



UNIVERSIDADE FEDERAL DE SANTA CATARINA
CAMPUS DE FLORIANÓPOLIS
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
BACHARELADO EM CIÊNCIAS DA COMPUTAÇÃO

RIAN BLOCK SERENA

AUTENTICAÇÃO UTILIZANDO SERVIDORES RADIUS E O PROTOCOLO EAP

FLORIANÓPOLIS
2025

Rian Block Serena

AUTENTICAÇÃO UTILIZANDO SERVIDORES RADIUS E O PROTOCOLO EAP

Proposta de Trabalho de Conclusão de Curso de Graduação em Ciência da Computação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientadora: Profa. Dra. Carla Merkle Westphall
Coorientador: Me. Eduardo Freitas Hoffmann

FLORIANÓPOLIS
2025

Serena, Rian Block

AUTENTICAÇÃO UTILIZANDO SERVIDORES RADIUS E O
PROTOCOLO EAP / Rian Block Serena ; orientadora, Carla Merkle
Westphall, coorientador, Eduardo Freitas Hoffmann, 2025.

68 p.

Trabalho de Conclusão de Curso (graduação) - Universidade
Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências
da Computação, Florianópolis, 2025.

Inclui referências.

1. Ciências da Computação. 2. RADIUS, eduroam, Protocolo
EAP, Vulnerabilidades.. I. Westphall, Carla Merkle. II. Hoffmann,
Eduardo Freitas. III. Universidade Federal de Santa Catarina. Graduação
em Ciências da Computação. IV. Título.

Rian Block Serena

**Título: AUTENTICAÇÃO UTILIZANDO SERVIDORES RADIUS E O PROTOCOLO
EAP**

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do título de Bacharel e aprovado em sua forma final pelo Curso Ciências da Computação.

Local Florianópolis, 24 de Novembro de 2025.

Coordenação do Curso

Banca examinadora

Prof.(a) Carla Merkle Westphall, Dr.(a)
Orientadora

Eduardo Freitas Hoffmann, Me.
Instituição UFSC
Coorientador

Caciano dos Santos Machado, Dr.(a)
Instituição UFSC

Florianópolis, 2025

AGRADECIMENTOS

Quero dedicar este espaço às pessoas que estiveram comigo ao longo de toda esta caminhada e que, de diferentes formas, tornaram este trabalho possível. Depois de tanto escrever em linguagem acadêmica, é no mínimo estranho escrever algo pessoal neste trabalho.

Agradeço, antes de tudo, à minha prima Gabi. Foi ela quem me incentivou a tentar a UFSC, quem acreditou em mim quando eu ainda duvidava, e quem esteve presente em cada fase, revisando meu texto tarde da noite e sanando as minhas várias dúvidas sobre a estrutura de um TCC. Sem essa ajuda e sem essa paciência toda, eu não estaria aqui finalizando este TCC.

Agradeço também ao meu companheiro de vida e noivo, Thiago, que ouviu todas as minhas reclamações, crises e frustrações durante este trabalho.

Sou grato aos meus pais pelas oportunidades que me deram e pelos ensinamentos que levo comigo. Eles formaram muito do que sou hoje e, por consequência, muito do que está escrito nestas páginas.

E por fim, agradeço aos meus amigos, que contribuíram de formas grandes e pequenas, das mais variadas formas possíveis, enviando apoio nos momentos certos ou simplesmente entendendo minhas ausências enquanto eu finalizava este projeto.

A todos vocês, meu sincero agradecimento.

RESUMO

O intuito deste projeto é analisar a segurança da rede *Education Roaming* (eduroam), explorando as vulnerabilidades encontradas em cada uma das tecnologias que incorporam esse tipo de rede, como os protocolos *Extensible Authentication Protocol* (EAP) e a infraestrutura RADIUS que são amplamente utilizados para a autenticação dos usuários. O eduroam é um serviço mundial de acesso Wi-Fi federado adotado por diversas instituições de ensino e pesquisa. Os usuários se autenticam utilizando suas credenciais institucionais, entretanto, configurações incorretas e a falta de uma validação adequada nos certificados digitais pela parte dos clientes podem expor graves vulnerabilidades. Em estudos recentes foram demonstradas as possibilidades de ataques do tipo *evil twin*, que é um ponto de acesso malicioso que imita a rede legítima. Esse tipo de ataque é capaz de roubar credenciais de login de usuários além de outros possíveis dados sensíveis. Com isso, observa-se a necessidade de investigar essas e outras vulnerabilidades que podem ser encontradas na autenticação da rede eduroam, focando em possíveis falhas de configuração e implementação tanto no protocolo EAP quanto nos servidores *Remote Authentication Dial-In User Service* (RADIUS). A partir deste trabalho, que busca estudar esses protocolos de forma independente entendendo as vulnerabilidades de cada uma dessas tecnologias, o intuito é realizar simulações de ataques do tipo *evil twin* e *man-in-the-middle* em ambiente de teste controlado, como a configuração de um *rogue server*. Bem como, observar como essas vulnerabilidades são gerenciadas em diferentes SOs, como IOS Android e Windows. Além da análise de diferentes SOs também busca-se a análise das vulnerabilidades individuais de cada uma dessas tecnologias e se continuam presentes quando elas são combinadas. Espera-se com isso, mapear e documentar as fragilidades de segurança no eduroam, propondo formas de tornar a rede ainda mais robusta, com recomendações de configurações que possam mitigar os riscos.

Palavras-chave: RADIUS; eduroam; Protocolo EAP; Vulnerabilidades.

ABSTRACT

The objective of this project is to analyze the security of the Education Roaming (eduroam) network by exploring the vulnerabilities found in the technologies that comprise it, such as the Extensible Authentication Protocol (EAP) and the RADIUS infrastructure, both widely used for user authentication. Eduroam is a global federated Wi-Fi access service adopted by numerous education and research institutions. Users authenticate using their institutional credentials; however, misconfigurations and the lack of proper digital certificate validation on the client side can expose severe vulnerabilities. Recent studies have demonstrated the feasibility of Evil Twin attacks, in which a malicious access point impersonates the legitimate network. This type of attack is capable of stealing user login credentials as well as other sensitive data. Therefore, there is a need to investigate these and other vulnerabilities within eduroam authentication, focusing on potential configuration and implementation flaws in both the EAP protocol and Remote Authentication Dial-In User Service (RADIUS) servers. Through this work, which seeks to study these protocols independently to understand their individual vulnerabilities, the goal is to perform simulations of Evil Twin and Man-in-the-Middle attacks in a controlled test environment, such as setting up a rogue server. Furthermore, the project aims to observe how these vulnerabilities are handled across different operating systems, such as iOS, Android, and Windows. In addition to analyzing different OSs, this study also investigates whether the individual vulnerabilities of each technology persist when combined. Ultimately, it is expected to map and document security weaknesses in eduroam, proposing ways to make the network more robust through configuration recommendations that can mitigate these risks.

Keywords: RADIUS; eduroam; EAP protocol; vulnerabilities.

LISTA DE FIGURAS

- Figura 1: Encapsulamentos do protocolo EAP em uma rede do padrão IEEE 802.1X.
- Figura 2: Exemplo de hierarquia entre servidores RADIUS da rede eduroam.
- Figura 3: Processo completo de autenticação com PEAP-MSCHAPv2.
- Figura 4: Representação de um ataque *man-in-the-middle*.
- Figura 5: Representação de um ataque *evil twin*.
- Figura 6: Representação do ambiente utilizado, com os ataques evil twin e man in the middle em conjunto.
- Figura 7: Alerta do Windows sobre falha na validação do certificado MD5.
- Figura 8: Tela de configuração e conexão do Android MD5.
- Figura 9: Tela de configuração e conexão no IOS MD5.
- Figura 10: Notificação do iPhone sobre certificado não confiável MD5.
- Figura 11: Captura do tráfego no Wireshark durante o procedimento de conexão MD5.
- Figura 12: Captura de logs do FreeRADIUS após a conexão com o *supplicant* MD5.
- Figura 13: Notificação do iPhone sobre certificado não confiável TTLS.
- Figura 14: Notificação do Windows sobre falha na validação do certificado TTLS.
- Figura 15: Tela de configuração da conexão no Android TTLS.
- Figura 16: Captura do tráfego no Wireshark durante o procedimento de conexão TTLS.
- Figura 17: Captura de logs e credenciais no FreeRADIUS após a conexão com o *supplicant* TTLS.
- Figura 18: Notificação do Windows sobre falha na validação do certificado PEAP.
- Figura 19: Captura do tráfego EAP-PEAP através do Wireshark.
- Figura 20: Captura do tráfego EAP-PEAP através do Wireshark.
- Figura 21: Tela de conexão do iOS.
- Figura 22: Notificação do iPhone sobre certificado não confiável PEAP.
- Figura 23: Tela de conexão e configuração no Android selecionando PEAP e “não verificar certificado”.
- Figura 24: Tela de conexão e configuração no Android selecionando PEAP e “não verificar certificado”.
- Figura 25: Captura de logs no FreeRADIUS após a conexão com o *supplicant* PEAP.
- Figura 26: Notificação do Windows sobre falha na validação do certificado GTC.
- Figura 27: Notificação do iPhone sobre certificado não confiável GTC.
- Figura 28: Captura do tráfego com inner GTC através do Wireshark.
- Figura 29: Captura de logs no FreeRADIUS após a conexão com o *supplicant* GTC.
- Figura 30: Captura da tela de conexão e configuração do Android durante ataque de *downgrade*.
- Figura 31: Captura de tela do Android tentando forçar a utilização de *inner* GTC.
- Figura 32: Captura do Wireshark durante tentativas de *downgrade* do *inner method*.
- Figura 33: Conteúdo do Pacote RADIUS observado através do Wireshark.
- Figura 34: Captura dos logs do RADIUS durante o ataque ao Android.
- Figura 35: Mensagem de erro ao conectar-se à rede eduroam em iPhone.
- Figura 36: Captura dos pacotes RADIUS durante a conexão.
- Figura 37: Notificação do iPhone sobre certificado não confiável.
- Figura 38: Captura de logs RADIUS durante o ataque no iPhone.
- Figura 39: Exibição de falha ao se conectar à rede eduroam em *tablet* e solicitação de reinserção de dados.
- Figura 40: Captura dos logs do servidor RADIUS durante a conexão com o *tablet*.
- Figura 41: Mensagem de erro no macOS quando o servidor tentou forçar métodos inseguros.
- Figura 42: Aviso sobre a impossibilidade de verificar o certificado no macOS.

Figura 43: Captura do Wireshark durante o ataque ao macOS.

Figura 44: Captura dos logs do FreeRADIUS durante a conexão com o macOS.

Figura 45: Mensagem de erro de conexão no ataque ao Windows.

Figura 46: Mensagem de certificado inválido no Windows.

Figura 47: Captura dos pacotes RADIUS através do Wireshark durante ataque ao Windows.

Figura 48: Captura dos logs RADIUS durante ataque ao Windows.

Figura 49: Tabela comparativa entre os métodos testados e o seu comportamento perante as vulnerabilidades.

Figura 50: Tabela de dispositivos que aceitaram a conexão automática ao AP falso.

LISTA DE ABREVIATURAS E SIGLAS

802.1X – Port Based Network Access Control (controle de acesso baseado em portas).
EAP – Extensible Authentication Protocol.
EAPoL – EAP over LAN.
RADIUS – Remote Authentication Dial-In User Service.
AAA – Authentication, Authorization and Accounting.
TLS – Transport Layer Security.
TTLS – Tunneled Transport Layer Security.
PEAP – Protected Extensible Authentication Protocol.
PAC – Protected Access Credential.
GTC – Generic Token Card.
SIM – Subscriber Identity Module.
AKA – Authentication and Key Agreement.
LDAP – Lightweight Directory Access Protocol.
SQL – Structured Query Language.
UDP – User Datagram Protocol.
TCP – Transmission Control Protocol.
SCTP – Stream Control Transmission Protocol.
DNS – Domain Name System.
NAPTR – Naming Authority Pointer.
SRV – Service Record.
VPN – Virtual Private Network.
SSID – Service Set Identifier.
MAC – Media Access Control.
NTLM – NT LAN Manager.
EAP-MD5 – EAP Message Digest 5.
EAP-TLS – EAP Transport Layer Security.
EAP-TTLS – EAP Tunneled TLS.
EAP-PEAP – Protected EAP.
EAP-MSCHAPv2 – EAP Microsoft Challenge-Handshake Authentication Protocol v2.
EAP-FAST – Flexible Authentication via Secure Tunneling.
CHAP – Challenge-Handshake Authentication Protocol.
PAP – Password Authentication Protocol.
NAS – Network Access Server.
IdP – Identity Provider.
SP – Service Provider.
NAI – Network Access Identifier.
AP – Access Point.
LAN – Local Area Network.
IEEE – Institute of Electrical and Electronics Engineers.
RFC – Request for Comments.
WLAN – Wireless Local Area Network.
VLAN – Virtual Local Area Network.
WPA2 – Wi-Fi Protected Access 2.
WPA3 – Wi-Fi Protected Access 3.
UFSC – Universidade Federal de Santa Catarina.
ACM – Association for Computing Machinery.
API – Application Programming Interface.
CAT – Configuration Assistant Tool (eduroam CAT).

CA – Certification Authority.
CN – Common Name.
VM – Virtual Machine.
UX – User Experience.
TI – Tecnologia da Informação.

SUMÁRIO

1. INTRODUÇÃO.....	13
1.1 OBJETIVOS.....	13
1.1.1 Objetivo Geral.....	13
1.1.2 Objetivos Específicos.....	14
2. FUNDAMENTAÇÃO TEÓRICA.....	15
2.1. IEEE 802.1X CONTROLE DE ACESSO BASEADO EM PORTAS.....	15
2.2. EXTENSIBLE AUTHENTICATION PROTOCOL (EAP).....	16
2.3. RADIUS-REMOTE AUTHENTICATION DIAL-IN USER SERVICE.....	19
2.4 PROCESSO DE AUTENTICAÇÃO NO EDUROAM.....	23
2.4.1. Fragilidades conhecidas.....	26
2.4.2. Eduroam Configuration Assistant Tool (CAT).....	28
2.5. VULNERABILIDADES EXPLORADAS.....	29
3. DESENVOLVIMENTO.....	33
3.1 FERRAMENTAS UTILIZADAS.....	33
3.2 ILUSTRAÇÃO DO AMBIENTE DE EXPERIMENTAÇÃO.....	33
3.3. CONFIGURAÇÃO DO AMBIENTE.....	34
3.4. PROCEDIMENTOS.....	35
3.5. EVIDÊNCIAS COLETADAS.....	36
3.5.1 EAP MD5.....	36
3.5.2 EAP-TLS.....	39
3.5.3 EAP-TTLS.....	40
3.5.4 EAP-PEAP.....	41
3.5.5 EAP-GTC.....	44
3.5.6 EAP-FAST.....	45
3.5.7 Simulação de evil twin e man-in-the-middle.....	45
3.5.7.1 Celular Android.....	46
3.5.7.2 iPhone.....	48
3.5.7.3 Tablet Android.....	50
3.5.7.4 MacOS.....	51
3.5.7.5 Windows.....	52
3.5.8 Tabelas comparativas dos resultados obtidos.....	54
3.6 RESULTADOS E DISCUSSÃO.....	55
3.6.1 Conclusões com base nos Resultados.....	58
3.6.2 Recomendações e Melhorias.....	59
4. CONCLUSÃO.....	61
REFERÊNCIAS.....	62
APÊNDICE A.....	64
APÊNDICE B.....	69

1. INTRODUÇÃO

O crescente aumento de dispositivos conectados à internet em ambientes acadêmicos, especialmente a de redes sem fio (Wi-fi) trouxe à tona preocupações a respeito da segurança de conexão com essas redes. O eduroam é um exemplo disso, pois trata-se de um serviço mundial de acesso Wi-Fi seguro e federado, utilizado por milhões de usuários em instituições de ensino e pesquisa ao redor do mundo. Porém, a autenticidade e integridade desse acesso depende fortemente de duas tecnologias centrais: o EAP e a infraestrutura baseada em servidores RADIUS, que são responsáveis pelo processo de autenticação e autorização dos usuários.

Apesar da robustez teórica dessas tecnologias, diversos estudos apontam que a má configuração dos protocolos EAP e a ausência de validação adequada dos certificados digitais por parte dos usuários são uma das principais fontes de vulnerabilidades exploráveis. Trabalhos como os de Franken et al. (2015) e Balduzzi et al. (2018) demonstram empiricamente que muitos dispositivos conectam-se automaticamente a servidores maliciosos quando o certificado do servidor não é verificado.

Além disso, documentos técnicos como o FreeRADIUS Technical Guide (2014) e a análise de Gast (2012) ressaltam que a própria arquitetura distribuída do RADIUS embora eficiente e amplamente adotada torna-se um ponto de vulnerabilidade quando implementada sem verificações estritas de integridade, autenticação do servidor e políticas adequadas de roteamento.

Diante disso, este trabalho propõe uma investigação dessas vulnerabilidades já encontradas, com foco nos protocolos EAP e RADIUS que são utilizados nas conexões do tipo 802.1X, como o eduroam. O objetivo da análise consiste em entender em conceito e na aplicação como essas vulnerabilidades podem ser exploradas e se podem resultar no comprometimento da integridade da rede como um todo. Essas considerações também poderiam tangenciar recomendações de configuração e práticas que contribuíssem para tornar o eduroam e redes semelhantes mais robustas frente a ameaças contemporâneas.

A organização do texto está distribuída da seguinte forma: Após a introdução e os objetivos, a Seção 2 apresenta a fundamentação teórica, cobrindo as tecnologias e padrões utilizados pela rede eduroam, como o *Institute of Electrical and Electronics Engineers* (IEEE) 802.1X, EAP, RADIUS e logo após se encontra um trecho que trata a respeito do próprio eduroam e seu funcionamento, incluindo processo de autenticação e fragilidades. A seção 2.5 sintetiza as vulnerabilidades já relatadas em trabalhos anteriores. A seção 3 descreve a proposta experimental, as ferramentas, o desenho do ambiente e os ataques a serem simulados. A Seção 4 reúne as considerações finais e o cronograma.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Avaliar a segurança da infraestrutura de autenticação do eduroam, com ênfase nas tecnologias utilizadas na ampla maioria das universidades e instituições de pesquisa, que são os protocolos EAP e os servidores RADIUS, considerando as principais vulnerabilidades e

como elas podem ser exploradas. Também está previsto demonstrar os problemas na prática e propor melhorias e sugestões que aumentem a segurança dessas redes Wi-Fi acadêmicas.

1.1.2 Objetivos Específicos

Os objetivos específicos deste trabalho são os seguintes:

- Compreender individualmente o funcionamento e as vulnerabilidades dos protocolos EAP e RADIUS;
- Configurar um ambiente de testes controlado que simule a arquitetura da rede eduroam para realizar ataques às vulnerabilidades que foram encontradas durante a revisão bibliográfica;
- Verificar, através dos resultados obtidos, quais cenários apresentam ameaça para o usuário;
- Formular um conjunto de recomendações que possam aumentar a segurança da rede eduroam e de redes similares.

2. FUNDAMENTAÇÃO TEÓRICA

A metodologia adotada neste trabalho é de natureza exploratória e experimental, com o intuito de compreender em detalhes o funcionamento de cada uma das tecnologias empregadas na arquitetura, identificando possíveis vulnerabilidades e propondo melhorias de segurança baseadas em testes práticos realizados em ambiente controlado.

Neste primeiro momento, será apresentado o estudo bibliográfico realizado sobre as tecnologias que formam a arquitetura da rede eduoam, além de apresentar uma revisão dos artigos já publicados referentes a vulnerabilidades já documentadas nesse tipo de rede.

2.1. IEEE 802.1X CONTROLE DE ACESSO BASEADO EM PORTAS

O IEEE 802.1X é um padrão de controle de acesso à rede baseado em portas, *Port Based Network Access Control* (PNAC), é projetado para garantir que somente dispositivos autenticados e autorizados consigam se conectar a uma rede local. Como discute Minella (2016), foi inicialmente desenvolvido para redes *Ethernet* cabeadas, mas tornou-se fundamental também em redes sem fio como em Wi-Fis corporativas que fazem uso do modo *Wi-Fi Protected Access - Enterprise (WPA-Enterprise)*, fornecendo um mecanismo robusto de autenticação antes de permitir o tráfego de dados do cliente na rede.

No modelo 802.1X, existem três entidades principais, como descrito nas obras de Minella (2016), mas também de Stallings (2014), sendo elas:

- Suplicante é o dispositivo ou usuário que solicita o acesso à rede (ex: computador, smartphone);
- Autenticador é o dispositivo de rede que controla o ponto de entrada na rede, geralmente um *switch Ethernet* nos casos de rede cabeada ou um ponto de acesso Wi-Fi nos casos de redes sem fio, ele é o responsável por realizar o bloqueio do tráfego quando o cliente não está autenticado;
- Servidor de autenticação é usualmente um servidor dedicado com a função de verificar as credenciais apresentadas pelo suplicante, decidindo se ele pode ou não se conectar a rede. Normalmente são usados servidores RADIUS que implementam as funções de Autenticação, Autorização e Contabilização (AAA).

Em uma rede Wi-Fi corporativa que faz uso do 802.1X, o computador do usuário seria o suplicante, o ponto de acesso sem fio o autenticador, e o servidor de autenticação seria um servidor RADIUS localizado na rede da organização.

O funcionamento do 802.1X pode ser comparado a um porteiro que só permite a passagem após a identificação e autenticação bem-sucedida do solicitante. Em (Minella, 2016), verifica-se que antes de realizar a autenticação, o autenticador coloca a porta de rede do suplicante em um estado restrito, em que todo o tráfego de dados é bloqueado, sendo permitido somente que informações de controle de autenticação trafeguem pela porta.

Desta forma, o 802.1X define duas portas lógicas associadas a cada ponto de conexão do autenticador. A primeira é a porta não controlada, usada exclusivamente para a troca inicial de mensagens EAP e demais informações de autenticação, ela permanece aberta

mesmo quando o cliente ainda não foi validado. A segunda é a porta controlada, que permanece bloqueada até que o processo de autenticação seja concluído com sucesso. Somente após a validação das credenciais o autenticador libera essa porta, permitindo o tráfego normal de dados do usuário.

Em outras palavras, assim que um dispositivo suplicante se conecta a uma rede que faz o uso do 802.1X, o autenticador inicia uma comunicação de autenticação através do protocolo *EAP over LAN* (EAPoL) na porta não controlada, e solicita a identificação do suplicante, o autenticador também atua como um retransmissor, repassando as respostas do suplicante ao servidor de autenticação agindo apenas como um intermediário durante essa fase, fazendo o encapsulamento das mensagens EAP recebidas do suplicante em pacotes RADIUS e enviando para o servidor de autenticação, o contrário também acontece, na qual o autenticador recebe os pacotes RADIUS do servidor de autenticação os encapsula em EAPoL e envia ao suplicante. Durante todo esse processo de autenticação, nenhuma outra comunicação como requisições *Dynamic Host Configuration Protocol* (DHCP), *ping* ou acesso a recursos é permitido (Minella, 2016; Stallings, 2014).

Se as credenciais do suplicante forem validadas com sucesso pelo servidor de autenticação, o autenticador então desbloqueia a porta controlada e permite o acesso do dispositivo à rede. A partir desse momento, o suplicante passa a ter acesso aos recursos da *Local Area Network* (LAN) de acordo com as políticas definidas, pois o servidor de autenticação pode colocar o suplicante em uma porta específica ou limitar as permissões das políticas definidas e utilizadas na rede (Minella, 2016).

Ainda, no caso de falha na autenticação, seja por credenciais inválidas ou rejeitadas, o acesso à rede permanece bloqueado pois a porta controlada não é liberada e o suplicante não consegue se autenticar na rede. Conforme as políticas de segurança estabelecidas na rede o suplicante pode reiniciar o processo de autenticação ou ser isolado.

Assim, na abordagem proposta por Minella (2016), o IEEE 802.1X é o responsável por prover um padrão de autenticação a nível de enlace que em conjunto com o protocolo EAP e servidores RADIUS, estabelecendo um controle rigoroso de quem pode acessar a rede. Somente após o cumprimento bem-sucedido dessa etapa de autenticação baseada em portas é que o dispositivo cliente passa a ter o tráfego normal desbloqueado na rede, o que contribui significativamente para a segurança das redes que fazem o uso desse padrão (Minella, 2016; Stallings, 2014).

2.2. EXTENSIBLE AUTHENTICATION PROTOCOL (EAP)

Com base na obra de Stallings (2014), e em suas definições, podemos afirmar que o EAP é um arcabouço de autenticação flexível definido pela *Internet Engineering Task Force* (IETF) através da *Request for Comments* (RFC) 3748 (Aboba et al, 2004). Ele suporta múltiplos métodos de autenticação diferentes que também são conhecidos como métodos EAP. São esses métodos que tornam o protocolo flexível, pois possibilita escolher o mecanismo adequado conforme o ambiente.

Ao contrário dos protocolos de enlace convencionais, o EAP não foi projetado para transportar dados de usuário, mas sim para transportar apenas credenciais e mensagens de

autenticação. O próprio EAP define formatos de mensagem genéricas, enquanto cada tecnologia de rede define como essas mensagens EAP serão encapsuladas através dos métodos EAP (Aboba, 2004).

