



**INSTITUTO
FEDERAL**
Rio Grande do Norte

**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO RIO
GRANDE DO NORTE**

**DIRETORIA ACADÊMICA DE GESTÃO E TECNOLOGIA DA INFORMAÇÃO
PROJETO INTEGRADOR**

DOCUMENTAÇÃO PROJETO KYRIOS

José Bezerra
Kalvin Klein

Natal/RN
2024

JOSÉ BEZERRA
KALVIN KLEIN

DOCUMENTAÇÃO PROJETO KYRIOS

Orientador: Galileu Batista de Sousa

Natal/RN
2024

Sumário

Sumário.....	3
1 KYRIOS.....	4
2 PLATAFORMA WEB.....	5
2.1 Criação e Acesso de Usuário.....	5
2.2 Página Inicial, Menus e Submenus.....	9
3 FERRAMENTAS, TÉCNICAS, BANCOS DE DADOS.....	13
3.1 Tcpdump.....	13
3.2 Decompilando o APK.....	13
3.3 API VirusTotal.....	15
3.4 Biblioteca Whois.....	16
4 AMBIENTE DA ANÁLISE DINÂMICA.....	17
4.1 Emulador Android.....	17
4.2 ADB (Android Debug Bridge).....	18
4.2 Python.....	19
4.1 Docker.....	19
5 Considerações finais.....	20
6 Agradecimentos.....	20
Referências.....	20

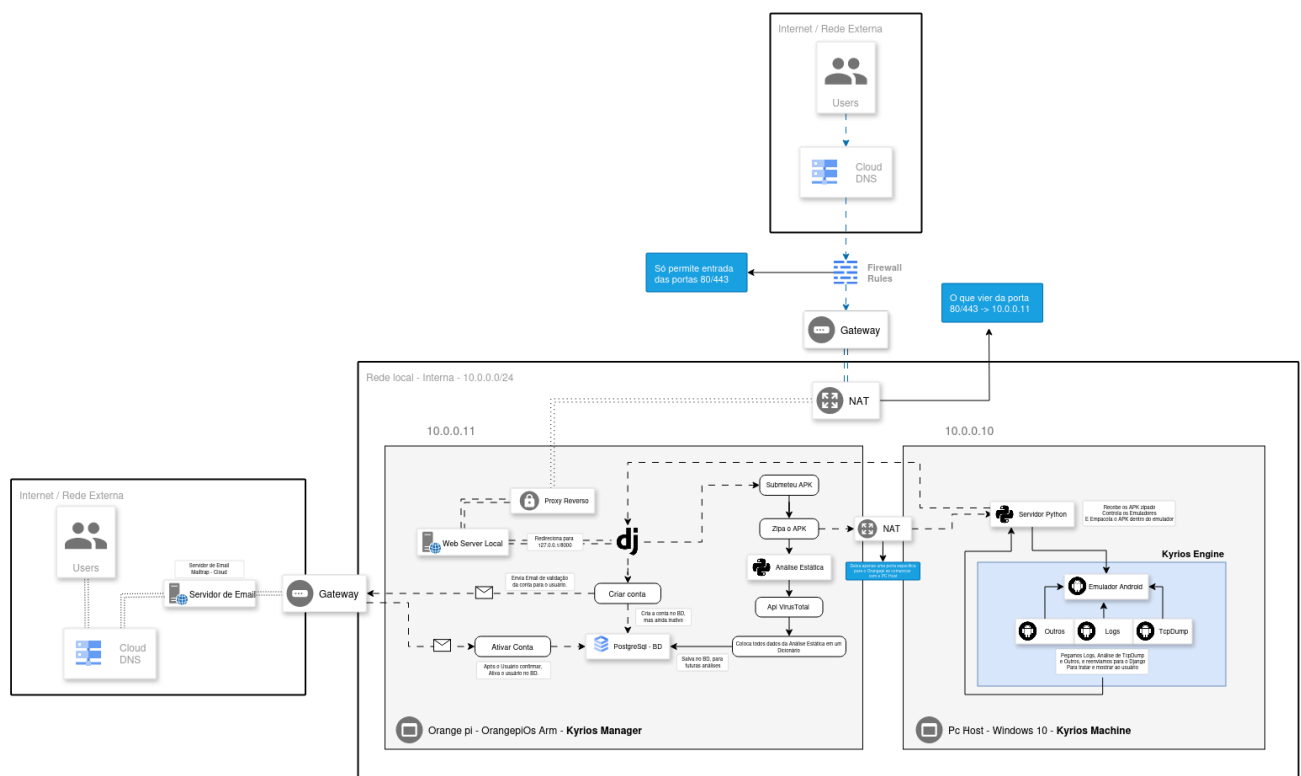
1 KYRIOS

Kyrios é uma plataforma web open-source (no endereço: www.github.com/kakanetwork/kyrios), utilizada para análise de programas maliciosos em arquivos *APK (Android Application Package)*. A nossa plataforma se destaca pela centralização e agilidade no acesso às informações sobre programas maliciosos, assim como a utilização de processos para análise dinâmica e estática dos arquivos em questão.

Kyrios trabalha com o backend de análise executado em contêineres Docker para um isolamento de processos. Android SDK Tools que inclui ferramentas essenciais para emulação de um ambiente Android. Linguagem Python para intercomunicação das etapas de análise. Framework Django para o desenvolvimento da página web para usuários que desejam apenas uma análise rápida.

Abaixo segue um fluxograma do projeto integrado com todas as ferramentas:

Figura 1 - Fluxograma de processos executados na solicitação de uma análise pelo usuário

