INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

Flávio de Souza Lima

AVALIAÇÃO DE RISCOS DE SEGURANÇA CIBERNÉTICA NO SOFTWARE SAHANA EDEN DISASTER MANAGEMENT SYSTEM UTILIZANDO CONCEITOS DE HACKER ÉTICO

Campos dos Goytacazes/RJ

2023

2023 FLAVIO DE SOUZA LIMA MPSAEG / IFF

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

FLÁVIO DE SOUZA LIMA

AVALIAÇÃO DE RISCOS DE SEGURANÇA CIBERNÉTICA NO SOFTWARE SAHANA EDEN DISASTER MANAGEMENT SYSTEM UTILIZANDO CONCEITOS DE HACKER ÉTICO

Renato Gomes Sobral Barcellos

(Orientador)

Rogério Atem de Carvalho

(Coorientador)

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de Mestre em Sistemas Aplicados à Engenharia e Gestão.

Campos dos Goytacazes/RJ

2023

Biblioteca Anton Dakitsch

CIP – Catalogação na Publicação

L732a

Lima, Flávio de Souza
AVALIAÇÃO DE RISCOS DE SEGURANÇA CIBERNÉTICA NO SOFTWARE SAHANA EDEN DISASTER MANAGEMENT SYSTEM UTILIZANDO CONCEITOS DE HACKER ÉTICO / Flávio de Souza Lima - 2023.
126f.: il. color.

Orientador: Renato Gomes Sobral Barcellos
Coorientador: Rogério Atem de Carvalho

Dissertação (mestrado) - Instituto Federal de Educação, Ciência e Tecnologia Fluminense, Campus Campos Centro, Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão, Anton Dakitsch, RJ, 2023.
Referências: f. 123 a 126.

1. Sahana Eden. 2. Cibersegurança. 3. Vulnerabilidades. 4. Pentest. 5. Segurança da Informação. I. Barcellos, Renato Gomes Sobral, orient. II. Carvalho, Rogério Atem de, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da Biblioteca Anton Dakitsch do IFF com os dados fornecidos pelo(a) autor(a).

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA FLUMINENSE

PROGRAMA DE PÓS-GRADUAÇÃO EM SISTEMAS APLICADOS À ENGENHARIA E GESTÃO

Flávio de Souza Lima

AVALIAÇÃO DE RISCOS DE SEGURANÇA CIBERNÉTICA NO SOFTWARE SAHANA EDEN DISASTER MANAGEMENT SYSTEM UTILIZANDO CONCEITOS DE HACKER ÉTICO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação do Instituto Federal de Educação, Ciência e Tecnologia Fluminense, no Curso de Mestrado Profissional em Sistemas Aplicados à Engenharia e Gestão (MPSAEG), como parte dos requisitos necessários à obtenção do título de Mestre em Sistemas Aplicados à Engenharia e Gestão.

Aprovado(a) em 14 de setembro de 2023.

Banca Examinadora:

Documento assinado digitalmente
gov.br RENATO GOMES SOBRAL BARCELLOS
Data: 12/11/2023 15:10:21-0300
Verifique em https://validar.iti.gov.br

Renato Gomes Sobral Barcellos, Doutor em Geociências.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense
(Orientador)

Documento assinado digitalmente
gov.br ROGERIO ATEM DE CARVALHO
Data: 13/11/2023 10:29:03-0300
Verifique em https://validar.iti.gov.br

Rogério Atem de Carvalho, Doutor em Engenharia de Produção.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense

(Coorientador)

Documento assinado digitalmente
gov.br CLAUDIO MICELI DE FARIAS
Data: 27/11/2023 11:24:35-0300
Verifique em https://validar.iti.gov.br

Cláudio Miceli de Farias, Doutor em Informática.
Universidade Federal do Rio de Janeiro

Documento assinado digitalmente
gov.br DANIEL VASCONCELOS CORREA DA SILVA
Data: 12/11/2023 15:15:39-0300
Verifique em https://validar.iti.gov.br

Daniel Vasconcelos Corrêa da Silva, Doutor em Computação.
Instituto Federal de Educação, Ciência e Tecnologia Fluminense

Dedico esta dissertação de mestrado à comunidade científica e aos profissionais da área de segurança da informação, que dedicam suas vidas ao estudo e à prática de proteger nossos sistemas e dados valiosos. Agradeço por sua perseverança e dedicação em garantir a privacidade e a segurança dos dados confidenciais em um mundo cada vez mais digitalizado. Espero que este trabalho possa contribuir para o avanço do conhecimento e para o desenvolvimento de soluções cada vez mais eficazes e inovadoras para a segurança da informação.

AGRADECIMENTOS

Gostaria de expressar minha gratidão a Deus, que me concedeu sabedoria e iluminou meu caminho para que eu pudesse chegar até aqui com saúde e perseverança. Sem a sua energia eu não teria conseguido superar os desafios que encontrei nesta jornada.

Também gostaria de agradecer à minha família, meus pais por me ensinarem os principais valores da educação e dos estudos, meus filhos, Gabriel e Eric por compreenderem minha ausência nos momentos que me dediquei a este trabalho e especialmente à minha esposa Maísa, que sempre esteve ao meu lado durante os momentos difíceis, compartilhando o quarto e o seu tempo de descanso comigo enquanto eu trabalhava neste projeto. Sua paciência, apoio e amor foram fundamentais para que eu pudesse seguir em frente e me concentrar nos estudos.

Gostaria de estender minha gratidão ao meu companheiro de trabalho, Vander Silva, que me incentivou e apoiou para que eu pudesse ingressar no Mestrado e alcançar meus objetivos. Sua colaboração e suporte foram essenciais para minha jornada.

Por fim, quero expressar minha sincera gratidão ao meu orientador Doutor e Professor Renato Barcellos, que dedicou seu tempo e conhecimento para me guiar nesta jornada. Ele sempre esteve disponível para esclarecer minhas dúvidas e me apoiar em todas as etapas do trabalho, independentemente do horário. Seu comprometimento e apoio foram fundamentais para o sucesso deste projeto.

E, é claro, não posso esquecer de agradecer às minhas cafeteiras de espresso que me forneceram energia quando o esgotamento chegava. Agradeço a todos que contribuíram para esta conquista e espero retribuir essa confiança com trabalho e dedicação contínuos.

RESUMO

Devido aos crescentes ataques cibernéticos nos últimos anos tornou-se necessária a prática de análise de segurança do ambiente tecnológico pelas organizações em busca de fragilidades e mitigá-las. **SAHANA EDEN** é um sistema de gestão de desastres utilizado com a missão de salvar vidas, provendo uma gestão eficaz da informação para as organizações melhor se prepararem para responder a desastres. A importância de identificar riscos de segurança, garante a integridade, confidencialidade e disponibilidade do sistema. O objetivo deste trabalho é realizar um teste de penetração no sistema de gestão de desastres **SAHANA EDEN**, identificar vulnerabilidades, possíveis riscos associados e propor soluções para mitigá-los. Foram utilizadas as ferramentas de gestão de vulnerabilidades e análise de aplicações web. Como resultado, foram identificadas algumas vulnerabilidades e propostas as devidas recomendações para mitigá-las. Ao final do trabalho, concluiu-se que com a realização de testes de penetração e identificação de vulnerabilidade foi possível reduzir os riscos associados a ataques cibernéticos e tornar um ambiente tecnológico mais seguro.

Palavras-chave: Sahana Eden, Cibersegurança, Vulnerabilidades, Pentest, Segurança da Informação.

Campos dos Goytacazes/RJ
2023

ABSTRACT

Cyberattacks have grown exponentially in recent years and a practice is needed by their prevention organizations. One of the ways is to analyze the security of the technological environment in search of weaknesses and mitigate them. **SAHANA EDEN** is a disaster management system used with the mission of saving lives, providing effective information management for organizations to better prepare to respond to disasters. The importance of identifying security risks ensures the integrity, confidentiality, and availability of the system. The objective of this work is to perform a penetration test in the **SAHANA EDEN** disaster management system, identify vulnerabilities, possible risks associated and propose solutions to mitigate them. Vulnerability management and web application analysis tools were used. As a result, some vulnerabilities were identified, and the appropriate recommendations were proposed to mitigate them. At the end of the discussion, it was concluded that by taking penetration tests and identifying vulnerabilities it was possible to reduce the risks associated with cyber attacks and make a safer technological environment.

Keywords: Sahana Eden, Cybersecurity, Vulnerabilities, Pentest, Information Security.

Campos dos Goytacazes/RJ
2023

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API	Application Programming Interface
CDN	Content Delivery Network
CORS	Cross-Origin Resource Sharing
CWE	Common Weakness Enumerations
DBMS	Database Management Systems
ECB	Eletronic CodeBook
FTP	File Transfer Protocol
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JWT	JSON Web Token
LDAP	Lightweight Directory Access Protocol
MD5	Message-Digest algorithm 5
NIST	National Institute of Standards and Technology
PCI DSS	Payment Card Industry Data Security Standard
PKCS	Public-Key Cryptography Standards
SHA-1	Secure Hash Algorithm 1
SMTP	Simple Mail Transfer Protocol
SOAP	Simple Object Access Protocol
SQL	Structured Query Language
TLS	Transport Layer Security
URL	Uniform Resource Locator
XML	Extensible Markup Language

SUMÁRIO

1. INTRODUÇÃO

As ameaças à infraestrutura de rede de computadores estão aumentando e constantemente mudando todos os dias e, ataques Hackers estão mais sofisticados a cada fraqueza descoberta nos sistemas de rede de computadores tentando danificar tais sistemas de segurança (WANG; YANG, 2017). As vulnerabilidades de aplicativos da Web são um dos problemas de segurança mais comuns na Internet atualmente (DOUPÉ; COVA; VIGNA, 2010).

As Infraestruturas Críticas (IC) - instalações, serviços, bens e sistemas, exercem significativa influência na vida de qualquer pessoa e na operação de setores importantes para o desenvolvimento e manutenção do país, como é o caso do setor industrial. Elas são importantes pelas facilidades e utilidades que fornecem à sociedade e, principalmente, por subsidiar, na forma de recurso ou serviço, outras Infraestruturas Críticas, mais complexas ou não (GHORBANI; BAGHERI, 2008).

A gestão de riscos em Infraestruturas Críticas está assumindo um papel cada vez mais importante para a prestação de serviços seguros contra falhas tecnológicas (segurança). Nos últimos dez anos a segurança e os desastres naturais ampliaram significativamente sua finalidade. As autoridades internacionais, em diferentes níveis, decidiram enfrentar o problema por meio de uma padronização de procedimentos para análise e avaliação de riscos. As últimas decisões da União Europeia (2008) e da Comissão Europeia (2012) e a introdução de normas internacionais específicas de gestão de riscos reforçam esta política comum (ISO/IEC 27001, 2013; ISO 28000, 2007; ISO 31000, 2009) que são amplamente utilizadas na atualidade (SAPORI; SCIUTTO; SCIUTTO, 2014).

SAHANA EDEN *Disaster Management System* é um sistema de informações de gerenciamento de desastres de código aberto desenvolvido no Sri Lanka logo após o 3º maior terremoto registrado, desencadeando em diversos tsunamis devastadores no Oceano Índico, que vitimaram cerca de 230.000 pessoas. A interface principal do SAHANA EDEN provê acesso a componentes interconectados, mas independentes. Esses componentes interagem com cada outro através de um conjunto de bancos de dados compartilhados para fornecer serviços de informação baseados na Web. O sistema SAHANA EDEN pode ser dimensionado para utilização em um único notebook para uma rede totalmente distribuída e inclui uma estrutura de sincronização que permite que os indivíduos vão para o campo com uma cópia de dados existente, e mais tarde sincronizar com o servidor SAHANA EDEN central quando a conectividade com a Internet se tornar disponível (CURRION; SILVA; VAN DE WALLE, 2007).

Ao longo de décadas, o “ciclo de desastres” incorporou a sequência de fases, como: “prevenção, preparação, resposta, recuperação e reabilitação”. Com a adoção das fases do ciclo, os gestores e planejadores, principalmente para eventos de catástrofe natural, começaram a selecionar responsabilidades relacionadas com as fases para determinados setores da sociedade (BURKLE JR., 2019).

Em um cenário de guerra ou terrorismo, todos esses eventos por definição, sofrem de respostas consistentes de mitigação, recuperação e de estratégias de reabilitação e passa ser interesse dos cibercriminosos afetar os sistemas de gestão de desastres no intuito de impossibilitar a execução das fases como resposta, recuperação e consequentemente reabilitação, passando a ser dos principais motivos para a proteção do SAHANA contra ciberterrorismo.

O que é *ethical hacking*?

Com o crescimento da Internet, a segurança computacional tornou-se uma grande preocupação para as empresas e governo. Estas querem ter vantagens através da Internet para ampliar sua permeabilidade na sociedade, no comércio eletrônico, com publicidades, disseminar informações e acessos, expandir seus lucros, otimizar a influência no consumidor final, aumentar sua velocidade de transmissão de informações, etc. Neste contexto amplia-se a preocupação com a segurança e o risco de serem “hackeados”, ao mesmo tempo, os clientes potenciais destes serviços estão preocupados no controle e no sigilo das informações pessoais como números de cartões de crédito, identidade pessoal, senhas e endereço residencial.

Em sua busca por uma maneira de abordar o problema, organizações públicas e privadas, perceberam que uma das melhores maneiras de avaliar as ameaças de intrusão ao seu interesse seria ter profissionais independentes de segurança computacional no intuito de testarem a segurança através de invasões de seus sistemas computacionais. Este conceito é semelhante a ter auditores independentes que entram em uma organização para verificar seus registros contábeis. No caso da segurança computacional, essas “equipes de tigre” ou “hacker ético” empregam as mesmas ferramentas e técnicas como os intrusos, mas não danificam os sistemas de destino e nem roubam informações. Em vez disso, eles avaliam a segurança dos sistemas de destino e retornam para os proprietários com as vulnerabilidades encontradas e instruções de como remediá-las (PALMER, 2001).

O teste de penetração é uma ferramenta valiosa de avaliação de garantia que beneficia tanto os negócios quanto suas operações. Um teste bem-sucedido produz a capacidade de comandar as instalações do sistema para fazer diferente do que seus proprietários esperavam que

eles fizessem, para ganhar o total ou pelo menos controle substancial de um host de uma forma confiável normalmente reservada para pessoal da operação, ou para adquirir o gerenciamento de um aplicativo.

Teste de penetração é a estratégia de encontrar uma porta aberta para penetrar no sistema alvo em uma abordagem ética com uma visão de auditar e retificar a infraestrutura de segurança do sistema. Ajuda a proteger a organização contra falhas/desastres financeiros e perdas, preservando a imagem corporativa. Ele identifica e aborda riscos antes mesmo que as violações de segurança realmente ocorram. O teste de penetração também oferece um problema de segurança de investimentos para a alta administração (SHAH; MEHTRE, 2015).

2. OBJETIVO GERAL

Este trabalho tem como objetivo a condução de um teste de penetração abrangendo a aplicação do sistema SAHANA EDEN e sua infraestrutura, com o propósito de identificar potenciais riscos de segurança. O teste de penetração é executado de forma a focalizar especificamente na avaliação dos riscos presentes no SAHANA EDEN, o que, por sua vez, abre oportunidades para futuras pesquisas envolvendo outros componentes da infraestrutura que não estão exclusivamente relacionados ao sistema SAHANA EDEN, bem como outras estratégias no planejamento do teste.

Por meio deste processo, buscou-se avaliar as vulnerabilidades da aplicação e da infraestrutura, permitindo uma compreensão aprofundada dos riscos potenciais que podem ser explorados por atacantes mal-intencionados. Ao identificar e entender esses riscos, torna-se possível tomar medidas proativas para fortalecer a segurança do SAHANA EDEN e garantir sua proteção contra ameaças cibernéticas.

Vale ressaltar que o escopo do teste de penetração abrange exclusivamente o SAHANA EDEN, mas os resultados obtidos podem servir como base para pesquisas futuras que abordem outros elementos da infraestrutura, aumentando assim a abrangência e a compreensão geral da segurança do ambiente em questão.

Além disso, as estratégias adotadas durante o planejamento do teste foram selecionadas para garantir que as avaliações de segurança fossem abrangentes e efetivas. Dessa forma, o trabalho contribui significativamente para o aprimoramento contínuo da segurança do SAHANA EDEN e pode fornecer percepções valiosas para o desenvolvimento de melhores práticas em testes de penetração e segurança de sistemas similares.

2.1 Objetivos específicos

- Definir a metodologia de planejamento do teste de penetração;
- Planejar o teste de penetração;
- Executar a fase de descoberta com base no método escolhido;
- Executar a fase de varredura e enumeração da aplicação SAHANA EDEN e serviços do sistema operacional, mapeando possíveis vulnerabilidades;
- Executar a etapa de exploração das vulnerabilidades identificadas;
- Executar tentativa de escalada de privilégios com base na exploração de vulnerabilidades;
- Executar etapas de pós exploração de vulnerabilidades (exfiltração de dados).
- Identificar as ações necessárias para mitigar as vulnerabilidades detectadas após exploração das mesmas;
- Identificar os procedimentos de rotina para monitoramento do servidor e do software para o estabelecimento de um protocolo de monitoramento;

3. METODOLOGIA

Este estudo representa uma pesquisa aplicada, de natureza qualitativa, com o objetivo de explanação e emprego de procedimentos ex-post-facto, como indicado por Silva e Menezes (SILVA; MENEZES, 2005). A abordagem qualitativa foi adotada para permitir uma análise aprofundada e compreensiva dos fenômenos em questão, focando na compreensão dos aspectos subjetivos e complexos relacionados à aplicação do sistema SAHANA EDEN e sua infraestrutura.

O objetivo explicativo desta pesquisa visa identificar e explicar as possíveis vulnerabilidades de segurança presentes no SAHANA EDEN, explorando como elas podem surgir e impactar a integridade e confidencialidade dos dados.

O enfoque ex-post-facto refere-se ao caráter retrospectivo desta pesquisa, onde são analisados eventos passados, ou seja, dados já existentes e incidentes de segurança anteriores no SAHANA EDEN. Essa abordagem permite investigar ocorrências reais e seus impactos, bem como compreender como as medidas de segurança foram ou não efetivas em situações prévias.

Assim, por meio deste delineamento de pesquisa, espera-se contribuir significativamente para a prática e o aprimoramento da segurança do SAHANA EDEN. Os resultados obtidos podem auxiliar na identificação de pontos críticos e na implementação de medidas preventivas e corretivas, fortalecendo o sistema contra possíveis ataques cibernéticos e garantindo a continuidade de sua eficácia em cenários de emergência.

O EDEN, uma abreviatura para *Emergency Development ENvironment* (Ambiente de Desenvolvimento de Emergência), é uma ferramenta confiável que pode ser acessada através do site oficial da SAHANA Foundation, uma comunidade dedicada à construção de tecnologias para a gestão de emergências (https://sahanafoundation.org/products/eden/#), conforme pode ser visto na figura 1.

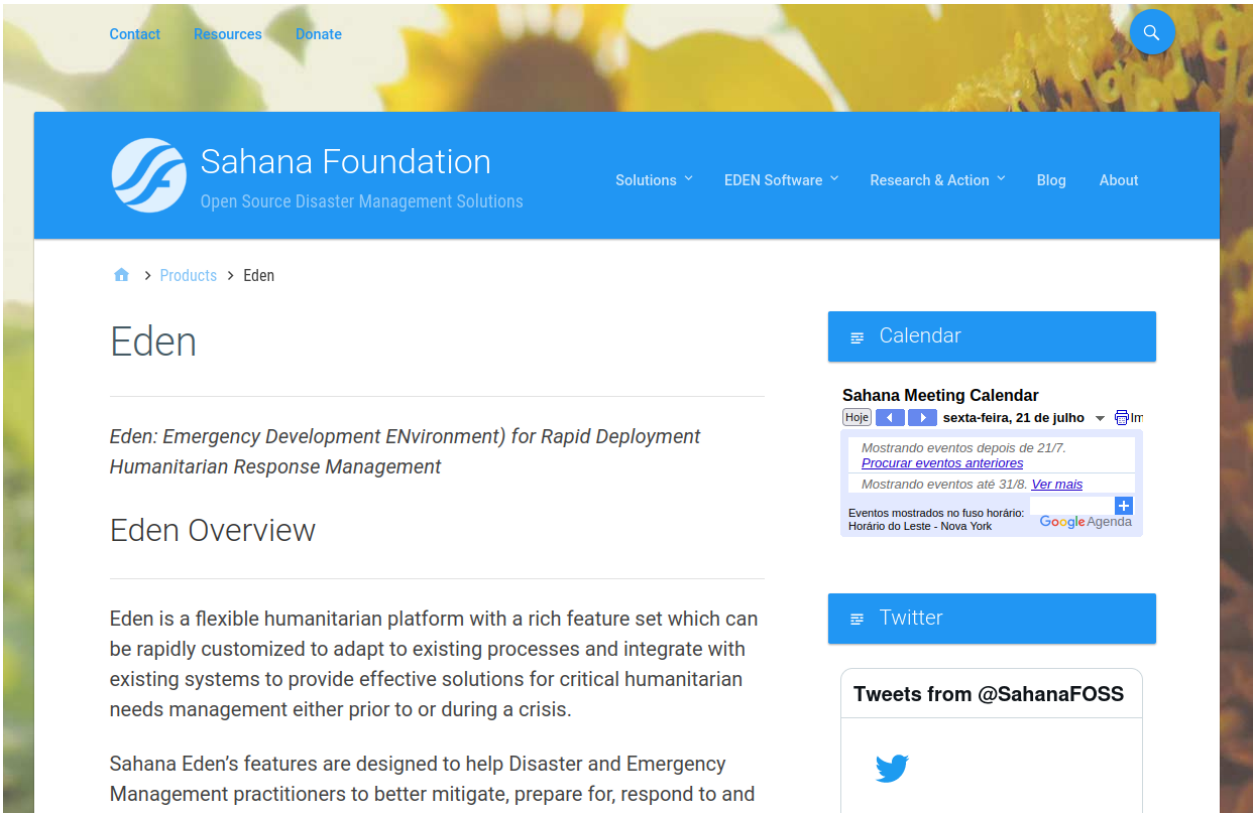

Figura 1 - Site da comunidade SAHANA Foundation.

Fonte: https://sahanafoundation.org/products/eden/#

Ao visitar o site, os usuários têm a oportunidade de obter informações detalhadas sobre o EDEN, explorando diversos recursos disponíveis. Um dos principais aspectos é a possibilidade de realizar o download do software, permitindo que organizações, agências governamentais e equipes de resposta a desastres possam implantar e utilizar essa valiosa ferramenta em suas operações de socorro e recuperação.

O site oferece acesso a manuais detalhados, que fornecem orientações e instruções passo a passo sobre como configurar e utilizar o EDEN de forma eficiente. Esses manuais foram elaborados com foco na facilidade de compreensão, garantindo que tanto os usuários novatos quanto os experientes possam aproveitar plenamente os recursos do software.

A seção de funcionalidades apresenta uma visão geral das capacidades do EDEN permitindo que os interessados compreendam completamente o que a ferramenta pode oferecer em termos de gestão de recursos, coordenação de equipes, coleta de dados, análise de informações e tomada de decisões estratégicas durante emergências.

A SAHANA Foundation também disponibiliza um blog regularmente atualizado, trazendo notícias, estudos de caso, histórias de sucesso e artigos técnicos relevantes relacionados

ao uso do EDEN em situações reais de desastres. Essas percepções fornecem uma visão mais profunda do impacto positivo que o EDEN pode ter em operações de resposta a emergências.

Por fim, a página wiki é uma fonte abrangente de documentação que oferece um conhecimento detalhado sobre o EDEN. Com uma vasta coleção de informações, tutoriais e guias, a página wiki atende a todos os aspectos da ferramenta, tornando-a uma referência valiosa para usuários que desejam explorar ainda mais suas funcionalidades e aprofundar seus conhecimentos, como pode ser visto na figura 2.

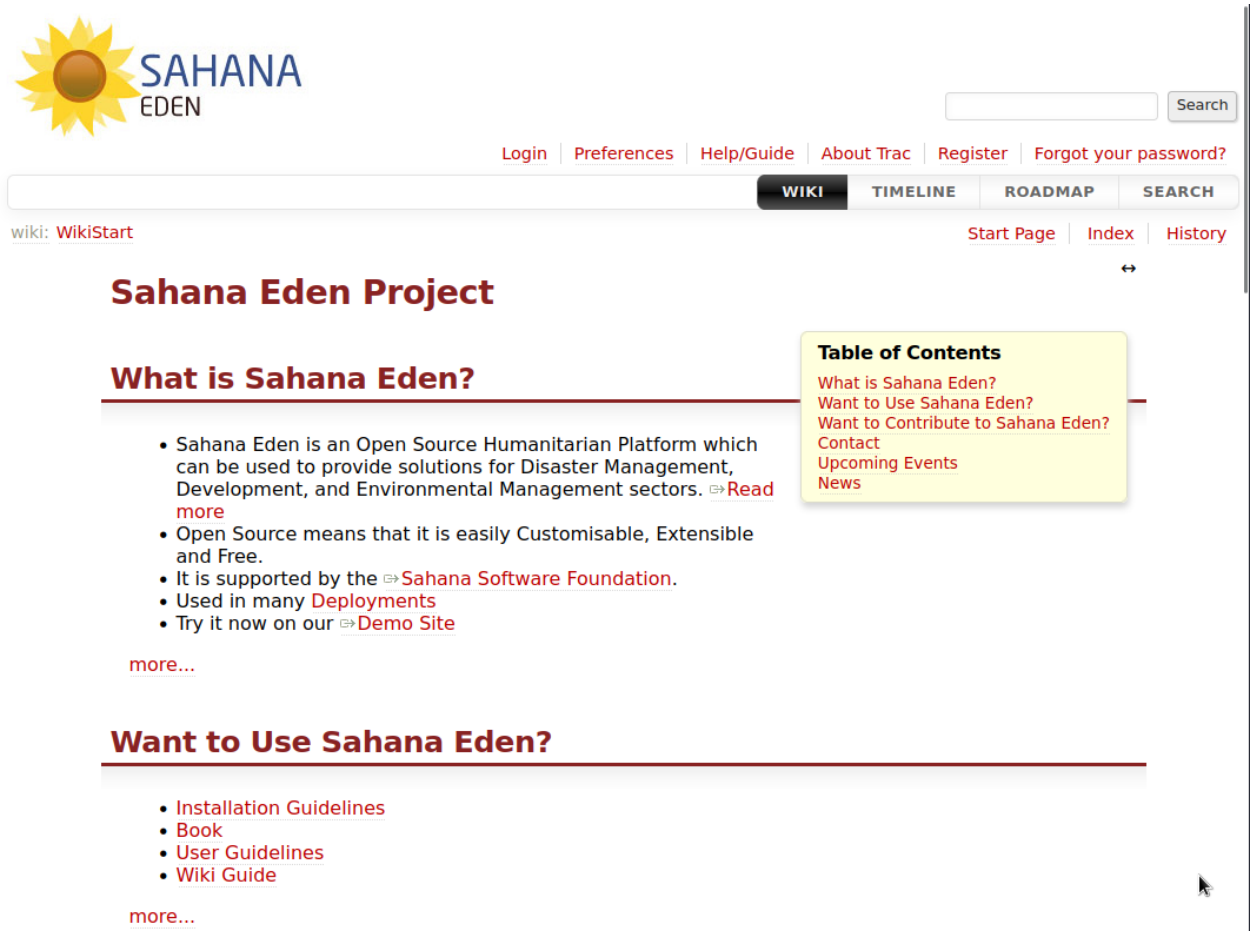

Figura 2 - Site Wiki do SAHANA EDEN.

Fonte: https://eden.sahanafoundation.org/

Em resumo, o EDEN é uma solução completa e versátil para situações de emergência, e o site da SAHANA Foundation é uma fonte rica e centralizada para acessar todas as informações essenciais sobre o software. Seja para realizar o download, consultar manuais detalhados, explorar funcionalidades, ler o blog ou aprofundar-se na documentação da ferramenta, o site oferece uma experiência completa para capacitar e apoiar todos aqueles envolvidos na resposta a crises e desastres.

Dentre os métodos mais utilizados no teste de penetração ou pentest, temos o OSSTMM (*Open Source Security Testing Methodology Manual*) mantido pelo ISECOM (*Institute for Security and Open Methodologies*) que é uma comunidade de segurança aberta, co-fundada em Nova Iorque e Barcelona. É um dos padrões profissionais mais completos e comumente utilizado em auditorias de segurança para revisar a Segurança de Sistemas da Internet. Inclui uma estrutura que descreve as etapas que seriam realizadas para a implementação da auditoria (LOPEZ DE JIMENEZ, 2016).

O NIST SP 800-115 (NIST *Special Publication* 800-115, *Technical Guide to Information Security Testing and Assessment*) é um guia do governo dos Estados Unidos para realizar testes de penetração (pentest) em sistemas de informação. Ele é publicado pelo *National Institute of Standards and Technology* (NIST) e é um dos principais padrões utilizados pelo governo dos EUA para avaliar a segurança de seus sistemas de informação. O guia fornece uma metodologia abrangente para planejar, realizar e relatar testes de penetração, incluindo diretrizes para a seleção de alvos, a identificação de vulnerabilidades, a exploração de brechas de segurança e a geração de relatórios. Ele também inclui diretrizes para garantir a legalidade e a ética dos testes de penetração (SCARFONE et al., 2008).

OWASP (*Open Web Application Security Project*) é uma organização aberta e sem fins lucrativos que trabalha na melhoria de segurança de software dedicado a habilitar nas organizações o conceito de desenvolver, adquirir, operar e manter aplicações que possam ser confiáveis. É uma metodologia exclusiva para testar aplicações web e concentra-se nas ameaças de segurança específicas para aplicativos, o que ajuda a garantir a eficiência dos testes. Um dos documentos mais utilizados inclui o “OWASP *Top Ten*”, um guia amplamente adotado contendo os 10 maiores riscos de segurança em aplicações Web (LOPEZ DE JIMENEZ, 2016).

Da metodologia OWASP iremos focar no método OWASP *Top ten*-2021, que pode ser obtido em https://owasp.org/Top10/ , que contém a listagem das 10 vulnerabilidades mais comuns em aplicações web, suas respectivas informações, recomendações de como se prevenir e cenários exemplos de ataque. Método este que cobre grande parte dos riscos críticos atuais, sendo sua versão atual de 2021, substituindo a de 2017, conforme figura 3.

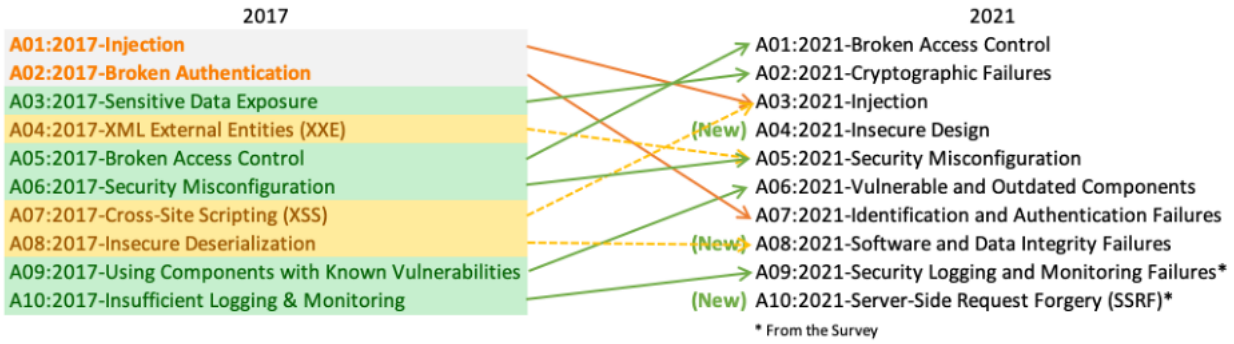

Figura 3 - Evolução do método em comparação a 2017.

Fonte: https://owasp.org/Top10/

A lista com as 10 vulnerabilidades são enumeradas de A01 a A10, sendo a primeira a mais crítica e assim sucessivamente.

A cada vulnerabilidade listada, são também elencados as Enumerações de Fraquezas Comuns, conhecidos como CWE (*Common Weakness Enumeration*), que é uma lista desenvolvida pela comunidade de tipos comuns de vulnerabilidades de software e hardware que têm diversas ramificações de segurança, lista essa que pode ser consultada no website https://cwe.mitre.org/data/index.html. Uma "fraqueza" é uma condição em um software, firmware, hardware ou componente de serviço que, sob certas circunstâncias, pode contribuir para a introdução de vulnerabilidades. A lista de CWE e a taxonomia de classificação associada servem como uma linguagem que pode ser usada para identificar e descrever esses pontos fracos em termos de CWEs. O principal objetivo do CWE é evitar vulnerabilidades no código-fonte educando programadores de software e hardware, designers, arquitetos sobre como remover os erros mais comuns antes que o software e o hardware sejam entregues, direcionados nas comunidades de desenvolvimento e segurança (AL-BOGHDADY; WASSIF; EL-RAMLY, 2021).

O CWE ajuda desenvolvedores e profissionais de segurança a:

- Descrever e discutir os pontos fracos de software e hardware em uma linguagem comum;
- Verificar se há pontos fracos nos produtos de software e hardware existentes;
- Avaliar a cobertura de ferramentas voltadas para esses pontos fracos;
- Aproveitar um padrão de linha de base comum para esforços de identificação, mitigação e prevenção de fraquezas;
- Evitar vulnerabilidades de software e hardware antes da implantação.

A01:2021-*Broken Access Control*

O controle de acesso reforça a política de forma que os usuários não possam agir fora de suas permissões pretendidas. As falhas geralmente levam à divulgação não autorizada de informações, modificação ou destruição de todos os dados ou à execução de uma função comercial fora dos limites do usuário. Vulnerabilidades comuns de controle de acesso incluem:

- Violar o princípio de privilégio mínimo ou negação por padrão, em que o acesso deve ser concedido apenas para recursos, funções ou usuários específicos, mas está disponível para qualquer pessoa.
- Ignorar as verificações de controle de acesso modificando a URL (violação de parâmetros ou navegação forçada), o estado interno do aplicativo ou a página HTML ou usando uma ferramenta de ataque que modifica solicitações de API.
- Permitir visualizar ou editar a conta de outra pessoa, fornecendo seu identificador exclusivo (referências diretas inseguras de objetos)
- Acessar a API com controles de acesso ausentes para POST, PUT e DELETE.
- Elevar privilégio. Atuar como usuário sem estar conectado ou atuar como administrador quando conectado como usuário.
- Manipular metadados, como reprodução ou adulteração de um token de controle de acesso JSON *Web Token* (JWT), ou um cookie ou campo oculto manipulado para elevar privilégios ou abusar da invalidação de JWT.
- A configuração incorreta do CORS permite acesso à API de origens não autorizadas/não confiáveis.
- Forçar a navegação para páginas autenticadas como usuário não autenticado ou para páginas privilegiadas como usuário padrão.

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluídas são CWE-200: *Exposure of Sensitive Information to an Unauthorized Actor* (Exposição de informações confidenciais a um ator não autorizado), CWE-201: *Insertion of Sensitive Information Into Sent Data* (Inserção de informações confidenciais em dados enviados) e CWE-352: *Cross-Site Request Forgery* (Falsificação de solicitações entre sites).

A02:2021-*Cryptographic Failures*

A primeira coisa é determinar as necessidades de proteção dos dados em trânsito e em repouso. Por exemplo, senhas, números de cartão de crédito, registros de saúde, informações pessoais e segredos comerciais exigem proteção extra, principalmente se esses dados estiverem sob leis de privacidade, por exemplo, Regulamento Geral de Proteção de Dados da UE (GDPR) ou regulamentos, por exemplo, proteção de dados financeiros como o padrão de segurança de dados PCI (PCI DSS). Para todos esses dados:

- Algum dado é transmitido em texto não criptografado? Isso diz respeito a protocolos como HTTP, SMTP, FTP também usando atualizações TLS como STARTTLS. O tráfego externo da Internet é perigoso. Verifique todo o tráfego interno, por exemplo, entre balanceadores de carga, servidores web ou sistemas de back-end.
- Algum algoritmo ou protocolo criptográfico antigo ou fraco é usado por padrão ou em código mais antigo?
- As chaves criptográficas padrão estão em uso, as chaves criptográficas fracas são geradas ou reutilizadas ou o gerenciamento ou rotação de chaves adequado está ausente? As chaves criptográficas são verificadas nos repositórios de código-fonte?
- A criptografia não é aplicada, por exemplo, há alguma diretiva de segurança de cabeçalho HTTP (navegador) ou cabeçalhos faltando?
- O certificado do servidor recebido e a cadeia de confiança foram devidamente validados?
- Os vetores de inicialização são ignorados, reutilizados ou não gerados suficientemente seguros para o modo criptográfico de operação? Um modo de operação inseguro, como o ECB, está em uso? A criptografia usada quando autenticada é mais apropriada?
- As senhas estão sendo usadas como chaves criptográficas na ausência de uma função de derivação de chave base de senha?
- A aleatoriedade é usada para fins criptográficos que não foram projetados para atender aos requisitos criptográficos? Mesmo se a função correta for escolhida, ela precisa ser semeada pelo desenvolvedor e, caso contrário, o desenvolvedor substituiu a funcionalidade de semeadura forte incorporada a ela por uma semente que não possui entropia/imprevisibilidade suficientes?

- As funções de hash obsoletas, como MD5 ou SHA1, estão em uso ou as funções de hash não criptográficas são usadas quando são necessárias funções de hash criptográficas?
- Métodos de preenchimento criptográfico obsoletos, como PKCS número 1 v1.5, estão em uso?
- As mensagens de erro criptográficas ou as informações do canal lateral são exploráveis, por exemplo, na forma de ataques de oráculo de preenchimento?

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluídas são CWE-259: *Use of Hard-coded Password* (Uso de senha codificada), CWE-327: *Broken or Risky Crypto Algorithm* (Algoritmo de criptografia quebrado ou arriscado) e CWE-331 *Insufficient Entropy* (Entropia insuficiente).

A03:2021-*Injection*

Algumas das injeções mais comuns são SQL, NoSQL, comando do sistema operacional, mapeamento relacional de objeto (ORM), LDAP e injeção de linguagem de expressão (EL) ou biblioteca de navegação de gráfico de objeto (OGNL). A revisão do código-fonte é o melhor método para detectar se os aplicativos são vulneráveis a injeções. Testes automatizados de todos os parâmetros, cabeçalhos, URL, cookies, JSON, SOAP e entradas de dados XML são fortemente encorajados. As organizações podem incluir ferramentas de teste de segurança de aplicativos estáticos (SAST), dinâmicos (DAST) e interativos (IAST) no pipeline de CI/CD para identificar falhas de injeção introduzidas antes da implantação da produção.

