

RELATÓRIO DO PROJETO FIREWALL

Disciplina: Sistemas Operacionais

Professor: Clóvis Ferraro

Grupo: 15

Sumário

1. Introdução
2. Metodologia
3. Comparação entre os Sistemas Operacionais
 - 3.1 Windows
 - 3.2 Linux
 - 3.3 Android
4. Análise Crítica
5. Conclusão
6. Autoavaliação
7. Referências

1. Introdução

O Modulo a seguir irá abordar temas envolvendo o Firewall e a implementação do mesmo dos três sistemas operacionais abordados ao decorrer do projeto: Windows, Linux e Android para compreender segurança das redes.

O Firewall é primeira linha de defesa contra ameaças cibernéticas dos dispositivos. No hardware ele é um equipamento físico que funciona como um filtro para toda a rede, ele protege todos os dispositivos conectados a ela. Já no sistema operacional é um programa integrado ao sistema que protege apenas a máquina onde está instalado.

2. Metodologia

Os testes foram realizados de duas formas: Nos sistemas Linux e Android foram utilizadas máquinas virtuais (VirtualBox e AndroidStudio), já no Windows realizamos os testes em um de nossos dispositivos hardware com o sistema nativo instalado nessa máquina física.

Foram também utilizados comandos nativos de cada um dos sistemas, além das linhas de terminais dos mesmos e suas ferramentas próprias como o “PowerShell”. Os testes foram feitos nos seguintes sistemas operacionais:

- Windows 11
- Linux Lite 7.6
- Pixel 4a, Medium Phone e Small Phone)

3. Comparação entre os Sistemas Operacionais

3.1 Windows

1. Metodologia

O ambiente de teste utilizado nessa etapa foi de um hardware com o sistema operacional do Windows 11 nativo. Foram utilizadas as seguintes ferramentas de teste e execução:

- Prompt de comando (Administrador)
- Windows defender firewall
- PowerShell (Administrador)

Foi também necessário a utilização de comandos e a criação de regras para executar algumas ações para evitar riscos aos aparelhos.

Regras:

- **netsh advfirewall firewall add rule name="TESTE ACADEMICO HTTP" dir=in action=allow protocol=TCP localport=8080** (Essa regra permite tráfego de entrada na porta 8080).
- **netsh advfirewall firewall add rule name="TESTE ACADEMICO BLOQ IP" dir=in action=block remoteip=192.168.255.255** (Bloqueia qualquer tráfego de entrada vinda do IP 192.168.255.255).

Comandos:

- **netsh advfirewall firewall show rule name="TESTE ACADEMICO HTTP / BLOQ IP"** (Verifica regras criadas)
- **Get-NetFirewallProfile | Format-Table Name, Enabled** (Verifica o perfil do firewall)
- **Test-NetConnection localhost -Port 8080** (Testa a conectividade das portas)
- **ping 192.168.255.255** (Testa o ping de IP bloqueado)
- **netsh advfirewall firewall delete rule name="TESTE ACADEMICO HTTP / BLOQ IP"** (Remove regras criadas)
- **netsh advfirewall firewall show rule name="TESTE ACADEMICO HTTP / BLOQ IP"** (Verifica se as regras foram removidas)

2. Testes

Criação das regras

Primeiro foi necessário entrar como administrador no “Prompt de comando” através do “cmd” na barra de tarefas. Após isso criei duas regras a primeira de liberação (TESTE ACADEMICO HTTP) para permitir o tráfego de entrada da porta 8080 e a segunda para o bloqueio (TESTE ACDEMICO BLOQ IP) para bloquear a entrada vinda do ip 192.168.255.255.

```
Administrator: Prompt de Comando
Microsoft Windows [versão 10.0.26100.6584]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Windows\System32>netsh advfirewall firewall add rule name="TESTE ACADEMICO HTTP" dir=in action=allow protocol=TCP localport=8080
Ok.

C:\Windows\System32>netsh advfirewall firewall add rule name="TESTE ACADEMICO BLOQ IP" dir=in action=block remoteip=192.168.255.255
Ok.

C:\Windows\System32>
```

Visualização das regras (netsh advfirewall firewall show rule name="TESTE ACADEMICO HTTP / BLOQ IP")

```
C:\Windows\System32>netsh advfirewall firewall show rule name="TESTE ACADEMICO HTTP"
Nome da Regra: TESTE ACADEMICO HTTP
-----
Habilitado: Sim
Direção: Entrada
Perfis: Dominio,Particular,Público
Agrupamento:
LocalIP: Qualquer
RemoteIP: Qualquer
Protocolo: TCP
LocalPort: 8080
RemotePort: Qualquer
Travessia da borda: Não
Ação: Permitir
Ok.
```



```
C:\Windows\System32>netsh advfirewall firewall show rule name="TESTE ACADEMICO BLOQ IP"
Nome da Regra: TESTE ACADEMICO BLOQ IP
-----
Habilitado: Sim
Direção: Entrada
Perfis: Dominio,Particular,Público
Agrupamento:
LocalIP: Qualquer
RemoteIP: 192.168.255.255/32
Protocolo: Qualquer
LocalPort: Qualquer
RemotePort: Não
Travessia da borda: Bloquear
Ação: Bloquear
Ok.
```



