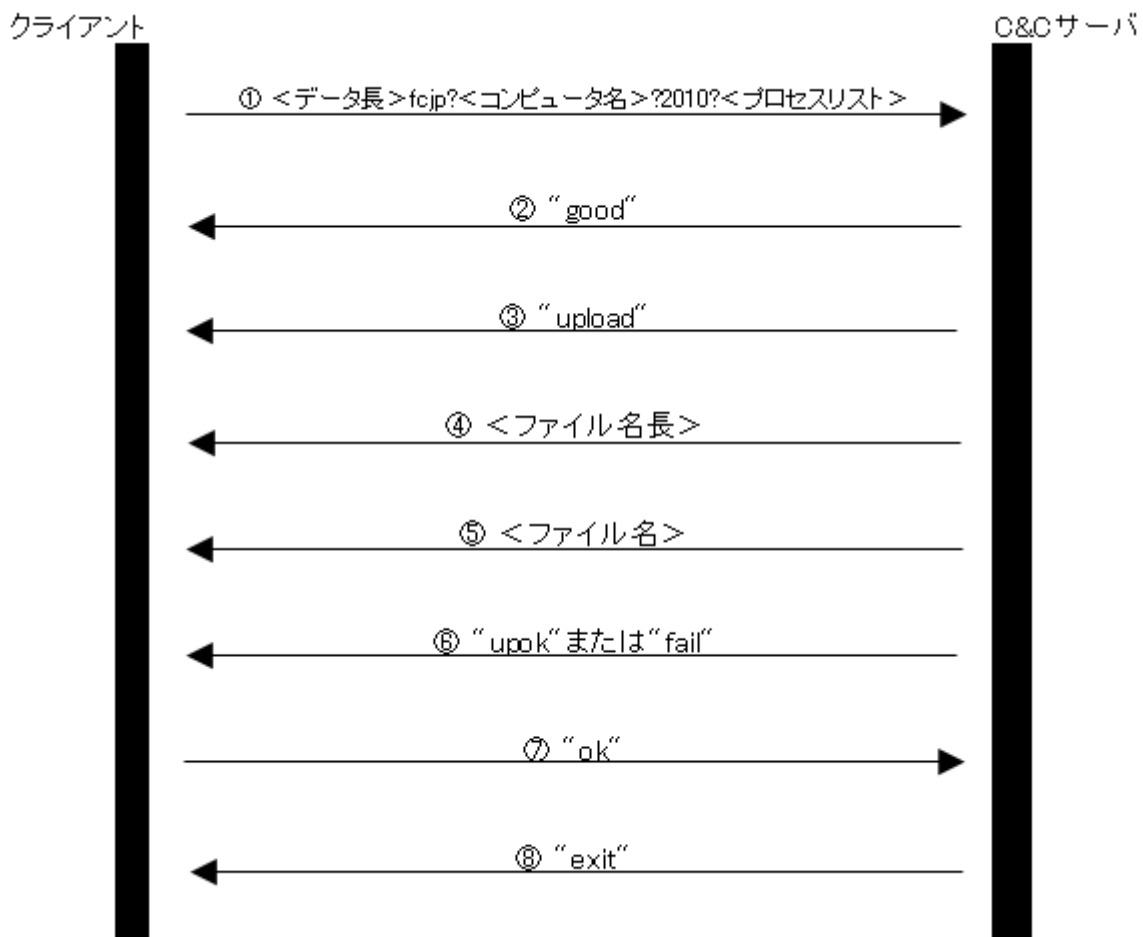
通信用コマンド：

シーケンス	向き	内容	意味
①	クライアント→C&C サーバ	<データ長>fcjp?<コンピュータ名>? <固定値"2010">?<プロセスリスト>	C&C サーバへの接続開始
②	C&C サーバ→クライアント	"good"	接続成功
③	C&C サーバ→クライアント	"upload"	クライアントへのファイルのアップロード
④	C&C サーバ→クライアント	<ファイル名長>	アップロードするファイル名の長さ(4バイト)
⑤	C&C サーバ→クライアント	<ファイル名>	アップロードするファイル名

