Flávio Ferreira da Cunha

Portfólio de soluções com Linux, programas FOSS (*Free and Open Source Software*) *Shell Script* e outras tecnologias para resolução de problemas de Cybersecurity e monitoração de serviços.

São Paulo

2025

RESUMO

Este portfólio tem como objetivo mostrar alguns projetos que foram idealizados por mim com a utilização da linguagem *Shell Script* e outras tecnologias a fim de solucionar problemas, obter informações importantes e também para automatizar tarefas repetitivas do dia a dia.

Como profissional da área de segurança da informação entendo que toda sorte de conhecimento adquirido ao longo dos anos pode ser utilizado para compor um projeto ou resolver um problema. Por exemplo, recentemente tive que elaborar uma solução para exibir em uma interface *web* os dez países que mais executavam varreduras de portas no *firewall* Linux de um cliente. Para realizar este trabalho foi utilizado o sistema de detecção de intrusão baseado em *host Portsentry*, o servidor *web* Apache, as linguagens PHP, *Java script* e *Shell Script, o framework CSS Bootstrap,* uma API do *Google Charts* e o banco de dados MariaDB.

Muitas vezes as soluções que criamos ou os problemas que resolvemos fazem parte de um conjunto sistêmico composto por várias partes e entender como cada um delas funciona é fundamental para a compreensão apropriada do todo.

Lista de figuras

Sumário

1 PROJETO 1

1.1 Interface *web* para exibição de varreduras em *firewall* de borda classificadas de acordo com o País.

Neste projeto foram utilizados o ids de *host Portsentry*, o servidor *web* Apache, as linguagens PHP, *JavaScript* e *Shell Script, o framework CSS Bootstrap,* uma API do *Google Charts* e o banco de dados MariaDB.

O arquivo de configuração do *Portsentry* foi configurado para executar o *script shell* scan-detect.sh toda vez que detectar uma varredura, conforme mostra a figura 1.

O parâmetro $TARGET$ captura o endereço ip da origem da varredura e $PORT$ a porta de serviço alvo.

Figura 1 - Linha que indica que o *Portsentry* deve executar o *script shell scan-detect.sh* em casos de varreduras.

```
KILL_RUN_CMD="/bin/scan-detect.sh $TARGET$ $PORT$"
```

Fonte: Própria.

O código em shell que captura as varreduras do *Portsentry* e armazena em uma base de dados MariaDB é exibido na figura 2.

Figura 2 - Código em *Shell* que captura as varreduras do *Portsentry* e armazena em uma base de dados no MariaDB.

```
#!/bin/bash

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"

if [ $# -ne 2 ] ; then
        echo "Erro: Use $0 [IP] [Porta]"
        exit 1
fi

DATA=$(date +%Y%m%d)
HORA=$(date +%H%M%S)

Pais_geo=`geoiplookup $1 | cut -d: -f2 | tr -d '[:space:]' | cut -d, -f2`

# Inserindo dados da varredura na tabela scan do banco MariaDB

mysql -e "INSERT INTO scan (end_ip, porta, pais, data, hora) VALUES ('$1', $2,
'$Pais_geo',  $DATA, $HORA)" portsentry

# Enviando a notificacao por E-mail
#echo -e "\n $DATA \n $HORA\n\n IP $1 Fazendo SCAN na Porta $2 \n\n
$Pais_geo \n\n"    | mail -s "Alerta Scan! IP: $1 $Pais_geo Porta: $2"
fcunha@xxx.com.br

# Bloqueando em mangle
iptables -t mangle -I PREROUTING -s $1 -j DROP
```

Fonte: Própria.

A aplicação PHP deve conectar na base e efetuar uma "*query*" na tabela *scan* para obter os dez primeiros países com o maior número de varreduras, conforme figura 3.