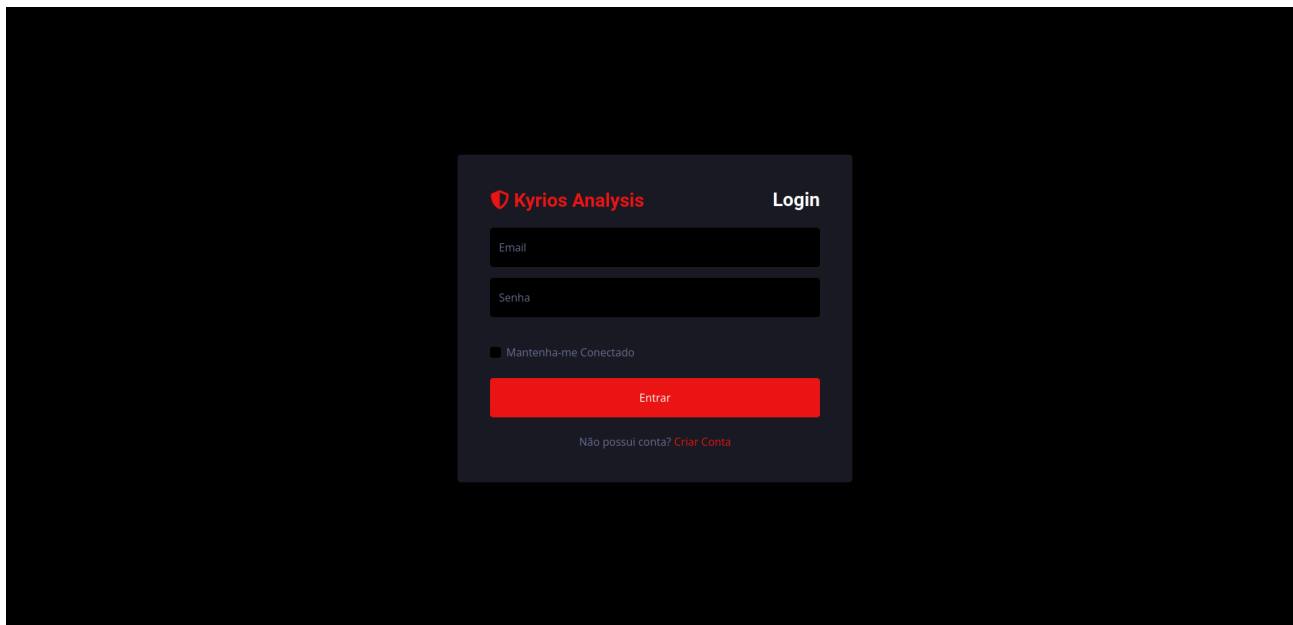


Fonte: Próprio autor

2 PLATAFORMA WEB

A plataforma web do projeto tem como propósito facilitar o acesso daqueles que buscam uma análise rápida e segura do seu *APK*. Para isso, os usuários devem acessar o endereço <https://analizador.cloud/> e ver a seguinte aba:

Figura 2 - página de login de usuário na plataforma web

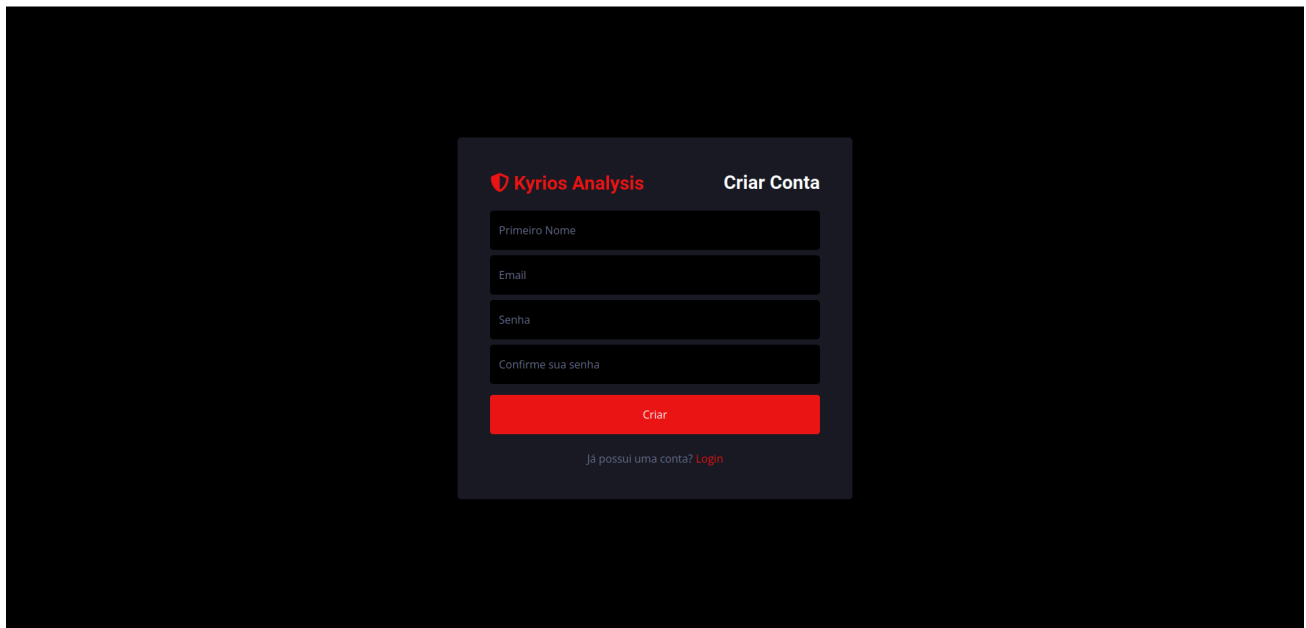


Fonte: Próprio Autor (<https://analizador.cloud/>)

Aqui o usuário deve preencher os campos de *e-mail* e senha se já possuir uma conta ou clicar em “Criar Conta” para uma nova conta, ao clicar em “Criar Conta” o usuário será encaminhado para uma página de criação de conta.

2.1 Criação e Acesso de Usuário

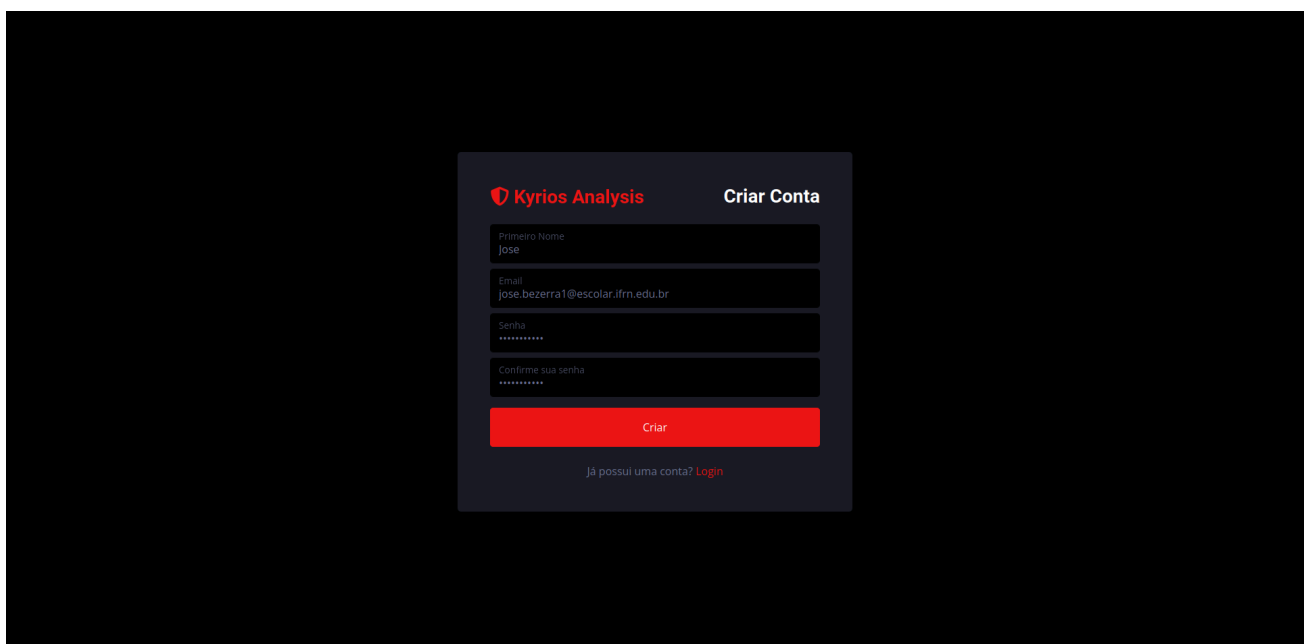
Figura 3 - Página para criação de conta para novos usuários



