実証実験で使用したシステムの仕様

# 動的解析システムソースコード

平成 29 年 2 月 1 日

## 改訂履歴

| 版数 | 日付 | 変更理由 | 変更内容 | 変更者 |
|---|---|---|---|---|
| 1.0 | H29/2/1 | 新規作成 | 1.0 版作成 | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |

目次

# 1　はじめに

本書は、平成 28 年度の総務省施策である「スマートフォン上のアプリケーションにおける利用者情報の取扱いに係る実証調査研究の請負」の動的解析システムのソースコードである。

# 2　ソースコード

本章ではソースコードについて記す。

## 2.1　動的解析システム

動的解析システムのソースコード

```
#! /usr/bin/perl


### 動的解析システム ###
#
# 使い方
#   $ dyana.pl -i <デバイス情報ファイル> [ -r <結果出力ファイル> ] 対象ログファイル...¥n";
#
# パラメータ
#   デバイス情報ファイル: 端末のデバイスに関する情報ファイル名を指定します。
#   結果出力ファイル: 動的解析の結果を出力するファイル名を指定します。
#                     省略時は標準出力に解析結果を出力します。
#   対象ログファイル: 動的解析を行う対象のログファイル名を指定します。
#                     複数指定した場合は、解析結果の出力時に、個々の解析結果の前にファイル名を出力します。
# 概要
#   本システムは、対象ログファイル内で、デバイス情報ファイルに'検索用の値'として記載されたデータが
```

```
#   送信されていないかを検索します。
#   検索にあたっては、'検索用の値' と、英文字を全て小文字化した値、英文字を全て大文字化した値、
#   またそれらを、md5, sha1, sha256, sha512 でハッシュした値を用います。
#   対象ログファイル内で、これらの値のいずれかにマッチした場合、その値が送信された IP アドレスを
#   解析結果として出力します。(結果出力ファイルが指定されていた場合、該当ファイルに格納します)
#
# 動作条件など
# ・CentOS7 で動作確認をしています。
# ・Perl5 で動作確認しています。
# ・perl module として、Getopt, Encode, Digest が必要です。
# ・whois コマンドを呼び出しています。パスの通った場所に whois コマンドが必要です。
# ・解析対象のログファイルは、Burp Suite Free Edition の proxy 機能により取得したものを想定しています。
#
# デバイス情報ファイルについて
# ・送信情報として表示する一意の'結果表示用のタイトル'と、'検索用の値'の対をタブコードで区切り、
#   行単位にテキストデータで指定します。
# ・結果表示用のタイトルは、一意である必要があります。
# ・結果表示用のタイトルは、送信する情報が判断できる内容が良いでしょう。
#

use strict;
use Getopt::Std;
use Encode qw/decode encode/;
```

```
use Encode::Guess qw/cp932 utf8/;
use Digest::MD5 qw/md5_hex/;
use Digest::SHA qw/sha1_hex sha256_hex sha512_hex/;

my %opt;
getopts("i:r:" => ¥%opt);
usage() if(!defined($opt{i}) || $#ARGV <0);

my $lang = $ENV{LANG} =~ /ja_JP¥.UTF¥-{0,1}8/i ? 'UTF-8' : 'Shift_JIS';
binmode(STDOUT, ":utf8") if($lang eq 'UTF-8');
my $header_ = "送信情報¥t 送信先¥n";
my $header = decode(guess_encoding($header_)->name, $header_);

my %dev; # デバイス情報
my @inf; # '結果表示用のタイトル'が記述されている順番の記憶域（連想配列では順不同となるため）

# デバイス情報ファイルの読込み
open(INF, $opt{i}) || die "$opt{i}: $!";
my @lines = <INF>; # ファイル全体を対象に guess を実施する
my $enc = guess_encoding(join("", @lines), qw/utf8 cp932/);
foreach (@lines) {
    # 日本語を内部コードに変換して、タブ文字で分割する
    s/¥s+$//; # chomp
```

```
    my @p = split(/¥t/, (ref $enc) ? decode($enc->name, $_) : $_); # タブ区切り
    if($#p == 1){ # 位置情報以外
        # 検索データをハッシュと共に、連想配列に記憶する
        my @key = ($p[1], md5_hex($p[1]), sha1_hex($p[1]), sha256_hex($p[1]), sha512_hex($p[1]));
        if((my $lk = lc($p[1])) ne $p[1]){
            # 小文字も検索対象にする
            push(@key, $lk, md5_hex($lk), sha1_hex($lk), sha256_hex($lk), sha512_hex($lk));
        }
        if((my $uk = uc($p[1])) ne $p[1]){
            # 大文字も検索対象にする
            push(@key, $uk, md5_hex($uk), sha1_hex($uk), sha256_hex($uk), sha512_hex($uk));
        }
        $dev{$p[0]} = ¥@key;
        push(@inf, $p[0]);
    } elsif($#p == 2){ # 位置情報
        my @tmp = ($p[1], $p[2]);
        $dev{$p[0]} = ¥@tmp;
        push(@inf, $p[0]);
    }
}
close INF;

my $rfile = "";
```

```
my %org = ();
my $anum = @ARGV; # 解析ファイル数
my $acnt = 0;
foreach my $tfile (@ARGV) {
    open(IN, $tfile) || next;