シーケンス	向き	内容	意味
⑥	C&C サーバ→クライアント	<ファイルデータ>	アップロードするファイルのデータ
⑦	C&C サーバ→クライアント	"upok"または"fail"	C&C サーバ側アップロード成功または失敗 失敗時はアップロードしたファイルを削除
⑧	クライアント→C&C サーバ	"ok"	クライアント側アップロード成功(ファイル実行)
⑨	C&C サーバ→クライアント	"exit"	接続終了(再接続)

#### 通信シーケンス：

※一つのコマンドの処理が終了した場合には、次のコマンドの受信を待ち受ける。通信途中で通信に失敗した場合は、3 秒間待機した後に接続処理を繰り返す。



#### 5.2.3.6. 本攻撃における脆弱性利用の環境依存性

攻撃メールに添付されていたファイルによる脆弱性の利用が成功するには、このファイルを開くコンピュータ環境において以下の条件がすべて整っている必要がある。

- 脆弱性の存在する **Microsoft Word** がインストールされている
- 細工したオブジェクトポインタのアドレスが利用環境と合致する
- **Windows OS** が利用されている
- **Windows OS** においてデータ実行防止機能（**DEP**）が利用されていない

##### 1) 脆弱性の存在する **Microsoft Word** がインストールされている

マルウェア 2 は **Microsoft Word** ドキュメント内の不正なオブジェクトポインタの脆弱性を利用してシェルコードを実行する。したがって、この脆弱性を持つ **Microsoft Word** がインストールされていない場合には脆弱性の利用に失敗する。

##### 2) 細工したオブジェクトポインタのアドレスが利用環境と合致する

脆弱性に利用されるオブジェクトのポインタでは、メモリ中に読み込まれている細工したデータのアドレスを直接指定する必要がある。メモリ中のどのアドレス位置に細工したデータが読み込まれるかを攻撃者が事前に特定することは困難であり、ある特定の環境におけるアドレス値を使用した場合、別の環境においてはアドレス値が異なるため実行に失敗するなど環境依存性が高くなる。今回解析した検体マルウェア 2 においてはファイルデータが読み込まれたスタックメモリ上のアドレスを直接指定しているが、スタックメモリ上のどのアドレスに意図したデータが読み込まれているかは、ドキュメントファイルをダブルクリックしてアプリケーションを起動したか、アプリケーションを起動してからファイルメニューからドキュメントファイルを開いたかなど起動状態によって異なり、今回の検体ではドキュメントファイルをダブルクリックして開いた場合のみ適切なアドレスが参照されシェルコードの実行が行われる。また、ダブルクリックして開いた場合でも解析のためにデバッガ上で読み込んで実行した場合には、使用されるアドレスが異なりシェルコードの実行が行われなかった。環境にあったアドレス値を指定すればよいので各環境で動作するようファイルを書き換えることは用意であるが、それぞれの環境専用のファイルとなる。

##### 3) **Windows OS** が利用されている

脆弱性の利用に成功した場合に実行されるシェルコードでは、**Windows API** を使用している。このため、**Windows OS** 環境以外では実行に失敗する。

##### 4) **Windows OS** においてデータ実行防止機能（**DEP**）が利用されていない

解析対象検体マルウェア 2 ではスタックメモリ上に展開されたシェルコードを実行するため、**DEP** が有効な場合にはシェルコードが実行される際に実行が防止される。

## 6. 「Microsoft Wordの脆弱性を利用した攻撃」で利用される脆弱性の分析

### 6.1. 脆弱性の概要

攻撃メールに添付されていたファイル「マルウェア 2」は、Microsoft Word ドキュメント内の不正なオブジェクトポインタの脆弱性を利用してシェルコードを実行する。この脆弱性は現在では既知の脆弱性であり、次の脆弱性識別番号で確認されている。

JVNDB-2006-000296 (CVE-2006-2492)

「Microsoft Word における不正なオブジェクトポインタによるメモリ破壊の脆弱性」

<http://jvndb.jvn.jp/ja/contents/2006/JVNDB-2006-000296.html>

この脆弱性の概要、影響を受けるソフトウェアとバージョンおよび解決方法については、アプリケーションの開発元であるマイクロソフト社からセキュリティ情報番号「MS06-027」として 2006 年 6 月 14 日に公開されている。

