Criação de Malware

Professor: Killdary Aguiar de Santana

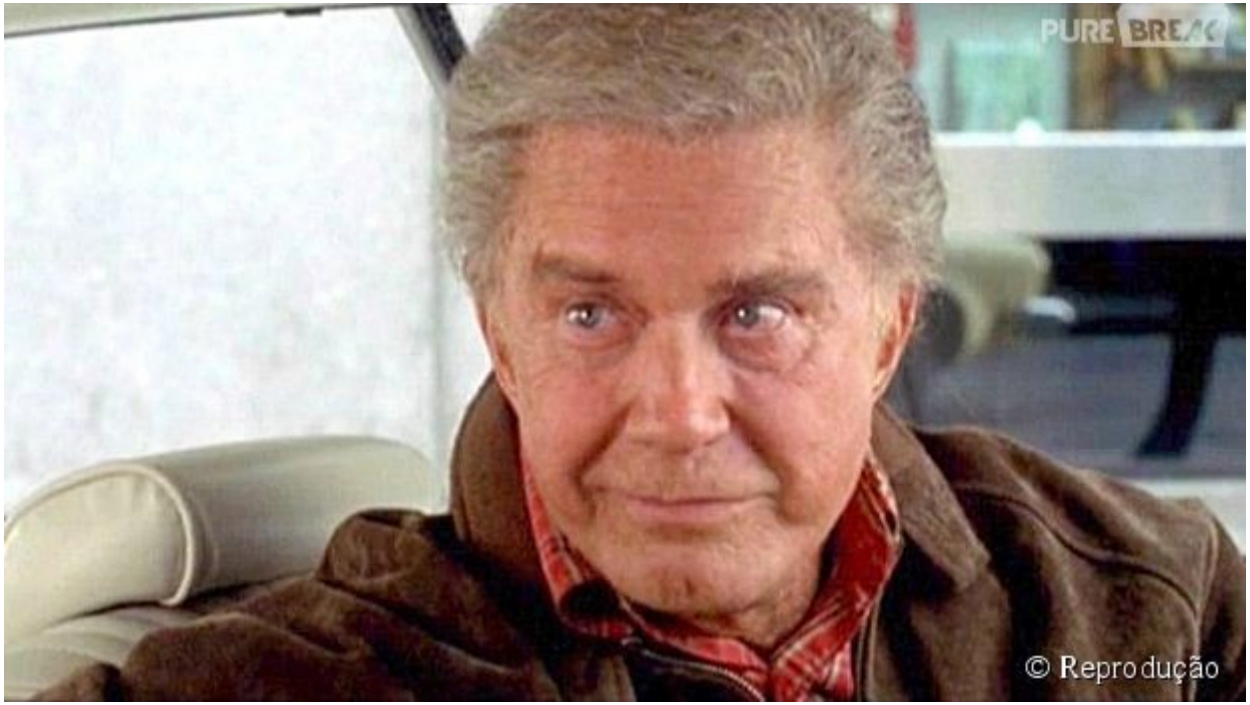
Quem sou eu...

- Graduado em Ciências da Computação pela UVA;
- Especialista em Administração e Segurança de redes;
- Perito Forense Computacional;
- Membro Fundador da Associação dos Peritos em Computação Forense do Estado do Ceará - APECOF-CE;
- Membro Fundador da empresa Tiwhu Tecnologia;
- Entusiasta de tecnologias embarcadas;
- Usuário e defensor de software livre desde 2010;
- Nerd nas horas vagas;

Oferecimento

Tiwhu Tecnologia

Atenção



Com Grandes Poderes Vem Grandes Responsabilidades

Chega de Papo vamos começar

Por que aprender a criar malwares?

Antes de aprendermos a nos **defender** temos que saber **atacar**.

Segurança

- Já existem diversos problemas no uso da computação hoje:
  - Invasões;
  - Ataques;
  - Malwares;
  - Botnets;
- O que iremos aprender será como criar um malware e entender o funcionamento e desenvolvimento por trás do software malicioso.

Malware

Malware

- O "malware", termo do inglês "malicious software"
- É um software destinado a infiltrar-se em um sistema de computador alheio de forma ilícita, com o intuito de causar alguns danos, alterações ou roubo de informações (confidenciais ou não).
- Ele pode aparecer na forma de:
  - código executável;
  - scripts de conteúdo ativo;
  - e outros softwares.