Um aplicativo é vulnerável a ataques quando:

- Os dados fornecidos pelo usuário não são validados, filtrados ou sanitizados pelo aplicativo.
- Consultas dinâmicas ou chamadas não parametrizadas sem escape sensível ao contexto são usadas diretamente no interpretador.
- Os dados hostis são usados nos parâmetros de pesquisa de mapeamento objeto-relacional (ORM) para extrair registros confidenciais adicionais.
- Dados hostis são usados diretamente ou concatenados. O SQL ou comando contém a estrutura e os dados maliciosos em consultas dinâmicas, comandos ou procedimentos armazenados.

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluídas são CWE-79: *Cross-site Scripting*, CWE-89: *SQL Injection* e CWE-73: *External Control of File Name or Path*.

A04:2021-*Insecure Design*

Projeto inseguro é uma categoria ampla que representa diferentes pontos fracos, expressos como "desenho de controle ausente ou ineficaz". O design inseguro não é a fonte de todas as outras 10 principais categorias de risco. Há uma diferença entre design inseguro e implementação insegura. Nós diferenciamos entre falhas de design e defeitos de implementação por um motivo, eles têm causas e remediações diferentes. Um design seguro ainda pode ter defeitos de implementação levando a vulnerabilidades que podem ser exploradas. Um design inseguro não pode ser corrigido por uma implementação perfeita, pois, por definição, os controles de segurança necessários nunca foram criados para se defender contra ataques específicos. Um dos fatores que contribuem para o design inseguro é a falta de perfil de risco de negócios inerente ao software ou sistema que está sendo desenvolvido e, portanto, a falha em determinar qual nível de design de segurança é necessário.

Gerenciamento de Requisitos e Recursos

Colete e negocie os requisitos de negócios para um aplicativo com a empresa, incluindo os requisitos de proteção relativos à confidencialidade, integridade, disponibilidade e autenticidade de todos os ativos de dados e a lógica de negócios esperada. Leve em consideração o quão exposto o aplicativo ficará e se precisa de segregação de *tenants* (além do controle de acesso). Compile os requisitos técnicos, incluindo requisitos de segurança funcionais e não funcionais. Planejar e negociar o orçamento abrangendo todo o projeto, construção, teste e operação, incluindo atividades de segurança.

Design seguro

O design seguro é uma cultura e metodologia que avalia constantemente as ameaças e garante que o código seja projetado e testado de forma robusta para evitar métodos de ataque conhecidos. A modelagem de ameaças deve ser integrada em sessões de refinamento (ou atividades semelhantes). Procure mudanças nos fluxos de dados e controle de acesso ou outros controles de segurança. No desenvolvimento da história do usuário, determine o fluxo correto e os estados de falha, certifique-se de que sejam bem compreendidos e acordados pelas partes responsáveis e impactadas. Analisar suposições e condições para fluxos esperados e de falha,

certificar de que eles ainda sejam precisos e desejáveis. Determine como validar as suposições e impor condições necessárias para comportamentos adequados. Certifique-se de que os resultados sejam documentados na história do usuário. Aprenda com os erros e ofereça incentivos positivos para promover melhorias. O design seguro não é um complemento nem uma ferramenta que você pode adicionar ao software.

Ciclo de vida de desenvolvimento seguro

O software seguro requer um ciclo de vida de desenvolvimento seguro, alguma forma de padrão de projeto seguro, metodologia de estrada pavimentada, biblioteca de componentes seguros, ferramentas e modelagem de ameaças. Entre em contato com seus especialistas em segurança no início de um projeto de software durante todo o projeto e manutenção de seu software. Considere utilizar o *Software Assurance Maturity Model* (SAMM) da OWASP para ajudar a estruturar seus esforços de desenvolvimento de software seguro.

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluem CWE-209: *Generation of Error Message Containing Sensitive Information* (Geração de mensagem de erro contendo informações confidenciais), CWE-256: *Unprotected Storage of Credentials* (Armazenamento desprotegido de credenciais), CWE-501: *Trust Boundary Violation* (Violação de limite de confiança) e CWE-522: *Insufficiently Protected Credentials* (Credenciais protegidas insuficientemente).

A05:2021-*Security Misconfiguration*

Sem um processo de configuração de segurança de aplicativo repetível e coordenado, os sistemas correm um risco maior. O aplicativo pode estar vulnerável se:

- Houver falta de proteção de segurança apropriada em qualquer parte da pilha de aplicativos ou permissões configuradas incorretamente em serviços de nuvem.
- Recursos desnecessários são ativados ou instalados (por exemplo, portas, serviços, páginas, contas ou privilégios desnecessários).
- As contas padrão e suas senhas ainda estão habilitadas e inalteradas.
- O tratamento de erros revela rastreamentos de pilha ou outras mensagens de erro excessivamente informativas para os usuários.

- Para sistemas atualizados, os recursos de segurança mais recentes são desativados ou não configurados com segurança.
- As configurações de segurança nos servidores de aplicativos, estruturas de aplicativos (por exemplo, Struts, Spring, ASP.NET), bibliotecas, bancos de dados, etc., não são definidas para valores seguros.
- O servidor não envia cabeçalhos ou diretivas de segurança ou eles não são definidos como valores seguros.
- O software está desatualizado ou vulnerável.

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluem CWE-16 *Configuration* e CWE-611 *Improper Restriction of XML External Entity Reference*.

A06:2021-*Vulnerable and Outdated Components*

Os componentes vulneráveis são um problema conhecido que lutamos para testar e avaliar o risco e é a única categoria que não possui vulnerabilidades e exposições comuns (CVEs) mapeadas para os CWEs incluídos; portanto, é usado um peso padrão de exploits/impacto de 5,0. As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluem o CWE-1104: Uso de componentes de terceiros não mantidos e os dois CWEs do Top 10 2013 e 2017.

- Desconhecer as versões de todos os componentes (tanto do lado do cliente quanto do lado do servidor). Isso inclui componentes utilizados diretamente, bem como dependências aninhadas.
- Se o software estiver vulnerável, sem suporte ou desatualizado. Isso inclui o sistema operacional, servidor web/aplicativo, sistema de gerenciamento de banco de dados (DBMS), aplicativos, APIs e todos os componentes, ambientes de tempo de execução e bibliotecas.
- Não verificar vulnerabilidades regularmente e não utilizar boletins de segurança relacionados aos componentes que usa.
- Não corrigir ou atualizar a plataforma, as estruturas e as dependências subjacentes de maneira oportuna e baseada em risco. Isso geralmente acontece em ambientes em que a

aplicação de patches é uma tarefa mensal ou trimestral sob controle de alterações, deixando as organizações sujeitas a dias ou meses de exposição desnecessária a vulnerabilidades corrigidas.

- Se desenvolvedores de software não testarem a compatibilidade de bibliotecas atualizadas, atualizadas e corrigidas.
- Não proteger as configurações dos componentes.

A07:2021-*Identification and Authentication Failures*

A confirmação da identidade do usuário, autenticação e gerenciamento de sessão são essenciais para proteção contra ataques relacionados à autenticação. Pode haver pontos fracos de autenticação se o aplicativo:

- Permite ataques automatizados, como preenchimento de credenciais, em que o invasor possui uma lista de nomes de usuário e senhas válidas.
- Permite força bruta ou outros ataques automatizados.
- Permite senhas padrão, fracas ou conhecidas, como "Senha1" ou "admin/admin".
- Usa recuperação de credenciais fraca ou ineficaz e processos de esquecimento de senha, como "respostas baseadas em conhecimento", que não podem ser tornadas seguras.
- Usa armazenamentos de dados de senhas de texto simples, criptografadas ou com hash fraco.
- Tem autenticação multifator ausente ou ineficaz.
- Expõe o identificador de sessão na URL.
- Reutilize o identificador de sessão após o login bem-sucedido.
- Não invalida corretamente IDs de sessão. Sessões de usuário ou tokens de autenticação (principalmente tokens de logon único (SSO)) não são invalidados adequadamente durante o logoff ou um período de inatividade.

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) são CWE-297: *Improper Validation of Certificate with Host Mismatch* (Validação imprópria de

certificado com incompatibilidade de host), CWE-287: *Improper Authentication* (Autenticação imprópria) e CWE-384: *Session Fixation* (Fixação de sessão).

A08:2021-Software and Data Integrity Failures

Falhas de software e integridade de dados estão relacionadas a código e infraestrutura que não protegem contra violações de integridade. Um exemplo disso é quando um aplicativo depende de *plug-ins*, bibliotecas ou módulos de fontes, repositórios e redes de distribuição de conteúdo (CDNs) não confiáveis. Um processo de desenvolvimento baseado em integração e entrega contínua, (CI/CD) inseguro pode introduzir o potencial de acesso não autorizado, código malicioso ou comprometimento do sistema. Por fim, muitos aplicativos agora incluem a funcionalidade de atualização automática, onde as atualizações são baixadas sem verificação de integridade suficiente e aplicadas ao aplicativo confiável anteriormente. Os invasores podem carregar suas próprias atualizações para serem distribuídas e executadas em todas as instalações. Outro exemplo é onde objetos ou dados são codificados ou serializados em uma estrutura que um invasor pode ver e modificar é vulnerável à desserialização insegura.

As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) incluem CWE-829: *Inclusion of Functionality from Untrusted Control Sphere* (inclusão de funcionalidade de esfera de controle não confiável), CWE-494: *Download of Code Without Integrity Check* (download de código sem verificação de integridade) e CWE-502: *Deserialization of Untrusted Data* (desserialização de dados não confiáveis).

A09:2021-Security Logging and Monitoring Failures

Esta categoria é para ajudar a detectar, escalar e responder a violações ativas. Sem registro e monitoramento, as violações não podem ser detectadas. Registro, detecção, monitoramento e resposta ativa insuficientes ocorrem a qualquer momento:

- Eventos auditáveis, como logins, logins com falha e transações de alto valor, não são registrados.
- Avisos e erros não geram mensagens de registro, inadequadas ou pouco claras.
- Logs de aplicativos e APIs não são monitorados para atividades suspeitas.
- Os logs são armazenados apenas localmente.

- Os limites de alerta apropriados e os processos de escalonamento de resposta não estão em vigor ou são eficazes.
- Testes de penetração e varreduras por ferramentas de teste dinâmico de segurança de aplicativos (DAST) (como OWASP ZAP) não acionam alertas.
- O aplicativo não pode detectar, escalar ou alertar sobre ataques ativos em tempo real ou quase em tempo real.

Permitir o vazamento de informações ao tornar os eventos de registro e alerta visíveis para um usuário ou invasor.

O registro e o monitoramento podem ser difíceis de testar, geralmente envolvendo entrevistas ou perguntando se ataques foram detectados durante um teste de penetração. Não há muitos dados CVE/CVSS para esta categoria, mas detectar e responder a violações é fundamental. Ainda assim, pode ser muito impactante para responsabilidade, visibilidade, alerta de incidentes e perícia. As *Common Weakness Enumerations* (CWEs - Enumerações de Fraqueza Comum) são CWE-778 *Insufficient Logging to include* (Registro insuficiente para incluir) CWE-117 *Improper Output Neutralization for Logs* (Neutralização de saída imprópria para registros), CWE-223 *Omission of Security-relevant Information* (Omissão de informações relevantes de segurança) e CWE-532 *Insertion of Sensitive Information into Log File* (Inserção de informações confidenciais no arquivo de registro).

A10:2021-Server-Side Request Forgery (SSRF)

As falhas de SSRF ocorrem sempre que um aplicativo da Web está buscando um recurso remoto sem validar a URL fornecida pelo usuário. Ele permite que um invasor force o aplicativo a enviar uma solicitação criada para um destino inesperado, mesmo quando protegido por um firewall, VPN ou outro tipo de lista de controle de acesso à rede (ACL).

Como os aplicativos da web modernos fornecem aos usuários finais recursos convenientes, buscar uma URL se torna um cenário comum. Como resultado, a incidência de SSRF está aumentando. Além disso, a gravidade do SSRF está aumentando devido aos serviços em nuvem e à complexidade das arquiteturas.

O OWASP *Web Security Testing Guide*, que pode ser consultado no website https://owasp.org/www-project-web-security-testing-guide/ é um outro método que abrange a parte de teste de segurança em tempo de desenvolvimento da aplicação, o que pode ser adotado

como uma prática de segurança na concepção da aplicação web, o que não é parte do estudo desta dissertação, mas que abre uma oportunidade de pesquisa sobre o mesmo.

O ISSAF (Information Systems Security Assessment Framework) foi suportado pelo OISSG (Open Information Systems Security Group) embora não seja mais mantido e, portanto, um pouco desatualizado. É uma metodologia estruturada para análise de segurança em múltiplos domínios e detalhes específicos de teste ou testes para cada um deles. Isto visa fornecer procedimentos muito detalhados para o teste de sistemas de informação que reflitam situações reais (LOPEZ DE JIMENEZ, 2016).

É usado principalmente para atender aos requisitos de avaliação de organizações e também pode ser usado como referência para novas implementações relacionadas à segurança da informação. Um de seus pontos fortes é que ele vincula etapas individuais do teste de penetração com as respectivas ferramentas de teste de penetração. Ele visa fornecer um guia abrangente na condução do teste e pode ser uma boa base para desenvolver sua própria metodologia personalizada.

O PTES (Penetration Testing Execution Standard), é um padrão aberto, iniciado em 2009 após uma discussão que surgiu entre alguns dos membros fundadores sobre o valor (ou falta de) de testes de penetração na indústria. Fornece uma metodologia estruturada destinada a especificar o que são avaliações de penetração e quais as etapas envolvidas. Além disso, um dos objetivos do PTES é fornecer aos clientes uma referência para determinar a qualidade dos testes realizados para por testadores de penetração. Esta estrutura visa aumentar a qualidade global de avaliações de testes de penetração e especialmente suporte às empresas para definir o que eles precisam para realizar uma avaliação de segurança aos seus sistemas e o que esperar das diferentes fases dos testes de penetração (DINIS; SERRAO, 2014).

A Figura 4 apresenta de forma hierárquica o processo estruturado para a avaliação de segurança, dividido em cinco etapas distintas.

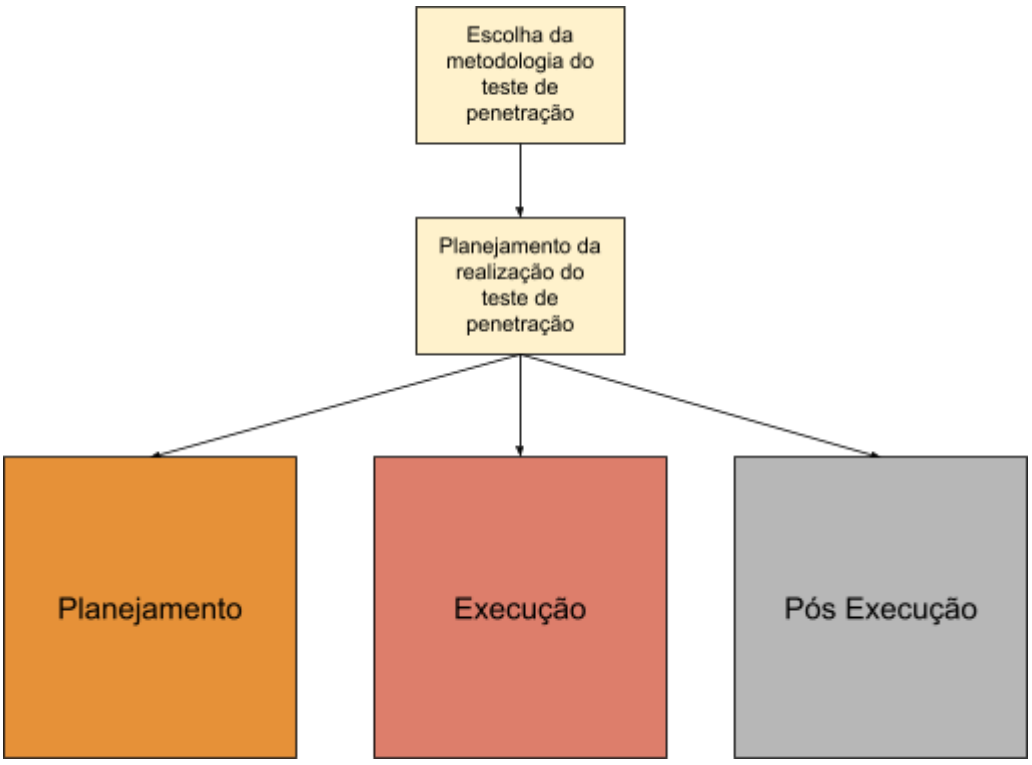

Figura 4 - Ilustração do processo de avaliação de segurança.

Fonte: Autor

Na primeira etapa, a seleção do método de teste de invasão é realizada com base nas metodologias atuais disponíveis. Nesta etapa, são analisadas diferentes abordagens de teste de penetração, a fim de escolher a mais adequada para avaliar a segurança do SAHANA EDEN.

A segunda etapa é dedicada ao planejamento com base na metodologia escolhida anteriormente. Nessa fase, são definidos os principais pilares que irão nortear a avaliação de segurança.

A terceira etapa representa o planejamento detalhado das atividades de teste, incluindo a preparação de ambientes de teste, a definição de cenários de ataques, a seleção de ferramentas e a elaboração de um plano de execução. Essa etapa é fundamental para garantir que o teste de penetração seja conduzido de forma organizada e estruturada.

A quarta etapa envolve a execução propriamente dita dos testes de penetração, onde são simulados ataques e exploradas possíveis vulnerabilidades no SAHANA EDEN. Nesse estágio, são realizados testes de intrusão, identificação de falhas de segurança e avaliação do desempenho do sistema sob condições adversas.

Por fim, a quinta camada é a fase de pós-execução, onde os resultados dos testes são analisados e interpretados. Os dados coletados são cuidadosamente avaliados para compreender a

gravidade e a extensão das vulnerabilidades identificadas. Com base nas descobertas, são elaborados relatórios detalhados contendo recomendações para mitigar os riscos identificados e melhorar a segurança geral do SAHANA EDEN.

3.1 Escolha da metodologia do teste de penetração

Realizar um comparativo entre as metodologias mais utilizadas, citando-se escopo, estrutura, credibilidade perante ao mercado, as suas vantagens e desvantagens conforme demonstra a tabela 1.

A escolha da metodologia foi baseada especificamente para atender um escopo específico considerado simples, ou seja uma metodologia que não fosse muito complexa, mas que pudesse dar os insumos necessários para um teste bem sucedido. Não há um melhor método universal para todos os ambientes porque há uma dependência de vários fatores, como as necessidades específicas de segurança da organização, as habilidades e recursos disponíveis da equipe de segurança e o orçamento.

No entanto, para um ambiente simples constituído de um servidor e poucas aplicações, o que descreve esta dissertação, o NIST SP 800-115 e o OWASP Top 10 são boas escolhas. Ambos oferecem uma metodologia assertiva para planejar, realizar e relatar testes de penetração, e são amplamente aceitos como padrões para a avaliação da segurança da informação. Além disso, eles oferecem recursos e ferramentas que podem ajudar a equipe a identificar e corrigir problemas de segurança de maneira eficiente.

Utilizaremos o NIST SP 800-115 como base principal para todas as etapas do teste de penetração e o OWASP Top 10 especificamente para a realização dos testes da aplicação Web do SAHANA EDEN.

Tabela 1 - Comparativo das metodologias mais utilizadas para teste de penetração.

Metodologia	Escopo	Estrutura	Credibilidade	Vantagens	Desvantagens
OSSTMM (Open Source Security Testing Methodology Manual)	Fornecer avaliação a segurança de redes, sistemas e aplicativos, identificando vulnerabilidades e ameaças, incluindo testes de segurança física e de engenharia social.	Dividido em cinco etapas: Reconhecimento, Descoberta, Validação, Exploração e Relatórios.	É considerado um dos métodos de pentest mais rigorosos e abrangentes.	Abordagem abrangente que cobre todos os aspectos da segurança, incluindo segurança física e social. Ênfase na validação de vulnerabilidades para garantir que sejam reais e exploração realista de vulnerabilidades.	É uma metodologia complexa e pode levar mais tempo para ser executada do que outras metodologias. O foco em segurança física e social pode não ser relevante para algumas organizações.
NIST SP 800-115 (National Institute of Standards and Technology Special Publication 800-115)	Fornecer orientações para testes de segurança em redes e sistemas. Usa uma variedade de técnicas para identificar vulnerabilidades, incluindo varreduras de portas, análise e exploração de vulnerabilidades.	Dividido em quatro etapas: Planejamento, Coleta de Informações, Análise e Relatórios.	É um método de pentest respeitado e amplamente adotado pelo setor governamental americano.	É uma metodologia amplamente adotada pelo setor governamental, o que pode aumentar a credibilidade do relatório final. O foco no planejamento pode ajudar a garantir que os testes sejam realizados de maneira consistente e completa.	A metodologia pode ser muito genérica e não cobrir especificidades de cada organização. A abordagem pode não ser tão abrangente quanto outros métodos.
PTES (Penetration Testing Execution Standard)	Fornecer uma estrutura padrão para a realização de testes de penetração. Aborda diversas técnicas, incluindo testes de engenharia social, testes de aplicativos, redes e serviços web	Dividido em sete etapas: Pré-teste, Inteligência, Ameaças e Modelagem de Risco, Testes de Segurança, Análise de Vulnerabilidades, Exploração e Relatórios.	É uma metodologia respeitada e utilizada por empresas e organizações governamentais.	Aborda todas as áreas relevantes para testes de penetração, incluindo inteligência, modelagem de risco e análise de ameaças. Fornecimento de modelos de relatórios para facilitar a padronização do processo de relatório final.	O processo pode ser muito complexo e levar mais tempo do que outras metodologias. Algumas etapas podem não ser relevantes para todas as organizações.
OWASP (Open Web Application Security Project)	Fornecer orientações para garantir a segurança de aplicativos web. Usa técnicas para identificar vulnerabilidades em aplicativos web, incluindo injeção SQL, cross-site scripting (XSS) e ataques de força bruta.	Dividido em quatro etapas: coleta de informações, testes de segurança, análise de resultados e relatórios.	É um projeto reconhecido mundialmente e amplamente adotado por empresas e organizações governamentais.	Aborda vulnerabilidades específicas de aplicativos web que outras metodologias podem não cobrir adequadamente. O foco na comunidade e na colaboração ajuda a garantir que as informações e orientações estejam sempre atualizadas.	O foco específico em aplicativos web pode não ser relevante para todas as organizações. A abordagem pode não ser tão abrangente quanto outros métodos.
ISSAF (Information Systems Security Assessment Framework)	Fornecer uma estrutura padrão para avaliação de segurança de sistemas de informação. Aborda diversas técnicas de testes de segurança, incluindo engenharia social, testes de aplicativos e testes de rede.	Dividido em oito etapas: Planejamento, Coleta de Informações, Análise de Vulnerabilidades, Exploração, Consolidação de Resultados, Documentação, Revisão e Relatório Final.	É amplamente utilizado por empresas e organizações governamentais em todo o mundo.	Foca em uma abordagem completa para a avaliação de segurança de sistemas de informação, incluindo análise de vulnerabilidades e engenharia social. É uma metodologia flexível, que permite que as organizações adaptem a metodologia para suas necessidades específicas.	Não é mais mantido, portanto é ainda utilizado devido algumas vantagens da metodologia. Algumas etapas podem ser consideradas excessivamente detalhadas ou complexas para organizações menores.

Fonte: o Autor

3.2 Planejamento do teste de penetração

Nesta etapa, será utilizado a metodologia por fases para o planejamento e desenvolvimento dos testes de penetração, dentre as fases, como visto na figura 5, temos 3 pilares:

Planejamento. É um elemento essencial para o sucesso de uma avaliação de segurança. Nessa fase, deve-se coletar informações cruciais para a execução do processo, como a identificação dos ativos a serem avaliados, a seleção das ameaças relevantes contra esses ativos, os controles de segurança a serem aplicados para mitigar tais ameaças, além do desenvolvimento da abordagem a ser adotada na avaliação.

Uma avaliação de segurança deve ser tratada como um projeto, onde é fundamental contar com um plano de gerenciamento de projetos que aborde diversos aspectos, tais como metas e objetivos claros, escopo bem definido, requisitos específicos, funções e responsabilidades das equipes envolvidas, limitações conhecidas, fatores de sucesso, suposições, recursos necessários, cronograma de atividades e entregas esperadas.

Para esta dissertação em particular, decidiu-se optar por um escopo mais restrito, requisitos mínimos e equipes reduzidas, uma vez que esse é o propósito específico do estudo.

Dessa forma, ao utilizar uma abordagem cuidadosamente planejada, pode-se garantir que a avaliação de segurança seja conduzida de maneira eficiente, com foco nos aspectos mais relevantes para os objetivos do projeto. A ênfase na organização e no detalhamento do planejamento permitirá alcançar resultados sólidos, mesmo com recursos limitados, contribuindo para o sucesso geral desta dissertação e a obtenção de conclusões significativas para a área de segurança da informação.

Execução. Os principais objetivos da fase de execução são identificar vulnerabilidades e validá-las, explorando-as quando apropriado. Esta fase deve abordar as atividades associadas com o objetivo método e técnica de avaliação.

Pós-Execução. A fase de pós-execução concentra-se nas ações de mitigação das vulnerabilidades, preparação dos relatórios, estabelecer recomendações de mitigação caso as mesmas não puderem ser mitigadas e desenvolver uma conclusão final (SCARFONE et al., 2008).

A fase de planejamento será abordada como parte da metodologia da dissertação e as fases de execução e pós-execução em discussão e resultados.

•Definição do ponto de vista do teste. Interno ou Externo, *Black box* ou *Gray box*.
•Coleta dos dados do sistema e aplicação a ser testada.
•Definição das ferramentas e técnicas de varredura e descoberta.
•Definição de técnicas de exploração das vulnerabilidades.

•Execução da varredura e descoberta com base no planejamento.
•Execução da exploração das vulnerabilidades.

•Preparação do relatório das vulnerabilidades e explorações.
•Ações de mitigação das vulnerabilidades.
•Definir procedimentos de rotina para monitoramento.

Figura 5 - Planejamento faseado do teste de penetração.

Fonte: Autor

3.3 Etapa planejamento

3.3.1 Definição do ponto de vista do teste

Os testes podem ser executados de vários pontos de vista, por exemplo, com que facilidade um invasor externo ou um insider mal-intencionado ataca com sucesso um sistema? Outro aspecto dos pontos de vista, a saber, o conhecimento prévio que os avaliadores têm do alvo ou ambiente alvo.

O teste de segurança externo é realizado fora do perímetro de segurança da organização. Isso oferece a capacidade de visualizar a postura de segurança do ambiente conforme ela aparece fora do perímetro de segurança, geralmente como vistos na Internet, com o objetivo de revelar vulnerabilidades que podem ser exploradas por um atacante.

Para testes de segurança interno, os avaliadores trabalham na rede interna e assumem a identidade de uma pessoa com informações privilegiadas ou um invasor que penetrou nas defesas do perímetro. Este tipo de teste pode revelar vulnerabilidades que podem ser exploradas e demonstra o dano potencial que esse tipo de invasor pode causar. O teste de segurança interno também se concentra na segurança e configuração no nível do sistema, incluindo configuração de aplicativos e serviços, autenticação, controle de acesso e fortalecimento do sistema.

O teste de segurança aberto, também conhecido como *White Box* ou *Gray Box*, este último o nível de conhecimento da estrutura a ser testada é parcial. Envolve a realização de testes externos e/ou internos com o conhecimento e consentimento da equipe de TI da organização, permitindo uma avaliação abrangente da postura de segurança da rede ou do sistema. Como a equipe de TI está totalmente ciente e envolvida nos testes, pode ser capaz de fornecer orientação para limitar o impacto do teste. Os testes também podem fornecer uma oportunidade de treinamento, com equipe observando as atividades e métodos utilizados pelos avaliadores para avaliar e potencialmente contornar as medidas de segurança implementadas. Isso fornece contexto aos requisitos de segurança implementados e mantidos pela equipe de TI e também pode ajudar a ensinar a equipe de TI como realizar testes.

O teste de segurança oculto, também conhecido como *Black Box*, adota uma abordagem adversária ao realizar testes sem o conhecimento da equipe de TI da organização, mas com total conhecimento e permissão da alta administração. Algumas organizações designam um terceiro confiável para garantir que o alvo organização não inicia medidas de resposta associadas ao ataque sem primeiro verificar se um ataque está realmente em andamento (por exemplo, que a atividade sendo detectada não se origina de um teste). Este tipo de teste é útil para testar controles técnicos de segurança, resposta da equipe de TI a incidentes de segurança percebidos e conhecimento e implementação da política de segurança da organização. Testes ocultos podem ser conduzidos com ou sem aviso (SCARFONE et al., 2008).

Na tabela 2 temos as vantagens e desvantagens para os tipos de testes *Black Box* e *Gray Box*.

Tabela 2 - Vantagens e desvantagens dos testes do tipo *Black Box* e *Gray Box*.

Tipo de teste	Vantagens	Desvantagens
Gray Box	• O *pentester* tem alguma informação sobre o sistema, o que pode permitir que ele concentre seus esforços em áreas específicas. • Pode ser mais eficiente, pois o *pentester* não precisa gastar tempo descobrindo informações que já são conhecidas.	• As informações fornecidas podem ser limitadas, o que pode limitar a eficácia do teste. • Pode ser difícil determinar a extensão das informações fornecidas, o que pode levar a resultados menos precisos.
Black Box	• O *pentester* não tem informações prévias sobre o sistema, o que pode permitir que ele descubra vulnerabilidades que não seriam detectadas de outra forma. • O *pentest* simula uma situação realista de ataque, onde um invasor não tem acesso privilegiado às informações internas do sistema.	• Pode ser menos eficiente do que um *pentest Gray Box*, já que o *pentester* precisa gastar mais tempo descobrindo informações básicas sobre o sistema. • Pode ser mais difícil de executar, já que o *pentester* precisa trabalhar com informações limitadas e descobrir o máximo possível por conta própria.

Fonte: Autor

Em geral, um *pentest Gray Box* pode ser mais eficiente, mas pode ser limitado pelas informações fornecidas. Já um pentest *Black Box* pode ser mais desafiador, mas pode fornecer uma visão mais completa e realista da segurança do sistema. A escolha entre os dois depende do objetivo e do escopo do teste de penetração.

3.3.2 Coleta dos dados do sistema e aplicação

Nesta etapa, serão coletadas as informações conforme o tipo de estratégia adotada em relação ao teste. Foi escolhido o *Gray Box*, em que teremos o conhecimento parcial da estrutura do SAHAN EDEN, como o IP externo, portas de comunicação e serviços associados a estas portas. Dados de IP serão suprimidos deste documento em face da confidencialidade da informação.

3.3.3 Definição das ferramentas e técnicas de varredura e descoberta

Nesta seção, discutimos e exploramos as principais ferramentas de varredura de vulnerabilidade de rede que podem ser benéficas para segurança cibernética, foram 10 ferramentas minuciosamente avaliadas e dentre elas as que foram escolhidas melhores se adaptam a realidade do teste. Na tabela 3 foi realizado um comparativo de todas as 10 ferramentas avaliadas.

1) Nessus: é um sistema escalável software de nível empresarial com mais de 80.000 usuários configuráveis plug-ins projetados para testar vulnerabilidades para uma variedade de dispositivos e aplicativos. Por exemplo, o Nessus pode escanear portas, identificar problemas de aplicativos da web, descobrir SO/software e tentar as credenciais padrão. Nessus categoriza cada

vulnerabilidade em diferentes níveis de risco ('Crítico', 'Alto', 'Médio', 'Baixo' e 'Nenhum') com base no padrão aberto da indústria Common Vulnerability Scoring System (CVSS) (WILLIAMS et al., 2017). O Nessus costumava ser uma ferramenta de código aberto, mas atualmente não é mais gratuita, é uma ferramenta proprietária comercializada pela empresa Tenable.

2) Nmap: Network Mapper é um utilitário gratuito e de código aberto para descoberta de rede e auditoria de segurança que é considerada uma das ferramentas de segurança mais populares devido à sua flexibilidade, potência, portabilidade e facilidade de uso. Ele é executado em todos os principais sistemas operacionais de computador e os pacotes binários oficiais estão disponíveis para Linux, Microsoft Windows, FreeBSD, OpenBSD, Solaris, IRIX, Mac OS X, HP-UX, NetBSD e Sun OS. O Nmap Scripting Engine (NSE) permite que os usuários escrevam seus próprios scripts para automatizar uma ampla variedade de tarefas de rede por meio do Nmap. Embora o NSE não seja um scanner de vulnerabilidade abrangente (o número de scripts disponíveis para o NSE é consideravelmente menor que do Nessus e do OpenVAS), os recursos de exploração de vulnerabilidade do NSE, sua integração com o restante das bibliotecas e ferramentas do pacote Nmap e a interface simples que a biblioteca NSE “vuln” fornece para criar relatórios de vulnerabilidade bem organizados que também podem conter referências CVE podem ser muito úteis para nossos propósitos (CHALVATZIS; KARRAS; PAPADEMETRIOU, 2019). O Nmap pode ser usado para escanear uma rede tão grande quanto ter milhares de hosts de computador, e até mesmo tão pequena quanto ter um único host. O Nmap pode ser encontrado em muitos sistemas, como Kali Linux.

3) Greenbone OpenVAS: O OpenVAS é um scanner de vulnerabilidades gratuito e de código aberto que se originou da versão GPL do Nessus (versão 2) depois que se tornou proprietário em 2005. O que é importante para nossos propósitos é que os plug-ins do OpenVAS ainda são escritos em Nessus Attack Scripting Language (NASL). O verificador de segurança real é acompanhado por um feed atualizado regularmente de testes de vulnerabilidade de rede (NVTs), mais de 128.000 no total. Todos os produtos OpenVAS são Software Livre. A maioria dos componentes é licenciada sob a GNU General Public License (GNU GPL).” Por ser de código aberto e gratuito ao usar NASL, é uma ótima alternativa (CHALVATZIS; KARRAS; PAPADEMETRIOU, 2019). O OpenVAS não é o scanner mais fácil de instalar e usar, mas é um dos scanners de segurança mais poderosos que você pode usar gratuitamente. Ele pode verificar milhares de vulnerabilidades e oferece gerenciamento falso positivo dos resultados da verificação.

4) Retina CS: Retina pode encontrar vulnerabilidades de rede, problemas de configuração e patches ausentes. Ele fornece varredura e correção e oferece suporte à verificação de vulnerabilidades em dispositivos móveis, servidores, aplicativos da Web e até nuvens privadas. A empresa mantenedora BeyondTrust decidiu não comercializar mais a ferramenta e descontinuou o produto definitivamente em 31 de dezembro de 2020 https://www.beyondtrust.com/vulnerability-management.

5) Microsoft Baseline Security Analyzer (MBSA): O MBSA pode identificar quaisquer service packs ausentes, patches de segurança e configurações incorretas de segurança. Você também pode especificar um único endereço IP ou um intervalo de endereços IP a serem verificados. Ele pode verificar senhas fracas, atualizações do Windows ou vulnerabilidades administrativas do SQL. Embora seja gratuito e fácil de usar, ele não verifica as configurações avançadas do Windows e está disponível apenas para o sistema operacional Windows. Atualmente também não é mais mantido pela Microsoft.

6) Nexpose Community Edition: Nexpose Community Edition verifica vulnerabilidades de rede, aplicativos da web, banco de dados e ambientes virtuais e pode ser instalado em Windows, Linux ou máquinas virtuais. No entanto, ele é limitado a apenas 32 endereços IP de cada vez, portanto, não é viável verificar uma rede de grande porte.

7) Dirbuster: É um aplicativo java multiprocessado projetado para força bruta de diretórios e nomes de arquivos em servidores da web/aplicativos. Muitas vezes um servidor web tem páginas e aplicativos ocultos dentro dele. O DirBuster tenta encontrá-los. No entanto, ferramentas dessa natureza geralmente são tão boas quanto os diretórios e a lista de arquivos que acompanham. A lista foi gerada do zero, rastreando a Internet e coletando o diretório e os arquivos que são realmente usados pelos desenvolvedores! O DirBuster vem com um total de 9 listas diferentes, o que torna o DirBuster extremamente eficaz para encontrar esses arquivos e diretórios ocultos. O DirBuster também tem a opção de executar uma força bruta pura, sem a necessidade de utilizar uma lista de palavras-chave. O projeto DirBuster original está inativo. No entanto, a equipe OWASP ZAP o bifurcou e criou um complemento de navegação forçada que pode ser carregado no OWASP ZAP.

8) BurpSuite: é uma ferramenta de código aberto para executar ou testar a segurança em um aplicativo ou sistema. PortSwigger Company cria Burp Suite usando a linguagem de programação Java. A capacidade de interceptar HTTP torna-se uma prioridade no Burp Suite. A principal função do Burp Suite é interceptar e exibir mensagens HTTP de maneira estruturada. O Burp Suite fornece ao testador uma breve visão geral do sistema de destino, todas as mensagens

e parâmetros enviados. Além disso, ele também fornece uma GUI com controle total sobre todas as mensagens – solte, encaminhe, repita, modifique, envie mais tarde e assim por diante. Assim, o testador pode projetar diferentes cenários de ataque e executá-los manualmente por meio do Burp Suite. Os resultados podem ser visualizados e analisados diretamente pelo testador. O Burp Suite possui uma versão comercial, como XSS, Brute Force e outros (RIADI; IFANI; KUSUMA, 2022).

9) Wapiti: É uma ferramenta open-source de varredura de vulnerabilidades de aplicações WEB, que procura por vulnerabilidades varrendo e injetando pacotes para identificar aplicações vulneráveis (AL ANHAR; SURYANTO, 2021). É gratuito e escrito em Python. Este poderoso scanner é capaz de detectar uma variedade de vulnerabilidades, incluindo Inclusão de Arquivo Malicioso, Injeção de SQL, Injeção de XML, Fixação de Sessão, Injeção de Comando, Vazamento de Informações e Cross-Site Scripting, Brute Force login, HTTP Security Headers, Log4Shell Vulnerability Detection, dentre outros. O Wapiti oferece a conveniência de gerar relatórios de resultados em formatos TXT, HTML, XML e JSON, facilitando a análise e o acompanhamento das descobertas de segurança. Sua natureza de código aberto torna-o uma escolha confiável para avaliar a segurança de aplicações web, fornecendo uma ferramenta valiosa para identificar e mitigar potenciais ameaças e vulnerabilidades. (CHAHAL; BALI; KHOSLA, 2022).

10) Owasp ZAP: O OWASP Zed Attack Proxy (ZAP) é uma ferramenta de teste de penetração integrada e de fácil utilização, projetada para identificar vulnerabilidades em aplicativos da web. Sua abordagem busca atender a um público diversificado, abrangendo desde desenvolvedores até testadores funcionais, independentemente do nível de experiência em segurança. Dessa forma, o ZAP é uma escolha ideal tanto para aqueles que estão dando os primeiros passos no campo de testes de penetração, como também uma valiosa adição à caixa de ferramentas de testadores experientes. Com uma interface intuitiva e recursos poderosos, o OWASP ZAP oferece a capacidade de realizar avaliações abrangentes de segurança em aplicações web, fornecendo resultados valiosos para fortalecer a proteção contra ameaças cibernéticas e garantir a robustez dos sistemas em questão (MAKINO; KLYUEV, 2015).

