Aspectos Práticos da Codificação Segura.
- Parte 1 -

André Ricardo Abed Grégio[1], Luiz Gustavo C. Barbato[1 2],
Luiz Otávio Duarte[1 2], Antonio Montes[1 2]

[1]Laboratório Associado de Computação e Matemática Aplicada
Instituto Nacional de Pesquisas Espaciais

[2]Divisão de Segurança de Sistemas de Informação
Centro de Pesquisas Renato Archer

GTS 02.05

Terminologia

```
$ alias problema="o que será resolvido"
```

- **Problema:** Codificação Insegura

```
$ ln -s defeito bug
```

- **Defeito**: Imperfeição existente no software, em estado latente, que, ao ser ativado, produz um resultado inesperado
- **Erro**: Manifestação do **defeito**, que pode levar a uma **falha**
- **Falha**: Inabilidade do software em cumprir um requisito operacional a ele atribuído

```
$ export VULNERABILIDADE=${VULNERABILIDADE}" de software"
```

- **Vulnerabilidade**: Manifstação do **defeito** que deixa o software exposto a uma ação maliciosa

Estatísticas

- **Dados do NVD/NIST** mostram que no **ano de 2005** uma grande porcentagem dos problemas de segurança encontrados em sistemas é devida a **má codificação** dos mesmos.
- Do total de vulnerabilidades encontradas, pelo menos **74%** eram decorrentes de **erros de codificação**.
- Do início do ano **até agosto**, mais de *50%* dos erros eram devidos a má **validação de entrada** envolvendo **extravazamento de *buffers*.**

Motivação

- As estatísticas mostram que apesar de conhecido o problema ele não está sendo solucionado
- Pequenas distrações na codificação geram grandes conseqüências
- Muitos acham que código por si só não pode ser seguro
- Os erros são sempre os mesmos independentemente do porte da empresa de software
- O principal problema é a codificação de software e a solução pode ser a Codificação Segura

Objetivo

Apresentar e discutir técnicas de codificação segura baseando-se em exemplos práticos de erros comumente cometidos nos sistemas de código aberto disponibilizados na Internet.

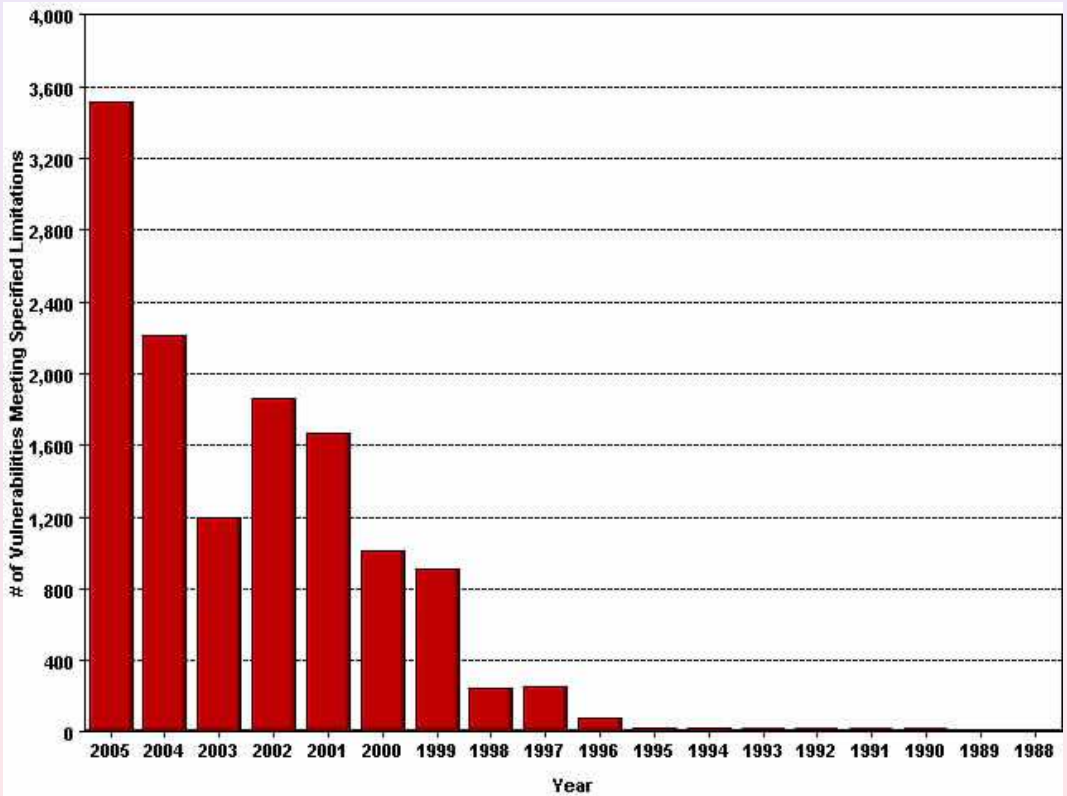
Codificação vs. Programação

- Várias literaturas tratam de outras etapas do ciclo de desenvolvimento de software.
- Ciclo básico de desenvolvimento de software
  - Análise de requisitos, Projeto do sistema, Modelagem do sistema, **Codificação** e Testes
- Como o objetivo deste curso é a etapa de escrever código foi adotado o termo codificação

Roteiro do Microcurso

- Vulnerabilidades de Software
- Codificação Segura
- Wrappers
- Estudos de Casos
- Considerações Finais

Estatísticas



Fonte: `http://nvd.nist.gov`

Vulnerabilidades mais encontradas

- **Estatísticas levantadas no dia: 02/11/2005**
- OWASP - Open Web Application Security Project
  - `http://www.owasp.org`
- OSVDB - Open Source Vulnerability Database
  - `http://www.osvdb.org`
- NVD - National Vulnerability Database
  - `http://nvd.nist.gov`

OWASP

1	Entrada não Validada
2	Controle de Acesso Falho
3	Autenticação e Gerenciamento de Sessão Não Confiável
4	Problemas de Cross-Site Scripting (XSS)
5	Extravazamento de Buffers
6	Defeitos na Passagem de Parâmetros
7	Tratamento Impróprio de Erros
8	Armazenamento Inseguro de Informações
9	Negação de Serviço
10	Gerenciamento Inseguro de Configurações

Fonte: `http://www.owasp.org/documentation/topten.html`

OSVDB

5490	Manipulação de entrada
1350	Vazamento de informação
1260	Negação de serviço
450	Autenticação
405	Má configuração
315	Desconhecidos
225	Condição de corrida
180	Outros
135	Criptografia
90	Sequestro de sessão
45	Infraestrutura

Fonte: `http://www.osvdb.org`

NVD

6809	Validação de entrada
2913	Projeto
1351	Condição de exceção
1216	Validação de acesso
650	Configuração
217	Outros
215	Condição de corrida
176	Ambiental

Fonte: `http://nvd.nist.gov`

A Grande Vilã: Entrada de Dados

- OWASP: Vulnerabilidade mais crítica
- OSVDB: 5490 entradas de 19845
- NVD: 6809 entradas de 12911

Entrada de Dados: Onde Estão?

- Entrada padrão
- Parâmentros de inicialização
- Variáveis de ambiente
- Sockets
- Arquivos
- Objetos compartilhados
- Pipes
- Demais IPC
- Etc ...

Foco principal deste microcurso

- Como a grande vilã é a entrada de dados, esta primeira parte do curso será destinada a tentar solucionar este problema: VALIDAÇÃO
- Vulnerabilidades decorrentes da má validação de entrada serão discutidas assim como formas de evitá-las

Vulnerabilidades a serem discutidas

- Format String
- Injections
  - OS Command
  - SQL
- Cross Site Scripting (XSS)
- Buffer Overflow
  - Stack
  - Heap

Format String

- O recurso do *format string* foi desenvolvido para resolver o problema de formatação de caracteres
- Muitos programas com saídas textuais precisam de formas de apresentar seus resultados, como casas decimais em números, posições e espaçamento dos caracteres, etc.
- Utilizar formatadores para extrair informações, driblar checagens de tamanho de buffers, etc.

Format String

```
1  int main (int argc, char ** argv) {
2
3     double num = 456.0 / 123.0;
4
5     printf ("%f\n", num);
6
7     return 0;
8
9  }
```

Format String

```
1 $ ./fs
2 3.707317
```

Format String

```
1  int main (int argc, char ** argv) {
2
3     double num = 456.0 / 123.0;
4
5     printf ("%6.2f\n", num);
6
7     return 0;
8
9  }
```

Format String

```
1 $ ./fs2
2   3.71
```

Format String

```
1  int main (int argc, char ** argv) {
2
3     char senha [] = "ABCD";
4
5     printf (argv[1]);
6     printf ("\n");
7
8     return 0;
9  }
```

Format String

```
1 $ ./fs3 "%x"
2 bfac17f0
```

Format String

```
1 $ ./fs3 "%x %x %x %x"
2 bf826b20 bf826b08 80483eb 44434241
```

Format String

```
1  #define TAM_BUF 10
2  int main (int argc, char ** argv) {
3
4     char buf [TAM_BUF+1];
5     if (strlen(argv[1])<=TAM_BUF) {
6        sprintf (buf, argv[1]);
7        printf ("%s\n", buf);
8     } else {
9        printf ("Valor maior que %d bytes\n", TAM_BUF);
10    }
11
12    return 0;
13
14 }
```

Format String

```
1 $ ./fs4 "0123456789"
2 0123456789
```

Format String

```
1 $ ./fs4 "0123456789a"
2 Valor maior que 10 bytes
```

Format String

```
1 $ ./fs4 "%30s"
2                          c§¿itä·
3 Segmentation fault
```