Malware

- Principais tipos de malwares
  - Vírus;
  - Worm;
  - Trojan (ou cavalo de troia);
  - Keylogger;
  - Screenlogger;
  - Spyware;
  - Adware;
  - Backdoor;
  - Exploits;
  - Sniffers;
  - Port Scanners;
  - Bot;
  - Rootkit;
  - Quantum.

O que iremos aprender

- Iremos aprender a criar um software malicioso que controla o computador da vítima;
- Utilizaremos a linguagem de programação Python para criar o malware;
- Para este projeto vocês irão aprender conceitos de:
  - Redes;
  - Criptografia;
  - Sistemas operacionais..

Malware

Redes de Computadores

Redes de Computadores

- **Redes de computadores** estabelecem a forma-padrão de interligar computadores para o compartilhamento de recursos físicos ou lógicos.
- Esses recursos podem ser definidos como:
  - Unidades de CD-ROM
  - Diretórios do disco rígido;
  - Impressoras;
  - Scanners;
  - Ect.

Redes de Computadores

- A Internet é um amplo sistema de comunicação que conecta muitas redes de computadores.
- Existem várias formas e recursos de vários equipamentos que podem ser interligados e compartilhados, mediante meios de:
  - Acesso;
  - Protocolos;
  - e Requisitos de Segurança.
- Os meios de comunicação podem ser:
  - linhas telefônicas;
  - Cabo;
  - Satélite;
  - Comunicação sem fios (wireless).

Redes

- Camadas OSI:

Redes

- Camadas TCP/IP:

Redes

- **Camada de aplicação**:
  - É responsável por prover serviços para aplicações de modo a separar a existência de comunicação em rede entre processos de diferentes computadores;
  - É a camada mais próxima ao usuário;
- Aplicações podem ser classificadas em 3 tipos:
  - Cliente-Servidor;
  - Peer-to-Peer;
  - Híbrida.

Redes

- Agora que conhecemos os conceitos básicos de redes que precisaremos vamos construir uma aplicação **Cliente-Servidor** em **Python**;
- Para iniciarmos usaremos o módulo **sockets** do Python que nos permitirá construir uma aplicação Cliente que faça requisição de dados para o Servidor.
- Um **socket** é o elo entre os processos do servidor e do cliente. Ele é a "porta" na qual os processos enviam e recebem mensagens.
- Segundo **JAMES F KUROSE**: "socket é a interface entre a camada de aplicação e a de transporte dentro de uma máquina".

Redes

- Cliente:

```
import socket
HOST = '127.0.0.1'
PORT = 5000
tcp = socket.socket
(socket.AF_INET,
socket.SOCK_STREAM)
dest = (HOST, PORT)
tcp.connect(dest)
menssagem = raw_input()
while menssagem <> '\x18':
    tcp.send (menssagem)
    msg = raw_input()
tcp.close()
```

- Servidor:

```
import socket
HOST = ''
PORT = 5000
tcp =
socket.socket(socket.AF_INET,socket.SOCK_
STREAM)
orig = (HOST, PORT)
tcp.bind(orig)
tcp.listen(1)
while True:
    con, cliente = tcp.accept()
    print 'Concetado por', cliente
    while True:
        msg = con.recv(1024)
        if not msg: break
        print cliente, msg
    print 'Finalizando conexao do
cliente', cliente
    con.close()
```

Redes

- Bom criamos uma aplicação
  - Cliente que envia uma mensagem e recebe a mesma mensagem do Servidor;
  - Servidor que recebe a aplicação do Cliente e a retorna para o mesmo.
- Já que conseguimos esta etapa vamos passar um arquivo entre o Cliente e o Servidor.

Redes

- Cliente:

```
import socket
HOST = 'localhost'
PORT = 5000


s =
socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
s.connect((HOST,PORT))
arq = open('ex1.jpg', 'r')

for i in arq.readlines():
    s.send(i)
arq.close()
s.close()
```

- Servidor:

```
import socket
HOST = ''
PORT = 5000
s = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
s.bind((HOST, PORT))
s.listen(1)
conn, addr = s.accept()
arq = open('rcv.jpg', 'w')
while 1:
    dados = conn.recv(1024)
    if not dados:
        break
    arq.write(dados)
arq.close()
conn.close()
```

Redes

- As aplicações anteriores permitem termos uma ideia bem simples do envio e recebimento de dados entre dois computadores, porem o Servidor só pode conectar um único Cliente por vez, se quiséssemos conectar diversos Clientes teríamos que usar conceitos de **programação concorrente**, ou seja **Threads**.