Fonte: Próprio Autor (<https://analizador.cloud/signup>)

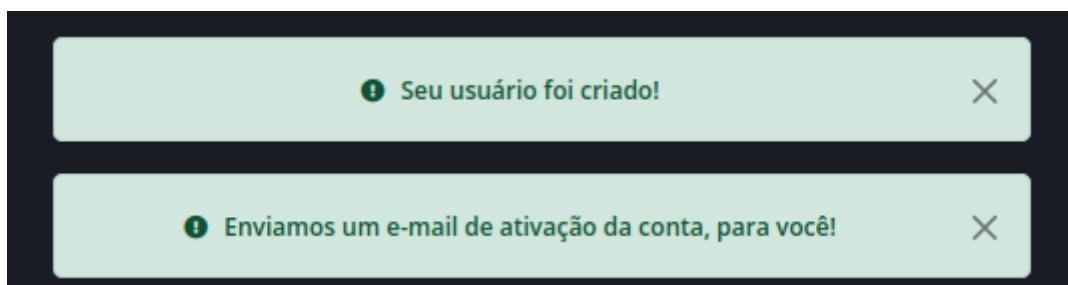
Para criar uma nova conta o usuário deve preencher os campos de “Primeiro Nome” (que será utilizado como nome de usuário dentro da plataforma). “*E-mail*” que deve ser válido porque será enviado um correio para confirmação de conta. “Senha” e “Confirmação de Senha” que devem conter no mínimo 8 caracteres sendo pelo menos um deles um número e uma letra. Ao preencher os campos e clicar no botão “Criar”, será notificado de que um *e-mail* de validação foi enviado, nesse *e-mail* o usuário deve confirmar a criação da conta para utilizar a plataforma. Ao verificar o *e-mail* de confirmação verificar se o envio foi considerado lixo eletrônico e ou *spam*.

Figura 4 - Exemplo de preenchimento de novo usuário



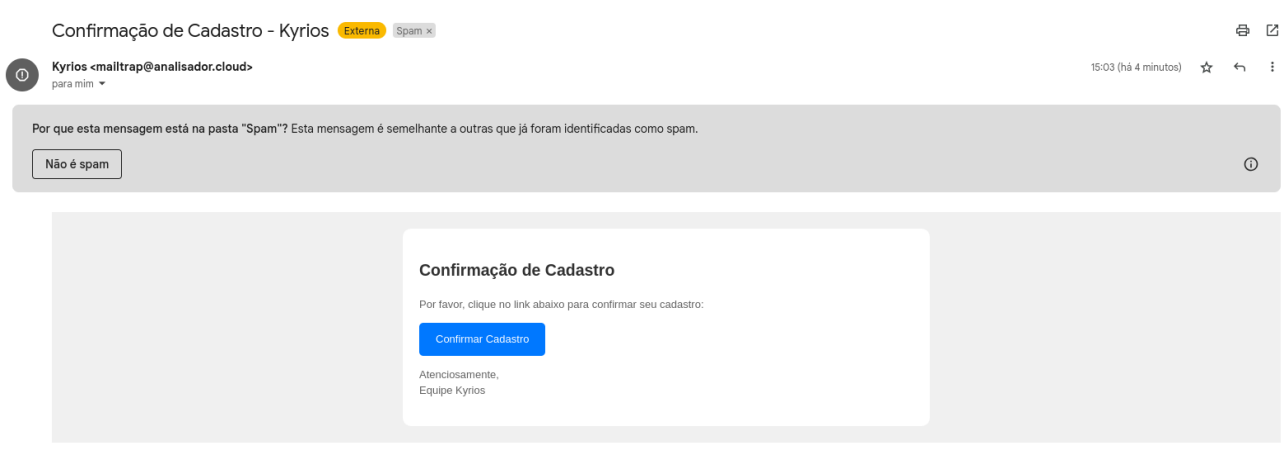
Fonte: Próprio autor (<https://analizador.cloud/signup>)

Figura 5 - Notificações de usuário criado e de *e-mail* enviado



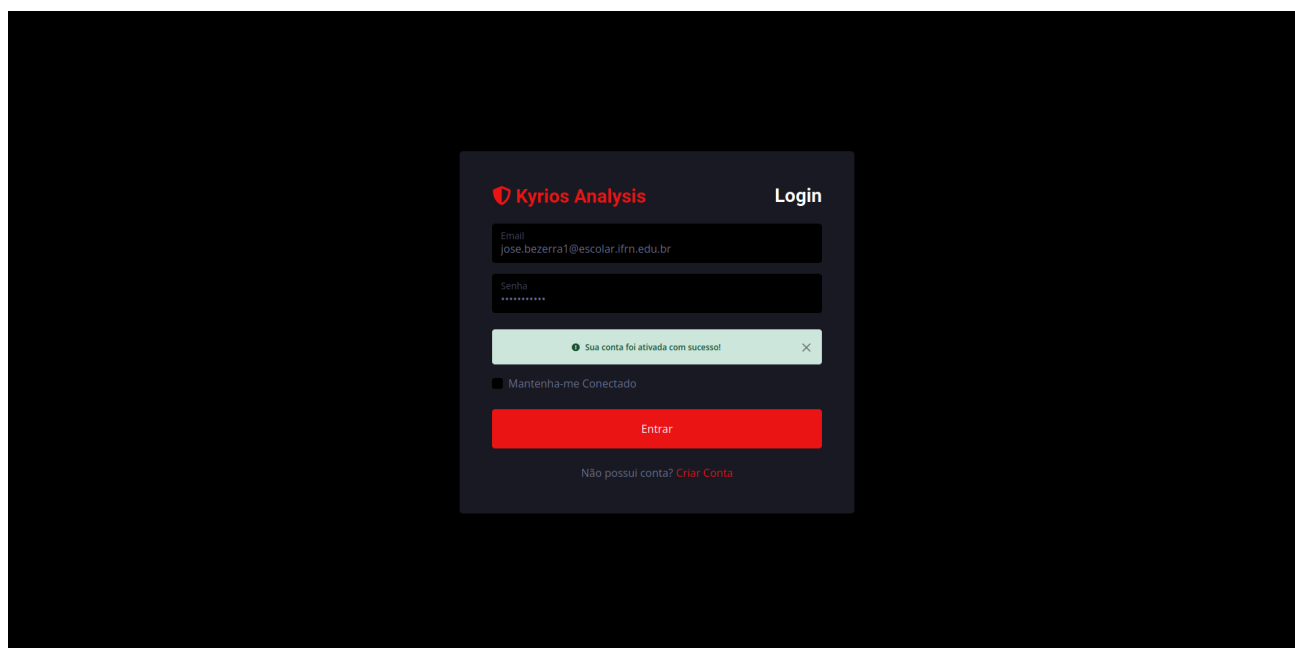
Fonte: Próprio autor (<https://analizador.cloud/signup>)