Format String - Dicas para evitar

- Nunca passe diretamente a entrada do usuário para as funções de formatação (validação)
  - Valide o conteúdo do buffer
- Sempre utilize os formatadores de string
  - Utilize os formatadores nas funções de formatação

Definição

- Vulnerabilidade que permitem a execução de comandos através de uma aplicação que geralmente é Web.
  - OS Command
  - SQL

SQL

- SQL - Structured Query Language
- Linguagem de manipulação de banco de dados
  - Permite não somente consultas como também manipulações dos dados e até mesmo gerenciamento do banco
  - Em determinados sistemas gerenciadores de banco de dados é possível executar comandos do sistema operacional

SQL Injection

- Enviar comandos SQL para uma aplicação de forma a fazer com que ela execute operações ilícitas
- A injeção mais comum em aplicações, principalmente as voltadas à Web

SQL Injection

```
var sql = "SELECT * FROM usuarios WHERE login = '" + formusr\
          + "' AND password = '" + formpwd + "'";
```

SQL Injection

```
SELECT * FROM usuarios WHERE login = aluno AND password = \
aluno123;
```

SQL Injection

Se no lugar do nome de usuário no formulário for inserido, por exemplo, *' or 1=1 - -*, a senha que será utilizada não fará menor importância no processo de autenticação. Esta entrada de dado resulta no seguinte comando SQL:

```
SELECT * FROM usuario WHERE username = ' ' or 1=1 -- AND
                            password = 'naoimporta';
```

SQL Injection

```
SELECT * FROM usuario WHERE 1=1;
```

SQL Injection

O perigo das injeções SQL é a possibilidade de subverter um banco de dados de tal forma que um atacante possa, entre outras coisas:

- inserir dados;
- remover dados/tabelas;
- recuperar informações sensíveis não só do banco, mas do sistema.

SQL Injection - Caracteres utilizados

’ ou ”	Indicadores de *string*
- - ou #	Comentário em uma única linha
+	Adição, concatenação ou espaço em URL
=	Operador de igualdade
<> ou !=	Operadores de diferença
>	Operador “maior que”
<	Operador “menor que”
( )	Expressão ou delimitador de hierarquia
@	Variáveis
;	Execução em sequência

SQL Injection

```
1 CREATE TABLE usuarios (lista_usuarios blob);
2 LOAD DATA INFILE '/etc/passwd' INTO TABLE usuarios;
3 SELECT * FROM usuarios;
```

OS Command

- Enviar comandos de sistema operacional para uma aplicação de forma a fazer com que ela execute operações ilícitas
- Estes comandos podem ser injetados através dos campos visíveis ou ocultos de um formulário junto com os valores de entrada

OS Command

O programa recebe o nome da pessoa, e executa o comando *grep* para procurar o nome dentro de um arquivo de telefones.

```
1  int main (int argc, char ** argv) {
2  {
3     char cmd[128], nome[256];
4     gets(nome);
5     sprintf(cmd,"grep %s telefones.txt\n",nome);
6     write(1,"Content-Type: text/plain\n\n",27);
7
8     system(cmd);
9
10    return 0;
11 }
```

OS Command

```
$ echo 'name=xxx telefone ; id ; cat' | lynx -post_data \
http://127.0.0.1/cgi-bin/phone.cgi
```

Demais Injections

- O princípio das demais injeções é o mesmo
- O fator variante é a tecnologia
- Há várias outras injeções como *LDAP injection*, *XPath injection*, *PHP injection*, etc.

`http://www.webappsec.org/projects/threat/classes_of_attack.shtml`

Injections - Dicas para evitar

- Valide a entrada de dados
  - Nunca procure por caracteres indesejáveis
  - Aceite somente os caracteres que forem válidos para a entrada
- Em aplicações em rede valide sempre o dado na aplicação servidora
  - A validação feita do lado do servidor é mais confiável
- Nunca confie na entrada de dados
  - Valide a entrada mesmo que a origem seja teoricamente confiável
- Trate os campos de entrada como literais
  - Depois de validar a entrada insira caracteres que o represente como literal

O que é?

- XSS ou CSS
- Um usuário mal intencionado pode enviar determinadas informações para sites Web com páginas dinâmicas, que quando forem acessadas por uma aplicação cliente, como o navegador Web, este executa operações ilícitas.

Cross Site Scripting

```
1  #!/usr/bin/perl
2
3  $usuario=<STDIN>;
4
5  chomp($usuario);
6
7  if ($usuario=~/usuario=(.*)/) {
8     $usuario = $1;
9  }
10
11 #http://www.perlfect.com/articles/url_decoding.shtml
12 $usuario =~ s/%([\dA-Fa-f]{2})/pack("C", hex($1))/eg;
13
14 printf ("Content-Type: text/html\n\n");
15 printf ("Seja Bem Vindo: %s\n", $usuario);
```

Cross Site Scripting

- Um teste que pode ser feito para verificar a existência de vulnerabilidade é enviar algum código HTML ao invés do nome do usuário, como por exemplo:

```
<script>alert('Teste')</script>
```

- Se o navegador apresentar uma janela do tipo *pop-up* com a mensagem “Teste”, o CGI está vulnerável a ataques XSS.

Cross Site Scripting - Dicas para evitar

- Valide a entrada de dados
  - Nunca procure por caracteres indesejáveis
  - Aceite somente os caracteres que forem válidos para a entrada
- Em aplicações em rede valide sempre o dado na aplicação servidora
  - A validação feita do lado do servidor é mais confiável
- Nunca confie na entrada de dados
  - Valide a entrada mesmo que a origem seja teoricamente confiável
- Trate os campos de entrada como literais
  - Depois de validar a entrada insira caracteres que o represente como literal

Buffer Overflow

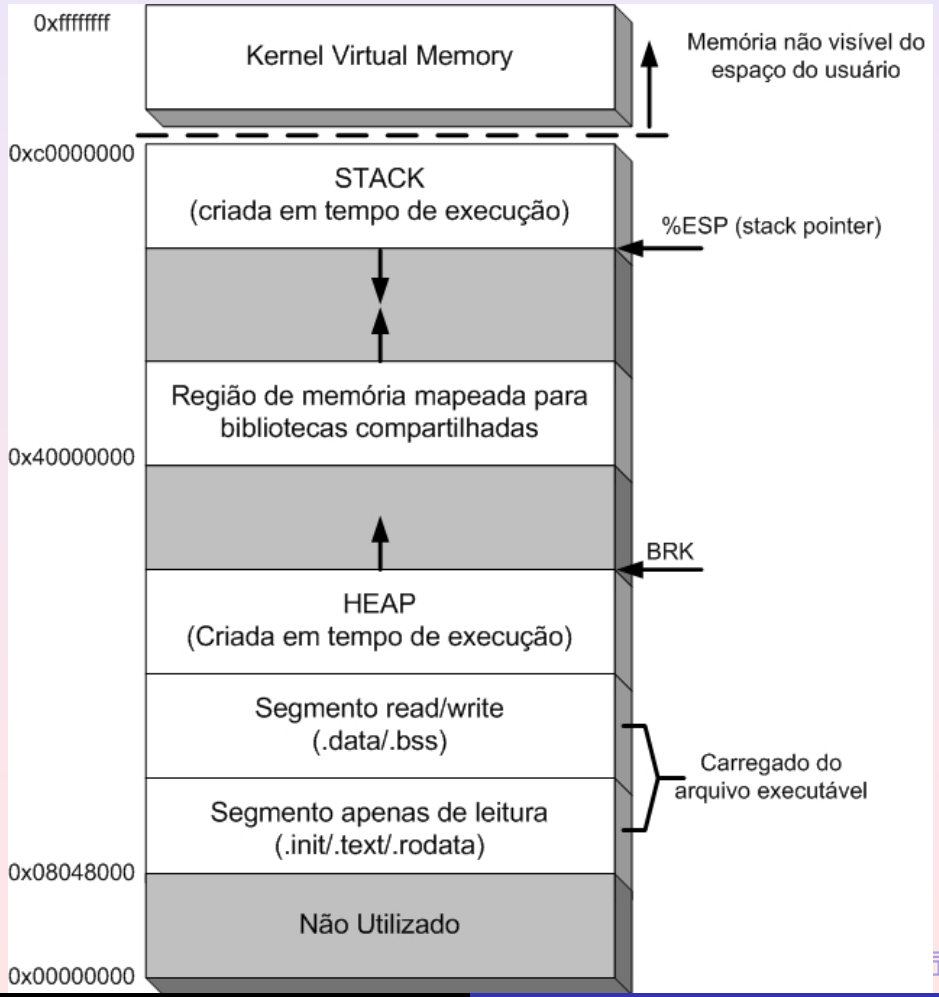
O que é?

- Buffers são espaços contiguamente alocados de memória que suportam inúmeras instâncias de um mesmo tipo de dado.
- Um buffer é extravazado se:
  - Mais dados do que o suportado tentam são armazenados;
  - Problemas na manipulação de ponteiros ou índices.