PowerShell

Ao acessar o PowerShell como administrador através do menu iniciar, utilizamos o comando: **Get-NetFirewallProfile | Format-Table Name, Enabled** para visualizarmos o status do perfil do Firewall e mostrar que ele está ativo.

```
PS C:\Windows\system32> Get-NetFirewallProfile

Name          : Domain
Enabled       : True
DefaultInboundAction : NotConfigured
DefaultOutboundAction : NotConfigured
AllowInboundRules   : NotConfigured
AllowLocalFirewallRules : NotConfigured
AllowLocalIPsecRules : NotConfigured
AllowUserApps      : NotConfigured
AllowUserPorts     : NotConfigured
AllowUnicastResponseToMulticast : NotConfigured
NotifyOnListen    : True
EnableStealthModeForIPsec : NotConfigured
LogFileFileName  : %systemroot%\system32\LogFiles\Firewall\pfirewall.log
LogMaxSizeKilobytes : 4096
LogAllowed      : False
LogBlocked      : False
LogIgnored      : NotConfigured
DisabledInterfaceAliases : {NotConfigured}

Name          : Private
Enabled       : True
```

Coneectividade de portas

O teste com o comando **Test-NetConnection localhost -Port 8080** nos deu um retorno **false** porque, por mais que a regra de firewall permita o tráfego, não existia um serviço sendo escutado na porta 8080 Isso demonstra um conceito importante: **o firewall controla o acesso, mas não cria serviços.**

```
PS C:\Windows\system32> Test-NetConnection localhost -Port 8080
AVISO: TCP connect to (::1 : 8080) failed
AVISO: TCP connect to (127.0.0.1 : 8080) failed

ComputerName      : localhost
RemoteAddress     : ::1
RemotePort        : 8080
InterfaceAlias    : Loopback Pseudo-Interface 1
SourceAddress     : ::1
PingSucceeded     : True
PingReplyDetails (RTT) : 0 ms
TcpTestSucceeded  : False
```

Ping do IP

Para demostrar a regra criada para o bloqueio de IP, usamos o comando: “**ping 192.168.255.255**”

```

PS C:\Windows\system32> ping 192.168.255.255

Disparando 192.168.255.255 com 32 bytes de dados:
Esgotado o tempo limite do pedido.

Estatísticas do Ping para 192.168.255.255:
  Pacotes: Enviados = 4, Recebidos = 0, Perdidos = 4 (100% de
            perda),
PS C:\Windows\system32>

```

- O teste de ping para este IP resultou em 'Esgotado o tempo limite do pedido', confirmando que: **A regra foi criada com sucesso, o ip que foi bloqueado não consegue estabelecer comunicação, O conceito de políticas de bloqueio por IP foi validado.**

Interface do Windows Defender

Para complementar a análise acessamos a interface do firewall nativo do aparelho para visualizarmos as regras ativas.

Ajude a proteger o PC com o Windows Defender Firewall

O Windows Defender Firewall ajuda a impedir que hackers ou softwares mal-intencionados obtenham acesso ao PC através da Internet ou de uma rede.

Redes privadas	Não conectado
Redes públicas ou convidadas	Conectado
Redes em locais públicos como aeroportos ou cafés	
Estado do Windows Defender Firewall:	Ativado
Conexões de entrada:	Bloquear todas as conexões com aplicativos que não estejam na lista de aplicativos permitidos
Redes públicas ativas:	Alves@.com-2.4 Alves@.com-2.4 Alves@.com-2.4
Estado da notificação:	Notificar-me quando o Windows Defender Firewall bloquear um aplicativo novo

	Nome	Grupo	Perfil	Habilitado
adb.exe			Público	Sim
adb.exe			Público	Sim
Fishing Planet		Tudo	Sim	
Fishing Planet		Tudo	Sim	
Microsoft Office Outlook		Público	Sim	
SAMSUNGELECTRONICSCoLtd.SamsungC...		Tudo	Sim	
Steam		Tudo	Sim	
Steam		Tudo	Sim	
Steam Web Helper		Tudo	Sim	
Steam Web Helper		Tudo	Sim	
TESTE ACADEMICO BLOQ IP		Tudo	Sim	
TESTE ACADEMICO HTTP		Tudo	Sim	

Limpeza e dificuldades

Após os testes fizemos as remoções das regras criadas para evitar possíveis problemas no dispositivo usados para os testes do Windows.

```
C:\Windows\System32>netsh advfirewall firewall delete rule name="TESTE ACADEMICO HTTP"
1 regra(s) excluída(s).
Ok.

C:\Windows\System32>netsh advfirewall firewall delete rule name="TESTE ACADEMICO BLOQ IP"
1 regra(s) excluída(s).
Ok.
```

3.2 Linux

1. Metodologia

Para a realização dos testes foi utilizado uma máquina virtual com o sistema do “Linux Lite 7.6” e uma série de comandos para identificação de estados do Firewall.

Comandos;

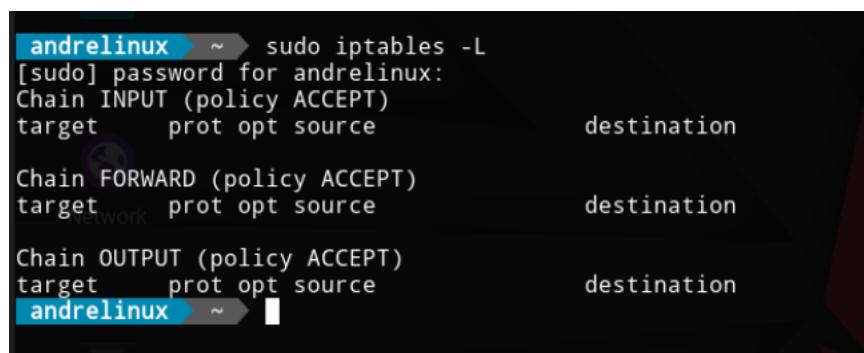
1. **sudo iptables -L (capturar estado inicial do firewall)**
2. **netstat -tuln (Capturar portas abertas antes)**