<http://www.microsoft.com/japan/technet/security/bulletin/MS06-027.msp>

### 6.2. 攻撃手法

この脆弱性は、利用者が脆弱性を持つアプリケーション（Microsoft Word）で細工された DOC ファイルを開くことによって利用される。攻撃者が利用者に細工した DOC ファイルを開かせる方法としては、以下の方法などが考えられる。

- ・ メールに添付して利用者へ送りつける
- ・ メールに DOC ファイルを指す URL を書いて利用者へ送りつける
- ・ SEO によって利用者に DOC ファイルを指す URL をクリックさせる
- ・ 元々公開されている正規 DOC ファイルを不正 DOC ファイルに置き換えておき利用者が開くのを待つ
- ・ 正規ウェブページを改ざんしてウェブページ閲覧時に不正 DOC ファイルを自動的に開かせる

### 6.3. 攻撃状況

この脆弱性は、マイクロソフト社のブログによれば、2006 年 5 月 19 日の時点でこの脆弱性を利用した標的型攻撃が行われていることが確認されている。

<http://blogs.technet.com/msrc/archive/2006/05/19/429353.aspx>

この脆弱性に対するセキュリティパッチの公開は 2006 年 6 月 14 日に行われている。

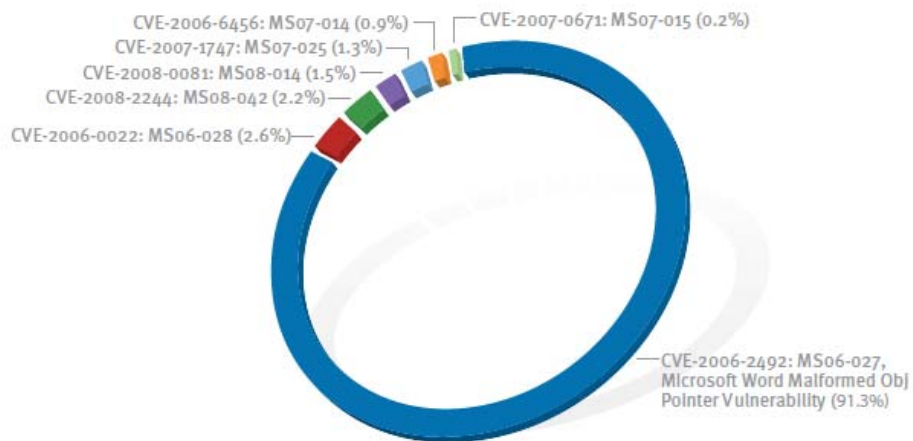
<http://www.microsoft.com/japan/technet/security/bulletin/MS06-027.msp>

したがって、セキュリティパッチ公開前に攻撃が行われていたことから、この攻撃はゼロデイ攻撃であったと言える。また、この脆弱性を利用した攻撃はこれまでに多数行われていることが確認されている。例えば、マイクロソフト社が公開したレポート「Microsoft Security Intelligence Report Volume 6」では、2008

年下半期において最も攻撃に利用された脆弱性であることが報告されている。

(出典 : 「Microsoft Security Intelligence Report Volume 6 July through December 2008」

[http://www.techfiles.de/presse/pressemappen/SIR\\_V6/MicrosoftSecurityIntelligenceReport\\_Nr6\\_July-Dec2008.pdf](http://www.techfiles.de/presse/pressemappen/SIR_V6/MicrosoftSecurityIntelligenceReport_Nr6_July-Dec2008.pdf))



また、ウェブサイト上で公開されているセキュリティ研究者からは、2010年2月においてもこの脆弱性を利用した標的型攻撃が行われていることが報告されている。

## 7. 「脆弱性を利用した攻撃」への対策

### 7.1.脆弱性/攻撃情報の入手と対応

脆弱性を利用した攻撃を防ぐためにまず行なうべきことは、脆弱性および攻撃情報を入手し、その脅威を認識することである。既知の脆弱性は、日本国内においては、IPA と JPCERT/CC が共同運営している「JVN」で確認することができる。また、ゼロデイ攻撃を含む実際の攻撃発生状況については、セキュリティベンダーなど日々攻撃を監視/対応している信頼できる組織が情報源となる。

