# Aula - Brute Force

sábado, 5 de junho de 2021 09:45

Prof. Fábio Henrique Cabrini
fabio.cabrini@usp.br (acadêmico) - fabio.cabrini@fatec.sp.gov.br
fabio.cabrini@gethelix.com.br (corporativo)
Professor universitário, co-founder & CEO Helix Platform,
Fiware Evangelist
Lattes: http://lattes.cnpq.br/3044213933175294

**Tema da aula:** Redirecionamento de entrada/saída padrão em sistemas POSIX, Brute Force, Multi-thread, webcrawler e Linux

**O que vamos aprender e ser capazes de realizar após essa aula?**

- *Identificar e aplicar o recurso de redirecionamento de entrada e saída padrão através de redirecionadores e pipelines no terminal Linux;*
- *Ataque de Brute Force com WebCrawler;*
- *Desenvolver um programa em Linguagem C capaz de redirecionar a saída de um processo filho à entrada de um processo pai utilizando pipelines baseadas nos system calls.*

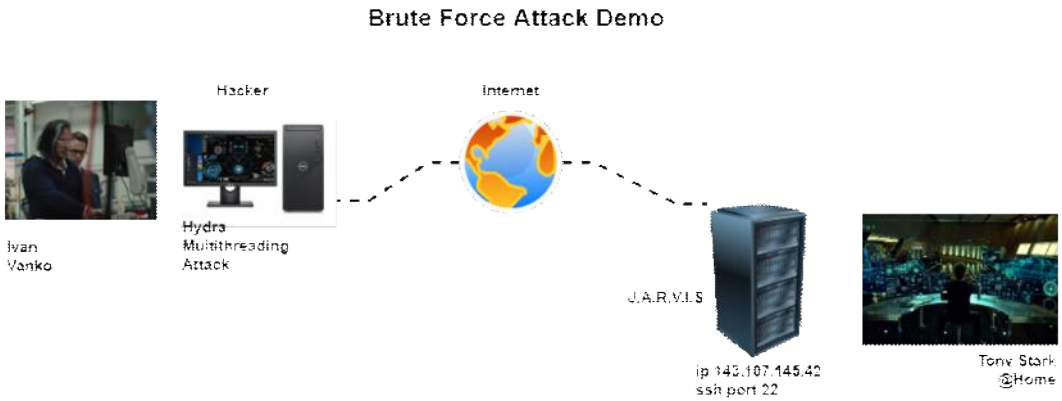
**Recursos:** Terminal Linux ou (Windows Subsystem for Linux) e gcc

# Brute Force Attack

segunda-feira, 14 de junho de 2021 10:05

**Estudo de caso:** Os ataques de *brute force* são utilizados para obter acesso a sistemas computacionais remotos, como:
Páginas *web*, SSH, FTP, banco de dados, etc...

**Demonstração:** Ivan Vanko realizando um ataque de Brute Force no SSH do J.A.R.V.I.S.




**Ferramenta:** Hydra

**Demonstração:** uso de *multithreading*

*(Iron Man 2 - Hammer Drones)*

**Atenção:** Art. 154-A. Invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita: (Incluído pela Lei nº 12.737, de 2012) Vigência Pena - detenção, de 3 (três) meses a 1 (um) ano, e multa.

**Observações:**

- A demonstração do ataque está sendo realizada em ambiente de Sandbox (apartado)!
- Utilizar o comando time (Windows) para medir o tempo do ataque
- Utilizar duas máquinas Linux (Server e Client)
- Ativar o htop para observar o consumo de CPU
- Instalar o Hydra através do comando: **sudo apt install hydra -y**