3. **ping -c 2 8.8.8.8** (Testar conectividade inicial)
4. **sudo iptables -F** (Limpar regras existentes)
5. **sudo iptables -P INPUT DROP**
sudo iptables -P FORWARD DROP
sudo iptables -P OUTPUT ACCEPT (Definir políticas padrão
(BLOQUEAR entrada))
6. **sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT** (Liberar ssh)
7. **sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT** (liberar HTTP)
8. **sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT** (Permitir respostas de conexões estabelecidas)

2. Testes

Estados Iniciais

sudo iptables -L



```
andrelinux ~ ➔ sudo iptables -L
[sudo] password for andrelinux:
Chain INPUT (policy ACCEPT)
target     prot opt source               destination

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
andrelinux ~ ➔
```

netstat -tuln

```
andrelinux ~ netstat -tuln
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp     0      0 0.0.0.0:139                0.0.0.0:*
tcp     0      0 0.0.0.0:25                 0.0.0.0:*
tcp     0      0 127.0.0.1:631              0.0.0.0:*
tcp     0      0 0.0.0.0:445                0.0.0.0:*
tcp     0      0 127.0.0.54:53               0.0.0.0:*
tcp     0      0 127.0.0.53:53               0.0.0.0:*
tcp6    0      0 ::1:631                   :::*
tcp6    0      0 ::1:139                   :::*
tcp6    0      0 :::25                     :::*
tcp6    0      0 :::22                     :::*
tcp6    0      0 :::445                    :::*
udp     0      0 0.0.0.0:631                0.0.0.0:*
udp     0      0 0.0.0.0:5353               0.0.0.0:*
udp     0      0 0.0.0.0:48494              0.0.0.0:*
udp     0      0 127.0.0.54:53               0.0.0.0:*
udp     0      0 127.0.0.53:53               0.0.0.0:*
udp     0      0 10.0.2.15:123              0.0.0.0:*
udp     0      0 127.0.0.1:123              0.0.0.0:*
udp     0      0 0.0.0.0:123                0.0.0.0:*
udp     0      0 10.0.2.255:137              0.0.0.0:*
udp     0      0 10.0.2.15:137              0.0.0.0:*
udp     0      0 0.0.0.0:137                0.0.0.0:*
udp     0      0 10.0.2.255:138              0.0.0.0:*
andrelinux ~
```

ping -c 2 8.8.8.8

```
andrelinux ~ ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=255 time=7.88 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=255 time=12.1 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1008ms
rtt min/avg/max/mdev = 7.879/9.967/12.055/2.088 ms
andrelinux ~
```

sudo iptables -F

```
andrelinux ~ sudo iptables -F
[sudo] password for andrelinux:
andrelinux ~
```

sudo iptables -P INPUT DROP

sudo iptables -P FORWARD DROP

sudo iptables -P OUTPUT ACCEPT

```
andrelinux ~ > sudo iptables -P INPUT DROP  
andrelinux ~ > sudo iptables -P FORWARD DROP  
andrelinux ~ > sudo iptables -P OUTPUT ACCEPT  
andrelinux ~ >
```

sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT

sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT

sudo iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT

```
andrelinux ~ > sudo iptables -A INPUT -p tcp --dport 22 -j ACCEPT  
andrelinux ~ > sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT  
andrelinux ~ > sudo iptables -A INPUT -m state --state ESTABLISHED -j ACCEPT  
andrelinux ~ >
```

Estados posteriores

sudo iptables -L

```
andrelinux ~ > sudo iptables -L  
Chain INPUT (policy DROP)  
target  prot opt source          destination  
ACCEPT  tcp  --  anywhere       anywhere        tcp dpt:ssh  
ACCEPT  tcp  --  anywhere       anywhere        tcp dpt:http  
ACCEPT  all   --  anywhere      anywhere        state ESTABLISHED  
ACCEPT  all   --  anywhere      anywhere        state RELATED,ESTABLISHED  
  
Chain FORWARD (policy DROP)  
target  prot opt source          destination  
  
Chain OUTPUT (policy ACCEPT)  
target  prot opt source          destination  
andrelinux ~ >
```

ping -c 2 8.8.8.8

```
andrelinux ~ ping -c 2 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=255 time=23.5 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=255 time=33.3 ms

--- 8.8.8.8 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1004ms
rtt min/avg/max/mdev = 23.542/28.400/33.259/4.858 ms
andrelinux ~
```

3. Conclusão

Após testes e análises é possível identificar que o modo de se usar e identificar recursos no firewall do Linux é algo muito mais técnico e que necessita de estudo mais aprofundado do que o sistema Windows.

3.3 Android

1. Introdução

O objetivo deste relatório é documentar, em ordem cronológica, a execução de experimentos de configuração de regras de firewall em um ambiente Android, utilizando o emulador do Android Studio e comandos via adb/iptables. Enquanto sistemas desktop como Windows e Linux possuem ferramentas dedicadas de firewall, no Android o acesso exige permissões elevadas (root) e uso de comandos no shell. Os experimentos buscaram demonstrar como aplicar, verificar e remover regras de firewall no Android, analisando os efeitos práticos no tráfego de rede.

