サイバー情報共有イニシアティブ（J-CSIP） 運用状況 [2019 年 7 月～9 月] 《 付録 》 ～標的型攻撃に関連すると思われるウイルスの解析事例～

2019 年 10 月 31 日
IPA（独立行政法人情報処理推進機構）
セキュリティセンター

目次

1　はじめに

サイバー攻撃は、あらゆる企業・組織に対して試みられている。このような状況において、自組織（あるいはISAC[1]等の会員組織）に試みられた攻撃を分析し、その情報を蓄積し、他の組織と情報共有するといった活用を行っていくことは一定の意義があるものと考える。特に、攻撃者によって意図的に狙われて攻撃された場合（標的型攻撃）、分析・蓄積・共有を重ねた結果、自組織や関連業界が過去に受けた攻撃との関係性（連続性）の有無や攻撃手口の類似性等の点について、把握できる場合がある。こういった分析には様々な観点や方法があり、その一つとして、攻撃に用いられたウイルス等の不正ファイルの解析がある。

IPA セキュリティセンターでは、J-CSIP の運用をはじめとして、サイバー攻撃の情報を分析するため、必要に応じてウイルス等の解析を行っている。その中には、広く無差別にばら撒かれたウイルスと思われるものだけでなく、特定の業種・業界を狙った攻撃に使用されたと思われるウイルスもある。そして、これらの情報について、必要な範囲で情報共有を進めている。

この活動の中で、本四半期、標的型攻撃で使用された可能性のあるウイルスを公開情報より入手した。詳しく解析を行ったところ、このウイルスは、「アイコンや拡張子の偽装」、「特定のセキュリティソフトの停止」、「特定の時間帯のみ動作を行う」、「不正接続先から特定の応答が得られないと動作を止める」といった、ウイルス自身の存在、攻撃活動の露見、そしてウイルス解析者による解析を避けるような様々な仕掛けがみられるものであった。

本書は、このウイルスについて解析を行った結果を、ひとつの事例として報告する。これは、あくまで特定のウイルス 1 件の解析結果を参考情報として説明するものであり、今後のサイバー攻撃への直接的な対策となるものではない。一方、このようなウイルスの特徴や動作の仕組みを把握することは、ウイルス解析者のみならず、企業・組織のセキュリティ担当者においても、サイバー攻撃への対応・対策を検討する上で、役立つ可能性があるであろうと考え、報告するものである。

本書の対象読者

本書では、次の方々を主な対象読者と想定している。

- 企業の CSIRT[2]や ISAC 等、組織のセキュリティを扱う部門の方
- ウイルスの解析等を行う方、ウイルスの解析を外部専門組織へ依頼して業務を遂行する方

[1] Information Sharing and Analysis Center（ISAC、アイザック）。同じ業界の民間事業者同士でサイバーセキュリティに関する情報を共有し、サイバー攻撃への防御力を高めることを目指して活動する民間組織。

[2] Computer Security Incident Response Team（CSIRT、シーサート）。組織内の情報セキュリティ問題を専門に扱う、インシデント対応チーム。

2　ウイルスの概要

本ウイルスは実行形式のファイルであり、Windows 上でウイルスを実行する（ファイルを開く）と感染活動を開始する。ファイルの外見は、図 1 で示すようにアイコンを Word 文書ファイルのように偽装しており、RLO[3]により拡張子を Word 文書ファイルの「.docx」に見せかけている（実際の拡張子は、「.scr」[4]である）。

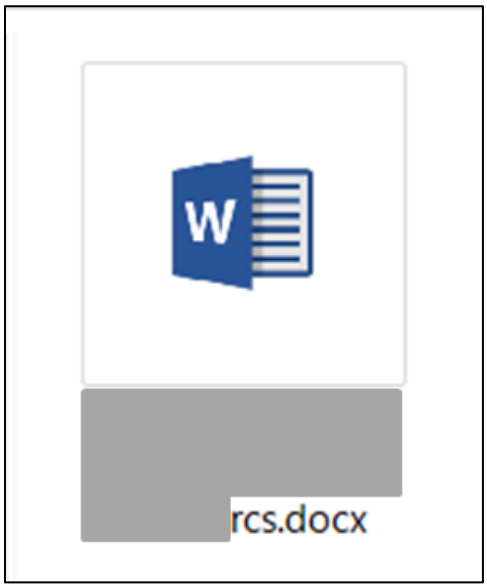


図 1 ファイルの外見

本ウイルスのように、ファイルのアイコンを Word 等の文書ファイルに見せかけるという細工は、過去の標的型攻撃を含め、ウイルスに多くみられる手口である。これによって、実際は実行形式のファイル（exe ファイル、scr ファイル等）であるにも関わらず、利用者にはあたかも文書ファイルであるかのように誤認させ、ファイルを開かせようとしている。

本ウイルスが実行されると、特定のセキュリティソフトの動作を停止させ、表示用ダミーファイル[5]が表示される。その後、ウイルスの感染を永続化[6]する動作が行われた後に、不正接続先への通信が行われる。不正接続先への通信は複数パターン発生し、特定の時間にのみ、または不正接続先から特定の応答が返ってきた場合のみ通信を行うという、条件付きのものがある。条件に該当しない場合は、一定期間動作を止め（sleep）、再度不正接続先への通信を行う。

不正接続先からは、攻撃者の用意した別のウイルス（以降、二次ウイルス）が端末（パソコン）へダウンロードされ、実行される仕組みとなっている。このため、このウイルスは一般的に「ダウンローダ」というカテゴリに分類できる。なお、ダウンロードされる二次ウイルスについては、本書では範囲外とする。