Malware

Criptografia

Criptografia

- **Criptografia** é o estudo a aplicação de técnicas para comunicação e armazenamento seguro de dados em sistemas computacionais(entre outros);
- A palavra Criptografia vem do grego "Kryptós", "oculto" e "Graphein", "escrita", portanto "escrita oculta (ou secreta)".

Criptografia

- Termos-chave:
  - **Texto plano/puro**: dados não-encriptados;
  - **Texto cifrado**: dados encriptados;
  - **Chave(key)**: dados utilizados para encriptar um texto plano, ou desencriptar um texto cifrado;
  - **Algoritmo de Criptografia**: Método matemático empregado para encriptar/desencriptar dados com o uso das chaves de criptografia.

Criptografia

- **Encriptação** é a conversão de dados legíveis para um formato ilegível (texto cifrado) por pessoas não-autorizadas, usando uma chave e um algorítimo criptográfico;
- Seu objetivo é proteger a privacidade ao armazenarmos dados ou trocamos informações comoutras pessoas.
- O receptor da mensagem encriptada pode decriptá-la e ler seu conteúdo, no formato original.

Criptografia

- Criptoanálise é o processo de transformação de dados cifrados em dados legíveis sem que se conheça a chave de encriptação;

  Portanto, trata-se de "quebrar" a encriptação dos dados para obter acesso ao conteúdo das mensagens;

  Porém com o intuito de descobrir falhas nos algoritmos para torná-los mais seguros, validá-los ou descartá-los

  Ex.: Criptografia WEP

Criptografia

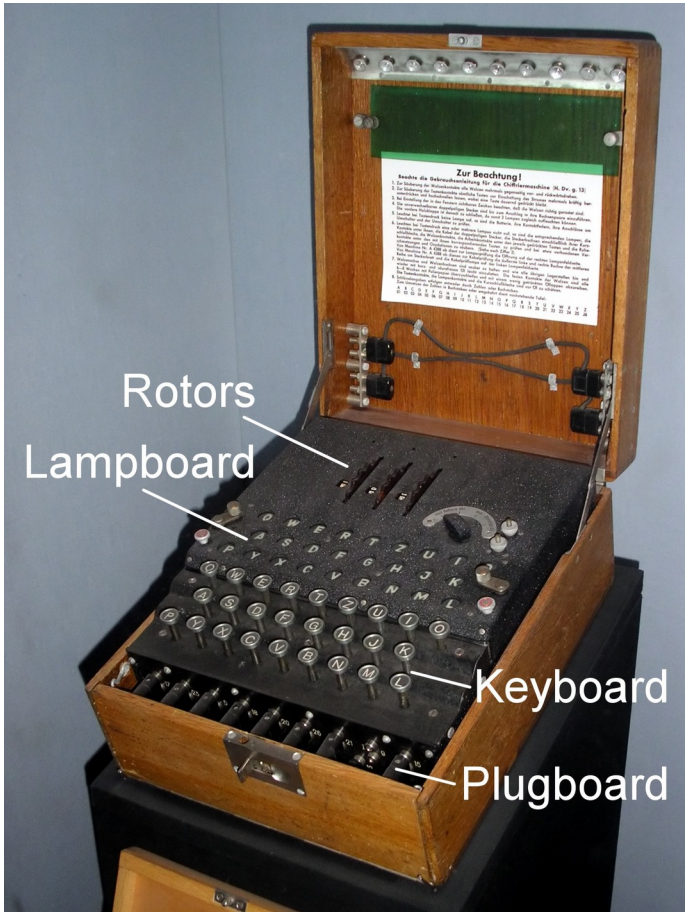
- Basicamente, existem 3 tipos de criptografia:
  - Criptografia de Chave Privada ou Simétrica;
  - Criptografia de Chave Pública ou Assimétrica.
  - Hash.

Criptografia

- A Criptografia de Chave Privada utiliza uma única chave;

  O emissor utiliza essa chave para encriptar a mensagem, e o receptor utiliza a mesma chave para decriptá-la(chave compartilhada - shared key);

- Por utilizar a mesma chave na encriptação e decriptação, trata-se de uma técnica de Criptografia simétrica.