Figura 6 - *E-mail* de confirmação



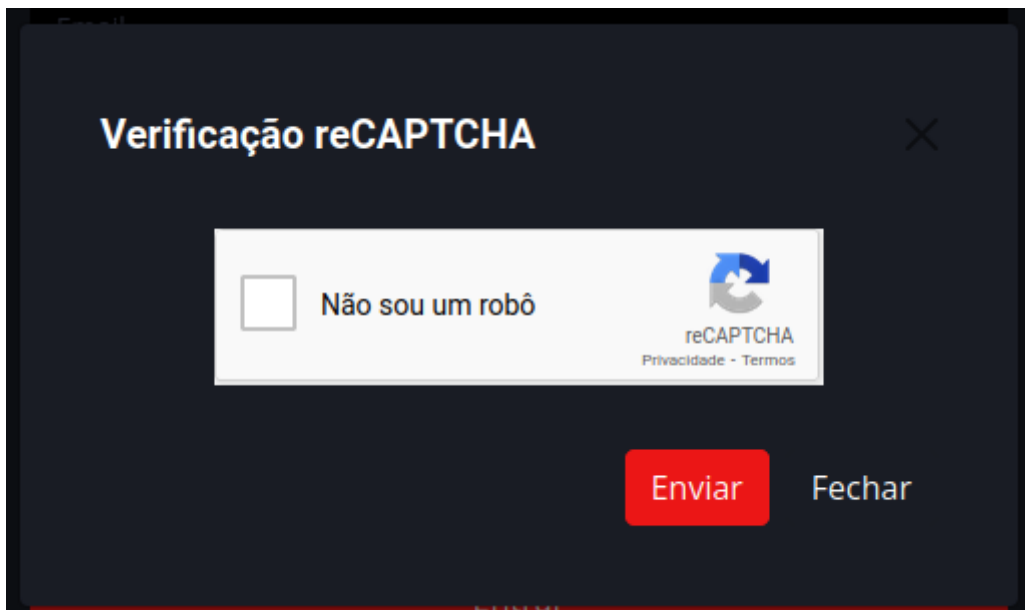
Fonte: Próprio autor

Figura 7 - Acessando a conta criada e confirmada



Fonte: Próprio autor

Figura 8 - reCaptcha para evitar a excessividade acessos simultâneos



Fonte: Próprio autor

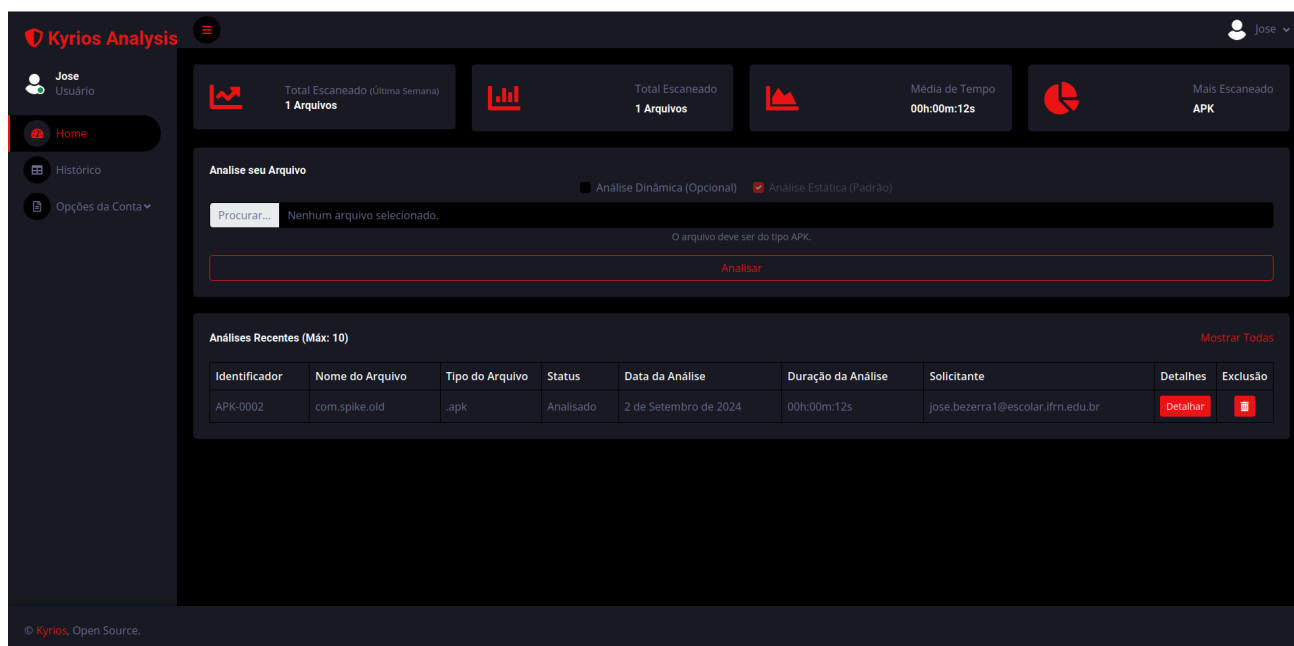
2.2 Página Inicial, Menus e Submenus

Com a efetivação do acesso, o usuário irá acessar a página inicial do seu perfil. Na página inicial haverá as seguintes informações:

- Total Escaneado (Última Semana): Índice que mostra quantos arquivos foram submetidos pelo usuário dentro de um prazo de 7 dias.
- Total Escaneado: Total de arquivos submetidos pelo usuário.
- Média de Tempo: Mostra o tempo total despendido de arquivos em análises
- Mais Escaneado: Exibe qual extensão de arquivo teve mais submissão.
- Analise seu Arquivo: Aqui é dado ao usuário duas opções Análise Dinâmica e Estática, a Dinâmica é mais lenta porém mais precisa, Estática é obrigatória e o mínimo necessário para que haja uma Análise.
 - Análise Dinâmica (Opcional): Opção que o usuário pode deixar selecionado ou não, para que o arquivo enviado seja analisado dinamicamente, ao selecionar essa opção o arquivo será executado para uma análise mais precisa de dados que são obtidos apenas na execução do arquivo .
 - Análise Estática: Faz uso de API, Banco de Dados e técnicas de análise de malware estática (por exemplo: obtenção da hash do arquivo).
- Botão Procurar...: Esse botão tem como função abrir o diretório de arquivos local para o usuário selecionar um arquivo.
- Botão Analisar: Botão que dará início ao processo de análise do arquivo selecionado pelo usuário.

