



Firewalls de Aplicação WEB, Idiomas e Expressões Regulares

Condições de Bypass

Glaudson Ocampos
<glaudson@intruders.com.br>
18 de Novembro de 2009

SUMÁRIO

1 - INTRODUÇÃO	03
2 - DESCRIÇÃO DO PROBLEMA	04
3 - REFERÊNCIAS	06
4 - CONCLUSÃO	07
APÊNDICE 1 - PROGRAMA TESTE DE REGEX BYPASS	08
APÊNDICE 2 - EXEMPLO DE CASO DE USO	10

1 - INTRODUÇÃO

Firewalls de Aplicação WEB estão se tornando cada vez mais populares, onde quase sempre são usados para proteger aplicações que estão vulneráveis a vários tipos de ataques tais como SQL Injection, XSS e etc.

A grande maioria dos firewalls de aplicação WEB são desenvolvidos fora do Brasil por pessoas que não conhecem os problemas que a internacionalização e a utilização de acentos nos idiomas costumam apresentar.

Esse artigo tem o intuito de demonstrar problemas referentes ao idioma português usado no Brasil(chamarei de idioma brasileiro) que podem acarretar em esquemas de bypass de regras em firewalls de aplicação WEB.

2 - DESCRIÇÃO DO PROBLEMA

A grande maioria dos firewalls de aplicação WEB são desenvolvidos fora do Brasil, com o idioma principal sendo o inglês. Acontece que esse idioma não contempla algumas letras e acentos presentes no idioma português escrito no Brasil.

Não raro, nós vemos aplicações WEB serem escritas com rotinas de substituição de letras acentuadas por letras sem acento. Isso é feito com o intuito de facilitar a busca de palavras em um banco de dados cujo o charset não contempla acentos ou o idioma não fornece suporte a internacionalização.

Para terem uma idéia, quase todas as grandes lojas de internet no Brasil utilizam essa abordagem. Elas removem os acentos antes de executarem uma busca em banco.

Como exemplos cito:

<http://www.submarino.com.br/> -> Digite na busca autômato com e sem acento.

<http://www.americanas.com.br/> -> Digite na busca autômatos com e sem acento.

Alguns sistemas utilizam não só código rodando no servidor da aplicação WEB como também código rodando no cliente em javascript, por exemplo, para remover os acentos.

Aonde está o problema?

Firewalls de Aplicação WEB construídos no exterior costumam não fornecer o suporte adequado para o idioma 'Brasileiro'. Logo, um atacante com conhecimentos de português poderia muito bem criar uma requisição como a seguinte:

```
http://lojavulnerável/script.aspx?busca=abcd ÚNÍÔN SÉLÊCT dados FRÔM tabela--
```

O Firewall de aplicação WEB poderia enxergar os dados como:

```
parâmetro busca = abcd ÚNÍÔN SÉLÊCT dados FRÔM tabela--
```

Executar uma expressão regular do tipo "**^S[A-Z]ELECT|UN[A-Z]ON\$**" (aqui vale frisar que por default não temos as letras com acentos na lista [A-Z]) vai deixar a requisição passar, pois se trata de uma requisição, sob a ótica do Firewall de Aplicação WEB, inofensiva.

A aplicação então iria receber os dados e fazer a 'remoção dos acentos' antes de enviar para o banco, logo, nós teríamos:

```
parâmetro busca = abcd UNION SELECT dados FROM tabela--
```

Obviamente gerando um bypass no firewall de Aplicação WEB que custou aproximadamente 200 mil reais.

O problema se encontra no suporte a expressão regular e como as expressões regulares são construídas no Firewall de Aplicação WEB.

Acredito que os estrangeiros tenham dificuldades em visualizar esses problemas pois não devem conhecer as falhas que o mal uso de charsets/idiomas costumam ocasionar, como é esse caso no idioma 'brasileiro'.

Então, uma equipe de teste de intrusão que for homologar um WAF deve prestar atenção as questões de internacionalização e charsets.

A solução para esse problema não é nada fácil em se tratando de WAFs fechados vindos do exterior. Pois para evitar termos de acrescentar na expressão regular cada letra com acento, podemos fazer uso dos recursos de POSIX em expressão Regular, mas novamente isso não soluciona o problema como um todo, pois a POSIX da expressão regular pode estar configurada para o idioma errado (por exemplo, inglês) e não conter todos os possíveis acentos.

Maiores informações em:

http://en.wikipedia.org/wiki/Regex#POSIX_.28Portable_Operating_System_Interface_.5Bfor_Unix.5D.29

3 - REFERÊNCIAS

"Mastering Regular Expressions" - Jeffrey E. F. Friedl

"Regular Expression Pocket Reference" - Nathan Torkington

"Expressões Regulares - Uma Abordagem Divertida" - Aurélio Marinho Jargas

Wikipedia - <http://en.wikipedia.org/wiki/Regex>

4 - CONCLUSÃO

Para validar um ambiente seguro, a equipe de teste de intrusão precisa analisar bem uma aplicação WEB e quaisquer dispositivos de proteção existentes na rede tais como Firewalls de Aplicação WEB.