Tabela 3 - Tipos de de ferramentas de varreduras e descoberta

Ferramenta	Tipo de ferramenta	Licença	Versão	Última atualização	Custo aproximado	Observações
Nessus	Scanner de Vulnerabilidades	Proprietária	10.5.1	Março de 2023	A partir de US$ 3.390	–
Nmap	Scanner de rede	Livre	7.93	Setembro de 2022	Gratuito	–
Greenbone OpenVAS	Scanner de Vulnerabilidades	Livre (Community Edition) Proprietária	22.4.0	Julho de 2022	Não informado pelo fabricante	Versão gratuita com feeds de atualizações limitados
Retina CS	Scanner de vulnerabilidades	Proprietária	2020	Junho de 2020	Produto descontinuado pelo fabricante	–
Microsoft Baseline Security Analyzer	Hardening	Livre	2.3	Dezembro de 2013	Gratuito	Somente para Windows. Descontinuado.
Nexpose Vulnerability Scanner	Scanner de vulnerabilidades	Livre (Community Edition) Proprietária	6.6.39	Março de 2019	Não informado pelo fabricante	Versão gratuita não está mais disponível. Limitações de uso
Dirbuster	Ferramenta de descoberta	Livre	1.0RC1	Setembro de 2008	Gratuito	O projeto está inativo e o OWASP ZAP o substituiu
BurpSuite Community Edition	Ferramenta de testes de segurança	Livre (Community Edition)	2023.4.2	Abril de 2023	A partir de US$ 449/ano na versão professional	–
Wapiti	Ferramenta de testes de segurança	Livre	3.1.7	Março de 2023	Gratuito	–
Owasp Zap	Ferramenta de testes de segurança	Livre	2.12.0	Outubro de 2022	Gratuito	–

Fonte: Autor

Baseado na coleta de dados, o autor optou por realizar um teste externo, adotando o nível de conhecimento *Gray Box* para a avaliação de segurança do sistema e aplicação em questão. Para a coleta dos dados do sistema e aplicação, foi empregada uma abordagem manual, permitindo uma análise detalhada e minuciosa das possíveis vulnerabilidades.

As ferramentas escolhidas para a varredura incluíram o Nmap, Greenbone OpenVAS, Wapiti e OWASP ZAP. O Nmap é uma ferramenta de código aberto amplamente utilizada para a exploração de redes e mapeamento de portas, fornecendo informações valiosas sobre os ativos e serviços expostos na infraestrutura alvo. O Greenbone OpenVAS é uma solução de varredura de vulnerabilidades, capaz de identificar potenciais brechas de segurança e fornecer uma análise abrangente do ambiente avaliado. O Wapiti é uma ferramenta específica para testes de segurança em aplicações web, detectando vulnerabilidades como Injeção de SQL, Cross-Site Scripting e muito mais. Por fim, o OWASP ZAP é uma ferramenta integrada, fácil de usar e altamente eficaz para identificar vulnerabilidades em aplicativos web.

Com base nos resultados que serão obtidos pelas ferramentas de varredura, o autor planejou a estratégia de exploração, escolhendo as ferramentas adequadas para explorar as vulnerabilidades encontradas. Essa seleção cuidadosa garante que as explorações sejam feitas de forma controlada e ética, com o objetivo de validar a gravidade das vulnerabilidades e fornecer recomendações para a correção e mitigação das mesmas.

A tabela 4 demonstra de forma resumida as etapas e métodos escolhidos na fase de planejamento.

Tabela 4 - Resumo das etapas e métodos escolhidos no planejamento

	Etapa	Método escolhido
Planejamento	Ponto de vista do teste	Externo
	Nível de conhecimento	Gray Box
	Coleta dos dados de sistema e aplicação a ser testada	Manual
	Ferramentas de varredura/descoberta	Nmap, Greenbone OpenVAS, Wapiti e OWASP ZAP
	Definição das técnicas de exploração	a definir

Fonte: Autor

4. RESULTADOS E DISCUSSÃO

Ainda na etapa de planejamento, iremos coletar os dados de sistema e aplicação do SAHANA EDEN, etapa fundamental para início da fase de execução dos testes.

4.1 Coleta de dados

Nesta etapa, todo o processo foi conduzido manualmente, abrangendo desde a coleta de informações sobre a configuração do servidor onde o SAHANA EDEN está instalado até a identificação do sistema operacional, versões de softwares, kernel e o equipamento a ser utilizado para executar a varredura de vulnerabilidades.

A abordagem manual adotada nesse levantamento de dados possibilitou uma análise minuciosa e detalhada de todos os aspectos do ambiente do SAHANA EDEN. Ao realizar essa coleta de informações de forma cuidadosa e manual, é possível garantir uma compreensão precisa da infraestrutura em questão, permitindo identificar possíveis pontos fracos e áreas críticas que merecem uma atenção especial durante a varredura de vulnerabilidades.

A tabela 5 ilustra todos os dados coletados do sistema e aplicação do SAHANA EDEN, dados confidenciais como endereço IP e qualquer outra informação que possa impactar na privacidade e confidencialidade do sistema são suprimidas durante todo o documento.

Tabela 5 - Coleta de dados sistema e aplicação SAHANA EDEN

	Tipo	CPU	Memória	Disco	Rede	SO	Kernel	IP
Servidor	Virtual	Intel 4 cores de 2.59 Ghz	4GB	32 GB	1 Gbps	Debian Buster versão 10.2.1-6	Linux 5.10.0-9-amd64	Suprimido

	Nome	Versão
Softwares	SAHANA EDEN	eden-dev-56c535867 (2021-10-15 19:54:31)
	Postgres SQL	psql (PostgreSQL) 13.4 (Debian 13.4-0+deb11u1)
	Python	3.9.2
	Web Server	gunicorn/20.1.0
	Web2py	2.24.1-stable+timestamp.2023.03.23.05.07.17
	lxml	4.6.4.0
	ReportLab	3.6.2
	Shapely	1.8.0
	xlrd	2.0.1
	xlwt	1.3.0

	Tipo	CPU	Memória	Disco	SO
estação de varredura	Notebook	Intel Core i7-8565 1.80 Ghz	8GB	N/A	Virtual Box - Kali Linux - Debian 6.1.27-1kali1 (2023-05-12)

Fonte: Autor

4.2 Etapa Execução

Uma vez com os dados coletados, prepara-se para a varredura de vulnerabilidade. Foi utilizada a estação de varredura com a configuração conforme descrita na tabela 5 e as ferramentas de varredura e descoberta da tabela 4. Na varredura, foram considerados aproximadamente 128.000 NVT's (Network Vulnerability Tests), sendo uma base de dados com vulnerabilidades potenciais conhecidas, cerca de 189.000 CVE's (Common Vulnerabilities and Exposure) lista das vulnerabilidades publicadas por fabricantes e pesquisadores e aproximadamente 1.020.000 CPE's (Common Platform Enumeration) lista de nomenclaturas padronizadas para produtos utilizados em TI.

Para cada ferramenta escolhida na etapa dos testes, será seguido o fluxo de trabalho conforme figura 6. Caso uma mesma vulnerabilidade seja descoberta em mais de uma ferramenta, a etapa de exploração ou mitigação será executada apenas uma única vez.

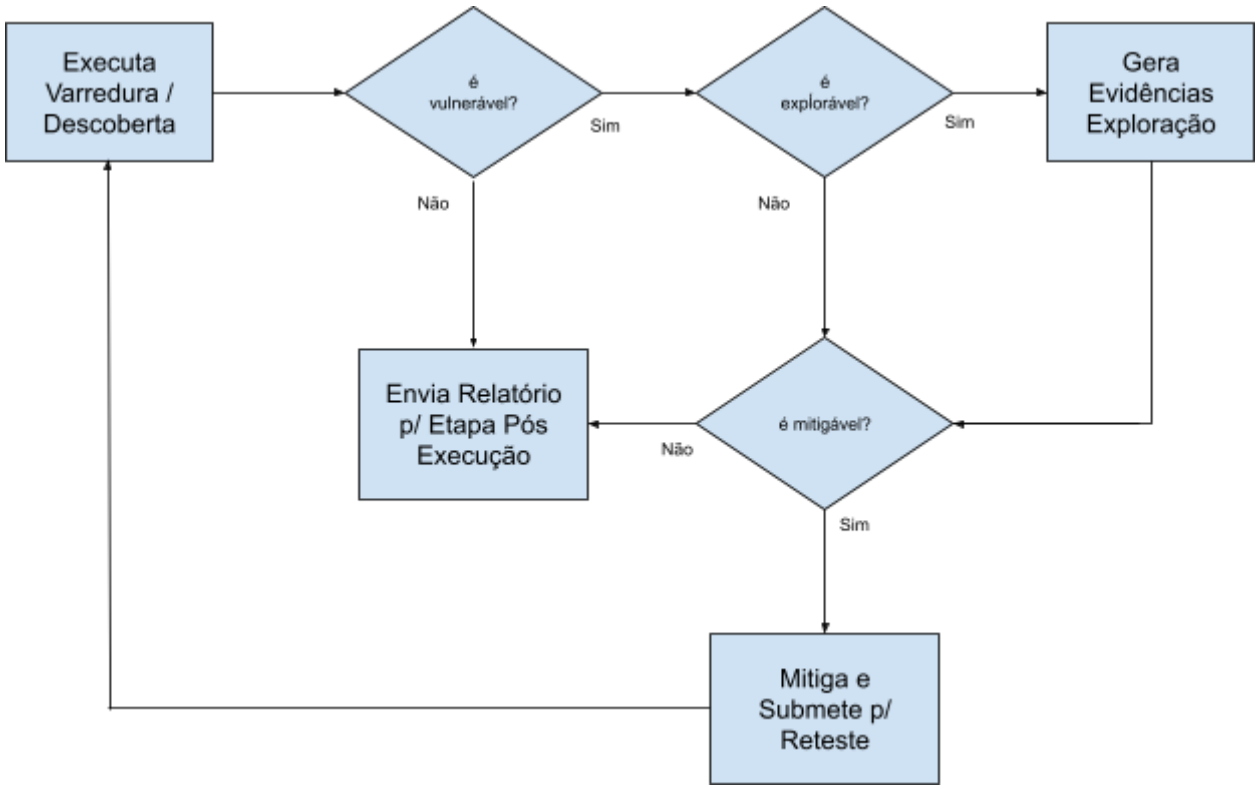

Figura 6 - Fluxograma de execução da varredura, exploração e mitigação.

Fonte: Autor

4.2.1 Nmap

Foi executada a varredura utilizando o Nmap, a versão no momento da execução era a 7.93 e a data de execução em 11/02/2023.

O Nmap é um software de arquitetura básica e não possui banco de dados, ou seja uma ferramenta simples por linha de comando.

Parâmetros utilizados no nmap:

-sC - executa a varredura utilizando scripts padrão de detecção

-sV - exibe as informações de versões dos serviços detectados

-p8000,60000 - varre somente as portas TCP 8000 e TCP 60000, portas estas fornecidas pelo time de TI de acordo com a estratégia Gray Box de varredura.

Por último o endereço IP, este foi suprimido por questões de confidencialidade.

Comando completo do nmap:

nmap -sC -sV -p8000,60000 2xx.xxx.xxx.xxx

Resultado da varredura do nmap

Starting Nmap 7.93 (https://nmap.org) at 2023-02-11 23:20 -03

Nmap scan report for 2xx.xxx.xxx.xxx Host is up (0.0047s latency).

```
PORT   STATE SERVICE  VERSION
8000/tcp  open  http-alt gunicorn
|_http-title: xxxxx_xxx
|_http-server-header: gunicorn
| fingerprint-strings:
|   FourOhFourRequest:
|         HTTP/1.0 400 BAD REQUEST
|         Server: gunicorn
|         Date: Sun, 12 Feb 2023 02:20:58 GMT
|         Connection: close
|         Content-Type: application/json
```

| Set-Cookie: session_id_eden=1xx.xxx.xxx.xx-dce4bacb-3df7-4313-bda7-4627bfc7fd37; HttpOnly; Path=/; SameSite=Lax
| Content-Length: 67
| {"status": "failed", "statuscode": "400", "message": "BAD REQUEST"}
| GenericLines:
| HTTP/1.1 400 Bad Request
| Connection: close
| Content-Type: text/html
| Content-Length: 193
| <html>
| <head>
| <title>Bad Request</title>
| </head>
| <body>
| <h1> Bad Request </h1>
| Invalid Request Line 'Invalid HTTP request line: '''
| </body>
| </html>
| GetRequest:
| HTTP/1.0 200 OK
| Server: gunicorn
| Date: Sun, 12 Feb 2023 02:20:53 GMT
| Connection: close
| X-Powered-By: web2py
| Content-Type: text/html; charset=utf-8
| Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
| Expires: Sun, 12 Feb 2023 02:20:53 GMT
| Pragma: no-cache
| Set-Cookie: session_id_eden=1xx.xxx.xxx.xxx-75ec71d7-76e2-455c-a50d-d3468cebbcc9; HttpOnly; Path=/; SameSite=Lax
| <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

```
|       <html xmlns="http://www.w3.org/1999/xhtml" lang="pt-br">
|       <head>
|       <meta http-equiv="content-type" content="text/html; charset=utf-8" />
|       <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
|       <title>xxxxx_xxx</title>
|       <meta name="application-name" content="eden" />
|       <meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1" />

60000/tcp open  ssh    OpenSSH 8.4p1 Debian 5 (protocol 2.0)
| ssh-hostkey:
|   3072 affd579aad6d4175b9968d978e4dff07 (RSA)
|   256 fb9a60d60a79671befae8a89b001cb5a (ECDSA)
|_  256 90c5f11fbdb4446c0d7a280f691fb525 (ED25519)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 136.83 seconds
```

4.2.2 Greenbone OpenVAS

Foi executada a varredura utilizando o Greenbone OpenVAS, a versão no momento da execução era a 22.4 e a data de execução em 03/06/2023.

A arquitetura da ferramenta, como mostra a figura 7, do Greenbone OpenVAS *Community Edition* é agrupada em três partes principais:

- Aplicativo de verificação executável que executa testes de vulnerabilidade (VT) em sistemas de destino;
- Instância executável do gerenciador de vulnerabilidade Greenbone (gvmd);
- Greenbone Security Assistant (GSA) com a instância do Greenbone Security Assistant (gsad).

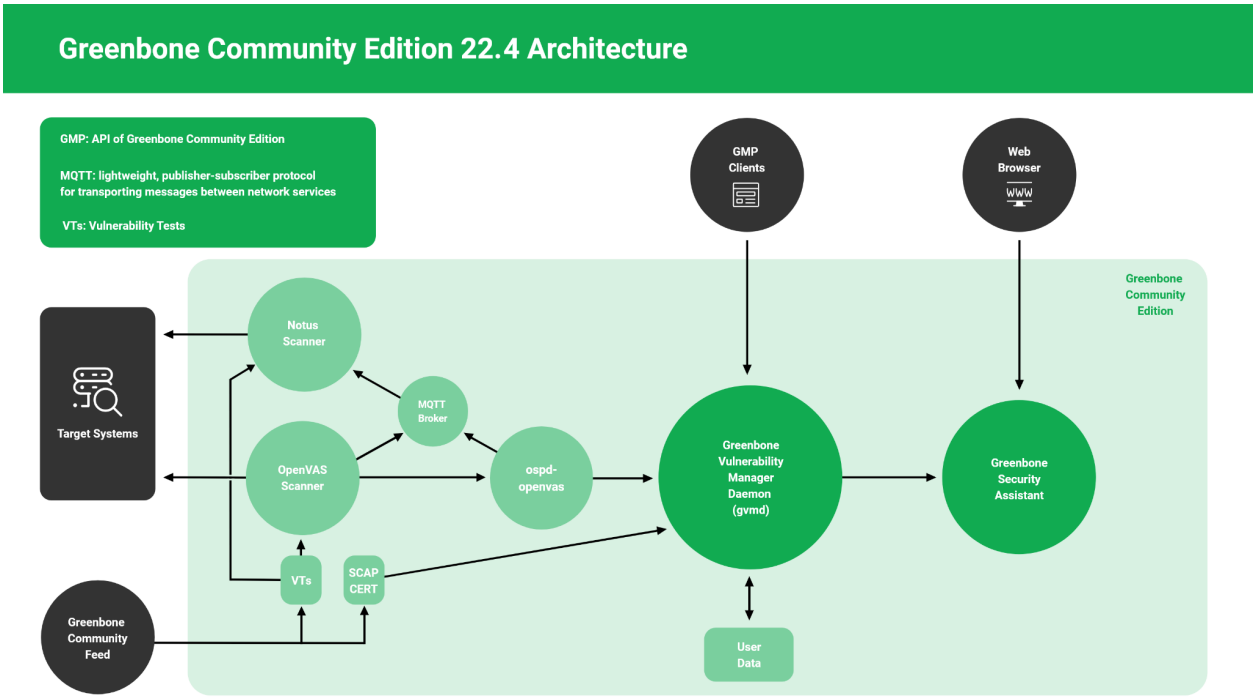

Figura 7 - Arquitetura do Greenbone OpenVAS *Community Edition*.
Fonte: https://greenbone.github.io/docs/latest/architecture.html

O Greenbone *Vulnerability Manager* (gvmd) é o serviço central que consolida a verificação simples de vulnerabilidades em uma solução completa de gerenciamento de vulnerabilidades. O gvmd controla o OpenVAS Scanner via Open Scanner Protocol (OSP).

O serviço em si oferece Greenbone *Management Protocol* (GMP) baseado em XML e sem estado. O gvmd também controla um banco de dados SQL (PostgreSQL) onde todos os dados de configuração e resultados de verificação são armazenados centralmente. Além disso, o gvmd também lida com o gerenciamento de usuários, incluindo controle de permissões com grupos e funções. E, finalmente, o serviço possui um sistema de tempo de execução interno para tarefas agendadas e outros eventos.

O Greenbone *Security Assistant* (GSA) é a interface da web com a qual um usuário controla as varreduras e acessa as informações de vulnerabilidade. É o principal ponto de contato para um usuário. Ele se conecta ao gvmd por meio do servidor da Web a instância do Greenbone *Security Assistant* (gsad) para fornecer um aplicativo da Web completo para gerenciamento de vulnerabilidades. A comunicação ocorre por meio do Greenbone *Management Protocol* (GMP) com o qual o usuário também pode se comunicar diretamente por meio de diferentes ferramentas.

Feed Status

São todos os tipos de dados que a ferramenta utiliza para a execução da varredura de vulnerabilidades.

Das atualizações de dados comunitários, iremos abordar as mais importantes e as que iremos utilizar nesta dissertação, que são, NVT, SCAP e GVMD_DATA. A versão da base de dados das atualizações é de 02/06/2023, conforme demonstra a figura 8.

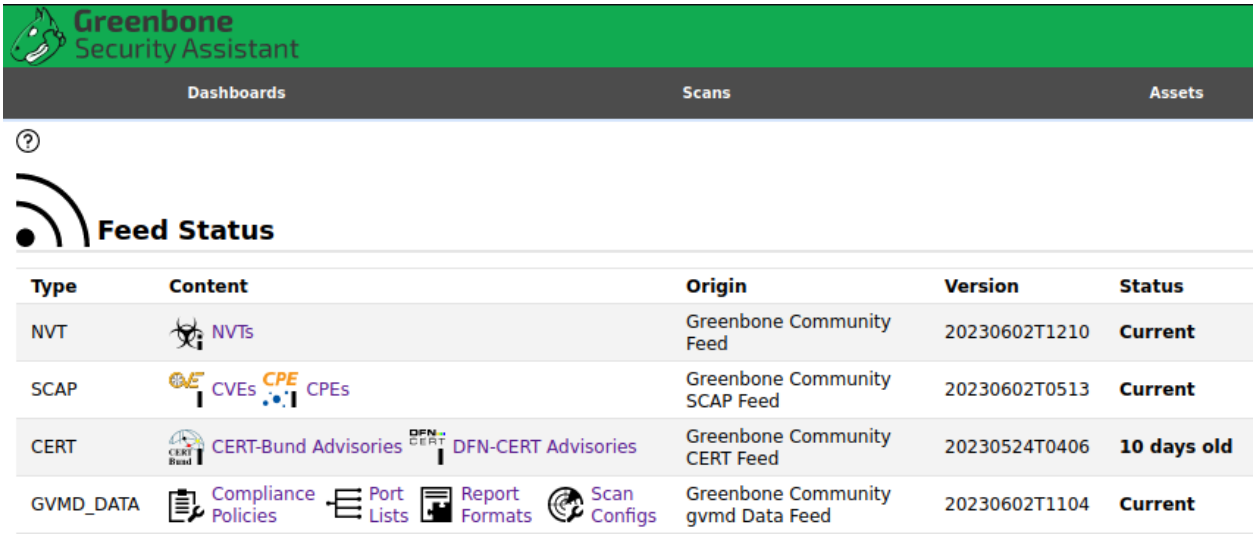

Figura 8 - Base de dados comunitários para varredura de vulnerabilidades.

Fonte: Autor

NVT

Network Vulnerability Tests - NVT's ou simplesmente VT' s são rotinas de teste usadas pelo sistema. Eles fazem parte do Greenbone Community Feed, que é atualizado regularmente. Os VTs incluem informações sobre data de desenvolvimento, sistemas afetados, impacto de vulnerabilidades e correção. A combinação de vários NVTs é usada para detectar a vulnerabilidade.

Na prática, funciona da seguinte forma: o host de destino é atacado com um plug-in pré-escrito. Se as informações do o feedback do plug-in for válida, está provado que o host de destino contém a vulnerabilidade relacionada.

O script é um tipo de plug-in, que é escrito em uma linguagem de programação específica. Atualmente, na maioria das varreduras de vulnerabilidades de segurança os

mecanismos usam scripts para verificação de vulnerabilidades. Essa técnica de uso de scripts facilita muito a manutenção e atualização de dados de vulnerabilidades (WANG et al., 2020).

Todos os VTs existentes podem ser exibidos selecionando **SecInfo > NVTs** na barra de menus.

Para todos os VTs, conforme figura 9, as seguintes informações são exibidas:

Nome

Nome do VT.

Família

Família de VTs à qual o VT pertence. Ex. Boletins Windows, Geral.

Criado

Data e hora da criação.

Modificado

Data e hora da última modificação.

CVE (*Common Vulnerabilities and Exposures*)

CVE que é verificado para uso do VT.

Tipo de solução

Solução para a vulnerabilidade. As seguintes soluções são possíveis:

Um patch de fornecedor está disponível.

Uma solução alternativa está disponível.

Uma mitigação por configuração está disponível.

Nenhuma correção está e estará disponível.

Nenhuma solução existe.

Gravidade

A gravidade da vulnerabilidade (CVSS) é exibida como uma barra para apoiar a análise dos resultados.

QoD

QoD é a abreviação de *Quality of Detection* e representa o percentual de confiabilidade da detecção de uma vulnerabilidade.

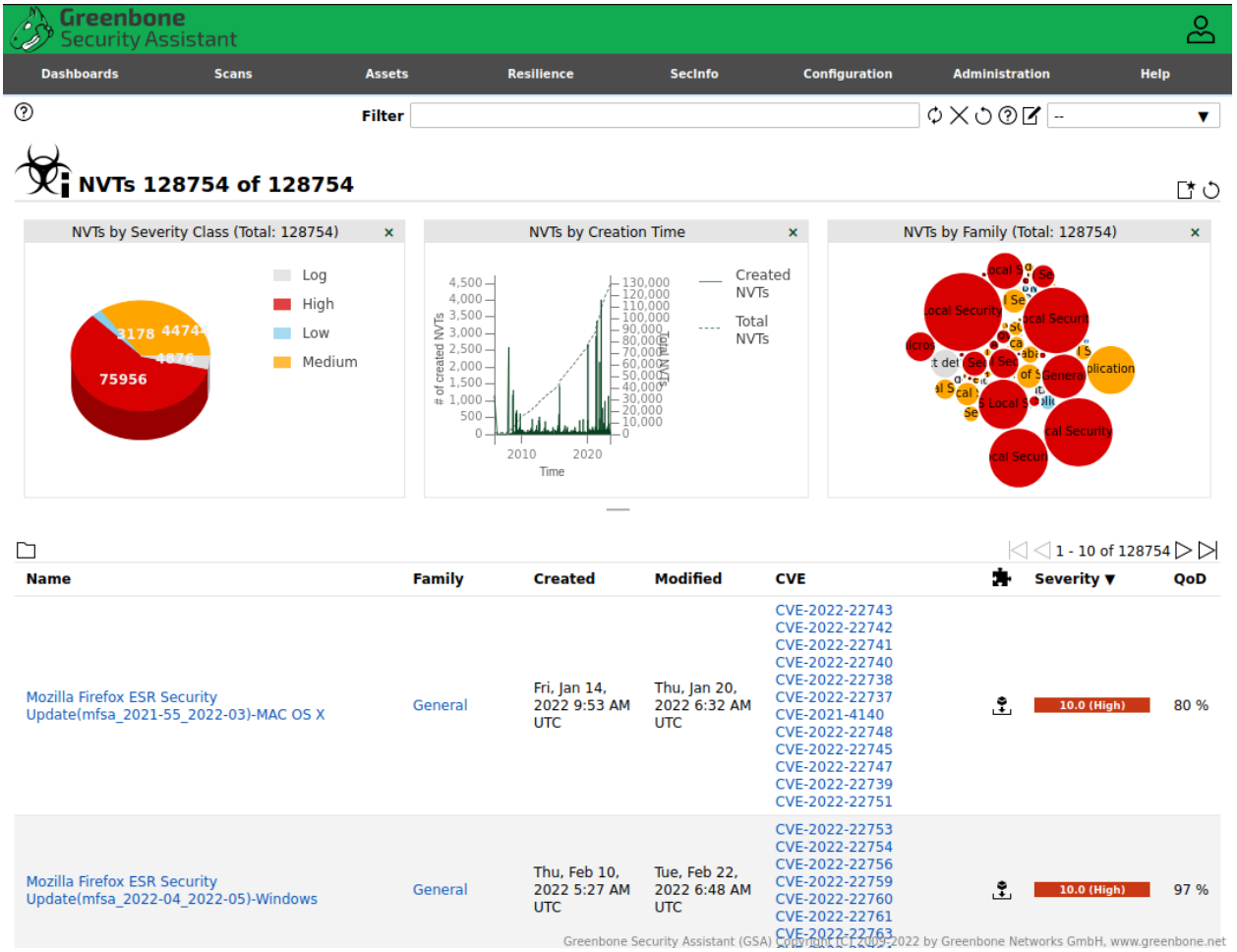

Figura 9 - Ilustra a quantidade de VT's suportados pelo Greenbone OpenVAS.

Fonte: Autor

GVMD_DATA

Esta base de dados faz parte da configuração padrão da própria ferramenta. Tem como fonte a atualização comunitária. Existem 4 tipos de dados, que serão listados a seguir, mas nesta dissertação, iremos efetivamente apenas 2 deles com maiores detalhes.

***Compliance Policies* (Políticas de Compliance)**

Contém uma lista de dados pré-definidos de configurações de equipamentos, dados esses que demonstram se um dispositivo está compliance com o baseline de segurança ou não, dentro das políticas de compliance contém VT's específicos para validação.

Port Lists (Lista de Portas)

Contém uma lista com as principais portas de rede que podem ser utilizadas na varredura, comumente se utilizam em uma varredura a configuração padrão, que varrem inúmeras portas, mas como nosso teste trata-se de um *Gray Box*, foi customizada a lista de portas para a varredura somente nas portas que estavam sendo utilizadas pelo SAHANA EDEN.

***Reports Formats* (Formatos de Relatórios)**

Esta configuração traz os formatos disponíveis onde os relatórios poderão ser gerados, PDF, TXT, CSV, etc.

Scan Configs (Configurações de Varredura)

Esta configuração lista quais são as possíveis combinações de NVT's que podem ser utilizadas em uma varredura, por exemplo, para descobrir qual sistema operacional está sendo utilizado na máquina alvo, existe uma configuração chamada *Host Discovery,* que rodam apenas 2 NVT's. Na nossa varredura iremos utilizar todos os NVT's possíveis e a configuração para isto é chamada de *Full and Fast*, como demonstrado na figura 10.

SCAP

O *Security Content Automation Protocol* (SCAP) é um conjunto de especificações que padronizam o formato e nomenclatura pela qual as informações de falha de software e configuração de segurança são comunicadas, ambas para máquinas e humanos. O SCAP é uma estrutura multifuncional de especificações que suportam configuração, vulnerabilidade e verificação de patches, atividades de conformidade de controle técnico e medição de segurança. SCAP fez uso de suítes de especificações abertas mantidas individualmente que são acessíveis ao público. O NIST dos EUA reuniu essas especificações abertas para desenvolver SCAP, levando assim vantagem de suas capacidades individuais (ADETUNJI; BUTAKOV; ZAVARSKY, 2018).

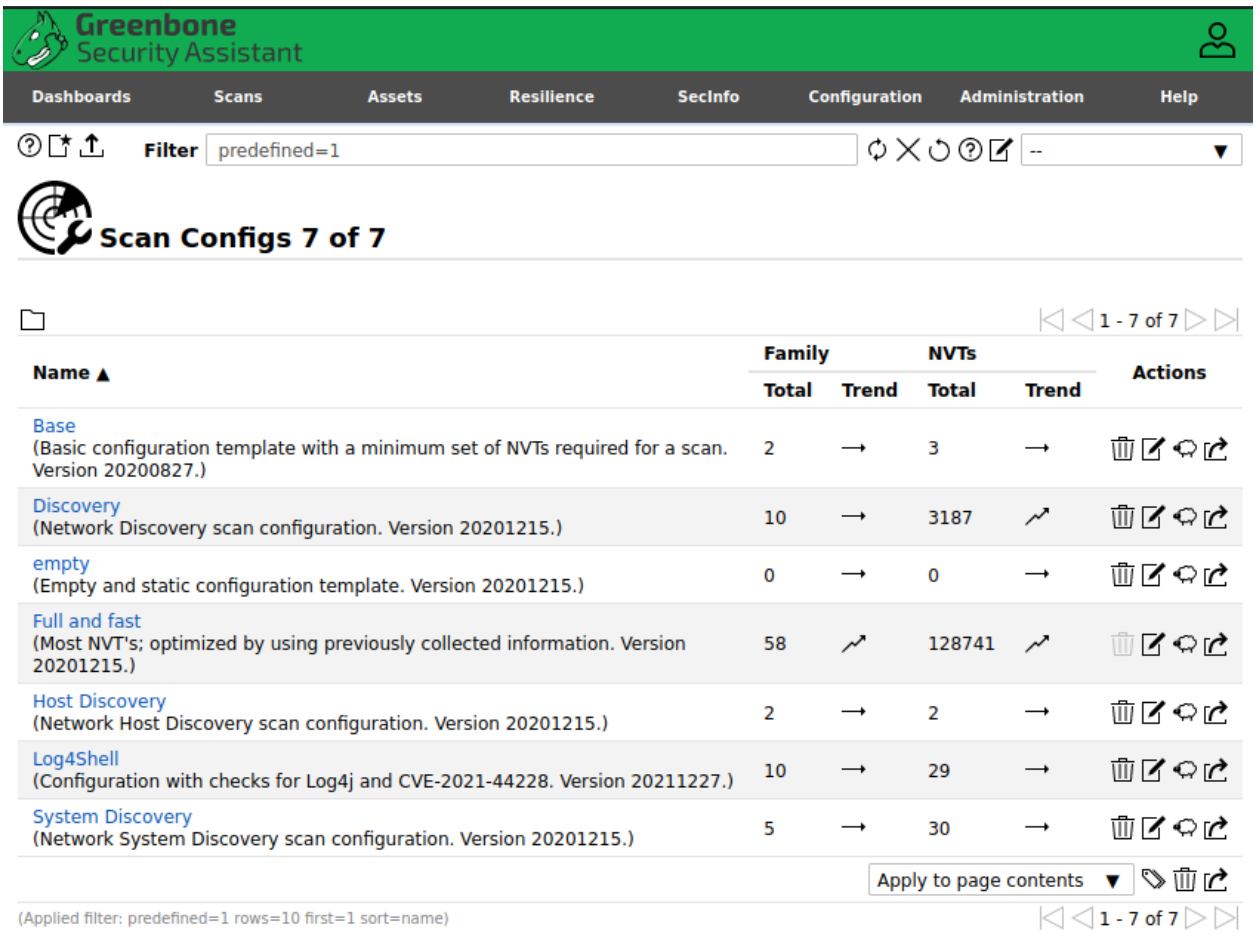

Figura 10 - Lista com todas as configurações padrões de varreduras.

Fonte: Autor

Os objetivos para o desenvolvimento do SCAP incluem padronizar o gerenciamento de segurança do sistema, promovendo a interoperabilidade de produtos de segurança e promovendo o uso de expressões padrão de conteúdo de segurança.

De 5 categorias que o protocolo possui, Linguagens, Formatos de Relatórios, Enumerações, Medidas/Sistema de Pontuação e Integridade, iremos abordar basicamente duas, que são Enumerações e Medidas/Sistema de Pontuação.

Enumerações

Cada enumeração SCAP define uma nomenclatura padrão (formato de nomenclatura) e um dicionário oficial ou lista de itens expressos usando essa nomenclatura. São *Common Platform Enumeration* (CPE), *Common Configuration Enumeration* (CCE) e *Common Vulnerabilities and Exposures* (CVE).

CPE

O CPE *(Common Platform Enumeration)* enumeração de plataforma comum é uma forma de descrever hardware, aplicações e sistemas operacionais (incluindo firmware) de forma legível por máquina, onde o registro NVD CVE (*National Vulnerability Database - Common Vulnerability Enumeration*) indica a lista de CPEs, início e fim das versões da vulnerabilidade.

Os CPEs são armazenados em um dicionário principal mantido pelo NVD, que os anexa a um CVE se um produto for afetado. Como exemplo, o Vetor CPE cpe:2.3:o:siemens: simatic_s7-1500_firmware:*:*:*:*:*:*:*:* pode ser quebrado da seguinte forma - CPE Versão 2.3 (emitida em 2011), a 'Parte' é uma opção sistema operacional (:o:), o Fornecedor é a Siemens, o Produto é o Firmware SIMATIC S7-1500 e os asteriscos representam qualquer versão, atualização, idioma, Edição, Idioma, Edição de Software, Target Software, Target Hardware é afetado (THOMAS et al., 2020).

CCE

CCE *(Common Configuration Enumeration)* enumeração de configuração comum são identificadores padronizados para problemas e exposições de segurança de configuração e mantidos pela Corporação MITRE. CCE emite um identificador exclusivo a cada configuração particular relacionada à segurança.

Por outro lado, os IDs CCE fornecem uma etiqueta para diferenciar perfis. A lista CCE pode correlacionar dados de configuração em várias fontes de informação com rapidez e precisão. Atualmente, os Identificadores CCE correlacionam definições de configuração em diferentes documentos com melhores práticas do *Center for Internet Security* (CIS), Instituto Nacional de Padrões e Tecnologia (NIST), Agência de Segurança Nacional (NSA), e Agência de Sistemas de Informação de Defesa (DISA). Existem vários benefícios quando usamos o CCE, como correlação rápida de dados, facilidade de coleta de métricas para uso em consciência de situação, e implementação em automação (LIN; CHEN; LAIH, 2008)**.**

CVE

O CVE *(Common Vulnerabilities and Exposures)* vulnerabilidades e exposições comuns são classificados como um dicionário de identificadores CVE para conhecimento público vulnerabilidades, que inclui três aspectos: 1) Número de identificador CVE (por exemplo, “CVE-2017-6798”, 2) Descrição da vulnerabilidade ou exposição de segurança padronizada, e 3) algumas referências pertinentes (por exemplo, id CWE).

Como o banco de dados CVE fornece apenas um método de referência de vulnerabilidades, nos utilizamos o CVE *Details*, que é uma interface web (www.cvedetails.com), conforme figura 11, para a integração dos dados retirados do CVE e do National Vulnerability Database (NVD). Por meio do CVE *Details*, não apenas recuperamos as informações detalhadas de vulnerabilidades (como identificador CVE número, descrição da vulnerabilidade, pontuação CVSS avaliada por especialistas), mas também estatísticas de vulnerabilidade para fornecedores e produtos (HAN et al., 2017).

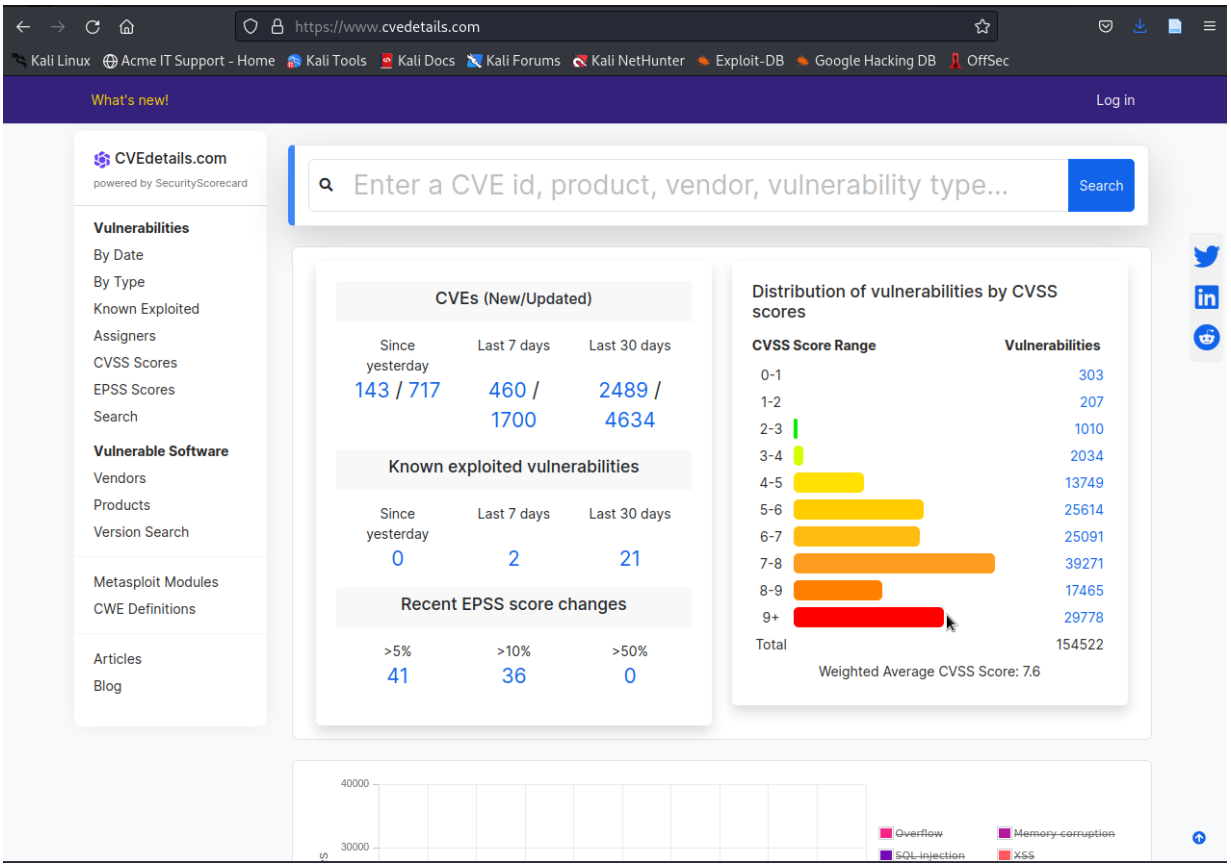

Figura 11 - Site CVE Details.