**Ataque realizado com a ferramenta Hydra:** https://tools.kali.org/password-attacks/hydra

**Comando:**

**Localização do target:** nmap -sP network/mask

**Brute force:** hydra -l tony -P /tmp/wordlist.txt -V -f  -t 4 192.168.0.54 ssh

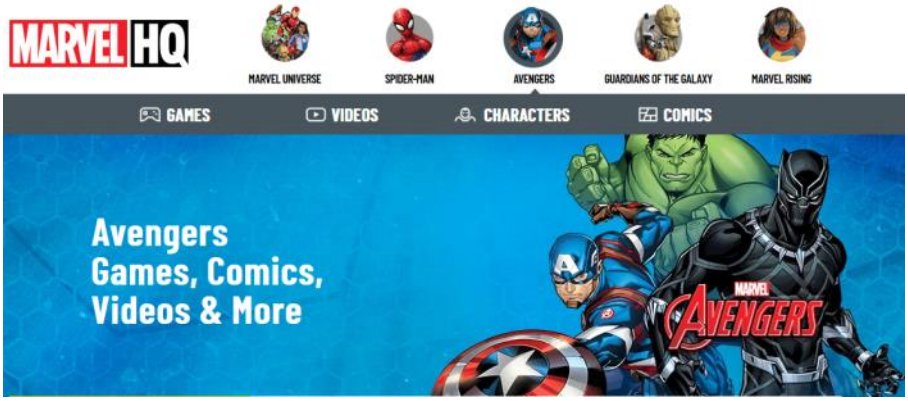
# Hora de trabalhar!

segunda-feira, 14 de junho de 2021 15:56

O *website* dos Avengers pode ser uma grande fonte de potenciais palavras para compor uma *wordlist* mais acertiva. Isso pode reduzir o tempo do ataque : )

**Problema:** Como extrair as palavras do *site* Avengers e construir a *wordlist* utilizando apenas comandos do Linux?

**Dica:** Claro, o uso de código em C pode ser a solução!

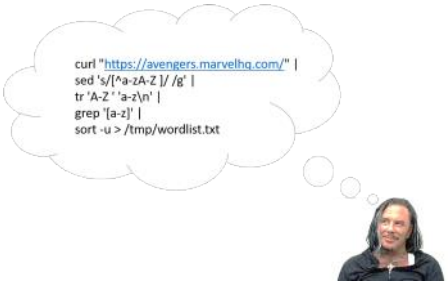



https://avengers.marvelhq.com/

# Como gerar uma wordlist?

sábado, 5 de junho de 2021 10:45

**Exemplo:** Aplicação de *pipelines* e redirecionamento para geração automática de uma *wordlist* em ambiente Linux. **Web Crawler**




**Comando:**

```
curl "https://avengers.marvelhq.com/" |
sed 's/[^a-zA-Z ]/ /g' |
tr 'A-Z ' 'a-z\n' |
grep '[a-z]' |
sort -u > /tmp/wordlist.txt
```

Como saber quantas palavras foram capturadas?

```
wc -w < wordlist.txt
```

Como incluir uma senha no fim da *wordlist*?

```
echo "insper" >> wordlist.txt
```

```
tail wordlist.txt
```

**Cuidado ao utilizar o redirecionador ">"!** Ele sobrescreve o arquivo já existente...

```
echo "insper" > wordlist.txt
```

Visualizando o conteúdo do arquivo wordlist.txt

```
cat wordlist.txt
```

Como localizar um *password* dentro do arquivo?

```
cat wordlist.txt | grep "anonymous"
```

Como tratar as mensagens de erro padrão?

```
ls -zz
```

```
ls -zz 2> erro.txt
```

**Utilizando pipes e redirecionadores**

Acrescentando um *password*, ordenando e gerando um novo arquivo.

```
echo "insper" >> wordlist.txt | sort -u wordlist.txt > wordlist_new.txt
```

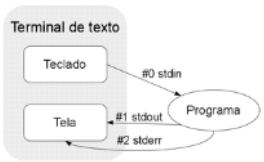
Vamos verificar se a palavra "insper" foi acrescentada.

```
cat wordlist_new.txt | grep "insper"
```

**Comandos**:

1. **curl** obtains the HTML contents of a web page (could use wget on some systems).
2. **sed** replaces all characters (from the web page's content) that are not spaces or letters, with spaces.
3. **tr** changes all of the uppercase letters into lowercase and converts the spaces in the lines of text to newlines (each 'word' is now on a separate line).
4. **grep** includes only lines that contain at least one lowercase character (removing any blank lines).
5. **sort** sorts the list of 'words' into alphabetical order, and the -u switch removes duplicates.

