Programação I

Capítulo 10

"Show me the code"[1]
Linus Torvalds

1 "Mostre-me o código!" Exclamação corrente no mundo do Open Source sempre que alguém quer mostrar algum programa e não revela o que está dentro...

Todo o bom hacker deve ter noções de programação. Caso contrário, não será considerado nada além de um script kiddie incapaz de criar seus próprios scanners e exploits. Devemos ter consciência de que os métodos aplicados para a segurança não param de evoluir, e torna-se cada vez mais difícil encontrar boas ferramentas pela internet. Nosso propósito não é transformar o leitor em um expert em programação, mas demonstrar as técnicas e linguagens mais utilizadas para que ele possa adaptar seus projetos para qualquer situação. Esta aula abordará temas como algoritmos e introdução às linguagens de programação mais utilizadas.

1-Algoritmos

Existem várias definições para algoritmos. Podemos entendê-los como uma receita genérica que pode ser utilizada na criação de qualquer programa em qualquer linguagem. Ele conta com uma série de operações primitivas que devem ser interligadas. Estudaremos nesse tópico a estrutura básica dos algoritmos, bem como seus componentes e organização.

1.1-Estrutura do Algoritmo

Os algoritmos seguem uma estrutura básica para que não ocorra confusão caso haja alguma necessidade de alteração:

Algoritmo – Nome do algoritmo criado
Variáveis – Área para a declaração de variáveis
Procedimentos – Declaração dos procedimentos utilizados
Funções – Declaração das funções utilizadas
Início – Corpo do Algoritmo
Fim – Término do algoritmo

1.2-Variáveis

São as unidades de armazenamento de informações para o seu algoritmo. Existem diversos formatos especialmente designados para funções específicas:
Inteiro: qualquer número inteiro, negativo, nulo ou positivo.
Real: qualquer número real, negativo, nulo ou positivo.
Caractere: qualquer conjunto de caracteres alfanuméricos.
Lógico: tipo especial de variável que armazena apenas os valores V e F, onde V representa VERDADE e F FALSO

As variáveis devem ser declaradas para que os programas possam interpretá-las. Para tanto, vamos utilizar a seguinte notação: *nome: formato.*

1.3-Operadores Aritméticos, Lógicos e Relacionais

Durante a execução do seu programa, muitas variáveis poderão ter seus valores alterados em função
Aritméticos
+ = Adição
- = Subtração
* = Multiplicação
/ = Divisão
Quociente = Quociente da divisão de inteiros
Resto = Resto da divisão de inteiros
EXP(a,b) = Exponenciação a^b
Operadores Relacionais

= – igual
¹ – diferente
< – menor
> – maior
£ – menor ou igual
³ – maior ou igual

Operadores lógicos

e – e lógico ou conjunção
ou – ou lógico ou disjunção
não – negação

Comandos de entrada e saída e de controle de fluxo

Entrada e Saída

São os comandos responsáveis pela troca de informações entre a máquina e os meios exteriores, como impressoras e monitores, e possuem a seguinte estrutura:

– *Entrada de dados-*
Leia(variável1, variável2...)

– *Saída de dados-*
Imprima(variável1, variável2...)

Controle de Fluxo

São os passos que devem ser seguidos para a resolução de problemas impostos pelo programador. Os comandos de controle de fluxo estão organizados da seguinte maneira: seqüência, seleção e repetição.

Seqüência – São executados exatamente na ordem descrita pelo programador.

Início
Comando 1
...
Comando n
Fim

Seleção – São utilizados na tomada de decisões quando houver condições.

Se(Expressão Lógica)
Então faça Instrução1

Repetição – Utilizados para repetir um conjunto de ações.

Enquanto (Expressão Lógica) **faça** Instrução

Para valor inicial **até** valor final **faça** Instrução

Repita
Instrução
Até (Expressão Lógica)

1.5-Idealização de um algoritmo

Um algoritmo surge a partir de um problema. Devemos analisá-lo e criar uma situação em que a solução seja possível de uma forma lógica. Vamos partir de uma situação comum, como a troca de um pneu furado.

Trocando o pneu:
se o pneu estiver furado, **então faça:**
–Abrir o porta-malas
–Pegar o estepe
–Pegar as ferramentas
–Afrouxar os parafusos com a chave
–Suspender o carro
–Retirar os parafusos com a chave
–Colocar o estepe
–Fixar os parafusos
–Abaixar o carro
–Apertar os parafusos
–Guardar as ferramentas e o estepe
Fim

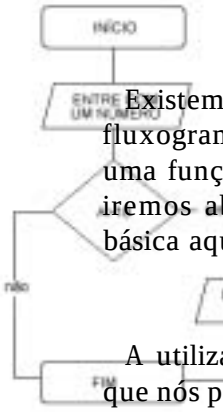
1.6-Considerações sobre algoritmos

Os algoritmos podem ser estruturados de diversas formas. Isso vai depender principalmente da experiência de cada um. O exemplo anterior da troca de pneu poderia ser escrito de diversas formas, mas sem perder o seu foco inicial. Outro fator importante que influencia na sua criação, é o padrão imposto por empresas, cada qual com seus métodos para auxiliar na documentação. Seria interessante para o hacker descobrir estes métodos, pois eles podem trazer informações valiosas.