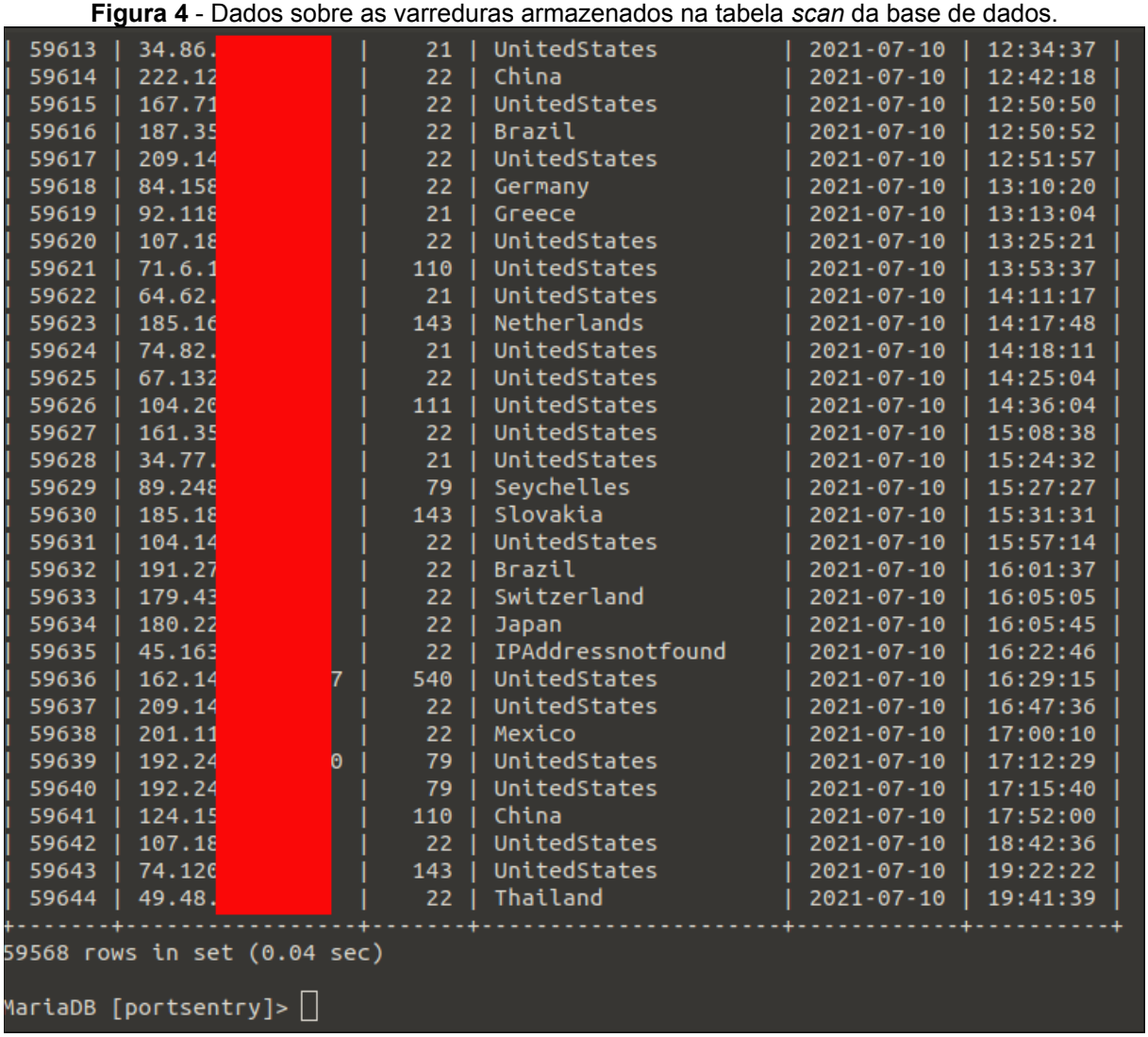
Figura 3 - Código em PHP para classificar os dez primeiros países com o maior número de varreduras.

```
<?php

$conexao = mysqli_connect('127.0.0.1', 'varredura', 'senha', 'portsentry');
mysqli_set_charset($conexao,"utf8");

function listaScan($conexao) {
    $listas = array();
    $resultado = mysqli_query($conexao, "select pais,COUNT(*) from scan GROUP
BY pais ORDER BY COUNT(*) DESC LIMIT 10");

    while($lista = mysqli_fetch_assoc($resultado)) {
        array_push($listas, $lista);
    }

    return $listas;
}

?>
```

Fonte: Própria.

A figura 4 exibe como os dados das varreduras são armazenados na tabela *scan* da base de dados.

Figura 4 - Dados sobre as varreduras armazenados na tabela *scan* da base de dados.



59613	34.86.	21	UnitedStates	2021-07-10	12:34:37
59614	222.12	22	China	2021-07-10	12:42:18
59615	167.71	22	UnitedStates	2021-07-10	12:50:50
59616	187.35	22	Brazil	2021-07-10	12:50:52
59617	209.14	22	UnitedStates	2021-07-10	12:51:57
59618	84.158	22	Germany	2021-07-10	13:10:20
59619	92.118	21	Greece	2021-07-10	13:13:04
59620	107.18	22	UnitedStates	2021-07-10	13:25:21
59621	71.6.1	110	UnitedStates	2021-07-10	13:53:37
59622	64.62.	21	UnitedStates	2021-07-10	14:11:17
59623	185.16	143	Netherlands	2021-07-10	14:17:48
59624	74.82.	21	UnitedStates	2021-07-10	14:18:11
59625	67.132	22	UnitedStates	2021-07-10	14:25:04
59626	104.20	111	UnitedStates	2021-07-10	14:36:04
59627	161.35	22	UnitedStates	2021-07-10	15:08:38
59628	34.77.	21	UnitedStates	2021-07-10	15:24:32
59629	89.248	79	Seychelles	2021-07-10	15:27:27
59630	185.18	143	Slovakia	2021-07-10	15:31:31
59631	104.14	22	UnitedStates	2021-07-10	15:57:14
59632	191.27	22	Brazil	2021-07-10	16:01:37
59633	179.43	22	Switzerland	2021-07-10	16:05:05
59634	180.22	22	Japan	2021-07-10	16:05:45
59635	45.163	22	IPAddressnotfound	2021-07-10	16:22:46
59636	162.14 7	540	UnitedStates	2021-07-10	16:29:15
59637	209.14	22	UnitedStates	2021-07-10	16:47:36
59638	201.11	22	Mexico	2021-07-10	17:00:10
59639	192.24 0	79	UnitedStates	2021-07-10	17:12:29
59640	192.24	79	UnitedStates	2021-07-10	17:15:40
59641	124.15	110	China	2021-07-10	17:52:00
59642	107.18	22	UnitedStates	2021-07-10	18:42:36
59643	74.120	143	UnitedStates	2021-07-10	19:22:22
59644	49.48.	22	Thailand	2021-07-10	19:41:39

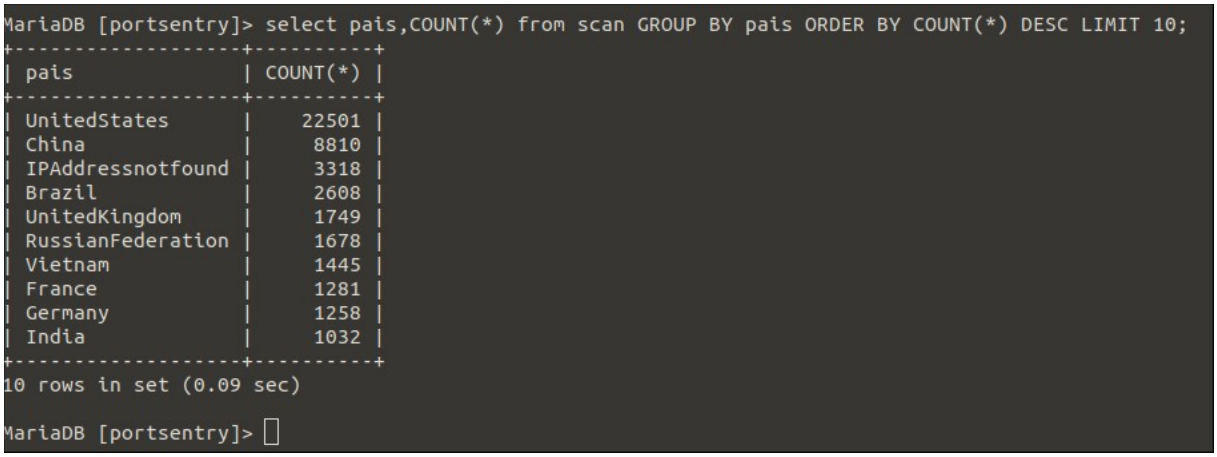
59568 rows in set (0.04 sec)

MariaDB [portsentry]>

Fonte: Própria.