本ウイルスの動作概要を、図 2 に示す。

---

[3] Right-to-Left Override。テキストの表示を右から左へ流れるように変更する特殊な文字（制御文字）。

[4] .scr の拡張子のファイルは、スクリーンセーバーファイルである。拡張子が異なるだけで、内容は exe ファイル等と同じ実行形式のファイルである。

[5] 3.2 章で詳しく説明する。

[6] Windows の再起動時等に自動的にウイルスが実行されるようにすること。

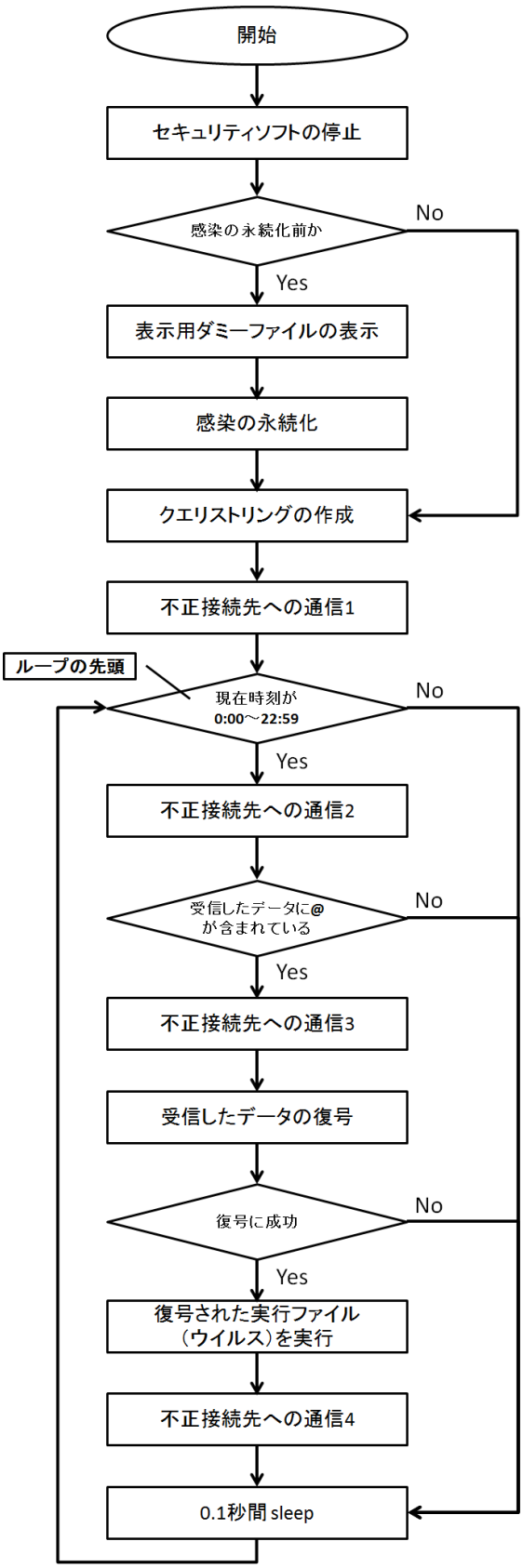


図 2 ウイルスの動作概要図

3 ウイルスの解析

本章では、ウイルスの動作を詳しく解析した結果、判明した点を説明する。

3.1 セキュリティソフトの停止

本ウイルスを実行すると、まず、特定のセキュリティソフトに関するプロセスが起動しているか否かを確認し、起動していた場合は、当該プロセスを停止するという処理を行う。これにより、本ウイルスだけでなく、不正接続先からダウンロードされる二次ウイルスが、当該セキュリティソフトによって検知されることを回避しているものと思われる。

ここでは、数文字ずつに分割した文字列[7]を結合し、「●●●NT.exe」[8]という文字列を作成している。そして、CreateThread 関数を使い新たなスレッドを作成し、その文字列を渡す。スレッド内の処理は、渡された文字列と一致するプロセスを探し、停止するというものである。本ウイルスでは、この 1 種類のプロセスのみを停止対象としているが、同種の別のウイルスでは、この処理を繰り返すことで、セキュリティソフト関連の複数のプロセスを停止させている場合がある。

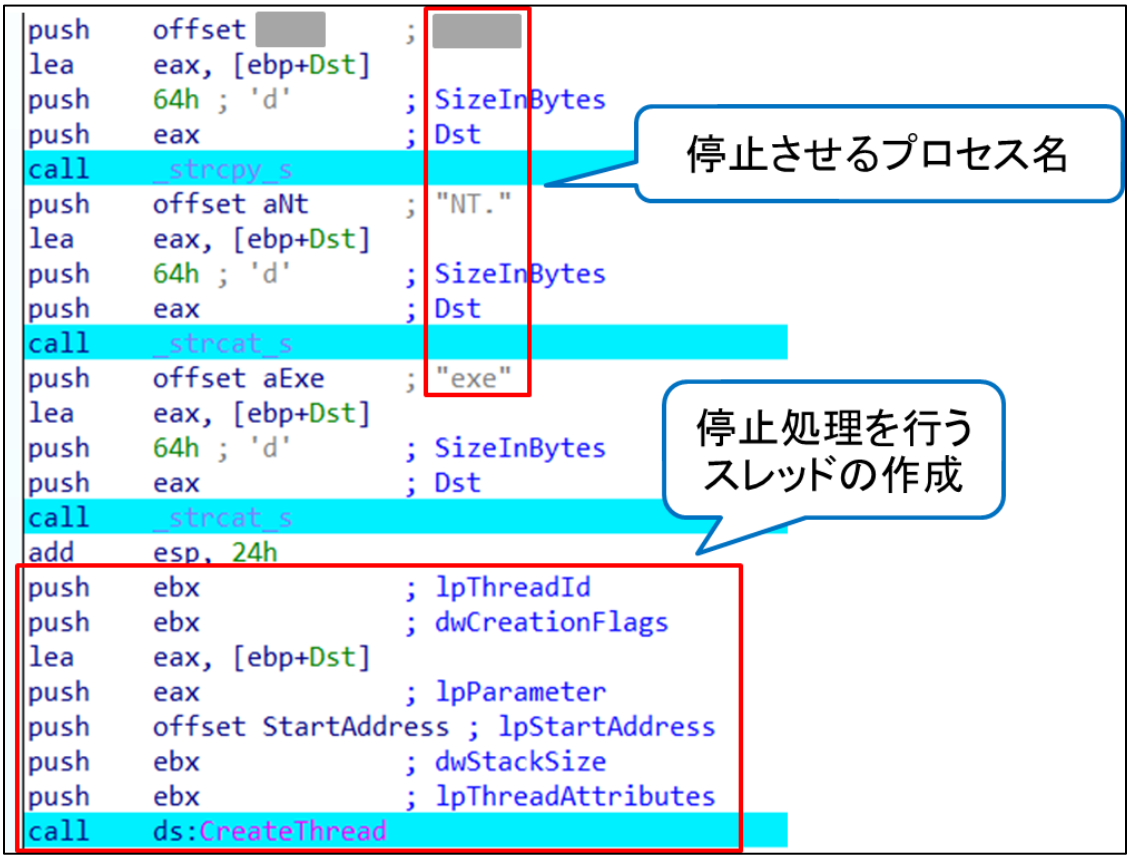


図 3 停止させるプロセス名の作成と停止処理用のスレッドの作成

[7] 解析された際に文字列が簡単に抽出されないよう、このような処理となっていると思われる。
[8] 特定のセキュリティソフトのプロセス名と思われる。

3.2 表示用ダミーファイルの表示

セキュリティソフトの停止処理の後、ウイルス自身の内部（ファイルの末尾部分）に埋め込まれている Word 文書ファイルのバイナリデータを取り出し、自身と同一のフォルダに、別のファイルとして保存する。そして、その文書ファイルを開き、画面に表示する（図 4）。