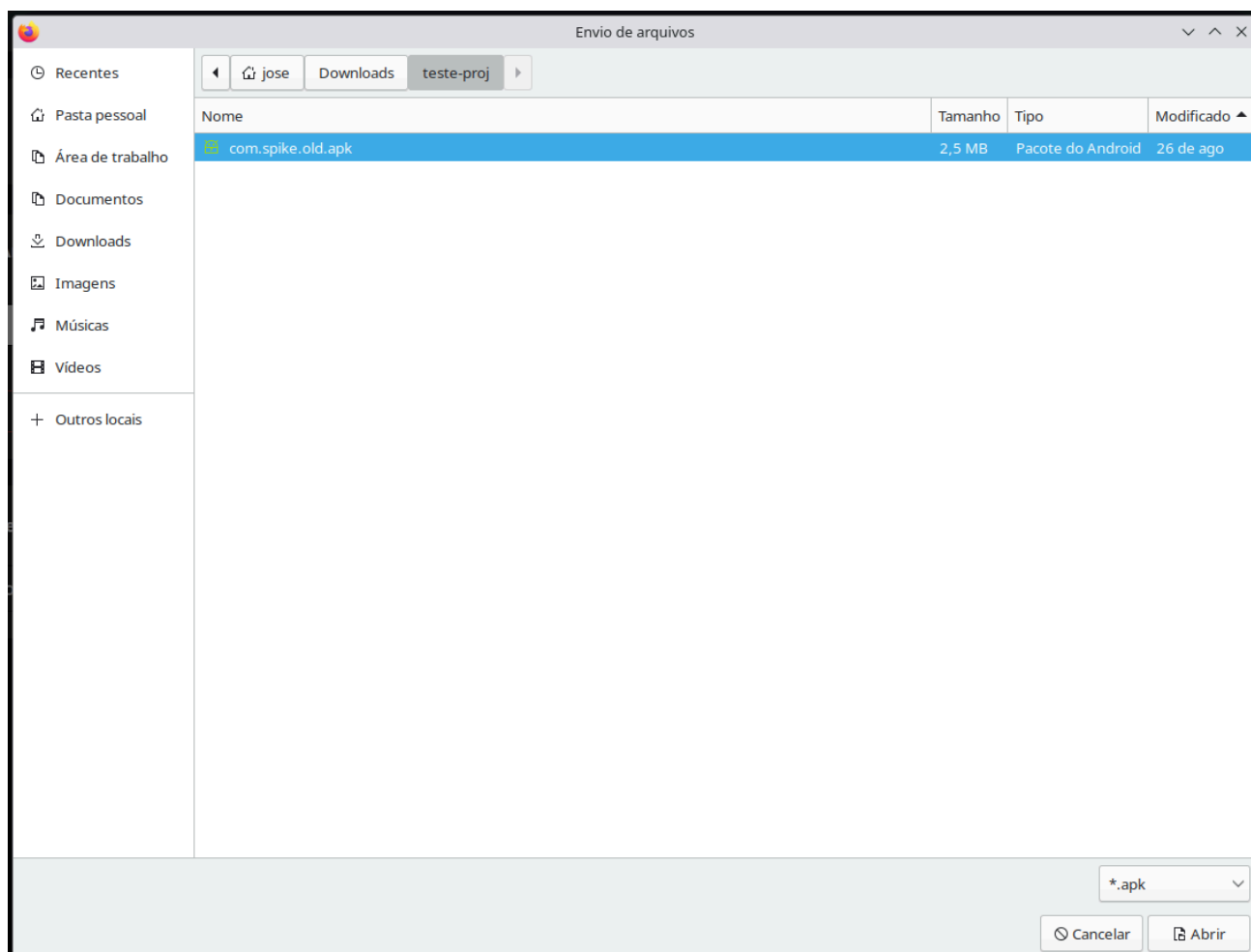
- **Análise Recentes:** Mostra o total das 10 últimas análises feitas pelo usuário, para acessar análises anteriores “Mostrar Todos”.
 - **Identificador:** Identificação dada ao arquivo quando é submetido ao processo de análise.
 - **Nome do Arquivo:** Nome original do arquivo enviado.
 - **Tipo do Arquivo:** Extensão do arquivo enviado.
 - **Status:** Status da análise do arquivo.
 - **Data da Análise:** Dia/Mês/Ano em que o arquivo foi enviado para análise
 - **Duração da Análise:** Tempo despendido para análise desse arquivo.
 - **Solicitante:** E-mail do usuário que enviou o arquivo
 - **Detalhes:** Botão para abrir mais detalhes da análise (Hash,IPs,URLs,Strings e etc)
 - **Exclusão:** Excluir o arquivo enviado do histórico
- **Histórico/Mostrar Todos:** Exibe todas os arquivos enviados pelo usuário e seus detalhes
- **Opções de Conta:**
 - **Criar uma nova conta:** Volta ao processo de criação de Conta
 - **Sair da Conta:** Volta para o processo de acesso de usuário

Figura 9 - Página inicial do usuário



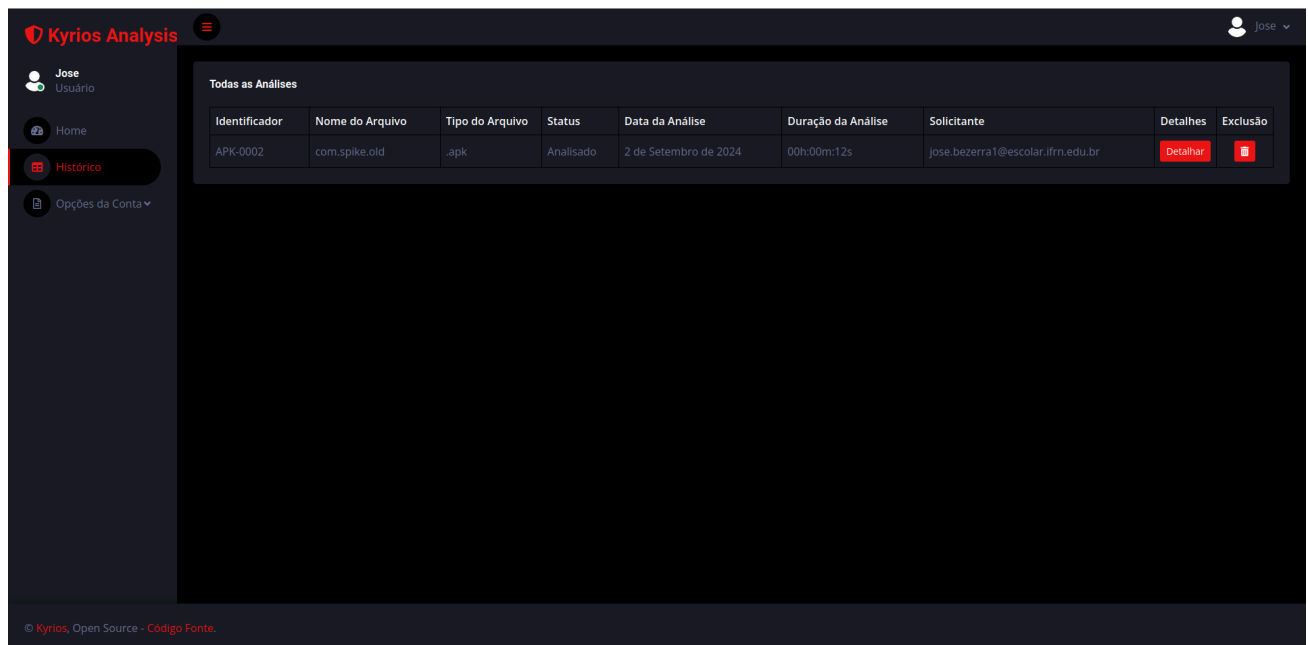
Fonte: Próprio autor (<https://analizador.cloud/home>)