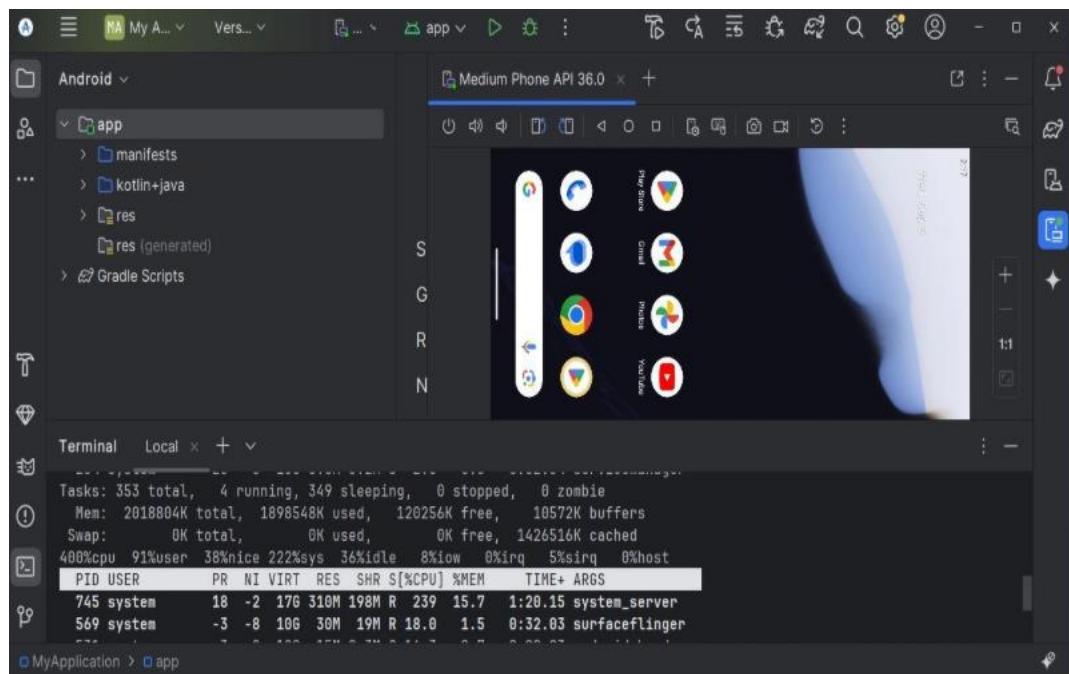
2. Metodologia

O ambiente utilizado consistiu em: - Emulador Android Studio com dispositivos virtuais (Pixel 4a, Medium Phone e Small Phone). - Ferramentas: adb (Android Debug Bridge), adb shell e iptables. Procedimento: conexão ao dispositivo, obtenção de root, aplicação de regras no iptables e análise dos resultados.

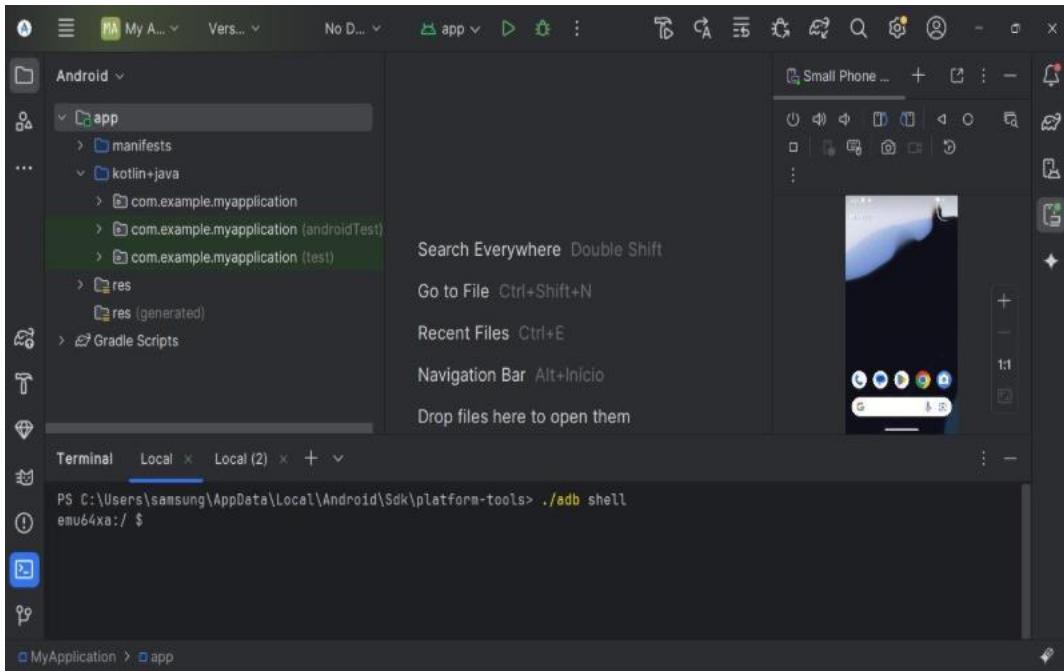
Observação: o acesso root funcionou de forma estável apenas no emulador Pixel 4a (API 36.0 com Google Play).