Uma das principais vantagens do EAP é possibilitar a autenticação forte em redes de acesso, permitindo o uso de diferentes métodos de autenticação sem necessidade de modificar a infraestrutura de enlace. Durante o processo de autenticação em uma rede que utiliza o 802.1X, o suplicante comunica-se com o autenticador por meio do protocolo EAPoL na interface de enlace local. O autenticador, por sua vez, encapsula essas mensagens EAP em pacotes RADIUS e os encaminha ao servidor de autenticação conforme ilustra a Figura 1.

Com isso, todos os detalhes da autenticação como a verificação de certificados e senhas são tratados exclusivamente entre o suplicante e o servidor, permanecendo transparente para o dispositivo autenticador. Ou seja, o autenticador está no caminho da comunicação, mas não interfere na lógica da autenticação tornando-se essencial, e também "invisível" no conteúdo do processo (Aboba, 2004).

Essa separação e transparência de dados no autenticador permite que ele suporte qualquer método EAP padronizado sem precisar implementar cada um deles, pois basta que ele encaminhe as mensagens EAP para o servidor RADIUS responsável por processá-las.

Essa flexibilidade é fundamental, quando consideramos que existem dezenas de métodos EAP documentados em RFCs ou especificações de fornecedores, e cada uma com requisitos específicos de autenticação e segurança (Aboba, 2004; Stanley, Walker e Aboba, 2005).

Vamos especificar um pouco mais sobre os principais métodos EAP:

- **EAP-MD5** (EAP *Message Digest* 5) é o método EAP mais simples e antigo, ele funciona com o servidor via autenticador enviando um desafio EAP-*Request* do tipo MD5-*Challenge*, contendo um *nonce* aleatório. O cliente responde com um EAP-*Response* contendo o *hash* MD5 da combinação do *nonce* com sua senha secreta. O servidor compara esse *hash* com o esperado pois ele conhece a senha e decide entre EAP-*Success* e EAP-*Failure*. Nesse método não há troca de certificados nem túnel criptografado, apenas autenticação unidirecional do cliente, o que faz com que esse método não apresente um bom nível de segurança;
- **EAP-TLS** (EAP *Transport Layer Security*) utiliza um túnel TLS completo para autenticação mútua via certificados X.509. Funciona com servidor RADIUS enviando seu certificado ao cliente, que então o valida. Em seguida, o cliente envia seu próprio certificado para o servidor. Em essência, é semelhante a um *handshake* TLS onde há troca de mensagens e validação de certificados em ambos os lados. Após o acordo, ambos derivam chaves de sessão, 4 tramas EAPoL-*Key*, para criptografar o tráfego Wi-Fi. Possui um nível de segurança muito alto devido ao suporte a autenticação mútua o que faz com que mesmo que a senha do usuário seja comprometida, o invasor ainda precisaria do certificado privado do cliente para conseguir obter o acesso, além de também fazer uso do mesmo TLS dos *Hypertext Transfer Protocol Secure* (HTTPS) modernos, garantindo mais confidencialidade e integridade;
- **EAP-TTLS** (EAP *Tunneled* TLS) é semelhante ao EAP-TLS, mas simplifica os requisitos do lado do cliente. Ele estabelece primeiro um túnel TLS seguro usando

somente um certificado do servidor e depois dentro desse túnel criptografado, qualquer método legado pode ser usado para autenticar o cliente como *Password Authentication Protocol* (PAP), *Challenge-Handshake Authentication Protocol* (CHAP), *Microsoft Challenge Handshake Authentication Protocol version 2* (MSCHAPv2), sem expor as senhas em texto claro na rede. Possui um nível de segurança muito alto no transporte, mas a segurança final depende muito do método interno utilizado para autenticar o cliente;

- **EAP-PEAP** (EAP *Protected* EAP) foi desenvolvido pela Microsoft/Cisco/*Rivest-Shamir-Adleman* (RSA) e funciona de modo muito parecido com o EAP-TTLS, criando primeiro um túnel TLS usando apenas o certificado do servidor e dentro desse túnel, normalmente utiliza o método MSCHAPv2 (EAP-MSCHAPv2) ou *Generic Token Card* (GTC) para autenticar o cliente por senha. O nível de segurança desse método é semelhante ao do EAP-TTLS. O túnel TLS protege a senha contra a espionagem externa, mas a segurança final depende do método interno utilizado, no caso o MSCHAPv2 é conhecido por ter fraquezas, mas também protege a senha por *hash* com *nonces*;
- **EAP-FAST** (EAP *Flexible Authentication via Secure Tunneling*) foi criado pela Cisco como alternativa ao PEAP/TTLS para organizações que não podiam usar certificados. Ele também estabelece um túnel TLS, mas em vez de certificados de cliente usa *Protected Access Credentials* (PAC) que são segredos provisionados ao cliente com antecedência. Seu funcionamento se dá através da criação de um túnel TLS mútuo e protegido entre o servidor e o cliente, e seu diferencial é que ao invés de validar o certificado do cliente, o servidor exige que o cliente apresente um PAC válido que é semelhante a um arquivo de configuração seguro. Esse PAC pode ser entregue ao cliente manualmente de forma *offline* ou automaticamente por meio de um canal protegido. Uma vez no túnel, as credenciais do usuário são trocadas para realizar a autenticação. Seu nível de segurança depende da proteção do PAC, esse método é menos robusto que o EAP-TLS, mas superior a métodos que só utilizam senha.

Há também métodos específicos para certas tecnologias, como o EAP-SIM (EAP *Subscriber Identity Module*) e EAP-AKA (EAP *Authentication and Key Agreement*), usados por operadoras de telefonia móvel para autenticar dispositivos através do cartão *Subscriber Identity Module* (SIM).

A RFC 4017 (Stanley, Walker e Aboba, 2005) define requisitos mínimos de segurança para métodos EAP em redes *Wireless Local Area Network* (WLAN), recomendando o uso apenas de métodos que suportam autenticação mútua e protejam as credenciais dentro de um canal seguro normalmente via TLS, em vez de opções ultrapassadas e vulneráveis como o EAP-MD5.

O EAP por si só não provê confidencialidade ou integridade além do necessário para sua própria sequência de mensagens. A conferência ACM WiSec que é um dos principais eventos internacionais dedicados à segurança de redes sem fio, demonstrou através de análises empíricas, ataques e avaliações rigorosas, que a segurança depende fortemente do método EAP escolhido. Utilizando a rede eduoam como exemplo, é fundamental a utilização

de um método que implemente um túnel TLS protegido para garantir que as credenciais sensíveis do usuário não sejam expostas em texto claro (Vanhoeft e Van de Veen, 2015).

Entretanto, a efetividade dessas medidas de proteção requer configurações corretas principalmente do lado dos clientes(suplicante). Se o usuário não verificar a identidade do servidor de autenticação deixando de validar o certificado digital do servidor RADIUS, por exemplo, ele até pode fazer uso de um método que implemente o tunelamento protegido como o EAP-TLS/PEAP, mas mesmo assim ele pode ser induzido a se conectar a um falso ponto de acesso/servidor e ter suas credenciais roubadas por um atacante. Esses incidentes reais nas redes eduroam demonstraram essa vulnerabilidade, ressaltando a importância de usar EAP com autenticação mútua e de conscientizar os usuários para nunca ignorar os avisos de certificados de servidores (Vanhoeft e Van de Veen, 2015).

Figura 1: Encapsulamentos do protocolo EAP em uma rede do padrão IEEE 802.1X.



Fonte: Autor (2025).

2.3. RADIUS-REMOTE AUTHENTICATION DIAL-IN USER SERVICE

Foram estudadas e analisadas as obras de Dirk van der Walt, (2011) e FreeRADIUS Technical Guide, (2014), com o intuito de entender os diferentes tipos de configuração disponíveis e suas características.

O RADIUS é um protocolo de comunicação voltado para os serviços de Autenticação, Autorização e Contabilização (AAA) em redes. Desenvolvido inicialmente em 1991 pela empresa Livingston para gerenciar acessos discados (*dial-up*), foi posteriormente padronizado pela IETF na RFC 2865 (Rigney et al., 2000) e consolidou-se como base para controle de acesso em diversos ambientes, incluindo provedores de Internet, redes corporativas e redes Wi-Fi.

No contexto das redes modernas, o RADIUS atua como o servidor *back-end* típico na implementação do IEEE 802.1X, tanto em redes cabeadas quanto em redes sem fio, sendo responsável por validar as credenciais dos usuários e transmitir as políticas de autorização aos equipamentos de acesso, como *switches* e *access points* (APs).

Arquiteturalmente, o RADIUS segue um modelo de cliente/servidor na camada de aplicação. O dispositivo de acesso à rede chamado de *Network Access Server* (NAS), que pode ser um roteador, *switch* ou ponto de acesso *wireless*, incorpora um cliente RADIUS. Quando um usuário tenta se conectar, o NAS coleta as credenciais fornecidas como o nome de usuário e senha, ou um certificado de cliente e envia um *Access-Request* para o servidor

RADIUS contendo essas informações e outros atributos de contexto como o endereço de rede, porta física, *Service Set Identifier* (SSID) da rede e outras informações referentes a rede LAN (Rigney, 2000; FreeRADIUS Technical Guide, 2014).

A comunicação RADIUS tradicionalmente utiliza *User Datagram Protocol* (UDP) através da porta 1812 para autenticação e 1813 para contabilização, com proteção básica por meio de um segredo compartilhado entre o cliente eo servidor que é usado tanto para autenticar os pacotes quanto para ofuscar a senha do usuário via MD5. O servidor RADIUS, ao receber o *Access-Request*, verifica as credenciais usando um método compatível – pode ser uma autenticação simples (PAP/CHAP) ou um método EAP encapsulado dentro do RADIUS. Os dados de referência para validação podem residir no próprio servidor ou em bases externas, como diretórios *Lightweight Directory Access Protocol* (LDAP)/*Active Directory*, bancos *Structured Query Language* (SQL), etc (Rigney, 2000; Dirk van der Walt, 2011; FreeRADIUS Technical Guide, 2014).

Após processar a solicitação de autenticação, o servidor RADIUS envia de volta ao NAS uma resposta indicando o resultado, através de um dos três códigos possíveis.

- ***Access-Accept***: indica que o usuário foi autenticado com sucesso e está autorizado a acessar a rede;
- ***Access-Reject***: indica falha na autenticação podendo ser credenciais inválidas, credenciais rejeitadas ou outro motivo, o acesso a rede é negado ao usuário;
- ***Access-Challenge***: indica que são necessárias informações adicionais do usuário para completar a autenticação. Esse código é usado, por exemplo, em autenticação de múltiplos fatores solicitando um *token*, *Personal Identification Number* (PIN) ou segunda senha ou em protocolos EAP que requerem múltiplas trocas de mensagem. Durante um diálogo EAP prolongado, o uso de *Access-Challenge* permite estabelecer um túnel seguro entre o suplicante e o servidor via EAP dentro do RADIUS, antes do envio das credenciais finais, mantendo essas credenciais ocultas do NAS intermediário (Aboba e Simon, 2003; FreeRADIUS Technical Guide, 2014).

Junto com o resultado da autenticação, as respostas RADIUS podem carregar diversos atributos de autorização que instruem o NAS sobre os privilégios de acesso do usuário autenticado. Por exemplo, no caso de um *Access-Accept*, o servidor pode incluir atributos definindo a *Virtual Local Area Network* (VLAN) na qual o usuário deve ser colocado, o endereço *Internet Protocol* (IP) a ser atribuído, o tempo máximo de conexão ou parâmetros de qualidade de serviço permitidos para aquela sessão, entre outros .

Essa capacidade de transmitir parâmetros de autorização permite integrar o RADIUS com políticas de controle de acesso granulares, um usuário pode ser autenticado com sucesso, mas ainda assim ter limitações específicas de acordo com os atributos recebidos por exemplo, ser colocado em uma VLAN de visitantes ou ter largura de banda reduzida para certos serviços (Dirk van der Walt, 2011; FreeRADIUS Technical Guide, 2014).

Uma característica importante do RADIUS é o suporte a encapsulamento de EAP em suas mensagens, viabilizando o modelo de autenticação distribuída do 802.1X. Conforme mencionado e exemplificado na seção anterior, o NAS (autenticador) não precisa implementar os detalhes de cada método EAP; ele atua repassando as mensagens entre o suplicante e o servidor.

Para isso, o RFC 3579 (Aboba e Simon, 2003) definiu o atributo *EAP-Message* no RADIUS, através do qual pacotes EAP são transportados dentro dos campos de atributo dos pacotes RADIUS.

Em termos práticos, quando o NAS recebe uma mensagem EAP do cliente (por exemplo, via EAPoL), ele a encapsula em um *Access-Request* contendo o atributo *EAP-Message* e a envia ao servidor RADIUS. O servidor processa essa mensagem executando uma etapa de um *handshake* EAP-TLS ou validando um desafio de senha e responde com outro pacote RADIUS *Access-Challenge*, *Access-Accept* ou *Reject* contendo também um atributo *EAP-Message* para transportar a próxima mensagem EAP de volta ao suplicante.

Esse mecanismo permite que praticamente qualquer método EAP funcione sobre RADIUS de forma transparente, aproveitando a infraestrutura existente destes servidores para a autenticação para o EAP. O RADIUS também suporta a interconexão de múltiplos servidores em uma hierarquia de *proxies*, o que é amplamente explorado em cenários de *roaming* entre domínios distintos. Nessa configuração, um servidor RADIUS pode encaminhar solicitações de autenticação para outro servidor RADIUS, formando uma cadeia de retransmissão até alcançar o servidor responsável pelo domínio do usuário, é possível observar um exemplo dessa hierarquia na Figura 2 (Roam Whitepaper, 2020; FreeRADIUS Technical Guide, 2014).

Uma entidade que faz o uso destes servidores em âmbito global é a federação eduroam, que provê acesso Wi-Fi federado mundialmente para usuários acadêmicos. Na eduroam, os pontos de acesso das instituições (NAS) encaminham as solicitações via RADIUS ao servidor RADIUS local, o qual age como *proxy* repassando-as a um servidor RADIUS regional ou nacional, e assim sucessivamente até o servidor RADIUS da instituição de origem do usuário.

Para viabilizar esse roteamento, cada identidade de usuário é formatada como um *Network Access Identifier* (NAI) com o domínio da instituição (por exemplo, *usuario@universidade.br*), de forma semelhante a um endereço de e-mail. Os servidores RADIUS ao longo do caminho reconhecem o sufixo do NAI (*realm*) e sabem para qual próximo servidor encaminhar a requisição, até alcançar o destino correto (Gast, 2012; Roam Whitepaper, 2020).

Com essa arquitetura, um usuário pode se autenticar em qualquer lugar da federação eduroam usando as credenciais da sua própria instituição, pois a solicitação sempre será encaminhada ao servidor RADIUS apropriado que poderá validá-las. Esse modelo de *proxies* RADIUS provê escalabilidade e flexibilidade, mas também introduz desafios de segurança e desempenho em cenários globais como o aumento da latência e mais pontos suscetíveis a falhas (Roam Whitepaper (2020).

Em termos de segurança, o RADIUS possui algumas limitações conhecidas. Por operar sobre UDP e utilizar apenas um esquema simples de segredo compartilhado, os pacotes RADIUS não são protegidos por criptografia ponta a ponta. A única proteção aplicada é a ofuscação da senha do usuário no atributo *User-Password*, utilizando um *hash* MD5 baseado no segredo compartilhado. No entanto, essa técnica não oferece confidencialidade robusta, e os demais campos do pacote, incluindo cabeçalhos e a maioria dos atributos são transmitidos em texto puro. Isso expõe informações sensíveis, como a

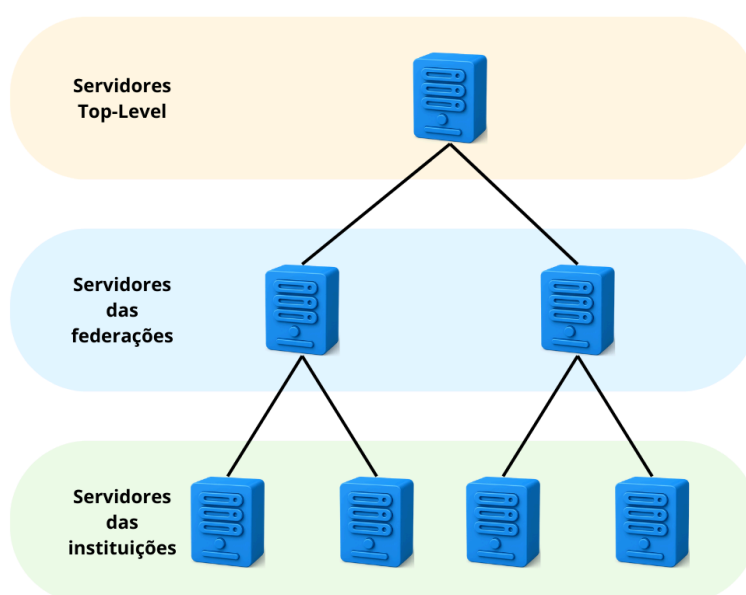
identidade do usuário, o endereço *Media Access Control* (MAC) do dispositivo (*Calling-Station-Id*), o nome do ponto de acesso, e o nome da rede visitada, entre outras informações (Dirk van der Walt, 2011; FreeRADIUS Technical Guide, 2014).

Isso significa que, mesmo quando métodos EAP seguros são empregados para proteger a senha do usuário, metadados pessoais ainda podem ser coletados por um atacante que consiga monitorar o tráfego RADIUS na rede. Outra preocupação está na confiabilidade: como o RADIUS utiliza o protocolo UDP para transporte, a perda de um único pacote durante uma sequência de múltiplas trocas EAP pode causar atrasos ou até a falha completa da autenticação. Isso ocorre porque o UDP não garante a entrega nem a ordem correta dos pacotes. Em ambientes com várias camadas de *proxies* RADIUS em série, como no eduroam, o impacto pode ser ainda maior, aumentando a probabilidade de retransmissões e *timeouts* no processo (FreeRADIUS Technical Guide, 2014; Roam Whitepaper, 2020).

Para mitigar essas questões, surgiram extensões e boas práticas como o uso de RADIUS sobre TLS (*RadSec*), definido no RFC 6614 (Winter et al., 2012) para estabelecer um túnel TLS entre servidores RADIUS, fornecendo confidencialidade e integridade aos pacotes AAA *end-to-end*. Implementações mais recentes também permitem operar o RADIUS sobre *Transmission Control Protocol* (TCP) ou *Stream Control Transmission Protocol* (SCTP), melhorando o controle de entrega dos pacotes (Roam Whitepaper. (s.d.)).

Em resumo, apesar das limitações apontadas, o RADIUS permanece amplamente adotado devido à sua simplicidade e interoperabilidade, e continua evoluindo com aprimoramentos de segurança para atender às demandas atuais.

Figura 2: Exemplo de hierarquia entre servidores RADIUS da rede eduroam.



Fonte: Autor (2025).

2.4 PROCESSO DE AUTENTICAÇÃO NO EDUROAM

O eduroam é uma infraestrutura federada de *roaming* Wi-Fi usada mundialmente por instituições acadêmicas, baseada no padrão IEEE 802.1X e em uma hierarquia de servidores RADIUS. Quando um usuário de uma instituição visita outra, suas credenciais são encaminhadas pela hierarquia de servidores RADIUS até o servidor de origem, que faz a verificação.

Como descrito no site oficial do eduroam, “a hierarquia RADIUS encaminha as credenciais do usuário para a instituição de origem onde elas são verificadas e validadas”. No funcionamento da arquitetura federada as credenciais fornecidas pelo usuário na instituição visitada são encaminhadas de forma segura, por uma cadeia hierárquica de servidores RADIUS até a instituição de origem, que é a única responsável por verificar e validar a identidade do usuário (eduroam, 2023).

Utilizando como base a arquitetura descrita em Roam Whitepaper (2020), vamos detalhar o fluxo de autenticação passo a passo:

1. Inicialização da autenticação 802.1X EAP-*Identity*;
2. O cliente (suplicante) se conecta ao SSID eduroam da instituição visitante (X) e inicia o processo 802.1X;
3. O cliente envia um EAP-*Response/Identity* contendo sua identidade externa no formato *usuario@realm* (ex: *rian.block@ufsc.br*). Este identificador revela o *realm* (domínio) da instituição de origem (UFSC);
4. O ponto de acesso (AP) ou controlador atua como autenticador e recebe essa mensagem EAP do cliente, encaminhando ao RADIUS local;
5. O Autenticador encapsula a mensagem EAP dentro de um pacote RADIUS *Access-Request* e o envia ao servidor RADIUS local da Instituição X, que é o provedor de serviço, SP;
6. No pacote são incluídos atributos RADIUS como *Calling-Station-Id* (MAC do cliente), *Called-Station-Id* (identificador do AP/rede) e o nome de usuário no atributo *User-Name*;
7. O servidor RADIUS local verifica o *realm* do usuário (*ufsc.br*), como não pertence à sua instituição, ele não tenta autenticar localmente. Em vez disso, segundo as regras de roteamento configuradas, ele encaminha (“faz *proxy*”) a requisição para o próximo salto na hierarquia eduroam, geralmente o servidor RADIUS nacional ou regional designado para o domínio de *ufsc.br*;
8. Roteamento na hierarquia de RADIUS. O pedido percorre a cadeia de *proxies* RADIUS conforme o domínio do usuário. Por exemplo, em *roaming* internacional o RADIUS da instituição X envia ao RADIUS nacional do país X, que reconhece que o *realm* termina em “.br” (Brasil). Este então encaminha ao RADIUS de topo regional/mundial, sendo ele, servidor eduroam continental, que por sua vez direciona ao RADIUS nacional brasileiro;
9. Finalmente, o RADIUS nacional do Brasil identifica que *ufsc.br* pertence à federação nacional brasileira e entrega a solicitação ao servidor RADIUS da UFSC (o *Identity Provider*, IdP).

Em cada salto, o *proxy* RADIUS atua repassando o conteúdo EAP nos atributos *EAP-Message* de requisições e respostas sem decodificar as credenciais internas, preservando a privacidade do usuário tanto quanto o método EAP permitir.

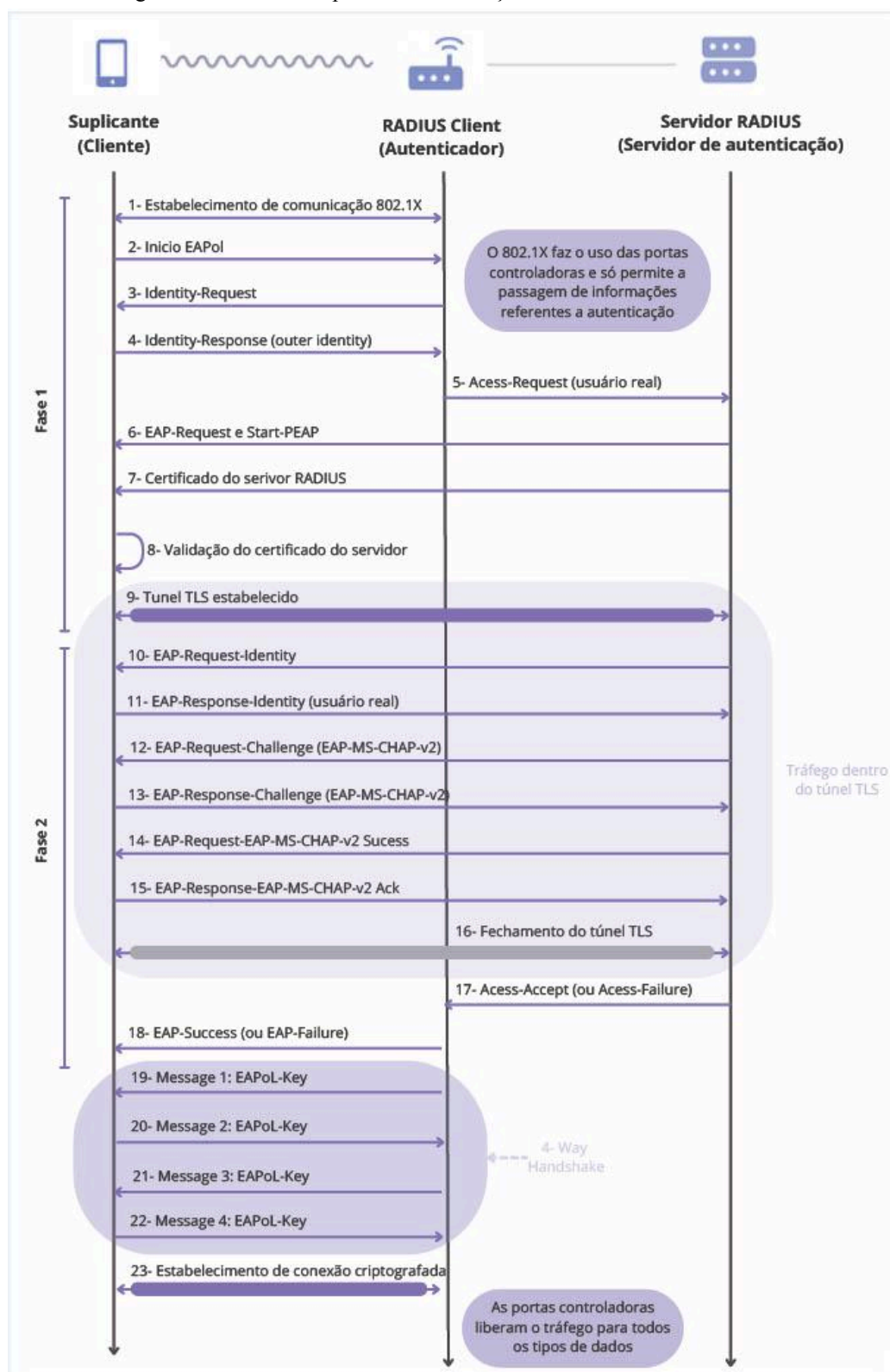
Após a autenticação bem-sucedida, o usuário visitante passa a ter acesso à rede da instituição anfitriã. Neste modelo federado, a autenticação da identidade é responsabilidade da instituição de origem, enquanto a autorização de uso de quais recursos estão disponíveis para o visitante é responsabilidade da instituição que está oferecendo o acesso.