    my $pfile = decode('utf8', $tfile);
    if(defined($opt{r}) && $rfile eq ""){
        $rfile = $opt{r};
        open(OUT, "> $rfile") || die "$rfile: $!";
        binmode(OUT, ":utf8") if($lang eq 'UTF-8');
        select OUT;
    }
    if($anum) { # ファイル名の表示
        ++$acnt;
        print STDERR "¥ranalyse: $acnt/$anum " if($rfile ne ""); # 進行表示
        if($anum > 1) {
            print "¥n" if($acnt > 1); # 2 ファイル目以降は改行後に表示する
            print "# $pfile¥n"; # ファイル名を表示する
        }
    }
    my $part = 0; # ログファイルの記述部分チェック用 (1=時刻+URL+IP アドレス, 2=通信内容)
    my %hit = (); # 検索対象がヒットした結果の格納域
    my $ip = "";  # 処理中相手 IP アドレスの格納域
```

```
my $tnum = 0; # 通信番号（通信の都度 1 カウントアップ）
while(<IN>) {
    s/¥s+$//; # chomp
    if(/^={54}$/) { # セパレータ行
        $part++;
    } else {
        if($part == 1){
            # 相手 IP アドレスを記録する
            my @d = split(/¥s+/);
            if($d[2] =~ /^¥[([0-9¥.]+)¥]$/){
                $ip = $1;
            } elsif($d[1] =~ /http*¥:¥/¥/([0-9¥.]+)[^¥d]/i) {
                # IP アドレス部が無い場合は URL から IP アドレスを取得する
                $ip = $1;
            } else {
                # IP アドレスが抽出できない場合はメッセージを出してスキップする
                print STDERR "no ip address ($tnum):  $_¥n";
                $ip = "";
                next;
            }
            $tnum++;
        } elsif($part == 2){
            # 送信データの検索処理
```

```
foreach my $k (keys %dev){
    # デバイス情報の全ての項目について実施する
    foreach my $s (@{$dev{$k}}){
        next if($ip eq ""); # IP アドレスを取得していない場合はスキップする
        # 登録されている値(ハッシュ含む)全てで検索を実施する
        if($_ =~ /¥Q$s¥E/){
            # マッチした場合の処理
            if(!defined($org{$ip})){
                # 相手 IP アドレスの情報で、whois による Organization 情報を取得していない場合、取得して記憶する
                if(open(WIS, "whois $ip | grep Organization |")){
                    my $w = <WIS>;
                    $w =~ /  ([^ ].+)¥s+$/;
                    $org{$ip} = $1;
                }
                close WIS;
                sleep 5; # whois を連続で送信しないためのウェイト
            }
            # マッチした結果をデバイス情報単位に、連想配列として記憶する
            my %tmp = defined($hit{$k}) ? %{$hit{$k}} : ();
            my $i = !defined($org{$ip}) ? "$ip" : "$ip($org{$ip})";
            $tmp{$i} = $tnum if(!defined($tmp{$i})); # 重複してなければ最初の通信番号を記憶する
            $hit{$k} = ¥%tmp;
        }
```

```
                }
            }
        } else {
            # 3回目のセパレータ時の処理（変数を初期化する）
            $part = 0;
            $ip = "";
        }
    }
}
close IN;

# 結果出力処理
my $hdflg = 0;
# デバイス情報ファイルに記載された順番に結果を出力する
foreach my $k (@inf) {
    # マッチした結果がある項目のみ処理する
    next if(! exists($hit{$k}));
    my %tmp = %{$hit{$k}} ;
    # 処理順に出力
    foreach my $i (sort {$tmp{$a} <=> $tmp{$b}} keys %tmp) {
        # 結果の出力
        print $header if(! $hdflg++); # ヘッダ表示
                                      # ヘッダはマッチした時にのみ、一度だけ表示する
```

```
            # 送信情報、タブ、送信先の IP アドレス(データが存在する場合は whois の結果も) を出力する
            print "$k¥t$i¥n";
        }
    }
}
print STDERR "done.¥n" if($anum && $rfile ne ""); # 進行状況の終了
exit 0;

# 使用方法
sub usage {
    $0 =~ /([^¥/]+)$/;
    print "usage: $1 -i info_file [-r result_file] target_file...¥n";
    exit 1;
}
```