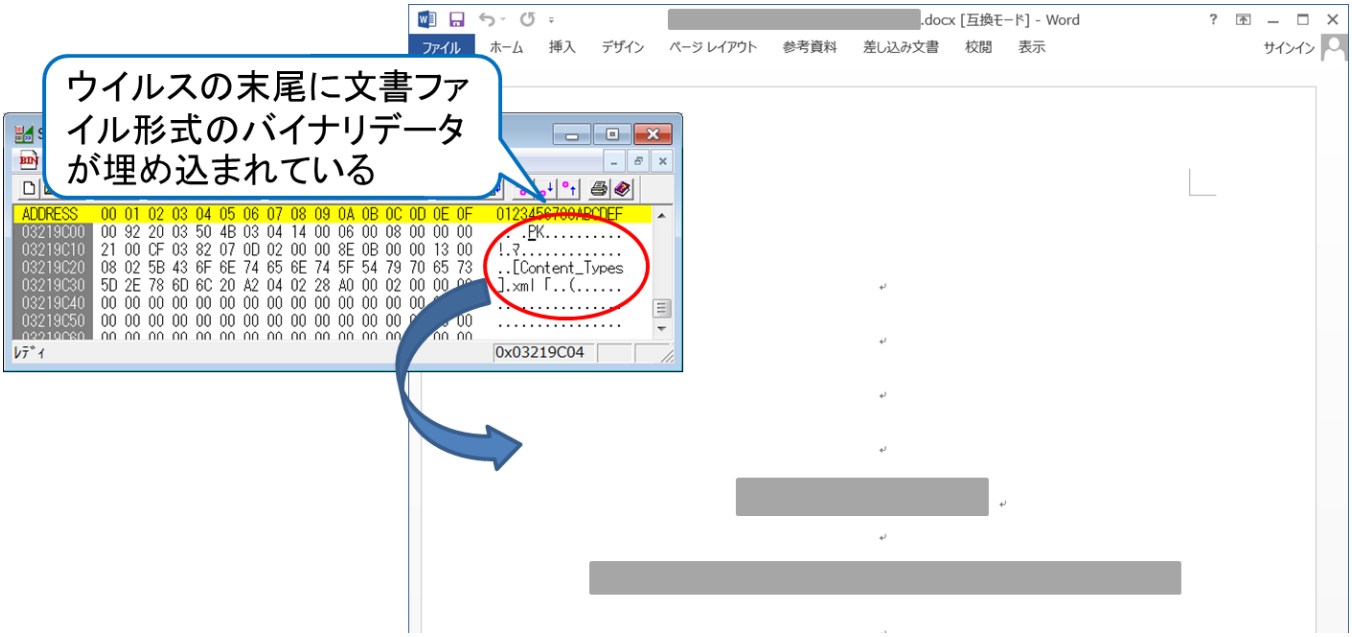


図 4 表示用ダミーファイルの表示

この文書ファイルには、特定の組織に関係するかのように見せかける内容が書かれていた。本ウイルスは Word 文書ファイルのアイコンを装っている実行形式のファイルであるため、このように結果的に Word 文書ファイルを画面表示することで、利用者に不審だと気付きにくくさせることを狙っている（利用者にとっては、Word 文書ファイルアイコンのファイルを開いたら、Word 文書が画面に表示された、という当然の動作に見える）。このような目的で使われる文書ファイルを、「表示用ダミーファイル」と呼ぶ[9]。

3.3 章で後述するが、ウイルスファイル自身は別のフォルダに移動するため、元々ウイルスが置かれていた場所には、この表示用ダミーファイルのみが残る（すなわち、ウイルスファイルと、文書ファイルが入れ替わる）仕掛けとなっている。これにより、一度ウイルスを実行してしまうと、利用者はウイルスの存在に気付きにくくなると思われる。また、ウイルスファイルが別のフォルダに移動した次回以降の起動時は、この動作（表示用ダミーファイルの生成と表示）は行われない仕組みになっている。

[9] 「囮ファイル」、「デコイファイル」とも呼ばれる。

3.3 感染の永続化

ウイルスであるか否かに関わらず、実行ファイルは Windows のシャットダウン等と同時に動作が終了し、その後は基本的には利用者によって再度実行されるまでは動作しない。そのため、Windows を起動したり、ログオフ状態からログオンした際、ウイルスが自動で実行されるよう、密かにシステムへ設定を行うことはウイルスに多くみられる動作である。これを、感染の永続化と呼ぶ。感染の永続化の方法は複数存在する[10]。

本ウイルスも、表示用ダミーファイルを表示した後、特定のレジストリへ、自分自身を自動実行用のファイルとして登録するという手口を使って感染の永続化を行う。

本ウイルスは、自分自身が実行された場所（フォルダ）を確認し、実行場所が環境変数 temp （以降、%temp%）で示されるフォルダでなければ、感染の永続化の処理を行う。

表 1 感染永続化用のレジストリ

レジストリキー	HKCU¥SOFTWARE¥Microsoft¥Windows¥CurrentVersion¥Run¥Ravirra
値	%temp%¥avirra.exe

まず、レジストリに表 1 のレジストリキーと値を登録する。その後、「%temp%」フォルダに「avirra.exe」というファイルが存在しない場合、ウイルス自身を「avirra.exe」というファイル名で「%temp%」フォルダに移動する。これにより、以後、利用者がログオンするたびに「%temp%¥avirra.exe」（ウイルス）が実行される状態になる（すなわち、感染が永続化する）。

この感染の永続化の処理は、既に永続化が済んでいる状況では実行されない。具体的には、自動実行用のファイルとして 2 回目以降にウイルスが実行された場合、「%temp%」フォルダが実行場所となるため、二重に行われることはない仕組みとなっている。

本ウイルスは、自分自身の隠し場所として「%temp%」フォルダ（Windows 上の一時的なファイルの保存場所）を使用しているとも言える。「%temp%」フォルダには他にも多数のファイルが置かれるため、ウイルスファイルを見つかりにくくする細工の一種と思われる。一方で、レジストリ上の自動実行用の設定として「%temp%」内のファイルが指定されていることは不自然であり、本ウイルスに限らず、このような自動実行用の設定を発見した場合は、不正な状態であることを疑ってみるべきである。

[10] 本ウイルスのような手口のほか、「スタートアップのフォルダにウイルスのショートカットファイルを作成する」、「Windows のサービスとしてウイルスを登録する」といった手口がある。

3.4 クエリストリングの作成

感染の永続化を終えると、1 分間の sleep をした後、不正接続先への通信を行う際に使用する、クエリストリングの文字列を作成する。クエリストリングとは、URL の末尾に「変数名=値」の形式で付加されるデータである。本ウイルスでは、図 5 のように 2 つのクエリ変数(「UID」と「ws」)が使用されることがあり、本章ではそれぞれが何の値であるのか説明する。