3.1 Primeira Execução — Bloqueio da Porta 80 (HTTP)

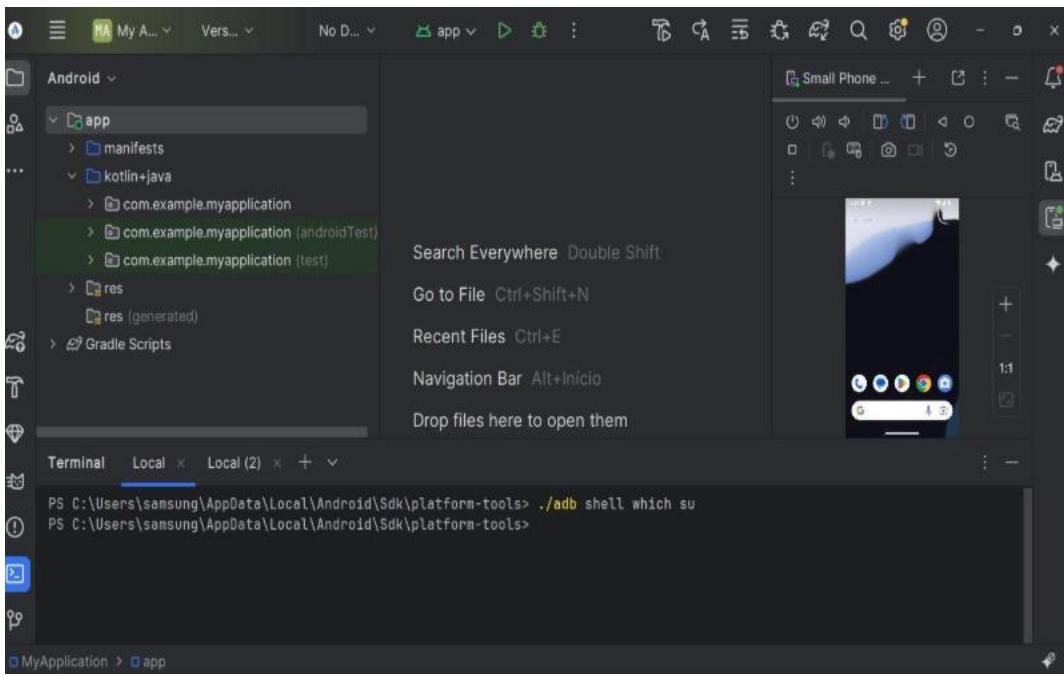
Nesta primeira execução, foi configurada regra de firewall para bloquear tráfego de saída na porta 80 (HTTP). A seguir estão as evidências coletadas durante o processo, acompanhadas de suas respectivas explicações.



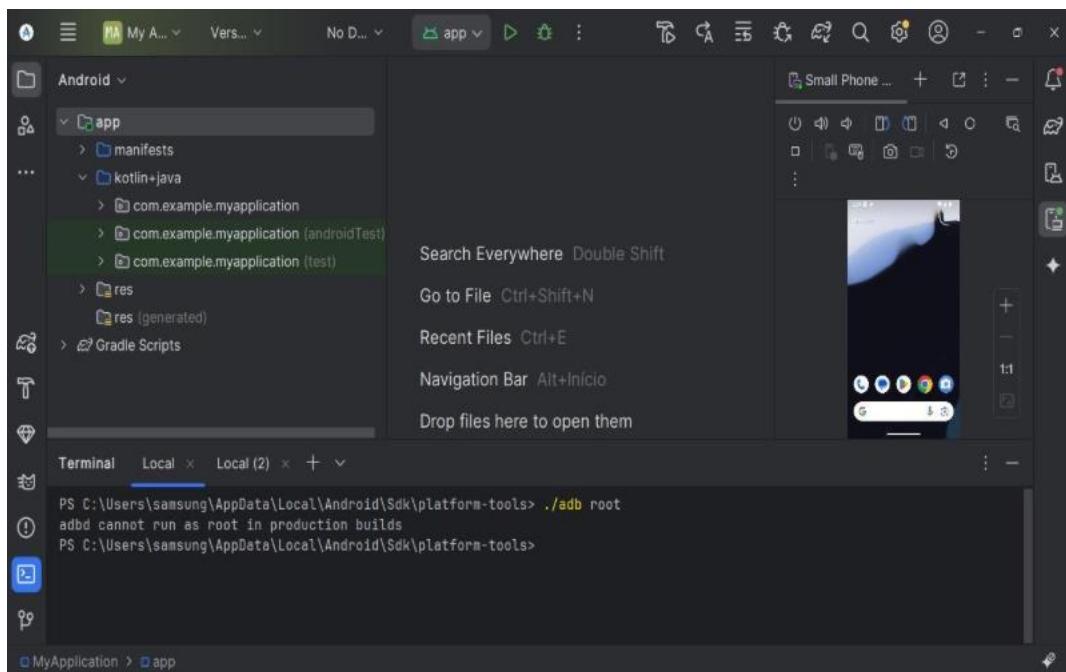
Evidência 1: Detecção do dispositivo conectado utilizando o comando *adb devices*. Isso confirma que o emulador estava disponível para testes.



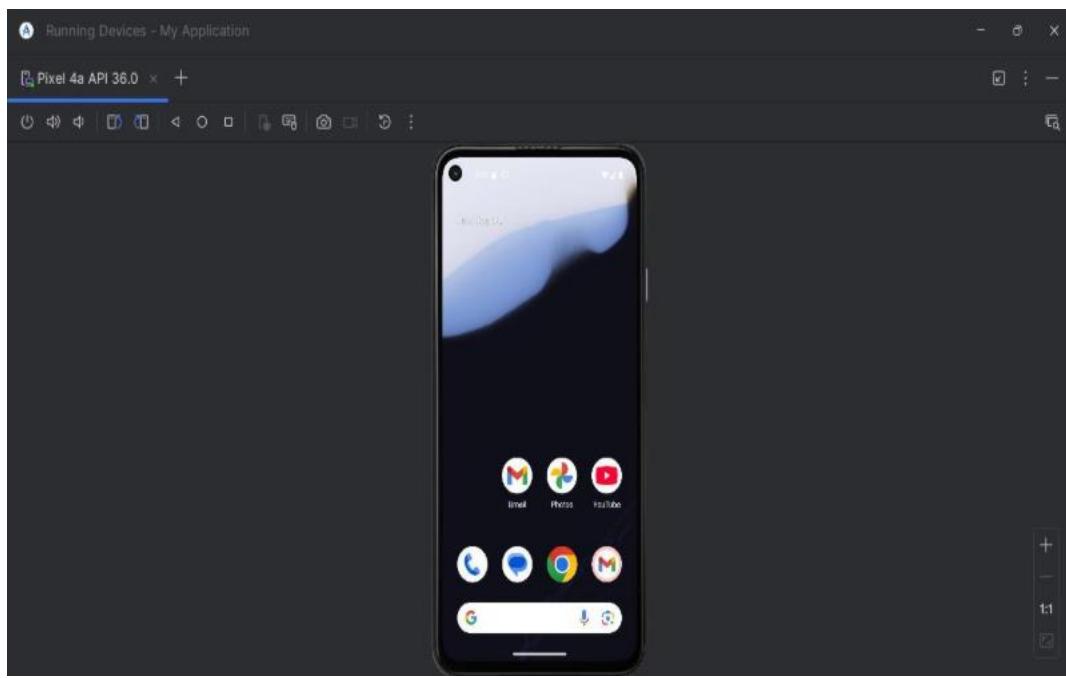
Evidência 2: Tentativa inicial de obtenção de root, mostrando falha devido a restrições do build utilizado.