Figura 10 - Gerenciador de Arquivos para selecionar o arquivo que vai ser enviado



Fonte: Próprio autor

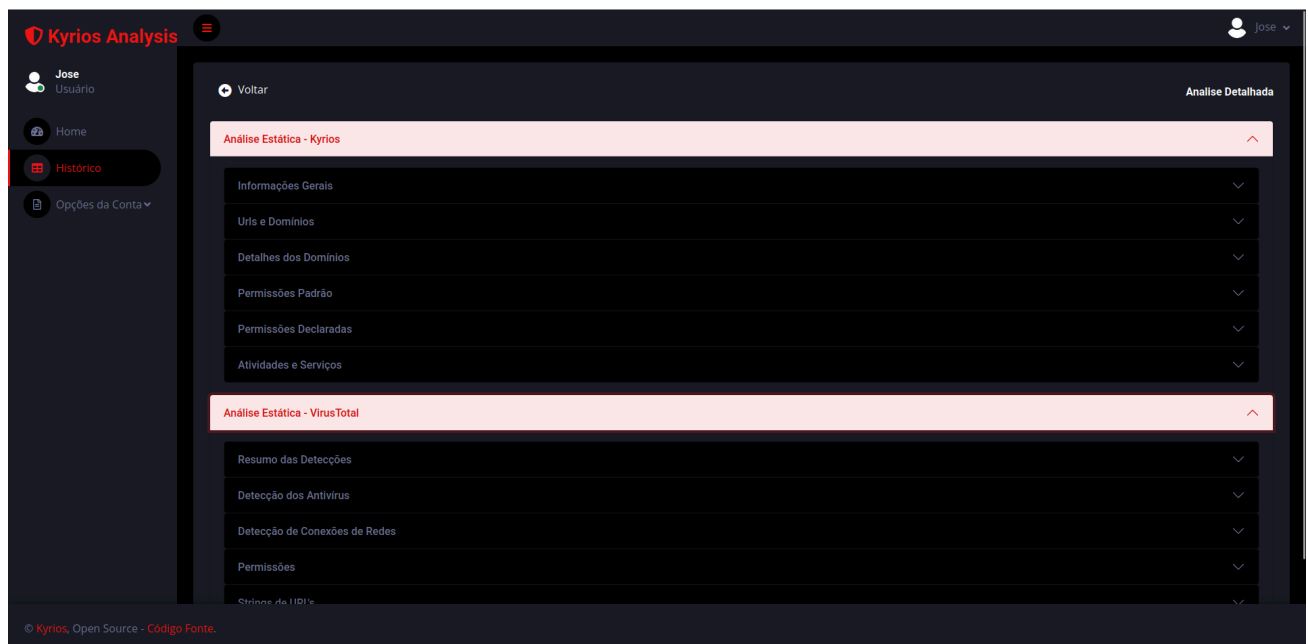
Figura 11 - Histórico completo de análises



Identificador	Nome do Arquivo	Tipo do Arquivo	Status	Data da Análise	Duração da Análise	Solicitante	Detalhes	Exclução
APK-0002	com.spike.old	.apk	Analisado	2 de Setembro de 2024	00h:00m:12s	jose.bezerra1@escolar.ifrn.edu.br	Detalhar	

Fonte: Próprio Autor (<https://analizador.cloud/table>)

Figura 12 - Detalhes da análise, página que exibe os resultados



Detalhes
Análise Estática - Kyrios
Informações Gerais
URLs e Domínios
Detalhes dos Domínios
Permissões Padrão
Permissões Declaradas
Atividades e Serviços
Análise Estática - VirusTotal
Resumo das Detecções
Detecção dos Antivírus
Detecção de Conexões de Redes
Permissões
Strings da IDI's

Fonte: Próprio Autor (<https://analizador.cloud/detalhar/APK-0002/>)

Com isso, o usuário consegue ter acesso rápido e fácil ao resultado das análises das ferramentas e resultados

3 FERRAMENTAS, TÉCNICAS, BANCOS DE DADOS...

Kyrios conta com ferramentas como TCPDUMP para captura de pacotes de rede confeccionados pela execução de arquivos maliciosos. Decompilador Java que retira do APK informações importantes como *AndroidManifest.xml*, esse arquivo que contém informações essenciais sobre o app para as ferramentas de build do Android, para o sistema operacional Android e para o Google Play, por exemplo, As permissões que o app precisa ter para acessar partes protegidas do sistema ou de outros apps. Também declare todas as permissões que outros apps precisam ter para acessar conteúdo desse app.

Kyrios também faz consulta a bancos de dados já consolidados para extrair mais informações já pertinentes e conhecidas sobre o arquivo enviado, essa consulta é feita através da API do VirusTotal (<https://docs.virustotal.com/docs/technology-integrations-list>) que conta com centenas de registros de engines de antivírus já cadastrados e muitos outros dados.

3.1 Tcpcdump

Tcpcdump (<https://www.tcpcdump.org/>) é um utilitário de rede que tem por função coletar pacotes trafegados na rede de computadores, com os cabeçalhos coletados, é possível distinguir quais tipos de informação o APK transmite. Essa ferramenta é altamente difundida e versátil sendo compatível com a maioria dos sistemas *Unix-like* sendo o Android um sistema linux modificado pelo Google para atender a sua demanda. A ferramenta tcpcdump já está presente originalmente nos sistemas Android e através da ferramenta de desenvolvedor adb (Android Debug Bridge) interagimos com a ferramenta tcpcdump por comandos no processo de análise dinâmica no emulador Android.

3.2 Decompilando o APK

O método de decompilação acontece através do uso de uma biblioteca do Python chamada pyaxmlparser (para mais informações: <https://github.com/appknox/pyaxmlparser>) que tem como função simples interpretar o arquivo Android XML chamado *AndroidManifest.xml* (<https://developer.android.com/guide/topics/manifest/manifest-intro?hl=pt-br>) que tem por função descrever informações essenciais sobre o app para as ferramentas de build do Android, para o sistema operacional Android e para o Google Play. Esse arquivo inclui os componentes do app, incluindo todas as atividades, serviços, broadcast receivers e provedores de conteúdo. As permissões que o app precisa ter para acessar partes protegidas do sistema ou de outros apps. Também declare todas as permissões que outros apps precisam

ter para acessar conteúdo desse app. Os recursos de hardware e software exigidos pelo app, que afetam quais dispositivos podem instalar o app pelo Google Play.

O sistema Android por via de regra, trabalha com todas essas permissões pré-estabelecidas nesse arquivo XML para poder compilar e executar o APK, com essas informações obtidas é possível estimar qual o intuito e objetivo do malware antes mesmo de sua execução.