Fonte: www.cvedetails.com

Medidas/Sistema de Pontuação

No SCAP, isso se refere à avaliação de características específicas de um fraqueza de segurança (por exemplo, vulnerabilidades de software e problemas de configuração de segurança) e, com base nessas características, gerando uma pontuação que reflete sua gravidade relativa. O SCAP as especificações do sistema de medição e pontuação são o *Common Vulnerability Scoring System* (CVSS) Sistema de Pontuação de Vulnerabilidade Comum e *Common Configuration Scoring System* (CCSS) Sistema de Pontuação de Configuração Comum.

CVSS

Os resultados das vulnerabilidades são mensurados quanto ao seu grau de exposição utilizando o CVSS (*Common Vulnerability Scoring System*) que é um framework público utilizado para avaliar e quantificar o impacto de vulnerabilidades identificadas em software. Organizações como CISCO, NIST (National Institute of Standards and Tecnology), ORACLE, geram estas pontuações de CVSS (MELL; SCARFONE; ROMANOSKY, 2006).

O CVSS é composto por 3 grupos: Base, Temporal e Ambiental. Cada grupo produz uma pontuação numérica que varia de 0 a 10 e um vetor, uma representação textual compactada que reflete os valores usados para derivar a pontuação. O grupo Base representa as qualidades intrínsecas de uma vulnerabilidade. O grupo Temporal reflete as características de uma vulnerabilidade que muda ao longo do tempo. O grupo Ambiental representa as características de uma vulnerabilidade que são exclusivas do ambiente de qualquer usuário.

CCSS

O sistema de pontuação de configuração comum (CCSS) é um conjunto de medidas da gravidade dos problemas de configuração de segurança de software publicados pela Divisão de Segurança Computacional do Laboratório de Tecnologia da Informação do Instituto Nacional de Padrões e Tecnologia (NIST). O CCSS resolve problemas de configuração de segurança de vulnerabilidades de software (WICAKSANA; WIRA, 2022).

Configuração do Host

Nesta etapa, configura-se o endereço da máquina para execução da varredura, a configuração é feita no menu “Assets->Hosts”. Apenas é configurado o IP do servidor do SAHANA EDEN, conforme pode ser visualizado na figura 12.

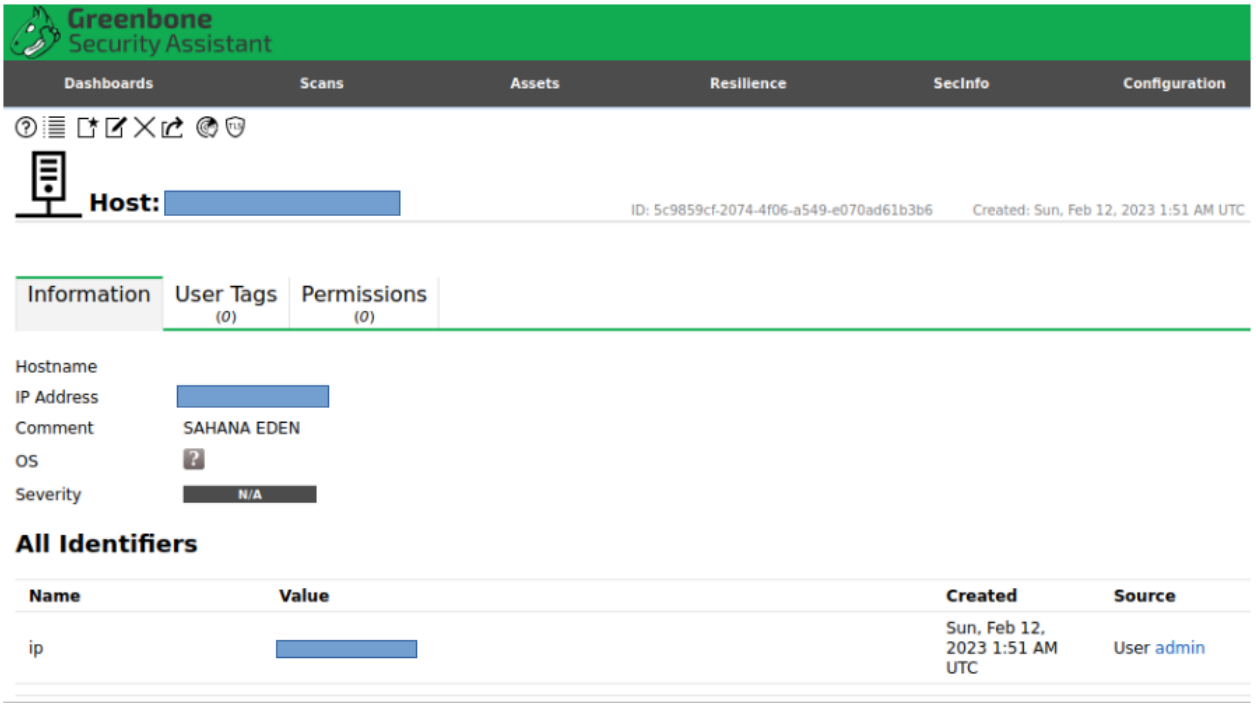

Figura 12 - Página de configuração do IP do servidor para varredura.

Fonte: Greenbone OpenVAS

Configuração das portas a serem varridas

Nesta etapa apenas configuramos as portas TCP 8000 e TCP 60000, portas estas fornecidas pelo time de TI de acordo com a estratégia *Gray Box* de varredura. A configuração é realizada através do menu "*Configurations->Port Lists*". As portas devem ser configuradas como demonstrado na figura 13. Tanto a porta de início e a de fim devem ter a mesma numeração, caso contrário o sistema entenderá que será um conjunto de portas e irá varrer todas que estão dentro daquele conjunto.

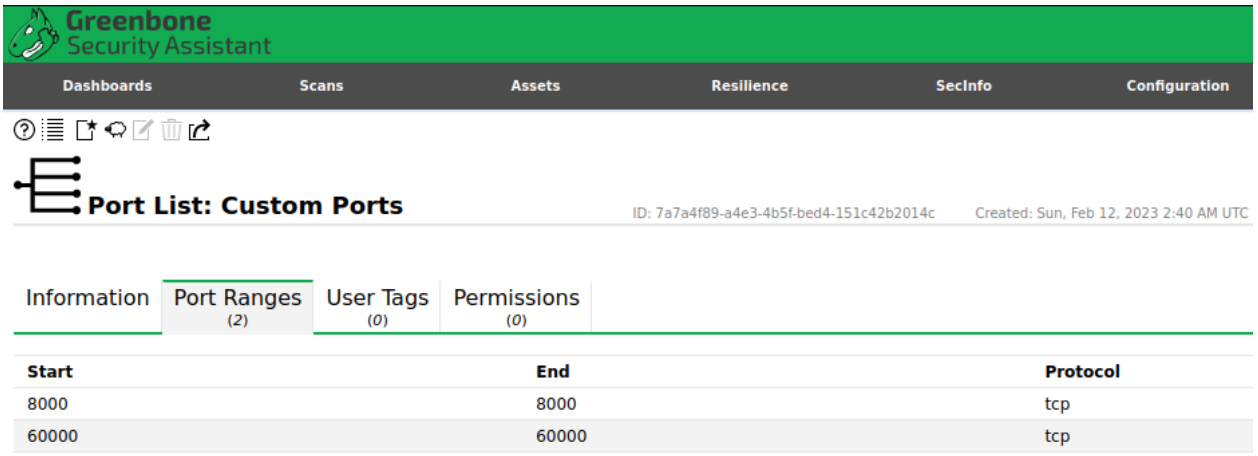

Figura 13 - Página de configuração da Lista de Portas utilizadas na varredura.

Fonte: Greenbone OpenVAS

Configuração do alvo

Nesta etapa, selecionamos o grupo de portas criadas anteriormente (*Custom Ports*) e a configuração do tipo de "*alive test*" escolhemos a opção "*consider alive*", essa configuração evita de que o greenbone utilize uma técnica de validação do endereço IP que comumente são bloqueadas por muitos firewalls modernos. Como já sabemos que esse IP existe, a opção "*consider alive"* é a mais adequada para o momento. A configuração é realizada em "*Configurations->Targets",* conforme pode ser visto na figura 14.

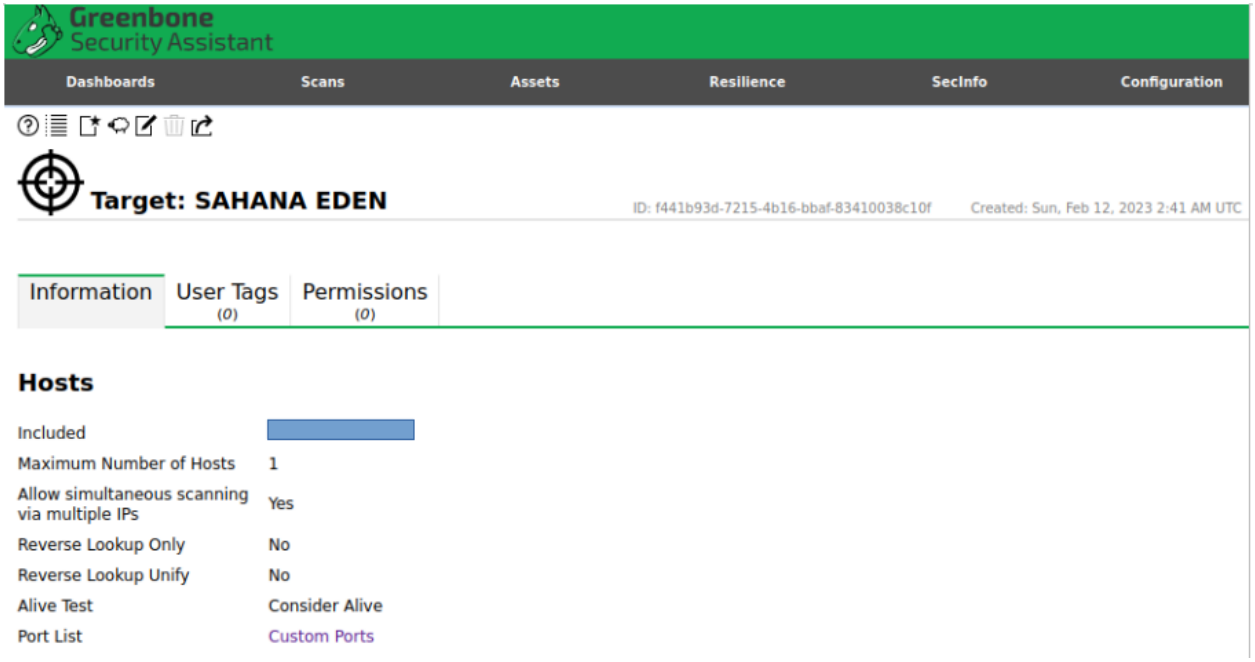

Figura 14 - Página de configuração do alvo da varredura.

Fonte: Greenbone OpenVAS

Configuração da tarefa

Nesta etapa é configurada a tarefa de varredura conforme o alvo configurado anteriormente, em "*Scan->Tasks*". Algumas configurações nesta fase irão determinar o quão será efetiva a varredura. Na opção "*Scanner/Type*", escolher a varredura pelo "*OpenVAS default*" e em "*Scan Config*" escolher o modo de varredura "*Full and Fast*" onde todos os VTs (*Vulnerabilities Tests)* são usados.

Definição do QoD (*Quality of Detection*), representa o quão sensível e confiável é a detecção de uma vulnerabilidade utilizando-se aquele VT. O valor padrão e recomendado é de

70%, as demais opções também são relacionadas a performance da varredura, conforme figura 15.

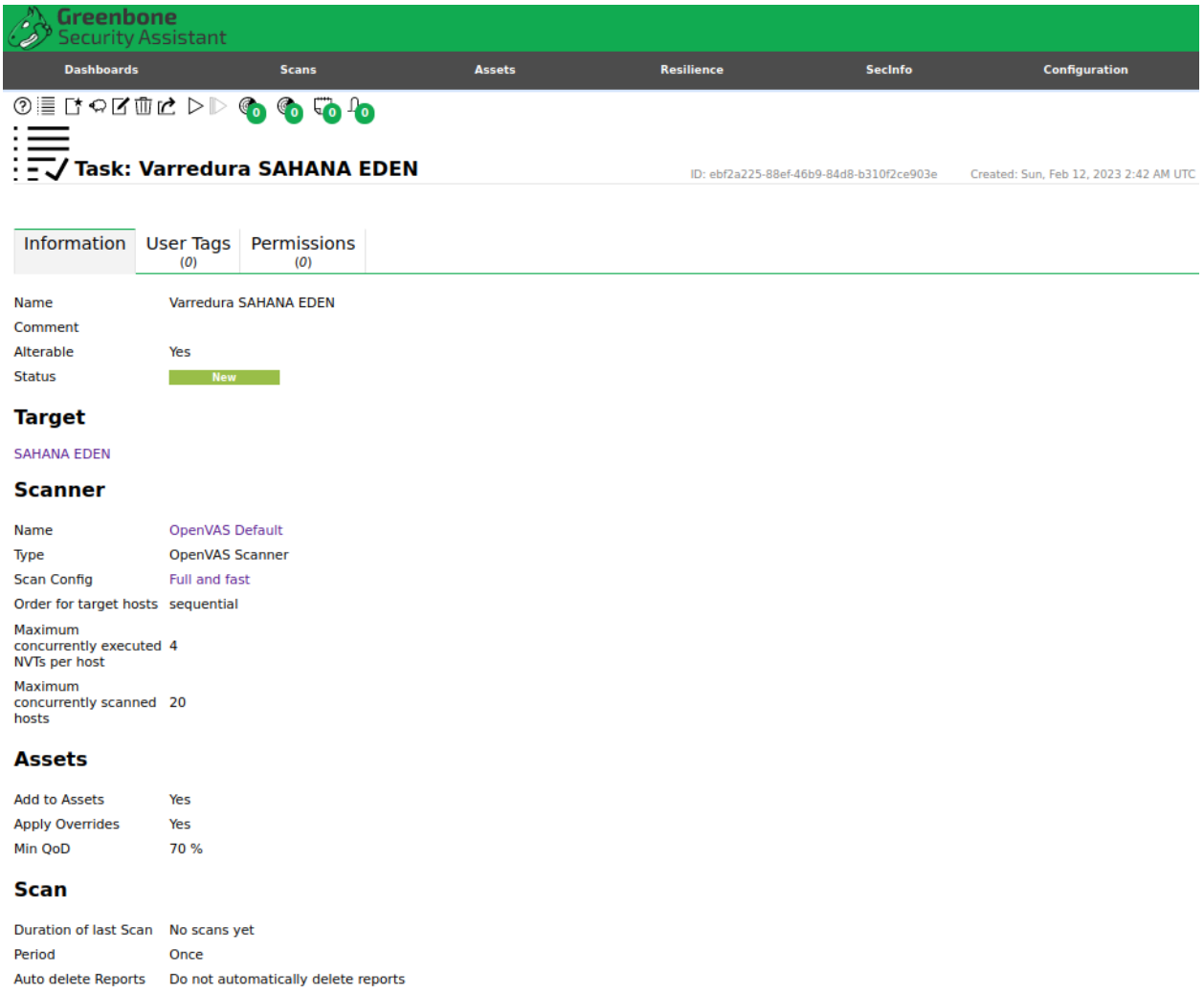

Figura 15 - Página de configuração da tarefa de varredura

Fonte: Greenbone OpenVAS

Execução e resultado da tarefa de varredura

Após realizada todas as configurações nos passos anteriores, é o momento de executar a varredura. No menu "*Scans->Tasks"* em seguida "*Actions"* e acionar o botão "*Start"*.

A varredura pelo Greenbone OpenVAS durou cerca de 1 hora como ilustra a figura 16. É possível identificar nas abas adicionais informações de todas as configurações, nós vamos focar na aba "*Results"* que onde é exibido os resultados dos VT's executados.

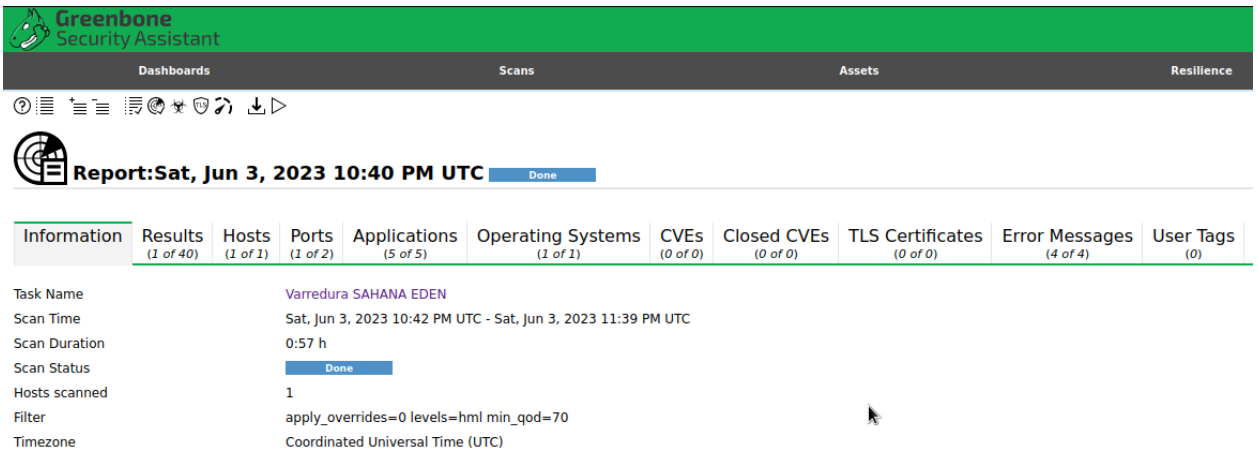

Figura 16 - Página do relatório da varredura do SAHANA EDEN

Fonte: Greenbone OpenVAS

Na aba “Results” (figura 16), podemos ver que temos 1 de 40 resultados, mas somente temos apenas 1 vulnerabilidade média sendo exibida em tela, isso se dá devido ao filtro padrão aplicado no greenbone, onde somente são exibidas vulnerabilidades com QoD (Quality of Detection) qualidade de detecção iguais ou acima de 70%.

A vulnerabilidade encontrada se refere ao uso do protocolo HTTP, protocolo este que por padrão não utiliza encriptação e trafega dados em texto claro, ou seja, a comunicação não passa por nenhuma camada de encriptação e pode ser interceptada, como podem ser vistos nas figuras 17 e 18.

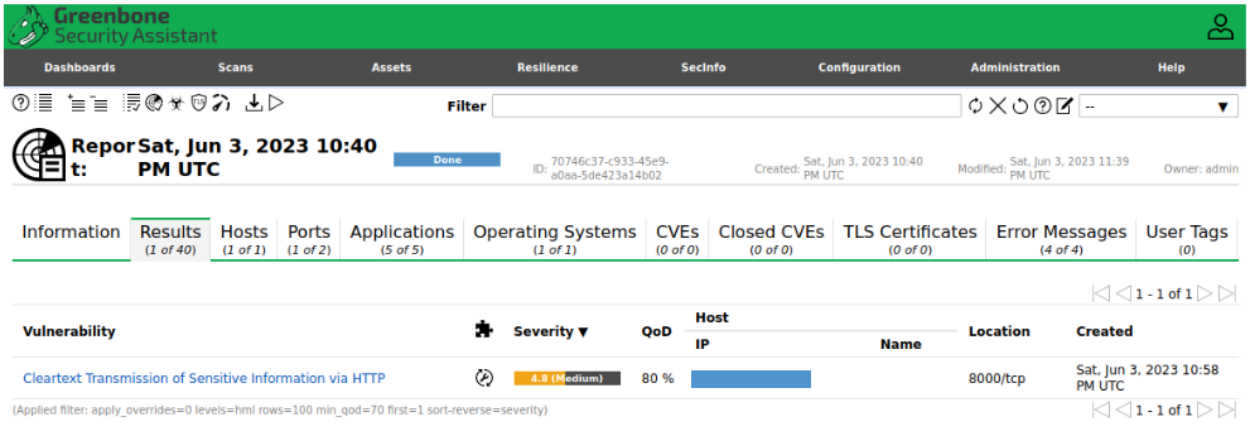

Figura 17 - Vulnerabilidade Transmissão Dados Sensíveis em Texto Claro

Fonte: Greenbone OpenVAS

Em “*Summary*”, um resumo informando que o dispositivo e a aplicação transmitem dados sensíveis como (*username*, *passwords*) em texto claro através do protocolo HTTP.

Em “*Detection Result*” contém as URLs que representam o resultado da detecção, exatamente duas URLs que trafegam esses tipos de dados sensíveis como senha por exemplo.

Em “*Detection Method*” explica com detalhes o VT usado para obter o resultado. Se observarmos o campo “*details*”, demonstra o ID no NVT utilizado.

Em “*Affected Software/OS*” explica que o sistema não força a encriptação para este tipo de comunicação sensível.

Em “*Impact*” mostra o impacto e o risco associado, que um atacante pode utilizar essa situação para comprometer ou escutar a comunicação entre o dispositivo e o sistema, usando técnicas de “*man-in-the-middle*”. Que é uma espécie de ataque onde um terceiro malicioso secretamente assume o controle do canal de comunicação entre dois ou mais pontos finais. O cenário comum envolve: dois pontos finais (vítimas) e um terceiro (agressor). O atacante tem acesso no canal de comunicação entre dois pontos finais, e pode manipular suas mensagens (CONTI; DRAGONI; LESYK, 2016).

Em “*Solution*”, mostra que existe um tipo de solução de contorno que pode ser aplicada, que é forçar a transmissão de dados por uma camada de criptografia, utilizando SSL/TLS e certificar que toda a comunicação seja redirecionada para o processo de encriptação antes mesmo de encaminhar dados sensíveis.

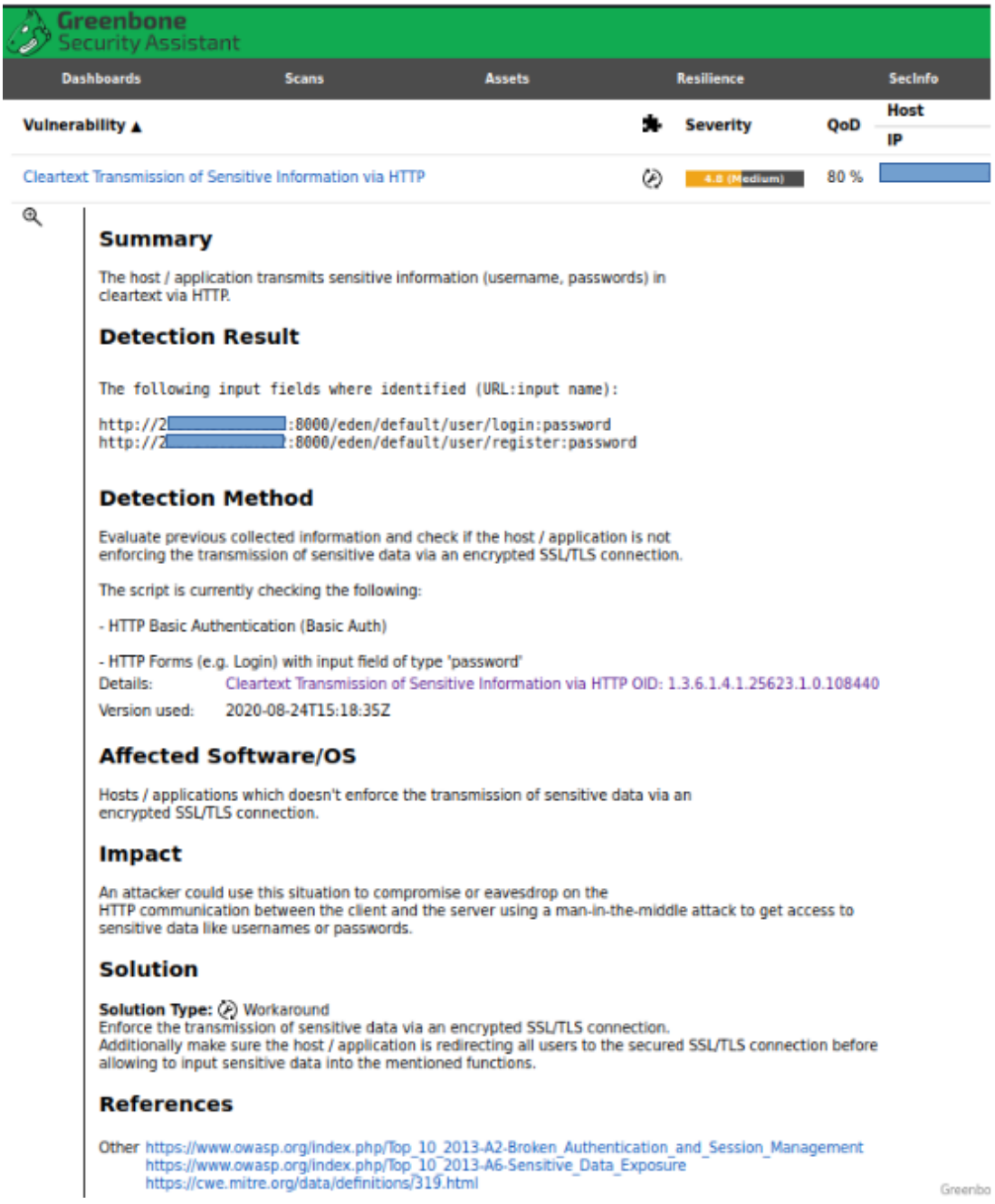

Figura 18 - Detalhes da Vuln. Transmissão Dados Sensíveis em Texto Claro

Fonte: Greenbone OpenVAS

Lembrando que se removermos o filtro aplicado que mostra apenas itens com QoD mínimo de 70%, iremos ver os demais resultados, que são exatamente a enumeração dos demais serviços, que podem ser bastante utilizados para o atacante obter mais informações sobre o ambiente. Como não é o foco dessa dissertação explicar todas as enumerações detectadas, iremos apenas mostrá-la no resumo na figura 19.

Podemos observar que tanto o Nmap quanto o Greenbone OpenVAS, conseguiram as mesmas informações de enumerações, como por exemplo a versão do SSH, seus algoritmos, versão do servidor Web, etc. informações essas sempre utilizadas para todo o processo de discovery e exploração.

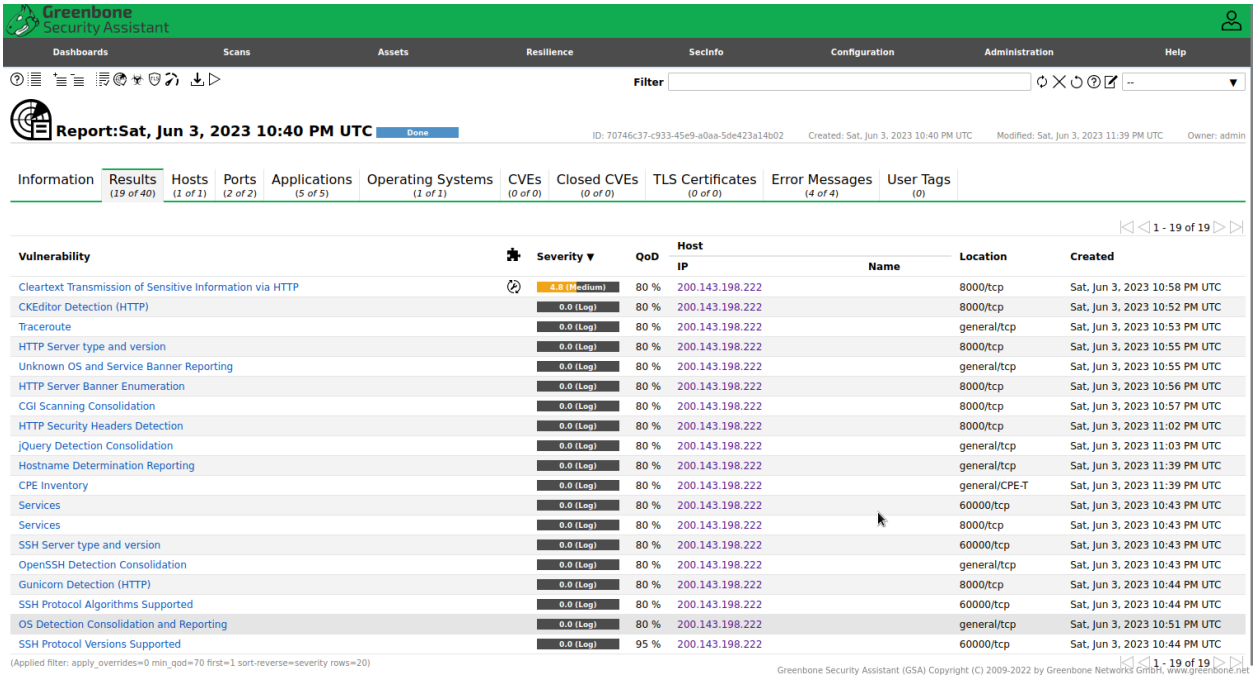

Figura 19 - Resultado da varredura sem o filtro de QoD aplicado

Fonte: Greenbone OpenVAS

A informação do NVT (figura 20), que demonstra a severidade da vulnerabilidade de acordo com o CVSS (Common Vulnerability Scoring System) V2, onde temos um exemplo da calculadora CVSS onde demonstra com clareza o cálculo relacionado a vulnerabilidade (figura 21), além de outros detalhes informados, como impacto, solução, mas principalmente para a QoD que foi considerada 80% neste caso, devido a ter encontrado um formulário dentro do SAHANA EDEN com a referência de “username” e “password” onde utiliza uma autenticação do tipo básica. De fato esse é o formulário da página de login de acesso ao SAHANA EDEN. Há chances da ferramenta falhar na detecção mesmo o QoD sendo alto, cabe o atacante sempre revalidar a informação.

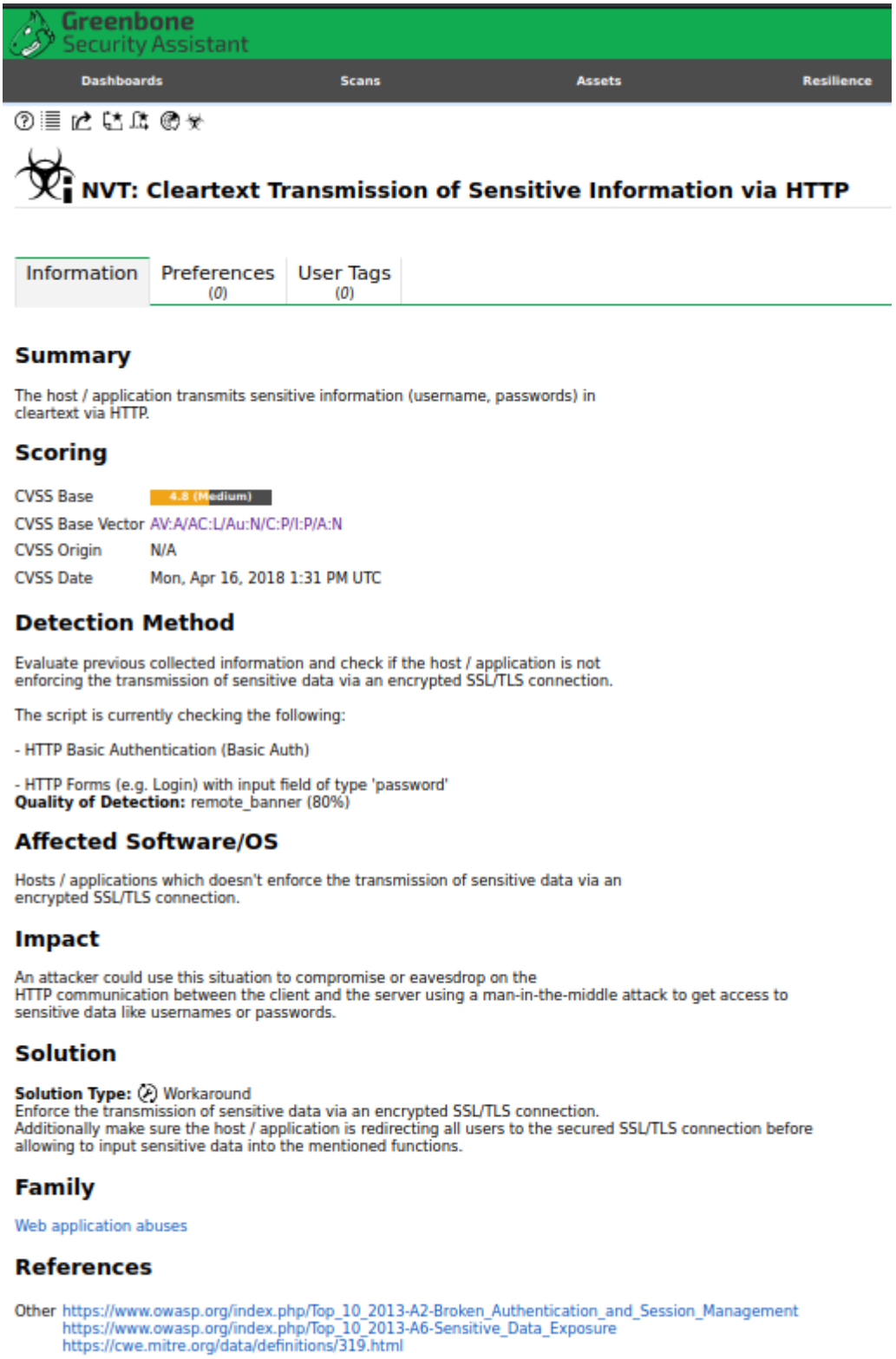

Figura 20 - NVT Transmissão Dados Sensíveis em Texto Claro

Fonte: Greenbone OpenVAS

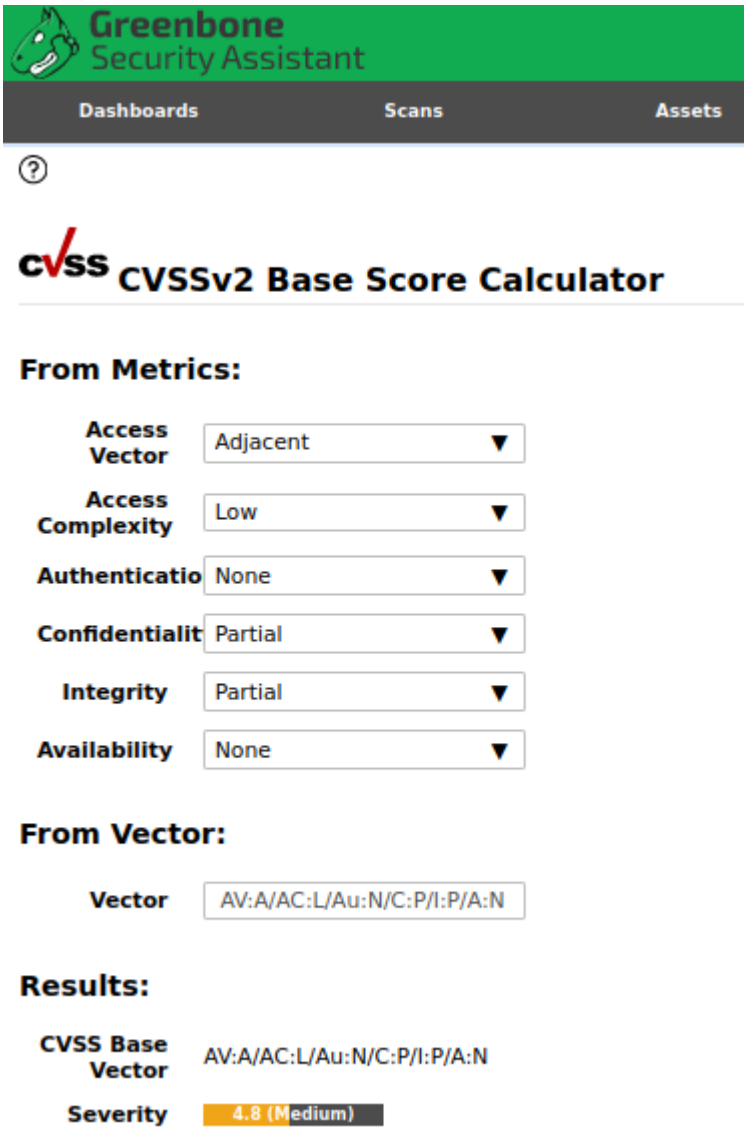

Figura 21 - Calculadora CVSS V2

Fonte: Greenbone OpenVAS

4.2.3 Wapiti

Foi executada a varredura utilizando o Wapiti, a versão no momento da execução era a 3.1.7 e a data de execução em 04/06/2023.

O Wapiti é um software standalone e não possui banco de dados, ou seja uma ferramenta simples por linha de comando similar ao Nmap, mas essa com foco exclusivo em aplicações Web.

A varredura durou cerca de 4 horas, identificaram 404 páginas entre URL's e formulários que foram rastreados no sistema. Foram 6 vulnerabilidade identificadas de acordo com a tabela 7. Somente serão explanadas nesta dissertação as categorias referentes às vulnerabilidades identificadas.

Content Security Policy Configuration

Content Security Policy (CSP) é uma camada adicional de segurança que ajuda a detectar e mitigar certos tipos de ataques, incluindo Cross Site Scripting (XSS) e ataques de *data injection*. O CSP é uma série de regras, é baseado no mecanismo de whitelist. O navegador irá limitar os recursos carregados pela página web de acordo com a whitelist do CSP, os recursos que não aparecem na whitelist de diretivas CSP não serão carregados na página da Web pelo navegador. Por exemplo, a diretiva CSP img-src https://google.com restringe páginas da Web para carregar apenas recursos de imagem no domínio google.com, e os recursos de imagem também devem usar o protocolo HTTPS. Imagens de qualquer outra fonte não serão exibidas na página web. O CSP é um mecanismo de defesa baseado no navegador, que geralmente é chamado de firewall no navegador. A lista de permissões do CSP é feita pelo site desenvolvedor e entra em vigor definindo a resposta HTTP/S do cabeçalho ou definir a meta tag (LV et al., 2023).