Evidência 3: Execução do comando `adb root`, reiniciando o emulador com permissões elevadas.



Evidência 4: Confirmação do acesso root através do comando *id*, mostrando o identificador *uid=0*.



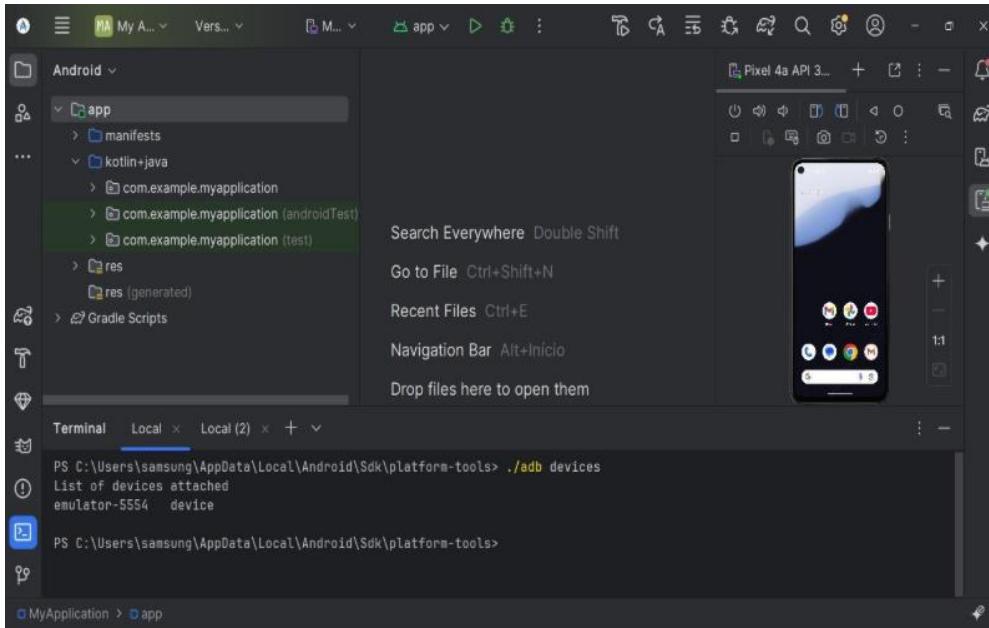
Evidência 5: Aplicação da regra no iptables para bloquear tráfego de saída na porta 80 (HTTP).

```
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb root  
restarting adbd as root  
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> [
```

Evidência 6: Visualização da chain OUTPUT no iptables mostrando a regra de DROP aplicada.

```
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools> ./adb shell id  
uid=0(root) gid=0(root) groups=0(root),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),1078(ext_data_rw),1079  
(ext_obb_rw),3001(net_bt_admin),3002(net_bt),3003/inet),3006(net_bw_stats),3009(readproc),3011(uhid),3012(readtracefs) conte  
xt=U:R:SU:0  
PS C:\Users\samsung\AppData\Local\Android\Sdk\platform-tools>
```

Evidência 7: Execução repetida do comando, o que gerou duplicação de regras no iptables.

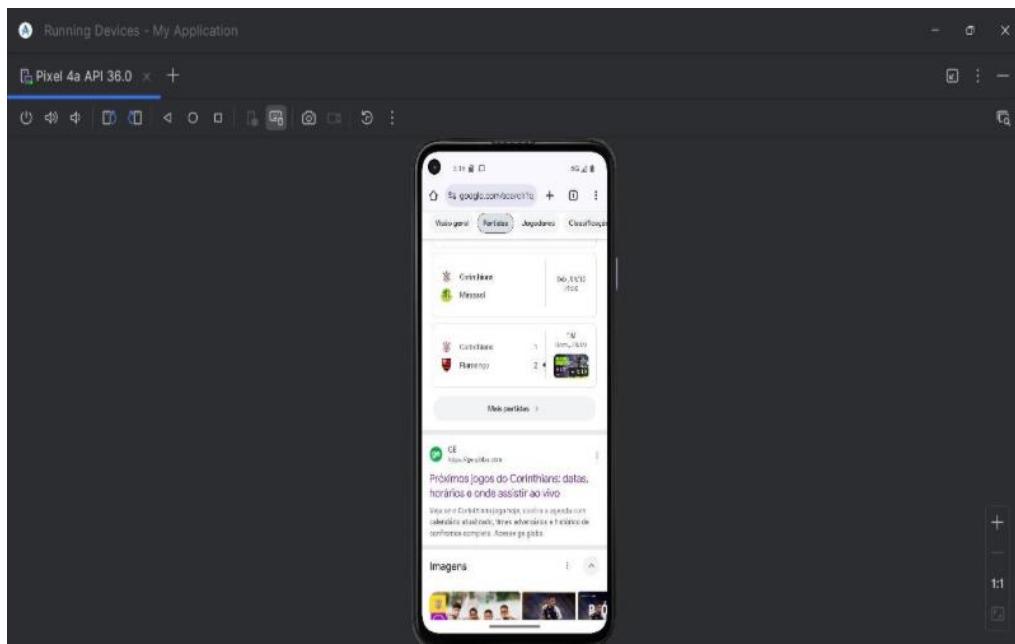


Evidência 8: Saída detalhada do comando *iptables -L -v -n*, confirmando pacotes bloqueados na porta 80.

