



Desenvolvimento de Aplicações OpenRAN

**Avaliação de Risco
Cibernético e Requisitos de
Segurança**

M3 - A3.5

Programa OpenRAN@Brasil - Fase 2

Funttel

Sumário

Lista de ilustrações	4
Lista de tabelas	5
Glossário	6
1 Introdução	7
2 Avaliação de risco de acordo com as etapas de desenvolvimento	10
2.1 Atividades do <i>Pentest</i>	11
2.1.1 Etapa de Preparação	11
2.1.2 Etapa de Reconhecimento	12
2.1.3 Etapa de Exploração	14
2.2 Riscos	17
2.2.1 Riscos do acesso remoto pelo protocolo SSH	17
2.2.2 Riscos da engenharia social	18
2.2.3 Riscos do <i>cluster Kubernetes</i> do Near-RT RIC	19
2.3 Recomendações	20
2.3.1 Protocolo SSH	20
2.3.2 Engenharia Social	22
2.3.3 <i>Cluster kubernete</i> do Near-RT RIC	24
2.4 Versões seguras da <i>Blueprint</i>	28
3 Relatório de avaliação de risco da solução finalizada	30
4 Descrição de requisitos e ferramentas para a melhoria de segurança ciber- nética da O-RU	31
5 Conclusão	33
6 Histórico de versões deste documento	34

7 Execução e aprovação 35

Lista de ilustrações

Figura 1 – Arquitetura dos componentes da <i>Blueprint</i> v1.	10
Figura 2 – Etapas do <i>textitPentest Gray Box</i>	11
Figura 3 – Pontos de segurança analisados nos arquivos de configuração do nó Master do cluster.	13
Figura 4 – Resumo dos pontos analisados no cluster.	14
Figura 5 – Terminal do agente malicioso acessando a <i>Blueprint</i>	15
Figura 6 – Terminal do agente malicioso após a conclusão do <i>Phishing</i>	17
Figura 7 – Amostra do teste de verificação da ferramenta CIS Kubernetes Benchmark	20
Figura 8 – Resultado quantitativo do teste de verificação da ferramenta CIS Ku- bernetes Benchmark , após a implementação das recomendações.	29
Figura 9 – Pontos que foram corrigidos e passam no teste de segurança.	29

Lista de tabelas

Tabela 1 – Mapeamento das soluções propostas para as questões-chave identificadas no documento <i>Study on O-RU Centralized User Management</i> da O-RAN Alliance	32
---	----

Glossário

Acrônimos

3GPP *3rd Generation Partnership Project*

API *Application Programming Interface*

IaC *Infraestrutura como Código*

NSA *Agência de Segurança Nacional*

OSC *O-RAN Software Community*

OWASP *Open Web Application Security Project*

RAN *Radio Access Network*

RIC *RAN Intelligent Controller*

SDL *Shared Data Layer*

SLSA *Supply-chain Levels for Software Artifacts*

STRIDE *Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege*

1 Introdução

Nos últimos anos, a evolução das redes de comunicação móvel tem sido impulsionada pela busca por maior flexibilidade, eficiência e redução de custos. Nesse contexto, a tecnologia Open-RAN (*Open Radio Access Network*) emergiu como um complemento importante aos esforços realizados para evoluir a RAN, introduzindo programabilidade, inteligência e interfaces abertas. Open-RAN oferece a possibilidade de desagregar os componentes da estação base, permitindo a interoperabilidade entre hardware e software de diferentes fornecedores. Isso não apenas promove a inovação, mas também cria um ecossistema mais dinâmico e competitivo.

Com o advento de Open-RAN, em especial a iniciativa O-RAN Alliance, surgem oportunidades significativas para o desenvolvimento de aplicações para atender demandas específicas de redes de acesso por rádio. Essas aplicações podem ser implementadas utilizando interfaces abertas e padronizadas, para melhorar a eficiência operacional, otimizar o desempenho da rede e proporcionar serviços diferenciados aos usuários finais. Este relatório aborda a segurança, que é um ponto que exige extrema atenção nas redes estruturadas pelo conceito O-RAN. Por se basear no padrão do 3GPP e, também, por fomentar a questão de redes virtualizadas, o O-RAN terá um impacto significativo em vários riscos já identificados nas redes 5G. No entanto, por ter como um dos principais pilares a questão de impulsionar uma rede desagregada com equipamentos e softwares de múltiplos fornecedores, surgem novos desafios de segurança cibernética. Neste contexto, foi analisada a plataforma de suporte, OpenRAN@Brasil *Blueprint*, bem como o desenvolvimento de xApps e rApps, componentes cruciais para a operação eficiente e otimizada das redes Open-RAN. Esses aplicativos desempenham um papel fundamental na

orquestração e automação das funções de rede, proporcionando melhorias significativas em termos de desempenho, gestão de recursos e experiência do usuário. A capacidade de desenvolver xApps e rApps independentes do fornecedor de hardware permite uma personalização mais granular das redes móveis, adaptando-as às necessidades específicas de diferentes operadoras e cenários de implementação.

As xApps são projetadas para operar na camada de controle próximo a tempo real, sendo implementadas no *Near-Real-Time RAN Intelligent Controller* (Near-RT RIC). Elas são responsáveis por tarefas críticas, como otimização de tráfego, alocação de espectro e gestão de interferências, respondendo rapidamente às mudanças nas condições de rede. Por outro lado, as rApps são executadas no *Non-Real-Time RAN Intelligent Controller* (Non-RT RIC) e focam em funções de otimização que não exigem respostas imediatas, como planejamento de rede a longo prazo e análise preditiva.

O desenvolvimento desses aplicativos envolve uma combinação de técnicas avançadas de programação e Inteligência Artificial/Aprendizado de Máquina (do inglês, *Machine Learning/Artificial Intelligence*, além de um profundo entendimento das especificidades das redes de telecomunicações. Este relatório se refere à **Atividade 3.5 - Avaliar o risco cibernético e estabelecer requisitos de segurança**, a qual foi dividida nas seguintes seções:

- Avaliação de risco de acordo com as etapas de desenvolvimento;
- Avaliação de risco da solução finalizada;
- Descrição de requisitos e ferramentas para a melhoria de segurança cibernética da O-RU.

Este relatório corresponde a um dos entregáveis do Projeto Open-RAN@Brasil desenvolvido em parceria entre a Rede Nacional de Ensino e Pesquisa (RNP), Universidade Federal de Campina Grande (UFCG) e Universidade Federal de Goiás (UFG).