A vulnerabilidade encontrada na raiz do site, o CSP não está configurado, como mostra a figura 22.

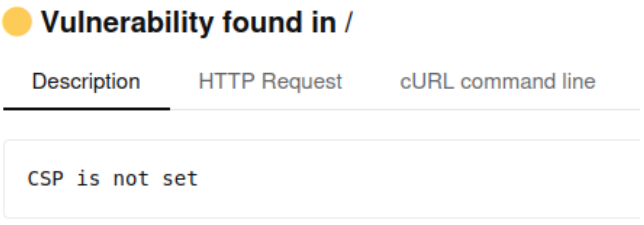

Figura 22 - Resultado da ausência de configuração de CSP

Fonte: Wapiti

A solução para configuração do CSP envolve adição do cabeçalho HTTP Content-Security-Policy na página WEB e especificar os valores para controlar qual recurso de agente de usuário é permitido para carregar a página. As referências podem ser encontradas em:

- https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP
- https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
- https://owasp.org/www-pdf-archive/2019-02-22_-_How_do_I_Content_Security_Policy_-_Print.pdf

HTTP Secure Headers

Cabeçalhos de segurança HTTP são um subconjunto de cabeçalhos HTTP padrão que mediam informações específicas de segurança entre o servidor e o cliente. O servidor Web envia as políticas de segurança necessárias dentro dos cabeçalhos de resposta para o navegador quando um usuário visita qualquer site. Ao receber a resposta, o navegador aplica essas regras de segurança que são políticas para determinado site no cliente. Alterar esses cabeçalhos em tempo real pode potencialmente desabilitar o mecanismo de defesa desejado, por exemplo, um controle de framing e expor um aplicativo aos tipos associados de ataques, por exemplo, *clickjacking* (AGARWAL, 2022).

Os cabeçalhos de segurança não foram identificados e que representam vulnerabilidades, os mesmos estão listados na tabela 6.

Tabela 6. Resultado das vulnerabilidades do cabeçalho HTTP usando o **Wapiti**

HTTP Secure Header	***Status***
X-Frame-Options	Not Set
X-XSS-Protection	Not Set
X-Content-Type-Options	Not Set
Strict-Transport-Security	Not Set

Fonte: Wapiti

X-Frame-Options

O cabeçalho *X-Frame-Options* protege contra ataques de *clickjacking* no lado do cliente. Ele permite que o servidor restrinja seu site para ser enquadrado dentro de outro site (por

exemplo, *iframe*) na mesma origem ou diferente. Por exemplo, pode-se desabilitar totalmente o enquadramento de seu site usando o atributo *DENY* ou permitir apenas o enquadramento das páginas de mesma origem usando o atributo *SAMEORIGIN* (AGARWAL, 2022).

O clickjacking é um ataque que esconde links por trás de botões, que tem como intenção redirecionar os usuários através de cliques e roubar informações sigilosas.

X-XSS-Protection

O cabeçalho de resposta HTTP X-XSS-Protection é uma funcionalidade do Internet Explorer, Chrome e Safari que impede páginas de carregarem quando eles detectam ataques de *scripting* entre sites (XSS) refletidos. Apesar destas proteções serem majoritariamente desnecessárias em navegadores modernos em sites utilizando uma forte *Content-Security-Policy* que desabilita o uso de JavaScript inline ('unsafe-inline'), eles ainda podem oferecer proteções para usuários de navegadores mais antigos que ainda não suportam CSP.

Os ataques Cross-Site Scripting (XSS) são um tipo de injeção, na qual scripts maliciosos são injetados em sites benignos e confiáveis. Os ataques XSS ocorrem quando um invasor usa um aplicativo da Web para enviar um código malicioso, geralmente na forma de um *script* do lado do navegador, para um usuário final diferente.

X-Content-Type-Options

Este cabeçalho HTTP visa impedir a vulnerabilidade de detecção do tipo MIME no lado do cliente, instruindo o navegador para não determinar o tipo MIME do conteúdo e obedecer aos valores especificados pelo cabeçalho *Content-Type*. Em qualquer momento se um usuário enviar qualquer arquivo para o qual o servidor envia sem cabeçalho *Content-Type* ou com valores inadequados, o navegador "fareja" o recurso para determinar seu tipo de conteúdo. Isso pode levar a situações perigosas, por exemplo, quando o cliente detecta o upload de arquivos como HTML. Ao definir o atributo nosniff para este cabeçalho, o servidor instrui o navegador a não farejar o tipo MIME de qualquer recurso. Por outro lado, descartar o cabeçalho deixa o aplicativo vulnerável a *sniffing* de conteúdo (AGARWAL, 2022).

Strict-Transport-Security

Cabeçalho de Segurança de Transporte Estrito. O cabeçalho HTTP Strict-Transport-Security (ou HSTS, para abreviar) permite que um servidor da web informe ao navegador que todas as conexões subsequentes para todas as solicitações devem ser estabelecidas exclusivamente através de HTTPS, nunca através de HTTP, utilizando um certificado válido.

Ajuda a prevenir vários ataques *man-in-the-middle* (MITM) que podem surgir em diferentes situações (LAVRENOVS; MELON, 2018).

A solução é usar as recomendações indicadas nas referências a seguir, para reforçar o cabeçalho HTTP.

- https://www.netsparker.com/blog/web-security/http-security-headers/
- https://www.keycdn.com/blog/http-security-headers
- https://owasp.org/www-chapter-ghana/assets/slides/HTTP_Header_Security.pdf

O site CSP Evaluator é uma excelente fonte para validar seus cabeçalhos, pode ser acessado pelo endereço https://csp-evaluator.withgoogle.com/.

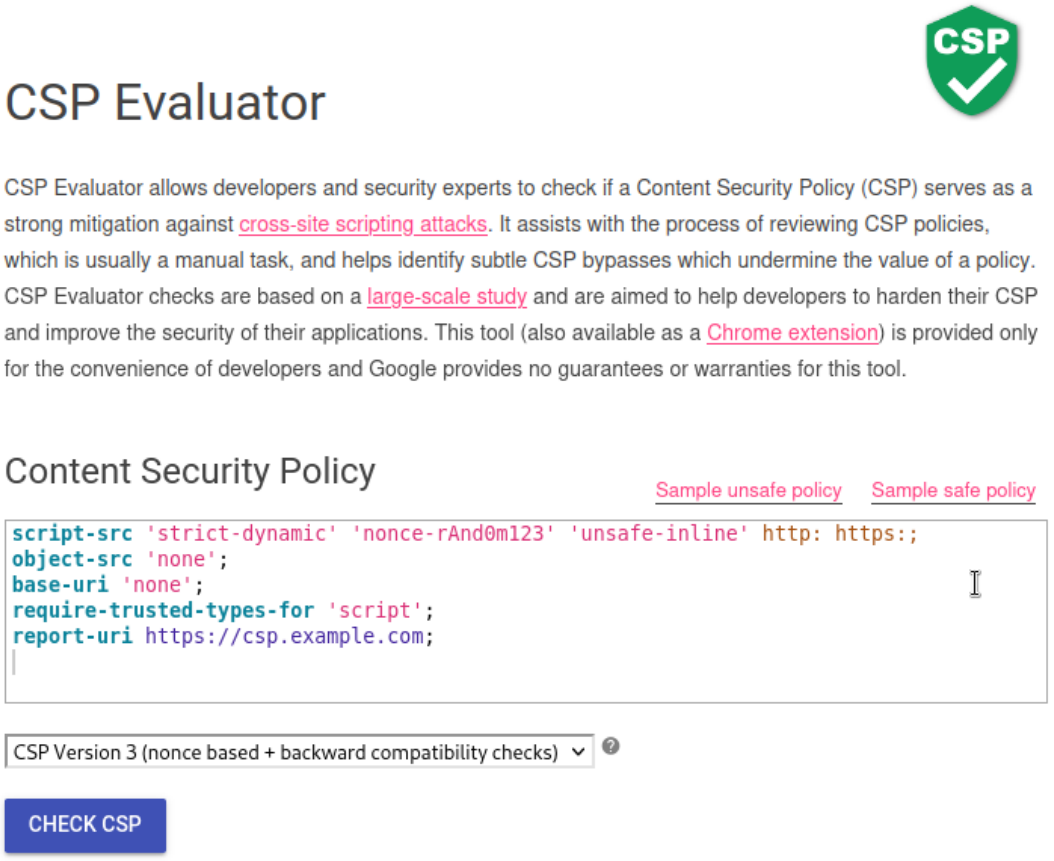

Figura 23 - Site CSP Evaluator

Fonte: https://csp-evaluator.withgoogle.com

Tabela 7 - Resultado resumido das vulnerabilidades usando o Wapiti

Category	***Number of vulnerabilities found***
Backup file	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0
Fingerprint web application framework	0
Fingerprint web server	0
Htaccess Bypass	0
HTTP Secure Headers	4
HttpOnly Flag cookie	0
Log4Shell	0
Open Redirect	0
Reflected Cross Site Scripting	0
Secure Flag cookie	1
SQL Injection	0
TLS/SSL misconfigurations	0
Server Side Request Forgery	0
Stored Cross Site Scripting	0
Subdomain takeover	0
Blind SQL Injection	0
XML External Entity	0
Internal Server Error	0
Resource consumption	0
Fingerprint web technology	0
HTTP Methods	0

Fonte: Wapiti

Secure Flag Cookie

A opção de *secure flag* pode ser configurada pelo servidor de aplicação enviando um novo cookie para o usuário com uma resposta HTTP. O propósito do secure flag é prevenir cookies de serem observados por partes não autorizadas devido a transmissão de um cookie enviado no formato de texto claro (não encriptado). A vulnerabilidade encontrada na raiz do site, *Secure flag* não está configurada, como mostra a figura 24.

Figura 24 - Resultado da ausência de configuração de Secure Flag
Fonte: Wapiti

A solução é quando gerar os cookies, tenha certeza que a configuração Secure Flag está habilitada, conforme referências abaixo:

- https://owasp.org/www-project-web-security-testing-guide/stable/4-Web_Application_Security_Testing/06-Session_Management_Testing/02-Testing_for_Cookies_Attributes.html
- https://owasp.org/www-community/controls/SecureCookieAttribute

4.2.4 OWASP ZAP

Foi executada a varredura utilizando o OWASP ZAP, a versão no momento da execução era a 2.12.0 e a data de execução em 03/06/2023.

Foram identificados 11 alertas conforme figura 25, 4 classificados com risco médio, 3 classificados com risco baixo e 4 apenas informativos, não há risco associado, mas são informações que podem favorecer o atacante a realizar movimentação lateral. Para estudo desta dissertação, vamos focar apenas nos riscos médios e baixos.

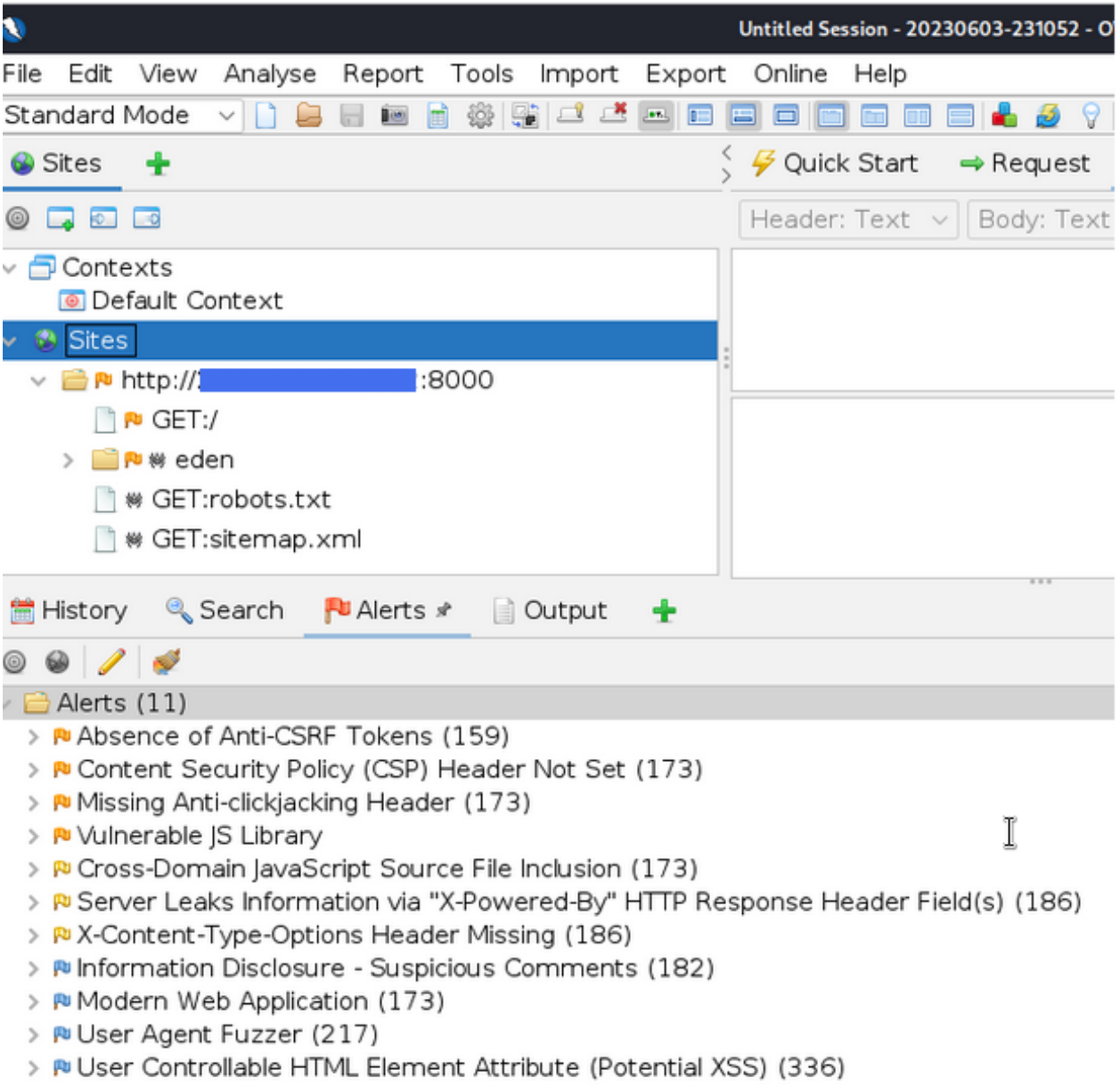

Figura 25 - Resultado da varredura utilizando o OWASP ZAP

Fonte: OWASP ZAP

Absence of Anti-CSRF Tokens

Nenhum token Anti-CSRF (*Cross-Site Request Forgery*) foi encontrado em um formulário de envio HTML. Uma falsificação de solicitação entre sites é um ataque que envolve forçar uma vítima a enviar uma solicitação HTTP para um destino de destino sem seu conhecimento ou intenção, a fim de executar uma ação como vítima. A causa subjacente é a funcionalidade do aplicativo usando ações previsíveis de URL/formulário de maneira repetível. A natureza do ataque é que o CSRF explora a confiança que um site tem para um usuário. Por outro lado, o *cross-site scripting* (XSS) explora a confiança que um usuário tem em um site.

Como o XSS, os ataques CSRF não são necessariamente *cross-site*, mas podem ser. A falsificação de solicitação entre sites também é conhecida como CSRF, XSRF, *one-click attack*, *session riding, confused deputy,* e *sea surf*.

Os ataques CSRF são eficazes em várias situações, incluindo:

- A vítima tem uma sessão ativa no site de destino.
- A vítima é autenticada via autenticação HTTP no site de destino.
- A vítima está na mesma rede local do site de destino.

CSRF é um conhecido ataque da Web que força um usuário a enviar solicitações HTTP indesejadas e controladas pelo invasor para um aplicativo da Web vulnerável no qual ele está autenticado no momento. O conceito crítico do CSRF é que as solicitações maliciosas são roteadas para o aplicativo da web por meio do navegador do usuário; portanto, eles podem ser indistinguíveis das solicitações benignas pretendidas autorizadas pelo usuário. Para evitar o CSRF, os desenvolvedores da Web devem implementar mecanismos de proteção explícitos. Se adicionar interação extra com o usuário não afetar muito a usabilidade, é possível forçar a reautenticação ou usar senhas de uso único/CAPTCHAs para evitar que as solicitações entre sites passem despercebidas (THAQI; VISHI; REXHA, 2022).

O CSRF tem sido usado principalmente para executar uma ação contra um site de destino usando os privilégios da vítima, mas técnicas recentes foram descobertas para divulgar informações obtendo acesso à resposta. O risco de divulgação de informações aumenta drasticamente quando o site de destino é vulnerável ao XSS, porque o XSS pode ser usado como uma plataforma para CSRF, permitindo que o ataque opere dentro dos limites da política de mesma origem.

A solução deve ser adotada tanto nas fases de Arquitetura e Design e de implementação.

Na fase de Arquitetura e Design, use uma biblioteca ou estrutura verificada que não permita que essa fraqueza ocorra ou forneça construções que tornem essa fraqueza mais fácil de evitar. Por exemplo, use pacotes anti-CSRF como o OWASP CSRFGuard.

Na fase de Implementação, certifique-se de que seu aplicativo esteja livre de problemas de *script* entre sites, porque a maioria das defesas CSRF pode ser contornada usando *script* controlado pelo invasor.

Na fase de Arquitetura e Design, gere um “nonce” exclusivo para cada formulário, coloque o “nonce” no formulário e verifique o “nonce” ao receber o formulário. Certifique-se de

que o “nonce” não seja previsível (CWE-330). Observe que isso pode ser ignorado usando XSS. Identificar operações especialmente perigosas. Quando o usuário executar uma operação perigosa, envie uma solicitação de confirmação separada para garantir que o usuário pretenda realizar essa operação. Observe que isso pode ser ignorado usando XSS. Use o controle de gerenciamento de sessão ESAPI. Este controle inclui um componente para CSRF. Não use o método GET para nenhuma solicitação que acione uma mudança de estado.

Na fase de Implementação, verifique o cabeçalho HTTP *Referer* para ver se a solicitação se originou de uma página esperada. Isso pode interromper a funcionalidade legítima, porque os usuários ou proxies podem ter desativado o envio do *Referer* por motivos de privacidade.

Content Security Policy (CSP) Header Not Set

Foi o mesmo item identificado no Wapiti, ausência de cabeçalhos CSP.

Missing Anti-clickjacking Header

Foi o mesmo item identificado no Wapiti, ausência do cabeçalho X-Frame-Options.

Vulnerable JS Library

A biblioteca identificada jquery-validation, versão 1.19.1 é vulnerável.

A solução que deve ser adotada é atualizar para a versão mais recente do *jquery-validation.*

Cross-Domain JavaScript Source File Inclusion

A página inclui um ou mais arquivos de script de um domínio de terceiros. Foi identificado na página scripts java como origem domínio do googleapis.com. Como isso não representa uma vulnerabilidade, inclusive é uma prática comum, utilizar scripts de fontes confiáveis, nesse caso, a solução é composta por aplicar um CSP (*Content Security Policy*) *Header* para uma *whitelist* de domínios de fontes confiáveis para scripts, inclusive para este domínio, onde será explanado mais adiante.

Server Leaks Information via “X-Powered-By” HTTP Response Header Field

O servidor da web/aplicativos está vazando informações por meio de um ou mais cabeçalhos de resposta HTTP "X-Powered-By". O acesso a essas informações pode facilitar aos invasores a identificação de outras estruturas/componentes dos quais seu aplicativo da Web depende e as vulnerabilidades às quais esses componentes podem estar sujeitos.

A solução que deve ser adotada é certificar de que seu servidor web, servidor de aplicativos, balanceador de carga, etc. esteja configurado para suprimir cabeçalhos "X-Powered-By".

X-Content-Type-Options Header Missing

Foi o mesmo item identificado no Wapiti, ausência do cabeçalho X-Content-Type-Options.

A tabela 8 demonstra o resultado consolidado da varredura. O resultado foi bem variável pois a estratégia da escolha de softwares de varredura foi bem específica, onde escolhemos ferramentas de descoberta e varredura de rede e ferramentas de varredura de aplicações Web, com isso foram descobertas no total de 11 fragilidades, totalizando 6 com risco médio, 5 com o risco baixo.

Na próxima etapa, ainda dentro da fase de execução, vamos explorar e mitigar as vulnerabilidades que são passíveis disto. Será apresentado o resultado da exploração e mitigação.

Ao final da exploração e mitigação de todas as possibilidades, será efetuado novamente a varredura comprovando a mitigação das vulnerabilidades.

Tabela 8 - Resultado consolidado da execução da varredura

		Vulnerabilidades Identificadas x Ferramentas										
		Server Leaks Information via "X-Powered-By" HTTP Response Header Field	Transmissão Dados Sensíveis em Texto Claro	CSP Não Configurado	Cabeçalho X-Frame Options Não configurado	Cabeçalho X-XSS-Protection Não configurado	Cabeçalho X-Content-Type -Options Não configurado	Cabeçalho Strict-Transport -Security Não configurado	Secure Flag Cookie Não configurado	Biblioteca Java Script Vulnerável	Cross-Domain Java Script Source File Inclusion	Ausência de tokens Anti-CSRF
Ferramentas	Nmap	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não
	Greenbone OpenVAS	Não	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Não
	Wapiti	Não	Não	Sim	Sim	Sim	Sim	Sim	Sim	Não	Não	Não
	OWASP ZAP	Sim	Não	Sim	Sim	Não	Sim	Não	Não	Sim	Sim	Sim

Fonte: Autor

4.3 Exploração

Nesta etapa iremos explorar as vulnerabilidades de forma a mostrar a fragilidade da configuração que a torna passível de execução. Iremos mostrar a exploração da vulnerabilidade identificada pelo Greenbone OpenVAS “Transmissão Dados Sensíveis em Texto Claro”.

O protocolo HTTP por padrão não realiza a encriptação de dados, ele precisa estar configurado para usar a encriptação utilizando TLS (*Transport Layer Security*) Camada de Segurança Transporte, é possivelmente o protocolo mais usado de segurança; é amplamente implantado para proteger o tráfego da web (HTTPS).

TLS é um conjunto de protocolos configurados dinamicamente, controlado por uma máquina de estado interna que utiliza uma grande coleção de algoritmos criptográficos. Isso gera grande flexibilidade para conectar clientes e servidores, potencialmente ao custo da segurança, portanto, o aplicativo TLS deve configurar e revisar cuidadosamente suas negociações e conexões antes de prosseguir. Assim, provamos a segurança em relação à escolha da versão do protocolo, conjunto de cifras e certificados das duas partes (BHARGAVAN et al., 2013).

Inicialmente a aplicação do SAHANA EDEN foi configurada apenas com o protocolo HTTP sem a encriptação, ou seja, sem o uso do TLS o que tornou a vulnerabilidade possível de ser explorada.

Na figura 26, podemos identificar que o próprio navegador web já alerta na parte superior que o site em questão possui uma “conexão não segura” mostrando um ícone de um cadeado cortado ou seja qualquer dado ali trafegado não vai ser encriptado e vai trafegar em texto claro, que pode ser facilmente capturado e utilizado.

Utilizaremos um *sniffer* de rede para capturar os dados de login e senha para demonstrar a vulnerabilidade. O *sniffer* de rede é um método para usar a interface local de rede, fazendo com que o computador possa detectar a transmissão de dados dele próprio e em algumas vezes até de outros computadores no mesmo segmento de rede. Com a tecnologia Ethernet, por exemplo, se o adaptador é definido em modo promíscuo, ele pode receber a transmissão de dados em todo o segmento, e enviá-lo para o operador sistema para tratamento adicional (BO YU, 2010).

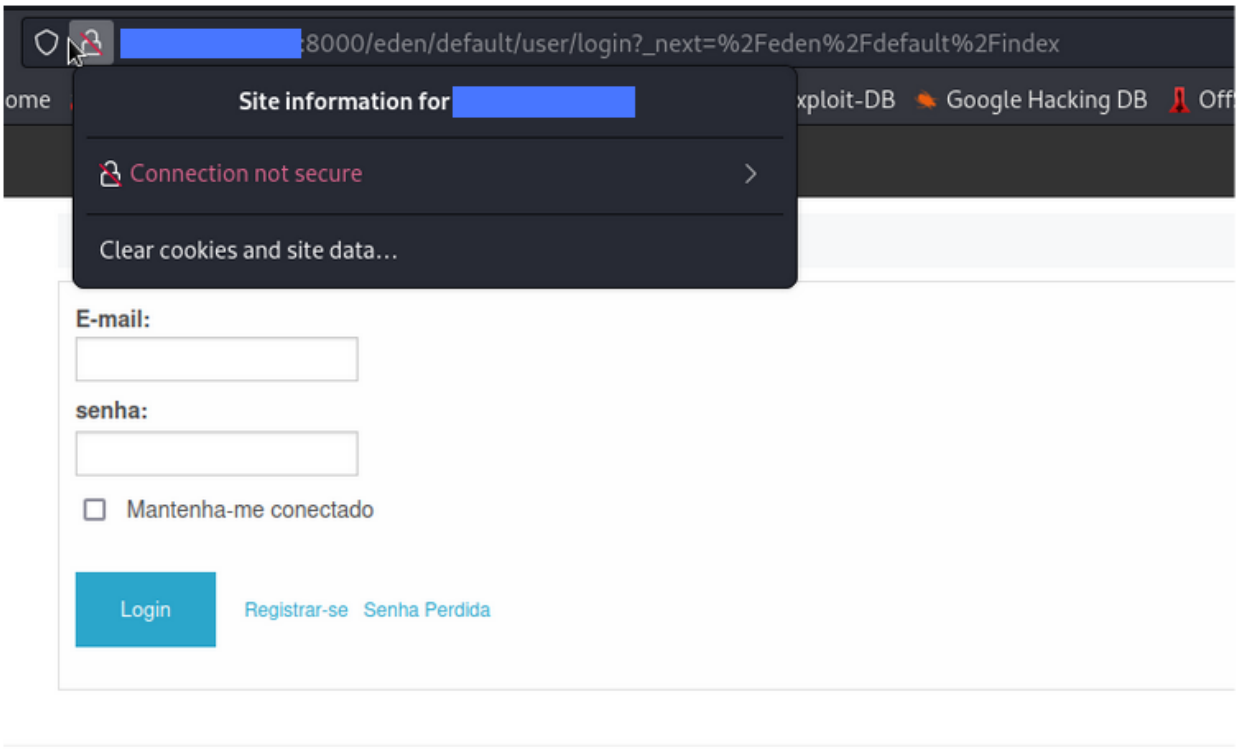

Figura 26 - Página de login do SAHANA EDEN sem encriptação

Fonte: Autor

Simulamos a execução do tcpdump, um utilitário do próprio Linux para utilizar a interface de rede local e habilitá-la no modo promíscuo de forma que ela possa capturar todos os dados trafegados. Utilizamos o comando **"tcpdump -i ens18 port 8000 -s0 -vv -X"** onde o parâmetro **"-i"** é a interface de rede onde passará o tráfego que será coletado, nesse caso é **"ens18"**, parâmetro "port" qual porta será escutada, nesse caso a **"8000"** significa um filtro onde o *sniffer* coletará os dados, se suprimirmos essa informação seriam coletados todos os tráfegos de dados, o que geraria um volume desnecessário para nosso experimento. O parâmetro **"-s"** significa o tamanho do frame a ser coletado o valor **"0"** significa usar o comprimento necessário para capturar os pacotes inteiros, o parâmetro **"-vv"** significa *verbose mode*, a ferramenta irá mostrar os resultados da captura em tela e por fim o parâmetro **"-X"** mostra em tela o pacote em hexadecimal e em ASCII, no formato necessário para a leitura dos dados por nós humanos, conforme pode ser visto nas figuras 27 e 28.

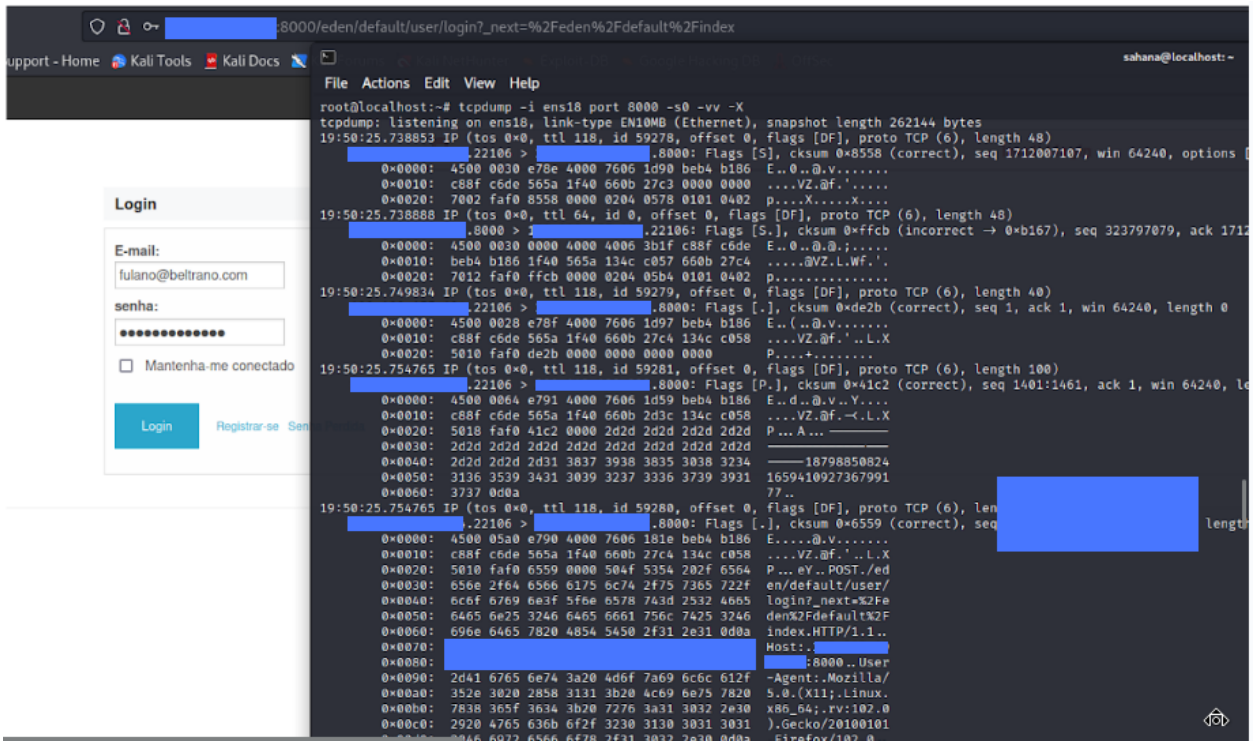

Figura 27 - Interface de rede em modo promíscuo usando o tcpdump

Fonte: Autor

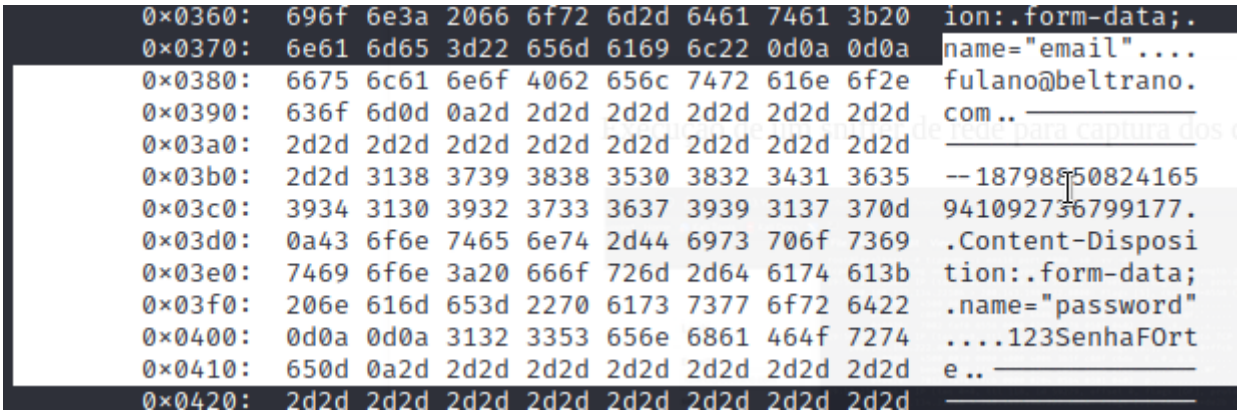

Figura 28 - Dados interceptados de usuário e senha sem encriptação

Fonte: Autor

Na figura 28, podemos observar os dados capturados do protocolo HTML trafegados na porta TCP 8000 usando o protocolo HTTP sem encriptação, os dados em texto claro como detectado pelo Greenbone OpenVAS. Podemos observar dados de login e senha capturados, "e-mail: fulano@beltrano.com" e "senha: 123SenhaFOrte". O que demonstra de fato a fragilidade do protocolo sem a encriptação aplicada.

4.4 Etapa pós execução

Na fase de pós-execução do teste de penetração, o foco se volta para uma análise aprofundada das vulnerabilidades detectadas durante a exploração. É nesse estágio que se busca identificar e priorizar as ações necessárias para mitigar as vulnerabilidades, visando corrigir as falhas de segurança e fortalecer a resistência do sistema. Essas ações não apenas almejam eliminar as brechas de segurança identificadas, mas também estabelecer uma base sólida para um protocolo de monitoramento contínuo, assegurando que o ambiente permaneça protegido contra ameaças emergentes.

Nesse contexto, é essencial estabelecer procedimentos de rotina para o monitoramento constante do servidor e do software do sistema SAHANA EDEN. Ao identificar os procedimentos adequados para o monitoramento contínuo, é possível detectar novas vulnerabilidades, anomalias de tráfego e atividades suspeitas de maneira proativa. A combinação desses dois aspectos na fase de pós-execução visa não apenas corrigir o que foi encontrado, mas também criar uma base sólida para a segurança contínua, em linha com as melhores práticas do NIST SP 800-155 e as recomendações do OWASP *Top Ten*.

4.4.1 Mitigação e Reteste

Nesta etapa iremos realizar a mitigação das vulnerabilidades que forem mais factíveis de serem realizadas com o conhecimento do autor, algumas delas envolvem um conhecimento mais aprofundado em linguagem de programação e para estas serão apenas sugeridas recomendações na etapa de pós-execução.

Para cada mitigação iremos realizar um reteste para ter a certeza que aquela vulnerabilidade foi de fato mitigada.

Vulnerabilidade: Server Leaks Information via “X-Powered-By” HTTP Response Header Field

Informações adicionais como nome da plataforma é exibido no cabeçalho, “X-Powered-By: web2py”, como pode ser visto na figura 29. Essa informação pode fazer com que o invasor descubra facilmente mais informações do ambiente para planejar um ataque mais preciso.

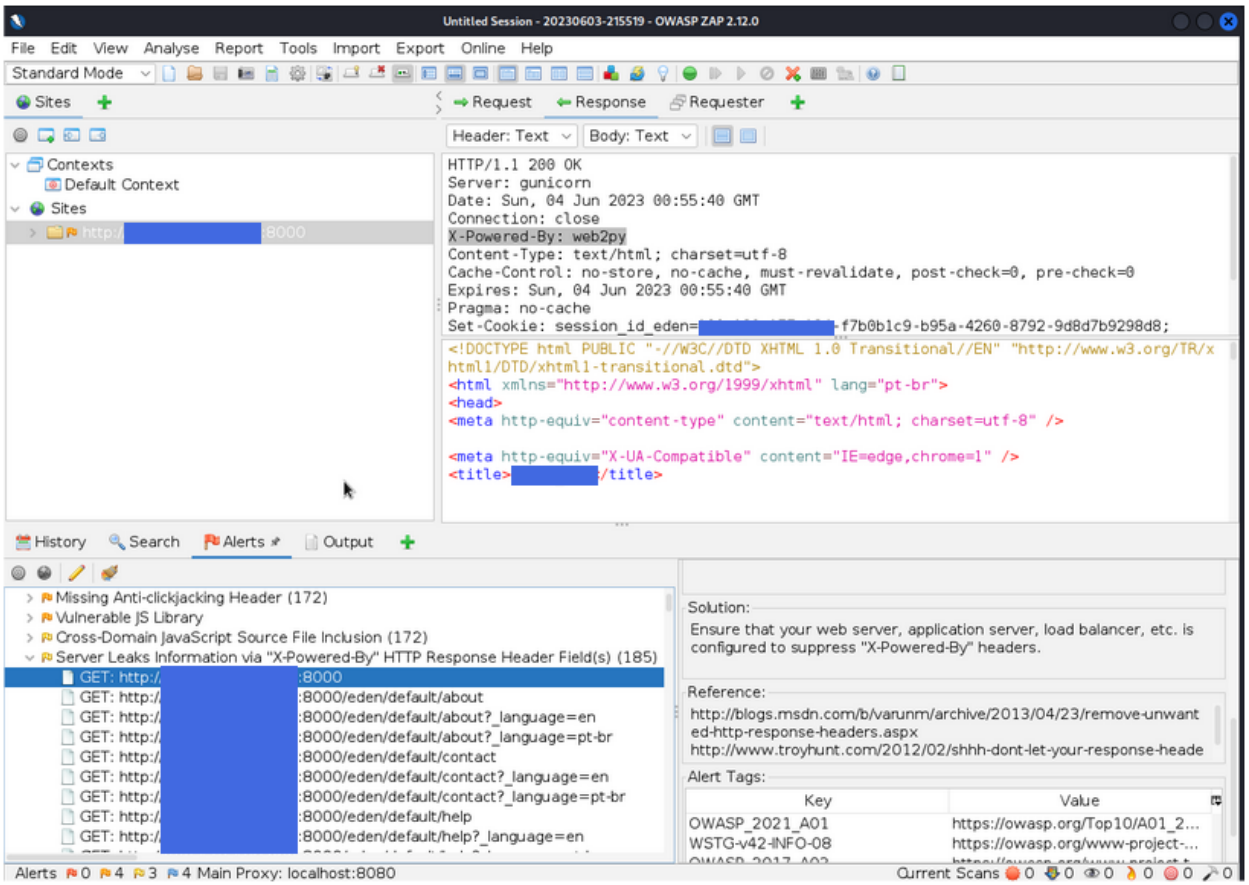

Figura 29 - Resultado OWASP ZAP mostrando o Header X-Powered-By

Fonte: Autor

Solução: Suprimir a informação de X-Powered_By

Mitigação:

Na pasta "/home/sahana/web2py/gluon" editar o arquivo "globals.py" procurar por "X-Powered" altere a variável e remova web2py e deixe em branco, conforme pode ser visto na figura 30.

Figura 30 - Arquivo de configuração web2py gluon - globals.py

Fonte: Autor

Vulnerabilidade: Transmissão de Informações Sensíveis em sem encriptação por HTTP

Solução:

Imponha a transmissão de dados confidenciais por meio de uma conexão SSL/TLS criptografada. Além disso, verifique se o host/aplicativo está redirecionando todos os usuários para a conexão segura SSL/TLS antes permitindo inserir dados sensíveis nas funções mencionadas.