A figura 5 apresenta o comando *SELECT* usado para contabilizar os dez países com maior número de ocorrências de varreduras.

Figura 5 - Comando SELECT usado para mostrar os dez países com maior ocorrências de varreduras.



```
MariaDB [portsentry]> select pais,COUNT(*) from scan GROUP BY pais ORDER BY COUNT(*) DESC LIMIT 10;
+-------------------+----------+
| pais              | COUNT(*) |
+-------------------+----------+
| UnitedStates      |    22501 |
| China             |     8810 |
| IPAddressnotfound |     3318 |
| Brazil            |     2608 |
| UnitedKingdom     |     1749 |
| RussianFederation |     1678 |
| Vietnam           |     1445 |
| France            |     1281 |
| Germany           |     1258 |
| India             |     1032 |
+-------------------+----------+
10 rows in set (0.09 sec)

MariaDB [portsentry]>
```

Fonte: Própria.

A figura 6, por sua vez, mostra o código *JavaScript* utilizado para integração com a *API do Google Charts.*

Figura 6 - Código em *Java Script* para integração com o *Google Charts*.

```
<!--<script type="text/javascript"src="https://www.gstatic.com/charts/loader.js"></
script>-->
<script type="text/javascript" src="loader.js"></script>
<script type="text/javascript">

// Load the Visualization API and the corechart package.
google.charts.load('current', {'packages':['corechart']});

// Set a callback to run when the Google Visualization API is loaded.
google.charts.setOnLoadCallback(drawChart);
```

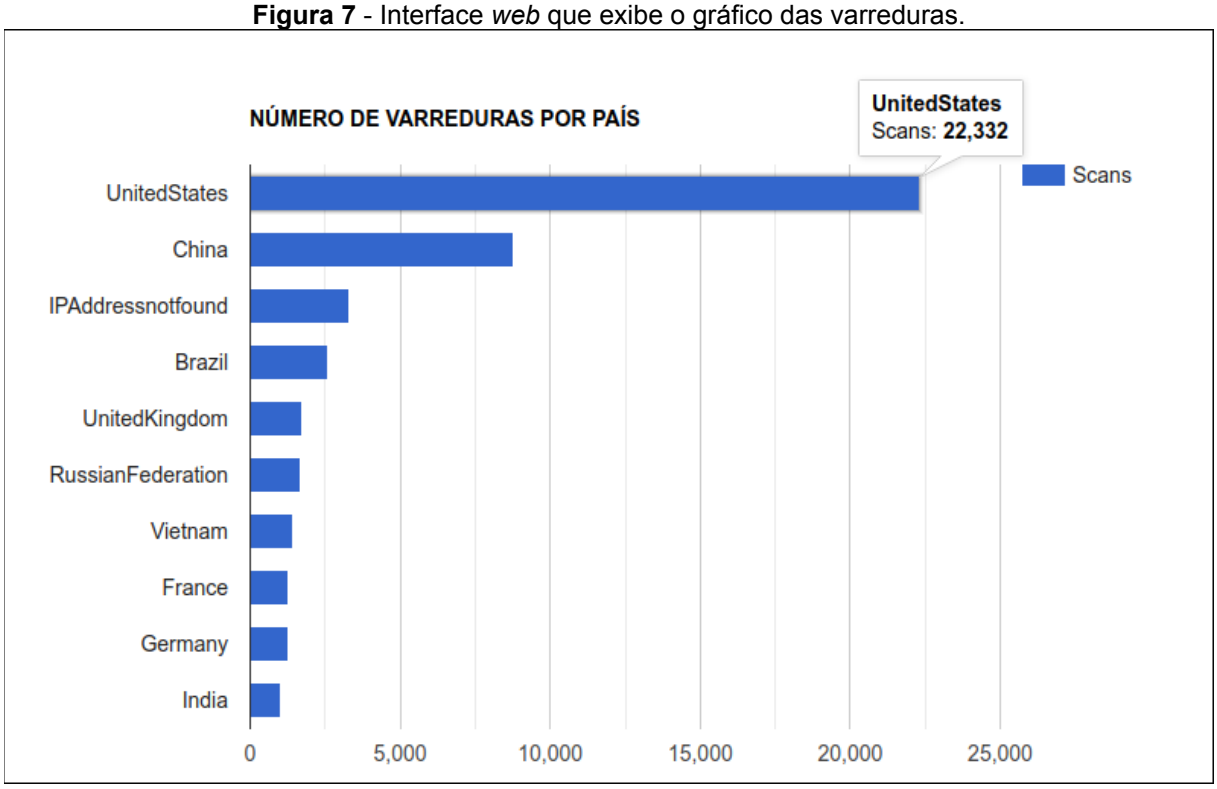
```
// Callback that creates and populates a data table,
// instantiates the pie chart, passes in the data and
// draws it.
      function drawChart() {

      // Create the data table.
      var data = new google.visualization.DataTable();
      data.addColumn('string', 'Topping');
      data.addColumn('number', 'Scans');
      <?php
      $users = listaScan($conexao);
      foreach($users as $user) :
      ?>

      data.addRows([
        ['<?= $user['pais'] ?>', <?= $user['COUNT(*)']?>]

      ]);
     <?php
      endforeach
      ?>

      // Set chart options
      var options = {'title':'NÚMERO DE VARREDURAS POR PAÍS',
                     'width':800,
                     'height':600};

      // Instantiate and draw our chart, passing in some options.
                                                  var chart = new
google.visualization.BarChart(document.getElementById('chart_div'));
      chart.draw(data, options);
    }
```

```
</script>
```

Fonte: Própria.