Objetivos do Relatório

O objetivo principal deste relatório é apresentar os resultados relacionados à Atividade 3.5, os quais envolvem testes e recomendações de segurança alinhados à padronização O-RAN para \times Apps. Vale destacar que esta atividade ainda está em andamento e, portanto, o presente relatório apresenta os resultados em seu estado atual, o qual terá evoluções até a conclusão.

Ao longo da realização da Atividade 3.5 foram identificadas algumas demandas que não estavam claras durante o planejamento inicial. Na verdade, isso era previsto dado que até mesmo os padrões da O-RAN estão em desenvolvimento e o software relacionado passa por atualizações regulares. Assim, algumas soluções propostas não foram inseridas no estado atual, mas poderão ser implementadas considerando versões futuras de desenvolvimento. Abordar essas necessidades são ações importantes e objetivos secundários relevantes.

2 Avaliação de risco de acordo com as etapas de desenvolvimento

A OpenRAN@Brasil *Blueprint* é uma imagem de máquina virtual que contém um ambiente já instalado e configurado para desenvolver e testar xApps no Near-RT RIC implementado pela *Open-RAN Software Community* (OSC). A Figura 1 mostra a arquitetura da *Blueprint*.

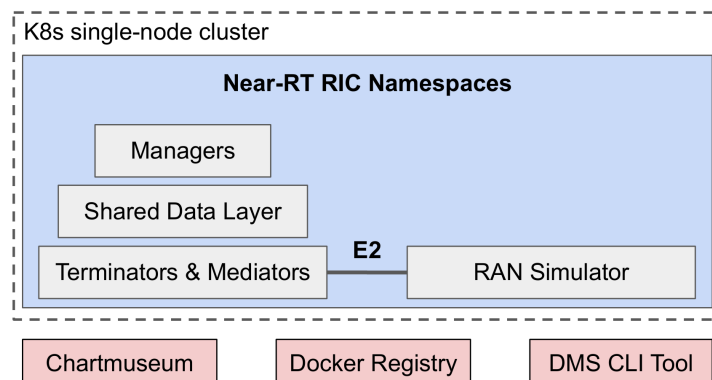


Figura 1 – Arquitetura dos componentes da *Blueprint* v1.

Neste ambiente virtual, foi realizado o *Pentest Gray Box*, que é uma abordagem híbrida combinando elementos do *Pentest White Box* e do *Pentest Black Box*. Nesse tipo de teste, o *pentester* possui algumas informações sobre o sistema alvo, mas não tem acesso completo a todos os detalhes. Isso permite uma avaliação mais realista da segurança dos sistemas, pois simula cenários em que um atacante externo tem conhecimento limitado.

Além disso, em comparação com o *Pentest Black Box*, reduz o risco de interrupção de serviços críticos durante o teste e permite uma investigação mais aprofundada

do ambiente, resultando em uma detecção mais precisa de vulnerabilidades. No decorrer deste documento, serão mencionados os riscos encontrados, incluindo descrição das vulnerabilidades, impacto potencial e recomendações de mitigação.

2.1 Atividades do *Pentest*

O teste de segurança feito na *Blueprint* possui as etapas mostradas na Figura 2.

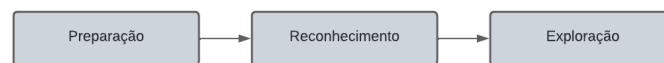


Figura 2 – Etapas do *textitPentest Gray Box*.

2.1.1 Etapa de Preparação

Em primeiro lugar, foi definido o escopo do *Pentest* no ambiente da *Blueprint* BOV-1-20240419_vBox, que consiste na análise de vulnerabilidades presentes na máquina virtual e no *cluster Kubernetes* em que foi implementado o Near-RT RIC da OSC. Com essa análise, é possível identificar os potenciais impactos na segurança deste ambiente virtual e propor recomendações para mitigação desses riscos. Além disso, foi proposto o desenvolvimento de versões seguras da *Blueprint*.

Para a realização dessa atividade, foram utilizadas ferramentas de cibersegurança como o **Nmap**, **Metasploit** e o **CIS Kubernetes Benchmark**. O **Nmap** é um utilitário gratuito e de código aberto para varredura de redes e auditoria de segurança que usa pacotes IP brutos de novas maneiras para determinar quais *hosts* estão disponíveis na rede, quais serviços (nome do aplicativo e versão) esses *hosts* estão oferecendo e quais sistemas operacionais (e versões do sistema operacional) eles estão executando. Essa ferramenta funciona bem em *hosts* únicos, como no caso da *Blueprint*.

Já o **Metasploit** é uma ferramenta amplamente utilizada para testes de penetração e avaliação de vulnerabilidades. Ele fornece um *framework* que simplifica

o processo de desenvolvimento, teste e execução de *exploits* contra uma ampla gama de alvos. No cenário da *Blueprint*, o **Metasploit** ajuda a verificar vulnerabilidades e gerenciar avaliações de segurança.

Por fim, o **CIS Kubernetes Benchmark** é um recurso valioso para proteger os *clusters Kubernetes* de forma eficaz. Ele fornece orientação detalhada e práticas recomendadas para reduzir os riscos de segurança, cumprir os padrões do setor e garantir que seu ambiente *Kubernetes* seja configurado de forma segura e resiliente contra possíveis ameaças. Essa ferramenta é essencial para avaliar o *cluster Kubernetes* em que foi implementado o Near-RT RIC da OSC e, seguindo as diretrizes descritas no **CIS Kubernetes Benchmark**, podem-se reduzir vulnerabilidades comuns associadas ao *Kubernetes*, como acesso não autorizado, configurações de contêiner inseguras e informações confidenciais expostas, protegendo as aplicações como xApps e dados importantes dentro do *cluster Kubernetes*. Além disso, ajuda a cumprir requisitos normativos e padrões do setor relacionados à segurança.

2.1.2 Etapa de Reconhecimento

Durante a fase de reconhecimento, foi possível identificar informações relevantes sobre o sistema alvo por meio da utilização da ferramenta **Nmap**. As informações adquiridas são listadas a seguir.