Terminamos aqui a nossa seção sobre algoritmos. Obviamente esse tema se estende muito além dessas linhas, mas nós decidimos por abordar o assunto de uma forma mais superficial, para não fugir do real propósito do livro, que é o universo hacker.

2-Fluxogramas

Basicamente, o fluxograma é a representação gráfica de um algoritmo, utilizada para a compreensão do controle de fluxo. Possui diversos elementos que representam desde base de dados até ações a serem executadas. É considerada uma ferramenta indispensável para a visualização de processos como veremos a seguir.



2.1-Componentes do fluxograma

Símbolo	Significado
	Início / fim do algoritmo
	Leitura e escrita (entrada e saída) de dados
	Instruções de atribuição
	Estrutura condicional

Existem vários itens que compõem o fluxograma, cada qual representando uma função dentro do algoritmo. Nós iremos abordar apenas a simbologia básica aqui:

2.1-Aplicação

A utilização do fluxograma permite que nós possamos enxergar quais os caminhos que serão seguidos pelo programa dependendo das condições impostas pelo desenvolvedor. Essa visualização permite a correção de erros e pode até demonstrar meios de otimizar o programa. No exemplo seguinte, veremos como fica o fluxograma do seguinte problema:

Entre com um número e mostre a palavra "OK" se este número for maior do que 10.

3-Linguagens de Programação

A existência de diversas linguagens de programação não é nenhuma novidade para os hackers. O problema está em adotar uma delas para seus propósitos. Apesar das semelhanças no desenvolvimento, algumas possuem recursos diferenciados como uma melhor adaptação para determinado tipo de ambiente ou rotinas simplificadas e com mais opções para o programador.

A plataforma mais utilizada para a produção de programa de utilização hacker é o Unix, pois a eficiência e a possibilidade de encontrar mais recursos a tornam perfeita para execução dos mais variados tipos de programas. Certas linguagens como PHP e Perl eram exclusivas do Unix, mas com o tempo, o Windows também ganhou espaço e utilitários que possibilitam seu desenvolvimento nesse ambiente.

Agora, nós iremos estudar algumas das mais populares linguagens de programação, divididas entre compiladas e interpretadas. A principal diferença entre as duas é que, na primeira, a utilização de um compilador é necessária para que o computador seja capaz de executar as tarefas desejadas, e a segunda pode ser escrita em um arquivo de texto comum (script), pois sua interpretação não requer qualquer tipo de tradução para que a máquina a entenda, para que você possa dar seus primeiros passos no mundo dos códigos-fonte.

3.1-Interpretadas

Linguagens interpretadas são aquelas que não precisam de compilação, sendo interpretadas pelo ambiente do jeito que foram escritas.

3.1.1-Shell Scripting

O Shell é um programa que permite a interação do usuário com os sistemas operacionais, tais como o Windows e o Linux. Já o shell script é a linguagem utilizada para a criação desses programas, na qual você pode escrever suas instruções em um arquivo-texto para executá-los.

Ele pode ser definido como uma série de comandos escritos em um arquivo-texto, muito parecido com arquivos batch do MS-DOS. Ele pode ser utilizado para escrever seus próprios comandos em um sistema operacional e automatizar tarefas, como administração de sistemas. Podemos dizer que Shell Scripting é quase uma exclusividade para programadores que utilizam o Unix como plataforma. Vamos ver um exemplo de script:

```
#!/bin/sh
#Teste de Shell scripting
clear
echo "Universidade"
```


Note que a primeira linha representa o interpretador do Shell. O caminho descrito deve coincidir sempre com o endereço do compilador no sistema, para que o programa possa ser executado.

3.1.2-Perl

Perl é uma das linguagens interpretadas mais utilizadas por desenvolvedores. Ela foi criada para rodar em plataformas Unix, mas sua versatilidade para produzir aplicações para a Web logo foi reconhecida, tornando-a mais popular ainda. Seus scripts estão presentes em grande parte das ferramentas de pesquisa, contadores, livros de visitas e formulários. Ela foi originalmente concebida para plataformas Unix, mas já encontra espaço de desenvolvimento no Windows também. A seguir, veremos um exemplo de codificação da linguagem Perl:

```
#!/usr/bin/perl5
# Exemplo de código em perl
print "Content-type: text/html\n\n";
print "Universidade";
exit;
```

A primeira linha do código representa a localização do interpretador perl dentro do servidor. Ela deve coincidir com o local no seu servidor.

3.1.-PHP

O PHP é caracterizado por ser *open source* e de uso geral. *Open Source* significa "código aberto", isto é, ela está aberta para que o usuário possa fazer as modificações que bem entender. Outra característica interessante é que, diferente de outras linguagens, ele é escrito junto com o próprio HTML no desenvolvimento da página. Os scripts são processados no servidor retornando apenas a codificação do HTML como resultado para o usuário. Dessa forma, o código fica seguro sem risco de ser copiado. Por ser uma linguagem *server-side*, o PHP é capaz de realizar qualquer coisa, além disso, ele pode rodar em todos os tipos de servidores e plataformas.