Mitigação:

Ação 1:

Criação de um certificado digital autoassinado para utilização como forma de encriptação.

Pré-requisitos:

Acesso a interface shell do Linux o instalação do OpenSSL e permissões administrativas.

1) Conexão a shell do linux do servidor Sahana e como root acessar a pasta "/home/sahana/web2py"

2) Uso do comando "openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes -keyout key.pem -out cert.crt -extensions san -config <(echo "[req]";

echo distinguished_name=req;

echo "[san]";

echo subjectAltName=IP:200.xxx.xxx.xxx

) -subj "/CN=200.xxx.xxx.xxx"", conforme figura 31.

É possível criar um certificado através de uma entidade certificadora pública, mas pra isso teríamos custo e não é o objetivo desta demonstração.

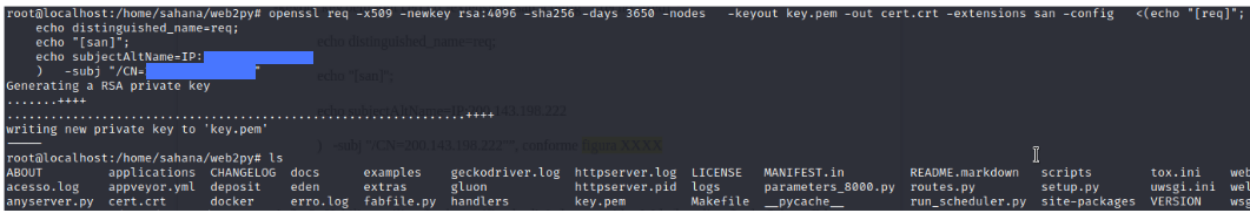

Figura 31 - Criação do certificado auto assinado

Fonte: Autor

Ação 2:

Habilitar utilização de certificado digital no servidor Web da aplicação do Sahana Eden e usar o certificado criado anteriormente.

"gunicorn --certfile=cert.crt --keyfile=key.pem --ssl-version=TLSv1_2 --workers 2 --access-logfile acesso.log --error-logfile erro.log wsgihandler:application -b 0.0.0.0:8000 &"

Após a execução do comando acima, é possível já ver que o certificado digital já está funcionando. Como trata-se de um certificado auto assinado, onde a própria fonte autoritativa geradora dos certificados, gera um certificado para si próprio sem uma requisição. Com isso o navegador faz um alarme de segurança, como mostrado na figura 32. Nesse caso não há riscos de

segurança, pois sabemos que nós mesmos criamos o certificado, mas o recomendado é utilizar uma entidade certificadora pública para esse tipo de situação.

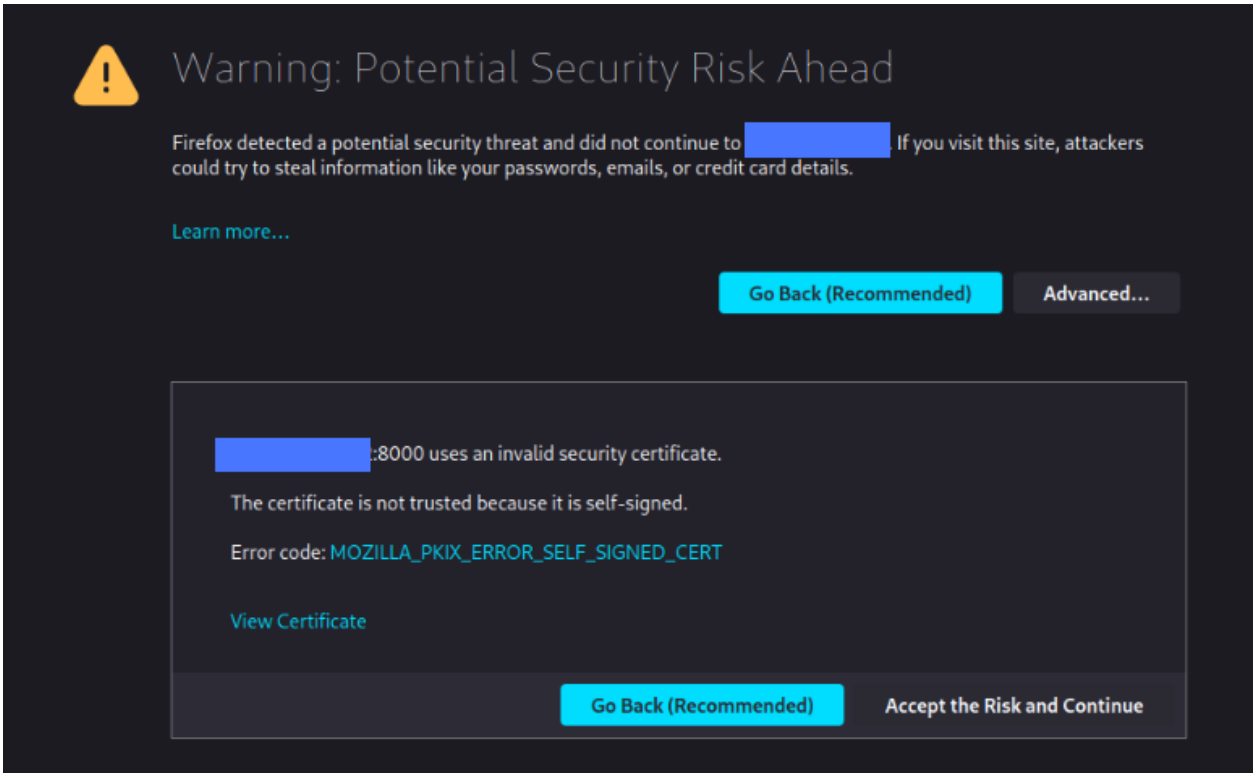

Figura 32 - Alerta do navegador para certificado auto assinado

Fonte: Autor

Ao clicar no botão "*advanced*" é exibido informações de segurança sobre o certificado, como por exemplo versão do protocolo TLS, atualmente sendo o mais utilizado o 1.2, versões anteriores do TLS já estão obsoletas e vulneráveis, como a versão 1.0 e .1.1. O tipo de encriptação, tamanho da chave e algoritmo de criptografia, conforme figura 33. Se clicarmos no botão "*View Certificate*" teremos mais informações sobre o certificado, como data de emissão, expiração, protocolo de encriptação e assinatura, entre outros atributos, como pode ser visto na figura 34.

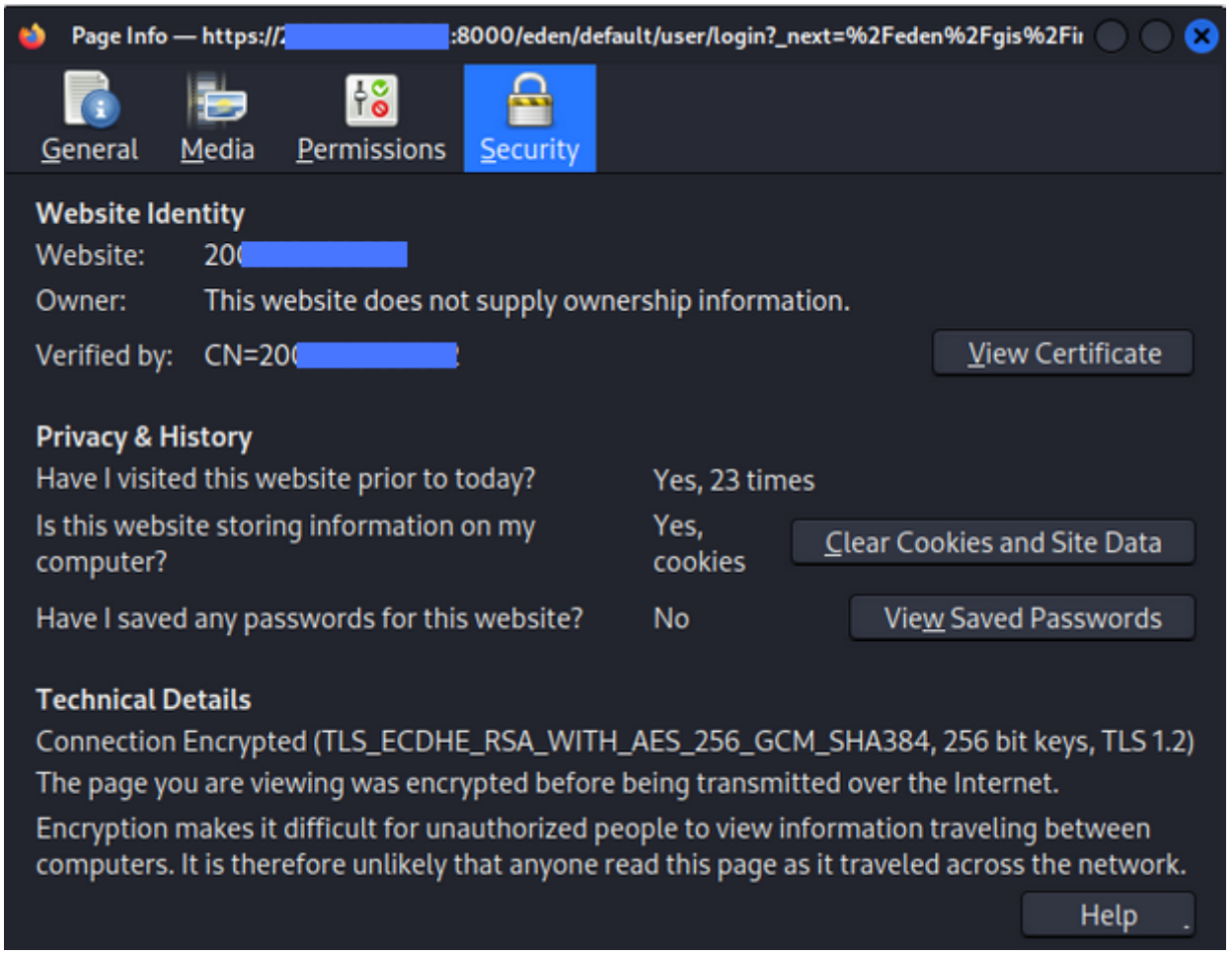

Figura 33 - Informações de segurança do site, com o certificado

Fonte: Autor

Figura 34 - Informações do certificado digital

Fonte: Autor

Após a implementação do certificado digital, será novamente realizada uma demonstração da exploração que fora realizada anteriormente sem a utilização do certificado digital, onde foi possível ler dados sensíveis como usuário e senha. O processo utilizado é o mesmo apresentado nas páginas 82 e 83, onde executamos o tcpdump com os parâmetros para colocar a interface de rede em modo promíscuo e realizar a captura dos pacotes para a tentativa de ler o seu conteúdo.

Como pode-se observar na figura 35, simulamos a tentativa de captura dos dados utilizando o mesmo usuario e senha utilizados anteriormente, mas na coluna de exibição dos

dados, já podemos ver que não há nada legível para nosso entendimento, ou seja a informação está encriptada e não pode ser lida, conforme figura 36.

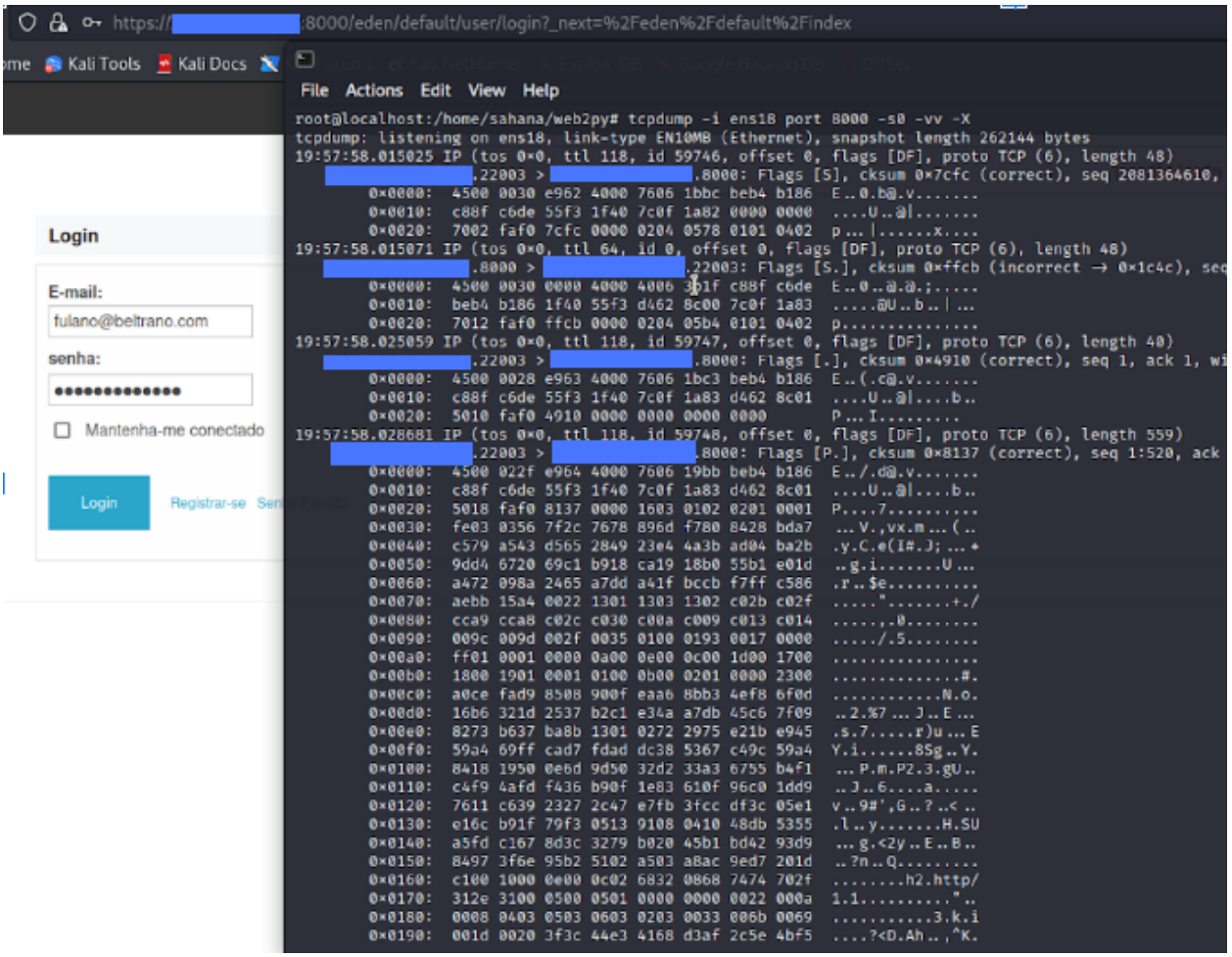

Figura 35 - Tentativa de interceptação de dados com encriptação aplicada

Fonte: Autor

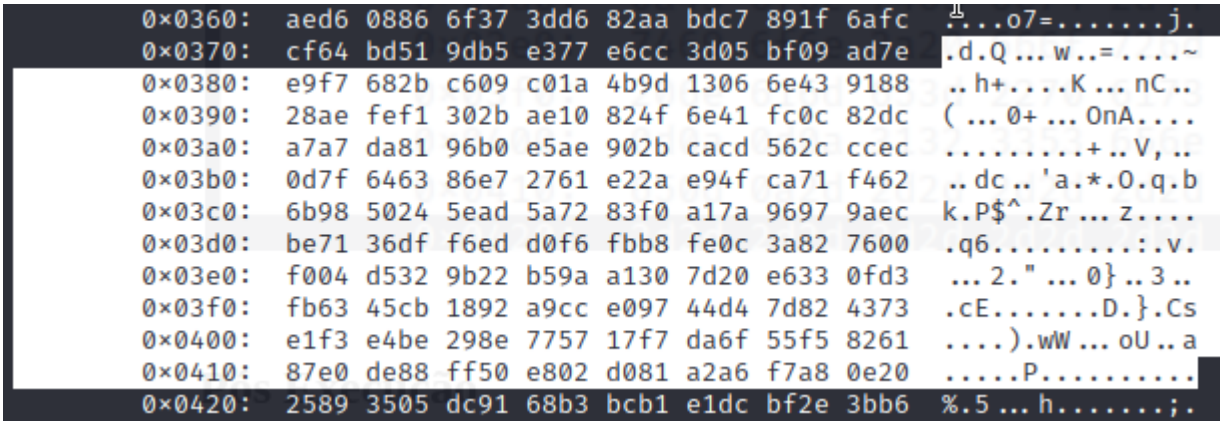

Figura 36 - Amostra dos dados encriptados durante a captura de rede

Fonte: Autor

Ação 3: Revalidar solução aplicada.

Após aplicação da solução da vulnerabilidade, foi realizada novamente a execução da tarefa de varredura com o greenbone e tivemos outras 2 vulnerabilidades, essas já oriundas da nova configuração da criptografia SSL/TLS habilitada. Conforme pode ser visto na figura 37.

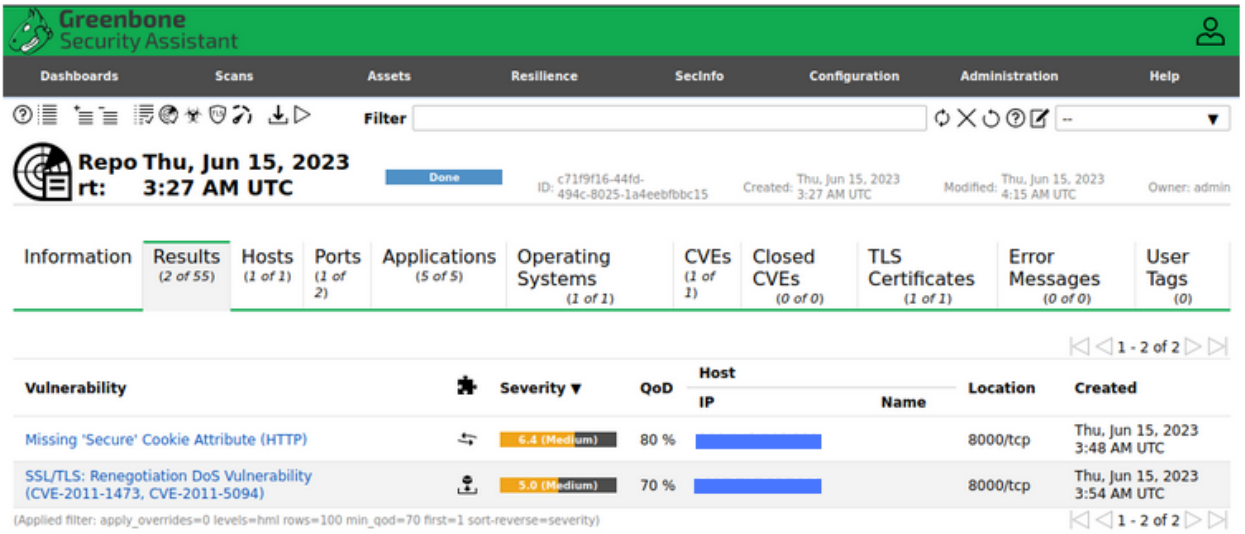

Figura 37 - Varredura com o Greenbone após a configuração de encriptação

Fonte: Greenbone OpenVAS

Uma delas já havia sido identificada pelo Wapiti que é o cookie de sessão com ausência de encriptação, conforme figura 38, mas para a solução teríamos como pré-requisito o certificado digital que foi configurado anteriormente.

Pode-se verificar que de fato mitigamos a vulnerabilidade "*Server Leaks Information via "X-Powered-By" HTTP Response Header Field*", mas apareceu uma nova vulnerabilidade relacionada ao TLS, configuração que foi realizada.

A vulnerabilidade "SSL/TLS Renegotiation DoS Vulnerability" permite que o atacante force uma renegociação da conexão SSL/TLS de uma conexão já estabelecida, realizando isso de forma massiva e automatizada o atacante pode tornar o serviço indisponível, que é o conceito de negação de serviço ou "Denial of Service". Como pode ser visto no detalhe da vulnerabilidade apresentada no relatório, conforme figura 39.

Figura 38 - Vulnerabilidade *Secure Flag* não configurado para *cookies*

Fonte: Greenbone OpenVAS

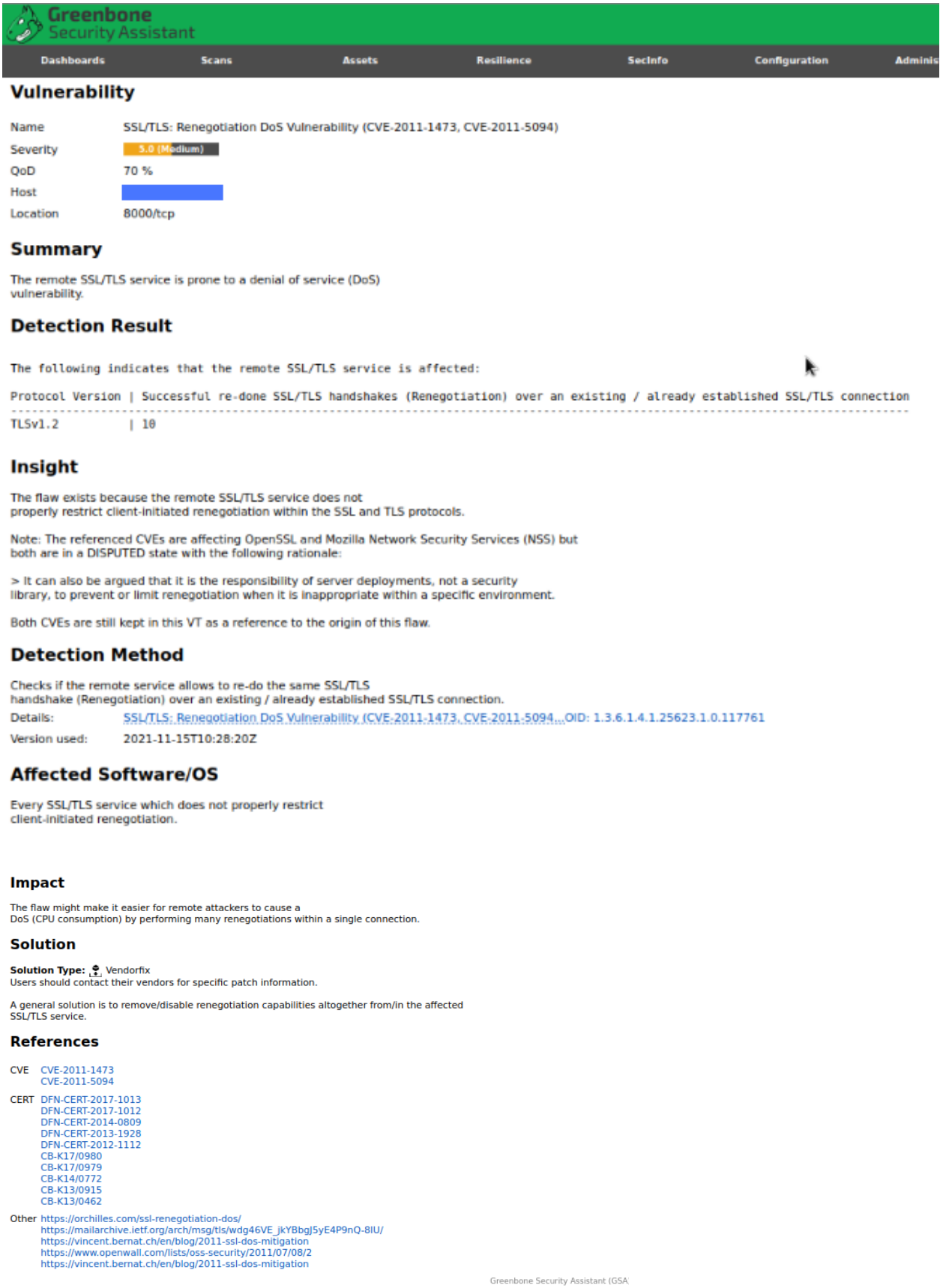

Figura 39 - Vulnerabilidade Negação de Serviço na renegociação SSL/TLS

Fonte: Greenbone OpenVAS

Vulnerabilidade: Missing ‘Secure’ Cookie Attribute (HTTP)

Com o mesmo descritivo da vulnerabilidade identificada pela ferramenta Wapiti e após termos habilitado a comunicação por TLS, é necessário aplicar a proteção de encriptação nos cookies de sessão para evitar ataques do tipo *hijacking* (sequestro) de sessão, onde o atacante com o *cookie* capturado, poderá utilizar em uma outra sessão se passando como se fosse a vítima.

Solução: Configurar a aplicação para habilitar a segurança no cookie de sessão

Mitigação:

No diretório da aplicação do Sahana Eden do Web2py, nesta ambiente estava configurado em “/home/sahana/web2py/eden/” no diretório “controllers”, no arquivo “default.py adicionado o parâmetro “response.cookies['session_id_eden']['secure'] = True” de proteção do cookie, forçando o mesmo a somente usar encriptação SSL, conforme figura 40.

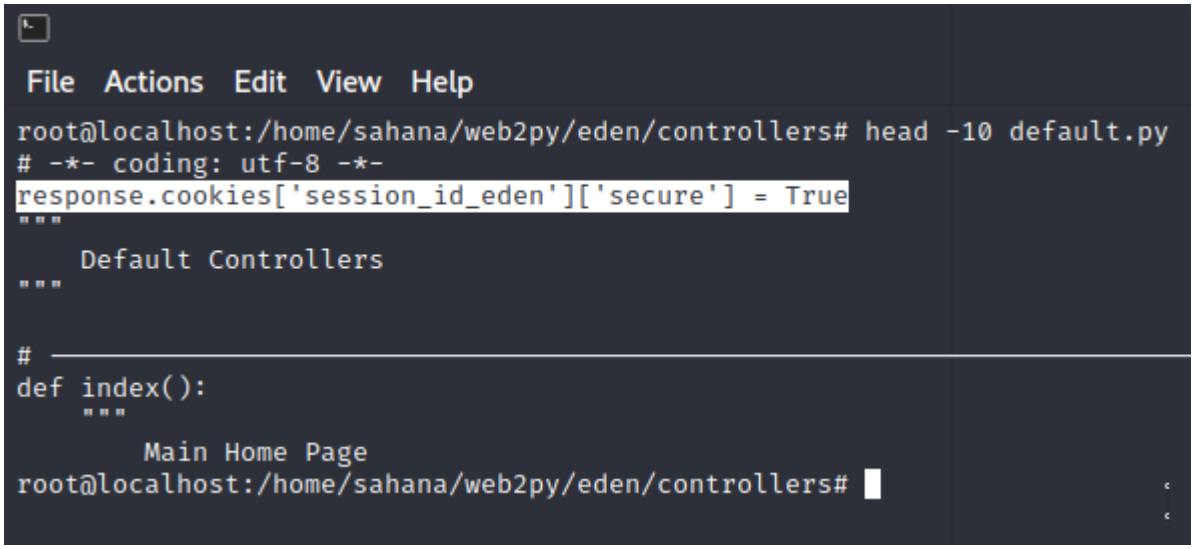

Figura 40 - Configuração da flag "secure" no cookie de sessão

Fonte: Autor

Para validar a configuração foi realizado um teste a aplicação pelo próprio navegador utilizando o modo de desenvolvimento/inspeção, navegando na página de login do SAHANA EDEN e verificando o parâmetro do cookie conforme figura 41.

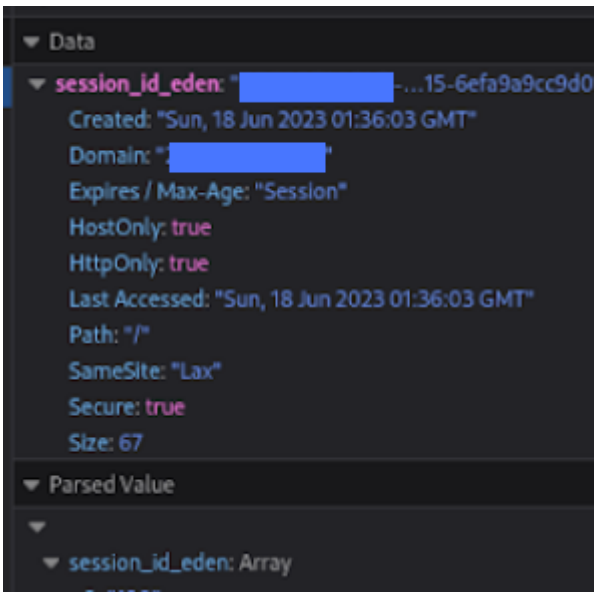

Figura 41 - Validação da configuração da flag "secure" no cookie de sessão

Fonte: Autor

Vulnerabilidade: SSL/TLS Renegotiation DoS Vulnerability

Não há exploração pública conhecida para esta vulnerabilidade, no entanto não é possível explorá-la, mas é possível validá-la quanto ao problema da renegociação.

Para validar a vulnerabilidade existente, foi utilizado o comando “openssl s_client -connect 200.xxx.xxx.xxx:8000” , figuras 42 e 43. Ao finalizar a troca de certificado e o handshake SSL foi digitado a letra “R” na tentativa de solicitar uma nova renegociação e foi aceito, isso demonstra a vulnerabilidade identificada, conforme demonstra a figura 44.

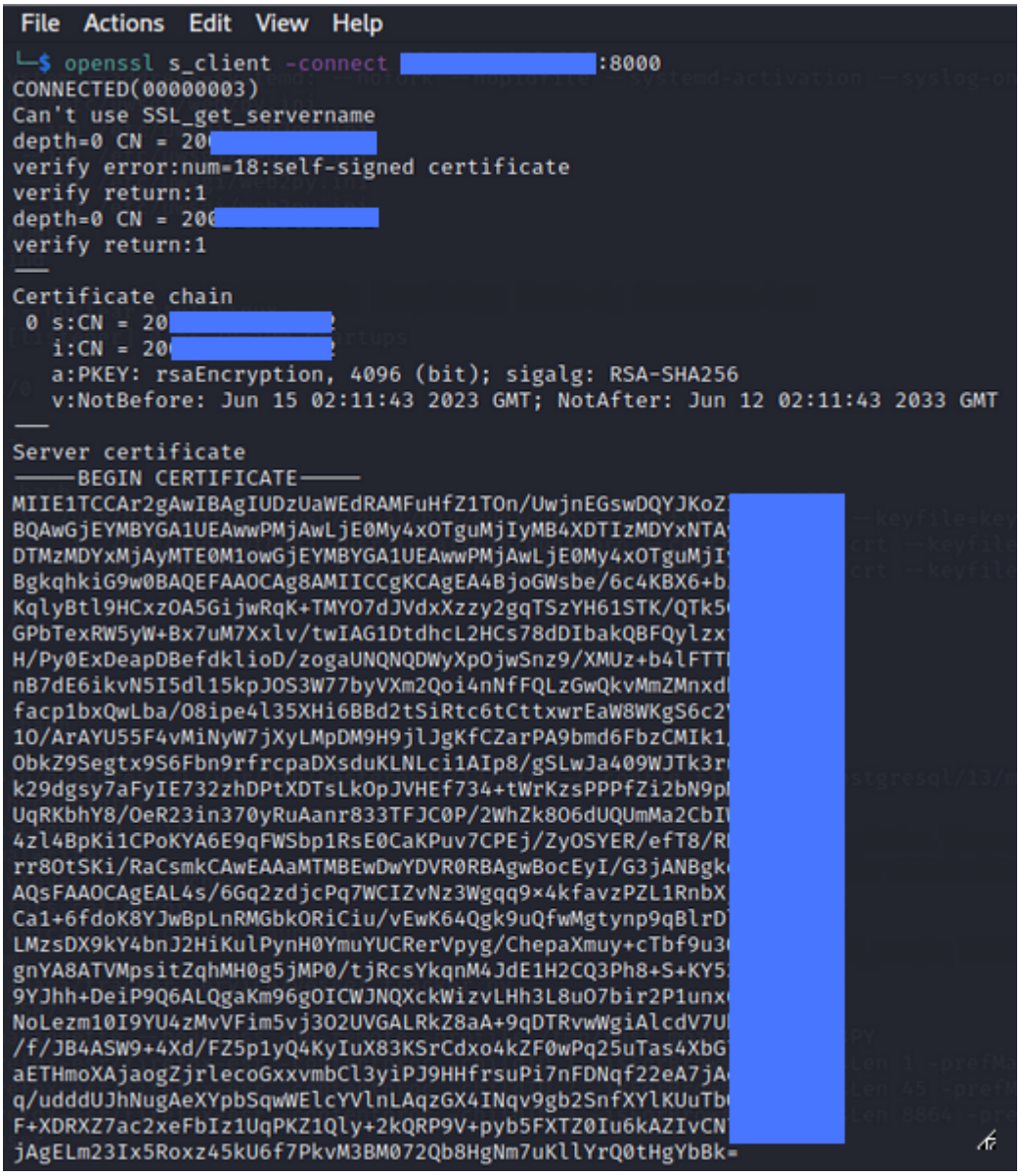

Figura 42 - Resultado do uso do comando openssl s_client

Fonte: Autor

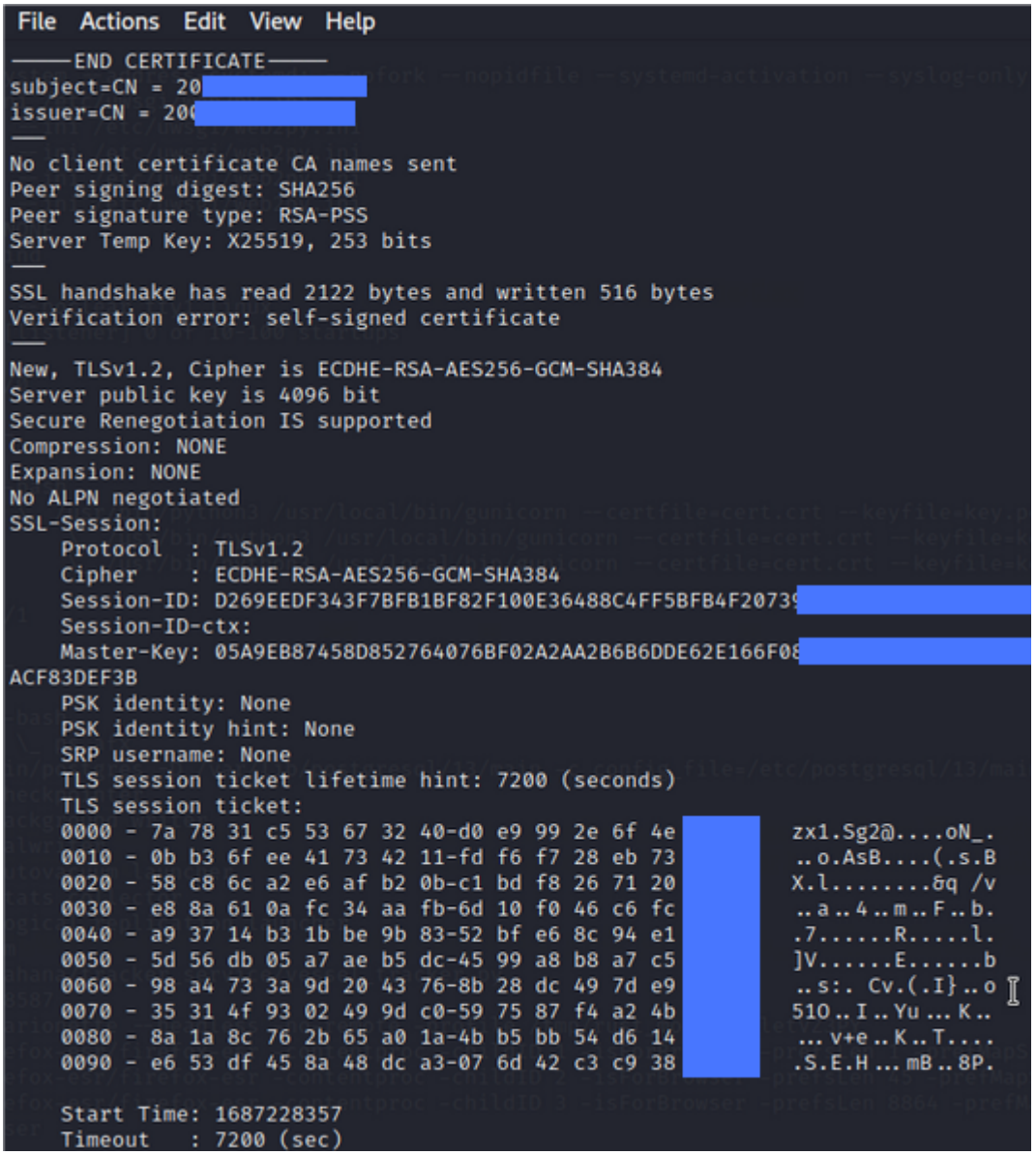

Figura 43 - Resultado (continuação) do uso do comando openssl s_client

Fonte: Autor

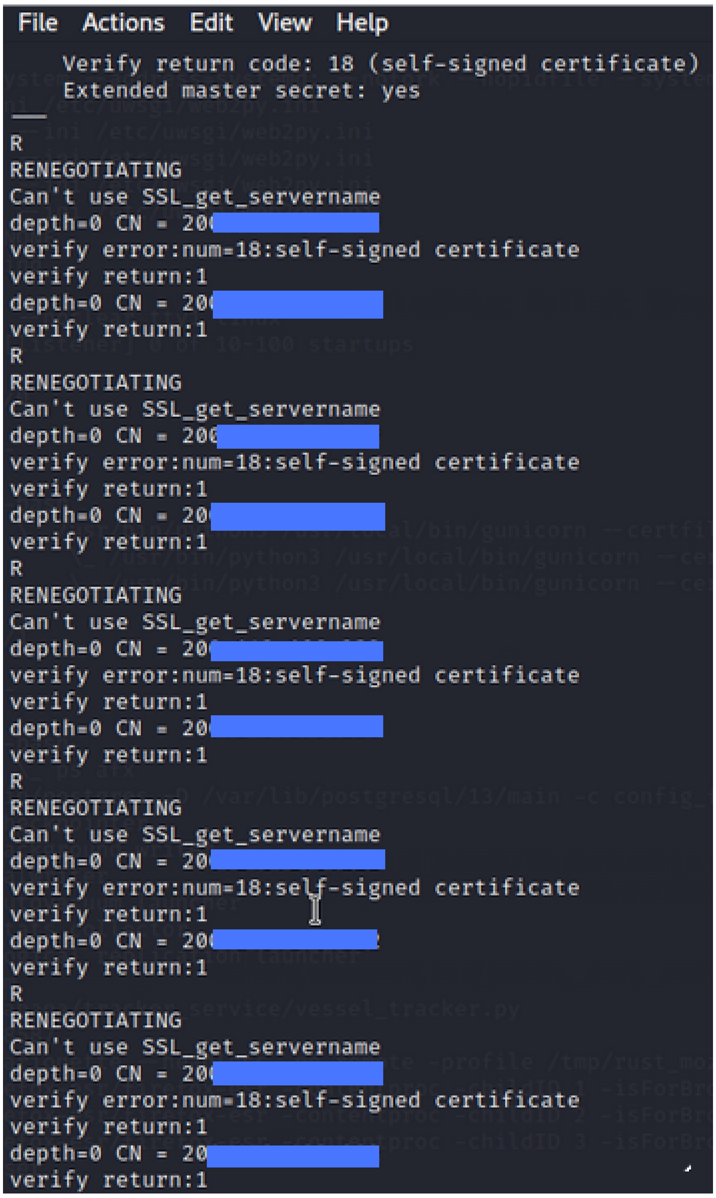

Figura 44 - Confirmação da vuln. SSL/TLS DoS Renegotiation

Fonte: Autor

Solução: Configurar a aplicação gunicorn para não permitir renegociações SSL.