- Endereço IP: 192.168.122.182
- Sistema Operacional: Linux
- Portas Abertas:
 - 22/tcp (SSH)
 - 5001/tcp (complex-link)
 - 8090/tcp (opsmessaging)

Tendo o conhecimento sobre a arquitetura de software implementada na *Blueprint*, como mostrado na Figura 1, é evidente a necessidade de uma varredura no *cluster Kubernetes* do Near-RT RIC da OSC. Esse procedimento foi realizado usando a ferramenta **CIS Kubernetes Benchmark**, que promove uma análise de segurança no *cluster*, fornecendo orientação detalhada e práticas recomendadas para reduzir os riscos de segurança.

Após executar a varredura, a ferramenta expõe os pontos que estão atendendo aos requisitos de segurança e aqueles que precisam ser analisados e corrigidos. Esses pontos são classificados em **PASS**, **WARM** e **FAIL**. A Figura 3 apresenta uma amostra do resultado deste reconhecimento.

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.3 Ensure that the controller manager pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.4 Ensure that the controller manager pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.5 Ensure that the scheduler pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.6 Ensure that the scheduler pod specification file ownership is set to root:root (Automated)
[PASS] 1.1.7 Ensure that the etcd pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.8 Ensure that the etcd pod specification file ownership is set to root:root (Automated)
[WARN] 1.1.9 Ensure that the Container Network Interface file permissions are set to 644 or more restrictive (Manual)
[WARN] 1.1.10 Ensure that the Container Network Interface file ownership is set to root:root (Manual)
[PASS] 1.1.11 Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)
[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)
[PASS] 1.1.13 Ensure that the admin.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.14 Ensure that the admin.conf file ownership is set to root:root (Automated)
[PASS] 1.1.15 Ensure that the scheduler.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.16 Ensure that the scheduler.conf file ownership is set to root:root (Automated)
[PASS] 1.1.17 Ensure that the controller-manager.conf file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.18 Ensure that the controller-manager.conf file ownership is set to root:root (Automated)
[PASS] 1.1.19 Ensure that the Kubernetes PKI directory and file ownership is set to root:root (Automated)
[PASS] 1.1.20 Ensure that the Kubernetes PKI certificate file permissions are set to 644 or more restrictive (Manual)
[PASS] 1.1.21 Ensure that the Kubernetes PKI key file permissions are set to 600 (Manual)
[INFO] 1.2 API Server
[WARN] 1.2.1 Ensure that the --anonymous-auth argument is set to false (Manual)
[PASS] 1.2.2 Ensure that the --basic-auth-file argument is not set (Automated)
[PASS] 1.2.3 Ensure that the --token-auth-file parameter is not set (Automated)
[PASS] 1.2.4 Ensure that the --kubelet-https argument is set to true (Automated)
[PASS] 1.2.5 Ensure that the --kubelet-client-certificate and --kubelet-client-key arguments are set as appropriate (Automated)
[FAIL] 1.2.6 Ensure that the --kubelet-certificate-authority argument is set as appropriate (Automated)
[PASS] 1.2.7 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Automated)
[PASS] 1.2.8 Ensure that the --authorization-mode argument includes Node (Automated)
[PASS] 1.2.9 Ensure that the --authorization-mode argument includes RBAC (Automated)
[WARN] 1.2.10 Ensure that the admission control plugin EventRateLimit is set (Manual)
[PASS] 1.2.11 Ensure that the admission control plugin AlwaysAdmit is not set (Automated)
[WARN] 1.2.12 Ensure that the admission control plugin AlwaysPullImages is set (Manual)
[WARN] 1.2.13 Ensure that the admission control plugin SecurityContextDeny is set if PodSecurityPolicy is not used (Manual)
[PASS] 1.2.14 Ensure that the admission control plugin ServiceAccount is set (Automated)
[PASS] 1.2.15 Ensure that the admission control plugin NamespaceLifecycle is set (Automated)
[FAIL] 1.2.16 Ensure that the admission control plugin PodSecurityPolicy is set (Automated)
[PASS] 1.2.17 Ensure that the admission control plugin NodeRestriction is set (Automated)
[PASS] 1.2.18 Ensure that the --insecure-bind-address argument is not set (Automated)
[PASS] 1.2.19 Ensure that the --insecure-port argument is set to 0 (Automated)
[PASS] 1.2.20 Ensure that the --secure-port argument is not set to 0 (Automated)
[FAIL] 1.2.21 Ensure that the --profiling argument is set to false (Automated)
[FAIL] 1.2.22 Ensure that the --audit-log-path argument is set (Automated)
[FAIL] 1.2.23 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)
[FAIL] 1.2.24 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)
[FAIL] 1.2.25 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)
```

Figura 3 – Pontos de segurança analisados nos arquivos de configuração do nó Master do cluster.

Além disso, o **CIS Kubernetes Benchmark** fornece um resumo quantitativo dos pontos verificados, e, no caso do *cluster* do Near-RT RIC da OSC, obteve-se o

resultado mostrado na Figura 4.

```
== Summary policies ==
0 checks PASS
0 checks FAIL
24 checks WARN
0 checks INFO