Buffer Overflow

- Pode ser dividido segundo a região onde ocorre dentro do espaço de endereçamento do processo.
  - *Stack Overflow*;
  - *Heap Overflow*;

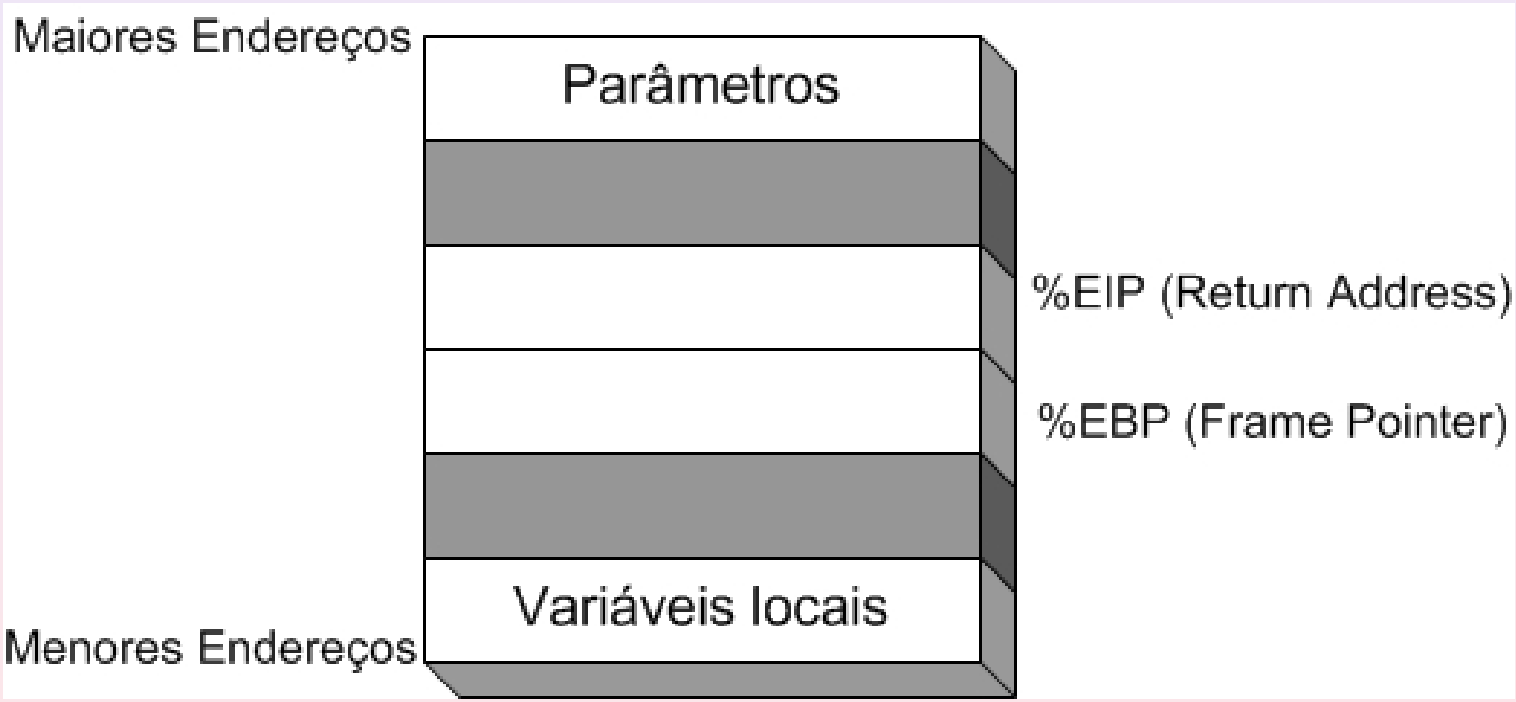
PAS


0xffffffff
Kernel Virtual Memory
Memória não visível do espaço do usuário
0xc0000000
STACK
(criada em tempo de execução)
%ESP (stack pointer)
Região de memória mapeada para bibliotecas compartilhadas
0x40000000
BRK
HEAP
(Criada em tempo de execução)
Segmento read/write
(.data/.bss)
Carregado do arquivo executável
Segmento apenas de leitura
(.init/.text/.rodata)
0x08048000
Não Utilizado
0x00000000

Stack

- É utilizada pra armazenar variáveis locais, parâmetros da função e dados necessários para recuperar os *stack frames* anteriores;
- É um bloco de memória contiguamente alocado, utilizando uma estrutura de pilha;
- Cada um dos elementos desta pilha é denominado *stack frame*;
- *Stack frames* são inseridos quando uma função é chamada e removido quando ela retorna.

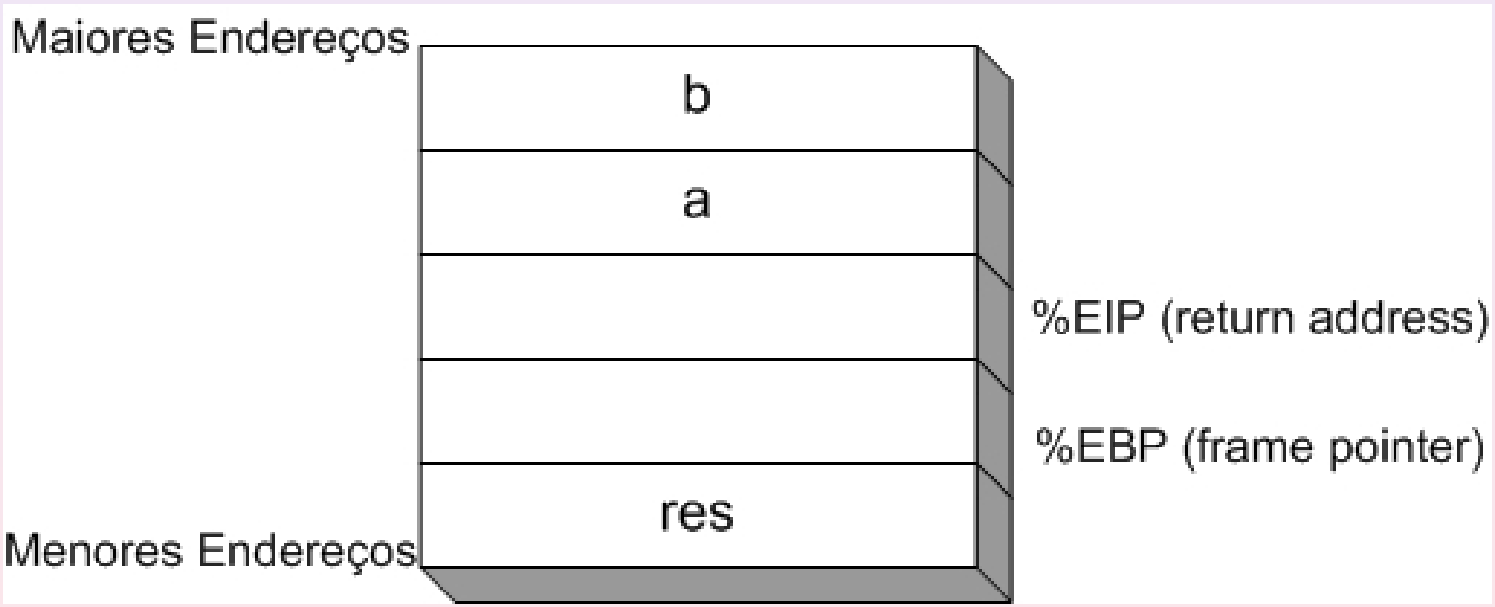
Stack


Maiores Endereços
Parâmetros
%EIP (Return Address)
%EBP (Frame Pointer)
Variáveis locais
Menores Endereços

Stack

```
1  int somar (int a, int b) {
2
3     int res = a + b;
4
5     return res;
6
7  }
8
9  int main (int argc, char ** argv) {
10
11    int resultado = somar (1,2);
12
13    printf ("Resultado = %d\n", resultado);
14
15    return 0;
16
17 }
```

Stack


Maiores Endereços
b
a
%EIP (return address)
%EBP (frame pointer)
res
Menores Endereços

Stack Overflow

- Utiliza um extravazamento de buffer para desviar o fluxo de execução de um programa;
- Manipulação do endereço de retorno (EIP).

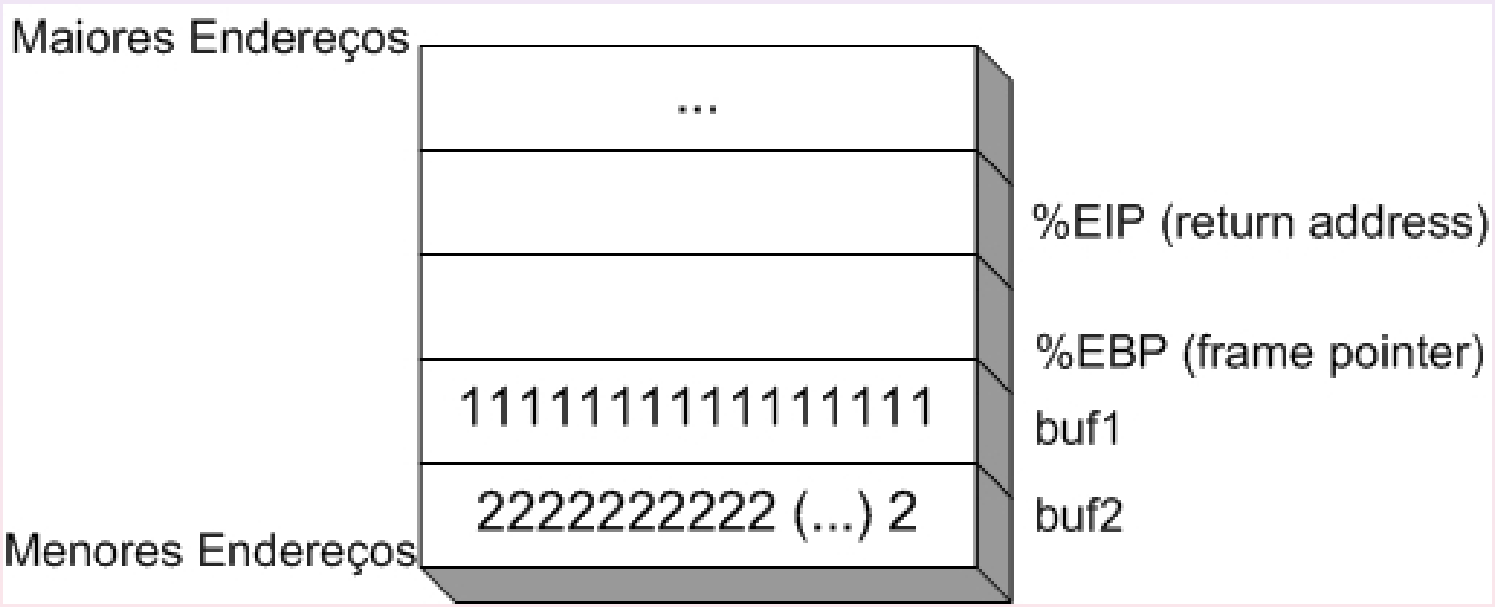
Stack Overflow

```
1  int main (int argc, char ** argv) {
2
3    char buf1 [16];
4    char buf2 [32];
5
6    printf ("stack\n");
7
8    memset (buf1, '1', 16);
9    memset (buf2, '2', 32);
10
11   printf ("buf1 = %s\n", buf1);
12   printf ("buf2 = %s\n", buf2);
13
14   return 0;
15
16 }
```