**Um pouco de teoria...**




Quando um processo quer manipular um arquivo ele usa o fd (*file descriptor*) "descritor de arquivo".

No Linux tudo é arquivo!

**Existem 3 descritores:**

**Entrada padrão (stdin)** é um *stream* (fluxo) para a entrada de texto.

Vinculada ao teclado... (teclado é a entrada padrão no Linux).

Descritor de arquivos número [**0**].

**Saída padrão (stdout)** é o *stream* para saída normal dos programas.

Vinculada ao terminal ou em uma janela de terminal.

Descritor de arquivos número [**1**].

**Erro padrão (stderr)** é o stream de saída de texto, só que é usado para exibir mensagens de erro.

Descritor de arquivos número [**2**].

# Pipeline em detalhes

sábado, 5 de junho de 2021 12:58

**Introdução:**

Sistemas Operacionais como o Linux possibilitam o uso de *pipelines*, ou seja, permitem o direcionamento da saída de um comando/processo para a entrada de outro.

As *pipes* permitem a comunicação entre processos.

No Linux podemos utilizar uma *pipe* "**|**" para conectar dois ou mais programas.

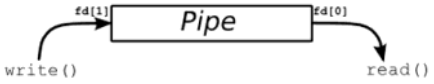
**Exemplo:** O diretório **/etc** no Linux tem uma grande quantidade de arquivos, o que dificulta a visualização.

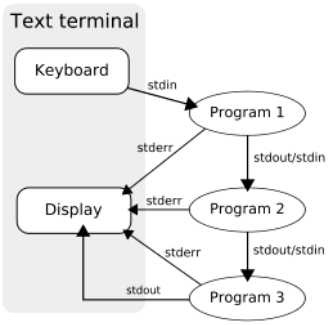
1. Ir até o diretório /etc com o comando cd /etc
2. Executar o comando ls -l
3. Executar o comando ls -l | grep "networks"
4. Analise a diferença entre os dois

**Obs.:** O comando **grep** só é executado quando o processo **ls -l** é finalizado!

**O que é uma Pipe?**

Pipe em inglês significa cano, tubo, ...




O **IPC** (*Inter Process Communication*) é usado quando processos com algum grau de parentesco precisam trocar dados entre si.

As *pipes* têm muita utilidade e também são muito utilizadas nos *shells* de alguns Sistemas Operacionais.

O Linux (*kernel*) através de suas chamadas de sistema prevê o uso de *pipeline*.

Esse tipo de *pipe* é chamada *half-duplex* (de uma via), pois as operações de leitura e escrita são mutuamente exclusivas, isto é, ou só se pode ler de uma *pipe* ou escrever.

**Curiosidade:** Pipes que suportam leitura e escrita ao mesmo tempo são chamados de ***stream pipes***.

**Vamos fazer um mergulho no código? Deep Diving**

**Exemplos de código em Linguagem C utilizando o compilador GCC**

**Exemplo 1:** Escrita de uma *string* na *pipe* pelo processo pai e leitura pelo processo filho, o programa utiliza os *syscalls*: pipe, **fork**, **close**, **write** e **read**.