== Summary total ==
70 checks PASS
11 checks FAIL
41 checks WARN
0 checks INFO
```

Figura 4 – Resumo dos pontos analisados no cluster.

Logo, essas informações de reconhecimento sobre a *Blueprint* permitiram definir dois pontos de exploração nessa máquina virtual: uma exploração via SSH e a realização de uma simulação de engenharia social usando o *Spear Phishing*. Além disso, os dados obtidos pelo **CIS Kubernetes Benchmark** evidenciaram que correções nos arquivos de configuração do *cluster* são necessárias para melhorar a segurança do *cluster Kubernetes* e, pensando nisso, soluções são mencionadas na seção de recomendações deste documento.

2.1.3 Etapa de Exploração

Nesta etapa, foram feitos dois ataques distintos para obter acesso à *Blueprint*. O primeiro é um ataque de força bruta explorando o serviço SSH e o segundo consiste em um ataque de *phishing* contendo uma carga maliciosa que cria uma *backdoor* no alvo. A seguir são apresentados mais detalhes sobre essas explorações.

1. Exploração via protocolo SSH (porta 22/tcp)

Com os recursos do **Nmap**, verificou-se que o serviço SSH está ativo na porta 22 e a versão do protocolo SSH é a SSH-2.0-OpenSSH_8.2p1. Desse modo, um ataque de força bruta foi simulado para tentar quebrar as credenciais de acesso remoto à *Blueprint*. Esse ataque foi baseado na utilização do módulo de ataque

um ataque *phishing* contendo uma carga maliciosa que permite abrir uma porta secreta (*backdoor*) de acesso à *Blueprint*. Para o desenvolvimento deste ataque, foram utilizadas as ferramentas **Brevo**, o **ngrok**, o **Metasploit** e habilidades de persuasão. A seguir, são descritos mais detalhes sobre este ataque.

O **Metasploit Framework** possui várias ferramentas de ataque e, utilizando o *Msfvenom*, criou-se um payload chamado 'Xapp_Update.sh', que gera uma *backdoor* no sistema alvo, por meio dessa carga 'linux/x86/meterpreter/reverse_tcp'. Esse *payload* explora uma conexão *reverse_tcp* que permite ao invasor assumir remotamente o sistema comprometido, tendo controle do sistema de arquivos e coletando informações sensíveis, como credenciais, utilizando outros módulos de ataque. Com o *payload* desenvolvido, o próximo passo do atacante foi colocá-lo disponível em um serviço web local, neste caso o *apache2*, e utilizar a ferramenta **ngrok** para criar um túnel seguro, atrás de *NAT* e *Firewall*, que expõem serviços locais para a internet.

Em seguida, o agente malicioso usou os serviços de e-mail do **Brevo** para criar e disparar os e-mails para os seus alvos. Na construção desses e-mails, o atacante utilizou as informações sobre o projeto e suas habilidades de persuasão para induzir a vítima a baixar o arquivo malicioso chamado 'Xapp_Update.sh', por meio do link disponibilizado, e executá-lo no ambiente da *Blueprint*.

Com o ataque sendo bem-sucedido, o agente malicioso pode identificar dados sensíveis e ativos críticos no ambiente da *Blueprint*, mas também tentar persistir o ataque e promover uma escalada de privilégios. A Figura 6 mostra o invasor acessando os recursos da *Blueprint*.

Desse modo, soluções são propostas na seção de recomendações para mitigar os riscos relacionados a esse ataque *phishing* e à vulnerabilidade do *Docker daemon*.

```
Active sessions
-----
Id  Name  Type  Information  Connection
--  ---  ---  ---  ---
1   meterpreter x86/linux openran-br @ 10.0.20.4 10.0.20.5:4444 → 10.0.20.4:57376 (10.0.20.4)
2   meterpreter x86/linux root @ 10.0.20.4 10.0.20.5:4444 → 10.0.20.4:53876 (10.0.20.4)

msf6 exploit(linux/local/docker_daemon_privilege_escalation) > sessions 2
[*] Starting interaction with 2...

meterpreter > ls
Listing: /home/openran-br
-----
Mode                Size  Type  Last modified  Name
-----
100600/rw----- 30005  fil   2024-06-25 23:41:07 -0300 .bash_history
100644/rw-r--r--  220   fil   2020-02-25 09:03:22 -0300 .bash_logout
100644/rw-r--r--  3883  fil   2023-12-12 15:55:57 -0300 .bashrc
040700/rwx----- 4096  dir   2024-06-12 08:51:57 -0300 .cache
040775/rwxrwxr-x  4096  dir   2024-04-19 09:15:11 -0300 .config
```

Figura 6 – Terminal do agente malicioso após a conclusão do *Phishing*.

2.2 Riscos

Nesta seção, serão mencionados os principais riscos cibernéticos encontrados na *Blueprint*.

2.2.1 Riscos do acesso remoto pelo protocolo SSH

Entre os principais riscos, destacam-se os ataques de força bruta, que consistem na tentativa de quebrar a senha do usuário por meio de ataques automatizados, e os ataques de dicionário, que usam dicionários de senhas para tentar acessar o sistema.