Criptografia

- A **Criptografia de Chave Pública** utiliza uma duas chaves distintas, de modo a obtermos comunicação segura através de canais de comunicação inseguros.

  Trata-se de uma técnica de Criptografia Assimétrica pelo fato de usar um par de chaves diferentes;

  Cada participante possui uma chave pública e uma chave privada. A chave privada é secreta e só o proprietário a conhece, ao passo que a chave pública é compartilhada com todos que se comunicarem conosco.

Criptografia

- Vamos conhecer 2 tipos de Criptografia de Chave Privada:
  - DES;
  - AES.

Criptografia

- Para os exemplos de criptografia no Python utilizaremos o modulo Crypto para a instalação abra o terminal e digite o código abaixo:

```
sudo pip install pycrypto
```

- Ou através do link:
  - `https://www.dlitz.net/software/pycrypto/`
- Debian/Ubuntu
  - `sudo apt-get install python-crypto`

Criptografia

- DES:
  - Data Encryption Standard
  - Criado pela IBM em 1977
  - Chaves de 56 bits.
    - 72 quatrilhões de combinações.
  - Valor alto, mas não para um computador potente.
  - Em 1997, foi quebrado por técnicas de "força bruta".
  - Este algorítimo possui um tamanho de chave de 8 bytes
  - E sua criptografia só irá funcionar em texto de tamanho igual ou múltiplo de 8.

Criptografia

- Exemplo de DES:

```
from Crypto.Cipher import DES
chave = "12345678"
des = DES.new(chave, DES.MODE_ECB)
texto_claro = "abcdfghi"
texto_cifrado = des.encrypt(texto_claro)
print '[+] Cifrado: %s' % texto_cifrado
print '[+] Decifrado: %s' %
des.decrypt(texto_cifrado)
```

Criptografia

- AES:
  - Advanced Encryption Standard.
    - Padrão de Criptografia Avançada.·
  - Anunciado no final de 2001.
  - Novo padrão de criptografia adotado pelo governo americano em 2002
  - Usa blocos de 128 bits:
    - chaves de 128, 192 e 256 bits.·
  - Para quebrá-lo um computador potente levaria milhões de anos.
  - Este algorítimo possui um tamanho de chave de 16, 24 ou 32 bytes.
  - E sua criptografia só irá funcionar em texto de tamanho igual ou múltiplo de 16.

Criptografia

- Exemplo de AES:

```
from Crypto.Cipher import AES
chave = "0123456789ABCDEF"
aes = AES.new(chave, AES.MODE_ECB)
texto_claro = "DEADBEEFDEADBEEF"
texto_cifrado = aes.encrypt(texto_claro)
print '[+] Cifrado: %s' % texto_cifrado
print '[+] Decifrado: %s' %
aes.decrypt(texto_cifrado)6501
```

Criptografia

E se eu quisesse criptografar um arquivo? È possível?

- A resposta é SIM, é possível e vamos criptografar neste momento uma imagem, no qual podemos fazer o mesmo processo para qualquer tipo de arquivo

Criptografia

- Criptografar arquivos:

```
import base64
from Crypto.Cipher import AES

chave = "0123456789ABCDEF"
aes = AES.new(chave, AES.MODE_ECB)
arquivo = raw_input()
arq_entrada = open(arquivo, "r")
arq_entrada = arq_entrada.read()
cryptoSaida = arq_entrada+'#'*(16-len(arq_entrada)%16)
texto_cifrado = base64.b32encode(aes.encrypt(cryptoSaida))
titulo_novo=base64.b32encode(aes.encrypt(arquivo+'#'*(16-len(arquivo)
%16)))
arq_saida = open(titulo_novo,'w')
arq_saida.write(texto_cifrado)
arq_saida.close()
```

Criptografia

- Descriptografar arquivos:

```
import base64
from Crypto.Cipher import AES

chave = "0123456789ABCDEF"
arquivo = raw_input()
arq_entrada = open(arquivo, "r")
arq_entrada = base64.b32decode(arq_entrada.read())
titulo_antigo=aes.decrypt(base64.b32decode(arquivo))
titulo_antigo=titulo_antigo.rstrip('#')
texto_recuperado=aes.decrypt(arq_entrada)
texto_recuperado=texto_recuperado.rstrip('#')
arq_saida2 = open(titulo_antigo,"w")
arq_saida2.write(texto_recuperado)
```