Figura 12 - Exemplo de retorno do arquivo *AndroidManifest.XML*, onde é mostrado permissões, execuções e serviços utilizados pelo APK após sua compilação e execução

```
1,  
"perms_declaradas": [  
  "com.google.android.gms.permission.AD_ID",  
  "com.google.android.finsky.permission.BIND_GET_INSTALL_REFERRER_SERVICE",  
  "com.deerslab.dinoTREX.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION",  
  "com.deerslab.dinoTREX.DYNAMIC_RECEIVER_NOT_EXPORTED_PERMISSION"  
],  
"atv_principal": "com.deerslab.dinoTREX.AndroidLauncher",  
"todas_atv": [  
  "com.deerslab.dinoTREX.AndroidLauncher",  
  "com.google.android.gms.ads.AdActivity",  
  "com.google.android.gms.ads.OutOfContextTestingActivity",  
  "com.google.android.gms.auth.api.signin.internal.SignInHubActivity",  
  "com.yandex.mobile.ads.common.AdActivity",  
  "com.google.android.gms.common.api.GoogleApiActivity"  
],  
"todas_srv": [  
  "com.google.android.gms.ads.AdService",  
  "com.google.android.gms.auth.api.signin.RevocationBoundService",  
  "com.google.firebase.components.ComponentDiscoveryService",  
  "androidx.work.impl.background.systemalarm.SystemAlarmService",  
  "androidx.work.impl.background.systemjob.SystemJobService",  
  "androidx.work.impl.foreground.SystemForegroundService",  
  "com.yandex.metrica.MetricaService",  
  "com.yandex.metrica.ConfigurationService",  
  "com.yandex.metrica.ConfigurationJobService",  
  "com.google.android.gms.measurement.AppMeasurementService",  
  "com.google.android.gms.measurement.AppMeasurementJobService",  
  "com.google.android.datatransport.runtime.backends.TransportBackendDiscovery",  
  "com.google.android.datatransport.runtime.scheduling.jobscheduling.JobInfoSchedulerService",  
  "androidx.room.MultiInstanceInvalidationService"  
]  
}
```

Fonte - <https://github.com/appknox/pyaxmlparser>

Ficará também disponibilizado no repositório do projeto um arquivo json que destrincha todas as permissões possíveis de serem solicitadas por um APK ao sistema Android.

3.3 API VirusTotal

Durante o processo de análise o APK enviado, é submetido também a teste com a API do VirusTotal ao qual tem em seus registros milhões de dados de arquivos anteriormente enviados a eles. Entre esses dados, fazemos a coleta de testes executados em mecanismos de antivírus do mercado para avaliar se o APK enviado já tem histórico malicioso previamente registrado e apurado. Assim como também IDS's (Intrusion Detection System ou Sistemas de Detecção de Intrusão), por exemplo Snort que é um sistema de detecção/prevenção de intrusões baseado em rede de código aberto (IDS/IPS), tem a capacidade de realizar análise de tráfego em tempo real e registro de pacotes em redes de Protocolo de Internet (IP). O Snort realiza análise de protocolos, busca e correspondência de conteúdo.

Para saber mais sobre a capacidade disponibilizada pela API do VirusTotal acesse a documentação oficial deles no endereço <https://docs.virustotal.com/docs/api-overview>.

A nossa consulta à API é executada através da biblioteca *requests* do Python (<https://requests.readthedocs.io/en/latest/>), sendo essa uma biblioteca amplamente difundida para tratamento de requisições HTTP. A biblioteca requests também é recomendado pela própria documentação do VirusTotal para ser usada na interação com o *endpoint* da API. Junto a isso é requisitado também para aqueles que farão uso próprio da API seja cadastrado na plataforma do próprio VirusTotal e faça solicitação de uma Chave de API para que seja mantida em segredo, por isso não a exibimos nesta documentação.

Figura 13 - Exemplo de uso da biblioteca requests e uso da API do VirusTotal, onde a API retorna um erro da CHAVE da API ao enviar uma chave errada ou que não exista.

```
REQUEST

$ python -m pip install requests

1 import requests
2
3 url = "https://www.virustotal.com/api/v3/files/upload_ur
4
5 headers = {
6     "accept": "application/json",
7     "x-apikey": "APIKEY"
8 }
9
10 response = requests.get(url, headers=headers)
11
12 print(response.text)

Try It!
```

```
RESPONSE 401 LOG

1 {
2   "error": {
3     "code": "WrongCredentialsError",
4     "message": "Wrong API key"
5   }
6 }
```

Headers ↗

Fonte - <https://docs.virustotal.com/reference/files-upload-url>

3.4 Biblioteca Whois

Para verificação de domínios e IP's encontrados na execução ou no código fonte do APK, interagimos com uma biblioteca Python <https://pypi.org/project/python-whois/> . Essa biblioteca desempenha o papel de consultar diretamente um servidor WHOIS em vez de passar por um serviço web intermediário como muitos outros fazem. Após a interação com a biblioteca extraímos dados como nomes de domínios, data de criação do domínio, servidor de nomes e *e-mail* do administrador do domínio. Essa etapa é importante para ser possível destrinchar com “quem” o APK pode vir a se comunicar para realizar envio de informações sensíveis coletadas do usuário, servidores de comando e controle ou solicitações forçadas e enganosas de propagandas e publicidade que infringem direitos.

4 AMBIENTE DA ANÁLISE DINÂMICA

A análise dinâmica ocorre em um emulador do sistema Android disponibilizado pela Google através do endereço (<https://developer.android.com/develop?hl=pt-br>), onde se encontra também toda a documentação para utilização, configuração e gerenciamento do emulador. Por sua vez a emulação ocorre em um ambiente isolado de contêiner Docker, disponível em (<https://docs.docker.com/>), que desempenha papel separar a máquina host do sistema, e a máquina emulada trazendo uma camada a mais de proteção, a comunicação entre as máquinas host e emulador ocorre através de um pequeno sistema cliente/servidor desenvolvido com sockets na linguagem python, e o servidor que sobe ao ar juntamente com o docker se comunica com o emulador através da ferramenta adb.

4.1 Emulador Android

A emulação do sistema operacional Android (<https://developer.android.com/studio/run/emulator-commandline?hl=pt-br>), que é instalada pelo gerenciador de pacotes SDK (<https://developer.android.com/tools/sdkmanager?hl=pt-br>) que é uma ferramenta de linha de comando que permite visualizar, instalar, atualizar e desinstalar pacotes do SDK do Android.

Após instalação do gerenciador SDK e do emulador, a emulação ocorre com a execução do comando `'emulator -avd Nexus_XL_API_30 -no-window -no-audio -no-snapshot-load -no-snapshot-save -gpu off -no-boot-anim -wipe-data -no-metrics'`, por se tratar de um dispositivo *sandbox* virtual descartável, ele é criado com recursos limitados, desabilitados e características descartadas, o comando consiste dos seguintes argumentos:

- `emulator`: para iniciar o emulador.
- `-avd`: AVD permite definir as características de um celular Android, smartwatch Wear OS ou dispositivo Android TV que você quer simular no Android Emulator.
- `Nexus_XL_API_30`: nome do dispositivo smartphone emulado.
- `-no-window`: desabilita exibição da tela emulado, como toda a comunicação ocorre por linhas de comando, se torna desnecessária a emulação de uma tela, fazendo com que esse recurso apenas consuma desempenho desnecessariamente.
- `-no-audio`: assim como a tela o áudio é desabilitado por se tratar de uma comunicação apenas por linhas de comunicação, o recurso de áudio é inviável e desnecessário.
- `-no-snapshot-load`: essa característica faz com que o emulador não busque por snapshot do sistema sendo criado do zero reduzindo significativamente a inicialização da emulação.