Com o acesso não autorizado concluído, o agente malicioso pode realizar escalada de privilégios no sistema da *Blueprint* por meio de técnicas de exploração como *PwnKit*, *Cron Jobs* e o *SUID*. Além disso, o invasor pode instalar *backdoors* e *malware* para acesso contínuo e exploração posterior.

Em explorações mais profundas do sistema alvo, o agente malicioso poderia obter dados importantes acerca do DNS e, assim, realizar ataques de envenenamento de cache de DNS ou sequestro de DNS. O que permite um total acesso à infraestrutura do Open-RAN, fazendo com que o invasor se torne um dos desenvolvedores no ambiente da *Blueprint*.

2.2.2 Riscos da engenharia social

A engenharia social pode resultar na perda irreparável de dados sensíveis, comprometendo a confidencialidade e a integridade das informações. Essas informações podem ser usadas de maneira prejudicial, como roubo de identidade ou chantagem. Assim, os principais riscos são:

- Ataques de *phishing*: mensagens digitais ou de voz que tentam manipular os destinatários para compartilharem informações confidenciais, baixarem software mal-intencionado, transferirem dinheiro ou ativos para as pessoas erradas ou adotarem alguma outra ação prejudicial;
- *Baiting*: ataque em que o agente malicioso induz as vítimas, consciente ou involuntariamente, a liberar informações confidenciais ou baixar código malicioso, tentando-as com uma oferta valiosa ou até mesmo um objeto valioso;
- *Pretexting*: situação em que o criminoso cria uma situação falsa para a vítima e se apresenta como a pessoa certa para resolvê-la. Muitas vezes (e o que é mais irônico), o golpista alega que a vítima foi afetada por uma violação de segurança e, em seguida, oferece-se para corrigir tudo se a vítima liberar informações importantes da conta, ou controlar o computador ou dispositivo da vítima. Tecnicamente falando, quase todos os ataques de engenharia social envolvem algum grau de *pretexting*.

Após a simulação de um ataque *Spear Phishing* a um dos desenvolvedores da *Blueprint*, que permitiu o acesso não autorizado à *Blueprint* por meio da criação de uma *backdoor*, foi possível a escalada de privilégios utilizando o módulo de exploração do **Metasploit** chamado `'exploit/linux/local/docker_daemon_privilage_escalation'`, que adquiriu acesso *root* ao sistema e, ao mesmo tempo, abriu uma nova conexão para persistir o ataque.

Este módulo obtém privilégios de *root* de qualquer conta de *host* com acesso ao *Docker daemon*, o que geralmente inclui contas no grupo `'docker'`. Isso significa que é

possível iniciar um contêiner em que o diretório é o diretório no seu anfitrião e o contêiner pode alterar o sistema de arquivos do *host* sem qualquer restrição. Logo, o invasor pode conseguir o acesso total ao sistema da *Blueprint* e, assim, ameaçar a infraestrutura do Open-RAN.

2.2.3 Riscos do *cluster Kubernetes* do Near-RT RIC

Os agentes maliciosos estão constantemente atacando os ambientes *Kubernetes*, sendo que a motivação varia do roubo de dados à mineração de criptomoedas e à negação de serviço (DoS), que pode atuar como um desvio para outras operações.

A Agência de Segurança Nacional (NSA) afirma que as três principais causas para um ambiente *Kubernetes* comprometido são ataques à cadeia de suprimentos, agentes mal-intencionados e ameaças internas.

Tendo como base as informações da NSA e do *Open Web Application Security Project* (OWASP) *Kubernetes* Top 10, seguem abaixo os dez principais riscos ao *cluster Kubernetes*:

- **K01:** Configurações de carga de trabalho inseguras;
- **K02:** Vulnerabilidades da cadeia de suprimentos;
- **K03:** Configurações RBAC excessivamente permissivas;
- **K04:** Falta de aplicação centralizada de políticas;
- **K05:** Registro e monitoramento inadequados;
- **K06:** Mecanismos de autenticação quebrados;
- **K07:** Controles de segmentação de rede ausentes;
- **K08:** Falhas no gerenciamento de segredos;
- **K09:** Componentes de *cluster* mal configurados;

- **K10:** Componentes *Kubernetes* desatualizados e vulneráveis.

Após aplicar a ferramenta **CIS Kubernetes Benchmark** para avaliar a segurança do *cluster*, ficou evidente que o *cluster* do Near-RT RIC apresenta seis dos dez principais riscos mencionados pelo OWASP Kubernetes: K01, K02, K05, K08, K09 e K10. Portanto, são necessárias correções para melhorar a segurança desse *cluster* e, assim, mitigar os riscos destacados anteriormente. A Figura 7 mostra que o *cluster* falhou nos testes de registro e monitoramento (K05).