Por fim, a figura 7 exibe o gráfico de varreduras na interface *web.*

Figura 7 - Interface *web* que exibe o gráfico das varreduras.



Fonte: Própria.

2 PROJETO 2

2.1 *Shell Script* para enviar *backup* de aplicações para um *bucket* S3 da *Amazon AWS* após integração com a ferramenta *AWS CLI*.

A figura 8 mostra o *shell script* para envio de *backup* de aplicações para um *bucket do Amazon S3*.

Figura 8: *Shell script* para envio de *backup* de aplicações para *bucket Amazon S3.*

```
#!/bin/bash

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"
DATA=$(date +%d-%m-%y)
DATA_SEMANA=$(date +%A)
MARCA_DO_DIA=$(date +%w)
DIR_BACKUP="/var/backup_web"
BUCKET="s3://nfe-xxxx"
MARCA_DO_DIA=$(date +%w)
SERVER="235"

### Array com o nome das aplicacoes ####
Aplicacao=("app1" "app2" "app3" "app4" "app5" "app6")

# Entrando no diretorio de backup

cd $DIR_BACKUP

> /tmp/syslog_sys

for((i=0; i<${#Aplicacao[@]}; i++)) ; do

    # Enviando as aplicacoes para o bucket da Amazon s3
```

```
        if aws s3 cp $DIR_BACKUP/$MARCA_DO_DIA-${Aplicacao[$i]}.$SERVER-
$DATA.tar.gz $BUCKET > /dev/null 2>&1 ; then
                        echo -e "Arquivo $MARCA_DO_DIA-${Aplicacao[$i]}.$SERVER-
$DATA.tar.gz copiado para o bucket $BUCKET com SUCESSO\n" >>
/tmp/syslog_sys
        else
                echo -e "FALHA na copia do arquivo $MARCA_DO_DIA-${Aplicacao[$i]}.
$SERVER-$DATA.tar.gz para o bucket $BUCKET \n" >> /tmp/syslog_sys
        fi

done

# Enviando arquivo de notificacao

ENVIO=$(cat /tmp/syslog_sys)

echo "$ENVIO" | mail -s "Backup Apps 235 AMAZON S3 de $DATA"
fcunha@xxx.com.br
```

Fonte: Própria.

A segurança é um fator importante dentro do contexto de *backups* para *buckets Amazon AWS S3*, portanto, as permissões de acesso devem ser configuradas adequadamente no IAM (*Identity and Access Management*). Esta ferramenta permite que os serviços e recursos da *Amazon AWS* sejam gerenciados com segurança.

A figura 9 exibe a configuração de um arquivo *JSON* no IAM para que um usuário tenha acesso total apenas em seu respectivo *bucket*.

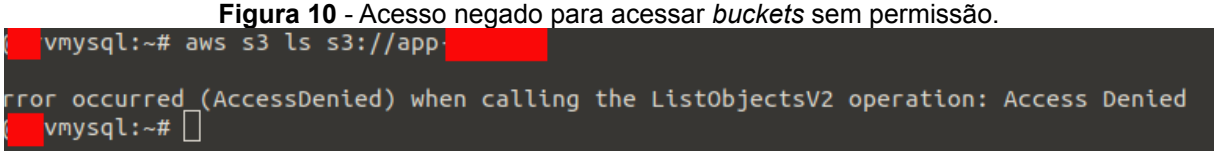
Figura 9 - Arquivo *JSON* configurado no IAM de um usuário para permitir acesso total apenas no *bucket* necessário.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "Stmt1548376342000",
         "Effect": "Allow",
         "Action": [
            "s3:*"
         ],
         "Resource": [
            "arn:aws:s3:::nfe-xxx",
            "arn:aws:s3:::nfe-xxx/*"
         ]
      }
   ]
}
```

Fonte: Própria.

Se o usuário em questão tentar fazer qualquer operação em outro *bucket* da empresa que não tenha permissão, a mensagem da figura 10 será exibida.

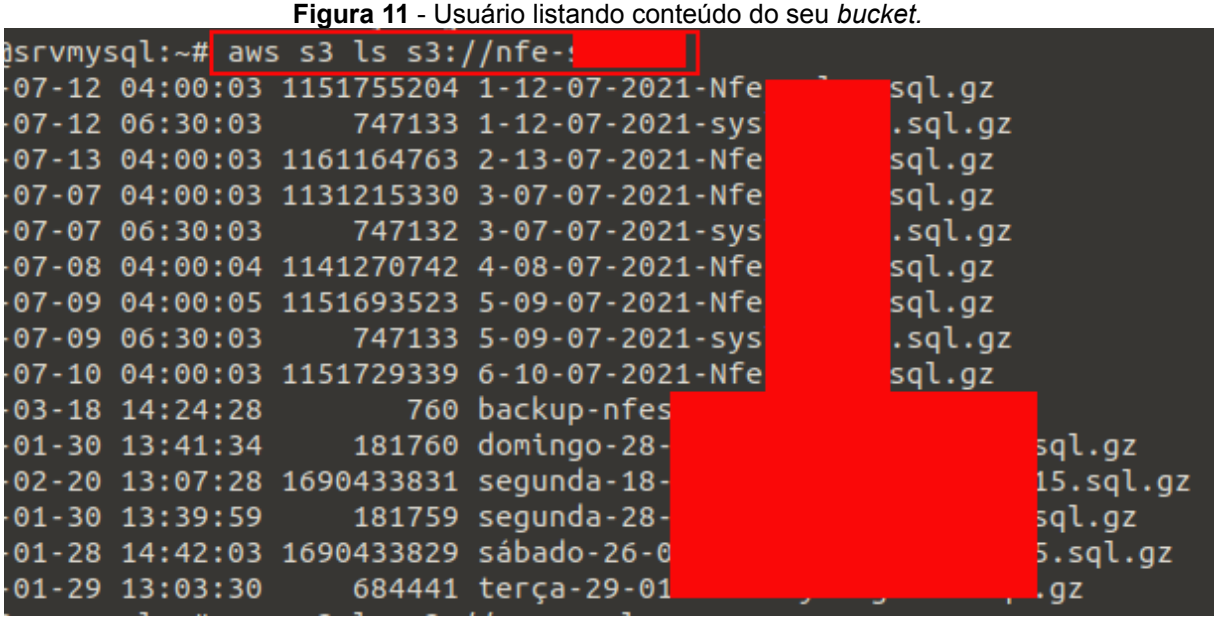
Figura 10 - Acesso negado para acessar *buckets* sem permissão.



Fonte: Própria.

Por fim, a figura 11, nos mostra o usuário listando o conteúdo do seu *bucket*.

Figura 11 - Usuário listando conteúdo do seu *bucket.*



Fonte: Própria.

3 PROJETO 3

3.1 *Daemon* em *Shell script* para gerenciamento de *Firewall*.

A criação de serviços personalizados em *Shell Script* para atender determinadas demandas faz parte do escopo das rotinas administrativas diárias de um profissional de segurança da informação Linux.

A estrutura do *script shell* da figura 12 é carregada na inicialização da máquina e pode ser usado para gerenciar o *firewall* (parar e inicializar).

Figura 12 - *Script Shell* para gerenciamento de *firewall*.