Desta forma, uma vez autenticado, o usuário visitante passa a ter acesso à rede da instituição X.

Em geral, o visitante recebe acesso à Internet e recursos públicos de X, mas não a sistemas internos restritos. Opcionalmente, se configurado, o RADIUS de X envia registros de *accounting* de volta ao servidor UFSC, informando a duração da sessão e volume de dados, embora muitas federações limitem esse recurso por questões de privacidade e carga (FreeRADIUS Technical Guide, 2014; Gast, 2012).

Na Figura 3 é exemplificado visualmente a troca de mensagens para estabelecer uma conexão no protocolo 802.1X.

Figura 3: Processo completo de autenticação com PEAP-MSCHAPv2.



Fonte: Adaptado de SECUREW2 (2022).

2.4.1. Fragilidades conhecidas

Apesar da escalabilidade e segurança conceitual do modelo federado do eduroam, existem pontos de falha e vetores de ataque relevantes como disponibilidade e roteamento estático. Devido ao uso de rotas RADIUS pré-configuradas, não há algoritmo de *failover* nativo no RADIUS. Se um servidor intermediário falha (por falta de resposta), *proxies* não saberão diferenciar qual nó no caminho caiu e podem marcar erroneamente servidores saudáveis como inoperantes (FreeRADIUS Technical Guide, 2014).

Na prática, isso pode causar atrasos perceptíveis para os usuários, que enfrentam *timeouts* durante o processo de autenticação. Além de que, uma falha em um único servidor pode levar a uma reação em cadeia, fazendo com que *proxies* sucessivos interpretem a falta de resposta como falha no próximo salto, que ativam mecanismos de *failover* que propagam o erro por toda a cadeia de *proxies*. Isso pode resultar na exclusão indevida de múltiplos caminhos funcionais. Para mitigar esse problema, recomenda-se a implementação da extensão RFC 5997 (Aboba, DeKok e Zorn, 2010) (RADIUS *Status-Server*), que permite aos *proxies* testar silenciosamente a disponibilidade dos servidores antes de marcá-los como inativos. Também é essencial configurar *timeouts* e *dead times* de forma conservadora, evitando reações precipitadas do sistema (Aboba et al., 2010).

Mesmo assim, um usuário final vê apenas uma mensagem de “autenticação falhou” genérica. O IEEE 802.1X não transmite o motivo como queda do servidor remoto ou credenciais inválidas, e isso dificulta a identificação da causa real do problema (Gast, 2012).

Ataques do tipo *Evil Twin* ilustrado na Figura 5, explora configurações incorretas do lado do cliente, especialmente quando o dispositivo não valida corretamente o certificado do servidor RADIUS antes de transmitir suas credenciais. Para garantir essa validação, é essencial que o certificado da autoridade certificadora (CA) raiz do eduroam ou da instituição seja instalado corretamente e que a verificação do nome do servidor esteja habilitada no perfil da rede Wi-Fi (Vanhoeft & Ronny, 2015).

Se isto não for feito, o dispositivo poderá automaticamente conectar-se a um AP falso que anuncie “eduroam” e opere um servidor RADIUS malicioso. Para conseguir roubar as credenciais, os atacantes geralmente utilizam dois ataques em conjunto, o *Evil Twin*, Figura 5 e o *Man In The Middle* mostrado na Figura 4.

Estudos experimentais mostraram que este ataque é eficaz, em um experimento com 37 usuários relativamente experientes em tecnologia da informação (TI), mais de um terço teve suas credenciais roubadas por um AP falso.

As credenciais do eduroam frequentemente são as mesmas usadas em outros serviços acadêmicos como *e-mail*, portais, *Virtual Private Network* (VPN), ampliando o risco (Vanhoeft & Ronny, 2015).

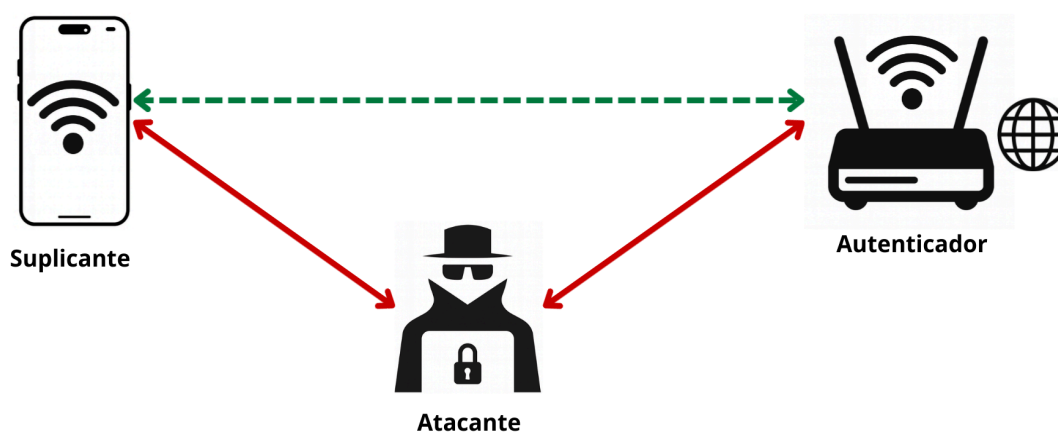
Para evitar isso, recomenda-se usar o eduroam *Configuration Assistant Tool* (CAT) para configurar automaticamente o perfil 802.1X nos dispositivos e garantir que o certificado do servidor seja validado. Também é recomendável habilitar a opção “*Identity Privacy*” (identidade anônima) nos clientes para usar uma identidade externa anônima (ex.: anon@instituicaoX.br), de modo que apenas o *realm* verdadeiro seja visível nos *proxies*, protegendo o nome de usuário real (eduroam, 2023).

A respeito da proteção de credenciais e privacidade, o eduroam recomenda fortemente métodos EAP com túnel TLS (por exemplo, PEAP, EAP-TTLS ou EAP-TLS) que cifram as credenciais no tráfego RADIUS. Métodos legados inseguros como EAP-MD5, *Lightweight Extensible Authentication Protocol* (LEAP) são proibidos. Dessa forma, mesmo que pacotes RADIUS sejam interceptados, as senhas circulam protegidas dentro do túnel TLS. Ainda assim, a identidade externa (usuario@realm) no *outer identity* pode vaziar para *proxies* intermediários. Por isso, o uso de *outer identity* anônima já mencionada é recomendado (Gast, 2012).

Com o intuito de fortalecer toda a camada de transporte RADIUS, está em adoção o RadSec (RADIUS sobre TLS, RFC 6614 (Winter et al., 2012)), que substitui UDP por TCP/TLS, permitindo criptografar todo o tráfego RADIUS e verificar mutuamente os servidores (Winter et al., 2012). Entre outros benefícios, está a detecção explícita de alcance dos *peers* via sessões TCP duradouras e a proteção dos atributos sensíveis de usuários. Em breve, mecanismos de descoberta dinâmica por *Domain Name System* (DNS) (*Naming Authority Pointer* (NAPTR)/*Service Record* (SRV)) deverão facilitar a escalabilidade desse modelo.

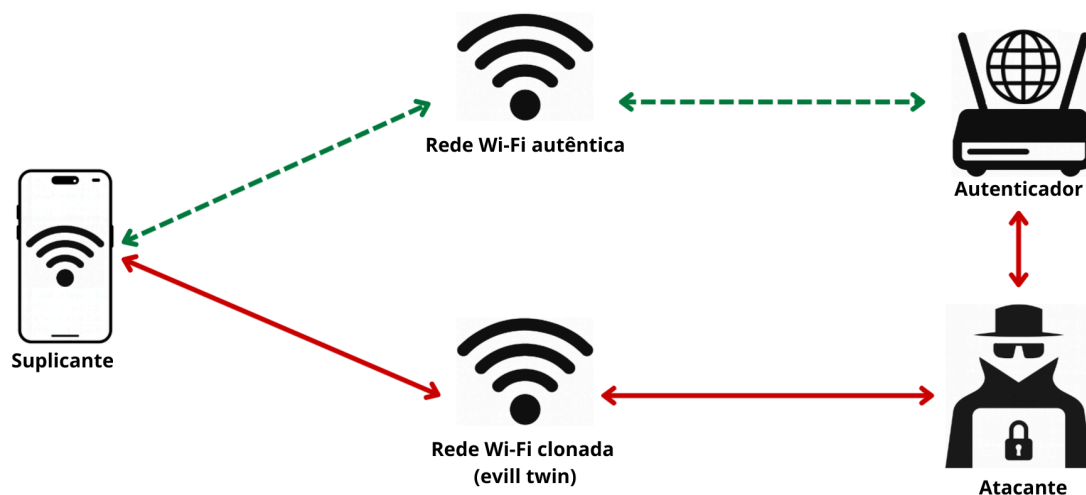
O eduroam equilibra conveniência de *roaming* e segurança usando padrões abertos (802.1X, EAP, RADIUS) e boas práticas de configuração. No entanto, sua segurança efetiva depende não só da robustez dos protocolos criptográficos, mas também da correta configuração dos dispositivos dos clientes e servidores em todos os domínios participantes (FreeRADIUS Technical Guide, 2014; Gast, 2012).

Figura 4: Representação de um ataque *man-in-the-middle*.



Fonte: Autor (2025).

Figura 5: Representação de um ataque *evil twin*.



Fonte: Autor (2025)

2.4.2. Eduroam Configuration Assistant Tool (CAT)

Diante das fragilidades expostas pela configuração manual dos dispositivos, a organização GÉANT, responsável pela coordenação técnica do eduroam na Europa e em outras regiões, desenvolveu o eduroam Configuration Assistant Tool (CAT). Trata-se de uma plataforma de serviço que visa garantir a segurança e a facilidade na configuração do acesso à rede para os usuários finais, mitigando os erros humanos que frequentemente levam a vulnerabilidades de segurança (GÉANT, 2023).

O funcionamento do CAT baseia-se em uma arquitetura dividida entre a administração institucional e o usuário final. Primeiramente, os administradores de rede das instituições de origem (Identity Providers - IdP) acessam a plataforma administrativa do CAT e inserem as configurações de segurança específicas de sua instituição. Isso inclui o upload do certificado da Autoridade Certificadora (CA) raiz, a definição dos nomes de servidor (CN) autorizados a realizar a autenticação RADIUS e a escolha dos métodos EAP suportados (como EAP-TTLS ou PEAP).

A partir desses dados, o sistema CAT gera instaladores personalizados e perfis de configuração específicos para os diversos sistemas operacionais suportados, incluindo Windows, macOS, Linux, iOS e Android. O objetivo central é automatizar o processo de provisionamento do suplicante 802.1X (GÉANT, 2023).

Para o usuário final, o processo é simplificado: ele acessa o portal de download do CAT, seleciona sua instituição de origem e baixa o instalador adequado ao seu dispositivo. Ao executar este instalador, o dispositivo é configurado automaticamente com parâmetros rigorosos de segurança que, manualmente, seriam complexos para um usuário leigo definir. As principais configurações aplicadas incluem:

- Instalação do Certificado CA: O certificado da CA raiz da instituição é instalado no repositório de confiança do dispositivo, especificamente para o perfil Wi-Fi do eduroam;
- Validação do Nome do Servidor: O perfil é configurado para exigir que o servidor de autenticação apresente um certificado válido e que o Common Name (CN) ou Subject Alternative Name (SAN) corresponda exatamente aos domínios definidos pela instituição (ex: radius.ufsc.br).
- Definição do Método EAP: O método de autenticação e a fase interna (inner method) são fixados, impedindo ataques de downgrade para métodos menos seguros (GÉANT, 2023).

A importância do eduroam CAT reside na sua capacidade de prevenir ataques de Evil Twin e Man-in-the-Middle. Conforme discutido anteriormente, a maioria desses ataques obtém sucesso porque o dispositivo do usuário não sabe distinguir entre o servidor RADIUS legítimo e um servidor falso montado pelo atacante. Ao utilizar o instalador do CAT, o dispositivo é instruído tecnicamente a rejeitar qualquer conexão onde o certificado do servidor não seja assinado pela CA correta ou não possua o nome de domínio esperado, abortando a conexão antes que as credenciais do usuário (hash ou texto claro) sejam enviadas (). Sendo um componente crítico da arquitetura de segurança do eduroam, servindo como a principal defesa contra a exploração de configurações incorretas no lado do cliente.

2.5. VULNERABILIDADES EXPLORADAS

Essa subseção foca na segurança da rede eduroam em específico, que é um serviço global de Wi-Fi federado que permite acesso seguro à internet e faz o uso da autenticação no padrão IEEE 802.1X. Apesar de utilizar métodos robustos como EAP-TTLS e PEAP, configurações incorretas, documentação incompleta e limitações das implementações nos dispositivos dos clientes podem tornar os usuários suscetíveis a ataques.

Foi realizada a revisão de artigos publicados que descrevem diversos tipos de ataques e suas chances de sucesso com base em uma amostragem real. Ambos os artigos relatam a realização dos ataques *evil twin*, e *Man-in-the-Middle* (MITM). Os ataques foram realizados em conjunto, e seguem a seguinte estratégia: 1) realiza-se o ataque *evil twin*, que consiste na criação de uma nova rede Wi-Fi com o mesmo SSID de uma rede Wi-Fi íntegra, nesse caso a rede “eduroam”; 2) Após a criação dessa rede, o intuito é fazer os dispositivos dos usuários se conectem a ela, pois como o SSID da rede é o mesmo o dispositivo vai tentar se conectar a rede que possuir uma maior intensidade de sinal. Explorando uma falha na configuração dos dispositivos dos usuários, a intenção é que o cliente não faça a validação correta do certificado que é fornecido pelo servidor RADIUS para a autenticação.

Essa falha de validação pode ocorrer principalmente por dois motivos documentados na literatura a primeira sendo por configurações padrão inseguras, nas quais o dispositivo aceita qualquer certificado sem alertar o usuário, e a segunda é através dos alertas exibidos de forma pouco clara, levando o usuário a aprovar um certificado inválido por não compreender seu significado.

Esse comportamento foi observado tanto por Franken et al. (2015), ao documentar dispositivos que aceitavam automaticamente certificados maliciosos. Enquanto por Balduzzi et al. (2018), foi documentado que muitos usuários autorizavam certificados inválidos sem perceber o risco envolvido.

O outro ataque é o *Man-in-the-Middle*, que basicamente consiste na captura dos dados enviados do dispositivo que está tentando se conectar ao AP malicioso. Durante esse processo de autenticação, por mais que o método EAP utilizado seja seguro, o túnel de conexão vai ser aberto entre o cliente e o atacante, pois o dispositivo ou o usuário aceitou o certificado apresentado, então ele interpreta como se fosse um servidor RADIUS verídico. Em seguida como parte da etapa de autenticação as credenciais do usuário são enviadas para a validação, mas ao invés de serem enviadas para o servidor RADIUS o atacante recebe essas credenciais, concluindo o ataque com sucesso, na qual ele obteve dados sensíveis do usuário.

Nos artigos de Franken et al. (2015) e Balduzzi et al. (2018) a principal falha de segurança ocorreu pela parte dos clientes, seja elas devido a configurações inadequados nos dispositivos dos clientes, orientações desatualizadas pela parte das universidades ou pela impossibilidade técnica de ativar verificações e validações mais seguras dos certificados e CAs como o *Common Name (CN) check* por exemplo.

No artigo de Franken et al. (2015) foram testados 507 dispositivos únicos com aproximadamente 350 usuários. Os dispositivos foram contados através dos endereços MAC, e desses 507 dispositivos, 52 % deles se mostraram vulneráveis ao ataque, ou seja, algo em torno de 264 dispositivos. Dentre esses dispositivos vulneráveis, 20 % usavam o método PAP como método interno de autenticação, o que resultou no vazamento das credenciais em texto não cifrado. Como afirmam os autores:

From a total of 507 devices, 52 % devices were vulnerable. A total 20% of the vulnerable devices used PAP as inner authentication method and thus were leaking authentication data in unencrypted form.” (Franken et al., 2015, p. 5).

Os autores também analisaram que os dispositivos Apple, que utilizavam os métodos EAP-TTLS/MS-CHAPv2, também eram vulneráveis, porém em uma escala menor. Dos 507 dispositivos analisados, 38 % deles eram da marca Apple e, dentre esses, 13 % deles estavam incorretamente configurados e vulneráveis. Com base nesses dados, também podemos supor que os demais dispositivos vulneráveis, cerca de 230 dispositivos, representam em sua maioria dispositivos com sistemas Android. Como afirmam os autores:

Apple devices were also vulnerable when using EAP-TTLS/MS-CHAPv2; thus, their share in the total number of devices is listed separately in Table 2. [...] Of all devices, 38 % were Apple devices, and of those 13 % were misconfigured (Franken et al., 2015, p. 5).

Já no artigo de Balduzzi et al, (2018), também foram realizados e documentados os mesmos ataques. O estudo contou com a participação de 37 usuários e 33 dispositivos que possuíam um conjunto de marca, modelo, sistema operacional e configuração de rede diferentes entre si. Foi obtido uma taxa de sucesso com os ataques de *evil twin* e *Man-in-the-Middle* em 37% dos usuários.

Mesmo sem os usuários realizarem qualquer tipo de interação nos dispositivos, ou alteração prévia nas configurações de rede, de forma passiva, foi possível roubar as credenciais de mais de um terço dos participantes. Como afirmam os autores:

Without any user interaction and even when the smartphones were in the users' pockets (i.e., with a fully passive attack), we were able to steal credentials from more than a third of the participants (Balduzzi et al., 2018, p. 470).

Ou seja, somente de estar perto de um AP malicioso, o dispositivo do usuário tentou se conectar automaticamente na rede Wi-Fi conhecida, porém nesse caso, se tratava de uma rede maliciosa, que estava utilizando o mesmo SSID da rede original. Com isso, foi possível obter as credenciais dos usuários sem levantar suspeitas sobre a veracidade e confiabilidade da rede.

Nesse mesmo artigo, também foi relatado uma taxa maior de sucesso no roubo das credenciais em dispositivos Android, evidenciando um problema mais grave dependendo do sistema operacional. Dos 9 dispositivos Apple presentes no experimento, nenhum desses dispositivos foi afetado pelo ataque, pois quando a tentativa de conexão automática com a rede maliciosa era iniciada o sistema *iPhone Operating System* (IOS), barrava a conexão, pois o certificado apresentado não passava na validação, fazendo com que o dispositivo nem chegasse a realizar o envio das credenciais do usuário ao atacante. Como afirmam os autores:

IOS never connects to an Evil Twin network if the server certificate is properly set in the Eduroam profile." (Balduzzi et al., 2018, p. 475).

Além de barrar a conexão com o AP malicioso devido a falha na validação do certificado do servidor, os autores também relataram que os dispositivos da Apple exibem um claro aviso de que não foi possível validar corretamente o certificado, alertando os usuários sobre um possível problema naquela rede, conforme descrito no trecho:

When the user tries to connect to a network whose certificate is not trusted, iOS shows a clear warning message." (Balduzzi et al., 2018, p. 475).

Essas vulnerabilidades relacionadas ao sistema operacional do dispositivo é fundamentada pelo sistema IOS da Apple ser um sistema mais fechado, e não permitir que o usuário altere as políticas de validação de certificados de rede. Enquanto o Android historicamente é mais aberto e flexível, permitindo customização de praticamente tudo, podendo ter suas configurações modificadas tanto pelo usuário como pela própria fabricante do dispositivo. E com isso se abre margem para os usuários que não possuem conhecimento suficiente configurarem as políticas de

validação de certificados de rede de forma insegura, além de em alguns casos também ser possível escolher o método EAP, o que pode acabar resultando na transmissão das credenciais do usuário no formato de texto puro, resultando em uma vulnerabilidade ainda mais grave

3. DESENVOLVIMENTO

Na etapa experimental foi utilizado um ambiente de testes controlado que reproduzia aspectos da rede eduroam. Foram realizados experimentos práticos de ataque e também avaliado se as vulnerabilidades identificadas na pesquisa bibliográfica estavam presentes no nosso ambiente de teste. O foco foi em explorar falhas de configuração dos clientes e da infraestrutura que comprometem a segurança durante a autenticação 802.1X do eduroam. Foram simulados principalmente ataques do tipo *evil twin* e técnicas de *man-in-the-middle* para capturar as credenciais. A partir desses experimentos, foi possível confirmar quais cenários representam uma ameaça aos usuários e propor medidas de mitigação e recomendações de configuração para tornar a rede mais segura.

3.1 FERRAMENTAS UTILIZADAS

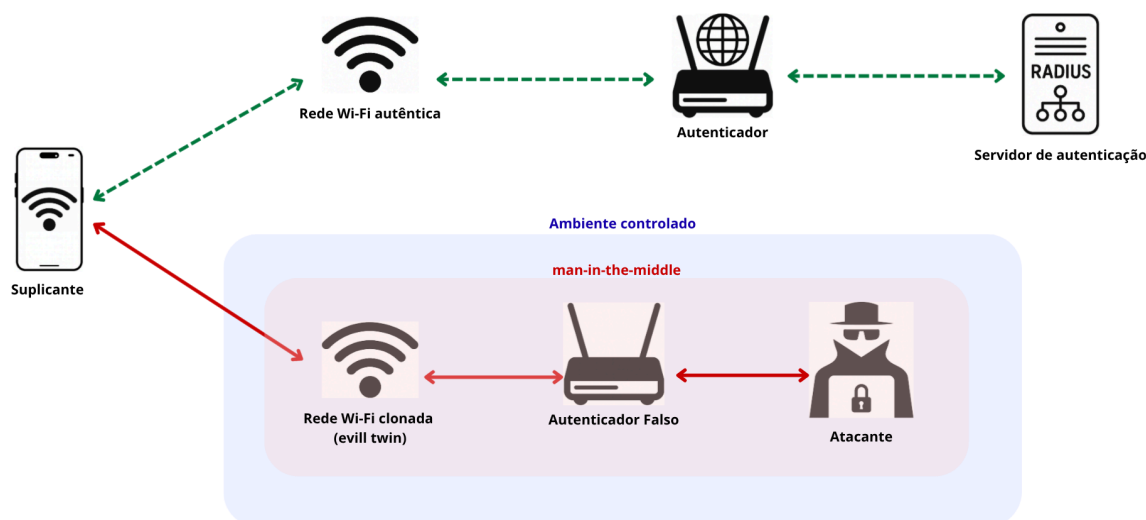
Para construir o ambiente de experimentação e conduzir os testes de invasão, foram utilizados as seguintes ferramentas e tecnologias:

- **Sistema Operacional de Testes:** *Kali Linux*, por incluir nativamente diversas ferramentas de auditoria de redes sem fio;
- **Servidor RADIUS:** *FreeRADIUS*, configurado como um servidor RADIUS malicioso. Essa versão permite registrar ou extrair credenciais de autenticação capturadas como, hashes MS-CHAPv2 ou senhas em claro de métodos PAP, demonstrando vulnerabilidades de *impersonation* (fraudes cibernéticas onde um criminoso se passa por uma pessoa ou entidade confiável) do servidor RADIUS;
- **Ponto de Acesso Falso:** Roteador físico utilizado como *Access Point* e com a configuração de segurança do Wi-Fi no modo *WPA2-Enterprise*, simulando um AP legítimo eduroam. Esse AP falso anunciou o SSID eduroam e utilizou o servidor RADIUS malicioso para autenticação;
- **Analizador de Tráfego:** *Wireshark* foi empregado para monitorar o tráfego das autenticações EAPOL/RADIUS no ambiente de teste. Auxiliando na visualização do *handshake* 802.1X/EAP e verificando, por exemplo, se informações sensíveis como, identidade externa do usuário trafegavam em texto claro;
- **Dispositivos Clientes de Teste:** Foram utilizados alguns dispositivos pessoais reais, configurados previamente para acesso ao eduroam legítimo (um *smartphone* Android, dois iPhone, um *tablet* Android, e um Macbook). Esses dispositivos atuaram como **suplicantes** nos experimentos, permitindo avaliar diferenças no comportamento e vulnerabilidades em diferentes sistemas operacionais.

3.2 ILUSTRAÇÃO DO AMBIENTE DE EXPERIMENTAÇÃO

O ambiente de testes foi montado de forma a isolar a rede experimental e não interferir na rede eduroam real em produção. A Figura 6 ilustra a arquitetura utilizada na realização do experimento:

Figura 6: Representação do ambiente utilizado, com os ataques evil twin e man in the middle em conjunto.



Fonte: Autor (2025).

3.3. CONFIGURAÇÃO DO AMBIENTE

O ambiente experimental foi configurado para reproduzir, de forma controlada, as condições necessárias aos ataques *evil twin* e *man-in-the-middle* durante o processo de autenticação 802.1X. A Figura 6 apresenta a topologia geral utilizada, composta por três elementos principais: o servidor RADIUS malicioso, o *Access Point* configurado como autenticador WPA2-Enterprise e os dispositivos clientes previamente configurados para o eduroam. Todos os componentes foram isolados da rede real, garantindo que nenhum tráfego legítimo fosse interceptado ou interferido.