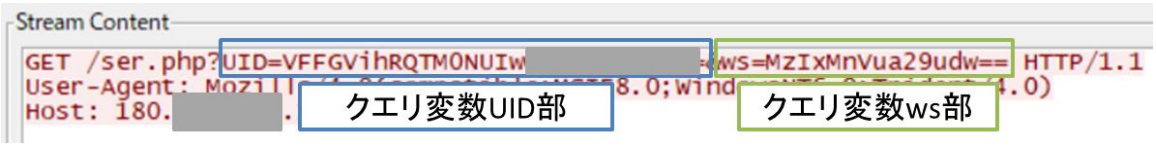


図 5 クエリストリングの内容例

3.4.1 クエリ変数 1「UID」

クエリ変数「UID」には、感染端末のホスト名とネットワークアダプタの MAC アドレスを文字列として結合し、その内容に排他的論理和(eXclusive OR:XOR)による暗号化と Base64 方式によるエンコードを行った値が設定される。

(1) 感染端末のホスト名および MAC アドレスの取得と文字列の結合

まず、gethostname 関数を実行して感染端末のホスト名を取得し、続いて GetAdaptersInfo 関数を実行して感染端末のネットワークアダプタの情報を取得する。取得したネットワークアダプタの情報の中から MAC アドレスを取り出して ASCII 文字列に変換し、先に取得した感染端末のホスト名と文字列を結合する(図 6)。

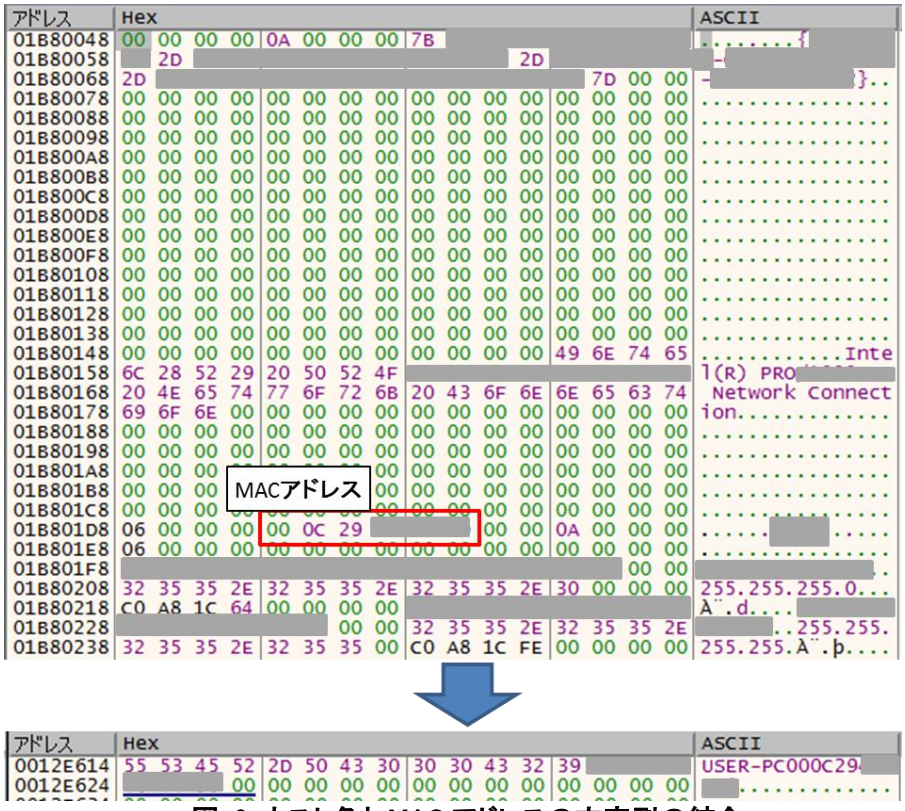


図 6 ホスト名と MAC アドレスの文字列の結合

本ウイルスがホスト名や MAC アドレスを取得して不正接続先へ送信している目的は不明であるが、不正接続先のサーバ側でホスト名と MAC アドレスの値を確認し、応答内容を制御したり、以降の攻撃活動のための情報として利用している可能性がある。

また、本ウイルスでは、ホスト名や MAC アドレスの値によって動作が変わるような仕組みは見られなかったが、標的型攻撃で使用されるウイルスには、感染端末のホスト名や MAC アドレスを確認し、攻撃対象の端末でない場合、それ以降の動作を行わないものが一部存在する。また、ウイルス解析で使用されるサンドボックスや仮想化ソフトウェアでは、固有のホスト名や MAC アドレスが使用されている場合があり、それらを検知して動作を停止するウイルスもある。いずれにせよ、ウイルスが動作している環境の情報を把握するため、攻撃者がこれらの情報を入手しようとすることは、よく見られる傾向である。

(2) XOR 暗号と Base64 方式によるエンコード

1 から 5 の数値を並べたものを「XOR キー」とし、ホスト名と MAC アドレスを結合した文字列に対して、1 バイトずつ順番に XOR による暗号化を行う。6 バイト目以降は、また 1 から順番に XOR キーを使用する（図 7）。