```
#!/bin/bash

### BEGIN INIT INFO
# Provides:          firewall
# Required-Start:    $remote_fs $syslog $named $network $time
# Required-Stop:     $remote_fs $syslog $named $network
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Host firewall
# Description:       Firewall is a Firewall de Host
### END INIT INFO



PF="/sbin/iptables"
IFACE_LAN="eth0"
```

```
# Função inicio – Carrega as regras do Firewall
inicio () {

# Definindo a política padrao
$PF -P INPUT DROP

## Regras

## Liberando tudo para loopback
$PF -A INPUT -s 127.0.0.1 -i lo -d 127.0.0.1 -j ACCEPT

# Liberando A entrada de pacotes ICMP 8 (limite de 5 por segundo, para teste de conectividade)

$PF -A INPUT -p 1 --icmp-type 8 -i $IFACE_LAN -m limit --limit 5/sec -j ACCEPT

# Acesso SSH

$PF -A INPUT -p 6 --syn -i $IFACE_LAN --dport 22 -m state --state NEW -j ACCEPT

# Acesso Apache HTTP e HTTPS

$PF -A INPUT -p 6 --syn -i $IFACE_LAN --dport 80 -m state --state NEW -j ACCEPT
$PF -A INPUT -p 6 --syn -i $IFACE_LAN --dport 443 -m state --state NEW -j ACCEPT

# Acesso ao servidor NTP
$PF -A INPUT -p 17 -i $IFACE_LAN --dport 123 -m state --state NEW -j ACCEPT

# Garantindo as respostas as solicitações do próprio firewall
```

```
$PF -A INPUT -m state --state RELATED,ESTABLISHED -j ACCEPT
}

parar () {

echo "Finalizando Firewall"

$PF -Z
$PF -Z -t nat
$PF -Z -t mangle
$PF -F
$PF -F -t nat
$PF -F -t mangle
$PF -P INPUT ACCEPT
$PF -P FORWARD ACCEPT
$PF -P OUTPUT ACCEPT

}

case $1 in

start)

inicio

;;

stop)

parar

;;
```

```
*) echo "Erro! Use $0 [start|stop]"
exit 1
esac

exit 0
```

Fonte: Própria.

4 PROJETO 4

4.1 *Shell Script* criado com interface *Dialog* para usuário do *Help Desk* gerenciar impressora em servidor Linux NFE.

Existem situações onde um determinado colaborador precisa de acesso root (*admin* do Linux) para executar uma ou mais tarefas. Para estes casos podemos criar programas em *Shell* com interfaces intuitivas e mais amigáveis para que estes colaboradores não precisem digitar comandos no terminal.

O código exibido na figura 13 permite que o usuário modifique a impressora do servidor NFE usando um programa em *Shell* com interface dialog.

Figura 13 - *Script Shell* para modificar impressora de servidor NFE.