C: > Users > fabio > C pipeline_ex1.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <unistd.h>
#define BUFFER 256
int main(void)
{
    int fd[2]; /*File descriptors*/
    pid_t pid; /*Variável para armazenar o pid*/
    if(pipe(fd)<0) {  /*Criando o Pipe */
        perror("pipe") ;
        return -1 ;
    }
    if ((pid = fork()) < 0)  /*Criando o processo filho*/
    {
        perror("fork");
        exit(1);
    }
    if (pid > 0) /*Processo Pai*/
    {
        close(fd[0]); /*fechar a leitura do Pipe*/
        char str[BUFFER] = "Aprendi a usar Pipes em C!";
        printf("\nString enviada pelo pai no Pipe: '%s'", str);
        write(fd[1], str, sizeof(str) + 1); /*Escrevendo a string no pipe*/
        exit(0);
    }
    else /* Processo Filho*/
    {
        char str_recebida[BUFFER];
        close(fd[1]); /*fechar a entrada de escrita do pipe*/
        read(fd[0], str_recebida, sizeof(str_recebida)); /*Lendo o que foi escrito no pipe*/
        printf("\nString lida pelo filho no Pipe : '%s'\n", str_recebida);
        exit(0);
    }
    return(0);
}
```

**Exemplo 2:** Construção de uma *pipeline* equivalente aos comandos Linux "**ls -i | sort -n**", utilizando os *syscalls*: **execl**, **fork**, **close**, **pipe** e **dup2**.

C: > Users > fabio > C pipeline_ex2.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
int main(void)
{
    int fd[2];
    pid_t childpid;
    pipe(fd);
    if ((childpid = fork()) == 0) { /*ls é o processo filho*/
        dup2(fd[1], STDOUT_FILENO);
        close(fd[0]);
        close(fd[1]);
        execl("/bin/ls", "ls", "-i", NULL);
        perror("Exec Error on ls");
    } else { /* sort é o o processo pai */
        dup2(fd[0], STDIN_FILENO);
        close(fd[0]);
        close(fd[1]);
        execl("/usr/bin/sort", "sort", "-n", NULL);
        perror("Exec Error on sort");
    }
exit(0);
}
```

# Check list

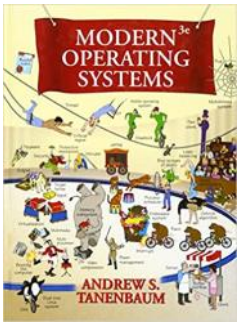
sexta-feira, 4 de junho de 2021 09:55

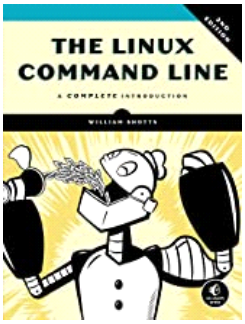
**O que aprendemos hoje?**

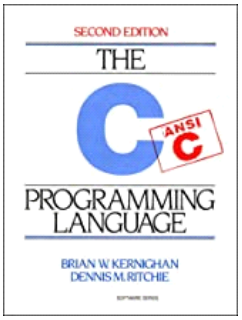
- [x] Identificar e aplicar os redirecionadores de *stream* (fluxo) utilizando os comandos "**<**", "**>**" e "**>>**";
- [x] Aplicar *pipelines* no terminal do Linux usando o comando "**pipe**";
- [ ] Mobilizar os conhecimentos já adquiridos sobre os *syscalls* para a construção de uma *pipeline* em linguagem C.

**Síntese:** *Os comandos de redirecionamento de fluxo e pipelines são de grande importância para o gerenciamento de sistemas operacionais, em especial aqueles que carecem de interface gráfica e que são gerenciados remotamente através do SSH (Secure Shell). Os sistemas operacionais executam a comunicação entre processos e manipulam arquivos seguindo padrões como o POSIX.*

**Referências:**



# Projeto em Dupla

segunda-feira, 14 de junho de 2021 16:02

**Tema:** Construção de um *web crawler* em Linguagem C

**Grupos:** duplas

**O que devemos entregar?** 

**Link do GitHub** contendo o **readme.md** com a descrição e manual do programa, **código desenvolvido em C** para a execução de um ***web crawler*** utilizando os ***syscalls***: **fork**, **dup2**, **execl**, **pipe** e **close**.

**Quando entregar?**

Dia **dd/mm** até às **hh:mm**

**Onde entregar?**

Plataforma