Stack Overflow

```
1 $ ./stack
2 stack
3 buf1 = 1111111111111111,
4 buf2 = 22222222222222222222222222222211111111111111111,
```

Stack Overflow



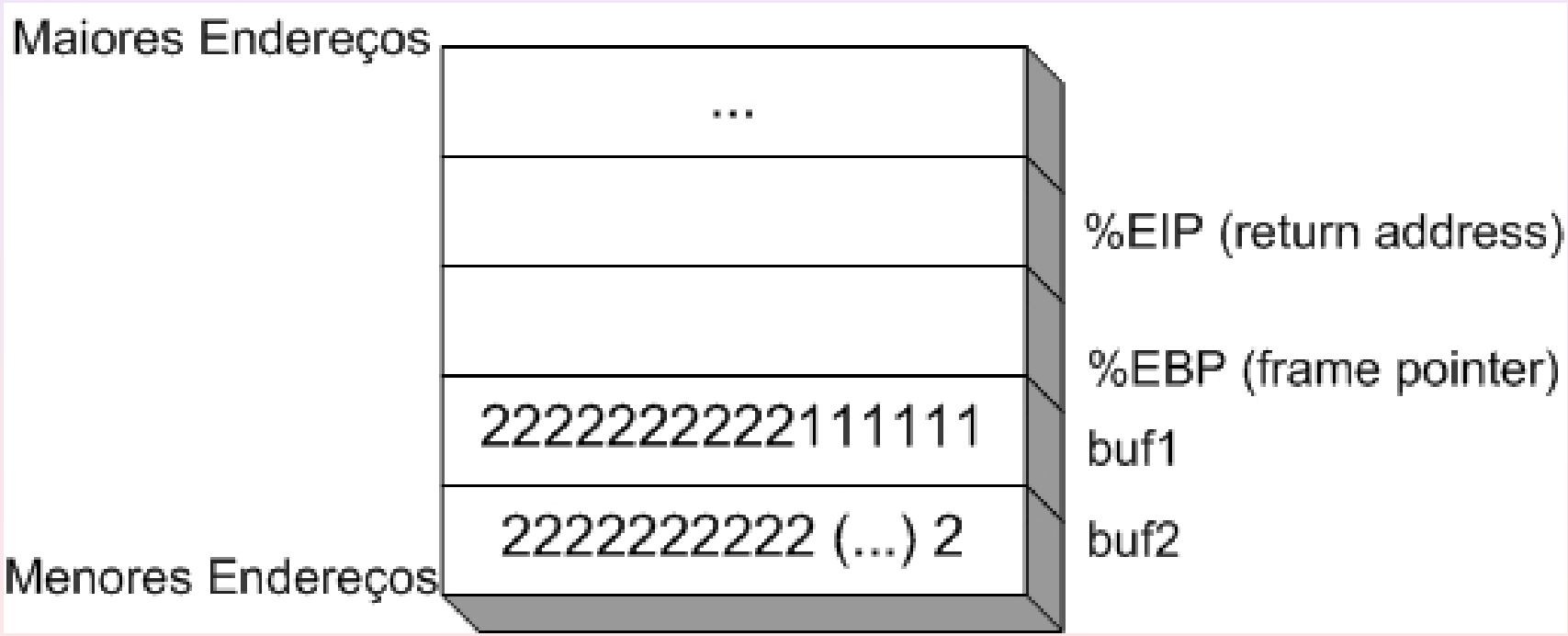
Stack Overflow

```
1  int main (int argc, char ** argv) {
2
3     char buf1 [16];
4     char buf2 [32];
5
6     printf ("stack2\n");
7
8     memset (buf1, '1', 16);
9     memset (buf2, '2', 32+atoi(argv[1]));
10
11    printf ("buf1 = %s\n", buf1);
12    printf ("buf2 = %s\n", buf2);
13
14    return 0;
15
16 }
```

Stack Overflow

```
1 $ ./stack2 10
2 stack2
3 buf1 = 222222222111111,
4 buf2 = 22222222222222222222222222222222222222111111,
```

Stack Overflow



Heap

- É uma região na memória de um programa destinada a utilização de variáveis **dinamicamente alocadas**;
- A área ocupada pelas variáveis dinamicamente alocadas não é conhecida em tempo de compilação;
- Este espaço é alocado em tempo de execução através de chamadas de sistema.

Heap

- Ponteiros

```
1  /* Ponteiro para "caracteres" */
2  char   * ptr_char;
3
4  /* Ponteiro para "inteiros" */
5  int    * ptr_int;
6
7  /* Ponteiro para "pontos flutuantes" */
8  float * ptr_float;
9
10 /* Ponteiro void */
11 void * ptr_void;
```

Heap

- Alocação Dinâmica de Memória

```
1 void *malloc(size_t size);
2 void *calloc(size_t nmemb, size_t size);
3 void *realloc(void *ptr, size_t size);
```

```
1 void free(void *ptr);
```

Heap Overflow

- Utilizar de um extravazamento de buffer para manipular o conteúdo de outros buffers
- A exploração depende muito de versões de sistemas operacionais e bibliotecas
- Em muitos casos são antecedidos por um Interger Overflow

Heap Overflow

```
1  int main (int argc, char ** argv) {
2     FILE * fd;
3     char * buf = (char *) malloc (8);
4     char * arquivo = (char *) malloc (16);
5
6     strcpy (arquivo, "/tmp/arquivo");
7
8     gets (buf);
9     printf ("Abrindo o arquivo: %s\n", arquivo);
10
11    fd = fopen (arquivo, "w");
12    fputs (buf, fd);
13
14    fclose(fd);
15    return 0;
16 }
```

Heap Overflow

```
1  $ echo "teste heap" | ./heap
2  Abrindo o arquivo: /tmp/arquivo
3
4  $ cat /tmp/arquivo
5  teste heap
```

Heap Overflow

```
1 $ perl -e 'print "A"x15' | ./heap
2 Abrindo o arquivo: /tmp/arquivo
```

- Mesmo que *buf* tenha o tamanho de 8 bytes, com uma entrada de 15 caracteres, o programa não apresenta problemas;

Heap Overflow

```
1 $ perl -e 'print "A"x16' | ./heap
2 Abrindo o arquivo:
3 Segmentation fault
```

- Com 16 caracteres uma falha de segmentação é gerada e o nome do arquivo é apagado;

Heap Overflow

```
1 $ perl -e 'print "A"x17' | ./heap
2 Abrindo o arquivo: A
```

- E apartir de 17 caracteres a variável *arquivo*, que armazena no nome do arquivo `/tmp/arquivo`, começa a ser sobrescrita. Então, os primeiros 16 bytes podem ser compostos de qualquer caracter.

```
1 $ perl -e 'print "A"x20' | ./heap
2 Abrindo o arquivo: AAAA
```

Buffer Overflow - Dicas para evitar

- Faça a checagem cuidadosa dos acessos aos *buffers* (validação).
  - Tanto de tamanho quanto de conteúdo;
- Evite a utilização de funções inseguras.
  - Funções que não permitem a checagem de limites;
  - Funções que não colocam o terminador de string.
- Sempre use o terminador de string.
  - Aloque sempre uma posição a mais e nunca a utilize;
  - Procure “zerar” o *buffer* antes de utilizar;
  - Force a inserção manualmente sempre que o *buffer* for alterado.

O que é Codificação Segura?

É escrever códigos não vulneráveis.

Expressões Regulares

- A expressão casa qualquer entrada que case com pelo menos um dos termos

```
"A|B|C" casa "A", "B", "C", "xABt", "Abcd"
  e não casa "abcd", "za%c"
```

Pseudo-Gramática

```
ER : TERMO '|' ER
   | TERMO
   ;
```

Pseudo-Gramática

```
TERMO : TERMO TERMO
      | '(' TERMO ')'
      | '^' TERMO
      | TERMO '$'
      | TERMO '+'
      | TERMO '*'
      | TERMO '?'
      | TERMO '{' %num ',' %num '}'
      | ATOMO
      ;
```

Pseudo-Gramática

```
ATOMO : '[' INTERV ']'
      | %char
      ;

INTERV : INTERV INTERV
       | %char - %char
       ;
```

Caracteres Especiais

- Alguns caracteres possuem significados especiais:

```
. ^ $ [ ] ( ) { } | * + ? \
```

- Portanto, quando forem utilizados com o significado original devem ser precedidos de:

```
\
```

- Nem todas as implementações conseguem tratar todos os caracteres especiais como ordinários
- O caracter “.” casa qualquer um

Validação de Endereços IP

- A seguir será apresentado um exemplo de utilização de expressões regulares para validar um endereço IP;
- A validação será feita em 10 tentativas;
- Cada tentativa possuirá 3 slides, um para explicar o que está sendo feito, outro para mostrar a ER e o último para mostrar os resultados.