```
#!/bin/bash

# 17/03/2021 - Modifica a impressora da maquina
# By Flávio Cunha

PATH="/bin:/sbin:/usr/bin:/usr/sbin:/usr/local/bin:/usr/local/sbin"
File_Print="/etc/cups/printers.conf"
Teste_Print="/etc/cups/teste.txt"
```

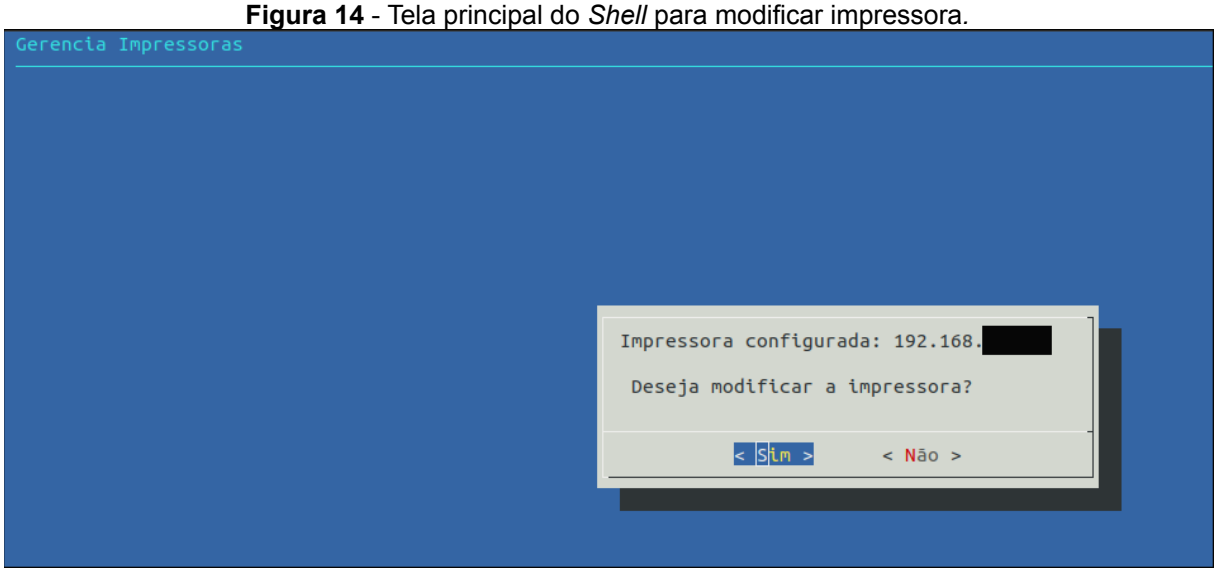
```
# Variável para validação de endereço IP.
Regex_IP="\b([0-9]{1,3}\.){3}[0-9]{1,3}\b"

# Funcao para mostrar impressora configurada

function mostra_ip() {
  # echo "Numero de argumentos: $#"
  Padrao=$(sudo sed -n '/^Location/p' $File_Print | awk '{print $2}')
                sudo dialog --backtitle 'Gerencia Impressoras' --yesno "Impressora configurada: $Padrao.\n\n Deseja modificar a impressora? \n\n" 0 0

  # echo $*
}

mostra_ip

if [ $? = 0 ]; then
                  Novo_IP=$(sudo dialog --backtitle 'Gerencia Impressoras' --stdout --inputbox "Entre com o IP da Impressora!\n\n Ex: 192.168.16.24" 0 0)

              [ $? -ne 0 ] && exit 1

               if [[ $Novo_IP =~ $Regex_IP ]] ; then
                    sudo sed -i "s/$Padrao/$Novo_IP/g" $File_Print && dialog --backtitle 'Gerencia Impressoras' --msgbox "Impressora $Novo_IP configurada com Sucesso!" 0 0
                    sudo /etc/init.d/cups stop
                    sudo /etc/init.d/cups start
                    # enviando teste de impressao
                    #sleep 5

                    TOTAL="10"
```

```
                        for ((i=1 ; i <= 10 ; i++)) ; do
                                STATUS="$i"
                                PORCENT=$(($STATUS*100/TOTAL))
                                 (echo $PORCENT; sleep 1) | dialog --gauge 'Atualizando
Impressora...' 8 40 0
                        done