### 7.2.直接的な対策

脆弱性または攻撃の存在を確認した後は、それに対する自身への影響を精査し、対策を講じる必要がある。脆弱性に対するセキュリティパッチが公開されている場合には、それを適用することが基本となる。セキュリティパッチが未公開であったり、セキュリティパッチを適用することで使用中のソフトウェアの動作に支障が出るなどでセキュリティパッチの適用を控えたりする場合には、設定を変更する、当該ソフトウェアの使用を控えるなど脅威を緩和するための善後策を導入する必要がある。

### 7.3.汎用的な対策

セキュリティパッチを適用できない場合や、標的型攻撃で未知の脆弱性を利用される可能性への対策として、脆弱性を利用した攻撃への汎用的な対策を講じておくことが有効である。多くの脆弱性を利用した攻撃への汎用的な対策としては以下が考えられる。

- Windows の DEP (Data Execution Prevention) 機能の利用
- より安全性の高い OS の利用 (例 : Windows XP SP2 の利用を Windows XP SP3、Windows Vista SP1、または Windows 7 の利用に変更するなど)
- 振る舞い検知機能を持つウイルス対策ソフトの利用

#### 7.3.1. Adobe Reader/Acrobat の脆弱性を利用した攻撃への汎用的な対策

本報告書の第 3 章で明らかにした Adobe Reader/Acrobat の脆弱性を利用した攻撃への汎用的な対策としては、以下が考えられる。

- Adobe JavaScript の無効化
- Adobe JavaScript ブラックリストフレームワークによる特定メソッドの無効化
- ブラウザ内での PDF 表示の無効化

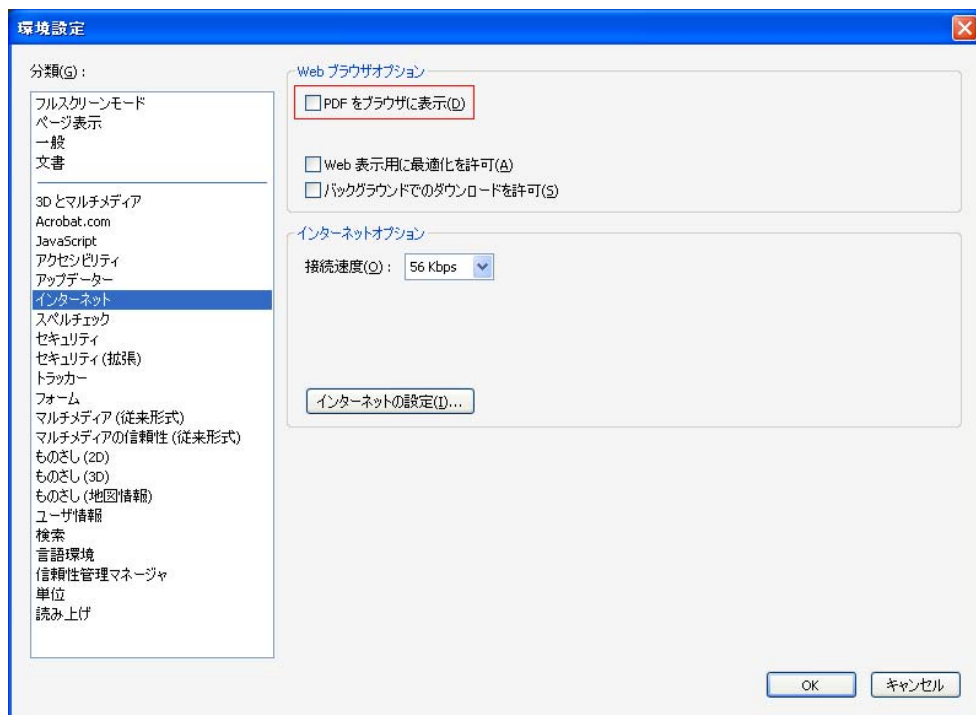
Adobe JavaScript の無効化の方法については過去の報告書で既出であるので、過去の報告書を参照されたい。(参考 : [http://www.ipa.go.jp/security/vuln/report/documents/newthreat\\_report\\_2009.pdf](http://www.ipa.go.jp/security/vuln/report/documents/newthreat_report_2009.pdf))