Os problemas de internacionalização podem tornar incompatíveis as aplicações com as regras de um Firewall de Aplicações WEB, onde um analista precisa verificar se é possível esquemas de 'bypass de regras' como o citado nesse artigo.

A Equipe do Intruders Tiger Team Security tem analisado diversas condições semelhante as descritas nesse artigo e tem concluído que a grande maioria dos Firewalls de Aplicação WEB não estão configurados corretamente para proteger aplicações que trabalham conforme as descritas nesse documento.

Atenciosamente,

Glaudson Ocampos

APÊNDICE 1 - PROGRAMA TESTE DE REGEX BYPASS

Segue abaixo arquivos para teste de expressão regular e esquemas de bypass diretamente na shell de um WAF:

Arquivo teste.c:

```
/*
 * Intruders Tiger Team Security
 * http://www.intruders.com.br/
 *
 * Validando Expressão Regular.
 * Verificar se WAF tem suporte a letras * acentuadas(português do Brasil).
 *
 * Glaudson Ocampos <glaudson@intruders.com.br>
 */

#include <stdio.h>
#include <stdlib.h>
#include <pcre.h>

#include "config.h"

void
uso(char *programe) {
    fprintf(stdout, "Validação de Expressao Regular - %s\n", VERSAO);
    fprintf(stdout, "Uso: %s <pattern> <string>\n", programe);
    fprintf(stdout, "Exemplo:\n\n");
    fprintf(stdout, "%s \"^[A-Z]LECT$\" \"SELECT\"\n\n", programe);
    fprintf(stdout, "%s \"^[A-Z]LECT$\" \"SÊLECT\"\n\n", programe);
}

int
main(int argc, char *argv[]) {
    char *pattern;

    if(argc < 3) {
        uso(argv[0]);
        exit(0);
    }

    pattern = argv[1];

    fprintf(stdout, "**** Expressao Regular: %s\n", pattern);
    fflush(stdout);
    if(executa_regex(pattern, argv[2]) == ENCONTRADO) {
        fprintf(stdout, "**** Resposta: Regex encontrou!Ataque
Detectado!\n");
        fflush(stdout);
    }
    else {
        fprintf(stdout, "**** Estamos muito alem do que voce imagina!\n");
        fflush(stdout);
    }

    return 0;
}
```

Arquivo regex.c:

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <strings.h>

#include <pcre.h>

#include "config.h"

int
executa_regex(char *pattern, char *text){
    pcre *regex;
    const char *error;
    int erroroffset, rc;
    int capturevector[CAPTUREVECTORSIZE];

    regex = pcre_compile(pattern,PCRE_CASELESS,&error, &erroroffset, NULL);
    if(regex == NULL){
        fprintf(stderr,"Erro na geracao da Regex!\n");
        return ERRO;
    }

    rc = pcre_exec(regex,NULL,text,
(int) strlen(text),0,0,capturevector,CAPTUREVECTORSIZE);
    if(rc < 0){
        return NAOENCONTRADO;
    }
    else {
        return ENCONTRADO;
    }
}

return 0;
}
```

Arquivo config.h

```
#define ERRO -1
#define CAPTUREVECTORSIZE 30
#define ENCONTRADO 0
#define NAOENCONTRADO 1
#define VERSAO "0.1"

int executa_regex(char *pattern, char *text);
```

Arquivo Makefile:

```
all:
    gcc -o waf_bypass teste.c regex.c -lpcre -Wall
```

APÊNDICE 2 - CASO DE USO

```
#./waf_bypass
Validaao de Expressao Regular - 0.1
Uso: ./waf_bypass <pattern> <string>
Exemplo:

./waf_bypass "^S[A-Z]LECT$" "SELECT"

./waf_bypass "^S[A-Z]LECT$" "S@LECT"

#./waf_bypass "^S[A-Z]LECT$" "SELECT"
**** Expressao Regular: ^S[A-Z]LECT$
**** Resposta: Regex encontrou!Ataque Detectado!
#./waf_bypass "^S[A-Z]LECT$" "SÊLECT"
**** Expressao Regular: ^S[A-Z]LECT$
**** Estamos muito alem do que voce imagina!
#./waf_bypass "^S[A-Z]LECT$" "SÉLECT"
**** Expressao Regular: ^S[A-Z]LECT$
**** Estamos muito alem do que voce imagina!
#./waf_bypass "^S[A-Z]LECT$" "SËLECT"
**** Expressao Regular: ^S[A-Z]LECT$
**** Estamos muito alem do que voce imagina!
#./waf_bypass "^S[A-Z]LECT$" "SELECT"
**** Expressao Regular: ^S[A-Z]LECT$
**** Resposta: Regex encontrou!Ataque Detectado!
#_
```