                dialog --msgbox 'Atualizacao concluida!!' 6 40

                        for ((i=0 ; i <= 2 ; i++)) ; do
                                sudo lp -o fit-to-page -o media=a4 $Teste_Print
                        done
                else
                  sudo dialog --backtitle 'Gerencia Impressoras' --msgbox 'Formato
de IP NAO VALIDO!' 0 0
            fi
        else
                sudo dialog --backtitle 'Gerencia Impressoras' --msgbox 'Muito Bem! \n\n
Bye! Bye!' 0 0
fi
```

Fonte: Própria.

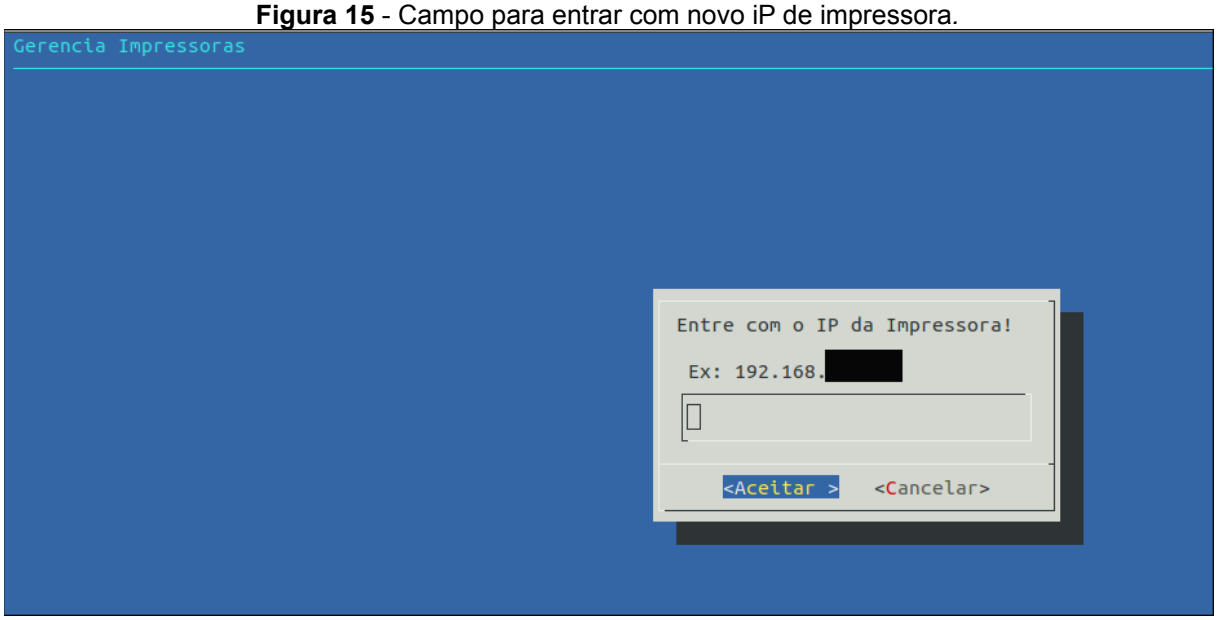
As figuras 14,15 e 16 mostrama algumas telas do programa.

Figura 14 - Tela principal do *Shell* para modificar impressora.



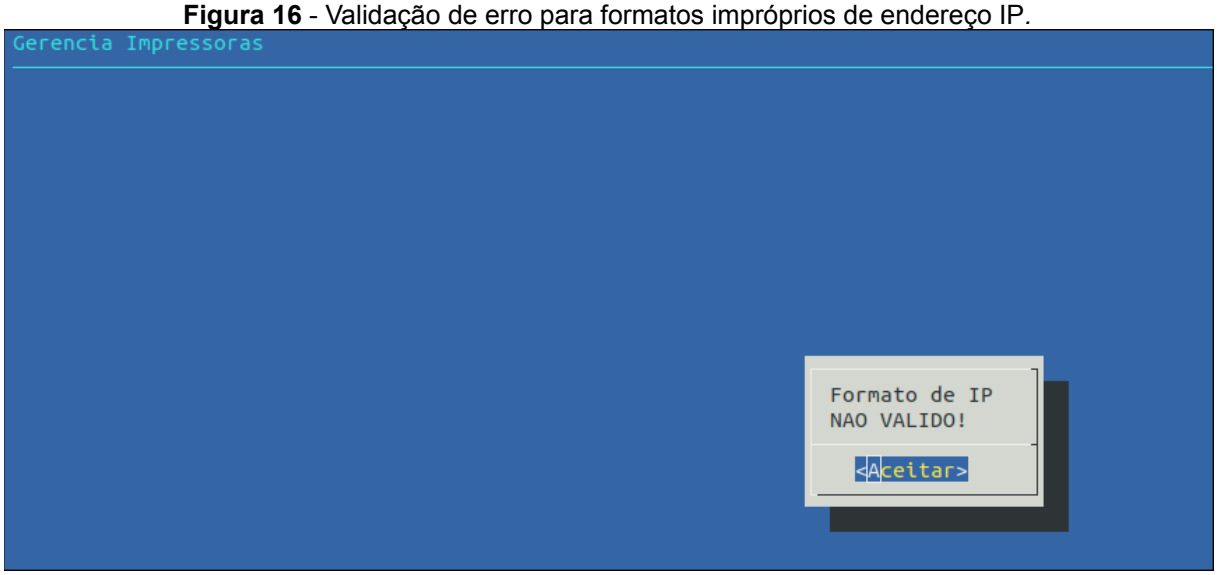
Fonte: Própria.

Figura 15 - Campo para entrar com novo iP de impressora.



Fonte: Própria.

Figura 16 - Validação de erro para formatos impróprios de endereço IP.

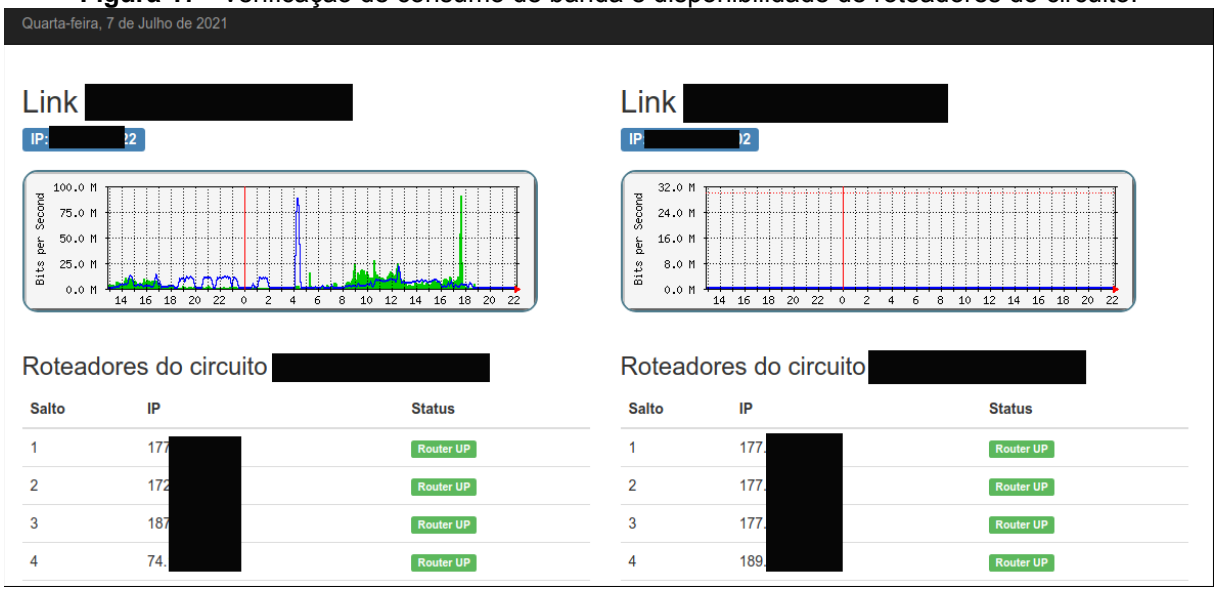


Fonte: Própria.

5 APÊNDICE A – Outros projetos interessantes

Interface para monitoração de consumo de banda de *links* de *internet* com verificação da disponibilidade dos roteadores do circuito via aplicação PHP.

Figura 17 - Verificação de consumo de banda e disponibilidade de roteadores do circuito.



Fonte: Própria.

A disponibilidade dos roteadores é verificada via Aplicação PHP através da função *exec* chamando o comando *ping*.

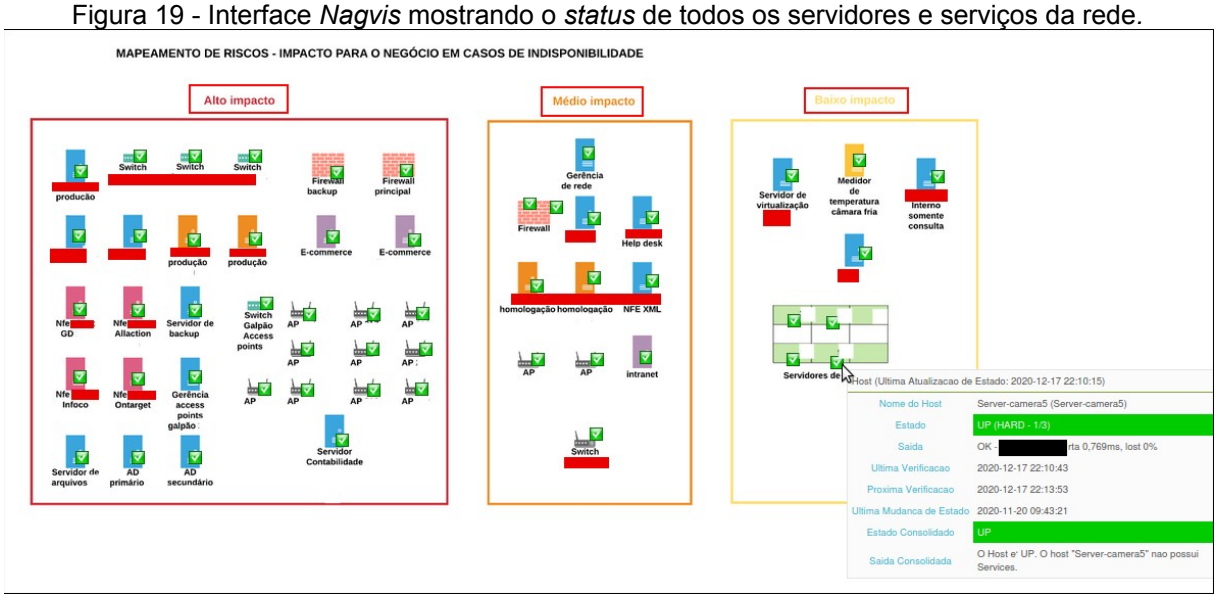
Figura 18 - Função exec do PHP fazendo uma chamada de sistema para execução do comando ping.