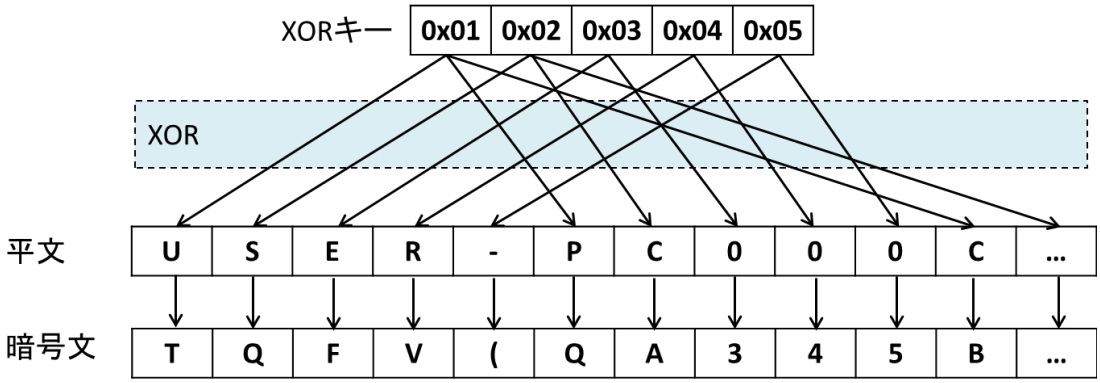


図 7 XOR による暗号化処理

XOR によって暗号化された値を更に Base64 方式でエンコードしたものが、1 つ目のクエリ変数「UID」の値となる（図 8）。

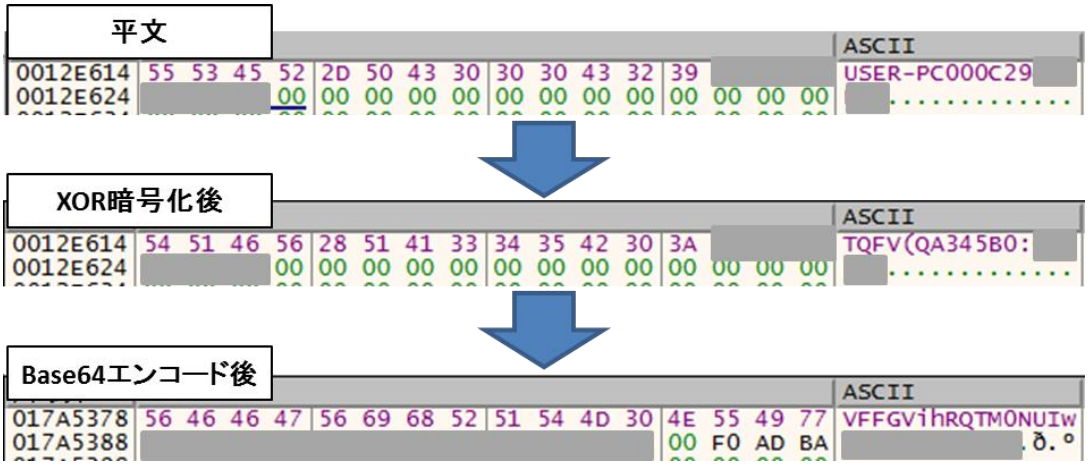


図 8「UID」の値の例

本ウイルスは、このように「UID」の値として送信するデータを二重に加工している。いずれも暗号化としては

非常に簡単な処理ではあるが、ウイルスの内部構造を解析しなければ、このデータを容易に復元することはできない。

3.4.2 クエリ変数 2「ws」

クエリ変数「ws」には、感染端末上のセキュリティ製品の有無と、OS の種別の確認結果を文字列として結合し、その内容を Base64 方式によるエンコードを行った値が設定される。

(1) 感染端末上のセキュリティ製品の有無の確認

セキュリティ製品に関連すると思われるレジストリパス(表 2)を順番に確認し、存在する場合はそれぞれ対応するレジストリキーの値を取得する。どのレジストリパスも存在しない、すなわち該当のセキュリティ製品が端末にインストールされていない場合は、"unkonw"という値となる("unknown"のミススペルであるのか、攻撃者が意図したものであるかは定かではない)。

表 2 ウイルスが確認するレジストリパス

#	レジストリパス	レジストリキー
1	HKLM¥SOFTWARE¥●●●●¥●●●●Protection¥ CurrentVersion	PRODUCTVERSION
2	HKLM¥SOFTWARE¥●●●●¥●●SP	●●●●●_GUID
3	HKLM¥SOFTWARE¥●●●●¥Liveup	mid

感染端末にどのようなセキュリティ製品がインストールされているかについて、攻撃者の関心があることが伺える。一方、世の中に出回っているウイルスの中には、10 種類を超えるセキュリティ製品の有無の確認処理を行うものもある。本ウイルスでは、なぜこの 3 種類のみを確認対象としているのかは不明である。

(2) OS の種別の確認

まず、GetNativeSystemInfo 関数または GetSystemInfo 関数を使用して、感染端末の OS の CPU ビット数(32 ビットまたは 64 ビットかを判断するためと思われる)を取得する。その後、RtlGetNtVersionNumbers 関数と GetVersionEx 関数、GetSystemMetrics 関数により OS のバージョンを特定し、それに応じて本ウイルスが独自に定義している値(表 3)を選択する。

表 3 ウイルスが独自に定義している OS のバージョンごとの値

OS のバージョン	値
Windows NT 4.0	1
Windows 95	2
Windows 98	3
Windows ME	4
Windows 2000	5
Windows XP	6
Windows XP Professional x64 Edition	7
Windows Server 2003	8
Windows Server 2003 R2	9
Windows Vista	10
Windows Vista Windows Server 2008	11
Windows 7	12

OS のバージョン	値
Windows Server 2008 R2	13
Windows 8	14
Windows Server 2012	15
Windows 8.1 Windows Server 2012 R2	16
Windows 10 Anniversary Update Windows Server 2016	18
上記以外の Windows 10 Windows Server 2016 の version 1709 以降	19
その他	0

攻撃者にとって、送り込んだウイルスがどのような環境で感染に成功したのかは、以降の攻撃活動を進めるために重要な情報である。また、2 章で述べたとおり、本ウイルスは二次ウイルスをダウンロードして実行することが主たる目的である。Windows の実行ファイルは、OS のバージョンや CPU ビット数に動作が依存することがあるため、攻撃者はこれらの情報を得ることで、感染端末にダウンロードさせる二次ウイルスを切り替えている可能性もある。

(3) 文字列の結合と Base64 方式によるエンコード

ここまでの処理で得られた情報を、OS の CPU ビット数、OS のバージョンの値、セキュリティ製品の有無（レジストリの値）の順番で文字列として結合し、Base64 方式でエンコードしたもの（図 9）が 2 つ目のクエリ変数「ws」の値となる。

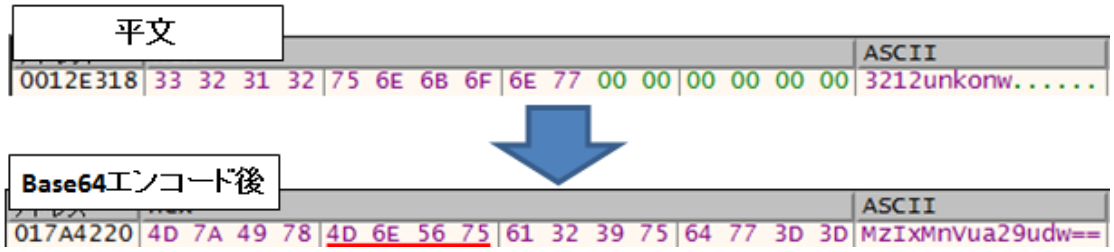


図 9「ws」の値の例

本ウイルスでは、「UID」の値と「ws」の値で加工（暗号化）の方式が異なっていた。一つの通信電文中で暗号化の方式が混在する事例は本ウイルス以外にも存在するが、本ウイルスがこのような処理をしている理由は不明である。強いて言うならば、「ws」の値を解読できても、ウイルスが何の情報を送ろうとしたのかは一見しても判別不能であるため、「ws」は Base64 方式によるエンコードのみで十分とした可能性が考えられる。