Configuração do Wireshark: Para conseguir avaliar se foi possível capturar dados sensíveis do usuário como as credenciais, foi instalado o wireshark seguindo a instalação padrão na máquina hospedeira da VM, foi necessário somente selecionar a placa de rede a qual o *Access Point* estava conectado para analisar todos os pacotes que por ali passaram;

Configuração da máquina virtual: Para permitir a troca direta de mensagens entre o *Access Point* e o servidor RADIUS, a interface de rede da máquina virtual foi ajustada para operar em modo *bridge*, vinculada à mesma interface física conectada ao AP. Esse ajuste assegura que o FreeRADIUS receba um endereço IP na mesma sub-rede do autenticador, requisito indispensável para o roteamento das mensagens EAP encapsuladas em RADIUS.

Configuração do servidor RADIUS (FreeRADIUS): O servidor de autenticação foi instalado em uma máquina virtual executando Kali Linux, devido ao suporte nativo a ferramentas de auditoria de redes sem fio. O FreeRADIUS foi configurado para atuar como servidor RADIUS malicioso, aceitando qualquer credencial enviada e registrando os dados recebidos durante o *handshake* EAP. Os principais parâmetros ajustados foram: definição do AP como cliente autorizado, inclusão do endereço IP do autenticador e a criação de um

segredo compartilhado para comunicação. Também foram configurados certificados digitais auto-assinados, simulando um cenário em que o cliente não valida corretamente a identidade do servidor. Métodos internos distintos foram ativados (PAP e MS-CHAPv2) para avaliar diferenças no comportamento dos dispositivos diante de configurações inseguras, em alinhamento com resultados documentados na literatura.

Autenticador (*Access Point*): Um roteador físico foi configurado exclusivamente para operar como *Access Point*, anunciando o SSID “eduroam” e utilizando como segurança Wi-Fi o WPA2-*Enterprise*. Na interface administrativa do dispositivo, o modo de operação foi alterado para *Access Point*, a rede Wi-Fi foi habilitada e, na seção de segurança, foi selecionada a opção WPA2-*Enterprise*, inserindo-se o mesmo segredo compartilhado definido no FreeRADIUS. Dessa forma, o AP encaminha todas as mensagens EAP para o servidor malicioso, completando a simulação de um ponto de acesso falso com aparência legítima para os dispositivos clientes.

Configuração dos dispositivos clientes: Foram utilizados dispositivos reais previamente configurados para acesso ao eduroam verídico, incluindo *smartphones* Android, iPhones, um *tablet* Android e um MacBook. Cada dispositivo possuía o perfil eduroam instituído por sua instituição de origem. Alguns deles foram propositalmente configurados de forma insegura, como ausência de validação do certificado do servidor ou aceitação automática de certificados desconhecidos, para reproduzir cenários descritos na literatura, nos quais dispositivos vulneráveis se conectam automaticamente a pontos de acesso maliciosos. O comportamento de cada sistema operacional diante dessas variações foi observado e registrado.

Essa configuração permitiu reproduzir fielmente os cenários necessários para execução dos ataques simulados, preservando a replicabilidade do experimento e garantindo o isolamento completo em relação à infraestrutura eduroam real.

3.4. PROCEDIMENTOS

A preparação do ambiente experimental iniciou-se com a criação de uma máquina virtual no VirtualBox, na qual foi instalado o sistema operacional Kali Linux utilizando as configurações padrão sugeridas pelo instalador. Após a instalação, procedeu-se à configuração do servidor RADIUS malicioso utilizando o FreeRADIUS, instalado via gerenciador de pacotes.

No FreeRADIUS, foram modificados apenas os arquivos necessários para viabilizar a simulação dos ataques. No arquivo *clients.conf*, foi incluído o endereço IP do *Access Point* utilizado como autenticador, juntamente com a chave secreta que permitiria a comunicação entre ambos. Em *authorize*, foi inserido um conjunto mínimo de credenciais genéricas exclusivamente para fins de teste, uma vez que, em ambientes reais, o armazenamento de usuários não ocorre no próprio RADIUS, mas em diretórios externos ou bancos de dados dedicados. Já no arquivo *eap*, foram habilitados métodos EAP adicionais e definido o método padrão utilizado durante a autenticação.

Na máquina hospedeira (Windows), instalou-se o Wireshark com suas configurações padrão, selecionando a interface de rede que estava conectada ao *Access Point*. Essa ferramenta foi utilizada para monitorar o tráfego entre autenticador e servidor RADIUS,

permitindo a observação das trocas EAP e dos encapsulamentos RADIUS durante o processo de autenticação.

O *Access Point* foi configurado em modo *WPA2-Enterprise*, informando o endereço IP do servidor RADIUS e a chave secreta definida no arquivo *clients.conf*. Para garantir a comunicação adequada, os endereços IP do AP e da máquina virtual foram ajustados para pertencer à mesma sub-rede, assegurando roteamento direto entre autenticador e servidor.

Para iniciar o experimento, o FreeRADIUS foi executado em modo de depuração utilizando o comando:

```
$ sudo freeradius -X
```

Esse modo permite visualizar em tempo real as mensagens EAP e os pacotes RADIUS recebidos. Com o servidor ativo, o *Access Point* já passa a anunciar o SSID “eduroam”, simulando o ponto de acesso malicioso.

Os dispositivos clientes, que já possuíam o perfil eduroam previamente configurado, tentaram se conectar automaticamente ao AP falso. Essa tentativa desencadeou o envio das credenciais de autenticação para o servidor RADIUS malicioso, permitindo a captura e análise do tráfego no Wireshark. No entanto, observou-se que nem todos os dispositivos concluíram a conexão automaticamente: alguns sistemas exibiram alertas de certificado inválido, impedindo ou dificultando a associação ao AP falso, comportamento já previsto pela literatura.

3.5. EVIDÊNCIAS COLETADAS

A etapa de coleta de evidências teve como objetivo examinar, de forma sistemática, potenciais vulnerabilidades em redes que utilizam a combinação de RADIUS e métodos EAP para autenticação de usuários. Partindo dessa premissa, foram conduzidos diferentes experimentos voltados à análise prática da segurança desses protocolos, avaliando tanto sua robustez criptográfica quanto seu comportamento diante de cenários adversos.

Inicialmente, foram testados os métodos EAP previamente descritos no referencial teórico, com o propósito de observar, na prática, o fluxo de autenticação, a exposição de atributos sensíveis e a forma como cada método se comporta em relação à captura de pacotes e análise via FreeRADIUS e Wireshark. Na sequência, foi estruturado o experimento principal do estudo, voltado a verificar se um ataque *man-in-the-middle* viabilizado por um ponto de acesso malicioso (*evil twin*) poderia representar riscos concretos aos usuários de arquiteturas baseadas em 802.1X, como o eduroam.

3.5.1 EAP MD5

Nos testes realizados com o método EAP-MD5, foi possível observar diferenças significativas no comportamento dos sistemas operacionais diante do certificado auto-assinado apresentado pelo servidor RADIUS. Em dispositivos com Windows 10 e iOS 26.0.1, o sistema exibiu notificações de segurança alertando o usuário sobre a impossibilidade de validar o certificado do servidor, solicitando confirmação manual para prosseguir com a conexão conforme as Figuras 7 e 10.

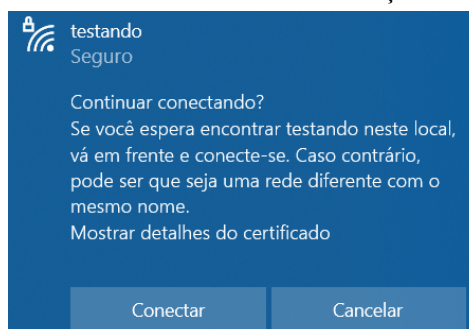
Em contraste, dispositivos Android permitiram a conexão sem qualquer aviso ao usuário. Outro aspecto relevante observado durante os experimentos é que, embora o método tenha sido executado dentro de um túnel TLS (PEAP-MD5), o mecanismo interno de autenticação permanece baseado no desafio-resposta MD5. Assim, o túnel TLS efetivamente impede que observadores externos na rede capturem o tráfego EAP-MD5, garantindo que o *Challenge* e o *Response* não fiquem visíveis em uma simples análise de pacotes via Wireshark.

Entretanto, se o cliente não valida o certificado do servidor, torna-se possível estabelecer um ponto de acesso malicioso (*evil twin*) que encerra o túnel TLS que pode ser analisado na Figura 8. Nesse cenário, o atacante opera o servidor RADIUS falso e, portanto, obtém acesso direto ao conteúdo descriptografado do método interno (Figura 12). Isso inclui o *hash* derivado da senha produzido pelo EAP-MD5:

MD5-Response = MD5(ID || senha || challenge)

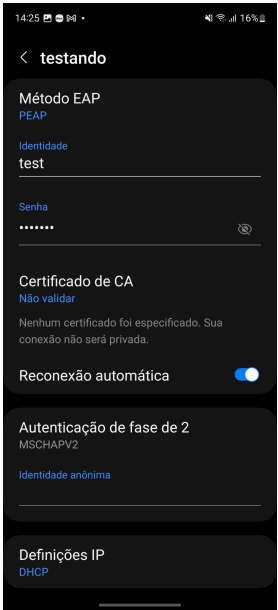
Embora a senha não seja transmitida em texto claro, esse valor pode ser facilmente coletado nos *logs* do servidor falso e submetido a ataques *offline* de força bruta, permitindo a recuperação da senha original. Assim, mesmo com a presença do túnel TLS, a segurança do método depende integralmente da validação rígida do certificado por parte do cliente, caso contrário, o túnel será estabelecido com o servidor errado e o atacante terá acesso ao material sensível do método interno.

Figura 7: Alerta do Windows sobre falha na validação do certificado MD5.



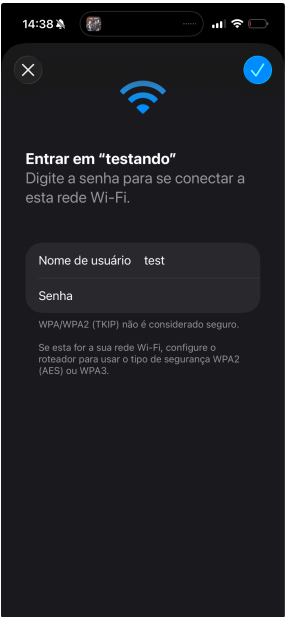
Fonte: Autor (2025).

Figura 8: Tela de configuração e conexão do Android MD5.



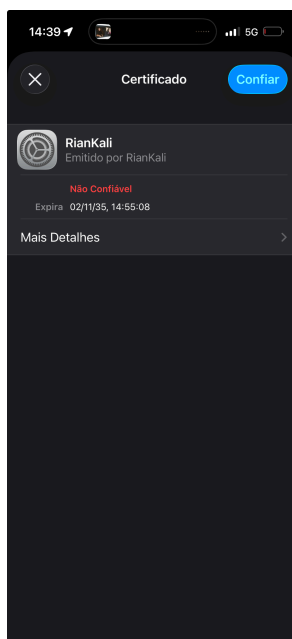
Fonte: Autor (2025).

Figura 9: Tela de configuração e conexão no IOS MD5.



Fonte: Autor (2025).

Figura 10: notificação do iPhone sobre certificado não confiável MD5.



Fonte: Autor (2025).

Figura 11: Captura do tráfego no Wireshark durante o procedimento de conexão MD5.

1.. 90.928688	192.168.56.189	192.168.56.100	RADIUS	170 Response, Identity
1.. 90.933747	192.168.56.189	192.168.56.100	RADIUS	172 Request, Protected EAP (EAP-MD5-CHALLENGE)
2.. 90.935339	192.168.56.100	192.168.56.189	RADIUS	185 Response, Legacy Nak (Response Only)
2.. 90.942304	192.168.56.189	192.168.56.100	RADIUS	106 Request, Protected EAP (EAP-PEAP)
2.. 90.944622	192.168.56.100	192.168.56.189	RADIUS	320 Client Hello
2.. 90.956883	192.168.56.189	192.168.56.100	RADIUS	1110 Request, Protected EAP (EAP-PEAP)
2.. 90.958123	192.168.56.100	192.168.56.189	RADIUS	185 Response, Protected EAP (EAP-PEAP)
2.. 90.967241	192.168.56.189	192.168.56.100	RADIUS	266 Server Hello, Certificate, Server Key Exchange, Server Hello Done
2.. 90.968810	192.168.56.100	192.168.56.189	RADIUS	306 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2.. 90.973253	192.168.56.189	192.168.56.100	RADIUS	181 Change Cipher Spec, Encrypted Handshake Message
2.. 90.974918	192.168.56.100	192.168.56.189	RADIUS	185 Response, Protected EAP (EAP-PEAP)
2.. 90.979469	192.168.56.189	192.168.56.100	RADIUS	159 Application Data
2.. 90.981398	192.168.56.100	192.168.56.189	RADIUS	238 Application Data
2.. 90.990569	192.168.56.189	192.168.56.100	RADIUS	191 Application Data
2.. 90.999651	192.168.56.100	192.168.56.189	RADIUS	286 Application Data
2.. 91.004412	192.168.56.189	192.168.56.100	RADIUS	207 Application Data
2.. 91.006215	192.168.56.100	192.168.56.189	RADIUS	238 Application Data
2.. 91.011411	192.168.56.189	192.168.56.100	RADIUS	159 Application Data
2.. 91.012917	192.168.56.100	192.168.56.189	RADIUS	238 Application Data
				214 Success

Frame 198: Packet, 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface \Dev	0000	cc 32 e5 23 86 32 08 00	27 7f 43 bf 08 00 45 00	..2 # 2.. 'C' E
Ethernet II, Src: PCSSystemtec_7f:43:bf (08:00:27:7f:43:bf), Dst: TplinkTechno_23:86:32 (cc:32:e	0010	00 6c 4c a3 00 00 40 11	3b 6c c0 a8 38 64 c0 a8	1l...@ ;l 8d
Internet Protocol Version 4, Src: 192.168.56.180, Dst: 192.168.56.189	0020	38 bd 07 14 d3 cb 00 58	ae 65 0b 0b 00 50 49 47	8...X e k PIG
User Datagram Protocol, Src Port: 1012, Dst Port: 54219	0030	e2 da 84 02 88 fd 5b f7	67 5e 51 ee 09 66 50 12	b...[g'Q' fP
RADIUS Protocol	0040	ed 6f 6c c8 b9 ee de f3	31 be 86 22 17 e7 87 33	ol... 1 " 3
	0050	4f 18 01 02 00 16 04 10	07 fa e0 9d a6 b9 a6 ea	0.....
	0060	99 33 fb 6f af 88 dc d6	18 12 c1 d3 8c 8e c1 d1	3 o.....
	0070	88 e0 84 84 8e d5 8f 7f	85 f6	

Fonte: Autor (2025).

3.5.2 EAP-TLS

O método EAP-TLS é amplamente reconhecido como o mecanismo de autenticação mais robusto dentro do conjunto de protocolos EAP, uma vez que estabelece autenticação mútua por meio de certificados digitais e elimina completamente a dependência de senhas. Entretanto, durante a execução dos experimentos deste trabalho, não foi possível concluir os testes práticos referentes a esse método devido a limitações no processo de envio do certificado de cliente para os dispositivos utilizados.

A configuração do EAP-TLS exige que cada equipamento possua um certificado digital próprio, acompanhado de sua respectiva chave privada, ambos emitidos ou assinados por uma autoridade certificadora confiável pelo servidor RADIUS. Apesar de o FreeRADIUS ter sido corretamente configurado com a infraestrutura de certificação (CA, certificados do servidor e arquivos PKCS#12 gerados para os clientes), o problema ocorreu durante a

transferência do certificado do cliente para o dispositivo que iria atuar como suplicante. Como resultado, o certificado de cliente não pôde ser instalado e o método EAP-TLS não pôde ser testado em ambiente físico.

3.5.3 EAP-TTLS

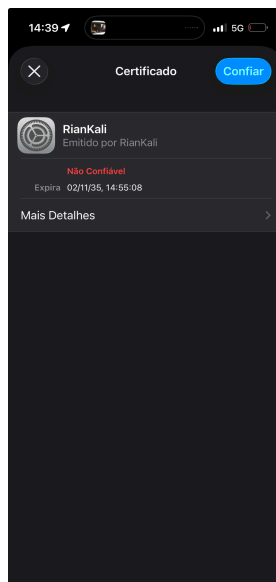
Nos testes realizados com o método EAP-TTLS utilizando o *inner method* PAP, observou-se que, em condições normais de operação, as credenciais do usuário não podem ser visualizadas na captura de tráfego (Wireshark) conforme é possível analisar na Figura 16. Isso ocorre porque todo o conteúdo interno da autenticação (incluindo o envio da senha) é encapsulado no túnel TLS estabelecido entre o cliente e o servidor RADIUS, impedindo que um observador externo acesse qualquer dado sensível durante a transmissão.

Entretanto, o comportamento do método muda completamente quando o cliente se conecta a um servidor malicioso, cenário possibilitado pela ausência de validação do certificado do servidor. Nesse caso, o túnel TLS passa a ser estabelecido diretamente com o atacante, que opera como *endpoint* da sessão. Como resultado, o tráfego interno do TTLS deixa de estar protegido contra o adversário, permitindo que o *inner* PAP seja processado em texto claro, mostrado na Figura 17. Nessas condições, o servidor falso recebe e registra explicitamente atributos como:

User-Name = "usuario"

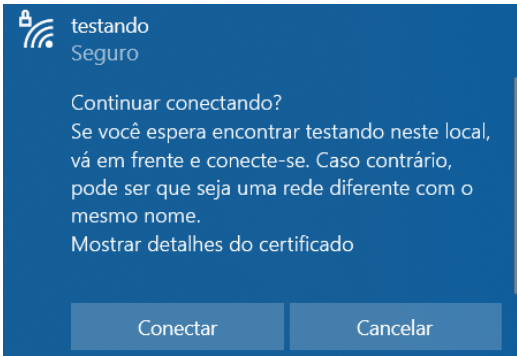
User-Password = "senha"

Figura 13: notificação do iPhone sobre certificado não confiável TTLS.



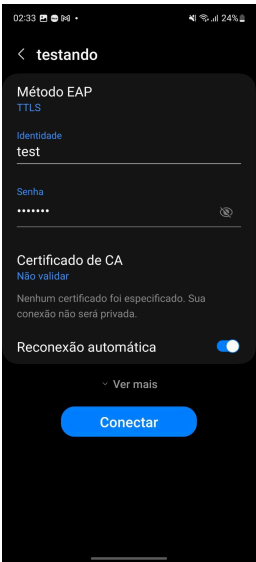
Fonte: Autor (2025).

Figura 14: Notificação do Windows sobre falha na validação do certificado TTLS.



Fonte: Autor (2025).

Figura 15: Tela de configuração da conexão no Android TTLS.



Fonte: Autor (2025).

Figura 16: Captura do tráfego no Wireshark durante o procedimento de conexão TTLS.

2.. 237.922245	192.168.56.189	192.168.56.100	RADIUS	170 Response, Identity
2.. 237.923170	192.168.56.100	192.168.56.189	RADIUS	106 Request, Tunneled TLS EAP (EAP-TTLS)
2.. 237.930948	192.168.56.189	192.168.56.100	RADIUS	316 Client Hello
2.. 237.934352	192.168.56.100	192.168.56.189	RADIUS	1106 Request, Tunneled TLS EAP (EAP-TTLS)
2.. 237.946150	192.168.56.189	192.168.56.100	RADIUS	185 Response, Tunneled TLS EAP (EAP-TTLS)
2.. 237.947641	192.168.56.100	192.168.56.189	RADIUS	274 Server Hello, Certificate, Server Key Exchange, Server Hello Done
2.. 237.965971	192.168.56.189	192.168.56.100	RADIUS	302 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
2.. 237.969288	192.168.56.100	192.168.56.189	RADIUS	185 Change Cipher Spec, Encrypted Handshake Message
2.. 238.008128	192.168.56.189	192.168.56.100	RADIUS	270 Application Data
2.. 238.010038	192.168.56.100	192.168.56.189	RADIUS	214 Success

Fonte: Autor (2025).

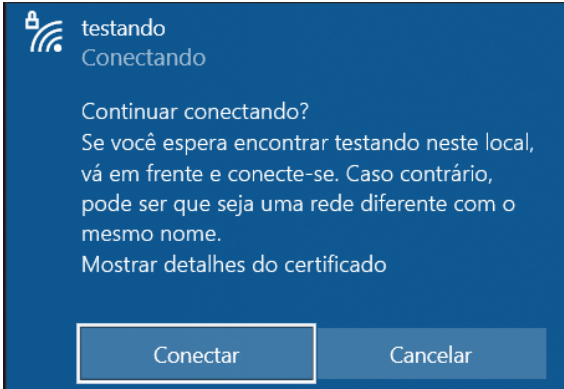
3.5.4 EAP-PEAP

Nos experimentos realizados com o método EAP-PEAP utilizando MSCHAPv2 como mecanismo interno de autenticação, verificou-se que nenhum dado sensível é exposto durante a captura de pacotes conforme Figuras 19 e 20. Assim como no EAP-TTLS, o PEAP estabelece inicialmente um túnel TLS entre o cliente e o servidor RADIUS, garantindo que todo o conteúdo da autenticação interna permaneça cifrado. Por esse motivo, ferramentas

como Wireshark exibem apenas pacotes genéricos do tipo *Protected EAP*, sem revelar o desafio ou a resposta associados ao MSCHAPv2.

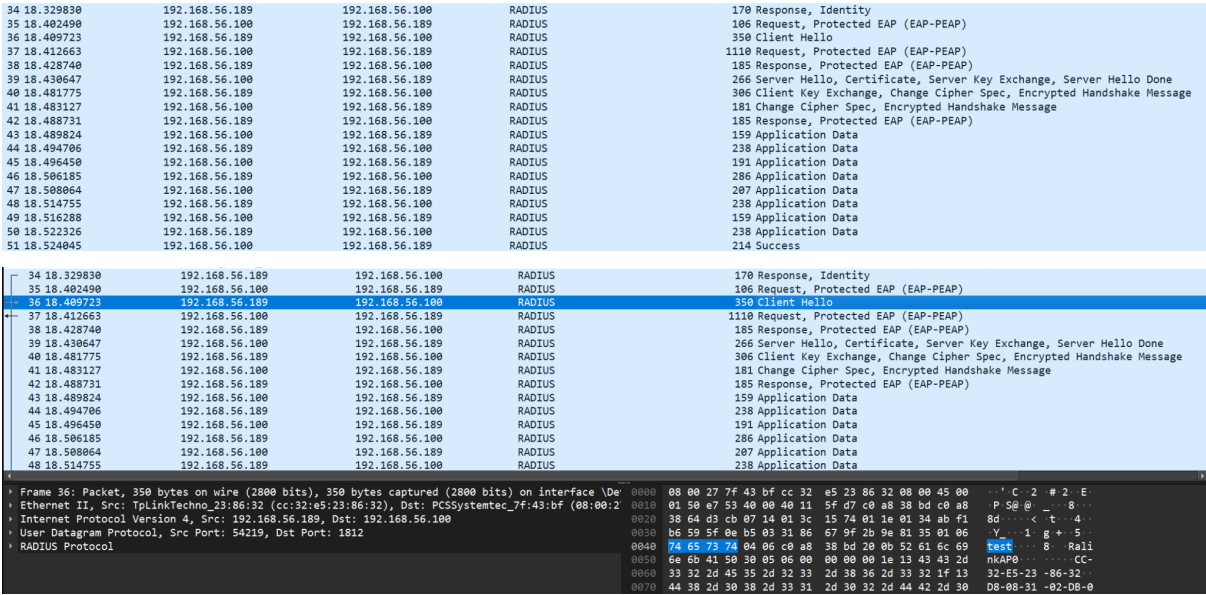
Além disso, mesmo quando o servidor falso opera como *endpoint* do túnel, situação que ocorre quando o cliente não valida o certificado do servidor, o método MSCHAPv2 não exibe a senha em texto claro como é possível visualizar na Figura 25. Em vez disso, o cliente envia um par *Challenge–Response* baseado no *hash* NTLM da senha. O servidor legítimo ou malicioso realiza a verificação comparando o *NT-Response* com o valor esperado, mas a senha nunca é transmitida nem armazenada no tráfego TLS descryptografado.

Figura 18: Notificação do Windows sobre falha na validação do certificado PEAP.



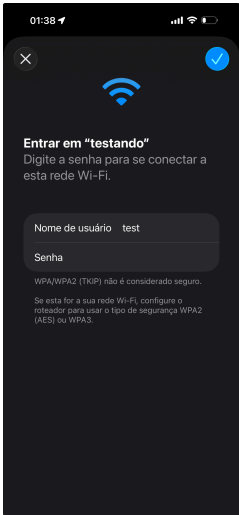
Fonte: Autor (2025).

Figuras 19 e 20: Captura do tráfego EAP-PEAP através do Wireshark.



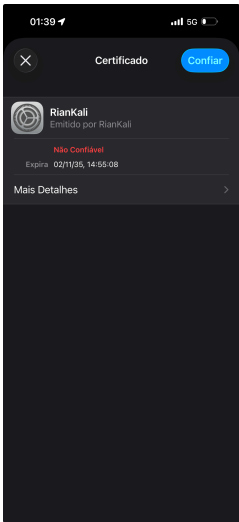
Fonte: Autor (2025).