```
[PASS] 1.2.18 Ensure that the --insecure-bind-address argument is not set (Automated)
[PASS] 1.2.19 Ensure that the --insecure-port argument is set to 0 (Automated)
[PASS] 1.2.20 Ensure that the --secure-port argument is not set to 0 (Automated)
[FAIL] 1.2.21 Ensure that the --profiling argument is set to false (Automated)
[FAIL] 1.2.22 Ensure that the --audit-log-path argument is set (Automated)
[FAIL] 1.2.23 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)
[FAIL] 1.2.24 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)
[FAIL] 1.2.25 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)
[WARN] 1.2.26 Ensure that the --request-timeout argument is set as appropriate (Automated)
```

Figura 7 – Amostra do teste de verificação da ferramenta **CIS Kubernetes Benchmark**.

2.3 Recomendações

2.3.1 Protocolo SSH

Para melhorar a segurança do protocolo de acesso remoto SSH contra ataques de força bruta ou dicionário, um conjunto de alterações são necessárias no arquivo ‘/etc/ssh/sshd_config’ presente no ambiente da *Blueprint*. A seguir são citadas essas alterações.

- **"PermitRootLogin no"**: Desabilita o acesso remoto do usuário *root*. Isso com certeza deve ser corrigido;
- **"PermitEmptyPassword no"**: Desabilita o login de usuários sem senha;

- **"LoginGraceTime 1m"**: Especifica o tempo permitido para autenticação bem-sucedida no servidor SSH. Quanto mais longo for o período de carência, mais conexões não autenticadas abertas poderão existir;
- **"MaxAuthTries 3"**: Define o número máximo de tentativas sucessivas de autenticação;
- **"MaxSessions 2"**: Define o número máximo de sessões simultâneas que o servidor SSH deve aceitar;
- **"ClientAliveInterval 200"**: Número de segundos que o servidor aguardará antes de enviar um pacote nulo para o cliente (para manter a conexão ativa);
- **"ClientAliveCountMax 1 "**: Define o número de mensagens vivas do cliente que podem ser enviadas sem que o servidor SSH receba quaisquer mensagens de volta do cliente;
- **"Port 2345"**: Alterar a porta do serviço é uma camada a mais de segurança para o servidor.

Após a realização dessas alterações, é necessário reiniciar o servidor SSH para que as modificações sejam implementadas. Além disso, a implementação de uma boa política de gerenciamento de credenciais é fundamental para conscientizar todos os usuários sobre a importância da política de senhas e de segui-la. Para isso, devem ser oferecidos treinamentos sobre segurança, além de promover reuniões para abordar o assunto, tirar dúvidas e indicar as melhores práticas.

Levando em consideração as melhores práticas, a seguir são listadas as principais:

- **Criação de senhas fortes**: As senhas devem ser fortes e sem dados importantes sobre si para dificultar o trabalho dos hackers. Estabelecer um número mínimo de dígitos, como de 8 caracteres, por exemplo, pode ajudar. A utilização de palavras

únicas ou até mesmo frases curtas não garantem a segurança da senha. É necessário variá-las o máximo possível, misturando letras, números e outros caracteres especiais. Se possível, a senha não deve formar nenhuma palavra que tenha sentido e não repetir nenhum símbolo;

- **Não utilizar a mesma senha em todas as contas:** Outro grande erro cometido por muitos usuários em busca de agilidade é utilizar a mesma senha para todos os seus logins, ou apenas trocar algumas letras ou números no final. Nesses casos, quando uma invasão acontece, a empresa pode sofrer com um efeito dominó, em que diversos acessos daquele usuário serão invadidos;
- **Troca periódica das senhas:** É preciso aplicar uma regra para todos os usuários, solicitando que a senha seja alterada a cada período estabelecido. Desse modo, os usuários adquirem o hábito de atualizar suas chaves de acesso, limpando um possível rastro que poderia permitir a entrada de desconhecidos;
- **Utilizar softwares para gerenciar senhas:** A melhor maneira de garantir que os colaboradores estejam utilizando senhas fortes é fazendo com que utilizem um gerenciador de senhas. Além disso, gerenciadores de senhas podem enviar notificações aos administradores de TI se as credenciais de login de um colaborador aparecerem em uma violação de dados para que possam solicitar que ele altere sua senha.

2.3.2 Engenharia Social

No cenário cibernético, os ataques de *phishing* são um dos tipos mais comuns que ocorrem atualmente. Isso significa que os colaboradores devem ser treinados regularmente sobre como detectá-los e como evitar cair em tentativas de *phishing*. Uma das melhores maneiras de treinar seus colaboradores sobre isso é enviando e-mails de *phishing* simulados, que permitem que as organizações vejam o quão bem treinadas estão para detectar tentativas de *phishing*.

Em relação à simulação desse tipo de ataque e outras técnicas de engenharia social, tem-se a seguir as principais recomendações ao receber um e-mail:

- Identifique a autenticidade dos remetentes e do domínio do e-mail;
- Em caso de dúvidas, não se arrisque;
- Fique atento a:
 - Comunicações com erros gramaticais e ortográficos;
 - Solicitações urgentes ou ameaças;
 - Pedidos incomuns para clicar em links e anexos;
 - Sites oficiais com endereços incorretos;
 - E-mails e comunicados não personalizados ou que fogem da identidade visual da marca.

O e-mail não é a única ferramenta que necessita de atenção. As mensagens de aplicativos de mensagens e em redes sociais têm o mesmo potencial de perigo e é possível encontrar links maliciosos, por exemplo, em postagens de amigos no Facebook ou em comentários postados por embaixadores de marcas falsas no Twitter ou no Discord.

Já a vulnerabilidade do *Docker daemon*, que permite uma escalada de privilégio dentro do ambiente da *Blueprint*, pode ser combatida de duas formas. A primeira solução é mais difícil de ser implementada e consiste em executar o *daemon Docker* como um usuário não *root* (modo *Rootless*), porém, isso deve ser implementado em dia zero, caso possível, ou seja, antes de todas os demais processos desenvolvidos no *Kubernetes*, pois todo o trabalho realizado em *cluster* já ativo é perdido e pode ocorrer a quebra do funcionamento dos xApps. O processo para modo *Rootless* encontra-se na documentação do *docke* neste link a seguir: <https://docs.docker.com/engine/security/rootless/>.

Outra solução para evitar o escalonamento de privilégios a partir de um contêiner é a necessidade de configurar os aplicativos do contêiner para serem executados

por usuários sem privilégios. O desenvolvedor pode remapear esse usuário para ser um usuário menos privilegiado no *host* em que o *Docker* está em execução. Ao usuário mapeado é atribuído um intervalo de UIDs que funcionam dentro do *namespace* como normal (0 a 65.536), mas não têm privilégios de *root*.

2.3.3 Cluster kubernetes do Near-RT RIC

Conforme mencionado na seção sobre riscos, o *cluster Kubernetes* do Near-RT RIC apresenta questões relacionadas aos riscos K01, K02, K05, K08, K09 e K10. Para mitigar esses riscos, são necessárias medidas específicas, seguindo as documentações oficiais do *Kubernetes*, do OWASP *Kubernetes* e do Benchmark CIS para *Kubernetes*. Essas diretrizes são essenciais para garantir a segurança e a confiabilidade do ambiente *Kubernetes* e proteger os *xApps* contra possíveis ameaças mencionadas no documento *O-RAN Study on Security for Near Real Time RIC and xApps*, fornecido pela O-RAN Alliance. A seguir, são listadas as principais recomendações para cada um desses riscos:

- **K01:** Configurações de carga de trabalho inseguras

RE01: É uma boa prática verificar regularmente as bibliotecas de terceiros de sua aplicação em busca de vulnerabilidades de segurança;

RE02: Ao construir contêineres, executá-los utilizando usuários que tenham o menor nível de privilégio possível no sistema operacional, apenas para cumprir o objetivo do contêiner.

- Definir 'runAsNonRoot True' para executá-lo como usuário não *root*;