Adobe JavaScript ブラックリストフレームワークによる特定メソッドの無効化は、Doc.media.newPlayer メソッドの脆弱性 (JVND-2009-002451)が発見されてから脆弱性が修正されたバージョンがリリースさ

れるまでの暫定的な対処方法としてアドビシステムズ社から公開された、特定の JavaScript API の呼び出しを無効にする方法である。この方法を用いることによって、Doc.media.newPlayer メソッドなど脆弱性が存在するメソッドのみを無効にすることができる。この方法は、Adobe JavaScript の使用を必要としており、Adobe JavaScript 全体の無効化を行いたくない場合に有効な方法である。設定方法の詳細は、アドビシステムズ社が公開している情報を参照されたい（参考：<http://kb2.adobe.com/jp/cps/236/236209.html>）。

ブラウザ内での PDF 表示の無効化は、脆弱性を利用するよう細工された PDF をウェブページ閲覧時にブラウザの PDF 表示用アドオンを利用して表示させる手口を防ぐために有効である。この設定は、Adobe Reader/Acrobat の「環境設定」メニューから行うことができる。

### Adobe Reader 9.3 における「PDF をブラウザに表示」の設定画面：



#### 7.3.2. Microsoft Word の脆弱性を利用した攻撃への汎用的な対策

本報告書の第 6 章で明らかにした Microsoft Word の脆弱性を利用した攻撃への汎用的な対策としては、以下が考えられる。

- MOICE (Microsoft Office Isolated Conversion Environment) の利用

MOICE (Microsoft Office Isolated Conversion Environment) は、マイクロソフト社が提供している、「Word/Excel/PowerPoint 2007 ファイル形式用 Microsoft Office 互換機能パック」に含まれる、Microsoft Office 2003 形式ファイルを Microsoft Office 2007 形式ファイルに変換するためのプログラムである。

MOICE についての詳細は、マイクロソフト社が公開している情報を参照されたい。

(参考 : <http://www.microsoft.com/japan/technet/security/advisory/937696.mspx>)

本報告書の第 6 章で明らかにしたように、Microsoft Office の脆弱性を利用した攻撃では、ファイル内に埋め込まれているデータを利用して別の実行ファイルを作成して実行することによって、実際の不正な動作を行うことが多い。しかし、MOICE のようなプログラムを利用してファイル形式を変換すると、本来のファイル形式とは別に埋め込まれているこのようなファイルデータは、変換後のファイルからは欠落すると考えられる。また、変換前と変換後のファイル形式の違いから、脆弱性の利用そのものできない可能性が高いと考えられる。さらに、変換前のファイルが脆弱性を利用するよう細工されている場合、正常な状態とは異なることが原因で変換処理に失敗する可能性も考えられる。

例として、MOICE をインストールした Microsoft Word 2003 を利用して本報告書の第 6 章で解析したファイルマルウェア 2 を開き、ファイル形式を変換した場合は元のファイル内に含まれていた内容（中国語の文章）が表示されるが、脆弱性の利用は行われなかった。

#### 7.4. ダメージコントロール

想定外のゼロデイ攻撃を受けたり、適切な対策が施されていないなど何らかの理由でマルウェアの実行が開始されてしまう可能性は否定できない。そこで、リスク管理としてマルウェアが実行されてしまった場合にその被害/影響範囲を極力抑えるための施策を施しておくこと（ダメージコントロール）が重要になる。これらの対策は組織内ネットワークのシステム管理者が実施することはもちろん、一般ユーザにおいてもパーソナルファイアウォールの適切な使用や不必要に Administrator 権限を持つユーザでログインしないなどの注意を払うことが望ましい。