Figura 21: tela de conexão do IOS.



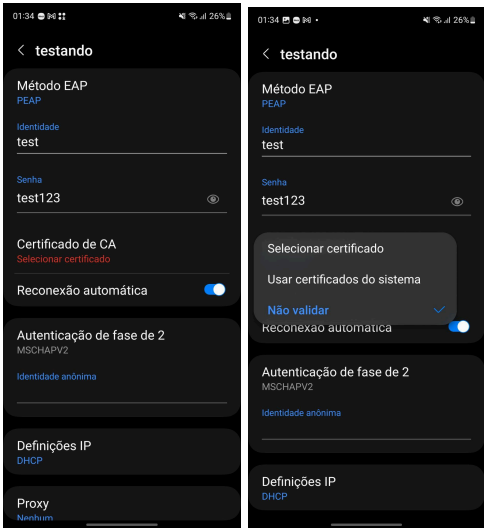
Fonte: Autor (2025).

Figura 22: notificação do iPhone sobre certificado não confiável PEAP.



Fonte: Autor (2025).

Figura 23 e 24: Tela de conexão e configuração no android selecionando o método PEAP e “não verificar certificado” respectivamente.



Fonte: Autor (2025).

3.5.5 EAP-GTC

Nos testes envolvendo o método EAP-PEAP utilizando o mecanismo interno *Generic Token Card* (GTC), observou-se que, assim como em outros métodos baseados em PEAP, todo o tráfego interno é encapsulado em um túnel TLS estabelecido entre o cliente e o servidor RADIUS. Consequentemente, a análise via Wireshark não revela qualquer informação sensível conforme Figura 28. O comportamento mais relevante surge quando o cliente não valida o certificado do servidor. Nessa condição, o atacante pode configurar um servidor RADIUS malicioso e estabelecer um túnel TLS plenamente funcional com o dispositivo da vítima. Como o *inner method* GTC transmite as credenciais em texto claro dentro do túnel, o servidor falso passa a receber diretamente atributos como:

User-Name = "usuario"

User-Password = "senha"

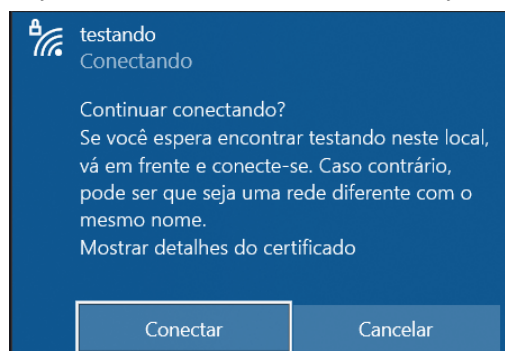
Esse resultado foi confirmado durante os experimentos em ambiente controlado utilizando um *supplicant* compatível com PEAP-GTC como pode ser visto na Figura 29. Entretanto, destaca-se que dispositivos móveis modernos, como Android e iOS, raramente oferecem suporte ao GTC como método interno de PEAP. Apesar dessa limitação prática, o experimento evidencia que o PEAP-GTC é particularmente vulnerável em cenários de ausência de validação de certificado, já que a credencial é exposta ao servidor malicioso de forma integral.

O método EAP-TTLS com *inner* GTC também apresenta comportamento semelhante ao do PEAP-GTC. No TTLS, o túnel TLS externo protege a troca de mensagens internas, impossibilitando a observação de credenciais na captura de tráfego. Assim, em um ambiente legítimo, o analisador de pacotes não revela informações além de mensagens TLS genéricas. E da mesma forma que no PEAP-GTC, quando o cliente deixa de validar o certificado do servidor, o atacante pode posicionar um servidor TTLS malicioso e estabelecer o túnel TLS diretamente com a vítima. Como o *inner method* GTC transfere a senha em texto claro, o servidor malicioso passa a registrar integralmente as credenciais recebidas:

User-Name = "usuario"

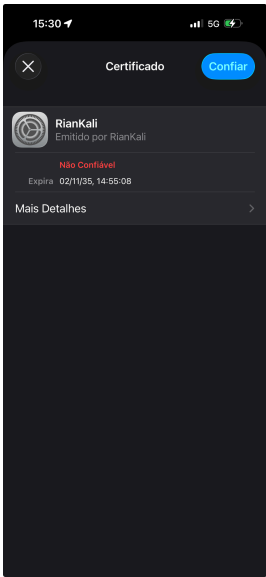
User-Password = "senha"

Figura 26: Notificação do Windows sobre falha na validação do certificado GTC.



Fonte: Autor (2025).

Figura 27: Notificação do iPhone sobre certificado não confiável GTC.



Fonte: Autor (2025).

Figura 28: Captura do tráfego com *inner* GTC através do Wireshark.

3.. 11.955637	192.168.56.189	192.168.56.100	RADIUS	170 Response, Identity
3.. 11.956334	192.168.56.100	192.168.56.189	RADIUS	115 Request, Generic Token Card EAP (EAP-GTC)
3.. 11.960851	192.168.56.189	192.168.56.100	RADIUS	186 Response, Legacy Nak (Response Only)
3.. 11.962107	192.168.56.100	192.168.56.189	RADIUS	106 Request, Protected EAP (EAP-PEAP)
3.. 11.967475	192.168.56.189	192.168.56.100	RADIUS	351 Client Hello
3.. 11.969455	192.168.56.100	192.168.56.189	RADIUS	1110 Request, Protected EAP (EAP-PEAP)
3.. 11.981529	192.168.56.189	192.168.56.100	RADIUS	185 Response, Protected EAP (EAP-PEAP)
3.. 11.982685	192.168.56.100	192.168.56.189	RADIUS	266 Server Hello, Certificate, Server Key Exchange, Server Hello Done
3.. 11.989673	192.168.56.189	192.168.56.100	RADIUS	306 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
3.. 11.990826	192.168.56.100	192.168.56.189	RADIUS	181 Change Cipher Spec, Encrypted Handshake Message
3.. 13.400794	192.168.56.189	192.168.56.100	RADIUS	185 Response, Protected EAP (EAP-PEAP)
3.. 13.401918	192.168.56.100	192.168.56.189	RADIUS	159 Application Data
3.. 13.406555	192.168.56.189	192.168.56.100	RADIUS	238 Application Data
3.. 13.407546	192.168.56.100	192.168.56.189	RADIUS	191 Application Data
3.. 13.422953	192.168.56.189	192.168.56.100	RADIUS	286 Application Data
3.. 13.424817	192.168.56.100	192.168.56.189	RADIUS	207 Application Data
3.. 13.430118	192.168.56.189	192.168.56.100	RADIUS	238 Application Data
3.. 13.431519	192.168.56.100	192.168.56.189	RADIUS	159 Application Data
4.. 13.449571	192.168.56.189	192.168.56.100	RADIUS	238 Application Data
4.. 13.453628	192.168.56.100	192.168.56.189	RADIUS	214 Success

Fonte: Autor (2025).

3.5.6 EAP-FAST

O método EAP-FAST chegou a ser configurado no servidor RADIUS, porém sua implementação não foi levada adiante, pois os dispositivos disponíveis para os experimentos não ofereciam suporte a esse tipo de autenticação. Como o FAST é adotado apenas por dispositivos e ambientes corporativos muito específicos, não havia condições técnicas e por não fazer parte das tecnologias normalmente empregadas em redes semelhantes à infraestrutura do eduroam. Não se enquadrando no escopo deste estudo.

3.5.7 Simulação de *evil twin* e *man-in-the-middle*

Também foi conduzida a simulação de um ataque *evil twin* associado a um cenário de *man-in-the-middle*, conforme proposto nos objetivos deste trabalho. Para isso, o servidor FreeRADIUS foi configurado para aceitar qualquer credencial apresentada, permitindo observar exclusivamente o comportamento dos métodos EAP frente a um servidor malicioso. Os métodos utilizados replicaram a configuração da rede eduroam da UFSC, cuja autenticação é baseada em PEAP com *inner* MSCHAPv2, informação verificada a partir da análise dos parâmetros de conexão da própria rede institucional.

Como discutido anteriormente, o PEAP-MSCHAPv2 não expõe a senha do usuário em momento algum do processo de autenticação, mesmo quando a conexão é estabelecida com um servidor falso. Nesses casos, o atacante tem acesso apenas ao desafio e à resposta derivados do MSCHAPv2, que podem ser submetidos a ataques *offline*, mas não revelam a senha diretamente.

Diante disso, e com o objetivo de demonstrar de forma prática o impacto de uma configuração insegura, buscou-se induzir um *downgrade* no método interno utilizado pelo cliente, tentando forçá-lo a substituir o MSCHAPv2 pelo método PEAP-PAP, que envia a senha em texto claro dentro do túnel TLS. Essa estratégia permitiria evidenciar como a ausência de validação do certificado do servidor abre espaço para que um atacante controle o *endpoint* TLS e visualize diretamente as credenciais do usuário. Contudo, como será discutido na subseção seguinte, dispositivos modernos não realizam esse *downgrade*, permanecendo restritos ao MSCHAPv2 independentemente da configuração do servidor malicioso.

3.5.7.1 Celular Android

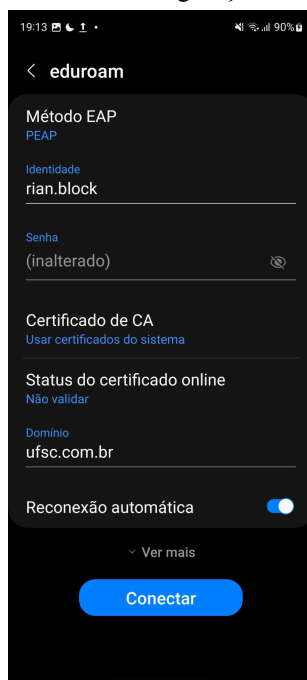
Nos testes realizados com dispositivo Android de API 33, observou-se que o cliente não realizou o *downgrade* do método interno de autenticação. Mesmo diante de um servidor malicioso configurado para oferecer métodos menos seguros, o dispositivo manteve o uso de PEAP-MSCHAPv2, recusando a substituição por opções como PEAP-PAP que transmitiriam a senha em texto claro dentro do túnel TLS como pode ser observado nas Figuras 32 e 34.

Outro ponto crítico identificado foi que o dispositivo conectou-se automaticamente à rede falsa configurada com o mesmo SSID da eduroam, sem apresentar qualquer alerta relacionado ao uso de um certificado auto-assinado. Tal comportamento decorre da configuração inicial realizada pelo próprio usuário, que havia selecionado a opção “Não validar certificado” ao registrar a rede original. Com isso, o Android passou a aceitar qualquer servidor como confiável, eliminando a verificação da cadeia de certificação e tornando-se suscetível a ataques *evil twin*.

Na etapa seguinte do experimento, buscou-se induzir o uso de métodos EAP considerados inseguros, configurando o servidor malicioso para aceitar exclusivamente *inner methods* vulneráveis, como PAP ou GTC, que expõem as credenciais do usuário em texto claro dentro do túnel TLS. O objetivo era verificar se o dispositivo Android realizaria o *downgrade* automático ao detectar que o servidor não oferecia mais MSCHAPv2.

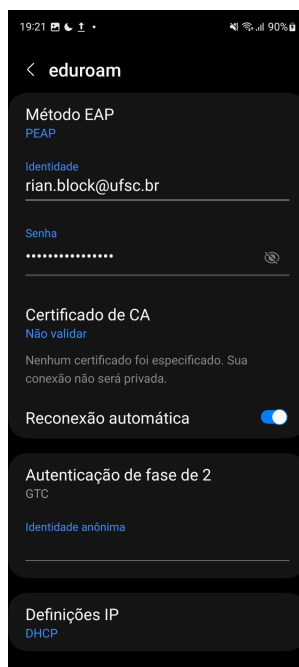
O dispositivo falhou no processo de associação e exibiu uma solicitação para reinserção dos dados da conta conforme Figura 31. E mesmo após a reinserção manual da identidade e da senha, o Android continuou incapaz de completar a autenticação, recusando-se a utilizar qualquer *inner method* diferente do originalmente configurado.

Figura 30: Captura da tela de conexão e configuração do Android ataque de downgrade.



Fonte: Autor (2025).

Figura 31: Captura de tela de conexão e configuração do Android tentando forçar a utilização de inner GTC.



Fonte: Autor (2025).

Figura 32: Captura do Wireshark durante as tentativas de downgrade do *inner method*.

No.	Time	Source	Destination	Protocol	Length	Info
12.	60.173050	192.168.56.189	192.168.56.100	RADIUS		182 Response, Identity
12.	60.173958	192.168.56.100	192.168.56.189	RADIUS		106 Request, Protected EAP (EAP-PEAP)
12.	60.181834	192.168.56.189	192.168.56.100	RADIUS		326 Client Hello
12.	60.184259	192.168.56.100	192.168.56.189	RADIUS		1110 Request, Protected EAP (EAP-PEAP)
12.	62.132636	192.168.56.189	192.168.56.100	RADIUS		191 Response, Protected EAP (EAP-PEAP)
12.	62.134238	192.168.56.100	192.168.56.189	RADIUS		266 Ignored Unknown Record
12.	62.605876	192.168.56.189	192.168.56.100	RADIUS		202 Alert (Level: Fatal, Description: Unknown CA)
12.	63.612206	192.168.56.189	192.168.56.100	RADIUS		202 Response, Protected EAP (EAP-PEAP)
12.	63.612552	192.168.56.100	192.168.56.189	RADIUS		86 Failure
12.	63.612740	192.168.56.189	192.168.56.100	RADIUS		86 Failure
12.	67.249078	192.168.56.189	192.168.56.100	RADIUS		182 Response, Identity
12.	67.250238	192.168.56.100	192.168.56.189	RADIUS		106 Request, Protected EAP (EAP-PEAP)
12.	67.253602	192.168.56.189	192.168.56.100	RADIUS		326 Client Hello
12.	67.255510	192.168.56.100	192.168.56.189	RADIUS		1110 Request, Protected EAP (EAP-PEAP)
12.	67.275753	192.168.56.189	192.168.56.100	RADIUS		191 Response, Protected EAP (EAP-PEAP)
12.	67.276752	192.168.56.100	192.168.56.189	RADIUS		266 Server Hello, Certificate, Server Key Exchange, Server Hello Done
12.	67.281641	192.168.56.189	192.168.56.100	RADIUS		202 Alert (Level: Fatal, Description: Unknown CA)
12.	68.256333	192.168.56.189	192.168.56.100	RADIUS		202 Response, Protected EAP (EAP-PEAP)
12.	68.290454	192.168.56.100	192.168.56.189	RADIUS		86 Failure
13.	08.621015	192.168.56.189	192.168.56.100	RADIUS		182 Response, Identity
13.	08.621670	192.168.56.100	192.168.56.189	RADIUS		106 Request, Protected EAP (EAP-PEAP)
13.	08.626187	192.168.56.189	192.168.56.100	RADIUS		326 Client Hello
13.	08.628034	192.168.56.100	192.168.56.189	RADIUS		1110 Request, Protected EAP (EAP-PEAP)
13.	08.643804	192.168.56.189	192.168.56.100	RADIUS		191 Response, Protected EAP (EAP-PEAP)
13.	08.644895	192.168.56.100	192.168.56.189	RADIUS		266 Server Hello, Certificate, Server Key Exchange, Server Hello Done
13.	08.653127	192.168.56.189	192.168.56.100	RADIUS		312 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
13.	08.654664	192.168.56.100	192.168.56.189	RADIUS		181 Change Cipher Spec, Encrypted Handshake Message
13.	08.660345	192.168.56.189	192.168.56.100	RADIUS		191 Response, Protected EAP (EAP-PEAP)
13.	08.661177	192.168.56.100	192.168.56.189	RADIUS		159 Application Data
13.	08.668364	192.168.56.189	192.168.56.100	RADIUS		244 Application Data
13.	08.669986	192.168.56.100	192.168.56.189	RADIUS		159 Application Data
13.	08.674152	192.168.56.189	192.168.56.100	RADIUS		244 Application Data
13.	08.675557	192.168.56.100	192.168.56.189	RADIUS		159 Application Data
13.	08.681679	192.168.56.189	192.168.56.100	RADIUS		244 Application Data
13.	08.683229	192.168.56.100	192.168.56.189	RADIUS		220 Success
14.	130.462135	192.168.56.189	192.168.56.100	RADIUS		182 Response, Identity
14.	130.462754	192.168.56.100	192.168.56.189	RADIUS		106 Request, Protected EAP (EAP-PEAP)
14.	130.466632	192.168.56.189	192.168.56.100	RADIUS		326 Client Hello
14.	130.468237	192.168.56.100	192.168.56.189	RADIUS		1110 Request, Protected EAP (EAP-PEAP)
14.	130.480225	192.168.56.189	192.168.56.100	RADIUS		191 Response, Protected EAP (EAP-PEAP)
14.	130.481128	192.168.56.100	192.168.56.189	RADIUS		266 Server Hello, Certificate, Server Key Exchange, Server Hello Done
14.	130.493453	192.168.56.189	192.168.56.100	RADIUS		312 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
14.	130.494859	192.168.56.100	192.168.56.189	RADIUS		181 Change Cipher Spec, Encrypted Handshake Message
14.	130.500300	192.168.56.189	192.168.56.100	RADIUS		191 Response, Protected EAP (EAP-PEAP)
14.	130.504438	192.168.56.100	192.168.56.189	RADIUS		159 Application Data
14.	130.508721	192.168.56.189	192.168.56.100	RADIUS		244 Application Data
14.	130.510436	192.168.56.100	192.168.56.189	RADIUS		159 Application Data
14.	130.515405	192.168.56.189	192.168.56.100	RADIUS		244 Application Data
14.	130.516409	192.168.56.100	192.168.56.189	RADIUS		159 Application Data
14.	130.521239	192.168.56.189	192.168.56.100	RADIUS		244 Application Data
14.	130.522807	192.168.56.100	192.168.56.189	RADIUS		220 Success

Fonte: Autor (2025).

Figura 33: Conteúdo do Pacote RADIUS observado através do Wireshark.

0000	08 00 27 7f 43 bf cc 32 e5 23 86 32 08 00 45 00	..'.C..2.#.2.E..
0010	00 a8 e7 29 40 00 40 11 60 a9 c0 a8 38 bd c0 a8	...)@.~...8...
0020	38 64 8b ef 07 14 00 94 54 d8 01 00 00 8c ea b7	8d.....T.....
0030	73 e0 f8 68 10 53 8a c5 d0 97 ed 1d ff 33 01 0c	s..h.S... ..3..
0040	72 69 61 6e 2e 62 6c 6f 63 6b 04 06 c0 a8 38 bd	r!an.blo ck...8..
0050	20 0b 52 61 6c 69 6e 6b 41 50 30 05 00 00 00	..Ralink AP0.....
0060	00 1e 13 43 43 2d 33 32 2d 45 35 2d 32 33 2d 38	...CC-32 -E5-23-8
0070	36 2d 33 32 1f 13 36 32 2d 34 39 2d 39 38 2d 39	6-32..62 -49-98-9
0080	30 2d 36 46 2d 33 46 0c 06 00 00 05 78 3d 06 00	0-6F-3F... ..x=...
0090	00 00 13 4f 11 02 01 00 0f 01 72 69 61 6e 2e 62	...0.... ..rian.b
00a0	6c 6f 63 6b 50 12 32 3b 31 54 64 c9 6c a7 31 fb	lockP.2; 1Td.1.1.
00b0	38 4c 25 64 8e 94	8L%d..

Fonte: Autor (2025).

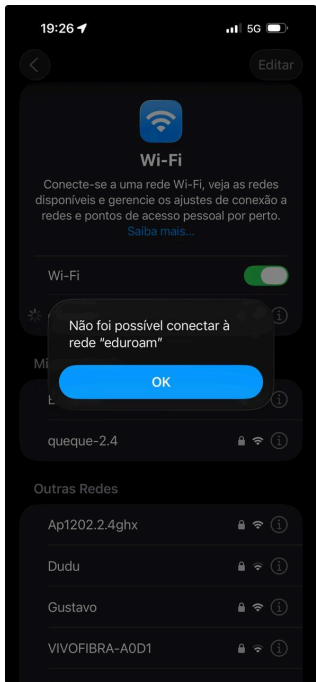
3.5.7.2 iPhone

Nos testes realizados com dispositivo iOS, observou-se um comportamento semelhante ao do Android no que diz respeito à resistência ao *downgrade* dos métodos internos de autenticação como pode ser analisado nas Figuras 36 e 38. Quando o servidor malicioso foi configurado para oferecer apenas métodos EAP inseguros, o iPhone não aceitou substituir o MSCHAPv2 e imediatamente exibiu uma mensagem de erro mostrado na Figura 35, indicando falha na autenticação. Assim como no caso do Android, o dispositivo recusou qualquer tentativa de negociação com *inner methods* menos seguros.

Além disso, diferentemente do Android configurado sem validação de certificado, o iPhone não se conectou automaticamente à rede falsa que replicava o SSID original. Ao detectar o certificado autoassinado apresentado pelo servidor malicioso, o sistema exibiu uma mensagem de alerta destacando que o certificado não podia ser verificado e solicitando

confirmação explícita do usuário antes de prosseguir conforme Figura 37. Embora ainda vulnerável caso o usuário aceite manualmente um certificado inválido.

Figura 35: Mensagem de erro ao conectar se a rede eduroam em iPhone.



Fonte: Autor (2025).

Figura 36: Captura dos pacotes RADIUS durante a conexão.

10. 479.603236	192.168.56.189	192.168.56.100	RADIUS	198 Response, Identity
10. 479.603910	192.168.56.100	192.168.56.189	RADIUS	106 Request, Protected EAP (EAP-PEAP)
10. 479.624670	192.168.56.189	192.168.56.100	RADIUS	354 Client Hello
10. 479.628019	192.168.56.100	192.168.56.189	RADIUS	1110 Request, Protected EAP (EAP-PEAP)
10. 479.640600	192.168.56.189	192.168.56.100	RADIUS	199 Response, Protected EAP (EAP-PEAP)
10. 479.641783	192.168.56.100	192.168.56.189	RADIUS	299 Server Hello, Certificate, Server Key Exchange, Server Hello Done
10. 479.676372	192.168.56.189	192.168.56.100	RADIUS	353 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
10. 479.677954	192.168.56.100	192.168.56.189	RADIUS	181 Change Cipher Spec, Encrypted Handshake Message
10. 479.684944	192.168.56.189	192.168.56.100	RADIUS	199 Response, Protected EAP (EAP-PEAP)
10. 479.685828	192.168.56.100	192.168.56.189	RADIUS	159 Application Data
10. 479.693165	192.168.56.189	192.168.56.100	RADIUS	268 Application Data
10. 479.694745	192.168.56.100	192.168.56.189	RADIUS	159 Application Data
10. 479.703454	192.168.56.189	192.168.56.100	RADIUS	252 Application Data
10. 479.704729	192.168.56.100	192.168.56.189	RADIUS	159 Application Data
10. 479.711626	192.168.56.189	192.168.56.100	RADIUS	252 Application Data
10. 479.712345	192.168.56.100	192.168.56.189	RADIUS	228 Success

Frame 1023: Packet, 198 bytes on wire (1584 bits), 198 bytes captured (1584 bits) on Interface \Device\NPF_{284C0138-96...}

Ethernet II, Src: TpLinkTechno_23:86:32 (cc:32:e5:23:86:32), Dst: PCSystemtec_7f:43:bf (08:00:27:7f:43:bf)

Internet Protocol Version 4, Src: 192.168.56.189, Dst: 192.168.56.100

User Datagram Protocol, Src Port: 35823, Dst Port: 1812

RADIUS Protocol

- Code: Access-Request (1)
- Packet identifier: 0x48 (72)
- Length: 156
- Authenticator: ee75854c0ef5b9f3782dab262af72ef7
- [The response to this request is in frame 1024]
- Attribute Value Pairs
 - AVP: t=User-Name(1) l=20 val=rian.block@ufsc.br
- Type: 1

0000 08 00 27 7f 43 bf cc 32 e5 23 86 32 08 00 45 00 ... C 2 # 2 E

0010 00 b8 e7 73 40 00 40 11 60 4f c0 a8 38 bd c0 a8 ... sg @ 'O 8

0020 38 64 8b ef 07 14 00 a4 5e 69 01 48 00 9c ee 75 8d ... i H u

0030 85 4c 0e f5 b9 f3 78 20 ab 26 2a f7 2e f7 01 14 L ... x 8

0040 72 69 63 6e 26 02 6c 6f e3 6b 40 75 66 73 63 28 rian.blo ck@ufsc

0050 62 72 04 06 c0 a8 38 bd 20 0b 52 61 6c 69 6e 6b br ... 8 Ralink

0060 41 50 30 05 06 00 00 00 00 1e 13 43 43 2d 33 32 AP0 ... CC-32

0070 2d 45 35 2d 32 33 2d 38 36 2d 33 32 1f 13 32 45 -E5-23-8 6-32 2E

0080 2d 44 46 2d 38 35 2d 32 42 2d 34 34 2d 35 30 0c -DF-85-2 B-44-98

0090 06 00 00 05 78 3d 06 00 00 00 13 4f 19 02 01 00 ... x ... 0

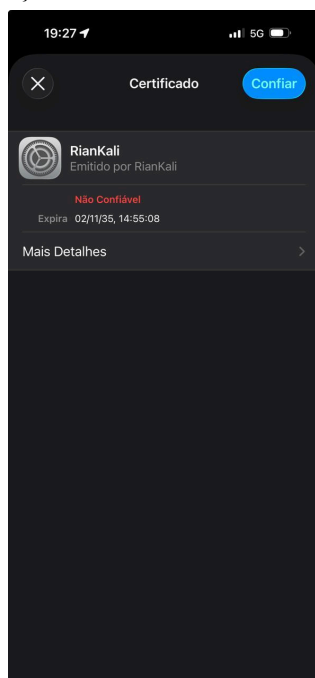
00a0 17 01 72 69 61 6e 2e 62 6c 6f 63 6b 40 75 66 73 rian.b lock@ufs

00b0 63 2e 62 72 50 12 e7 ff 25 e6 f6 e3 d9 f3 24 2d c.brP ... % ... \$

00c0 Se f7 e2 00 0f 35 A ... 5

Fonte: Autor (2025).