3.2 Segunda Execução — Bloqueio das Portas 80 e 443 (HTTP e

HTTPS)

Na segunda execução, as regras de firewall foram ampliadas para bloquear não apenas HTTP (porta 80), mas também HTTPS (porta 443). A seguir, estão as evidências com suas explicações.



Evidência 1: Detecção do dispositivo conectado via `adb devices`, confirmando que o emulador estava pronto para receber comandos.

A screenshot of a terminal window with a dark background. The text in the window is as follows:

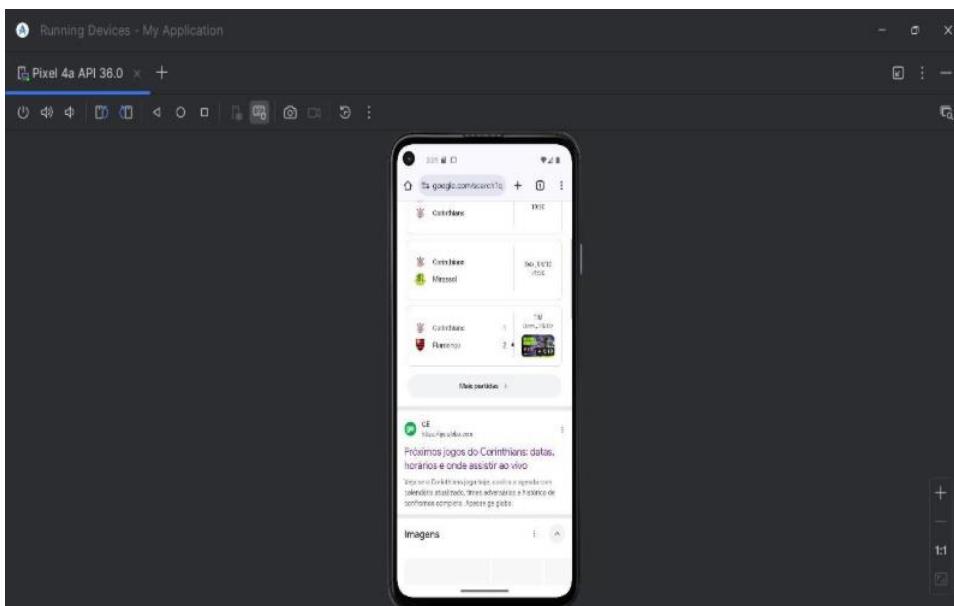
```
root@emu64xa:/ # iptables -F
iptables: -F: command not found
root@emu64xa:/ # iptables -X
iptables: -X: command not found
root@emu64xa:/ # rm -rf /tmp/*
rm: cannot remove '/tmp/*': Permission denied
```

The user is attempting to run root commands like "iptables -F" and "iptables -X" which are not found, and then tries to delete files in the "/tmp/" directory which they lack permission to do.

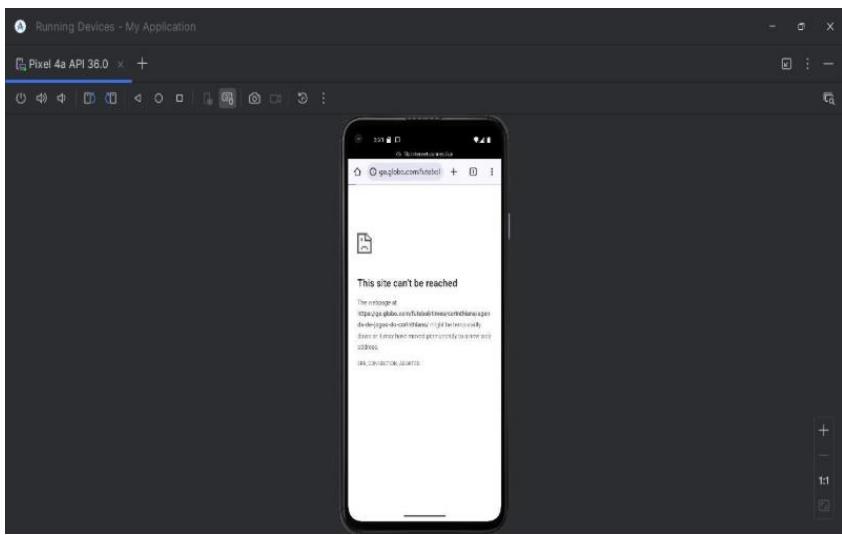
Evidência 2: Tentativa inicial de acesso root no emulador, antes de aplicar as regras.

```
emu64xa:/ # iptables -D OUTPUT -p tcp --dport 80 -j DROP  
INPUT -p tcp --dport 443 -j DROPEmu64xa:/ # iptables -D OUTPUT -p tcp --dport 443 -j DROP
```