Exemplo de sintaxe PHP:

```
<html>
   <head>
      <title>Exemplo</title>
   </head>
   <body>
```

```
<?php
 echo "O PHP é uma linguagem versátil!";
?>

  </body>
</html>
```

Este exemplo imprime na tela do usuário apenas a mensagem "O PHP é uma linguagem versátil!". Note que nós utilizamos toda a estrutura em HTML, a única novidade foi a inclusão da tag que representa a linguagem, no caso: **<?php** para iniciar e **?>** para finalizar. Para rodar o PHP, você necessita do próprio, mais um servidor e um navegador para Internet.

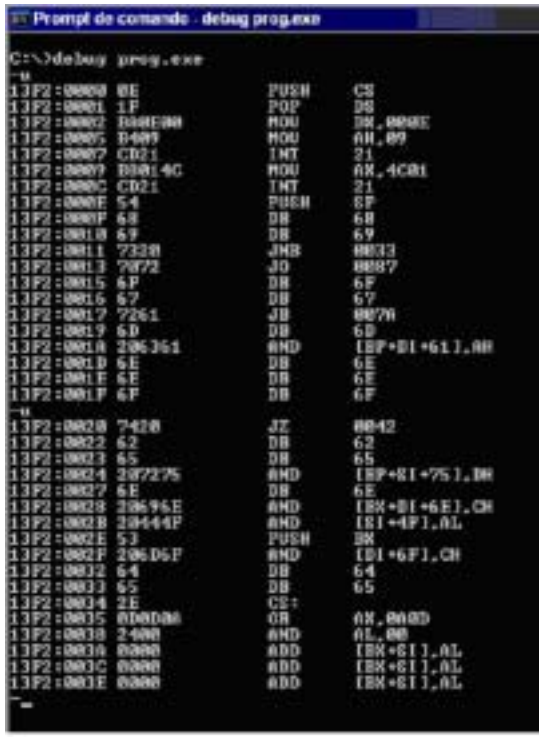
3.2-Compiladas

As linguagens compiladas têm seus códigos-fonte lidos por um programa chamado compilador, que por sua vez, cria um arquivo binário executável.

3.2.1-Assembler

Assembler é a linguagem de mais baixo nível que é composta praticamente de instruções (assembly) que representam instruções escritas em linguagem de máquina, tornando-a muito rápida em função de sua proximidade com o hardware. Uma das principais vantagens é a flexibilidade e controle que ela tem sobre o computador. As instruções dessa linguagem podem ser incorporadas em qualquer programa a ser compilado, isso porque, ao compilar um código qualquer, este será transformado em assembly.

Para fazer o teste, basta apenas entrar na linha de comando e utilizar o debug com o nome de um programa ".exe" qualquer e então pressionar u (unassebly) para ver o código em assembly, como mostra a figura a seguir:



Na primeira coluna, você encontra as posições de memória, na central as instruções e, na última, o conjunto de registradores utilizados para executar os comandos no programa.

3.2.2-Pascal

A linguagem Pascal foi desenvolvida na década de 70, e foi baseada em algumas linguagens como o PLI e ALGOL. Em 1983, a Borland lançou o Turbo Pascal, aproveitando o aquecimento do mercado de informática. Um programa escrito nessa linguagem deve ser iniciado com a palavra program, seguida por um cabeçalho, um ponto e vírgula (;), um bloco (begin & end), e terminando com um ponto final (.), como demonstramos na seguinte estrutura:

```
program Universidade;
var Nome: String;
begin
      write('Nota:');
      readln(nota);
      writeln('Nota: ',nota);
      readln;
end.
```

Podemos notar a simplicidade do código e a total semelhança com o que vimos anteriormente sobre algoritmos. Muitos programas de invasão e nuke são escritos em Pascal, como o WinNuke que veremos em outro capítulo.

3.2.3 - C

Considerada como uma linguagem de nível médio, o C, e suas variações, ainda é amplamente utilizadaopor desenvolvedores. Foi criado nos laboratórios Bell, em 1970, por Brian Kernighan e Dennis M. Ritchie a partir de uma linguagem até então conhecida como B. Daremos ênfase à variação conhecida como C++, a mais popular entre os programadores. A seguir, veremos um exemplo de tal codificação:

```
#include <iostream.h>

struct contador
{
 int num;

 void incrementa(void){num=num+1;};

 void comeca(void){num=0;};
};
```

```
void main()
{
  contador contador;

  contador.comeca();

  cout << contador.num << endl;
  contador.incrementa();
  cout << contador.num << endl;
}
```

A linguagem conhecida como Java foi criada com base na codificação C. Dessa forma, o bom conhecedor da linguagem C está qualificado para trabalhar com uma das linguagens mais utilizadas na Internet.

4- Conclusões

Acabamos de fornecer uma visão geral, bem como dicas sobre as linguagens mais utilizadas para o desenvolvimento de ferramentas para hackers. Nos próximos capítulos voltaremos a falar sobre programação, dando exemplos específicos de como ela pode ser utilizada para a criação de exploits e também na engenharia social.