Figura 37: Notificação do iPhone sobre certificado não confiável.



Fonte: Autor (2025).

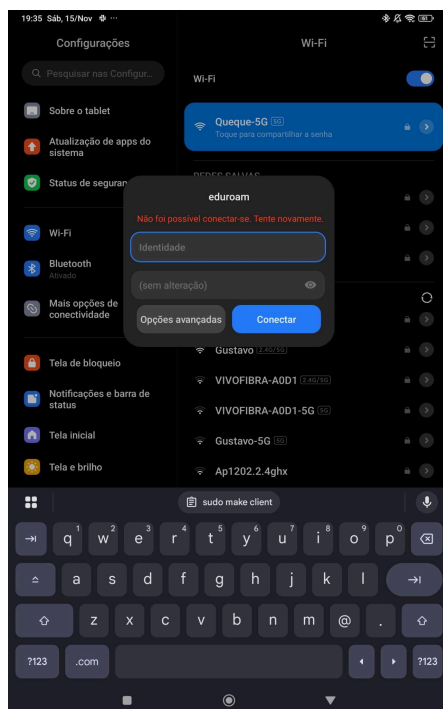
3.5.7.3 Tablet Android

Os testes realizados com o *tablet* Android apresentaram comportamento idêntico ao observado no dispositivo Android utilizado anteriormente. Assim como o *smartphone*, o *tablet* não realizou *downgrade* para métodos internos menos seguros quando o servidor malicioso passou a anunciar apenas opções vulneráveis, exibindo mensagem de erro e pedindo inserção de dados conforme Figura 39. O dispositivo insistiu na utilização de PEAP-MSCHAPv2, recusando qualquer tentativa de substituição automática do *inner method* originalmente configurado.

Da mesma forma, o *tablet* conectou-se automaticamente ao ponto de acesso malicioso quando a validação de certificado havia sido desativada pelo usuário durante a configuração inicial da rede. Nesse cenário, o dispositivo estabeleceu o túnel TLS com o servidor falso sem exibir alertas sobre o certificado auto-assinado, reproduzindo exatamente o comportamento previamente identificado no *smartphone* Android.

Como pode ser observado na Figura 40, apesar das credenciais não terem sido exibidas em texto puro, é possível notar que o usuário trafegou sem qualquer proteção, em texto puro inclusive pelo Wireshark. No caso de utilização de CPF isso pode ser uma falha relevante, mas que poderia ser mitigada com a utilização do *outer identity*.

Figura 39: Exibição de falha ao se conectar a rede eduroam em Tablet e solicitação de reinserção de dados.



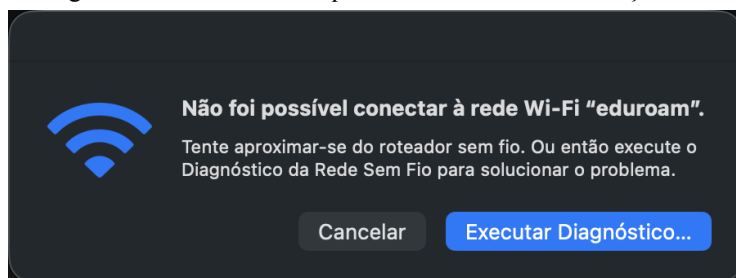
Fonte: Autor (2025).

3.5.7.4 MacOS

Os testes conduzidos em um dispositivo macOS revelaram um comportamento equivalente ao observado previamente no iPhone. Quando o servidor malicioso foi configurado para oferecer apenas métodos internos inseguros, o macOS recusou o *downgrade* e não concluiu o processo de autenticação. Conforme é possível observar na Figura 41, o sistema apresentou imediatamente uma mensagem de erro, indicando falha na negociação do método de segurança e impossibilidade de estabelecer uma conexão válida com o ponto de acesso falso, como mostra a.

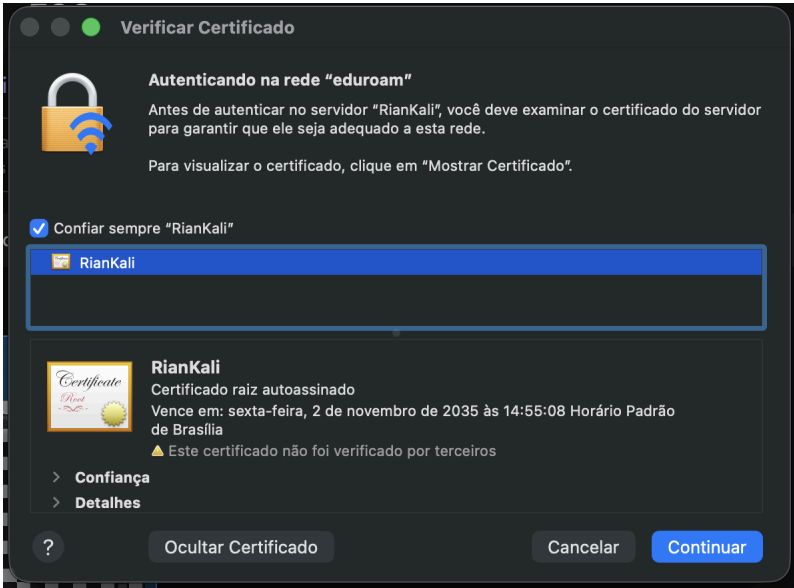
Além disso, o macOS não se conectou automaticamente à rede maliciosa que replicava o SSID da rede original. Assim que o servidor apresentou um certificado auto-assinado ou não pertencente à cadeia de confiança configurada, o sistema exibiu um alerta ao usuário, informando que o certificado não podia ser verificado e solicitando confirmação explícita antes de prosseguir (Figura 42).

Figura 41: Mensagem de erro no MacOS quando o servidor tentou forçar métodos inseguros.



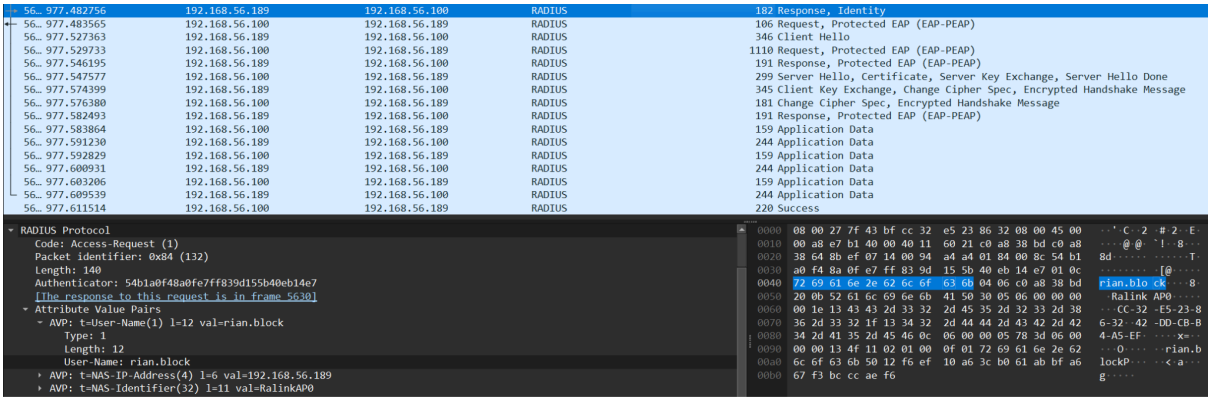
Fonte: Autor (2025).

Figura 42: Aviso sobre a impossibilidade de verificar o certificado no MacOS.



Fonte: Autor (2025).

Figura 43: Captura do Wireshark durante o ataque ao MacOS.



Fonte: Autor (2025).

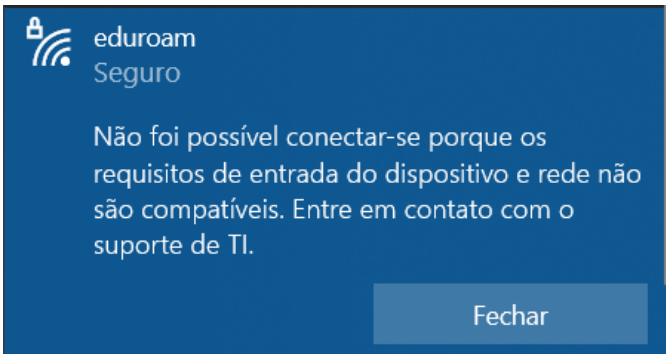
3.5.7.5 Windows

No caso do Windows, não foi possível reproduzir integralmente o cenário ideal, pois o equipamento utilizado não tinha a conexão original registrada no sistema. Por esse motivo, o teste foi conduzido simulando a tentativa de associação a um servidor malicioso que anunciava métodos internos depreciados ou inseguros, com o objetivo de verificar se o Windows realizaria *downgrade* ou aceitaria negociar autenticações fora do padrão recomendado.

Durante os experimentos, observou-se que o Windows não realizou o *downgrade* *inner method*. Mesmo quando o servidor malicioso passou a se utilizar da técnica de *downgrade*, o dispositivo se manteve na tentativa de uso do PEAP-MSCHAPv2, recusando a substituição por qualquer método menos seguro conforme Figura 45. Em casos em que o servidor tentou forçar explicitamente o uso de métodos incompatíveis com a política de

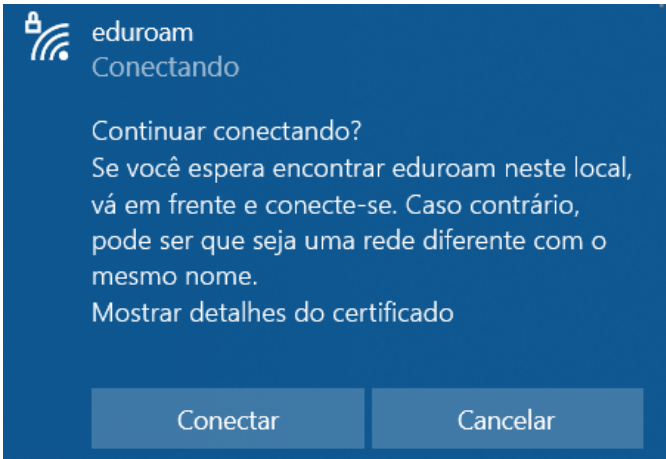
segurança do *supplicant*, o Windows exibiu uma mensagem de erro (Figura 44), indicando incompatibilidade entre a configuração da rede e o dispositivo cliente.

Figura 45: Mensagem de erro de conexão no ataque ao Windows.



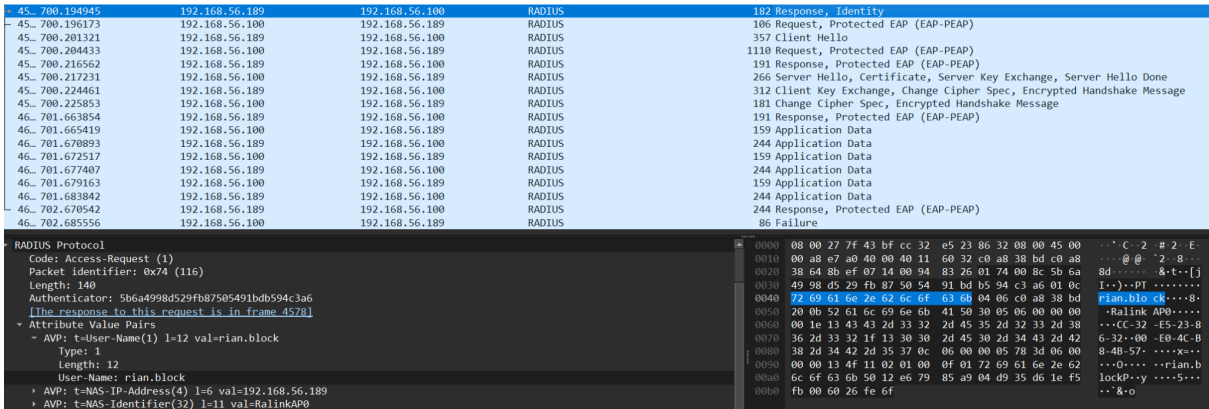
Fonte: Autor (2025).

Figura 46: Mensagem certificado invalido no windows.



Fonte: Autor (2025).

Figura 47: Captura dos pacotes RADIUS através do Wireshark durante ataque ao windows.



Fonte: Autor (2025).

3.5.8 Tabelas comparativas dos resultados obtidos

Figura 49: Tabela comparativa entre os métodos testados e o seu comportamento perante as vulnerabilidades.

Visibilidade de Credenciais por Método EAP		
Comparação da exposição de dados sensíveis em capturas de tráfego e logs de servidor		
MÉTODO EAP	VISÍVEIS NO WIRESHARK?	VISÍVEIS NOS LOGS?
EAP-PEAP (MSCHAPv2)	Não	Não (apenas o hash MSCHAPv2)
EAP-TTLS (PAP)	Não	Sim
EAP-TTLS (GTC)	Não	Sim
EAP-TLS	Não*	Não*
EAP-MD5	Não	Não (apenas hash MD5)
EAP-PEAP (GTC)	Não	Sim
* Não foi possível testar no ambiente experimental, portanto foi considerado somente o comportamento teórico.		

Fonte: Autor (2025).

Figura 50: Tabela de dispositivos que aceitaram a conexão automática ao AP falso.

CONEXÃO AUTOMÁTICA POR DISPOSITIVO	
DISPOSITIVO / SO	ACEITARAM CONEXÃO AUTOMÁTICA?
Windows (Notebook Dell)	Não
iOS (iPhone 16 Pro Max)	Não
macOS (Macbook Air)	Não
Android (Samsung S20 FE)	Sim*
Android (Tablet Xiaomi Mi Pad 5)	Sim*
* depende de configuração incorreta previamente e não validação do certificado.	

Fonte: Autor (2025).

3.6 RESULTADOS E DISCUSSÃO

Os experimentos realizados permitiram simular o comportamento real dos principais métodos EAP diante de três situações críticas: captura de tráfego em condições normais, exposição de dados sensíveis no servidor legítimo e exploração via ataque *evil twin* quando o cliente não valida o certificado do servidor. Os resultados evidenciam um padrão claro: quando o certificado é validado, todos os métodos encapsulados em TLS se mantêm seguros; quando não é validado, o grau de exposição varia conforme a força criptográfica do *inner method*. A seguir estão os achados organizados por método:

- **EAP-MD5:** No tráfego capturado via Wireshark não se observou o *Challenge* e *Response* do MD5, o que confirma que o túnel TLS funciona como previsto. Porém, no servidor falso que é o *endpoint* do túnel, o método interno aparece nos logs do servidor RADIUS. Com isso o atacante pode obter o valor MD5-*Response*, que é composto por:

MD5(ID || senha || challenge)

Esse valor não revela a senha em texto claro, mas permite ataques offline de força bruta, já que o MD5 é fraco e o espaço de busca para senhas comuns é pequeno. Além disso, como o atacante controla o servidor falso, ele pode impor um challenge estático (fixo) para todas as conexões. Isso viabiliza o uso de Rainbow Tables pré-computadas especificamente para esse desafio, anulando o custo computacional da força bruta e permitindo a descoberta instantânea da senha. O que resulta no comprometimento total da credencial. Assim, mesmo rodando dentro de TLS, o EAP-MD5 depende totalmente da verificação do certificado;

- **O EAP-TLS:** é considerado o padrão-ouro da autenticação wireless, pois não utiliza senhas e exige certificados do cliente e do servidor. No entanto, nos experimentos deste trabalho não foi possível concluir os testes porque o certificado do cliente não pôde ser transferido e instalado no dispositivo suplicante. Esse impedimento não altera a análise conceitual do método. A impossibilidade de testes práticos apenas reforça um ponto conhecido: EAP-TLS possui barreiras operacionais significativas. Cada cliente precisa de um certificado próprio, armazenado com chave privada. O FreeRADIUS estava corretamente configurado (CA, certificados, PKCS#12), mas sem instalar o certificado no dispositivo a autenticação não funciona. O resultado é importante para a discussão de que métodos considerados “perfeitos” em termos criptográficos podem ser inviáveis se a infraestrutura de distribuição e instalação de certificados não estiver madura. Em ambientes como o eduroam, essa dificuldade explica por que EAP-TLS é pouco adotado, apesar da segurança superior;
- **EAP-TTLS:** Nos testes com TTLS usando PAP, o comportamento normal foi que nenhuma credencial apareceu na captura de tráfego, pois todo o conteúdo interno fica cifrado dentro do TLS. Isso confirma que, quando o certificado é validado e o túnel é estabelecido com o servidor legítimo, o método é seguro mesmo utilizando um *inner method* fraco como o PAP. A situação muda completamente quando o cliente não valida o certificado. No cenário *evil twin*, o túnel TLS se forma com o servidor falso,

que passa a ser o endpoint legítimo da sessão. Ali, o PAP aparece em texto claro, incluindo:

User-Name = "usuario"

User-Password = "senha"

Tecnicamente isso ocorre porque o PAP não aplica nenhuma forma de desafio-resposta, ele envia credenciais diretamente para dentro do túnel. Se o túnel termina no atacante, a proteção deixa de existir. Assim, a segurança do TTLS-PAP depende 100% da validação do certificado.

- **EAP-PEAP:** Nos testes com PEAP-MSCHAPv2, nenhum dado sensível apareceu na captura do Wireshark. O MSCHAPv2 não envia a senha, apenas o *Challenge-Response* derivado do hash NTLM. Mesmo em ambientes *evil twin*, o atacante só recebe o *NT-Response*, que exige ataques complexos, embora possíveis, para recuperar a senha. É uma exposição menor do que PAP e GTC, mas ainda problemática. Os pontos de atenção a esse método são: MSCHAPv2 é criptograficamente considerado fraco; ataques *offline* contra NTLM já são amplamente documentados; ferramentas como as do projeto ASLEAP e Hashcat conseguem quebrar respostas MSCHAPv2 em condições realistas.

Portanto, PEAP-MSCHAPv2 não expõe a senha em texto claro, mas ainda permite ataques *offline* quando o túnel é estabelecido com o servidor falso. É mais seguro que PAP e MD5, porém ainda insuficiente caso o cliente não valide certificados;

- **EAP-GTC:** Nos experimentos com PEAP-GTC, o tráfego capturado via Wireshark não mostrou credenciais, confirmando que o TLS funciona. O problema surge no mesmo ponto crítico dos métodos anteriores: ausência de validação do certificado. O GTC envia credenciais em texto claro dentro do túnel. Quando o túnel termina no servidor malicioso, o atacante recebe:

User-Name = "usuario"

User-Password = "senha"

Isso foi observado na prática nos experimentos com *supplicant* compatível. Vale notar que poucos dispositivos modernos oferecem suporte ao GTC, o que reduz o impacto, mas entre todos os métodos testados, o GTC é o mais arriscado em cenários de *evil twin* por transmitir a senha de forma explícita. O TTLS-GTC apresentou comportamento idêntico, sendo seguro quando o TLS é legítimo e totalmente vulnerável quando o túnel é estabelecido com o atacante;

EAP-FAST: O EAP-FAST foi configurado no servidor, mas nenhum dos dispositivos disponíveis suportava o método. Isso limita a análise, mas gera um achado relevante: métodos menos comuns não são adequados para ambientes amplos como o eduroam, onde a compatibilidade entre plataformas é essencial. Por essa razão e por estar fora do conjunto de métodos empregados na prática, o FAST não foi incluído na comparação principal

Os testes revelam um padrão nítido. Quando o certificado do servidor é validado, todos os métodos encapsulados em TLS funcionam corretamente e de forma segura. Porém, quando o certificado não é validado, a proteção do TLS deixa de existir pois o atacante passa a ser o *endpoint* do túnel e a gravidade da exposição depende do *inner method*.

A simulação de ataque *evil twin* com de *man-in-the-middle* foi o experimento central deste trabalho, pois permitiu avaliar na prática o que acontece quando um cliente eduroam se conecta a um servidor malicioso que imita a infraestrutura legítima. Para isso, o FreeRADIUS foi configurado para aceitar qualquer credencial apresentada e para replicar a configuração da eduroam da UFSC, baseada em PEAP com *inner* MSCHAPv2. Essa configuração é alinhada com o ambiente real e garante que os resultados observados sejam relevantes para a simulação. Analisando exclusivamente o comportamento básico do PEAP-MSCHAPv2 ele se mostrou um dos mais seguros e confiáveis, mesmo quando o túnel TLS era estabelecido com o servidor falso, a senha do usuário nunca foi enviada em texto claro. O atacante passou a ter acesso apenas ao par desafio-resposta do MSCHAPv2, que pode ser usado em ataques *offline*, mas não expõe diretamente a senha.

Por esse motivo, o experimento buscou ir além do caso “normal” do PEAP-MSCHAPv2 e tentou induzir ativamente um *downgrade* do *inner method*. A ideia era configurar o servidor malicioso para oferecer apenas métodos internos inseguros, como PAP ou GTC, que enviam a senha em texto claro dentro do túnel TLS. Se algum dispositivo aceitasse substituir MSCHAPv2 por PAP ou GTC, seria possível demonstrar de forma inequívoca como a ausência de validação de certificado permite que um atacante controle o *endpoint* TLS e leia diretamente usuário e senha. Na prática, nenhum dos dispositivos modernos testados realizou esse *downgrade*. Tanto Android de API 33 (*smartphone* e *tablet*), quanto iOS, macOS e Windows se recusaram a trocar o *inner method*. Em todos os casos, o resultado foi falha de autenticação, exibição de mensagens de erro ou solicitação de reinserção das credenciais, mas nunca a aceitação automática de PAP ou GTC. Isso é um achado importante pois a pilha de EAP dos sistemas atuais implementa políticas que impedem, pelo lado do cliente, a negociação “para baixo” de métodos internos. Mesmo em um cenário controlado e com o servidor tentando forçar o *downgrade*, o cliente preferiu não conectar do que abrir mão do MSCHAPv2.

Embora todos os sistemas tenham se mostrado resistentes ao *downgrade*, o comportamento em relação à validação de certificados foi bem diferente entre plataformas. Nos dispositivos Android testados (*smartphone* e *tablet*), a vulnerabilidade mais grave apareceu na etapa de configuração inicial da rede. Ao selecionar a opção “não validar certificado” na primeira conexão à eduroam, o usuário instrui o sistema a aceitar qualquer servidor como confiável dali em diante. O efeito prático disso é que, quando foi criado um ponto de acesso malicioso com o mesmo SSID, o Android conectou automaticamente ao *evil twin*, sem exibir qualquer alerta sobre o uso de certificado auto-assinado.

No iPhone e no macOS, o resultado foi diferente. Ainda que esses dispositivos também não realizassem *downgrade* para métodos internos inseguros, eles mantiveram uma postura mais restritiva na validação de certificados. Ao tentar conectar na rede falsa, tanto iOS quanto macOS exibiram de imediato um alerta informando que o certificado apresentado não podia ser verificado, exigindo confirmação explícita do usuário para prosseguir. Na prática, isso não elimina o risco, um usuário desatento pode aceitar o certificado por engano, mas introduz uma barreira importante na experiência de uso. A necessidade de uma decisão consciente quebra a conexão automática com qualquer servidor que se apresente com o mesmo SSID e força o usuário a perceber que algo está fora do esperado.

O Windows, apesar de não estar com um perfil eduroam previamente configurado, apresentou comportamento alinhado com os iPhone e MacOs. Nos cenários em que o servidor malicioso anunciava apenas métodos internos depreciados, o *supplicant* não aceitou negociar fora do padrão de segurança, permaneceu tentando PEAP-MSCHAPv2 e exibiu mensagens de incompatibilidade quando isso era forçado pelo lado do servidor. A combinação desses resultados permite discutir o cenário de risco real da eduroam. Primeiro, os testes mostram que a simples presença de TLS e de MSCHAPv2 não é suficiente para garantir segurança se o usuário desabilita a validação de certificado. Nesse caso, tanto o *smartphone* quanto o *tablet* Android aceitaram formar o túnel TLS com o servidor atacante de forma automática, entregando a ele o desafio-resposta do MSCHAPv2. Embora isso não exponha a senha em texto claro, abre espaço para ataques *offline* que, na prática, podem ser viáveis dependendo da política de senhas da instituição. Segundo, os resultados deixam claro que tentativas de *downgrade* para *inner methods* mais fracos esbarram hoje em limitações do lado cliente; o risco principal, portanto, desloca-se do “protocolo errado” para a “configuração errada”.