```
<td><?php if (exec('ping -c 1 -w 1 $ip_router')) { echo '<span class="label label-success">Router UP</span>'; } else { echo '<span class="label label-danger">Router Down'; } ?></td>
```

Fonte: Própria.

Interface para monitoração de servidores e serviços da rede da empresa com Icinga, *plugins* de monitoração do *Nagios*, interface *Nagvis* e banco de dados *Sqlite*.

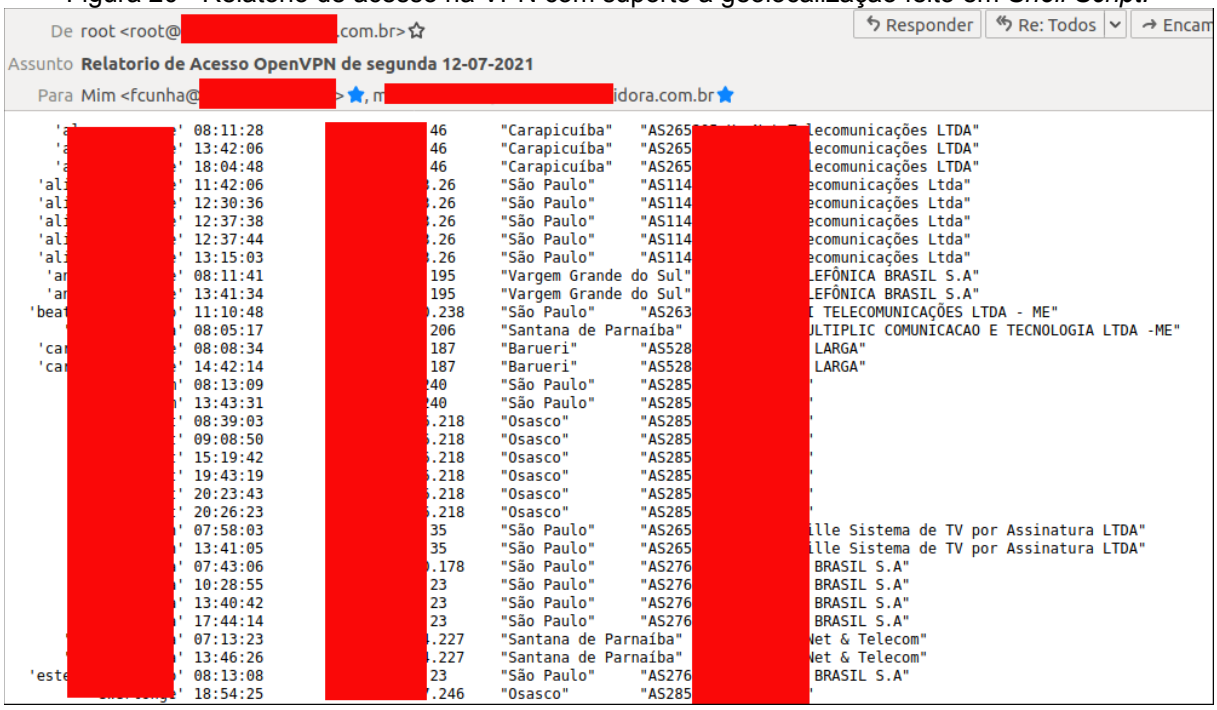
Figura 19 - Interface *Nagvis* mostrando o *status* de todos os servidores e serviços da rede.



Fonte: Própria.

Relatório de acesso de VPN implementado em *Shell Script* com suporte a geolocalização, que mostra o usuário, horário de conexão e desconexão, endereço IP, Cidade, provedor de acesso e que envia e-mail para os gestores responsáveis.

Figura 20 - Relatório de acesso na VPN com suporte a geolocalização feito em *Shell Script.*



Fonte: Própria.