Malware

Sistemas Operacionais

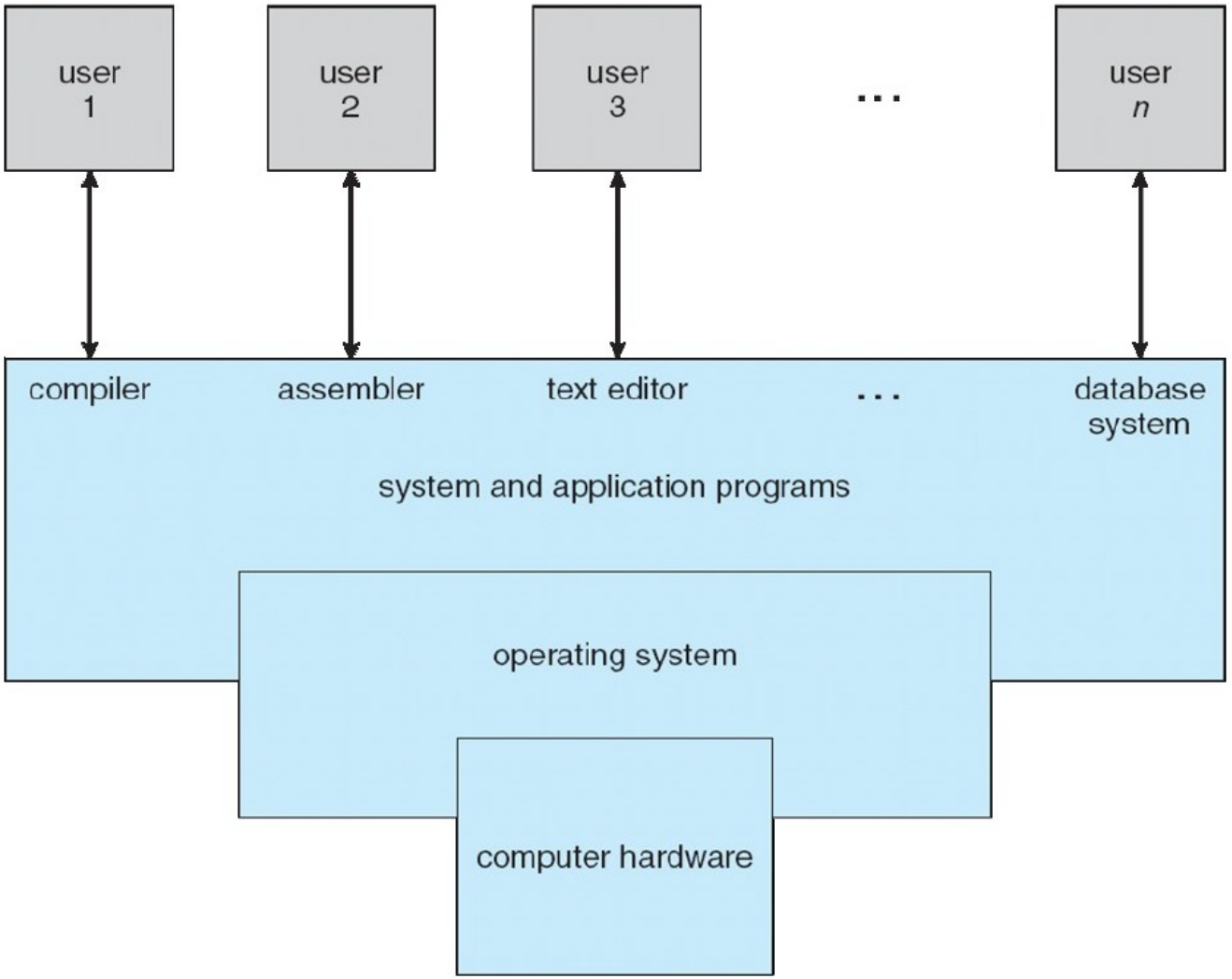
Sistemas Operacionais

- O Sistema Operacional(SO) é um software que opera entre o hardware e os programas voltados ao usuário;
- Os objetivos de um SO:
  - Executar os programas do usuário e tornar mais fácil a resolução de problemas dos mesmos;
  - Tornar o computador uma ferramenta conveniente para o uso;
  - Usar o hardware do computador de maneira eficiente;

Sistemas Operacionais

- O computador pode ser dividido em 4 componentes:
  - Hardware - fornece recursos básicos de computação
    - CPU, memória, dispositivos de E/S(I/O devices);
  - Sistema Operacional
    - Controla e coordena o uso do hardware entre as diversas aplicações e o usuário;
  - Programas - definem as formas em que os recursos do sistemas são usados para resolver os problemas de computação dos usuários
    - Compiladores, navegadores, banco de dados, jogos, vídeos, processadores de texto, etc.
  - Usuários
    - Pessoas, máquinas e outros computadoes.