Validação de Endereços IP

Primeira tentativa de validação:

Nesta primeira tentantiva, espera-se que a entrada seja quatro grupos de números separados por pontos.

Validação de Endereços IP

Primeira tentativa de validação:

```
[0-9]*.[0-9]*.[0-9]*.[0-9]*
```

Validação de Endereços IP

Primeira tentativa de validação:

- 192.168.10.1
- 1.1.1.1
- ..2.
- 199999.199999.19999.19999
- foobar
- foobar foobar

Validação de Endereços IP

Segunda tentativa de validação:

Na segunda tentativa o caractere “+” será utilizado ao invés do “” para evitar que grupos nulos de números sejam utilizados.*

Validação de Endereços IP

Segunda tentativa de validação:

```
[0-9]+.[0-9]+.[0-9]+.[0-9]+
```

Validação de Endereços IP

Segunda tentativa de validação:

- 192.168.10.1
- 1.1.1.1
- 199999999.0.0.0
- 4f0o3A4
- foobar1.1.1.1foobar
- foo1a1b1c1bar

Validação de Endereços IP

Terceira tentativa de validação:

Na terceira tentativa será utilizada a barra invertida antes dos pontos para evitar que qualquer caractere seja utilizado.

Validação de Endereços IP

Terceira tentativa de validação:

```
[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+
```

Validação de Endereços IP

Terceira tentativa de validação:

- 192.168.10.2
- 1.1.1.1
- 111111.11112222.1111122223334445555.999999999
- aaaaa0.0.0.0
- foo1.1.1.1bar

Validação de Endereços IP

Quarta tentativa de validação:

Na quarta tentativa será especificada a quantidade mínima e máxima de números em cada grupo ao invés de utilizar o caractere “+”.

Validação de Endereços IP

Quarta tentativa de validação:

```
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
```

Validação de Endereços IP

Quarta tentativa de validação:

- 192.168.10.1
- 999.999.999.999
- 300.300.300.300
- foo1.1.1.1bar
- foobar192.168.10.1

Validação de Endereços IP

Quinta tentativa de validação:

Nesta tentativa será limitado o início e fim da expressão regular tetando evitar que caracteres antes de depois da expressão regular sejam utilizados.

Validação de Endereços IP

Quinta tentativa de validação:

```
^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$
```

Validação de Endereços IP

Quinta tentativa de validação:

- 192.168.10.2
- 1.1.1.1
- 260.260.256.10
- 192.168.10.300
- 999.999.999.999

Validação de Endereços IP

Sexta tentativa de validação:

Nesta tentativa, o número de cada octeto pode variar de 0 até 199.

Validação de Endereços IP

Sexta tentativa de validação:

```
^[01]?[0-9]{1,2}\.[01]?[0-9]{1,2}\.
 [01]?[0-9]{1,2}\.[01]?[0-9]{1,2}$
```

Validação de Endereços IP

Sexta tentativa de validação:

- 192.168.10.1
- 1.1.1.1

Validação de Endereços IP

Sétima tentativa de validação:

Nesta tentativa, a expressão regular é desenvolvida para suportar endereços IPv4 no qual o número de cada um dos octetos seja maior ou igual a 200.

Validação de Endereços IP

Sétima tentativa de validação:

```
^2[0-9]{1,2}\.2[0-9]{1,2}\.2[0-9]{1,2}\.2[0-9]{1,2}$
```

Validação de Endereços IP

Sétima tentativa de validação:

- 200.200.201.201
- 213.213.213.213
- 260.250.210.211
- 299.299.299.299

Validação de Endereços IP

Oitava tentativa de validação:

Na oitava tentativa a expressão regular que será apresentada visa limitar o número máximo que pode ser utilizado em cada octeto: 255.

Validação de Endereços IP

Oitava tentativa de validação:

```
^2([0-4][0-9]|5[0-5])\.2([0-4][0-9]|5[0-5])\.
 2([0-4][0-9]|5[0-5])\.2([0-4][0-9]|5[0-5])$
```

Validação de Endereços IP

Oitava tentativa de validação:

- 200.201.202.203
- 255.255.255.255
- 200.200.200.200

Validação de Endereços IP

Nona tentativa de validação:

Nesta tentativa, somente os endereços IPv4 realmente válidos são aceitos como válidos. Endereços outros não são aceitos.

Validação de Endereços IP

Nona tentativa de validação:

```
^([01]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))\.
 ([01]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))\.
 ([01]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))\.
 ([01]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))$
```

Validação de Endereços IP

Décima tentativa de validação:

Nesta última tentativa, os três primeiros decimais seguidos de pontos são agrupados.

Validação de Endereços IP

Décima tentativa de validação:

```
^(([01]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))\.){3}
 ([01]?[0-9]{1,2}|2([0-4][0-9]|5[0-5]))$
```

Expressão Regular em Várias Linguagens

- Perl
- Shell Script
- C
- Phyton
- PHP
- C#
- ASP.NET
- Java

Exemplo de ER em Perl

```
1  #!/usr/bin/perl -Tw
2
3  use strict;
4
5  $ARGV[0] =~ "^[0-9]+\$" || print("FAIL\\n");
```

Exemplo de ER em Shell Script

```
1  #!/bin/sh
2
3  echo -n $1 | egrep "^[0-9]+$" || echo "FAIL"
4
5  if [ -z ‘echo -n $1 | grep -E "^[0-9]+$"‘ ]
6  then
7          echo "FAIL"
8  fi
```

Exemplo de ER em C

```
1  #include <stdio.h>
2  #include <regex.h>
3  int main(int ARGC, char *ARGV[]){
4      regex_t preg;
5      if(regcomp(&preg,"^[0-9]+$",REG_EXTENDED)) return(-1);
6      if(regexec(&preg,ARGV[1],(size_t) 0,NULL,0))
7      {
8          regfree(&preg);
9          printf("FAIL");
10     }
11     regfree(&preg);
12     return(0);
13 }
```

Exemplo de ER em Python

```
1  #! /usr/bin/env python
2  import re
3  import sys
4  p = re.compile("^[0-9]+$")
5  m = p.match( sys.argv[1] )
6  if m:
7      print m.group()
8  *else:
9      print 'FAIL'
```

Exemplo de ER em PHP

```
1  <?php
2     if (ereg ("^[0-9]+$", "1233")) {
3
4     } else {
5           echo "FAIL";
6     }
7  ?>
```

Exemplo de ER em C#

```
1  //Fonte:
2  //www.c-sharpcorner.com/Language/RegExpressionSample1.asp
3
4  // Function to test for Positive Integers.
5
6  public bool IsNaturalNumber(String strNumber)
7  {
8     Regex objNotNaturalPattern=new Regex("[^0-9]");
9     Regex objNaturalPattern=new Regex("0*[1-9][0-9]*");
10
11    return  !objNotNaturalPattern.IsMatch(strNumber) &&
12       objNaturalPattern.IsMatch(strNumber);
13 }
```

Exemplo de ER em ASP.NET

```
1  //Fonte:
2  //msdn.microsoft.com/library/default.asp?url=/library/en-us/\
3  //dnpag2/html/paght000001.asp
4
5  // Instance method:
6  Regex reg = new Regex(@"^[a-zA-Z'.]{1,40}$");
7  Response.Write(reg.IsMatch(txtName.Text));
8
9  // Static method:
10 if (!Regex.IsMatch(txtName.Text,
11                      @"^[a-zA-Z'.]{1,40}$"))
12 {
13   // Name does not match schema
14 }
```

Exemplo de ER em Java

```
1  //Fonte:
2  //www.cis.upenn.edu/~matuszek/cit597-2002/Examples/\
3  //regex-example.html
4
5  import java.util.regex.*;
6  public class RegexTest {
7    public static void main(String args[]) {
8      String pattern = "[a-z]+";
9      String text = "Now is the time";
10     Pattern p = Pattern.compile(pattern);
11     Matcher m = p.matcher(text);
12     while (m.find()) {
13       System.out.print(text.substring(m.start(),m.end())+"*");
14     }
15   }
16 }
```

Funções Inseguras e “Seguras”

- Vários livros de programação sugerem a não utilização de algumas funções C por serem inseguras
  - Building Secure Software - pág 152 e 153
- A implementação das funções que serão apresentadas é da Libc do OpenBSD 3.8 - 01/11/2005

Algumas funções Inseguras e “Seguras”

Função Insegura	Risco	Sugestão de Contorno
gets()	Atíssimo	fgets(buf, size, stdin)
strcpy()	Alto	strncpy()
strcat()	Alto	strncat()
sprintf()	Alto	snprintf() e precisão do formato
[*]scanf()	Alto	Use precisão do formato
vsprintf()	Alto	Use precisão do formato

Dúvidas!!!