3.5 不正接続先への通信

クエリストリングの文字列の作成が終わると、不正接続先への通信を行う。

本ウイルスは、次の 2 つの不正接続先（表 4）に対して合計 4 回の通信を行う。特に、不正接続先 2 への通信が、二次ウイルスのダウンロードであり、特定の条件下でなければ、この通信は行われない仕組みとなっている。本章では、それぞれの通信について詳しく説明する。

表 4 不正接続先の一覧

不正接続先 1	http://180.●●●.●●●.155/ser.php
不正接続先 2	http://180.●●●.●●●.155/●●●●●.jpg

3.5.1 不正接続先への通信 1

1 回目の通信は、ウイルスが起動してから一度だけ、不正接続先 1 に 3.4 章で作成したクエリストリングを付加した URL へ行われる（図 10）。この最初の通信で、不正接続先のサーバにてウイルスに感染した端末の情報を把握しようとしているものと思われる。

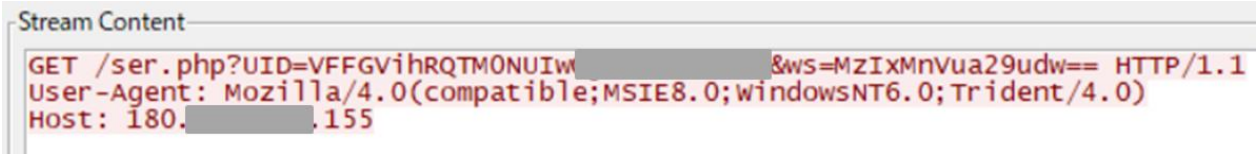


図 10 1 回目の通信の電文

1 回目の通信では、不正接続先からの応答電文について確認するような処理は無く、不正接続先からどのような値が返ってきても次の処理に進む。

なお、本ウイルスの通信の特徴として、HTTP リクエストヘッダ「User-Agent」について、ブラウザによる通信であるかのように装ってはいるが、正常な通信であれば；（セミコロン）の後ろにあるはずのスペースが無くなっている。この特徴は、構造の近い他のウイルスのいくつかの検体にも共通して見られる。これが攻撃者によるミスであるのか、理由は不明であるが、通信の監視等により類似したウイルスを発見することに役立つ可能性がある。

3.5.2 不正接続先への通信 2

ここから先は、この「不正接続先への通信 2」の処理を先頭として、ウイルスの処理全体でループ構造となっている。説明のため、この地点を「ループの先頭」と呼ぶ（全体のフローは図 2 を参照のこと）。

2 回目の通信は、現在時刻の確認と sleep の後に行われる。更に、不正接続先からの応答電文について、ウイルスが期待する文字が返ってこない場合は、その後の処理を実行しない仕組みとなっている。

(1) 現在時刻の確認と sleep 処理

まず、現在時刻が 0:00～22:59 の間であるかどうかを確認（図 11）し、時間内であれば、続いて約 23 分(1,373 秒)の sleep を実行する。その後、2 回目の通信を行う。この時、時間外（23:00～23:59）であった場合は、2 回目の不正接続先への通信は行わず、0.1 秒間の sleep を行った後、ループの先頭に戻る。すなわち、決められた時間内になるまで、待機するという動作である。

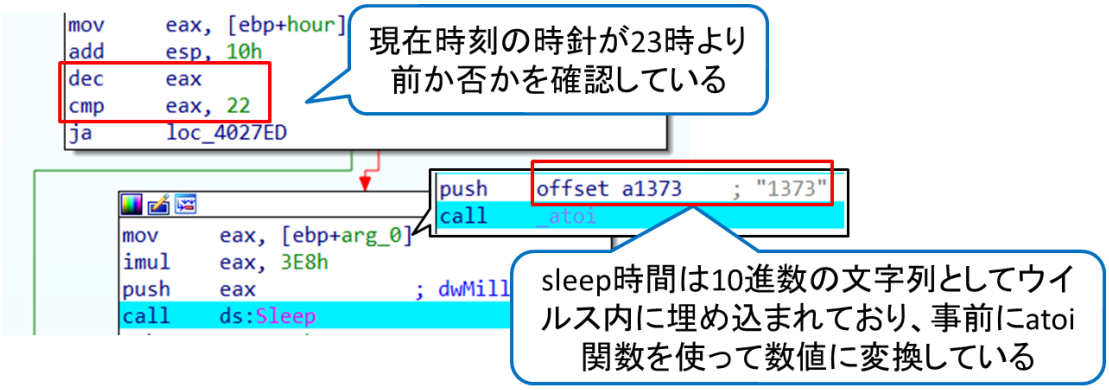


図 11 現在時刻の確認と 1,373 秒の sleep 処理

「0:00～22:59 の間のみ不正接続先との通信を行う」という挙動にどのような意味があるのか不明であるが、ここで注意したいのは、少なくとも「活動する時間帯を制限する機能を含んでいる」という点であり、ウイルスのほんの一部を修正することで、攻撃者は時間帯の指定を変更することができる。今回解析したウイルスにおいては、時間帯の指定はあまり意味をなさない（ほぼ 24 時間動作する）と思われるが、より活動時間帯を狭く絞り、動作を目立たせなくすることも可能である。同様に、1,373 秒という秒数の意図は不明であるが[11]、これにより通信を行う頻度が低くなるため、動作を目立たせなくするための細工となっている。

一般的に、利用者が端末を使用していないような深夜・早朝の時間帯に頻繁に通信が発生していると、組織内のネットワークの通信の監視や、通信ログの分析等により、ウイルスの活動が発覚する可能性が高くなる。このため、過去の標的型攻撃のウイルスにおいても、特定の時間帯のみ動作したり、通信の頻度を抑える仕組みを持ったものがあった。また、ウイルス解析の面においても、単純にウイルスを動作させただけでは十分に動作が確認できない等、解析を行いにくくする手口でもある。

(2) 不正接続先への通信と受信データの確認

2 回目の通信は、表 4 の不正接続先 1 に 3.4 章で作成したクエリストリングの「UID」のみを付加した URL へ行われる（図 12）。

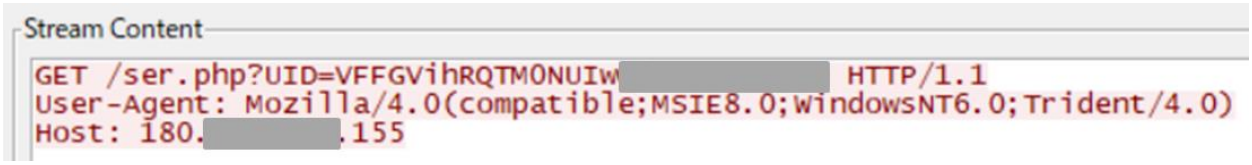


図 12 2 回目の通信の電文

そして、この 2 回目の通信では、不正接続先からの受信データ（応答電文の HTTP レスポンスボディ）に、”@”という文字（16 進数では 40h）が含まれているか否かを確認し（図 13）、含まれている場合のみ次の処理（不正接続先への通信 3）に進む。含まれていない場合は、0.1 秒間の sleep 処理を実行し、ループの先頭に戻る。