Sistemas Operacionais



Sistemas Operacionais

- Os SO's proporcionam um ambiente para a execução de programas e serviços para aplicações e usuários
- Serviços aos Usuários:
  - Interface do usuário(User Interface,UI)
    - UI, Command-Line (CLI), Graphics User Interface (GUI), Batch
  - **Execução do programa** -Carregan um programa na memoria e o executa, a execução final pode ser normal ou anormal (indicando erro)
  - **Operações de E/S (I/O operations)** - Um programa em execução pode exigir E/S, o que envolve um arquivo ou um periférico de E/S

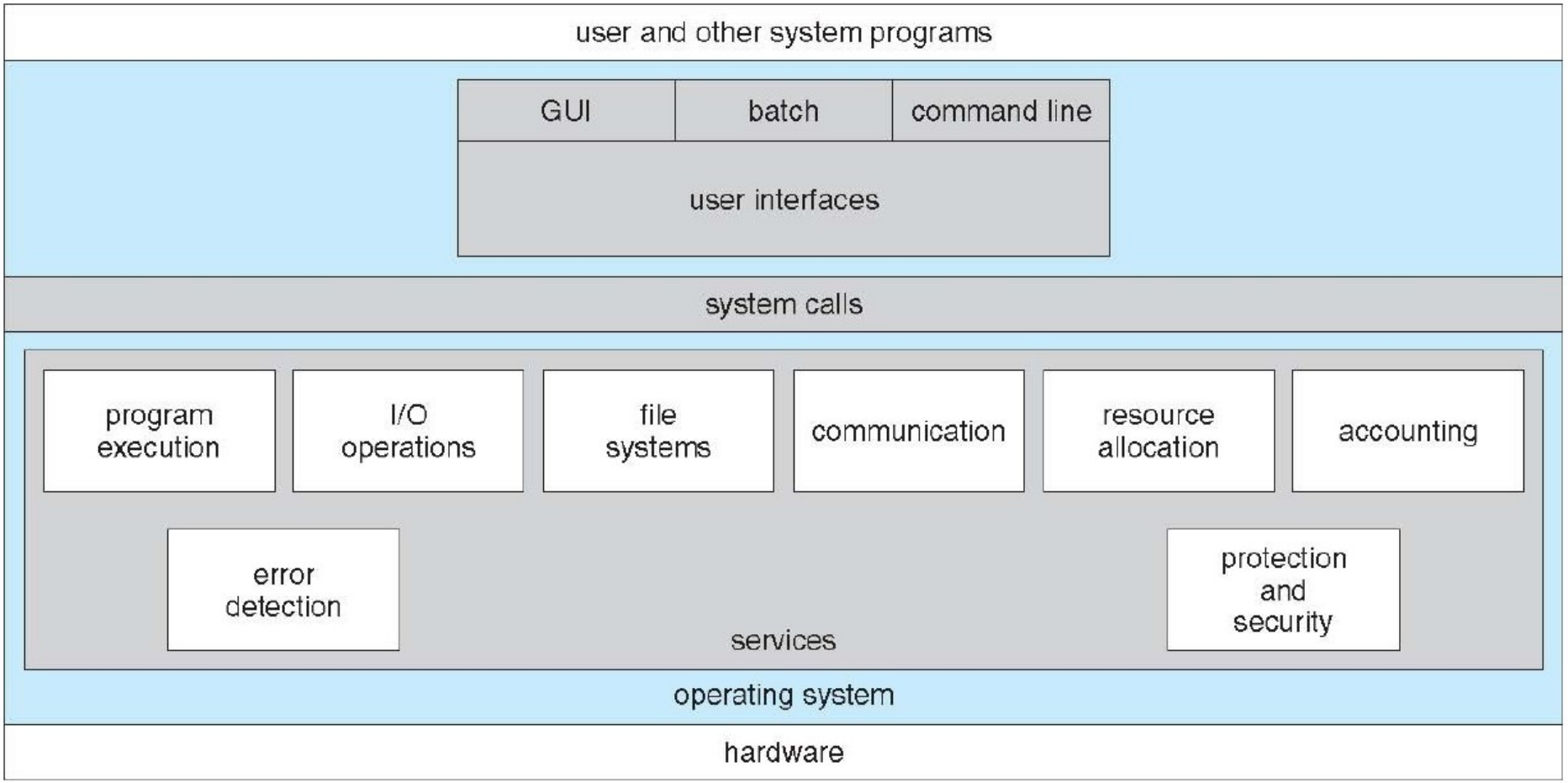
Sistemas Operacionais

- Serviços ao Usuário (Cont.):
  - **Manipulação de Sistemas de Arquivos (File-system manipulation)** - Programas necessitam ler e escrever arquivos e diretórios, criar e exclui-los. Pesquisá-los, listar informações, gerenciar permissões.
  - **Comunicação** – Processos podem trocar informações, no mesmo computador ou entre computadores em um rede
    - As comunicações podem ser via memória compartilhada ou através de troca de mensagens (troca de pacotes no SO)
  - **Detecção de Erros** – SO's precisam estar constantemente cientes dos possíveis erros
    - Podem ocorrer na CPU e na memória do hardware, em periféricos de E/S, no programa do usuário, etc
    - Para cada tipo de erro, os SO's devem tomar medidas adequadas para garantir a computação correta e consciente
    - Instalações de depuração podem aumentar consideravelmente as habilidades do usuário e programador para o uso eficiente do sistema

Sistemas Operacionais

- Serviços do Sistema:
  - Para assegurar o funcionamento eficiente do sistema em si através do compartilhamento de recursos
  - **Alocação de Recursos** - Quando vários usuários ou vários trabalhos devem ser atribuídos a cada um deles
    - Muitos tipos de Recursos - ciclos de CPU, memória principal, armazenamento de arquivos, didpositivos de E/S.
  - **Accounting** - Para manter o controle de quais e tipos de recursos do computador os usuários usam