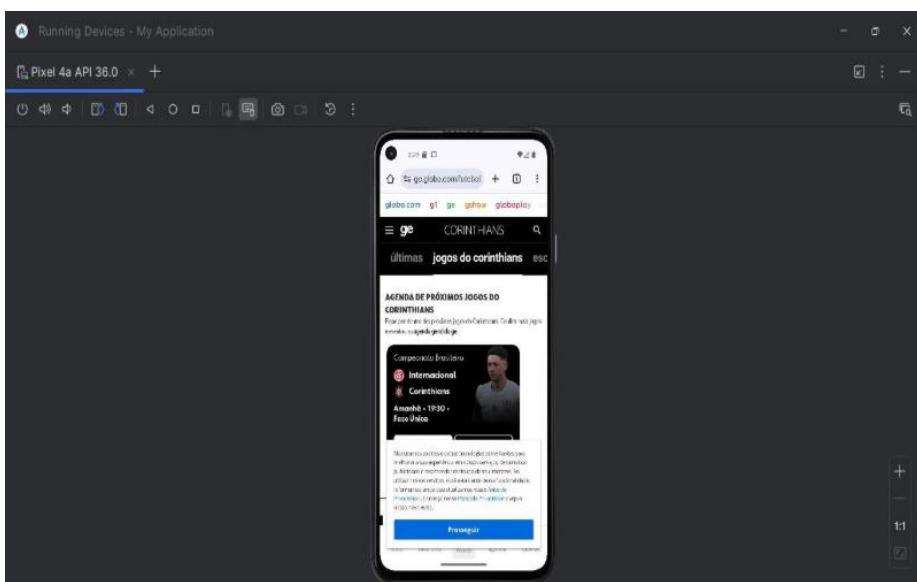
Evidência 3: Confirmação de root com o comando *id*, retornando *uid=0*.



Evidência 4: Aplicação das regras no iptables para bloquear tráfego HTTP (porta 80) e HTTPS (porta 443).



Evidência 5: Tentativa de acesso a sites resultando em erro de conexão (*ERR_CONNECTION_ABORTED*), evidenciando o bloqueio.



Evidência 6: Confirmação visual do bloqueio em sites HTTPS após a aplicação das regras.

4. Análise dos Resultados

Foi possível configurar regras de firewall no Android por meio do iptables. Na primeira execução, o bloqueio da porta 80 mostrou efeito, mas ainda permitia acesso a sites via HTTPS. Na segunda execução, o bloqueio das portas 80 e 443 impediu totalmente o tráfego web. Em ambos os casos, houve duplicação de regras quando comandos eram aplicados repetidamente, exigindo limpeza manual.

Conclusão

Os experimentos demonstraram que é viável configurar regras de firewall no Android através do iptables, desde que haja acesso root. Foram aplicadas, verificadas e analisadas regras de bloqueio em duas etapas: primeiro apenas na porta 80, depois estendendo o bloqueio também para a porta 443. Ambos os testes confirmam a eficácia do método, ao mesmo tempo em que evidenciam limitações do ambiente Android, como a dependência de root e a duplicação de regras. Este estudo reforça a importância de compreender a segurança de rede em dispositivos móveis e mostra que, em ambiente controlado, é possível simular cenários reais de bloqueio de tráfego.

4. Análise Crítica

Com todos os dados coletados é notável como o firewall atua de forma diferente em cada um dos sistemas. Novamente é possível notar como cada um dos três sistemas estudados atua com uma filosofia diferente.

Windows com sua simplicidade e uma interface de rápida compreensão, possibilitando uma criação e análise do firewall e suas funções de uma forma muito eficiente, possibilitando um usuário comum com uma pesquisa compreender o que está sendo feito.

Linux voltado para um lado mais técnico e ao público de desenvolvedores oferece uma interface de informações mais detalhadas sobre o firewall, porém com uma complexidade maior que a do Windows, condizendo com sua natureza mais analítica.

Por fim o sistema Android apresenta similaridade com o Linux, porém com uma maior simplicidade e menor dificuldade se comparado com Windows e Linux.

5. Conclusão

Este módulo nos apresenta um dos conceitos mais importantes e necessários no mundo da computação, o firewall e principalmente o conceito da defesa cibernetica, que é um tópico que possui extrema importância e deve ser enraizado no conhecimento geral de qualquer profissional de tecnologia.

Ao decorrer dos testes desenvolvemos a habilidade de adentrar e compreender o básico da manipulação e análise do firewall e ferramentas de mesma objetividade, entendemos a importância desses serviços para a segurança básica de qualquer dispositivo.

Em síntese, cada sistema oferece mecanismos robustos para controle de tráfego de rede, mas com diferentes níveis de complexidade e acessibilidade:

- Windows priorizando a usabilidade e integração de sistema
- Linux com sua tendência a usuários mais avançados.
- Android se mostra mais fechado priorizando uma segurança maior ao usuário final.

6. Autoavaliação

Principais dificuldades

Windows:

- A principal dificuldade desses testes foi o fato do risco que essas modificações e experimentações podem causar ao dispositivo, por não ser um ambiente controlado e separado do computador e ter uma troca direta com o sistema operacional é necessário cautela na hora de executar tais ações, porém com uma base de pesquisa e preparação anterior foi possível chegar a bons resultados e adquirir conhecimentos firmes sobre os temas abordados.

Android:

- Restrição inicial para obter root no emulador.
- Necessidade de reiniciar o adb em modo root.
- Diferenças entre bloqueio de HTTP e HTTPS.
- Possibilidade de perda total de conectividade ao bloquear múltiplas portas.
- Limitação de execução prática apenas no emulador Pixel 4a (API 36.0 com Google Play).

7. Referências

Firewall: <https://www.kaspersky.com.br/resource-center/definitions/firewall>

Vulnerabilidades de Firewall: <https://nsfocusglobal.com/pt-br/vulnerabilidade-de-firewall/>

Portas: <https://www.akamai.com/pt/glossary/what-are-ports>