[11] 不正接続先のサーバ側で、不正接続先への通信 1 と通信 2 の間に 1,373 秒の間隔があるか否かにより、本物のウイルスによる通信か、解析者による調査のための通信かを判断している可能性もある。

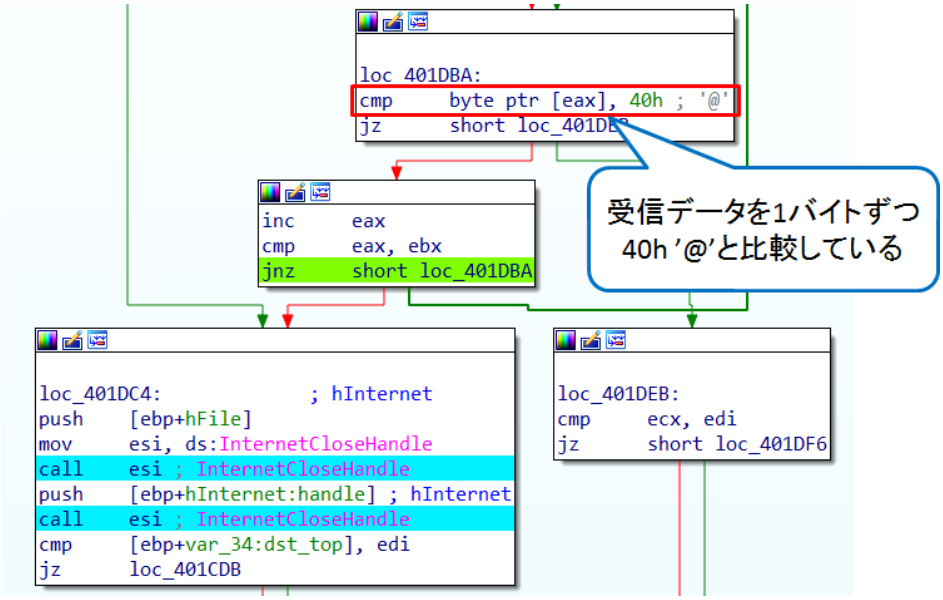


図 13 不正接続先からの受信データの確認

不正接続先から特定の応答が返ってこなければ次の動作に進まないようにすることで、閉鎖環境でのウイルス解析やサンドボックス等において、簡単には不正接続先 2 の情報が得られないようになっている。また、この次の処理は二次ウイルスのダウンロードと実行であるが、攻撃者が意図しない端末や、既に二次ウイルスの実行に成功している端末へはウイルスをダウンロードさせる必要はない（何度も通信していると監視により発覚しやすくなる可能性もある）ため、この応答電文によって制御しているといったことも考えられる。

3.5.3 不正接続先への通信 3

3 回目の通信では、暗号化されたデータがダウンロードされる。本ウイルスは、このデータを復号し、二次ウイルスを取り出してシステム内に保存した上で、それを実行する。本ウイルスの核となる機能である。

この通信でダウンロードされる二次ウイルスは、遠隔操作ウイルス（RAT）等であると思われる。そして、これにより、攻撃が次の段階（端末の乗っ取り、組織内ネットワークへの侵入等）へ進むものと考えられる。

(1) 不正接続先への通信とデータファイルのダウンロード

3 回目の通信は、表 4 の不正接続先 2 に対して行われる（図 14）。不正接続先からは暗号化されたデータが応答として返ってくるため、それを「%temp%」フォルダにダウンロード（保存）する。

この不正接続先 2 の URL は、末尾にある拡張子が「jpg」となっており、通信ログ等からは単に画像ファイルをダウンロードしているように見えるが、これは不正な通信であると見破られないようにするための細工である。このように、不正接続先との通信を正規の通信に見せかけるのは、多くのウイルスで見られる手口である。

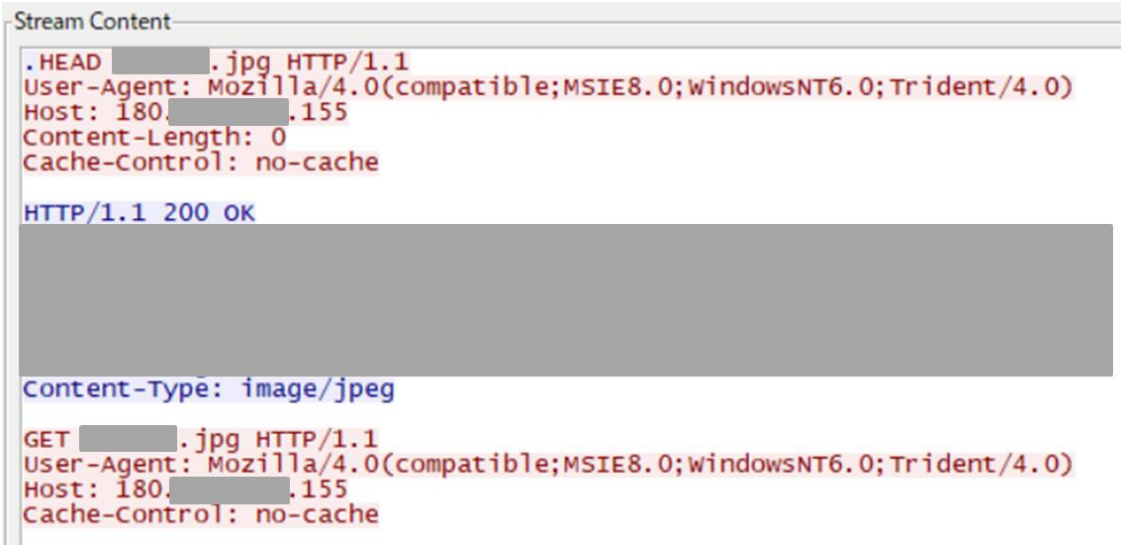
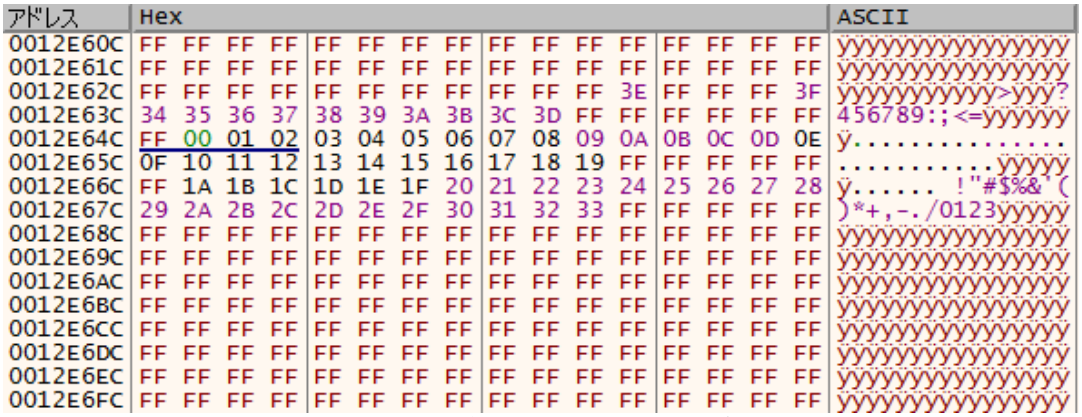