Do ponto de vista técnico, os testes indicam que a pilha EAP dos sistemas operacionais atuais implementa mecanismos mínimos de proteção contra *downgrade* de *inner method*, o que é um ponto positivo. Do ponto de vista de experiência do usuário e de configuração, porém, Android ainda permite escolhas que, na prática, anulam o benefício desses mecanismos. A opção de “não validar certificado” é apresentada ao usuário comum sem contexto suficiente sobre as consequências, e isso cria exatamente o cenário em que o *evil twin* se torna viável. Em contraste, iOS, macOS e Windows impõem mais restrições, exigindo interação do usuário quando encontram certificados não confiáveis, o que reduz a probabilidade de uma aceitação automática, ainda que não elimine a possibilidade de erro humano.

3.6.1 Conclusões com base nos Resultados

Os experimentos conduzidos permitiram observar, em condições mais próximas do real, como cada método EAP responde quando deslocado de um ambiente controlado para um cenário de ataque baseado em *evil twin* com *man-in-the-middle*. A análise experimental confirmou fragilidades já descritas na literatura, mas trouxe nuances práticas que dificilmente aparecem em estudos puramente teóricos.

O comportamento do EAP-MD5 evidenciou sua inadequação completa para ambientes modernos. Mesmo quando encapsulado em TLS, o método continua oferecendo segurança insuficiente em ataques de *brute-force offline* onde o túnel é estabelecido com um servidor fraudulento. O problema não está apenas no método, mas na suposição equivocada de que o TLS externo neutralizaria suas fraquezas internas. Os testes demonstraram que isso não ocorre quando o cliente é induzido a fechar o túnel com o atacante, possibilitando a captura do MD5-Response e posteriormente a sua quebra criptográfica, exibindo as credenciais do usuário.

Nos métodos baseados em túneis TLS, como PEAP e TTLS, o transporte se manteve criptograficamente sólido, mas a cadeia de confiança se rompeu exatamente na etapa mais vulnerável a de validação do certificado. Quando o *supplicant* ignorou essa verificação, o

atacante assumiu o papel do servidor legítimo e passou a receber o *inner method* enviado pelo cliente. Esse resultado reforça uma crítica recorrente ao modelo eduroam em que a segurança do túnel depende de uma decisão tomada pelo usuário. Nesses casos, PAP e GTC foram expostos integralmente, e os desafios MSCHAPv2 também ficaram acessíveis para ataques subsequentes porém não em texto puro.

O EAP-TLS demonstrou ser a opção mais resistente, mas a dificuldade prática de gerenciar certificados de cliente torna sua adoção limitada. A simulação enfrentou diversos desafios referentes à criação e compartilhamento desses certificados. Impossibilitando o teste desse método em nossa simulação. Porém, podemos afirmar que a sua robustez depende de um ecossistema de distribuição, revogação e gerenciamento de certificados que poucas instituições conseguem manter de forma consistente em escala. O contraste entre a segurança obtida e o esforço operacional exigido evidencia por que esse método, apesar de tecnicamente ideal, não é amplamente adotado em redes federadas como o eduroam.

Ao simular o fluxo completo da autenticação 802.1X, do AP ao servidor RADIUS malicioso, foi possível observar a natureza federada do eduroam e como sua confiança é distribuída em múltiplas camadas. Nesse contexto, a vulnerabilidade não surge em decorrência do protocolo, mas da interação entre camadas e, principalmente, das interfaces de configuração do *supplicant*. O ataque só teve sucesso na captura das credenciais quando o cliente aceitou certificados inválidos, o que desloca a responsabilidade da criptografia para a experiência de configuração adequada.

A análise entre sistemas operacionais expôs diferenças importantes de implementação e UX. iOS, macOS e Windows não se conectaram automaticamente a rede eduroam fraudulenta, exigindo confirmação do usuário, enquanto os dispositivos Android que foram configurados na rede verídica selecionando a opção de “não validar certificado” se conectaram automaticamente ao servidor RADIUS malicioso. Contudo vale ressaltar que nenhum dos dispositivos testados aceitou a transição para *inner methods* inseguros sugerida pelo servidor RADIUS malicioso. Mesmo quando o servidor tentou obrigar o uso de métodos inseguros, todos os dispositivos exibiram erros de conexão, evitando ataques por degradação.

Esses resultados dialogam com uma tendência reconhecida em pesquisas recentes na qual a segurança do eduroam não é vulnerável por falhas criptográficas, mas por permissões e flexibilidades de configuração herdadas de versões antigas dos sistemas operacionais. Além da liberdade dada ao usuário para ignorar a verificação do certificado.

A análise integral dos resultados indica que a resiliência do eduroam depende da convergência entre criptografia forte, configuração correta e um design de UX que não incentive e nem permita escolhas inseguras.

3.6.2 Recomendações e Melhorias

Os resultados obtidos neste trabalho mostraram que a segurança prática da eduroam depende menos da força criptográfica dos protocolos e mais da configuração correta dos dispositivos cliente e dos servidores RADIUS. Com base nisso, as melhorias a seguir têm foco em eliminar as condições que tornam o ataque *evil twin* viável e em reduzir a superfície de ataque associada ao uso do MSCHAPv2.

O ponto central dos ataques bem-sucedidos observados foi a possibilidade de o cliente ignorar ou desativar a validação de certificado. Em vários dispositivos Android, essa opção é oferecida ao usuário de forma pouco contextualizada, criando uma vulnerabilidade estrutural. Recomenda-se:

- Remover a opção “Não validar certificado” dos perfis fornecidos pelos IdPs;
- Distribuir perfis eduroam somente via CAT, já configurados com CA raiz e CN correto;
- Configurar o *supplicant* para recusar conexões automaticamente quando o nome do servidor não corresponder exatamente ao CN esperado.

objetivo: eliminar completamente a conexão automática com servidores maliciosos.

Como PEAP-MSCHAPv2 ainda é amplamente utilizado, recomenda-se endurecer sua configuração para minimizar danos como:

- Desabilitar GTC e PAP em todos os servidores RADIUS;
- Não anunciar *inner methods* inseguros na configuração do EAP no servidor;
- Configurar explicitamente o servidor para aceitar apenas MSCHAPv2 ou TLS;
- Impedir que fabricantes modifiquem perfis de RADIUS ou métodos através de customizações Android.

Embora não elimine o risco teórico, impede ataques de downgrade e evita exposição de senhas em texto claro.

O RADIUS tradicional não fornece confidencialidade fora do túnel EAP. RadSec (RADIUS over TLS) melhora a privacidade e impede coleta de metadados, com isso pode-se:

- Adotar RadSec para conexões *upstream* (instituição → nacional → regional);
- Verificar certificados mutuamente entre servidores;
- Usar SRV/NAPTR para descoberta automática.

Essas ações protegem a identidade externa, metadados do usuário e credenciais internas contra analisadores de tráfego como ataques *man-in-the-middle*.

Os experimentos também mostraram que o usuário expõe sua identidade real já no primeiro pacote de autenticação. Isso facilita rastreamento e reconhecimento por atacantes. Isso pode ser facilmente evitado ao:

- Configurar o *outer identity* como anon@instituição;
- Conservar a identidade real apenas no *inner* EAP.

Com isso se reduz a superfície de informação sensível antes da formação do túnel TLS.

Mesmo sem *downgrade*, o MSCHAPv2 permite *brute-force offline* se o atacante capturar o desafio e o NT-Response. Por isso recomenda-se:

- Políticas mínimas: 12+ caracteres, misto e alta entropia;
- Evitar senhas compartilhadas entre serviços institucionais;
- Aplicar limitação de tentativas no *Active Directory*.

Tornando ataques *offline* impraticáveis na prática.

Outro ponto fundamental é configurar o AP e o servidor RADIUS para impedir métodos obsoletos:

- Desabilitando EAP-MD5, LEAP, CHAP e métodos legados;
- Forçando criptografia WPA2-Enterprise ou WPA3-Enterprise com 192-bit *mode*;
- Verificando *logs* e rejeitando autenticações suspeitas.

Visando impedir que os dispositivos antigos exponham credenciais sem TLS.

4. CONCLUSÃO

Este trabalho avaliou a segurança da infraestrutura de autenticação do eduroam, com foco nos protocolos EAP e na arquitetura RADIUS. A combinação entre revisão conceitual e experimentos em ambiente controlado permitiu atingir os objetivos propostos.

O primeiro objetivo, de compreender o funcionamento e as vulnerabilidades dos protocolos EAP e RADIUS, foi alcançado ao demonstrar que métodos como EAP-MD5 continuam inseguros e que métodos baseados em túneis TLS dependem criticamente da validação correta do certificado. Os testes também confirmaram a robustez do EAP-TLS e evidenciaram suas limitações práticas de implantação em larga escala.

O segundo objetivo, de configurar um ambiente de testes que reproduzisse a arquitetura do eduroam, foi atendido com a implementação de um fluxo 802.1X realista envolvendo *supplicant*, ponto de acesso e servidor RADIUS malicioso. Essa configuração permitiu observar, na prática, como os ataques exploram vulnerabilidades existentes e quais etapas são determinantes para seu sucesso.

O terceiro objetivo, de identificar cenários que representam risco real ao usuário, mostrou que a maior e principal vulnerabilidade é quando o cliente aceita certificados inválidos. A comparação entre sistemas operacionais expôs diferenças importantes na experiência de configuração, com o Android permitindo conexões automáticas inseguras quando a validação é desativada, enquanto iOS, Windows e macOS exigem confirmação explícita.

Por fim, o objetivo de formular recomendações resultou em um conjunto de medidas práticas, como: utilizar o CAT na configuração de redes eduroam, proibir conexões sem validação de certificado, desativar *inner methods* obsoletos, usar identidade anônima, reforçar políticas de senha e adotar RadSec no transporte federado. Essas ações mitigam diretamente as vulnerabilidades observadas e fortalecem a resiliência do eduroam contra ataques baseados em engenharia de confiança.

Os resultados demonstram que a segurança do eduroam depende menos da força criptográfica dos protocolos e mais da eliminação de decisões inseguras no lado do cliente. O fortalecimento das configurações padrão e a adoção das recomendações apresentadas são passos essenciais para garantir uma infraestrutura de autenticação mais segura para universidades e instituições de pesquisa.

REFERÊNCIAS

- ABOULMAGD, O. et al. RFC 3748 – Extensible Authentication Protocol (EAP). IETF, 2004. Disponível em: <https://www.rfc-editor.org/rfc/rfc3748.html>.
- CANADIAN ACCESS FEDERATION (CAF). Evil-twin EAPHammer Attack Mitigation for eduroam. CANARIE, 25 mar. 2021. Disponível em: <https://www.canarie.ca/document/evil-twin-eaphammer-attack-mitigation-for-eduroam/>. Acesso em: 02 set. 2025.
- DORSEMAINE, B. et al. What Did Come Out of It? Analysis and Improvements of DIDComm Messaging. 2023.
- EPC 342-08 v12.0 – Guidelines on Cryptographic Algorithms Usage and Key Management. European Payments Council, 2022.
- FERGUSON, B.; KAUFMAN, C. Wi-Fi Security: Mastering 802.1X, RADIUS, and EAP. 1. ed. Pearson Education, 2014.
- FIEBRIG, D. et al. Careful with that Roam, Edu: Experimental Analysis of Eduroam Credential Stealing Attacks. 2019. Disponível em: <https://arxiv.org/pdf/1904.08376.pdf>.
- FreeRADIUS Technical Guide. Disponível em: <https://networkradius.com/doc/>.
- GÉANT. A guide to eduroam CAT for IdP administrators. Wiki GÉANT, 2023. Disponível em: <https://wiki.geant.org/spaces/H2eduroam/pages/121346267/A+guide+to+eduroam+CAT+for+IdP+administrators>. Acesso em: 04 dez. 2025.
- GEUER-POLTHIER, S. et al. Specifying SSI over EAP: Towards an Even Better Eduroam in the Future. 2021.
- GEUER-POLTHIER, S. et al. Roam Whitepaper v1.0. 2022. Disponível em: <https://roam.id/whitepaper.pdf>.
- NELSON, M. et al. CWSP – Certified Wireless Security Professional Official Study Guide. 2. ed. Wiley, 2017.
- OPPLIGER, R. Criptografia e Segurança de Redes. 6. ed. Rio de Janeiro: LTC, 2014.
- RIGNEY, C. et al. RFC 2865 – Remote Authentication Dial In User Service (RADIUS). IETF, 2000. Disponível em: <https://www.rfc-editor.org/rfc/rfc2865.html>.
- VAN DER WALT, D. FreeRADIUS Beginner's Guide. Packt Publishing, 2011.
- ROTH, V. et al. A Privacy-Preserving Cross-domain Network Access Service Using Sovrin Identifier. 2021.

RFC 5997 – Use of Status-Server Packets in the RADIUS Protocol. IETF, 2010. Disponível em: <https://www.rfc-editor.org/rfc/rfc5997>.

RFC 6614 – RADIUS over TLS. IETF, 2012. Disponível em: <https://www.rfc-editor.org/rfc/rfc6614.html>.

SECUREW2. *EAP-TLS vs PEAP-MSCHAPv2: Which Authentication Protocol is Superior*. SecureW2 Blog, 2022. Disponível em: <https://www.securew2.com/blog/eap-tls-vs-peap-mschapv2-which-authentication-protocol-is-superior>. Acesso em: 7 jul. 2025.

SIMMONS, R. et al. RFC 5216 – The EAP-TLS Authentication Protocol. IETF, 2008. Disponível em: <https://www.rfc-editor.org/rfc/rfc5216.html>.

VANHOEF, M.; PIESSENS, F. A Practical Investigation of Identity Theft Vulnerabilities in Eduroam. In: Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec), 2015.

APÊNDICE A

Logs, capturados na parte do servidor RADIUS malicioso em cada um dos métodos testados.

Figura 12: Captura de logs do freeradius após a conexão com o *suplicant* MD5.

```
(15) &reply::Framed-MTU += &session-state:Framed-MTU[*] -> 994
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.3 Handshake, ClientHello'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerHello'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Certificate'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerKeyExchange'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerHelloDone'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.1 Handshake, ClientKeyExchange'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.1 Handshake, Finished'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 ChangeCipherSpec'
(15) &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Finished'
(15) &reply::TLS-Session-Cipher-Suite += &session-state:TLS-Session-Cipher-Suite[*] -> 'ECDH
E-RSA-AES128-SHA'
(15) &reply::TLS-Session-Version += &session-state:TLS-Session-Version[*] -> 'TLS 1.1'
(15) } # update = noop
(15) [exec] = noop
(15) policy remove_reply_message_if_eap {
(15)   if (&reply:EAP-Message && &reply:Reply-Message) {
(15)     if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE
(15)   } else {
(15)     [noop] = noop
(15)   } # else = noop
(15) } # policy remove_reply_message_if_eap = noop
(15) if (EAP-Key-Name && &reply:EAP-Session-Id) {
(15)   if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE
(15) } # post-auth = noop
(15) Login OK: [test/<via Auth-Type = eap>] (from client routerplz port 0 cli 4E-14-B7-6B-A8-7A)
(15) Sent Access-Accept Id 116 from 192.168.56.100:1812 to 192.168.56.189:54219 length 172
(15) MS-MPPE-Recv-Key = 0x789734c775351ce296d5598febe9ea80498a86bd1850aad629527fe920bb0b33
(15) MS-MPPE-Send-Key = 0xb48cb7a6faa06b8cd9d2422f6d56093b26ab63f49d84111c06866bf9c0841f2
(15) EAP-Message = 0x030a0004
(15) Message-Authenticator = 0x00000000000000000000000000000000
(15) User-Name = "test"
(15) Framed-MTU += 994
(15) Finished request
Waking up in 4.9 seconds.
```

Fonte: Autor (2025).

Figura 17: Captura de logs e credenciais no freeradius após a conexão com o *suplicant* TTLS.

```
(12) [pap] = updated
(12) } # authorize = updated
(12) Found Auth-Type = PAP
(12) # Executing group from file /etc/freeradius/3.0/sites-enabled/inner-tunnel
(12) Auth-Type PAP {
(12) pap: Login attempt with password
(12) pap: Comparing with "known good" Cleartext-Password
(12) pap: User authenticated successfully
(12) [pap] = ok
(12) } # Auth-Type PAP = ok
(12) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/inner-tunnel
(12) post-auth {
(12)   if (0) {
(12)     if (0) -> FALSE
(12)   } # post-auth = noop
(12) Login OK: [test/test123] (from client routerplz port 0 via TLS tunnel)
(12) } # server inner-tunnel
(12) Virtual server sending reply
(12) eap_ttls: Got tunneled Access-Accept
(12) eap: Sending EAP Success (code 3) ID 5 length 4
(12) eap: Freeing handler
(12) [eap] = ok
(12) } # authenticate = ok
(12) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/default
(12) post-auth {
(12)   if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name =
= request:User-Name)) {
(12)     if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name =
= request:User-Name)) -> FALSE
(12)   } update {
(12)     &reply::Framed-MTU += &session-state:Framed-MTU[*] -> 994
(12)     &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
TTLS - recv TLS 1.3 Handshake, ClientHello'
(12)     &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
TTLS - send TLS 1.1 Handshake, ServerHello'
(12)     &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
TTLS - send TLS 1.1 Handshake, Certificate'
(12)     &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
TTLS - send TLS 1.1 Handshake, ServerKeyExchange'
```

Fonte: Autor (2025).

Figura 25: Captura de logs no freeradius após a conexão com o *suplicant* PEAP.

```

(43) update {
(43)   &reply::Framed-MTU += &session-state:Framed-MTU[*] -> 994
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.3 Handshake, ClientHello'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerHello'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Certificate'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerKeyExchange'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerHelloDone'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.1 Handshake, ClientKeyExchange'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.1 Handshake, Finished'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 ChangeCipherSpec'
(43)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Finished'
(43)   &reply::TLS-Session-Cipher-Suite += &session-state:TLS-Session-Cipher-Suite[*] -> 'ECDH
E-RSA-AES256-SHA'
(43)   &reply::TLS-Session-Version += &session-state:TLS-Session-Version[*] -> 'TLS 1.1'
(43)   } # update = noop
(43)   [exec] = noop
(43)   policy remove_reply_message_if_eap {
(43)     if (&reply:EAP-Message && &reply:Reply-Message) {
(43)       if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE
(43)     } else {
(43)       [noop] = noop
(43)     } # else = noop
(43)   } # policy remove_reply_message_if_eap = noop
(43)   if (EAP-Key-Name && &reply:EAP-Session-Id) {
(43)     if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE
(43)   } # post-auth = noop
(43) Login OK: [test/<via Auth-Type = eap>] (from client routerplz port 0 cli 00-E0-4C-B8-4B-57)
(43) Sent Access-Accept Id 81 from 192.168.56.100:1812 to 192.168.56.189:54219 length 172
(43) MS-MPPE-Recv-Key = 0x9f23517cb4e33a71eb8779277ba46a0f8db98d7f3f4a8af3db9d7b2f1aed0e40
(43) MS-MPPE-Send-Key = 0x74c53557c9641be1f4f3774a911b3afadd03a2edb20191865bc62bf875b042cf
(43) EAP-Message = 0x03090004
(43) Message-Authenticator = 0x00000000000000000000000000000000
(43) User-Name = "test"
(43) Framed-MTU += 994
(43) Finished request

```

Fonte: Autor (2025).

Figura 29: Captura de logs no freeradius após a conexão com o *suplicant* GTC.

```

(33) eap_gtc: # Executing group from file /etc/freeradius/3.0/sites-enabled/inner-tunnel
(33) eap_gtc: Auth-Type PAP {
(33) pap: Login attempt with password
(33) pap: Comparing with "known good" Cleartext-Password
(33) pap: User authenticated successfully
(33) eap_gtc: [pap] = ok
(33) eap_gtc: } # Auth-Type PAP = ok
(33) eap: Sending EAP Success (code 3) ID 9 length 4
(33) eap: Freeing handler
(33) [eap] = ok
(33) } # authenticate = ok
(33) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/inner-tunnel
(33) post-auth {
(33)   if (0) {
(33)     if (0) -> FALSE
(33)   } # post-auth = noop
(33) Login OK: [test/test123] (from client routerplz port 0 via TLS tunnel)
(33) } # server inner-tunnel
(33) Virtual server sending reply
(33) EAP-Message = 0x03090004
(33) Message-Authenticator = 0x00000000000000000000000000000000
(33) User-Name = "test"
(33) eap_peap: Got tunneled reply code 2
(33) eap_peap: EAP-Message = 0x03090004
(33) eap_peap: Message-Authenticator = 0x00000000000000000000000000000000
(33) eap_peap: User-Name = "test"
(33) eap_peap: Got tunneled reply RADIUS code 2
(33) eap_peap: EAP-Message = 0x03090004
(33) eap_peap: Message-Authenticator = 0x00000000000000000000000000000000
(33) eap_peap: User-Name = "test"
(33) eap_peap: Tunneled authentication was successful
(33) eap_peap: SUCCESS

```

Fonte: Autor (2025).

Figura 34: Captura dos logs do RADIUS durante o ataque ao Android

```

(23)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - recv TLS 1.1 Handshake, Finished'
(23)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 ChangeCipherSpec'
(23)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Finished'
(23)      &reply::TLS-Session-Cipher-Suite += &session-state:TLS-Session-Cipher-Suite[*] -> 'ECDH
E-RSA-AES128-SHA'
(23)      &reply::TLS-Session-Version += &session-state:TLS-Session-Version[*] -> 'TLS 1.1'
(23)      } # update = noop
(23)      [exec] = noop
(23)      policy remove_reply_message_if_eap {
(23)      if (&reply:EAP-Message && &reply:Reply-Message) {
(23)      if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE
(23)      else {
(23)      [noop] = noop
(23)      } # else = noop
(23)      } # policy remove_reply_message_if_eap = noop
(23)      if (EAP-Key-Name && &reply:EAP-Session-Id) {
(23)      if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE
(23)      } # post-auth = noop
(23) Login OK: [rian.block/cvia Auth-Type = eap] (from client routerplz port 0 cli 62-49-98-90-6F
-3F)
(23) Sent Access-Accept Id 23 from 192.168.56.100:1812 to 192.168.56.189:35823 length 178
(23) MS-MPPE-Recv-Key = 0x87624395aa3d3514f1eba6eca3a1b7d99f6676bae650c184f5e182c48616d170
(23) MS-MPPE-Send-Key = 0x436156d5a99a43e500478bfc66738ecd8434f75b0bd0222c39c102dbc4dcbc9d
(23) EAP-Message = 0x03080004
(23) Message-Authenticator = 0x00000000000000000000000000000000
(23) User-Name = "rian.block"
(23) Framed-MTU += 994

```

Fonte: Autor (2025).

Figura 38: Captura de logs RADIUS durante o ataque no iPhone.

```

New Tab Split View Copy Paste Find...
(7)      if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name ==
request:User-Name)) -> FALSE
(7)      update {
(7)      &reply::Framed-MTU += &session-state:Framed-MTU[*] -> 994
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - recv TLS 1.3 Handshake, ClientHello'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - send TLS 1.1 Handshake, ServerHello'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - send TLS 1.1 Handshake, Certificate'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - send TLS 1.1 Handshake, ServerKeyExchange'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - send TLS 1.1 Handshake, ServerHelloDone'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - recv TLS 1.1 Handshake, ClientKeyExchange'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - recv TLS 1.1 Handshake, Finished'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - send TLS 1.1 ChangeCipherSpec'
(7)      &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS P
EAP - send TLS 1.1 Handshake, Finished'
(7)      &reply::TLS-Session-Cipher-Suite += &session-state:TLS-Session-Cipher-Suite[*] -> 'ECDHE
-RSA-AES256-SHA'
(7)      &reply::TLS-Session-Version += &session-state:TLS-Session-Version[*] -> 'TLS 1.1'
(7)      } # update = noop
(7)      [exec] = noop
(7)      policy remove_reply_message_if_eap {
(7)      if (&reply:EAP-Message && &reply:Reply-Message) {
(7)      if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE
(7)      else {
(7)      [noop] = noop
(7)      } # else = noop
(7)      } # policy remove_reply_message_if_eap = noop
(7)      if (EAP-Key-Name && &reply:EAP-Session-Id) {
(7)      if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE
(7)      } # post-auth = noop
(7) Login OK: [rian.block@ufsc.br/cvia Auth-Type = eap] (from client routerplz port 0 cli 2E-DF-8
5-2B-44-90)
(7) Sent Access-Accept Id 71 from 192.168.56.100:1812 to 192.168.56.189:35823 length 186
(7) MS-MPPE-Recv-Key = 0x47e3b4f90014117fef327c81ef20178820c01fd344cfe06244edfc56d5ea5641
(7) MS-MPPE-Send-Key = 0x6412c226f676a8c5f4f157815cd728c29309858980e55b2229b2f386e931868c
(7) EAP-Message = 0x03080004
(7) Message-Authenticator = 0x00000000000000000000000000000000
(7) User-Name = "rian.block@ufsc.br"
(7) Framed-MTU += 994
(7) Finished request
Waking up in 4.9 seconds.
(3) Cleaning up request packet ID 67 with timestamp +20 due to cleanup_delay was reached

```

Fonte: Autor (2025).