- Porque algumas funções são inseguras?
- Será que as sugestões de contorno resolvem o problema?
- Qual a causa dos problemas?
- Como as funções foram implementadas?

strlen - Explicação

- Função utilizada para calcular o tamanho de uma string
- Retorna a quantidade de caracteres ou bytes

```
size_t strlen(const char *s);
```

strlen - Utilização

```
1  int main (int argc, char ** argv) {
2
3     char * exemplo = "quantidade de bytes";
4
5     printf ("%d\n", strlen(exemplo));
6
7     return 0;
8  }
```

```
1  $ ./teste_strlen
2  19
```

strlen - Problema

- Condição de parada: terminador de string

```
\0
```

```
1  int main (int argc, char ** argv) {
2
3     char a [32];
4     memset (a, 'a', 32);
5     printf ("Tamanho de a = %d\n", strlen(a));
6
7     return 0;
8  }
```

```
1  $ ./problema_strlen
2  Tamanho de a = 49
```

strlen - Solução

```
1  int main (int argc, char ** argv) {
2
3     char a [32];
4     memset (a, 'a', 31);
5     a[31] = '\0';
6     printf ("Tamanho de a = %d\n", strlen(a));
7
8     return 0;
9  }
```

```
1  $ ./solucao_strlen
2  Tamanho de a = 31
```

strlen - Implementação Libc

```
1  size_t
2  strlen(const char *str)
3  {
4          const char *s;
5
6          for (s = str; *s; ++s)
7                  ;
8          return (s - str);
9  }
```

strcpy, strncpy e strlcpy

- Funções utilizadas para copiar o conteúdo entre buffers

```
char * strcpy(char *dst, const char *src);

char * strncpy(char *dst, const char *src, size_t len);

size_t strlcpy(char *dst, const char *src, size_t size);
```

strcpy - Utilização

```
1  int main (int argc, char ** argv) {
2
3     char * origem = "conteudo a ser copiado";
4
5     char destino[32];
6
7     strcpy(destino,origem);
8
9     printf ("%s\n", destino);
10
11    return 0;
12 }
```

```
1  $ ./teste_strcpy
2  conteudo a ser copiado
```

strcpy - Problema

- Condição de parada: terminador de string

```
1  int main (int argc, char ** argv) {
2
3     char origem [32], destino [32];
4     memset (origem, 'a', 32);
5
6     strcpy(destino,origem);
7
8     printf ("Conteúdo de destino = %s\n", destino);
9     printf ("Tamanho de destino = %d\n", strlen(destino));
10
11    return 0;
12 }
```

```
1  $ ./problema_strcpy
2  Conteúdo de destino = aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaXXXXXXXXXXXXXX
3  Tamanho de destino = 49
```

strcpy - Solução

```
1  int main (int argc, char ** argv) {
2
3     char origem [32], destino [32];
4     memset (origem, 'a', 32);
5     origem[31] = '\0';
6
7     strcpy(destino,origem);
8
9     printf ("Conteúdo de destino = %s\n", destino);
10    printf ("Tamanho de destino = %d\n", strlen(destino));
11
12    return 0;
13 }
```

```
1  $ ./solucao_strcpy
2  Conteúdo de destino = aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
3  Tamanho de destino = 31
```

strcpy - Implementação Libc

```
1  char *
2  strcpy(char *to, const char *from)
3  {
4          char *save = to;
5
6          for (; (*to = *from) != '\0'; ++from, ++to);
7          return(save);
8  }
```

strncpy - Utilização

```
1  int main (int argc, char ** argv) {
2
3     char * origem = "conteudo a ser copiado";
4
5     char destino[32];
6
7     strncpy(destino,origem,32);
8
9     printf ("%s\n", destino);
10
11    return 0;
12 }
```

```
1 $ ./teste_strncpy
2 conteudo a ser copiado
```

strncpy - Problema

- Condição de parada: terminador de string ou N bytes forem lidos

```
int main (int argc, char ** argv) {

   char origem [32], destino [32];
   memset (origem, 'a', 32);

   strncpy(destino,origem,32);

   printf ("Conteúdo de destino = %s\n", destino);
   printf ("Tamanho de destino = %d\n", strlen(destino));

   return 0;
}
```

```
$ ./problema_strcpy
Conteúdo de destino = aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa\
aaaaXXXXXXXXXXXXXXX
Tamanho de destino = 81
```

strncpy - Solução

```
1  int main (int argc, char ** argv) {
2
3     char origem [32], destino [32];
4     memset (origem, 'a', 32);
5     origem[31] = '\0';
6
7     strncpy(destino,origem,32);
8
9     printf ("Conteúdo de destino = %s\n", destino);
10    printf ("Tamanho de destino = %d\n", strlen(destino));
11
12    return 0;
13 }
```

```
1  $ ./solucao_strncpy
2  Conteúdo de destino = aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
3  Tamanho de destino = 31
```

strncpy - Implementação Libc

```
1  char *
2  strncpy(char *dst, const char *src, size_t n)
3  {
4          if (n != 0) {
5                  char *d = dst;
6                  const char *s = src;
7
8                  do {
9                          if ((*d++ = *s++) == 0) {
10                                 /* NUL pad the remaining n-1 bytes */
11                                 while (--n != 0)
12                                         *d++ = 0;
13                                 break;
14                         }
15                 } while (--n != 0);
16         }
17         return (dst);
18 }
```

strlcpy - Utilização

```
1  int main (int argc, char ** argv) {
2
3     char * origem = "conteudo a ser copiado";
4
5     char destino[32];
6
7     strlcpy(destino,origem,32);
8
9     printf ("%s\n", destino);
10
11    return 0;
12 }
```

```
1  $ ./teste_strlcpy
2  conteudo a ser copiado
```

strlcpy - Reprodução do problema do strncpy

```
1  int main (int argc, char ** argv) {
2
3     char origem [32], destino [32];
4     memset (origem, 'a', 32);
5
6     strlcpy(destino,origem,32);
7
8     printf ("Conteúdo de destino = %s\n", destino);
9     printf ("Tamanho de destino = %d\n", strlen(destino));
10
11    return 0;
12 }
```

```
1  $ ./reproducao_problema_strncpy
2  Conteúdo de destino = aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
3  Tamanho de destino = 31
```

strlcpy - Implementação Libc

```
1  size_t strlcpy(char *dst, const char *src, size_t siz) {
2          char *d = dst;
3          const char *s = src;
4          size_t n = siz;
5          /* Copy as many bytes as will fit */
6          if (n != 0 && --n != 0) {
7                  do {
8                          if ((*d++ = *s++) == 0)
9                                  break;
10                 } while (--n != 0);
11         }
12         /* Not enough room in dst, add NUL and traverse rest of src */
13         if (n == 0) {
14                 if (siz != 0)
15                         *d = '\0';              /* NUL-terminate dst */
16                 while (*s++)
17                         ;
18         }
19         return(s - src - 1);    /* count does not include NUL */
20 }
```

strcat, strncat e strlcat

strcat

```
1  char *
2  strcat(char *s, const char *append)
3  {
4          char *save = s;
5
6          for (; *s; ++s);
7          while ((*s++ = *append++) != '\0');
8          return(save);
9  }
```

strcat, strncat e strlcat

strncat

```
1  char *
2  strncat(char *dst, const char *src, size_t n)
3  {
4          if (n != 0) {
5                  char *d = dst;
6                  const char *s = src;
7
8                  while (*d != 0)
9                          d++;
10                 do {
11                         if ((*d = *s++) == 0)
12                                 break;
13                         d++;
14                 } while (--n != 0);
15                 *d = 0;
16         }
17         return (dst);
18 }
```

strcat, strncat e strlcat

strlcat

```
1  size_t strlcat(char *dst, const char *src, size_t siz) {
2          char *d = dst;
3          const char *s = src;
4          size_t n = siz;
5          size_t dlen;
6          /* Find the end of dst and adjust bytes left but don't go past end */
7          while (n-- != 0 && *d != '\0')
8                  d++;
9          dlen = d - dst;
10         n = siz - dlen;
11
12         if (n == 0)
13                 return(dlen + strlen(s));
14         while (*s != '\0') {
15                 if (n != 1) {
16                         *d++ = *s;
17                         n--;
18                 }
19                 s++;
20         }
21         *d = '\0';
22         return(dlen + (s - src));       /* count does not include NUL */
23 }
```

strcmp e strncmp

strcmp

```
1  int
2  strcmp(const char *s1, const char *s2)
3  {
4          while (*s1 == *s2++)
5                  if (*s1++ == 0)
6                          return (0);
7          return (*(unsigned char *)s1 - *(unsigned char *)--s2);
8  }
```

strcmp e strncmp

strncmp

```
1  int
2  strncmp(const char *s1, const char *s2, size_t n)
3  {
4
5          if (n == 0)
6                  return (0);
7          do {
8                  if (*s1 != *s2++)
9                          return (*(unsigned char *)s1 - *(unsigned char *)--s2);
10                 if (*s1++ == 0)
11                         break;
12         } while (--n != 0);
13         return (0);
14 }
```

O que são?

- São invólucros utilizados para prévias validações
- Eles podem ser utilizados antes de chamadas de função
- Podem ser utilizados em aplicações a parte que intermedia uma comunicação
- Como exemplo de wrapper será utilizado a libsafe

Libsafe

- Previne *buffer overflows*
- É uma espécie de invólucro para a biblioteca padrão do C
- Intercepta e checa os limites dos argumentos antes de passar para as funções
- É transparente para os processos protegidos pois é carregada automaticamente (`/etc/ld.so.preload` ou LD_PRELOAD)