  - **Proteção e Segurança** - Os proprietários de informações armazenadas em multi usuário ou sistema de computador em rede pode querer cotrolar o usu dessas informações, processos concorrentes não devem interferir uns com os outros
    - **A Proteção** implica garantir que todo o acesso aos recursos do sistema é controlado
    - **A Segurança** do sistema contra estranhos requer autenticação do usuário, estende-se a defender os dispositivos E/S externos de tentativas de acesso inválidos

Sistemas Operacionais



Sistemas Operacionais

- Bom já demos uma nivelada no conceito de SO;
- Agora iremos ver como o **Python** manipula o SO;
- A manipulação do SO através do Python é feita pelo módulo **OS**.
- O módulo **OS** em Python fornece uma maneira fácil de usar funcionalidades se SO independe do sistema operacional;
- As funções que o módulo OS fornece lhe permite interagir com o sistema operacional subjacente que Python está sendo executado em - seja Windows, Mac ou Linux.

Sistemas Operacionais

- Bom vamos conhecer as principais funções deste módulo;
- Abram o console Python e digitem o comando abaixo

```
import os
```

- Depois de importar este módulo vamos testar vários comandos deste módulo;

Sistemas Operacionais

- **os.system('comando')**: Executar um comando shell;
- **os.environ['var']=x**: Cria uma variável de ambiente no sistema;
  - A função **environ** também permite listar, alterar e mostrar as variáveis de sistemas. Ex: **os.environ.keys()** lista todas as variáveis de sistema do Linux;
- **os.getcwd()**: Retorna o diretório de trabalho atual;
- **os.chroot(caminho)**: Altera a pasta atual na qual o Python se encontra;
- **os.getgid()**: Devolver o ID do grupo a quem o processo atual(Python) pertence;
- **os.getuid()**: Devolver o ID do usuário a quem o processo atual(Python) pertence;
- **os.getpid()**: Retorna o ID do processo atual(Python);
- **os.umask(máscara)**: Define novas permissões para um arquivo e retorna as permissões antigas;