Figura 40: Captura dos logs do servidorRADIUS durante a conexão com o tablet.

```

New Tab Split View Copy Paste Find...
(31) if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name =
= request:User-Name)) -> FALSE
(31) update {
(31)   &reply::Framed-MTU += &session-state:Framed-MTU[*] -> 994
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - rcv TLS 1.3 Handshake, ClientHello'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerHello'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Certificate'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerKeyExchange'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, ServerHelloDone'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - rcv TLS 1.1 Handshake, ClientKeyExchange'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - rcv TLS 1.1 Handshake, Finished'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 ChangeCipherSpec'
(31)   &reply::TLS-Session-Information += &session-state:TLS-Session-Information[*] -> '(TLS
PEAP - send TLS 1.1 Handshake, Finished'
(31)   &reply::TLS-Session-Cipher-Suite += &session-state:TLS-Session-Cipher-Suite[*] -> 'ECDH
E-RSA-AES128-SHA'
(31)   &reply::TLS-Session-Version += &session-state:TLS-Session-Version[*] -> 'TLS 1.1'
(31) } # update = noop address: you can
(31) [exec] = noop
(31) policy remove_reply_message_if_eap {
(31)   if (&reply:EAP-Message && &reply:Reply-Message) {
(31)     if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE
(31)   } else {
(31)     [noop] = noop > permission to access the web (you might be connected but behind a firewall)
(31)   } # else = noop
(31) } # policy remove_reply_message_if_eap = noop
(31) if (EAP-Key-Name && &reply:EAP-Session-Id) {
(31)   if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE
(31) } # post-auth = noop
(31) Login OK: [11510530908/<via Auth-Type = eap>] (from client routerplz port 0 cli EA-4D-3F-3F-7
9-B5)
(31) Sent Access-Accept Id 95 from 192.168.56.100:1812 to 192.168.56.189:35823 length 179
(31) MS-MPPE-Recv-Key = 0xfda8bb65007452b005ceff88a72b5348e080a47e7805ef3d0cdaa0038463a9b6
(31) MS-MPPE-Send-Key = 0xfb6a3dd365b59ce235f4a31b1bcc045e7089954f5d77be72fb8f2b19bc3c4bf9
(31) EAP-Message = 0x03080004
(31) Message-Authenticator = 0x00000000000000000000000000000000
(31) User-Name = "11510530908"
(31) Framed-MTU += 994
(31) Finished request
Waking up in 4.9 seconds.
(24) Cleaning up request packet ID 88 with timestamp +360 due to cleanup_delay was reached

```

Fonte: Autor (2025).

Figura 44: Captura dos logs do freeRADIUS durante a conexão com o MacOS.

```

(27) [pap] = noop
(27) } # authorize = updated
(27) Found Auth-Type = Accept
(27) Auth-Type = Accept, accepting the user
(27) # Executing section post-auth from file /etc/freeradius/3.0/sites-enabled/default
(27) post-auth {
(27)   if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name =
= request:User-Name)) {
(27)     if (session-state:User-Name && reply:User-Name && request:User-Name && (reply:User-Name =
= request:User-Name)) -> FALSE
(27)   }
(27)   update {
(27)     No attributes updated for RHS &session-state:
(27)   } # update = noop
(27)   [exec] = noop
(27)   policy remove_reply_message_if_eap {
(27)     if (&reply:EAP-Message && &reply:Reply-Message) {
(27)       if (&reply:EAP-Message && &reply:Reply-Message) -> FALSE
(27)     } else {
(27)       [noop] = noop > permission to access the web (you might be connected but behind a firewall)
(27)     } # else = noop
(27)   } # policy remove_reply_message_if_eap = noop
(27)   if (EAP-Key-Name && &reply:EAP-Session-Id) {
(27)     if (EAP-Key-Name && &reply:EAP-Session-Id) -> FALSE
(27)   } # post-auth = noop
(27) } # post-auth = noop
(27) Login OK: [rian.block/<via Auth-Type = Accept>] (from client routerplz port 0 cli 42-DD-CB-B4
-A5-EF)
(27) Sent Access-Accept Id 167 from 192.168.56.100:1812 to 192.168.56.189:35823 length 38
(27) Finished request
Waking up in 4.9 seconds.
(26) Cleaning up request packet ID 166 with timestamp +69 due to cleanup_delay was reached
(27) Cleaning up request packet ID 167 with timestamp +69 due to cleanup_delay was reached
Ready to process requests

```

Fonte: Autor (2025).

Figura 48: Captura dos logs RADIUS durante ataque ao Windows.

```

(7) [eap] = ok
(7) } # authorize = ok
(7) Found Auth-Type = eap
(7) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(7) authenticate {
(7) eap: Removing EAP session with state 0xe71df0bae114e92e
(7) eap: Previous EAP request found for state 0xe71df0bae114e92e, released from the list
(7) eap: Peer sent packet with method EAP PEAP (25)
(7) eap: Calling submodule eap_peap to process data
(7) eap_peap: (TLS) EAP Done initial handshake
(7) eap_peap: Session established. Decoding tunneled attributes
(7) eap_peap: PEAP state send tlv success
(7) eap_peap: Received EAP-TLV response
(7) eap_peap: Client rejected our response. The password is probably incorrect
(7) eap_peap: ERROR: We sent a success, but the client did not agree
(7) eap: ERROR: Failed continuing EAP PEAP (25) session. EAP sub-module failed
(7) eap: Sending EAP Failure (code 4) ID 9 length 4
(7) eap: Failed in EAP select
(7) [eap] = invalid
(7) } # authenticate = invalid
(7) Failed to authenticate the user
(7) Using Post-Auth-Type Reject
(7) # Executing group from file /etc/freeradius/3.0/sites-enabled/default
(7) Post-Auth-Type REJECT {
(7) attr_filter.access_reject: EXPAND %{User-Name}
(7) attr_filter.access_reject: rian.block
(7) attr_filter.access_reject: Matched entry DEFAULT at line 11
(7) [attr_filter.access_reject] = updated
(7) [eap] = noop
(7) policy remove_reply_message_if_eap {
(7) if ($reply:EAP-Message && $reply:Reply-Message) {
(7) if ($reply:EAP-Message && $reply:Reply-Message) -> FALSE
(7) else {
(7) [noop] = noop
(7) } # else = noop
(7) } # policy remove_reply_message_if_eap = noop
(7) } # Post-Auth-Type REJECT = updated
(7) Login incorrect (eap_peap: We sent a success, but the client did not agree): [rian.block] (from client routerplz port 0 cli 00-E0-4C-B8-4B-57)
(7) Delaying response for 0.999319 seconds
Waking up in 0.3 seconds.
Waking up in 0.6 seconds.
(7) (7) Discarding duplicate request from client routerplz port 35823 - ID: 115 due to delayed response
(7) Sending delayed response
(7) Sent Access-Reject Id 115 from 192.168.56.100:1812 to 192.168.56.189:35823 length 44
(7) EAP-Message = 0x04090004
(7) Message-Authenticator = 0x00000000000000000000000000000000
Waking up in 3.9 seconds.

```

Fonte: Autor (2025).

APÊNDICE B

Artigo científico sobre o TCC no formato definido pela Sociedade Brasileira de Computação (SBC).

Segurança da autenticação no eduroam: análise experimental de vulnerabilidades em EAP e RADIUS

Rian Block Serena¹, Carla Merkle Westphall¹, Eduardo Freitas Hoffmann¹

¹Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
Florianópolis – SC – Brasil

rian.serena@grad.ufsc.br, carla.westphall@ufsc.br, eduardo.hoffmann@ufsc.br

Abstract. *Eduroam is a global federated Wi-Fi access service widely used in universities. Its authentication relies on IEEE 802.1X, EAP methods and a hierarchy of RADIUS servers. In theory, TLS-based EAP methods provide strong protection, but in practice misconfigured client profiles and relaxed certificate validation can make evil twin and man-in-the-middle attacks feasible.*

This paper reports an experimental study that reproduces, in an isolated test-bed, the 802.1X authentication flow of the eduroam configuration used at UFSC. Using FreeRADIUS, a rogue access point and real devices (Android, iOS, macOS and Windows), we evaluate EAP-MD5, EAP-TTLS with PAP and GTC, and PEAP-MSCHAPv2 and PEAP-GTC; EAP-TLS and EAP-FAST are discussed conceptually due to client-side deployment limitations. Results show that correct server-certificate validation prevents credential disclosure, while Android profiles configured without validation automatically connect to rogue RADIUS servers, leaking PAP passwords in cleartext and exposing MSCHAPv2 responses to offline attacks. Additional experiments with TTLS/PEAP using GTC in a compatible supplicant confirm that GTC also exposes passwords in cleartext when server validation is disabled. We summarize concrete countermeasures for identity providers and administrators.

Keywords: eduroam, RADIUS, EAP, WPA-Enterprise, security.

Resumo. *O eduroam é um serviço mundial de acesso Wi-Fi federado amplamente utilizado em universidades. A autenticação baseia-se em IEEE 802.1X, em métodos EAP e em uma hierarquia de servidores RADIUS. Em teoria, métodos EAP encapsulados em TLS oferecem proteção robusta, mas perfis de clientes mal configurados e a desativação da validação de certificados tornam viáveis ataques do tipo evil twin e man-in-the-middle.*

Este artigo apresenta um estudo experimental que reproduz, em ambiente isolado, o fluxo de autenticação 802.1X da configuração eduroam utilizada na UFSC. Utilizando FreeRADIUS, um ponto de acesso malicioso e dispositivos reais (Android, iOS, macOS e Windows), são avaliados na prática os métodos EAP-MD5, EAP-TTLS com PAP e GTC, e PEAP-MSCHAPv2 e PEAP-GTC; os métodos EAP-TLS e EAP-FAST são discutidos conceitualmente devido a limitações de implantação nos clientes. Os resultados mostram que a validação correta do certificado do servidor impede o vazamento de credenciais, enquanto perfis Android sem validação conectam-se automaticamente a servidores RADIUS maliciosos, vazando senhas em claro em PAP e expondo respostas

MSCHAPv2 a ataques offline. Experimentos adicionais com TTLS/PEAP-GTC em supplicant compatível mostram que o GTC também expõe senhas em claro quando a validação do servidor é desativada. São sintetizadas contramedidas objetivas para provedores de identidade e administradores.

Palavras-chave: eduroam, RADIUS, EAP, WPA-Enterprise, segurança.

1. Introdução

O eduroam permite que usuários de instituições de ensino e pesquisa utilizem suas credenciais institucionais para acessar redes Wi-Fi em milhares de pontos no mundo. A solução apoia-se no padrão IEEE 802.1X para controle de acesso baseado em portas, no Extensible Authentication Protocol (EAP) para negociação de métodos de autenticação e em uma hierarquia de servidores Remote Authentication Dial-In User Service (RADIUS) para encaminhar pedidos até o provedor de identidade de origem [2, 1].

Na especificação, a combinação entre 802.1X, EAP e RADIUS, com túneis TLS e autenticação mútua, oferece propriedades fortes de confidencialidade e integridade. Na prática, porém, a segurança efetiva depende de detalhes de configuração em clientes e servidores. Perfis que permitem “não validar certificado”, tutoriais desatualizados e mensagens de alerta pouco claras criam condições em que um ponto de acesso malicioso pode ser aceito como legítimo e, a partir daí, capturar credenciais sensíveis.

Ataques do tipo *evil twin*, muitas vezes combinados com *man-in-the-middle*, exploram exatamente esse cenário: um atacante cria um AP com o SSID *eduroam*, fornece sinal mais forte que o AP legítimo e opera um servidor RADIUS fraudulento. Se o dispositivo vítima não valida corretamente o certificado do servidor, o túnel TLS é estabelecido com o atacante e métodos internos (*inner methods*) como PAP, GTC e MSCHAPv2 tornam-se visíveis no servidor malicioso.

Este artigo, derivado do Trabalho de Conclusão de Curso do autor, apresenta uma avaliação experimental em ambiente controlado da segurança prática da autenticação no eduroam. O foco é responder em que condições um servidor RADIUS malicioso obtém credenciais ou material sensível, qual o impacto da validação de certificados em clientes Android, iOS, macOS e Windows, e quais recomendações objetivas podem ser adotadas por provedores de identidade e administradores de rede para mitigar esses riscos.

2. Fundamentos e trabalhos relacionados

O eduroam apoia-se em três pilares técnicos: o protocolo Extensible Authentication Protocol (EAP), definido na RFC 3748 [2], o RADIUS como infraestrutura de autenticação, autorização e contabilização [1, 14], e o uso de 802.1X para controle de acesso em redes sem fio. Em redes WPA/WPA2-Enterprise, o suplicante usa EAP over LAN (EAPoL) até o ponto de acesso, que encapsula as mensagens em RADIUS e as encaminha ao servidor de autenticação. A arquitetura global de *network roaming*, com proxies nacionais e regionais roteando pedidos pelo *realm* do usuário, é descrita em documentos de arquitetura específicos do eduroam e em whitepapers industriais [10].

Diretrizes de segurança atuais recomendam o uso de métodos EAP baseados em TLS, com autenticação do servidor e, idealmente, mútua, e a desativação de métodos legados sem proteção adequada [3, 7, 8]. No plano federado, o uso de RADIUS over TLS

(RadSec) [4] protege metadados de autenticação entre proxies, enquanto guias técnicos do FreeRADIUS detalham configurações seguras e práticas de operação [5].

Estudos empíricos mostram que o principal vetor de ataque hoje não é a quebra da criptografia, mas sim perfis de clientes e tutoriais que permitem desativar a validação de certificados. Vanhoef e Piessens [12] demonstram, em redes eduroam reais, que configurações sem validação de certificado permitem a captura de credenciais por um AP malicioso. Fiebrig et al. [13] reforçam esse quadro com análises experimentais de ataques de roubo de credenciais, destacando a exposição de dispositivos Android. Como resposta operacional, a comunidade mantém o eduroam Configuration Assistant Tool (CAT), que gera perfis para múltiplos sistemas operacionais com CA raiz e nomes de servidor amarrados à política de cada IdP [11], reduzindo o espaço para erros manuais de configuração.

3. Metodologia experimental

A metodologia adotada é de natureza exploratória e experimental: construiu-se um ambiente de testes controlado que reproduz o fluxo de autenticação 802.1X da configuração eduroam da UFSC, com foco em ataques *evil twin* combinados com *man-in-the-middle*. Todos os experimentos foram realizados em rede isolada, sem interação com a infraestrutura eduroam em produção.

3.1. Infraestrutura e ferramentas

O servidor de autenticação foi implementado com FreeRADIUS em máquina virtual Kali Linux, conectada a um roteador físico configurado como ponto de acesso WPA2-Enterprise com SSID *eduroam*, e monitorada por Wireshark na máquina hospedeira; FreeRADIUS; Kali Linux em máquina virtual; roteador Wi-Fi em modo WPA2-Enterprise; Wireshark para captura de tráfego e análise de EAPOL e RADIUS; logs detalhados do servidor para inspeção dos atributos recebidos.

A interface de rede da máquina virtual foi configurada em modo *bridge*, na mesma sub-rede IPv4 do AP, de modo que as mensagens RADIUS trafegassem de forma idêntica a um cenário real, apenas isoladas da infraestrutura de produção.

3.2. Dispositivos de teste

Foram utilizados dispositivos pessoais reais, previamente configurados para acesso ao eduroam ou capazes de fazê-lo: smartphone Android; tablet Android; dois dispositivos iOS; notebook macOS; notebook Windows. Em parte dos dispositivos Android havia perfis configurados sem validação de certificado, reproduzindo o cenário crítico analisado no TCC; iOS, macOS e Windows utilizavam perfis que exigiam validação explícita do certificado do servidor.

3.3. Cenários experimentais

Os experimentos foram organizados em dois cenários principais:

1. **Cenário legítimo:** o AP experimental anuncia o SSID *eduroam* e encaminha as mensagens EAP ao servidor FreeRADIUS configurado como servidor de testes. Observa-se, via Wireshark, o que um atacante passivo na mesma LAN consegue enxergar durante o processo de autenticação, bem como o que fica registrado em logs do servidor legítimo.

2. **Cenário *evil twin* + MITM:** o mesmo AP passa a representar um ponto de acesso malicioso. O FreeRADIUS é configurado para aceitar qualquer credencial e registrar integralmente os atributos recebidos. Variam-se dois parâmetros principais: (i) se o cliente valida ou não o certificado do servidor, e (ii) qual método EAP está configurado no perfil do dispositivo, conforme descrito nas seções anteriores.

Para cada combinação de dispositivo, método EAP e política de certificado, foram coletados: capturas de pacotes, registros de servidor e capturas de tela das mensagens exibidas ao usuário, compondo o conjunto de evidências que fundamenta os resultados apresentados a seguir.

4. Resultados principais

4.1. Validação de certificados como fator determinante

Em todos os métodos encapsulados em TLS avaliados (TTLS, PEAP com MSCHAPv2 e com GTC), observou-se o mesmo padrão: quando a validação de certificados estava ativa e corretamente configurada, os dispositivos recusaram conexões com o servidor malicioso ou exigiram confirmação explícita do usuário. Nessas condições, um atacante passivo só enxerga tráfego TLS cifrado; senhas e tokens permanecem protegidos tanto na captura de pacotes quanto nos logs do servidor legítimo.

Quando a validação de certificados foi desativada, especialmente em perfis Android com a opção “não validar certificado”, o comportamento mudou de forma radical. Os dispositivos conectaram-se automaticamente ao AP malicioso e concluíram a autenticação com o servidor fraudulento, sem exibir novos avisos, reproduzindo o cenário mais grave identificado na etapa experimental do TCC.

4.2. Exposição de métodos internos

Do ponto de vista do servidor malicioso, o grau de exposição variou conforme o método interno (*inner method*) utilizado:

- **EAP-MD5:** o servidor obteve o desafio e a resposta MD5, calculada como

$$MD5(ID \parallel senha \parallel challenge),$$

suficiente para ataques de força bruta *offline*, agravados pela fragilidade conhecida do MD5. Como o atacante controla o servidor, pode ainda fixar o *challenge*, facilitando o uso de tabelas pré-computadas.

- **EAP-TTLS com PAP:** com validação de certificado, nenhum dado sensível aparece na captura de tráfego; o túnel TLS protege completamente o inner PAP. Quando o cliente aceita o certificado autoassinado do servidor malicioso, o túnel termina no atacante e o PAP passa a ser processado em texto claro. Os logs do servidor registram diretamente: `User-Name` e `User-Password`.
- **PEAP-MSCHAPv2:** tanto na captura de tráfego quanto no servidor malicioso, a senha nunca é exibida em texto claro. O atacante obtém apenas o par desafio-resposta do MSCHAPv2, baseado no hash NTLM da senha. Esse material pode ser submetido a ataques *offline*, mas exige esforço computacional maior e depende da política de senhas da instituição.

- **PEAP-GTC e TTLS-GTC:** em ambos os casos, o inner GTC transmite as credenciais em texto claro dentro do túnel TLS. Quando o túnel termina no servidor fraudulento por ausência de validação de certificado, o atacante passa a registrar integralmente usuário e senha, como observado nos logs do FreeRADIUS na etapa experimental do TCC.
- **EAP-TLS:** o método foi corretamente configurado no FreeRADIUS com CA própria e certificados de servidor, mas a transferência e instalação dos certificados de cliente nos dispositivos de teste não pôde ser concluída. Assim, não houve autenticações bem-sucedidas com EAP-TLS no ambiente de laboratório. A análise permanece conceitual: por não depender de senhas, EAP-TLS elimina a exposição de credenciais textuais, mas requer uma infraestrutura de emissão e gerenciamento de certificados de cliente que poucas instituições mantêm em escala [3].

O EAP-FAST também foi configurado no servidor, mas nenhum dos dispositivos disponíveis oferecia suporte a esse método, impedindo sua avaliação prática. Dado seu uso restrito a ambientes corporativos específicos, foi considerado fora do escopo principal da análise.

4.3. Diferenças entre sistemas operacionais

Os testes apontaram diferenças nítidas entre plataformas:

- **Android:** quando o perfil eduroam foi configurado com a opção “não validar certificado”, o dispositivo aceitou automaticamente o certificado autoassinado do servidor malicioso e concluiu a autenticação com o AP falso, sem qualquer alerta adicional. Mesmo assim, o supplicant recusou tentativas de *downgrade* de inner method: o dispositivo não passou de PEAP-MSCHAPv2 para PAP ou GTC, preferindo falhar a aceitar métodos internos mais fracos.
- **iOS e macOS:** os dispositivos não se conectaram automaticamente à rede falsa. Sempre que o servidor apresentou um certificado não confiável, o sistema exibiu avisos claros solicitando confirmação explícita do usuário. Tal como no Android, não houve *downgrade* de inner method: quando o servidor malicioso anunciou apenas métodos internos inseguros, a conexão falhou.
- **Windows:** embora o equipamento utilizado não tivesse o perfil eduroam original instalado, o comportamento foi similar ao de iOS e macOS. O sistema não aceitou negociar métodos internos inseguros e exibiu erros de incompatibilidade ou de certificado inválido quando o servidor tentou forçar a utilização de configurações fora do padrão.

Em resumo, todos os sistemas impediram *downgrade* de métodos internos, mas apenas Android permitiu, por configuração explícita do usuário, conexões automáticas a servidores com certificados não validados, reproduzindo exatamente o cenário mais crítico descrito no TCC.

5. Recomendações e conclusão

Os resultados obtidos reforçam a conclusão central do TCC: o elo mais fraco na segurança prática do eduroam não é o desenho criptográfico de EAP, TLS ou RADIUS, mas sim as opções de configuração oferecidas a usuários e administradores.

Do ponto de vista de IdPs e equipes de rede, destacam-se as seguintes recomendações:

- **Certificados e eduroam CAT:** eliminar, em tutoriais e suporte, qualquer instrução que sugira desativar a validação de certificados. Utilizar o eduroam CAT como meio padrão de distribuição de perfis, incluindo CA raiz, nomes de servidor (CN/SAN) e métodos EAP suportados, impedindo que o usuário final escolha “não validar certificado” [11].
- **Endurecimento de métodos EAP:** desabilitar métodos legados como EAP-MD5, LEAP, PAP e GTC nos servidores RADIUS. Nas redes que ainda dependem de senhas, restringir a configuração a PEAP-MSCHAPv2 bem configurado, com políticas de senha fortes; quando operacionalmente viável, planejar a adoção gradual de EAP-TLS.
- **Proteção do plano federado:** adotar RadSec (RADIUS over TLS) entre proxies nacionais, regionais e servidores de topo, com validação mútua de certificados [4, 10]. Isso reduz a exposição de atributos de identidade e protege o caminho federado contra observação de terceiros.
- **Privacidade de identidade:** configurar *outer identities* anônimas (por exemplo, anon@instituicao.br), preservando o nome de usuário real apenas na identidade interna protegida pelo túnel TLS, reduzindo vazamento de metadados em proxies intermediários.
- **Políticas de senha:** reforçar políticas de complexidade e comprimento mínimo para credenciais utilizadas em MSCHAPv2, e evitar que a mesma senha seja reaproveitada em serviços institucionais críticos, mitigando o impacto de ataques *offline*.

Como trabalho futuro, a ampliação da amostra de dispositivos, a análise de perfis eduroam reais de múltiplas instituições e a investigação de arquiteturas baseadas em identidades auto-soberanas podem aprofundar a discussão sobre como combinar segurança, privacidade e usabilidade na autenticação federada.

Em síntese, a experiência em laboratório confirma que a segurança prática do eduroam depende menos da introdução de novos protocolos e mais da eliminação de escolhas inseguras na experiência de configuração, em especial nos dispositivos Android. A adoção sistemática das recomendações acima fortalece a resiliência da infraestrutura frente a ataques de *evil twin* e *man-in-the-middle* em ambientes acadêmicos.

Referências

- [1] RIGNEY, C. et al. *RFC 2865 – Remote Authentication Dial In User Service (RADIUS)*. IETF, 2000.
- [2] ABOULMAGD, O. et al. *RFC 3748 – Extensible Authentication Protocol (EAP)*. IETF, 2004.
- [3] SIMMONS, R. et al. *RFC 5216 – The EAP-TLS Authentication Protocol*. IETF, 2008.
- [4] IETF. *RFC 6614 – RADIUS over TLS*. IETF, 2012.
- [5] FREERADIUS PROJECT. *FreeRADIUS Technical Guide*. Available at: <https://networkradius.com/doc/>.
- [6] VAN DER WALT, D. *FreeRADIUS Beginner’s Guide*. Packt Publishing, 2011.
- [7] FERGUSON, B.; KAUFMAN, C. *Wi-Fi Security: Mastering 802.1X, RADIUS, and EAP*. Pearson Education, 2014.

- [8] NELSON, M. et al. *CWSP – Certified Wireless Security Professional Official Study Guide*. 2nd ed., Wiley, 2017.
- [9] OPPLIGER, R. *Criptografia e Segurança de Redes*. 6. ed., Rio de Janeiro: LTC, 2014.
- [10] GEUER-POLTHIER, S. et al. *Roam Whitepaper v1.0*. 2022.
- [11] *A guide to eduroam CAT for IdP administrators*. Wiki GÉANT, 2023. Available at: <https://wiki.geant.org/spaces/H2eduroam/pages/121346267/A+guide+to+eduroam+CAT+for+IdP+administrators>. Accessed: 04 Dec. 2025.
- [12] VANHOEF, M.; PIESSENS, F. A Practical Investigation of Identity Theft Vulnerabilities in Eduroam. In: *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks (WiSec)*, 2015.
- [13] FIEBRIG, D. et al. Careful with that Roam, Edu: Experimental Analysis of Eduroam Credential Stealing Attacks. 2019.
- [14] ABOBA, B.; VOLLBRECHT, J. *RFC 3579 – RADIUS Support for EAP*. IETF, 2003.