- Definir 'AllowPrivilegeEscalation False' para não permitir que o processo filho obtenha mais privilégios do que seus pais;
- Definir um 'LimitRange' para restringir as alocações de recursos para cada tipo de objeto aplicável em um *namespace*.

- **K02:** Vulnerabilidades da cadeia de suprimentos

RE01: As imagens de contêineres podem ser pensadas como uma série de softwares, artefatos e metadados passados de um produtor para um consumidor, assim, deve ser usada criptografia assimétrica para assinar e verificar o artefato em cada etapa da cadeia de suprimentos para detectar adulteração dos próprios artefatos. O projeto Cosign é um projeto de código aberto com o objetivo de verificar as imagens de contêineres por meio de cada fase da cadeia de suprimentos. Isso também aumenta o nível *Supply-chain Levels for Software Artifacts* (SLSA) do pipeline de compilação, com um nível SLSA mais alto indicando um pipeline de construção mais resiliente;

RE02: As imagens de contêineres devem ser criadas usando os recursos e dependências mínimas para reduzir a superfície de ataque se a carga de trabalho for comprometida. Também é importante garantir que suas imagens base estejam atualizadas com os *patches* de segurança mais recentes. Pensando nisso, ferramentas como o *Docker Slim* estão disponíveis para otimizar as imagens por motivos de desempenho e segurança;

RE03: A varredura de vulnerabilidades em uma imagem tem como objetivo enumerar problemas de segurança conhecidos em imagens de contêiner e deve ser usado como uma primeira linha de defesa. Logo, ferramentas de código aberto como *Clair* e *trivy* podem analisar estaticamente as imagens de contêineres para vulnerabilidades conhecidas, como CVEs, e devem ser usadas no início do ciclo de desenvolvimento o mais razoavelmente possível.

- **K05:** Registro e monitoramento inadequados

RE01: Habilitar os *logs* de auditoria do *Kubernetes* que registram as ações executadas pela API para análise posterior. Os *logs* de auditoria ajudam a responder a perguntas relacionadas a eventos que ocorrem na API servidor em si;

RE02 Certificar-se de que os *logs* estejam monitorando chamadas de API anômalas ou indesejadas, especialmente qualquer falha de autorização (essas entradas

de *log* terão uma mensagem de status "Proibido"). Falhas de autorização podem significar que um invasor está tentando abusar de credenciais roubadas;

RE03: Os eventos do *Kubernetes* podem indicar qualquer recurso do *Kubernetes*, alterações e erros de estado, como cota de recursos excedida ou pods pendentes, bem como quaisquer mensagens informativas;

RE04: Aplicativos executados dentro do *Kubernetes* geram *logs* úteis de uma perspectiva de segurança. O método mais fácil para a captura desses *logs* é garantir que a saída seja gravada na saída padrão e nos fluxos de erro padrão.

- **K08:** Falhas no gerenciamento de segredos

RE01: o componente *etcd* do *cluster* é um banco de dados do tipo chave/valor altamente disponível responsável por centralizar dados de *cluster*, e abriga informações cruciais de configuração, como o *Kubernetes Secrets*. Logo, o acesso ao *etcd* deve ser limitado apenas ao *control plane*;

RE02: O *Kubernetes* oferece suporte à criptografia em repouso, a utilização desse suporte promove a criptografia dos recursos *Secret* no *etcd*, impedindo que as partes que obtiverem acesso aos seus backups do *etcd* visualizem o conteúdo desses segredos;

RE03: Certifique-se de habilitar e configurar os registros de auditoria do *Kubernetes* e centralizar seu armazenamento;

RE04: Sempre audite a configuração do RBAC de *plugins* e softwares de terceiros instalados no *cluster* para garantir que o acesso aos segredos do *Kubernetes* não seja concedido desnecessariamente.

- **K09:** Componentes de *cluster* mal configurados

RE01: Um bom começo é executar varreduras e auditorias regulares do **CIS Benchmark** focadas em configurações incorretas de componentes;

RE02: Uma cultura forte de Infraestrutura como Código (IaC) também pode ajudar a centralizar a configuração e a correção do *Kubernetes*, dando às equipes

de segurança visibilidade sobre como os *clusters* são criados e mantidos e reduz significativamente a probabilidade de erros de configuração;

RE03: Deve ser desabilitado o acesso anônimo para solicitações não autenticadas. Para isso, basta iniciar o *kubelet* com o sinalizador “`-anonymous-auth=false`”;

RE04: Os *Pods* do *cluster* devem seguir os padrões de segurança estabelecidos pela documentação oficial do *Kubernetes*, que define os padrões *pod* privilegiado, *pod* básico e *pod* restrito, que restringem a atuação dos *Pods* dentro do *cluster* e podem ser impostos no nível do *namespace*;

RE05: Limites de memória e CPU devem ser definidos para restringir os recursos de memória e CPU que um *pod* pode consumir em um nó e, portanto, evitar potenciais ataques DoS de cargas de trabalho maliciosas ou violadas e prevenir contra o mal funcionamento de um xApp pela falta de recursos computacionais.

- **K10:** Componentes *Kubernetes* desatualizados e vulneráveis:

RE01: em primeiro lugar, o *Kubernetes* e os componentes associados não podem ser deixados de fora do seu processo de verificação de vulnerabilidades CVE existente;

RE02: Ferramentas como **OPA Gatekeeper** podem ser usadas para escrever regras personalizadas que descubram componentes vulneráveis em um *cluster*. Elas devem ser executadas em uma cadência regular e rastreadas pela equipe de operações de segurança;

RE03: todos os softwares de terceiros devem ser auditados de forma independente antes da implantação para verificar se há RBAC excessivamente permissivo, acesso de *kernel* de baixo nível e registros de divulgação de vulnerabilidades históricas.

2.4 Versões seguras da *Blueprint*

Considerando todas as recomendações mencionadas na seção anterior, a maioria dessas soluções foi implementada para fortalecer a segurança do ambiente virtual da *Blueprint*. No entanto, algumas soluções não foram inseridas, pois certas medidas afetam o desempenho do ambiente para desenvolver e testar xApps no Near-RT RIC.

Em relação à proteção do protocolo de acesso remoto SSH, todas as medidas de proteção contra ataques de força bruta foram implementadas com sucesso. Quanto à adesão a uma boa política de gerenciamento de credenciais, apenas a recomendação de criação de uma senha forte foi seguida. Os outros pontos levantados não puderam ser implementados, pois o projeto encontra-se em uma etapa de desenvolvimento e sugestões como a troca periódica das senhas não são viáveis.

Um segundo ponto a ser mencionado é a vulnerabilidade do *Docker daemon*, que permite uma escalada de privilégios dentro do ambiente da *Blueprint*. Essa vulnerabilidade pode ser minimizada implementando contêineres com usuários sem privilégios de *root*. Outra solução, mais eficiente, consiste em executar o *Docker daemon* como um usuário não *root* (modo *Rootless*). No entanto, não foi possível utilizá-la, pois todo o trabalho realizado em um *cluster* já ativo seria perdido e isso poderia causar a quebra do funcionamento dos xApps. Dessa forma, as versões seguras das *Blueprints* continuam convivendo com o risco de ter o *Docker daemon* funcionando com privilégios de *root*.

Quanto às soluções relacionadas aos riscos K01, K02, K05, K08, K09 e K10, a maioria delas foi inserida no *cluster Kubernetes* das versões seguras da *Blueprint*. Assim, ocorreu uma redução no número de pontos que falharam nos testes de verificação da ferramenta **CIS Kubernetes Benchmark**, como mostrado nas Figuras 8 e 9.

Entretanto, não foi viável implementar todas as soluções propostas. A seguir, são apresentadas as recomendações do OWASP e do Benchmark CIS para *Kubernetes* que não podem ser efetivamente aplicadas.

Em primeiro lugar, a recomendação RE01 do risco K02 ainda não foi im-