Mitigação:

Consultando a última versão utilizada no site oficial e no repositório da ferramenta que podem ser encontradas nos links respectivamente a seguir https://gunicorn.org e https://github.com/benoitc/gunicorn, é a mesma versão utilizada no servidor SAHANA EDEN, gunicorn versão 20.1.0 de 28 de abril de 2021, conforme podem ser vistos nas figuras 45 e 46.

Figura 45 - Site principal do Gunicorn

Fonte: https://gunicorg.org

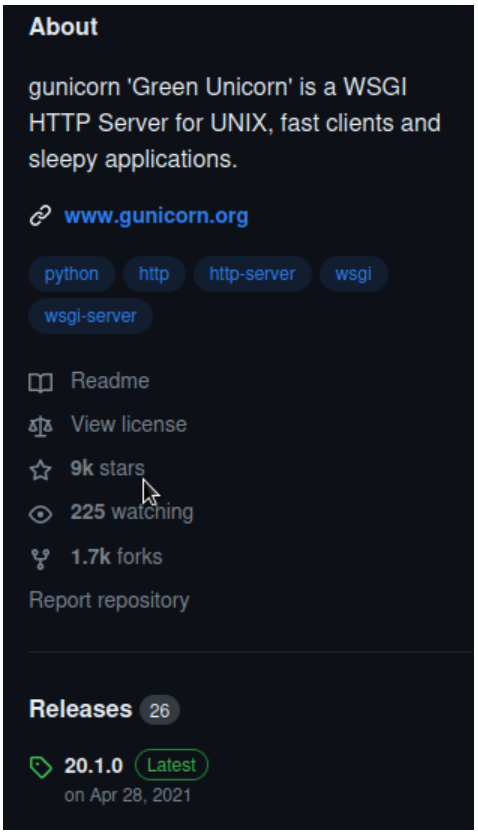

Figura 46 - Última versão do Gunicorn

Fonte: https://github.com/benoitc/gunicorn

Versão instalada da aplicação Gunicorn no SAHANA EDEN, conforme figura 47.

```
root@localhost:/home/sahana/web2py# gunicorn --version
gunicorn (version 20.1.0)
```

Figura 47 - Versão instalada do Gunicorn no SAHANA EDEN

Fonte: Autor

Ao consultar o código fonte da aplicação que atua na respectivamente na conexão SSL, o arquivo "/usr/local/lib/python3.9/dist-packages/gunicorn/workers/sync.py" foi possível identificar que é utilizado o método de conexão SSL pela função "wrap_socket()" que está considerado obsoleto, substituída pela função "SSLContext.wrap_socket()", informação esta podendo ser consultada no site https://docs.python.org/3/library/ssl.html#ssl-contexts. O código que demonstra o uso do "wrap_socket()" pode ser visto na figura 48.

```
def handle(self, listener, client, addr):
    req = None
    try:
        if self.cfg.is_ssl:
            client = ssl.wrap_socket(client, server_side=True,
                                     **self.cfg.ssl_options)

        parser = http.RequestParser(self.cfg, client, addr)
        req = next(parser)
        self.handle_request(listener, req, client, addr)
```

Figura 48 - Função SSL do arquivo sync.py

Fonte: Autor

Ao consultar o repositório da aplicação Gunicorn, não foi identificada nenhuma melhoria ou discussão que pudesse utilizar a funcionalidade de desabilitar a renegociação do SSL protegendo o software contra essa vulnerabilidade.

Ao consultar o site da biblioteca SSL https://docs.python.org/3/library/ssl.html# utilizada pelo Python para as conexões SSL utilizadas no aplicativo Gunicorn, continham informações necessárias para ajustar o código.

Foi implantado no código o novo método por "SSLContext.wrap_socket()" e com isso foi possível utilizar a função de desabilitar a renegociação. Foi substituido o código "**client = ssl.wrap_socket(client, server_side=True, **self.cfg.ssl_options)**" pelo código comentado abaixo e ilustrado na figura 49.

#inicializa do contexto SSL e configura o protocolo TLS 1.2 um dos atuais mais seguros e compatíveis.
ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)

#carrega o certificado digital e sua respectiva chave.
ssl_context.load_cert_chain(certfile="/home/sahana/web2py/cert.crt", keyfile="/home/sahana/web2py/key.pem")

#carrega as opções do contexto, desabilitando versões obsoletas do protocolo TLS e a renegociação, respectivamente.
ssl_context.options |= ssl.OP_NO_TLSv1 | ssl.OP_NO_TLSv1_1 | ssl.OP_NO_RENEGOTIATION

#carrega as cifras mais robustas do TLS v1.2.
ssl_context.set_ciphers("ECDHE-RSA-AES128-GCM-SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-CHACHA20-POLY1305")

#realiza outros tipos de validações e habilita outras opções de cifras
ssl_context.verify_mode = ssl.CERT_NONE
ssl_context.options |= ssl.OP_SINGLE_ECDH_USE
ssl_context.options |= ssl.OP_SINGLE_DH_USE
ssl_context.options |= ssl.OP_CIPHER_SERVER_PREFERENCE
ssl_context.verify_mode = ssl.CERT_NONE
ssl_context.verify_flags = ssl.VERIFY_CRL_CHECK_CHAIN | ssl.VERIFY_X509_STRICT

#inicializa a comunicação
client = ssl_context.wrap_socket(client, server_side=True)

```
def handle(self, listener, client, addr):
    req = None
    try:
        if self.cfg.is_ssl:
            ssl_context = ssl.SSLContext(ssl.PROTOCOL_TLSv1_2)
            ssl_context.load_cert_chain(certfile="/home/sahana/web2py/cert.crt", keyfile="/home/sahana/web2py/key.pem")
            ssl_context.options |= ssl.OP_NO_TLSv1 | ssl.OP_NO_TLSv1_1 | ssl.OP_NO_RENEGOTIATION
            ssl_context.set_ciphers("DEFAULT")
            ssl_context.verify_mode = ssl.CERT_NONE
            ssl_context.options |= ssl.OP_SINGLE_ECDH_USE
            ssl_context.options |= ssl.OP_SINGLE_DH_USE
            ssl_context.options |= ssl.OP_CIPHER_SERVER_PREFERENCE
            ssl_context.verify_mode = ssl.CERT_NONE
            ssl_context.verify_flags = ssl.VERIFY_CRL_CHECK_CHAIN | ssl.VERIFY_X509_STRICT

            client = ssl_context.wrap_socket(client, server_side=True)

        parser = http.RequestParser(self.cfg, client, addr)
        req = next(parser)
        self.handle_request(listener, req, client, addr)
```

Figura 49 - Ajustes realizados no arquivo sync.py

Fonte: Autor

Após a alteração do arquivo, foi reiniciado o servidor gunicorn, utilizando o comando "**gunicorn --certfile=cert.crt --keyfile=key.pem --suppress-ragged-eofs --workers 2**

--do-handshake-on-connect --access-logfile acesso.log --error-logfile erro.log wsgihandler:application -b 0.0.0.0:8000 &”

Uma vez no ar, é possível revalidar a conexão utilizando o comando “**openssl s_client -connect 200.xxx.xxx.xxx:8000**” e verificar se a opção de desabilitar a renegociação surtiu efeito. Ao digitar o primeiro “R” após completar o handshake, já é possível perceber que o servidor interrompe a conexão ao perceber que houve uma outra solicitação de renegociação.

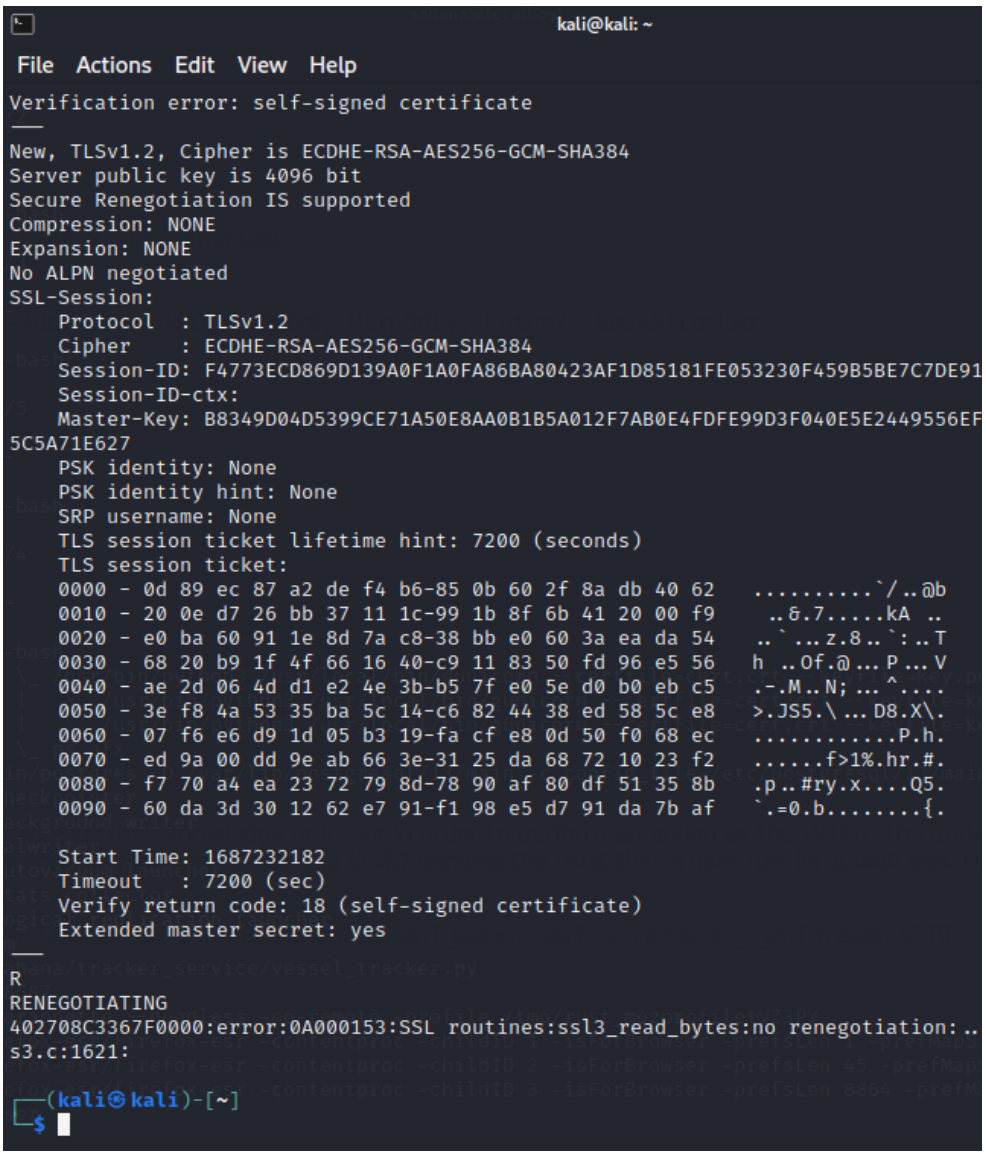

Figura 50 - Revalidação da mitigação da vuln. SSL/TLS DoS Renegotiation

Fonte: Autor

Usado o comando **nmap -Pn --script ssl-enum-ciphers 200.xxx.xxx.xxx -p 8000** para validar a versão do TLS e as cifras ativas.

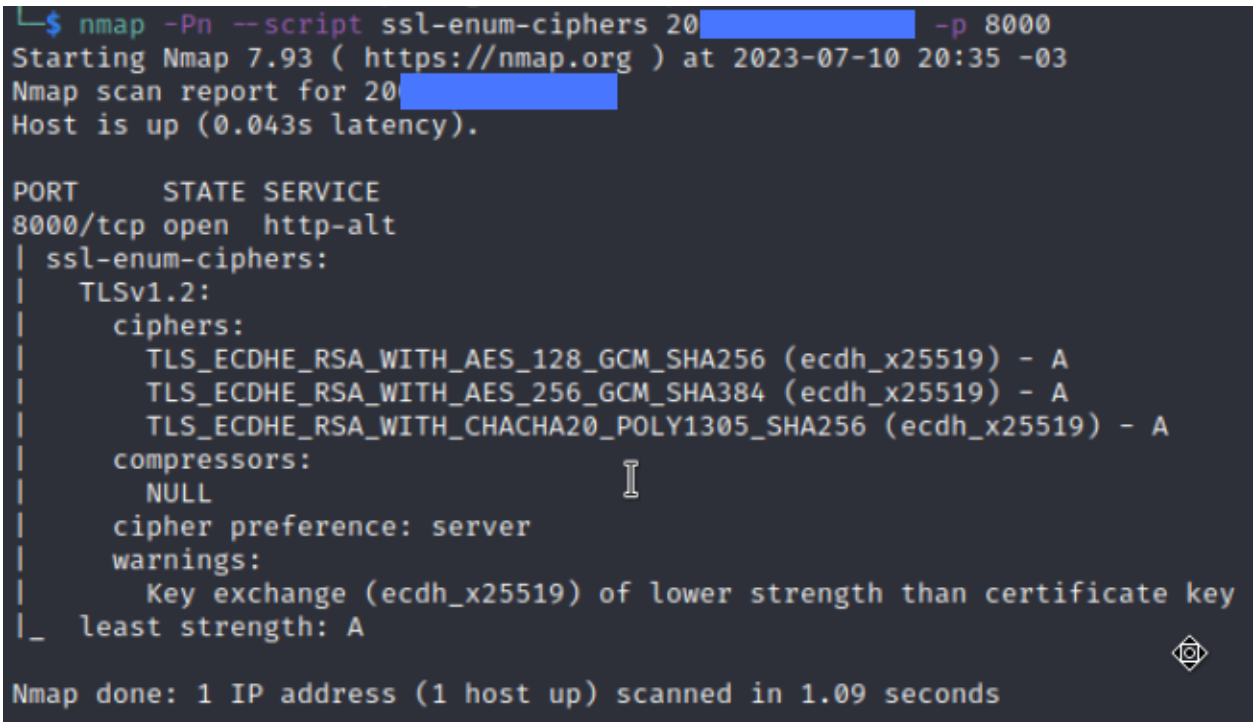

Figura 51 - Nmap revalidando a versão do TLS e as cifras

Fonte: Autor

Após aplicar a solução, foi novamente executada a tarefa de varredura com o greenbone e por fim as vulnerabilidades identificadas "*Secure Flag Cookie* não configurado" e "Vulnerabilidade SSL/TLS DoS *Renegotiation*" foram mitigadas, conforme podem se observadas na figura 52, no campo "*Severity*" podemos constatar o valor "0.0 Log" que significa que não foi encontrada nenhuma vulnerabilidade, apenas itens informativos.

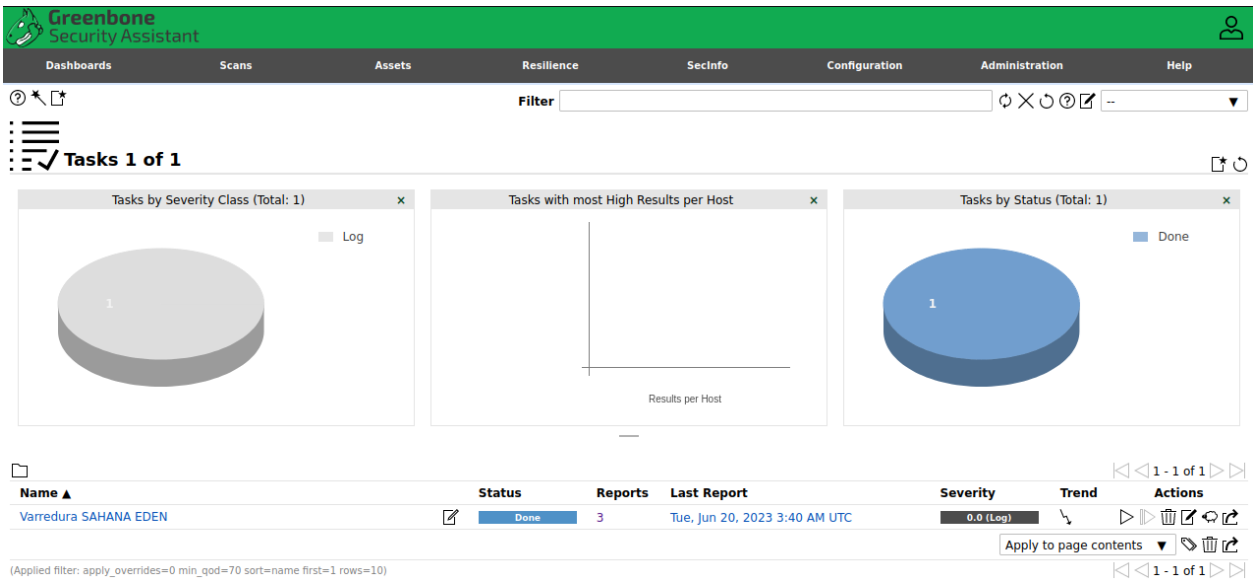

Figura 52 - Resultado final Greenbone OpenVAS

Fonte: Autor

Hardening aplicação Web2py

Etapa adicional de configurações frágeis que foram identificadas nas validações das configurações da instalação do SAHANA EDEN.

A instalação padrão do web2py, ativa automaticamente 2 websites adicionais, sendo eles o “welcome” e o “examples”, sites esses contendo exemplos e informações sobre o web2py, o que recomenda-se desativar, pois atacantes pode tentar explorá-los ou obter informações adicionais sobre a aplicação, conforme figuras 53, 54 e 55. Geralmente os aplicativos de varredura de vulnerabilidades não identificam estas informações. Já o site “admin” precisa ser protegido para acesso somente a máquina local ou então desabilitado.

```
root@localhost:/home/sahana/web2py/applications# ls -la
total 32
drwxr-xr-x  7 sahana sahana 4096 Jun 29 19:48 .
drwxr-xr-x 17 web2py sahana 4096 Jun 29 20:00 ..
drwxr-xr-x 16 sahana sahana 4096 Jun 29 19:48 admin
drwxr-xr-x 19 web2py sahana 4096 Jun 17 14:36 eden
drwxr-xr-x 15 sahana sahana 4096 Jun 29 19:48 examples
-rw-rw-r--  1 root   root      1 Mar 22 22:06 __init__.py
drwxr-xr-x  2 sahana sahana 4096 Jun 29 19:54 __pycache__
drwxr-xr-x 15 sahana sahana 4096 Jun 29 19:48 welcome
root@localhost:/home/sahana/web2py/applications#
```

Figura 53 - Lista das aplicações instaladas Web2py

Fonte: Autor

Os sites padrões da instalação “welcome” e “examples” podem ser utilizados por atacantes para descoberta de vulnerabilidades na aplicação web2py, scripts vulneráveis, etc.

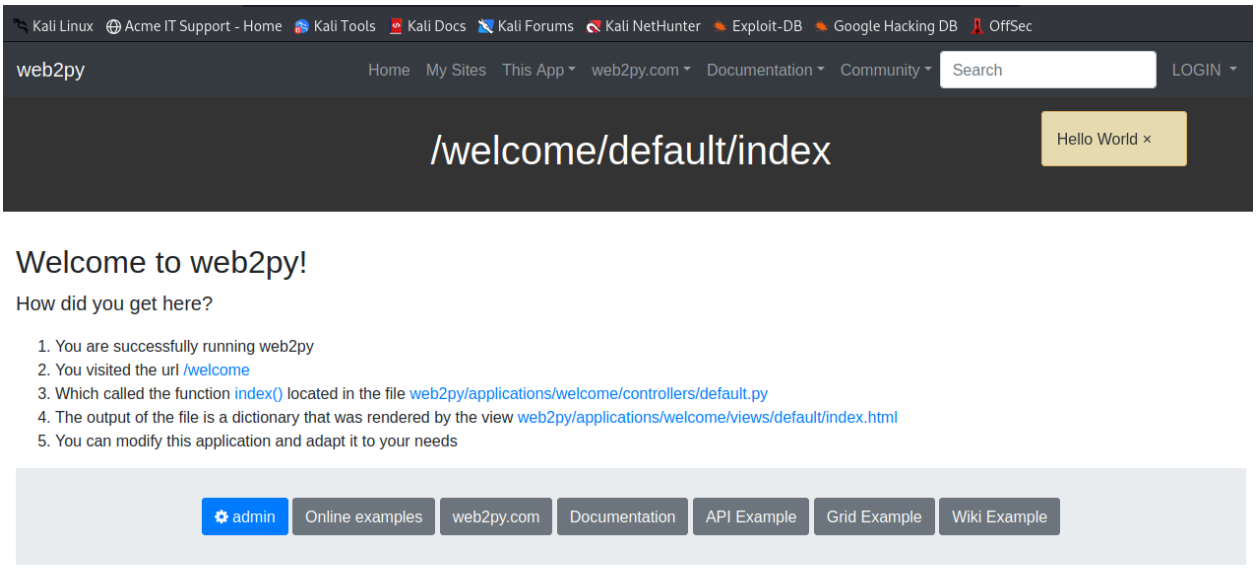

Figura 54 - Site padrão “*welcome”* Web2py

Fonte: Autor

Figura 55 - Site padrão “*examples”* Web2py

Fonte: Autor

Foram desinstalados os sites adicionais “welcome” e “examples” pela interface administrativa do web2py. Ao logar na ferramenta, foi acessado a página “*Installed Applications*” em cada um dos sites, “*welcome*” e “*examples*” acessado a configuração “*Manage->Uninstall*” isso os removeu em definitivos, conforme podem ser vistos nas figuras 56 e 57.

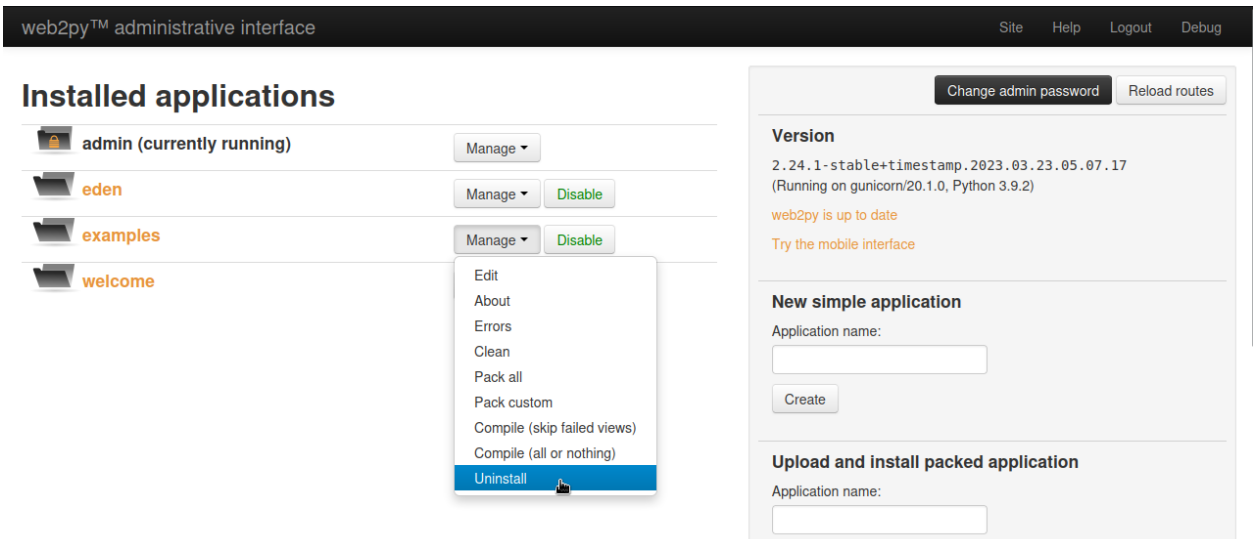

Figura 56 - Remoção do site “*examples*” do Web2py

Fonte: Autor

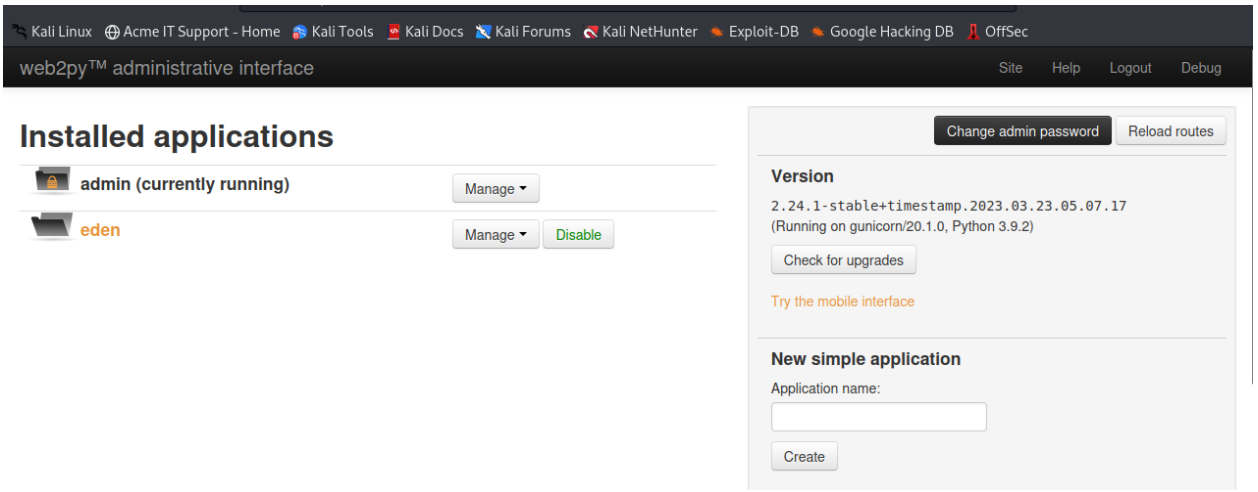

Figura 57 - Validação da remoção dos sites do Web2py

Fonte: Autor

Ausência de Tokens Anti-CSRF

Solução:

A solução implica em várias etapas desde a fase da arquitetura e design da aplicação quanto implantação de funcionalidades nos formulários e cabeçalhos do site, conforme citado nas páginas 76 a 78**.**

Mitigação:

Não foi possível a mitigação desta vulnerabilidade devido ao seu grau de complexidade, pois envolve etapas de arquitetura e design além de alterações em várias partes do código do SAHANA EDEN. Recomenda-se que siga as recomendações citadas nas páginas 76 a 78.

Content Security Policy (CSP) Header Not Set

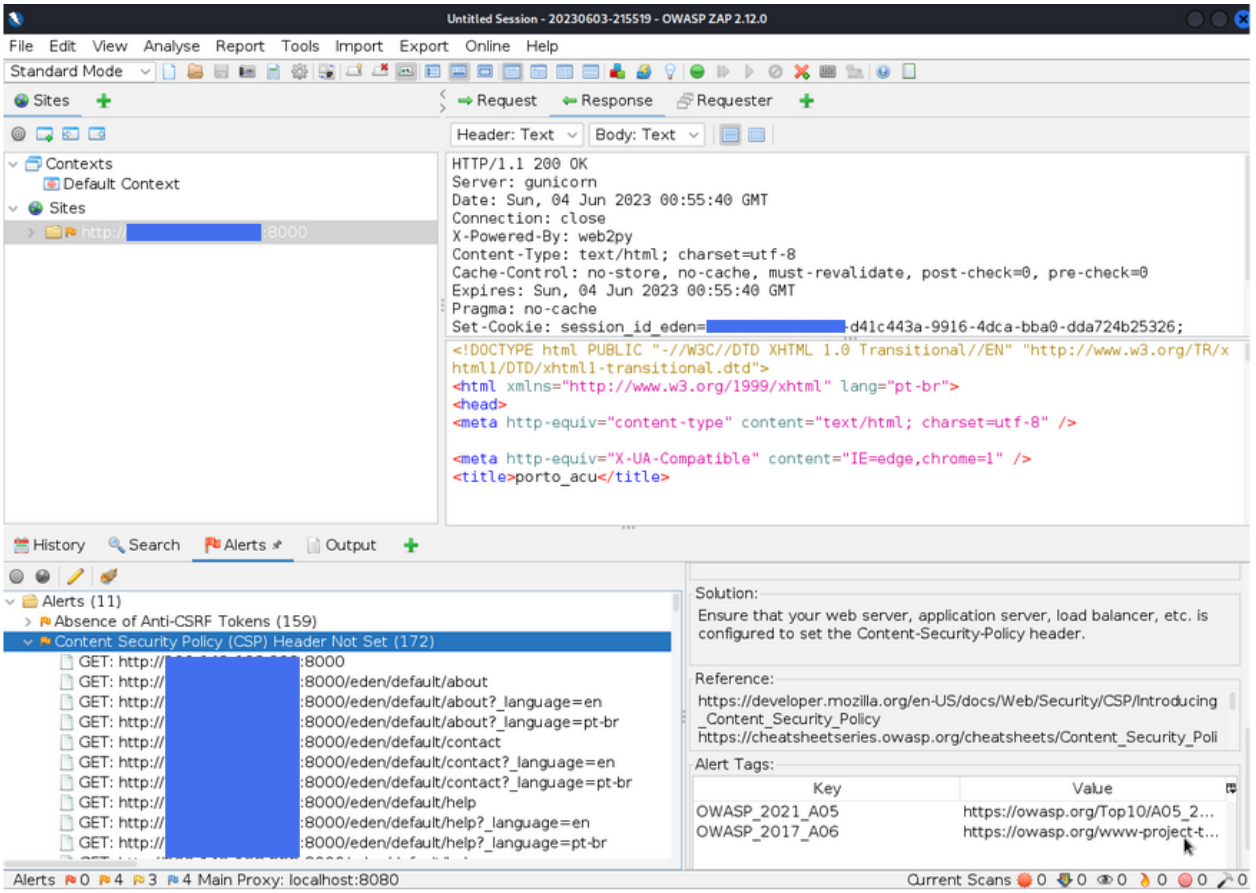

Figura 58 - Vulnerabilidade CSP Header Not Set

Fonte: OWASP ZAP

Solução: Configurar cabeçalhos de segurança na aplicação SAHANA EDEN.

Mitigação:

Esta configuração pode ser específica para cada tipo de aplicação, recomenda-se avaliar os cabeçalhos necessários conforme recomendações do OWASP Secure Headers Project que podem ser consultados e validados por ferramentas em https://owasp.org/www-project-secure-headers/index.html#configuration-proposal**.**

Nesta etapa de mitigação, que fora realizada de forma parcial, pois alguns cabeçalhos envolvem alterações em páginas do SAHANA EDEN, como por exemplo a chamada de scripts, CSS e figuras. Abaixo será listado os cabeçalhos correspondentes conforme figura 59.

Na pasta "**/home/sahana/web2py/gluon" editar o arquivo "globals.py**" inserir os headers

```
        CSP_directives = [
                "base-uri https://*/eden/ ; default-src 'self' ; script-src 'self' 'unsafe-inline' a
jax.googleapis.com ;  style-src 'self' 'unsafe-inline' ; connect-src 'self' ; img-src 'self' ; form-
action 'self' ; object-src 'none' ; frame-ancestors 'none' ; upgrade-insecure-requests ; block-all-m
ixed-content"
        ]
        self.headers['Content-Security-Policy'] = CSP_directives
```

Figura 59 - Configuração dos cabeçalhos CSP

Fonte: Autor

Foram configurados os seguintes cabeçalhos:

base-uri: Esta diretiva define um conjunto de URLs permitidos que podem ser usados no atributo src de uma tag base HTML. Esta diretiva foi configurada para permitir somente chamadas src para a raiz **"eden"** que são conteúdos exclusivos do SAHANA EDEN.

default-src*:* Esta diretiva define a política padrão para buscar recursos como JavaScript, imagens, CSS, fontes, solicitações AJAX, quadros, mídia HTML5. Nem todas as diretivas revertem para *default-src.* Esta diretiva foi configurada com o parâmetro **"*self*"** e que significa que por padrão somente poderá ser executado algum tipo de objeto apenas do próprio servidor e não remotamente.

script-src: Esta Diretiva define a origem de onde são executados os scripts em Java. Esta diretiva foi configurada com os parâmetros **"*self*"**, **"*unsafe-inline*"** e **"ajax.googleapis.com"** que significa que por padrão somente poderá ser executado algum tipo de objeto apenas do próprio servidor e não remotamente, permitir a execução de scripts inseguros na página,

manipuladores de eventos e diretamente do site de API's do Google respectivamente. O parâmetro ***"unsafe-inline"*** foi configurado pois teríamos que alterar várias partes do código de chamada a scripts do SAHANA EDEN. De qualquer forma o cabeçalho configurado protege de scripts externos minimizando a maioria dos ataques.

style-src: Esta diretiva protege o carregamento e a execução de estilos CSS e folhas de estilo. Esta diretiva foi configurada com os parâmetros **"*self*"** e **"*unsafe-inline"*** que significa que por padrão somente poderá ser executado algum tipo de objeto CSS apenas do próprio servidor e não remotamente, permitir a execução de CSS inseguros na página e manipuladores de eventos.

connect-src*:* Esta diretiva aplica-se a XMLHttpRequest (AJAX), WebSocket, fetch(), <a ping> ou EventSource. Se não for permitido, o navegador emula um código de status HTTP 400. Esta diretiva foi configurada com o parâmetro **"*self*"** que significa que por padrão somente poderá ser executado algum tipo de objeto apenas do próprio servidor e não remotamente.

img-src: Esta diretiva protege o carregamento de imagens (por exemplo, por meio de uma tag HTML <img>. Se essa diretiva não for configurada será sempre utilizada a **"*default-src"***. Esta diretiva foi configurada com o parâmetro **"*self*"** que significa que por padrão somente poderá ser executado algum tipo de imagem apenas do próprio servidor e não remotamente.

form-action: Esta diretiva permite que você especifique para qual local um formulário pode POSTAR. Esta diretiva foi configurada com o parâmetro **"*self*"** que significa que por padrão somente poderá ser usado o método POST do HTTP apenas do próprio servidor e não remotamente.

object-src: Esta diretiva especifica as fontes válidas para os elementos <object> e <embed>. Isso inclui recursos de plug-in do navegador, como controles Flash, Java e ActiveX. Esta diretiva foi configurada com o parâmetro **"*none*"** que significa bloquear qualquer carregamento de plugins como Java, Flash ou ActiveX.

frame-ancestors: Esta diretiva define origens válidas para incorporar o recurso usando <frame> <iframe> <object> <embed> <applet>. Esta diretiva foi configurada como **'none'** é equivalente a X-Frame-Options: DENY. A configuração aplicada foi **"none"** especificamente, isso significa que o URI fornecido não pode ser enquadrado dentro de um quadro ou tag iframe.

upgrade-insecure-requests: Converte automaticamente urls de http para https para links, imagens, javascript, css, etc.

blocked-all-missed-content: Bloqueia solicitações para URLs http não seguros.

Após a configuração dos cabeçalhos, foi utilizado o site https://csp-evaluator.withgoogle.com/ para validação dos cabeçalhos configurados, conforme pode ser visto na figura 60.

Figura 60 - Validação do CSP configurado no SAHANA EDEN

Fonte: https://csp-evaluator.withgoogle.com/

Missing Anti-clickjacking Header

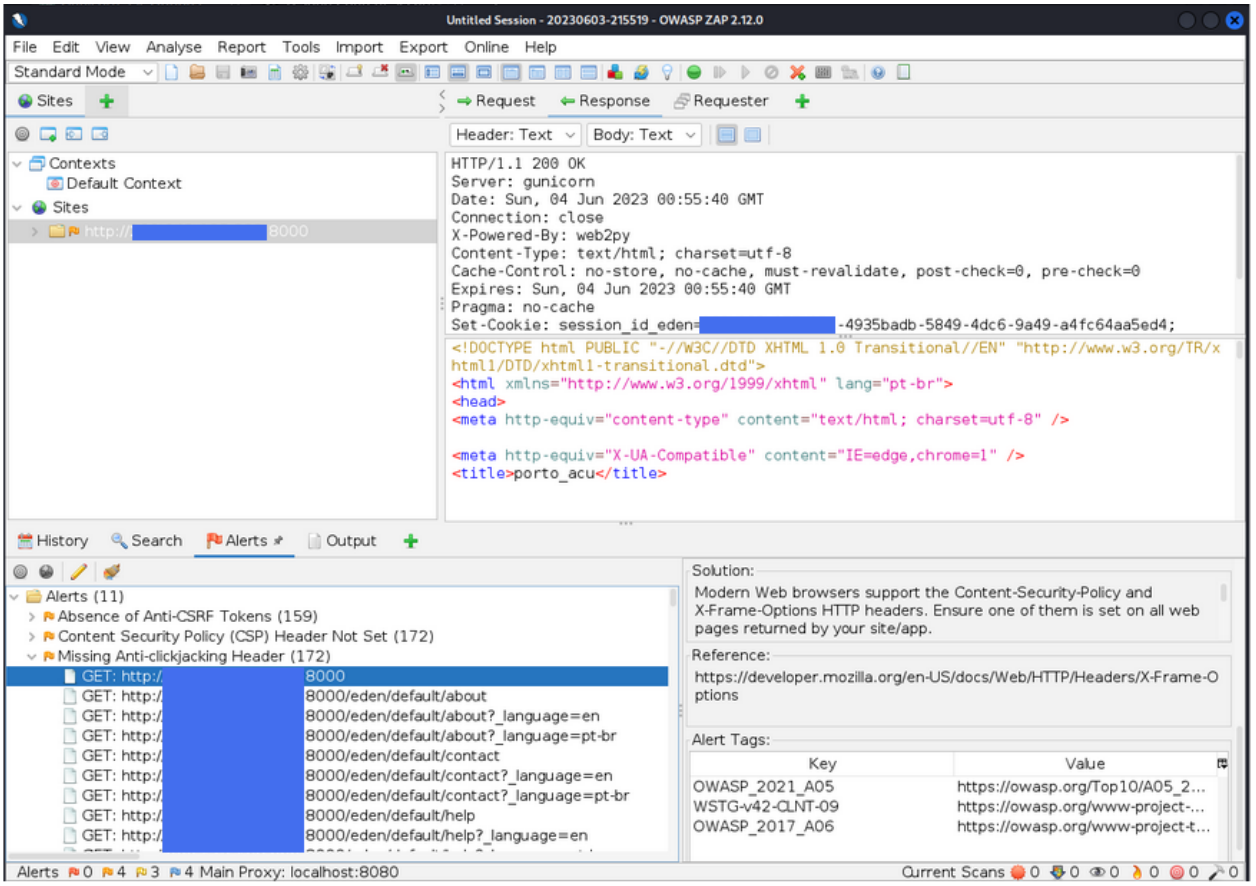

Figura 61 - Vulnerabilidade Missing Anti-clickjacking Header

Fonte: OWASP ZAP

Solução: Configurar o cabeçalho "X-Frame_Options"

O cabeçalho de resposta HTTP X-Frame-Options pode ser usado para indicar se um navegador deve ou não ter permissão para renderizar uma página em um <frame>, <iframe>, <embed> ou <object>. Os sites podem usar isso para evitar ataques de *click-jacking*, garantindo que seu conteúdo não seja incorporado a outros sites.