施策	効果
外部への通信の制限(ファイアウォール、プロキシの利用など)	マルウェアが行なう外部との通信の遮断によりバックドアが利用されることおよび内部情報が漏洩することを防ぐ。
不正通信の検知(IDS や監視サービスの利用など)	マルウェアが行なう外部との通信を検知することにより、マルウェアの存在を検知する。
アクセス可能リソースの制限(アクセス権限の適切な設定など)	バックドアが利用された場合に攻撃者がバックドアを経由してアクセスできるリソース範囲を最小限に留める。

## 8. まとめ

本章では今回の調査のまとめとして2ケースの攻撃を比較し、これまでの分析結果から攻撃者と攻撃の特徴について考察する。

### 8.1.2 ケースの攻撃の比較

今回の調査で解析したマルウェア1、マルウェア2の二つの標的型攻撃を比較した表を以下に示す。

内容	マルウェア1	マルウェア2	類似性(○:あり ×:なし △: 一部のみ類似)
メール送信日時	2009年12月25日	2010年3月18日	N/A
メール送信方法	無料のメールサービスAを利用	無料のメールサービスBを利用	△ (無料のメールサービスを利用)
メール差出人詐称	あり	あり	○ (実在の組織のアドレスを利用)
メール本文内容	日本語の誤送信メールを偽装	日本語の誤送信メールを偽装	○
添付ファイル	PDFファイル	DOCファイル	△ (広く利用されているアプリケーションの脆弱性を利用)
利用脆弱性種類	APSA09-07 (CVE-2009-4324)	MS06-027(CVE-2006-2492)	×
ダミー文章ファイルの表示	あり	あり	○
オリジナルファイルの無害化	なし	あり	×
バックドアインストール方法	svchostを利用してサービスとして動作	svchostを利用してサービスとして動作	○
バックドアインストール用プログラムの削除	あり	あり	○
バックドア通信方法	独自プロトコルとダウンロード機能	独自プロトコルとダウンロード機能	○
接続先ホスト(C&Cサーバ)	japanesecode.myfw.us	japanesecode.myfw.us	○

内容	マルウェア1	マルウェア2	類似性(○:あり ×:なし △: 一部のみ類似)
接続先ポート番号	13522	13281	△ (一般に利用されていないポ ート番号)

マルウェア1、マルウェア2の二つの標的型攻撃は、多くの類似点を持っている。脆弱性を利用した結果としてインストールされるバックドアプログラムは、使用するポート番号（「13522」と「13281」）や使用可能なコマンド（「Uninstall」の有無）など若干の違いがあるものの非常に類似しており、言うなればバージョンの異なる同じ種類のプログラムであると言える。バックドアへの接続は、接続先のC&Cサーバ(japanesecode.myfw.us)が同一である。また、使用するポート番号は異なっているが、使用する通信プロトコルは同一である。接続時に送信する固定値の数字（「1225」と「2010」）は異なっているが、この数字はプログラムのバージョン番号、または攻撃種類を表すIDである可能性が考えられる。これらの類似点は、これら異なる日付に異なるメール受信者に対して行われた二つの標的型攻撃が、同一の攻撃者の手によって行われたことを示唆していると言える。

この攻撃者の活動を推測するため、一連の攻撃に関係する日付を下の表にまとめる。なお、ファイル作成日付は、PDFおよびDOCファイルについてはファイルプロパティ内の情報を、実行ファイルについてはプログラムのコンパイル時に実行ファイルのPEヘッダ内に記録される日時の情報を参照した。

検体種類	マルウェア1		マルウェア2	
攻撃メール受信日時	2009年12月25日		2010年3月18日	
脆弱性情報公開日	2009年12月15日(APSA09-07)		2006年6月14日(MS06-027)	
ドメイン登録日 (japanesecode.myfw.us)	2009年8月24日		2009年8月24日	
細工された文章ファイル作成日時	マルウェア1	2009年12月22日 16:13:31	マルウェア2	不明(*1)
ダミー文章ファイル作成日時	正常なPDFファイル (ダミー)	2007年5月	正常なDOCファイル (ダミー)	2010年3月
ドロPPER作成日時	マルウェア1-1	2009年7月8日 15:05:00	マルウェア2-1	2010年3月10日 16:15:13
バックドアプログラム作成日時	マルウェア1-2	2009年7月8日 11:23:31	マルウェア2-2	2010年3月10日 11:07:41