- -no-snapshot-save: como não é feito o carregamento de snapshot, para evitar o uso de espaço de armazenado desnecessário também é ignorado o salvamento.
- -gpu off: por ser desabilitado o uso da tela, o recurso de emulação de uma placa gráfica se torna desnecessário.
- -no-boot-anim: Desativa a animação de inicialização do Android, reduzindo o tempo de boot.
- -wipe-data: Apaga todos os dados de usuário do AVD, restaurando-o para o estado de fábrica a cada inicialização, utilizado para que os ambientes subsequentes não sejam afetados pelos anteriores.
- -no-metrics: Desativa a coleta e exibição de métricas do emulador, como desempenho e uso de recursos.

Com esse comando é possível criar uma emulação de um dispositivo Android, todos esses comando estão embutidos na criação do contêiner Docker sendo assim automatizada quando o próprio vai ao ar.

4.2 ADB (Android Debug Bridge)

A ferramenta Android Debug Bridge, é uma ferramenta de linha de comando versátil que permite a comunicação entre um dispositivo Android (ou um emulador) e um computador. ADB faz parte do SDK do Android e é amplamente utilizado para depuração, instalação de aplicativos, acesso ao shell de Unix, transferência de arquivos, e outras operações de gerenciamento em dispositivos Android.

O ADB traz as funcionalidades de Depuração de Aplicativos permitindo conectar-se a dispositivos ou emuladores Android para depurar aplicativos em execução, oferecendo suporte a logs, breakpoints, e inspeção de estado do aplicativo. Instalação e desinstalação de aplicativos diretamente no dispositivo, facilitando o fluxo de testes. Acesso ao Shell de Unix no dispositivo Android, permitindo a execução de comandos diretamente no sistema operacional do dispositivo. Transferência de Arquivos entre o computador e o dispositivo Android, útil para manipulação de dados. Exemplos de Comandos:

- adb install app.apk: Inicia o processo de instalação de um aplicativo no dispositivo.
- adb shell: Abre um ambiente de linha de comando (shell) no dispositivo Android.
- adb push local_path /remote_path ou adb pull /remote_path local_path: Transfere arquivos ou diretórios do computador para o dispositivo Android e vice versa

ADB é uma ferramenta essencial que oferece uma interface robusta para interagir com dispositivos Android durante o ciclo de testes e implantação de aplicativos. Sua versatilidade

e ampla gama de comandos tornam o ADB uma peça chave para a análise de trabalho Android.

4.2 Python

Este projeto implementa uma aplicação cliente/servidor utilizando sockets em Python (<https://docs.python.org/3/library/socket.html>), onde o servidor interage com a ferramenta Android Debug Bridge (ADB) para executar comandos em dispositivos Android conectados. O objetivo é fornecer uma interface simplificada para controlar dispositivos Android remotamente por meio de comandos enviados pelo cliente.

O cliente é a máquina host que está hospedando a plataforma, ou localmente é a máquina hospedando o Docker que contém a emulação. E o Servidor é o Docker com a ferramenta ADB, o emulador e demais ferramentas de análise.

4.1 Docker

Utilizamos um contêiner Docker (<https://docs.docker.com/>) para fornecer um ambiente isolado, consistente e facilmente replicável para executar as análises dinâmicas. O contêiner permite encapsular as ferramentas utilizadas na análise, o emulador juntamente com a ferramenta ADB e o servidor Python para comunicação e integração com os objetos citados anteriormente.

O Dockerfile utilizado para confecção desse ambiente encontrasse no repositório do projeto (<https://github.com/kakanetwork/Kyrios>), após o processo de construção do contêiner, as análises subsequentes são apenas replicações dessa construção, para execução do contêiner o comando `docker run --device /dev/kvm -d [id_imagem]` coloca no ar o Docker:

- `docker run`: Este comando é usado para iniciar um novo contêiner a partir de uma imagem Docker especificada.
- `--device /dev/kvm`: A opção `--device` permite mapear um dispositivo do host para dentro do contêiner. No caso, `/dev/kvm` (Kernel-based Virtual Machine) é um dispositivo que permite a virtualização acelerada por hardware em sistemas Linux.
- `-d`: A flag `-` (detached mode) faz com que o contêiner seja executado em segundo plano (background), liberando o terminal para outros comandos.
- `[id_imagem]`: Especifica a imagem Docker a ser usada para criar o contêiner, nesse caso a criada a partir do Dockerfile do projeto.

Esta abordagem com Docker melhora a eficiência operacional do projeto, criar camadas de segurança e acelerando o ciclo de renovação do ambiente de análise

5 Considerações finais

Pretendemos a posteriori adicionar novas funcionalidades, melhorar eficiência dos resultados e tornar esse projeto algo mais próxima do idealizado a princípio, dado o curto prazo para a entrega acadêmica, alguns itens que gostaríamos que estivessem presentes no projeto serão deixados para atualizações futuras, e o repositório estará aberto como open source para todos aqueles que queiram contribuir com a idéia, o que está por vir:

- Regras Yara
- Melhor integração para múltiplas análises
- Ferramenta Frida
- Wireshark
- Scripts de análise específicos

6 Agradecimentos

Gostaríamos de expressar nossa sincera gratidão ao Professor Galileu por sua orientação e apoio durante o desenvolvimento deste projeto, e também agradecemos ao Professor Moisés Souto seus feedbacks críticos foram fundamentais para nosso aprendizado e desenvolvimento.

Em nome de José Bezerra e Calvin Klein

Referências

1. **DOCKER**, Inc. Docker Documentation. Disponível em: <https://docs.docker.com/>. Acesso em: 5 set. 2024.
2. **GOOGLE**. Android Developers. Disponível em: <https://developer.android.com/?hl=pt-br>. Acesso em: 5 set. 2024.
3. **TCPDUMP**. *Documentação*. Disponível em: <https://www.tcpdump.org/index.html#documentation>. Acesso em: 5 set. 2024.
4. **DJANGO**. *Documentação*. Disponível em: <https://docs.djangoproject.com/en/5.1/>. Acesso em: 5 set. 2024.
5. **PYTHON SOFTWARE FOUNDATION**. *Documentação do Python*. Disponível em: <https://docs.python.org/3/>. Acesso em: 5 set. 2024.
6. **VIRUSTOTAL**. *Documentação da API*. Disponível em: <https://docs.virustotal.com/reference/overview>. Acesso em: 5 set. 2024.
7. **HAN, Q.; MANDUJANO, S.; PORST, S.; SUBRAHMANIAN, V. S.; TETALI, S. D.; XIONG, Y.** *The Android Malware Handbook: Detection and Analysis by Human and Machine*. San Francisco: No Starch Press, 2023.