Mitigação:

Na pasta "**/home/sahana/web2py/gluon**" editar o arquivo "**globals.py**" e adicionar a linha "**self.headers['X-Frame-Options'] = 'SAMEORIGIN'**" permitindo somente carregar da mesma página somente.

Vulnerable JS Library

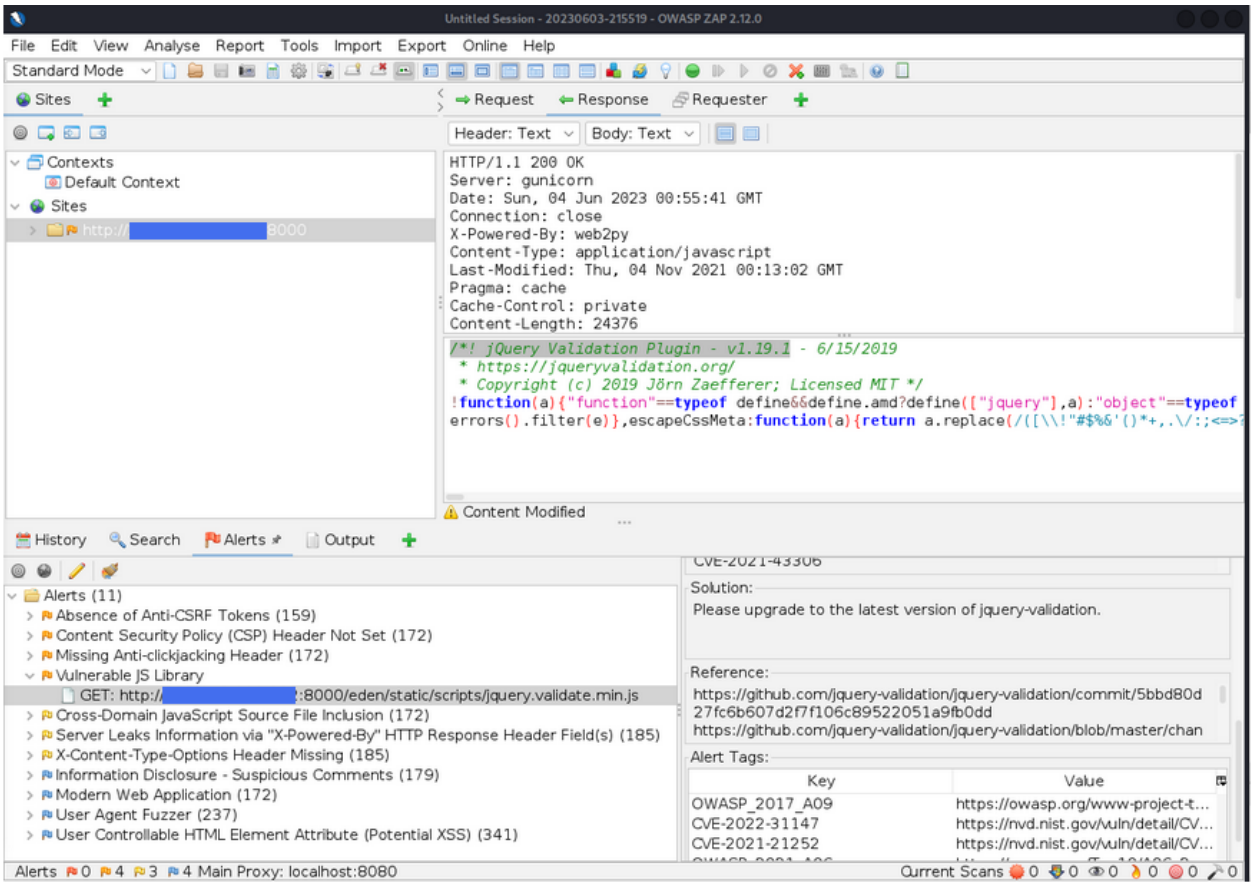

Figura 62 - Vulnerabilidade Vulnerable JS Library

Fonte: OWASP ZAP

Solução: Atualização para última versão do jquery-validation

Mitigação:

No site do repositório https://jqueryvalidation.org/, conforme figura 63. Foi baixada e instalada a última versão do jquery-validation, 1.19.5 de 01/07/2022, através das URLs.

https://cdn.jsdelivr.net/npm/jquery-validation@1.19.5/dist/jquery.validate.js

https://cdn.jsdelivr.net/npm/jquery-validation@1.19.5/dist/jquery.validate.min.js

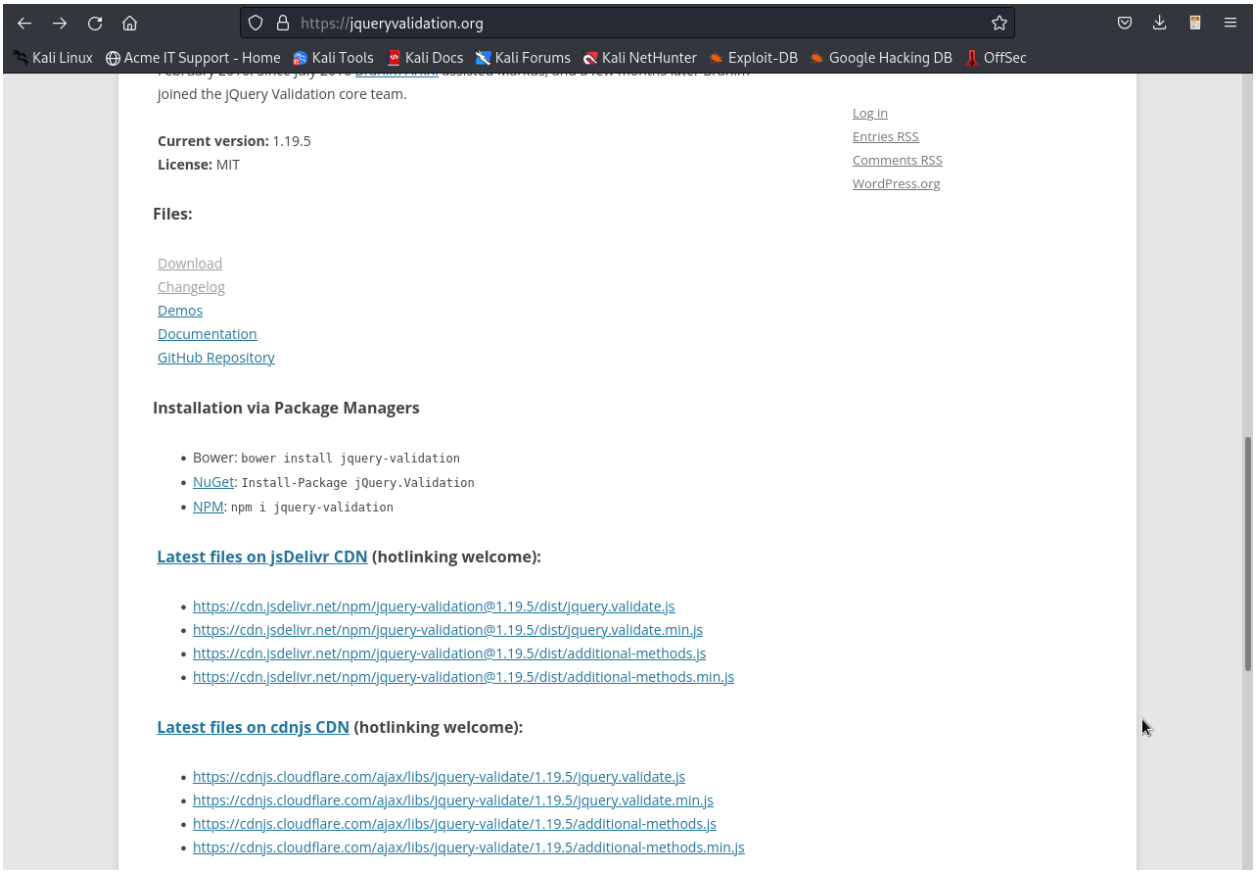

Figura 63 - Site repositório jquery-validation

Fonte: https://jqueryvalidation.org/

Acesse o diretório **“cd /home/sahana/web2py/applications/eden/static/scripts/”** Conforme figura 64.

```
root@localhost:/home/sahana/web2py# locate jquery.validate
/home/sahana/web2py/applications/eden/static/scripts/jquery.validate.js
/home/sahana/web2py/applications/eden/static/scripts/jquery.validate.min.js
root@localhost:/home/sahana/web2py#
```

Figura 64 - Diretório do jquery-validation no SAHANA EDEN

Fonte: Autor

Realizar uma cópia dos arquivos atuais, executar os comandos **“mv jquery.validate.js jquery.validate.js.old”** e **“mv jquery.validate.min.js jquery.validate.min.js.old”** e utilizar o comando **“wget”** do Linux para baixar os arquivos.

“wget https://cdn.jsdelivr.net/npm/jquery-validation@1.19.5/dist/jquery.validate.js”

“**wget https://cdn.jsdelivr.net/npm/jquery-validation@1.19.5/dist/jquery.validate.min.js**”

X-Content-Type-Options Header Missing

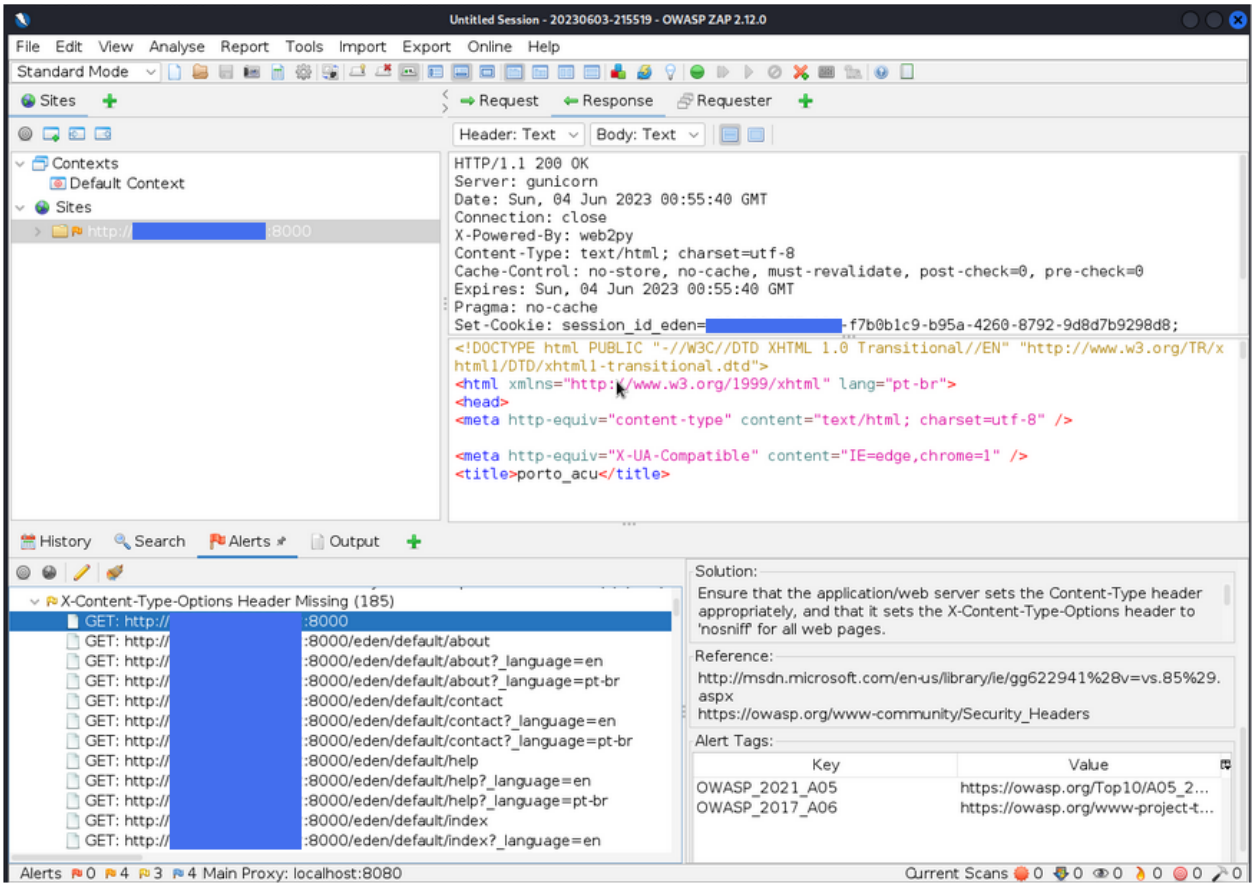

Figura 65 - Vulnerabilidade X-Content-Type-Options Header Missing

Fonte: OWASP ZAP

Solução: Configurar o cabeçalho X-Content-Type-Options

Mitigação:

Na pasta **"/home/sahana/web2py/gluon"** editar o arquivo **"globals.py"** e adicionar a linha **"self.headers['X-Content-Type-Options'] = 'nosniff' "** que bloqueia uma solicitação se o destino da solicitação for do tipo estilo e o tipo MIME não for text/css, ou do tipo script e o tipo MIME não for um tipo MIME JavaScript, protegendo assim o servidor de execuções de arquivos maliciosos.

Após a mitigação das vulnerabilidades, foi realizada uma nova varredura utilizando o OWASP ZAP e podemos observar que os itens remanescentes são os que foram parcialmente e os não mitigados, conforme demonstrado na figura 66.

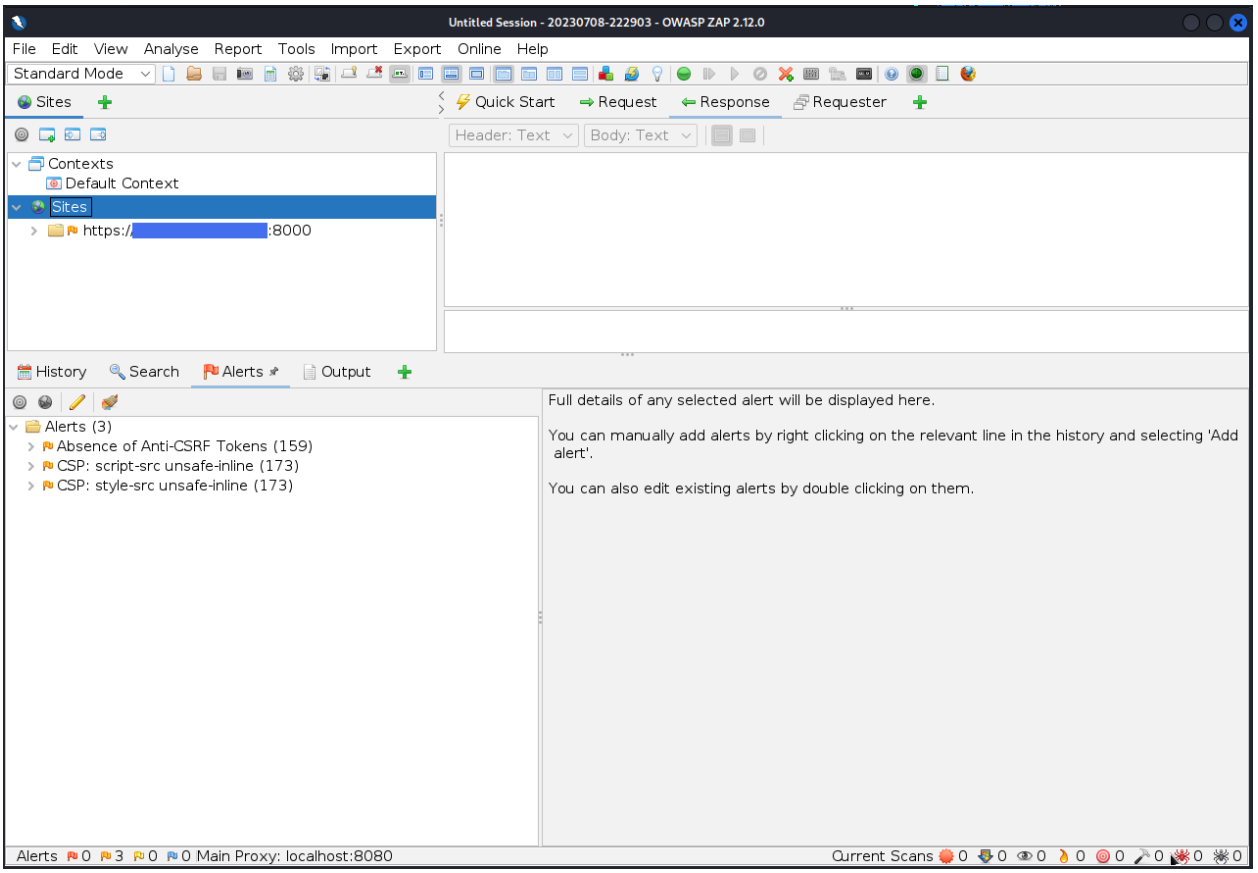

Figura 66 - Resultado final da revalidação pelo OWASP ZAP

Fonte: OWASP ZAP

Na tabela 9 demonstra o resultado consolidado de todas as vulnerabilidades identificadas, se as mesmas foram exploradas e/ou mitigadas completamente ou parcialmente, e por fim as recomendações, fazendo referência às páginas onde as mesmas foram citadas nesta dissertação.

Tabela 9 - Resultado geral consolidado

	Vulnerabilidades Identificadas x Exploração x Mitigação x Recomendações											
	Server Leaks Information via "X-Powered-By" HTTP Response Header Field	Transmissão Dados Sensíveis em Texto Claro	CSP Não Configurado	Cabeçalho X-Frame Options Não configurado	Cabeçalho X-XSS-Protection Não configurado	Cabeçalho X-Content-Type -Options Não configurado	Cabeçalho Strict-Transport-Security Não configurado	Secure Flag Cookie Não configurado	Biblioteca Java Script Vulnerável	Cross-Domain Java Script Source File Inclusion	Ausência de tokens Anti-CSRF	SSL/TLS Renegotiation DoS Vulnerability
Explorada	Sim	Sim	Não	Não	Não	Não	Não	Não	Não	Não	Não	Não
Mitigada	Sim	Sim	Parcial	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Não	Sim
Recomendações	–	–	pág. 108	–	–	–	–	–	–	–	pág. 76 a 78	–

Fonte: Autor

É importante estabelecer procedimentos de rotina sólidos para o monitoramento contínuo do servidor e do software SAHANA EDEN, visando identificar e mitigar vulnerabilidades. Com base no framework do NIST SP 800-155 e o OWASP Top Ten, citamos algumas recomendações:

4.4.2 Rotina para monitoramento e mitigação de vulnerabilidades:

Gestão de Configuração e Ativos

Manter um inventário atualizado de todos os ativos de hardware e software, incluindo versões e configurações.

Implementar um sistema de gestão de configuração para garantir que as configurações sejam consistentes e seguras.

Atualização de Software

Estabelecer um processo para monitorar e aplicar regularmente atualizações de software, incluindo patches de segurança.

Priorizar as atualizações com base em sua criticidade e potencial de impacto.

Avaliação de Vulnerabilidades

Realizar análises regulares de vulnerabilidades por meio de varreduras automatizadas.

Consultar boletins de segurança, base de dados de vulnerabilidades, como o NIST National Vulnerability Database (NVD) e o CVE Details (http://www.cvedetails.com) para identificar vulnerabilidades e exploits conhecidos.

Testes de Penetração e Avaliação de Aplicações

Realizar testes regulares de penetração para identificar vulnerabilidades específicas à aplicação.

Utilizar a lista do OWASP Top Ten para focar em ameaças mais críticas e exploradas.

Monitoramento de Registros e Análise de Logs

Implementar um sistema centralizado de gerenciamento de registros (SIEM) para coletar e analisar registros de eventos.

Monitorar logs em busca de atividades suspeitas, como tentativas de acesso não autorizado.

Detecção de Intrusões

Configurar sistemas de detecção de intrusões (IDS) e sistemas de prevenção de intrusões (IPS) para identificar comportamentos maliciosos ou anômalos.

Firewalls e Filtragem de Tráfego

Configurar firewalls para filtrar e inspecionar o tráfego de entrada e saída.

Utilizar listas de bloqueio e permissão baseadas em IPs confiáveis.

Avaliação de Segurança do Código-fonte

Implementar análises regulares de segurança no código-fonte da aplicação para identificar vulnerabilidades no desenvolvimento.

Utilizar ferramentas de análise estática de código (SAST) para identificar padrões de CWE.

Monitoramento de Ameaças Externas

Acompanhar fontes de ameaças externas, como feeds de inteligência de ameaças, para identificar tendências e ameaças emergentes.

Treinamento e Conscientização

Realizar treinamento regular de conscientização de segurança para a equipe, destacando os riscos e melhores práticas.

Security by Design

O princípio de "*Security by Design*" é fundamental para criar um ambiente seguro desde o início do desenvolvimento e reduzir a exposição a possíveis vulnerabilidades. Integrar as práticas de segurança no processo de design e desenvolvimento contribuirá significativamente para a prevenção de vulnerabilidades futuras, fortalecendo a postura geral de segurança do sistema SAHANA EDEN. Segue abaixo a lista com as principais ações que devem ser consideradas.

Implementar o conceito de "*Security by Design*" desde o início do ciclo de desenvolvimento, incorporando medidas de segurança diretamente no design da aplicação.

Realizar revisões regulares do projeto e da arquitetura para identificar e resolver possíveis problemas de segurança antes da implementação.

Introduzir análises de ameaças e modelagem de riscos para antecipar possíveis cenários de ataque e mitigar vulnerabilidades desde o design.

Garantir que os princípios de autenticação, autorização e criptografia sejam parte integrante da concepção da aplicação.

Estabelecer diretrizes claras para os desenvolvedores sobre as práticas de segurança recomendadas ao projetar e implementar novos recursos.

Ao implementar esses procedimentos de rotina, você estará fortalecendo a postura de segurança do sistema SAHANA EDEN e reduzindo a superfície de ataque a vulnerabilidades conhecidas. A segurança cibernética é uma jornada contínua, e a adaptação às novas ameaças é essencial para manter a integridade e a confidencialidade do sistema.

5. CONCLUSÃO

O sistema de gestão de desastres SANAHA EDEN, considerado um sistema de missão crítica, precisa que as principais características de segurança da informação sejam preservadas, são elas, integridade, disponibilidade e confidencialidade e consequentemente necessita estar protegido contra ataques cibernéticos.

Este trabalho contribui efetivamente tanto para o tema de segurança da informação quanto para os utilizadores de sistemas de gestão de emergências, que podem avaliar e optar por usar um sistema menos exposto a riscos. O que de fato oportuniza outras pesquisas, em identificar vulnerabilidades em outros sistemas de gestão de emergências utilizando esta mesma metodologia.

A presente dissertação teve como objetivo realizar uma avaliação de segurança cibernética abrangente no sistema SAHANA EDEN, com o intuito de avaliar sua maturidade de segurança e identificar possíveis vulnerabilidades. Para isso, uma metodologia de teste de penetração foi adotada, combinando elementos de testes externos com um nível moderado de conhecimento sobre o sistema, permitindo uma análise aprofundada e realista.

A etapa de planejamento foi cuidadosamente executada, resultando em um plano detalhado que definiu as etapas de execução e as ferramentas a serem utilizadas. Durante a fase de descoberta, foi possível identificar os ativos do sistema e da aplicação, fornecendo uma visão abrangente do ambiente. Posteriormente, na etapa de varredura e enumeração, o uso de ferramentas como o Nmap, Greenbone OpenVAS, Wapiti e OWASP ZAP possibilitou a identificação de vulnerabilidades significativas. Durante o processo, identificamos diversas vulnerabilidades que ameaçavam a segurança do sistema e a integridade dos dados dos usuários. Inclusive algumas delas foram exploradas, demonstrando as fragilidades de segurança encontradas.

A correção dessas vulnerabilidades foi um passo crucial para aprimorar a segurança do sistema, protegendo os dados sensíveis e minimizando o risco de exploração maliciosa. A implementação das ações de mitigação recomendadas fortaleceu significativamente a postura de segurança do sistema SAHANA EDEN, permitindo uma utilização mais segura e confiável para seus usuários.

A abordagem de procedimentos de rotina para monitoramento contínuo e mitigação de vulnerabilidades propõe um modelo de maturidade de governança ideal, com métodos eficazes

para garantir que os riscos sejam sempre minimizados.

As correções das vulnerabilidades e as ações de mitigação implementadas demonstraram o compromisso em garantir a segurança do sistema SAHANA EDEN. Contudo, ressaltamos que a segurança cibernética é um processo contínuo, e a manutenção de um alto padrão de segurança exige vigilância constante.

As recomendações apresentadas nesta dissertação têm o objetivo de fornecer orientações sólidas para aprimorar a segurança do sistema e proteger os dados confidenciais dos usuários. A adoção de medidas preventivas e a implementação das correções recomendadas reforçam a postura de segurança do sistema SAHANA EDEN e auxiliam na proteção contra ameaças emergentes.

Agradecemos a oportunidade de contribuir para o avanço da segurança cibernética por meio da identificação e correção das vulnerabilidades encontradas no sistema SAHANA EDEN. Esperamos que as conclusões e as ações recomendadas nesta dissertação sirvam como guia para garantir a proteção contínua do sistema contra ameaças em evolução.

6. REFERÊNCIAS

ADETUNJI, A. O.; BUTAKOV, S.; ZAVARSKY, P. Automated Security Configuration Checklist for Apple iOS Devices Using SCAP v1.2. Em: 2018 INTERNATIONAL CONFERENCE ON PLATFORM TECHNOLOGY AND SERVICE (PLATCON) 2018, Jeju. **Anais**... . Em: 2018 INTERNATIONAL CONFERENCE ON PLATFORM TECHNOLOGY AND SERVICE (PLATCON). Jeju: IEEE, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8472773/>. Acesso em: 26 jul. 2023.

AGARWAL, S. Helping or Hindering?: How Browser Extensions Undermine Security. Em: PROCEEDINGS OF THE 2022 ACM SIGSAC CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY 2022, Los Angeles CA USA. **Anais**... . Em: CCS '22: 2022 ACM SIGSAC CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY. Los Angeles CA USA: ACM, 2022. Disponível em: <https://dl.acm.org/doi/10.1145/3548606.3560685>. Acesso em: 29 jul. 2023.

AL ANHAR, A.; SURYANTO, Y. Evaluation of Web Application Vulnerability Scanner for Modern Web Application. Em: 2021 INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COMPUTER SCIENCE TECHNOLOGY (ICAICST) 2021, Yogyakarta, Indonesia. **Anais**... . Em: 2021 INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COMPUTER SCIENCE TECHNOLOGY (ICAICST). Yogyakarta, Indonesia: IEEE, 2021. Disponível em: <https://ieeexplore.ieee.org/document/9497831/>. Acesso em: 13 nov. 2022.

AL-BOGHDADY, A.; WASSIF, K.; EL-RAMLY, M. The Presence, Trends, and Causes of Security Vulnerabilities in Operating Systems of IoT's Low-End Devices. **Sensors**, [s. l.], v. 21, n. 7, p. 2329, 2021.

BHARGAVAN, K.; FOURNET, C.; KOHLWEISS, M.; PIRONTI, A.; STRUB, P. Implementing TLS with Verified Cryptographic Security. Em: 2013 IEEE SYMPOSIUM ON SECURITY AND PRIVACY 2013, Berkeley, CA. **Anais**... . Em: 2013 IEEE SYMPOSIUM ON SECURITY AND PRIVACY (SP) CONFERENCE. Berkeley, CA: IEEE, 2013. Disponível em: <http://ieeexplore.ieee.org/document/6547126/>. Acesso em: 31 jul. 2023.

BO YU. Based on the network sniffer implement network monitoring. Em: 2010 INTERNATIONAL CONFERENCE ON COMPUTER APPLICATION AND SYSTEM MODELING (ICCASM 2010) 2010, Taiyuan, China. **Anais**... . Em: 2010 INTERNATIONAL CONFERENCE ON COMPUTER APPLICATION AND SYSTEM MODELING (ICCASM 2010). Taiyuan, China: IEEE, 2010. Disponível em: <http://ieeexplore.ieee.org/document/5620505/>. Acesso em: 31 jul. 2023.

BURKLE JR., F. M. Challenges of Global Public Health Emergencies: Development of a Health-Crisis Management Framework. **The Tohoku Journal of Experimental Medicine**, [s. l.], v. 249, n. 1, p. 33–41, 2019.

CHAHAL, N. S.; BALI, P.; KHOSLA, P. K. A Proactive Approach to assess web application security through the integration of security tools in a Security Orchestration Platform. **Computers & Security**, [s. l.], v. 122, p. 102886, 2022.

CHALVATZIS, I.; KARRAS, D. A.; PAPADEMETRIOU, R. C. Evaluation of Security Vulnerability Scanners for Small and Medium Enterprises Business Networks Resilience towards Risk Assessment. Em: 2019 IEEE INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COMPUTER APPLICATIONS (ICAICA) 2019, Dalian, China. **Anais**... . Em: 2019 IEEE INTERNATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE AND COMPUTER APPLICATIONS (ICAICA). Dalian, China: IEEE, 2019. Disponível em: <https://ieeexplore.ieee.org/document/8873438/>. Acesso em: 3 jun. 2023.

CURRION, P.; SILVA, C. De; VAN DE WALLE, B. Open source software for disaster management. **Communications of the ACM**, [s. l.], v. 50, n. 3, p. 61–65, 2007.

DINIS, B.; SERRAO, C. External footprinting security assessments: Combining the PTES framework with open-source tools to conduct external footprinting security assessments. Em: INTERNATIONAL CONFERENCE ON INFORMATION SOCIETY (I-SOCIETY 2014) 2014, London, United Kingdom. **Anais**... . Em: 2014 INTERNATIONAL CONFERENCE ON INFORMATION SOCIETY (I-SOCIETY). London, United Kingdom: IEEE, 2014. Disponível em: <http://ieeexplore.ieee.org/document/7009066/>. Acesso em: 21 abr. 2023.

DOUPÉ, A.; COVA, M.; VIGNA, G. Why Johnny Can't Pentest: An Analysis of Black-Box Web Vulnerability Scanners. Em: KREIBICH, C.; JAHNKE, M. (Eds.). **Detection of Intrusions and Malware, and Vulnerability Assessment**. Lecture Notes in Computer ScienceBerlin, Heidelberg: Springer Berlin Heidelberg, 2010. v. 6201p. 111–131.

GHORBANI, A. A.; BAGHERI, E. The state of the art in critical infrastructure protection: a framework for convergence. **International Journal of Critical Infrastructures**, [s. l.], v. 4, n. 3, p. 215, 2008.

HAN, Z.; LI, X.; XING, Z.; LIU, H.; FENG, Z. Learning to Predict Severity of Software Vulnerability Using Only Vulnerability Description. Em: 2017 IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME) 2017, Shanghai. **Anais**... . Em: 2017 IEEE INTERNATIONAL CONFERENCE ON SOFTWARE MAINTENANCE AND EVOLUTION (ICSME). Shanghai: IEEE, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8094415/>. Acesso em: 28 jul. 2023.

LAVRENOVS, A.; MELON, F. J. R. HTTP security headers analysis of top one million websites. Em: 2018 10TH INTERNATIONAL CONFERENCE ON CYBER CONFLICT (CYCON) 2018, Tallinn. **Anais**... . Em: 2018 10TH INTERNATIONAL CONFERENCE ON CYBER CONFLICT (CYCON). Tallinn: IEEE, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8405025/>. Acesso em: 29 jul. 2023.

LIN, C.-H.; CHEN, C.-H.; LAIH, C.-S. A Study and Implementation of Vulnerability Assessment and Misconfiguration Detection. Em: 2008 IEEE ASIA-PACIFIC SERVICES COMPUTING CONFERENCE 2008, Yilan. **Anais**... . Em: 2008 IEEE ASIA-PACIFIC SERVICES COMPUTING CONFERENCE (APSCC 2008). Yilan: IEEE, 2008. Disponível em: <https://ieeexplore.ieee.org/document/4780851/>. Acesso em: 28 jul. 2023.

LOPEZ DE JIMENEZ, R. E. Pentesting on web applications using ethical - hacking. Em: 2016 IEEE 36TH CENTRAL AMERICAN AND PANAMA CONVENTION (CONCAPAN XXXVI) 2016, San Jose. **Anais**... . Em: 2016 IEEE 36TH CENTRAL AMERICAN AND PANAMA CONVENTION (CONCAPAN XXXVI). San Jose: IEEE, 2016. Disponível em:

<https://ieeexplore.ieee.org/document/7942364/>. Acesso em: 7 fev. 2023.

LV, Y.; SHI, W.; ZHANG, W.; LU, H.; TIAN, Z. Do Not Trust the Clouds Easily: The Insecurity of Content Security Policy Based on Object Storage. **IEEE Internet of Things Journal**, [s. l.], v. 10, n. 12, p. 10462–10470, 2023.

MAKINO, Y.; KLYUEV, V. Evaluation of web vulnerability scanners. Em: 2015 IEEE 8TH INTERNATIONAL CONFERENCE ON INTELLIGENT DATA ACQUISITION AND ADVANCED COMPUTING SYSTEMS: TECHNOLOGY AND APPLICATIONS (IDAACS) 2015, Warsaw, Poland. **Anais**... . Em: 2015 IEEE 8TH INTERNATIONAL CONFERENCE ON INTELLIGENT DATA ACQUISITION AND ADVANCED COMPUTING SYSTEMS: TECHNOLOGY AND APPLICATIONS (IDAACS). Warsaw, Poland: IEEE, 2015. Disponível em: <http://ieeexplore.ieee.org/document/7340766/>. Acesso em: 23 jul. 2023.

MELL, P.; SCARFONE, K.; ROMANOSKY, S. Common Vulnerability Scoring System. **IEEE Security and Privacy Magazine**, [s. l.], v. 4, n. 6, p. 85–89, 2006.

PALMER, C. C. Ethical hacking. **IBM Systems Journal**, [s. l.], v. 40, n. 3, p. 769–780, 2001.

RIADI, I.; IFANI, A. Z.; KUSUMA, R. S. Optimization and Evaluation of Authentication System using Blockchain Technology. **Emerging Science Journal**, [s. l.], v. 4, p. 225–240, 2022.

SAPORI, E.; SCIUTTO, M.; SCIUTTO, G. A Quantitative Approach to Risk Management in Critical Infrastructures. **Transportation Research Procedia**, [s. l.], v. 3, p. 740–749, 2014.

SCARFONE, K. A.; SOUPPAYA, M. P.; CODY, A.; OREBAUGH, A. D. **Technical guide to information security testing and assessment.** Gaithersburg, MD: National Institute of Standards and Technology, 2008. Disponível em: <https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-115.pdf>. Acesso em: 9 fev. 2023.

SHAH, S.; MEHTRE, B. M. An overview of vulnerability assessment and penetration testing techniques. **Journal of Computer Virology and Hacking Techniques**, [s. l.], v. 11, n. 1, p. 27–49, 2015.

SILVA, E. L. Da; MENEZES, E. M. **Metodologia da pesquisa e elaboração de dissertação**. 4. ed. Florianopolis: UFSC, 2005. Disponível em: <https://tccbiblio.paginas.ufsc.br/files/2010/09/024_Metodologia_de_pesquisa_e_elaboracao_de_teses_e_dissertacoes1.pdf>. Acesso em: 11 jun. 2021.

THAQI, R.; VISHI, K.; REXHA, B. Enhancing Burp Suite with Machine Learning Extension for Vulnerability Assessment of Web Applications. **Journal of Applied Security Research**, [s. l.], p. 1–19, 2022.

THOMAS, R. J.; GARDINER, J.; CHOTHIA, T.; SAMANIS, E.; PERRETT, J.; RASHID, A. Catch Me If You Can: An In-Depth Study of CVE Discovery Time and Inconsistencies for Managing Risks in Critical Infrastructures. Em: PROCEEDINGS OF THE 2020 JOINT WORKSHOP ON CPS&IOT SECURITY AND PRIVACY 2020, Virtual Event USA. **Anais**... .

Em: CCS '20: 2020 ACM SIGSAC CONFERENCE ON COMPUTER AND COMMUNICATIONS SECURITY. Virtual Event USA: ACM, 2020. Disponível em: <https://dl.acm.org/doi/10.1145/3411498.3419970>. Acesso em: 28 jul. 2023.

WANG, Y.; BAI, Y.; LI, L.; CHEN, X.; CHEN, A. Design of Network Vulnerability Scanning System Based on NVTs. Em: 2020 IEEE 5TH INFORMATION TECHNOLOGY AND MECHATRONICS ENGINEERING CONFERENCE (ITOEC) 2020, Chongqing, China. **Anais**... . Em: 2020 IEEE 5TH INFORMATION TECHNOLOGY AND MECHATRONICS ENGINEERING CONFERENCE (ITOEC). Chongqing, China: IEEE, 2020. Disponível em: <https://ieeexplore.ieee.org/document/9141812/>. Acesso em: 25 jul. 2023.

WANG, Y.; YANG, J. Ethical Hacking and Network Defense: Choose Your Best Network Vulnerability Scanning Tool. Em: 2017 31ST INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS (WAINA) 2017, Taipei, Taiwan. **Anais**... . Em: 2017 31ST INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS (WAINA). Taipei, Taiwan: IEEE, 2017. Disponível em: <http://ieeexplore.ieee.org/document/7929663/>. Acesso em: 13 nov. 2022.

WICAKSANA, A.; WIRA, J. C. **Security Analysis of Private Blockchain Implementation for Digital Diploma**, ICIC International 学会, 2022. Disponível em: <https://doi.org/10.24507/ijicic.18.05.1601>. Acesso em: 28 jul. 2023.

WILLIAMS, R.; MCMAHON, E.; SAMTANI, S.; PATTON, M.; CHEN, H. Identifying vulnerabilities of consumer Internet of Things (IoT) devices: A scalable approach. Em: 2017 IEEE INTERNATIONAL CONFERENCE ON INTELLIGENCE AND SECURITY INFORMATICS (ISI) 2017, Beijing, China. **Anais**... . Em: 2017 IEEE INTERNATIONAL CONFERENCE ON INTELLIGENCE AND SECURITY INFORMATICS (ISI). Beijing, China: IEEE, 2017. Disponível em: <http://ieeexplore.ieee.org/document/8004904/>. Acesso em: 3 jun. 2023.