`http://www.research.avayalabs.com/project/libsafe/index.html`

strncpy

```
1  char *strncpy(char *dest, const char *src, size_t n) {
2      static strncpy_t real_strncpy = NULL;
3      size_t max_size, len;
4
5      if (!real_strncpy)
6          real_strncpy = (strncpy_t) getLibraryFunction("strncpy");
7
8      if (_libsafe_exclude)
9          return real_strncpy(dest, src, n);
10
11     if ((max_size = _libsafe_stackVariableP(dest)) == 0) {
12         LOG(5, "strncpy(<heap var> , <src>)\n");
13         return real_strncpy(dest, src, n);
14     }
15
16     LOG(4, "strncpy(<stack var> , <src>) stack limit=%d)\n", max_size);
17
18     if (n > max_size && (len = strnlen(src, max_size)) == max_size)
19         _libsafe_die("Overflow caused by strncpy()");
20
21     return real_strncpy(dest, src, n);
22 }
```

strncat

```
1  char *strncat(char *dest, const char *src, size_t n) {
2      static strncat_t real_strncat = NULL;
3      size_t max_size;
4      uint dest_len, src_len;
5
6      if (!real_strncat)
7          real_strncat = (strncat_t) getLibraryFunction("strncat");
8
9      if (_libsafe_exclude)
10         return real_strncat(dest, src, n);
11
12     if ((max_size = _libsafe_stackVariableP(dest)) == 0) {
13         LOG(5, "strncat(<heap var> , <src>)\n");
14         return real_strncat(dest, src, n);
15     }
16     LOG(4, "strncat(<stack var> , <src>) stack limit=%d\n", max_size);
17     dest_len = strnlen(dest, max_size);
18     src_len = strnlen(src, max_size);
19
20     if (dest_len + n > max_size && dest_len + src_len >= max_size)
21         _libsafe_die("Overflow caused by strncat()");
22
23     return real_strncat(dest, src, n);
24 }
```

getLibraryFunction

```
1  /*
2   * returns a pointer to the implementation of 'funcName' in
3   * the libc library.  If not found, terminates the program.
4   */
5  static void *getLibraryFunction(const char *funcName)
6  {
7      void *res;
8
9      if ((res = dlsym(RTLD_NEXT, funcName)) == NULL) {
10         fprintf(stderr, "dlsym %s error:%s\n", funcName, dlerror());
11         _exit(1);
12     }
13     return res;
14 }
```

Objetivo

- Discutir problemas em alguns programas utilizados pela comunidade
- Serão utilizados softwares de código aberto
- Foram retirados do site da securiteam alguns anúncios de vulnerabilidades deste ano
  - `http://www.securiteam.com`
- Todas as vulnerabilidades discutidas serão tratadas
  - Foi pesquisado um anúncio por vulnerabilidade

Estudos de Casos

- wget and curl NTLM Username Buffer Overflow 16 Oct. 2005
- PHP HelpDesk Authentication Bypass 11 Oct. 2005
- xine based CD Player Format String 10 Oct. 2005
- SquirrelMail Address Add Plugin XSS 29 Sep. 2005
- SquirrelMail S/MIME Plugin Command Injection 17 Feb. 2005

wget and curl NTLM Username Buffer Overflow

"GNU Wget is a free software package for retrieving files using HTTP, HTTPS and FTP, the most widely-used Internet protocols. It is a non-interactive commandline tool, so it may easily be called from scripts, cron jobs, terminals without X-Windows support, etc."
"curl is a command line tool for transferring files with URL syntax, supporting FTP, FTPS, TFTP, HTTP, HTTPS, GOPHER, TELNET, DICT, FILE and LDAP. curl supports HTTPS certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, kerberos...), file transfer resume, proxy tunneling and a busload of other useful tricks."
A buffer overflow vulnerability in multiple vendor's implementations of curl and wget allows attackers to execute arbitrary code.

Fonte: http://www.securiteam.com/unixfocus/6O00E1PEAS.html

wget: Pontos Relevantes

- Software utilizado para “baixar” arquivos
- Problema com NTLM
- wget versão 1.10.1
- `http://ftp.gnu.org/pub/gnu/wget/`
- Arquivo: `wget-1.10.1/src/http-ntlm.c`

wget: Problema

```
1  char *
2  ntlm_output (struct ntlmdata *ntlm, const char *user, \
3                  const char *passwd, int *ready)
4  {
5     ...
6     char ntlmbuf[256];
7     ...
8     usr = user;
9     ...
10    userlen = strlen(usr);
11    ...
12    size=64;
13    ntlmbuf[62]=ntlmbuf[63]=0;
14
15    memcpy(&ntlmbuf[size], domain, domlen);
16    size += domlen;
17
18    memcpy(&ntlmbuf[size], usr, userlen);
19    size += userlen;
20    ...
21 }
```

wget: Reprodução

```
1  #define NTLMSTATE_TYPE2 1
2  char * ntlm_output (int ntlm, const char *user, const char *passwd, int *ready) {
3
4     const char *domain=""; /* empty */
5     const char *host=""; /* empty */
6     int domlen=strlen(domain);
7     int hostlen = strlen(host);
8     int size;
9     char ntlmbuf[256];
10
11    /* not set means empty */
12    if(!user)
13       user="";
14
15    if(!passwd)
16       passwd="";
17
18    switch(ntlm) {
19
20       case NTLMSTATE_TYPE2:
21          {
```

wget: Reprodução (cont.)

```
1           const char *usr;
2           int userlen;
3           usr = strchr(user, '\\');
4           if(!usr)
5              usr = strchr(user, '/');
6
7           if (usr) {
8              domain = user;
9              domlen = usr - domain;
10             usr++;
11          }
12          else
13             usr = user;
14          userlen = strlen(usr);
15
16          /* size is now 64 */
17          size=64;
18          ntlmbuf[62]=ntlmbuf[63]=0;
19
20          memcpy(&ntlmbuf[size], domain, domlen);
21          size += domlen;
22
23          memcpy(&ntlmbuf[size], usr, userlen);
24          size += userlen;
```

wget: Reprodução (cont.)

```
1           }
2           break;
3       }
4   }
5
6   int main (int argc, char ** argv) {
7
8       ntlm_output (NTLMSTATE_TYPE2, argv[1], argv[2], 0);
9
10      return 0;
11
12  }
```

wget: Teste

```
$ ./reproducao_wget ‘perl -e ’print "usuario"x34’‘ "senha"
Segmentation fault
```

wget: Solução - versão 1.10.2

```
1   ...
2   /* size is now 64 */
3   size=64;
4   ntlmbuf[62]=ntlmbuf[63]=0;
5
6   if(size + userlen + domlen >= sizeof(ntlmbuf))
7      return NULL;
8
9   memcpy(&ntlmbuf[size], domain, domlen);
10  size += domlen;
11
12  memcpy(&ntlmbuf[size], usr, userlen);
13  size += userlen;
14  ...
```

wget: Teste Solução

```
$ ./solucao_wget ‘perl -e ’print "usuario"x34’‘ "senha"
```

xine based CD Player Format String

"xine is a free multimedia player. It plays back CDs, DVDs, and VCDs. It also decodes multimedia files like AVI, MOV, WMV, and MP3 from local disk drives, and displays multimedia streamed over the Internet. It interprets many of the most common multimedia formats available - and some of the most uncommon formats, too."
By setting up a malicious CDDB server, an attacker can overwrite arbitrary memory locations with arbitrary data using Xine based CD Players.

Fonte: http://www.securiteam.com/unixfocus/6E00A00EBK.html

xine: Pontos Relevantes

- Software utilizado para reproduzir arquivos multimídia
- Problema com CDDB
- xine-lib versão 1.0.1
- `http://xinehq.de/index.php/releases`
- Arquivo: `xine-lib-1.0.1/src/input/input_cdda.c`

xine: Problema

```
1  static void _cdda_save_cached_cddb_infos(cdda_input_plugin_t \
2                                           *this, char *filecontent) {
3     ...
4     fprintf(fd, filecontent);
5     ...
6  }
```

xine: Reprodução

```
1  #include <stdio.h>
2  #define XINE_PATH_MAX   768
3  #define XINE_NAME_MAX   256
4  static void _cdda_save_cached_cddb_infos(char *cache_dir, char *filecontent) {
5     char   cfile[XINE_PATH_MAX + XINE_NAME_MAX + 1];
6     FILE  *fd;
7
8     if((cache_dir == NULL) || (filecontent == NULL))
9        return;
10
11    memset(&cfile, 0, sizeof(cfile));
12
13    snprintf(cfile, sizeof(cfile), "%s", cache_dir);
14
15    if((fd = fopen(cfile, "w")) == NULL) {
16       printf("input_cdda: fopen(%s) failed.\n", cfile);
17       return;
18    }
19    else {
20       fprintf(fd, filecontent);
21       fclose(fd);
22    }
23 }
```

xine: Reprodução (cont.)

```
1  int main (int argc, char ** argv) {
2
3     _cdda_save_cached_cddb_infos (argv[1], argv[2]);
4
5     return 0;
6  }
```

xine: Teste

```
1  $ ./reproducao_xine arquivo "%x %x %x %x"
2
3  $ cat arquivo
4  80486c0 bfe9f858 b7fa21d8 b7e6c17c
```

xine: Solução - versão 1.0.3a

```
1  static void _cdda_save_cached_cddb_infos(cdda_input_plugin_t \
2                                           *this, char *filecontent) {
3     ...
4     fprintf(fd, "%s", filecontent);
5     ...
6  }
```

xine: Teste Solução

```
1  $ ./solucao_xine  arquivo "%x %x %x %x"
2
3  $ cat arquivo
4  %x %x %x %x
```

PHP HelpDesk Authentication Bypass

> PHP Helpdesk is "a tool that allows administrators to handle tasks related to their organisation. This tool is used to record and monitor the progress of tasks assigned to people. This is an excellent and simple tool for handling tasks". PHP Helpdesk has a fault in the implementation of the cookie values set. Using crafted URL's it is possible to get full access to the system.

Fonte: `http://www.securiteam.com/unixfocus/6D00J0AEAU.html`

PHP HelpDesk: Pontos Relevantes

- Software de gerenciamento de tarefas
- Problemas na autenticação
- PHP HelpDesk versão 0.6.16
- `http://sourceforge.net/projects/phphelpdesk`
- Arquivo: `phphelpdesk/includes/checkuser.inc.php`

PHP HelpDesk: Problema

```
1  <?php   //check txtUsername and txtPassword and set variables
2     ...
3     $query = "SELECT * FROM security ";
4     $query .= "WHERE s_user='$txtUsername' AND \
5                     s_password='$txtPassword';";
6
7     $mysql_result = query($query);
8     ...
9  ?>
```

PHP HelpDesk: Reprodução

```
1  <?php
2
3     function query ($a) {print $a;};
4
5     $txtUsername = $_POST['txtUsername'];
6     $txtPassword = $_POST['txtPassword'];
7
8     $query = "SELECT * FROM security ";
9     $query .= "WHERE s_user='$txtUsername' AND \
10                    s_password='$txtPassword';";
11
12    $mysql_result = query($query);
13
14 ?>
```

PHP HelpDesk: Teste

```
$ echo "txtUsername= ’; select * from security -- " | \
lynx -dump -post_data http://127.0.0.1/reproducao_phphd.php

   SELECT  *  FROM  security WHERE s_user=’ ’; select * from \
   security --’ AND s_password=’’;
```

PHP HelpDesk: Solução versão - ???