Sistemas Operacionais

- **os.uname()**: retorna informações que identifica o sistema operacional atual.
- **os.listdir(caminho)**: Lista o conteúdo de uma pasta do sistema
- **os.path.isfile(caminho)**: Verifica se o caminho passado é um arquivo;
- **os.path.isdir(caminho)**: verifica se o caminho passado é uma pasta;
- **os.path.getsize(caminho)**: retorna o tamanho de um arquivo
- **os.mkdir(caminho)**: Cria uma pasta no caminho especificado.
- **os.makedirs(caminho)**: Cria arquivos de forma recursiva em uma pasta;
- **os.remove(caminho)**: Remove (excluir) um arquivo.
- **os.removedirs(caminho)**: Remove pastas de forma recursiva.
- **os.rename(src, dst)**: Renomeia o arquivo ou diretório src para dst.
- **os.rmdir(caminho)**: Remove uma pasta;

Sistemas Operacionais

- os.walk(): percorre de forma recursiva uma pasta

```
import os
for root, dirs, files in os.walk(".",
topdown=False):
    for name in files:
        print(os.path.join(root, name))
    for name in dirs:
        print(os.path.join(root, name))
```

Sistemas Operacionais

- **os.login()**: retorna o nome do usuário que executou o Python;
- Utilizando o módulo **pwd** recuperamos dados relacionados ao arquivo de senha(/etc/passwd) do Linux
- **pwd.getpwuid(ID)**: retorna a linha do arquivo pwd correspondente ao ID informado;
- **pwd.getpwnam(nome)**: retorna a linha do arquivo pwd correspondente ao nome de usuário informado;
- **pwd.getpwall()**: retorna todo o conteudo do arquivo ;

Sistemas Operacionais

- Vamos ver uma forma ainda mais simples de listar arquivos utilizando o modulo **glob**
- **glob.glob(caminho)**: lista o conteúdo de uma pasta, cada arquivo é apresentado com seu caminho inteiro;

Sistemas Operacionais

```
import os
import glob
pastaUsuario = os.environ['HOME']
for arquivos in glob.glob(pastaUsuario+"/*"):
  print arquivos

for arquivos in os.listdir(pastaUsuario):
  print arquivos

for root, dirs, files in os.walk(pastaUsuario, topdown=False):
    for name in files:
        arquivo=(os.path.join(root, name))
        print arquivo
        print name
    for name in dirs:
        print(os.path.join(root, name))
```

Sistemas Operacionais

- Exercício:
  - Crie, através da interface gráfica, uma pasta no diretório Imagens do seu usuário com o nome de exercicio, coloque 3 imagens dentro desta pasta. Agora crie um script que acesse essas 3 imagens e as criptografe.
  - Tempo: 15 minutos.

Sistemas Operacionais

- Exercício:
  - Aproveitando a solução anterior crie um código em Python para descriptografar os arquivos
  - Tempo: 15 minutos.

- Final:
  - Vamos para a etapa final primeiro crie um uma aplicação Cliente/Servidor na qual o Servidor envie mensagens para o Cliente e o mesmo as responda com algum texto;

- Final:
  - Com o nosso Cliente respondendo um texto qualquer, faça com que o Servidor envie um comando o Cliente o execute e retorne a resposta para o Servidor;

Sistemas Operacionais

- Final:
  - Passando esta etapa criptografe os dados entre o Cliente e o Servidor, para que a trafego de informações entre eles seja segura;

Sistemas Operacionais

- Final:
  - Parabéns você acaba de desenvolver um botnet com conceitos básicos de Redes, SO e Criptografia. Agora vocês possuem conhecimento suficiente para criar um vírus que sequestre os dados da de alguém e pedir resgate.

Malware

Dúvidas?

Obrigado

Referência

- https://pt.wikipedia.org/wiki/Malware
- https://pt.wikibooks.org/wiki/Redes_de_computadores/Camada_de_aplica%C3%A7%C3%A3o
- https://www.youtube.com/watch?v=cWld3rMD7Wk
- http://tecnologiadarede.webnode.com.br/news/noticia-aos-visitantes/
- https://docs.python.org/
- Mendes, D. Redes de Computadores -Teoria e Prática
- Oreilly.- Sarker, M. Python Network Programming
- Oreilly.- Python for Unix and Linux System Administration
- Oreilly.- Essential System Administration
- Oreilly.- Learning Python
- Oreilly.- Linux Networking Cookbook
- Oreilly.- Linux Security Cookbook
- Oreilly.- Mac OS X for Unix Geeks
- Oreilly.- Programming Python
- Oreilly.- Python Cookbook