図 14　3 回目の通信の電文

(2) データファイルの復号と二次ウイルスの実行

本ウイルスは、ダウンロードしたデータファイルを二段階に分けて復号し、二次ウイルス本体を取り出す。

まず、復号の一段階目として、ウイルスは図 15 に示す 256 バイトのテーブルを用意し、ダウンロードしたデータファイルについて Base64 方式相当のデコードを行う。デコード対象とならない文字はスキップする仕組みとなっている。そして、その結果を別ファイル「%temp%¥temp1.dat」へ保存する。



アドレス	Hex	ASCII
0012E60C	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E61C	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E62C	FF FF FF FF FF FF FF FF FF FF FF 3E FF FF FF 3F	ÿÿÿÿÿÿÿÿÿÿÿ>ÿÿÿ?
0012E63C	34 35 36 37 38 39 3A 3B 3C 3D FF FF FF FF FF FF	456789:;<=ÿÿÿÿÿÿ
0012E64C	FF 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E	ÿ..............
0012E65C	0F 10 11 12 13 14 15 16 17 18 19 FF FF FF FF FF	...........ÿÿÿÿÿ
0012E66C	FF 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27 28	ÿ...... !"#$%&'(
0012E67C	29 2A 2B 2C 2D 2E 2F 30 31 32 33 FF FF FF FF FF	)*+,-./0123ÿÿÿÿÿ
0012E68C	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E69C	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E6AC	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E6BC	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E6CC	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E6DC	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E6EC	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
0012E6FC	FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF	ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ

図 15　デコードのためのテーブル

復号の二段階目は、数値「A9h」をキーとした XOR である。一段回目で復号したファイル（temp1.dat）に対し、ファイルの先頭 1 バイトを除きファイル全体へ XOR を行い、その結果を更に別のファイル「%temp%¥temp2.dat」へ保存する。

最後に temp2.dat から、二次ウイルス本体の作成方法、ファイルサイズ、ファイル名、二次ウイルスの本体データを取得し、これを基にシステム内に二次ウイルスのファイルを作成する。この際、中間処理のためのファイルとして「%temp%¥temp3.dat」が作成される場合がある。二次ウイルスの作成まで終わると、後始末として temp1.dat～temp3.dat のファイルを削除し、痕跡を消す。そして、作成した二次ウイルスを実行する。

データの復号に失敗した場合、0.1 秒間の sleep 処理を行い、ループの先頭に戻る。

本ウイルスは、特定の条件下でなければ二次ウイルスのダウンロード先である不正接続先 2 への通信を発生させないだけでなく、二次ウイルス自体も暗号化が施されている。通信経路上でウイルスが暗号化されていることは珍しくはないが、全体的に、攻撃者はセキュリティソフトやネットワーク監視による検知を回避するだけでなく、セキュリティベンダ等に二次ウイルスを入手され、解析されることを避けようとしているものと思われる。

3.5.4 不正接続先への通信 4

3 回目の通信によって得られた二次ウイルスの実行まで成功した場合、本ウイルスは表 4 の不正接続先 1 に対して 4 回目の通信を行う（図 16）。3 回目の通信でダウンロードしたデータの復号に失敗した場合は、本通信は発生しない。

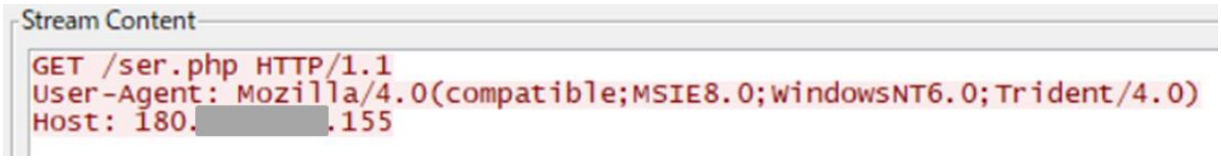


図 16 4 回目の通信の電文

4 回目の通信ではクエリストリングは付加せず、不正接続先からの受信データについても特段の処理は行っていない。

推測であるが、本通信は二次ウイルスが正しく復号・実行されたことを不正接続先（攻撃者）へ通知するための通信である可能性がある。

最後に、0.1 秒間の sleep 処理を行いループの先頭に戻る。すなわち、不正接続先への通信 2 から 4 までの 3 回の通信が 1 セットの処理として、ウイルスが動作中の間、繰り返されることになる。

4 おわりに

標的型攻撃に使用されるウイルスには、ファイルの実行時にウイルス感染に気付かせない細工がなされていることがある。本ウイルスのように、RLO 機能の悪用や、アイコンの偽装、表示用ダミーファイルを表示することで、利用者には一見して無害なファイルに見せかけるという手口もそのひとつである。

また、利用者による発見を回避する機能だけでなく、システム的なウイルス検知をかいくぐる機能も備わっている。特定のセキュリティソフトが起動している場合、それを停止するといった機能もこの一つである。さらに本ウイルスは、時間帯や特定の条件下でのみ不正接続先への通信処理を進めるといった機能があり、これも企業・組織のセキュリティ対策をかいくぐるための手口として備えられたものである可能性が高い。ウイルスの解析者に対して、挙動や不正接続先の特定を困難にするための仕掛けであるとも考えられる。

企業・組織のセキュリティを扱う部門等でインシデント対応等を行う際は、一例ではあるが、ウイルスには本書に記載したような動作や特徴があることを認識した上で、調査や対策を進めていただきたい。また、ウイルスの解析を行う部門等においては、不正接続先を特定する作業において、ウイルスが「特定の時間帯のみ不正通信を行う」、「通信先から特定の応答がない場合は動作を継続しない」といった動作をすることがありえるという点に留意いただきたい。

本書の内容が、企業や組織のセキュリティ対策の一助になれば幸いである。

以上