NÃO ENCONTRADA

PHP HelpDesk: Possível Solução

```
1  <?php   //check txtUsername and txtPassword and set variables
2     ...
3
4     if (!ereg ("^[a-zA-Z0-9]{1,8}$", $txtUsername ) ||
5         !ereg ("^[a-zA-Z0-9]{1,8}$", $txtPassword )) {
6        print $l_usernameorpasswordareincorrect;
7        return;
8     }
9
10    $query = "SELECT * FROM security ";
11    $query .= "WHERE s_user='$txtUsername' AND \
12                      s_password='$txtPassword';";
13
14    $mysql_result = query($query);
15    ...
16 ?>
```

PHP HelpDesk: Teste Possível Solução

```
$ echo "txtUsername= '; select * from security -- " | \
lynx -dump -post_data http://127.0.0.1/solucao_phphd.php

   Username or Password are incorrect!
```

SquirrelMail S/MIME Plugin Command Injection

> Squirrelmail S/MIME plugin ’enables the viewing of S/MIME-signed messages of the MIME "multipart/signed" format’. Remote exploitation of a command injection vulnerability in the Squirrelmail S/MIME plugin allows web mail users to execute arbitrary commands with the privileges of the web server.

Fonte: `http://www.securiteam.com/unixfocus/5NP0J20EUG.html`

SquirrelMail S/MIME Plugin: Pontos Relevantes

- Software utilizado como webmail
- Plugin utilizado assinar mensagens
- Problema no plugin
- smime versão 0.5
- `http://www.squirrelmail.org/plugin_download.php?id=54&rev=1054`
- Arquivo: smime/viewcert.php

SquirrelMail S/MIME Plugin: Problema

```
1  <?php
2    ...
3    if(!isset($cert)) $cert=$_GET['cert'];
4    ...
5    function x509_open($cert) {
6      global $cert_in_dir, $openssl;
7      ...
8      exec("$openssl x509 -in $cert_in_dir$cert -subject \
9      -issuer -dates -serial -fingerprint -noout 2>/tmp/err", $lines);
10
11     return array(
12       ...
13     );
14   }
15   ...
16   list ($ow, $is, $nb, $na, $sn, $fp) = x509_open($cert);
17   ...
18   show_field("Owner:", $ow);
19   ...
20 ?>
```

SquirrelMail S/MIME Plugin: Reprodução

```
1  <?php
2
3    $cert_in_dir="/tmp/";
4    $openssl="openssl";
5
6    if(!isset($cert)) $cert=$_GET['cert'];
7
8    function x509_open($cert) {
9       exec("$openssl x509 -in $cert_in_dir$cert -subject \
10      -issuer -dates -serial -fingerprint -noout 2>/tmp/err", $lines);
11
12      return $lines;
13   }
14
15   list ($ow, $is, $nb, $na, $sn, $fp) = x509_open($cert);
16
17   print "$ow, $is, $nb, $na, $sn, $fp";
18
19 ?>
```

SquirrelMail S/MIME Plugin: Teste

```
$ echo "http://127.0.0.1/reproducao_viewcert.php?cert=;date;" \
| lynx -dump -

   Sat Nov 5 11:16:58 BRST 2005, , , , ,
```

SquirrelMail S/MIME Plugin: Solução versão 0.6-1.4

```
1  <?php
2    ...
3    if(!isset($cert)) $cert=$_GET['cert'];
4    ...
5    function x509_open($cert) {
6       global $cert_in_dir, $openssl;
7       ...
8       $safe_cert = escapeshellarg("$cert_in_dir$cert");
9
10      exec("$openssl x509 -in $safe_cert -subject \
11      -issuer -dates -serial -fingerprint -noout 2>/tmp/err", $lines);
12
13      return array(
14      ...
15      );
16   }
17   ...
18   list ($ow, $is, $nb, $na, $sn, $fp) = x509_open($cert);
19   ...
20   show_field("Owner:", $ow);
21   ...
22 ?>
```

SquirrelMail S/MIME Plugin: Teste Solução

```
$ echo "http://127.0.0.1/solucao_viewcert.php?cert=;date;" \
| lynx -dump -

   , , , , ,
```

SquirrelMail Address Add Plugin XSS

> A XSS vulnerability has been detected in the Address Add Plugin for Squirrelmail. The problem is caused by insufficient input sanitation.

Fonte: `http://www.securiteam.com/unixfocus/5UP0L2KGWU.html`

SquirrelMail Address Add Plugin: Pontos Relevantes

- Sotware utilizado como webmail
- Plugin utilizado para adicionar o rementente no catálogo
- Problema no plugin
- address_add versão 2.0-1.2.8
- `http://www.squirrelmail.org/plugins_category.php?category_id=2`
- Arquivo: `/address_add-2.0/add.php`

SquirrelMail Address Add Plugin: Problema

```
1  <?php
2     ...
3     if (isset($_GET['first']))
4     $first = $_GET['first'];
5     $first = strip_tags(urldecode(urldecode($first)));
6     ...
7     print "<td bgcolor=\"$color[4]\" align=left><input name= \
8     \"addaddr[firstname]\" size=45 value=\"$first\"></td>\n";
9     ...
10 ?>
```

SquirrelMail Address Add Plugin: Reprodução

```
1  <?php
2     if (isset($_GET['first']))
3        $first = $_GET['first'];
4     $first = strip_tags(urldecode(urldecode($first)));
5     print "<td bgcolor=\"$color[4]\" align=left><input name= \
6     \"addaddr[firstname]\" size=45 value=\"$first\"></td>\n";
7  ?>
```

SquirrelMail Address Add Plugin: Teste

```
$ echo http://127.0.0.1/reproducao_add.php?first=%22%20 \
onMouseOver=%22alert\(\'teste\'\) | lynx -source -

<td bgcolor="" align=left><input name="addaddr[firstname]\"
size=45 value="" onMouseOver="alert(teste)"></td>
```

SquirrelMail Address Add Plugin: Solução versão 2.1-1.4.0

```
1  <?php
2     ...
3     if (isset($_GET['first']))
4        $first = $_GET['first'];
5     $first = clean(urldecode(urldecode($first)));
6     ...
7     print "<td bgcolor=\"$color[4]\" align=left><input name= \
8     \"addaddr[firstname]\" size=45 value=\"$first\"></td>\n";
9     ...
10 ?>
```

SquirrelMail Address Add Plugin: Solução versão 2.1-1.4.0

```
1  function clean($source) {
2     $source = strip_tags($source);
3     $source = str_replace ( array ('|', '+', '"', "'", '<', '>'\
4     , ':', '!', '(', ')', '\\', '/', '?', '='), '', $source);
5
6     $source = strip_tags($source);
7
8     return $source;
9  }
```

SquirrelMail Address Add Plugin: Teste Solução

```
$ echo http://127.0.0.1/solucao_add.php?first=%22%20 \
onMouseOver=%22alert\(\'teste\'\) | lynx -source -

<td bgcolor="" align=left><input name="addaddr[firstname]\"
size=45 value=" onMouseOveralertteste"></td>
```

Considerações Finais

Valide todas as entradas

Referências Bibliográficas

John Viega and Gary McGraw
Building Secure Software - How to Avoid Security Problems the Right Way.
Addilson-Wesley, 1st edition, 2004.
ISBN 0-201-78695-8

Referências Bibliográficas

Michael Howard and David LeBlanc and John Viega
19 Deadly Sins of Software Security.
McGraw-Hill Osborne Media, 1 edition, 2005
ISBN 0-072-26085-8

Referências Bibliográficas

John Viega and Matt Messier
Secure Programming Cookbook for C and C++.
O'Reilly, 1 edition, 2003
ISBN 0-596-00394-3

Referências Bibliográficas

Mark G. Graff and Kenneth R. Van Wyk
Secure Coding: Principles and Practices.
O'Reilly, 1 edition, 2003
ISBN 0-596-00242-4

Referências Bibliográficas

Michael Howard and David C. LeBlanc.
Writing Secure Code
Microsoft Press, 2nd edition, 2003.
ISBN 0-735-61722-8

Referências Bibliográficas

Greg Hoglund and Gary McGraw
Exploiting Software - How to Break Code.
Addilson-Wesley, 1st edition, 2002.
ISBN 0-201-71152-X

Referências Bibliográficas

Luiz Gustavo C. Barbato and Luiz Otávio Duarte and Antonio Montes
Programação Segura: Uma Introdução à Auditoria de Códigos.
GTS 02.2004 - Grupo de Trabalho em Segurança

Referências Bibliográficas

Luiz Gustavo C. Barbato and Luiz Otávio Duarte and Antonio Montes
Vulnerabilidades de Software e Formas de Minimizar suas Explorações.
GTS 01.2005 - Grupo de Trabalho em Segurança

Contatos

André Ricardo Abed Grégio
andre.gregio@lac.inpe.br
Luiz Gustavo C. Barbato
lgbarbato@lac.inpe.br
lgbarbato@cenpra.gov.br
Luiz Otávio Duarte
duarte@lac.inpe.br
Antonio Montes
antonio.montes@cenpra.gov.br