\*1 プロパティ内に情報が存在しない。意図的に情報が削除されている可能性がある。

## 8.2. 攻撃メールの特徴

今回調査した2ケースの攻撃に使用されたメールの特徴としては、以下が挙げられる。

- ・日本語や日本の時事に精通している人間がメール本文を作成する
- ・攻撃者の身元が分からないようにする（無料メールサービスを利用、差出人を詐称、接続元 IP アドレスを変更）
- ・同じ組織に対して頻繁に送信する
- ・攻撃メールの内容は標的に特化したものではなく汎用的な内容を利用する
- ・アプリケーションの脆弱性を利用するファイルをメールに添付する
- ・攻撃先から不審メールとして扱われ攻撃に失敗することも多い

### 8.3.マルウェアの特徴

今回調査した 2 ケースの攻撃に使用されたマルウェアの特徴としては、以下が挙げられる。

- ・中国語 OS 環境で作成している
- ・マルウェア作成者の身元が分からないようプロパティ情報を削除している
- ・ウェブ上などで公開されている内容を利用してダミーのドキュメントファイルを偽装のために開く
- ・ウイルス対策ソフトによる検出回避のためか、一度作成したマルウェアを改良して再利用している
- ・インストールするバックドアの機能は最低限に絞り攻撃意図を分からないようにする
- ・バックドアに利用するサーバは利用可能な時間を制限し、その制御のために DNS を変更する
- ・使用するポート番号やプロトコルは独自のものを使用する

### 8.4.利用された脆弱性の特徴

今回調査した 2 ケースの攻撃に利用された脆弱性の特徴としては、以下が挙げられる。

- ・一般に広く使用されておりメールに添付される頻度も高いアプリケーションの脆弱性を利用する
- ・使用する脆弱性の種類やアプリケーションの種類は固定せず、ゼロデイなどの新しい脆弱性だけでなく古い脆弱性を利用することもある

### 8.5.攻撃タイミングの特徴

今回調査した 2 ケースの攻撃の行われたタイミングの特徴としては、以下が挙げられる。

- ・マルウェアを用意してから攻撃を行うまで、ゼロデイ脆弱性の発見を待つ周到に用意して攻撃を行う（ケース 1）
- ・人々が興味を持つ社会的なニュースが発生したタイミングに合わせてマルウェアを用意して攻撃を行う（ケース 2）
- ・マルウェア作成から攻撃メール送信までが短期間に行われる場合（ケース 1）と、そうでない場合（ケース 2）がある

### 8.6.今後に向けて

今回の調査では、脆弱性を利用した脅威の調査として、IPA「不審メール 110 番」に相談のあった 2 件の標的型攻撃メールをサンプルとして、攻撃/被害状況を正確に掴むことが困難な標的型攻撃の調査を行った。

分析によって、攻撃の特徴をある程度明らかにすることができたが、あくまで2ケースのサンプルのみを基に行った分析であり、誰が攻撃対象になっているのか、どの程度の頻度で攻撃が行われているのか、どの程度の割合で攻撃が成功しているのか、攻撃が成功することによってどのような被害が発生しているのか、攻撃者が誰でどのような目的を持って攻撃を行っているのかなど、いまだ不明な点は多い。今後はさらに調査目的を明確にし、それに特化した調査を行っていくことが必要であろう。また、対策としては攻撃に脆弱性が利用されていることを踏まえ、セキュリティパッチの適用を徹底するなどの基本的な対策とともに、利用頻度の高い脆弱性に対しては Adobe Reader の JavaScript 機能を無効にする、Microsoft Office の MOICE を利用するなどの汎用的な対策を併用することが望ましい。さらに、基本的な対策のみで防御可能な攻撃が行われている状況を踏まえ、ファイアウォールを適切に利用するなどのシステム面の防備や、「怪しいメールを安易に開かない」といった注意など、基本的な対策を怠らざることの重要性も改めて認識したい。

以上