```

== Summary policies ==
0 checks PASS
0 checks FAIL
24 checks WARN
0 checks INFO

== Summary total ==
82 checks PASS
3 checks FAIL
37 checks WARN
0 checks INFO

```

Figura 8 – Resultado quantitativo do teste de verificação da ferramenta **CIS Kubernetes Benchmark**, após a implementação das recomendações.

```

[PASS] 1.2.19 Ensure that the --insecure-port argument is set to 0 (Automated)
[PASS] 1.2.20 Ensure that the --secure-port argument is not set to 0 (Automated)
[PASS] 1.2.21 Ensure that the --profiling argument is set to false (Automated)
[PASS] 1.2.22 Ensure that the --audit-log-path argument is set (Automated)
[PASS] 1.2.23 Ensure that the --audit-log-maxage argument is set to 30 or as appropriate (Automated)
[PASS] 1.2.24 Ensure that the --audit-log-maxbackup argument is set to 10 or as appropriate (Automated)
[PASS] 1.2.25 Ensure that the --audit-log-maxsize argument is set to 100 or as appropriate (Automated)
[WARN] 1.2.26 Ensure that the --request-timeout argument is set as appropriate (Automated)
[PASS] 1.2.27 Ensure that the --service-account-lookup argument is set to true (Automated)
[PASS] 1.2.28 Ensure that the --service-account-key-file argument is set as appropriate (Automated)
[PASS] 1.2.29 Ensure that the --etcd-certfile and --etcd-keyfile arguments are set as appropriate (Automated)
[PASS] 1.2.30 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Automated)
[PASS] 1.2.31 Ensure that the --client-ca-file argument is set as appropriate (Automated)
[PASS] 1.2.32 Ensure that the --etcd-cafile argument is set as appropriate (Automated)
[PASS] 1.2.33 Ensure that the --encryption-provider-config argument is set as appropriate (Manual)
[WARN] 1.2.34 Ensure that encryption providers are appropriately configured (Manual)
[PASS] 1.2.35 Ensure that the API Server only makes use of Strong Cryptographic Ciphers (Manual)

```

Figura 9 – Pontos que foram corrigidos e passam no teste de segurança.

plementada, pois está em fase de desenvolvimento. Quanto às recomendações RE04 e RE05 do risco K09, elas não foram inseridas devido a limitações na ferramenta `dms-cli` do `framework` OSC Python xApp. Essa ferramenta ainda não consegue trabalhar com os padrões de segurança estabelecidos pela documentação oficial do *Kubernetes* nem restringir os recursos de memória e CPU que um xApp pode consumir.

Portanto, considerando versões futuras da *Blueprint*, essas recomendações podem ser implementadas para ampliar o escopo de segurança desse ambiente. As imagens das versões seguras da *Blueprint* só podem ser acessadas por colaboradores autorizados do projeto “OpenRAN@Brasil” neste repositório: [GitHub - LAB-LAPSI-UFCG/openran-br-blueprint-security](https://github.com/LAB-LAPSI-UFCG/openran-br-blueprint-security).

3 Relatório de avaliação de risco da solução finalizada

O projeto ainda está em andamento, e portanto não foi possível avaliar o risco da solução finalizada. Entretanto, como pode ser visto na seção anterior, foram utilizadas ferramentas de análise da Blueprint que podem ser aplicadas à avaliação de risco da solução final.

Outro ponto a se destacar é que os itens de segurança do OpenRAN ainda estão em desenvolvimento, como pode ser observado pela atualização constante dos documentos do Grupo 11 (Segurança) da OpenRAN Alliance. Tem sido feito um acompanhamento desses requisitos e estabelecidas metodologias para avaliar os riscos quando a solução final estiver disponível.

4 Descrição de requisitos e ferramentas para a melhoria de segurança cibernética da O-RU

No estado atual das análises de especificações da O-RAN Alliance, os documentos *Study on O-RU Centralized User Management* e *Study on Security for Shared O-RU (SharedORU)*, foram desenvolvidos pelo *Security Work Group (WG11)*, com foco em requisitos e ferramentas para a segurança cibernética da O-RU.

Atualmente na versão 2.0, datada de junho de 2024, o documento *Study on O-RU Centralized User Management* tem como objetivo especificar o resultado do estudo sobre o gerenciamento centralizado de usuários O-RU realizado pelo *O-RAN Security Focus Group* e, em sua versão mais recente, contém 10 questões-chave e 3 propostas de soluções para abordar ou reduzir o risco associado a elas. Um mapeamento das soluções propostas para as questões-chave é mostrado na Tabela 1.

Atualmente na versão 5.0, datada de junho de 2024, o documento *Study on Security for Shared O-RU (SharedORU)* tem como objetivo fornecer o modelo de ameaça e a avaliação de risco para a O-RU compartilhada, identificando ameaças e riscos e recomendando potenciais controles de segurança para proteger contra essas ameaças por meio de proteção ou mitigação.

Este documento utiliza o modelo **STRIDE** para classificar ameaças. Em sua versão mais recente, são identificadas 57 ameaças à O-RU compartilhada, classificadas

Questão-chave	Sol. #1	Sol. #2	Sol. #3
QC #1: conta padrão em O-RU	X		
QC #2: conta padrão obtém função <i>sudo</i>	X		
QC #3: controlador O-RU deve acompanhar os usuários definidos localmente em cada O-RU gerenciado	X		
QC #4: como obter a configuração inicial do Servidor Central?		X	
QC #5: como a O-RU se conecta com segurança ao Servidor Central?	X		
QC #6: disponibilidade de servidor centralizado de gerenciamento de usuários para as O-RUs			X
QC #7: categorização de usuários que precisam ser gerenciados central e localmente			
QC #8: obtenção de dados iniciais de configuração de segurança a partir de um servidor de software de configuração		X	
QC #9: transporte seguro para o servidor de autenticação centralizado	X		
QC #10: armazenamento seguro de credenciais O-RU			

Tabela 1 – Mapeamento das soluções propostas para as questões-chave identificadas no documento *Study on O-RU Centralized User Management* da O-RAN Alliance

nos seguintes 7 grupos de ameaças: movimento lateral entre funções de rede, ameaças ao acesso do usuário, ameaças ao acesso a dados, ameaças à disponibilidade, ameaças de configuração, ameaças de controlador de host neuro e resiliência.

5 Conclusão

O desenvolvimento de aplicações Open-RAN é uma oportunidade estratégica para a comunidade brasileira, pois permite que o país participe de maneira efetiva dos esforços mundiais para evolução das redes de acesso sem fio. A absorção desse tipo de conhecimento permite que a indústria nacional de software crie soluções que podem ser adotadas globalmente, gerando negócios, empregos e renda a longo prazo. Além disso, contribui para a redução da dependência tecnológica na área estratégica crítica de telecomunicações.

Neste relatório, foram apresentadas as ações que vem sendo realizadas no contexto da Atividade 3.5, as quais contribuem para a identificação de ameaças e proposta de soluções no contexto de O-RAN e desenvolvimento de xApps e rApps seguros. Como essa atividade ainda está em execução, suas contribuições ainda passarão por evolução até a conclusão.

6 Histórico de versões deste documento

<u>Data de Emissão</u>	<u>Versão</u>	<u>Descrição das Alterações Realizadas</u>
DD/07/2024	1.0	Construção inicial do relatório
DD/MM/2024	1.?	Revisão de todo documento

7 Execução e aprovação

Executado por:

Edmar Candeia Gurjão

Amanda Barbosa Silva

José Edson Carneiro Agra Junior

Revisado por:

Aprovado por:

Data da emissão: 10/07/2024