



SUNSET SEC

Relatório de Teste de Penetração

SNS

SNS Pentest

Período de Teste: 2025-03-09 - 2025-03-11

Sunset Security

Professional Penetration Testing Services

Índice

Resumo Executivo	3
Recomendações	4
Escopo	5
Metodologia	6
Vulnerabilidades	7
Descobertas	11
CONCLUSÃO FINAL	11

Resumo Executivo

Foi realizado um teste de penetração do tipo Graybox na infraestrutura da SNS, conforme solicitado. Durante o teste, identificamos e exploramos múltiplas vulnerabilidades que permitiram o acesso completo ao servidor com privilégios de root, bem como acesso irrestrito ao banco de dados MySQL.

Aplicamos diversas técnicas de movimento lateral, inclusive, para ter certeza de que a escalação de privilégios poderia ser possível em mais de uma maneira. Encontramos vários aplicativos com SUID e testamos os mais promissores, porém sem sucesso. Entretanto, conseguimos atingir o nível mais alto do sistema, tendo utilizado de um exploit conhecido para a versão do Linux sendo utilizada pelo servidor.

Recomendações

Nenhuma recomendação fornecida.

Escopo

O escopo do projeto restringiu-se ao único servidor da SNS, conforme contratado. Nenhuma outro servidor ou serviço fora do endereço IP mas dentro da faixa do IP providenciado foi explorado.

Apenas o servidor e os serviços localizados no IP 3.219.254.0 foram alvo do teste de penetração, bem como seus repositórios GIT e dependências.

Metodologia

Utilizamos uma aproximação tática e metódica para o teste de penetração, empregamos uma gama ampla de ferramentas de descoberta, enumeração e exploração além de métodos manuais. Abaixo estão algumas das ferramentas, táticas e explorações empregadas:

Ferramentas Utilizadas Durante o Teste de Penetração

Esta tabela resume as principais ferramentas e técnicas empregadas durante nosso teste de penetração, desde o reconhecimento inicial até o comprometimento total do sistema e acesso aos dados.

Ferramenta/Técnica	Finalidade	Uso em Nosso Pentest
Nmap	Escaneamento de portas e descoberta de serviços	Identificação de portas abertas (22, 80) e enumeração inicial de serviços
Curl	Manipulação de requisições HTTP	Utilizado para contornar autenticação e acessar páginas administrativas protegidas
Nikto	Scanner de vulnerabilidades em servidores web	Descobriu o diretório `/dev/` exposto com listagem de diretório habilitada
GitTools	Análise de repositórios Git	Extração e análise do repositório Git exposto
Burp Suite	Proxy web e manipulação de requisições	Utilizado para analisar e manipular requisições para a aplicação web
Exploração SSRF	Server-Side Request Forgery	Explorou o arquivo `info.php` vulnerável para ler arquivos do servidor
SSH	Acesso via Shell Seguro	Acessou o servidor usando credenciais recuperadas
Shell PHP	Execução remota de comandos	Criação de shell ` `.phtml` para executar comandos no servidor
Escalação de Privilégios (OverlayFS)	Exploração de vulnerabilidade do kernel Linux	Escalou privilégios de www-data para root

Ferramenta/Técnica	Finalidade	Uso em Nosso Pentest
Cliente MySQL	Interação com banco de dados	Acessou e examinou o banco de dados após obter acesso root
Directory Traversal	Exploração do sistema de arquivos	Utilizado para navegar e descobrir arquivos sensíveis
Bypass de Upload de Arquivos	Teste de segurança de aplicações web	Contornou filtros de extensão para upload de arquivos maliciosos
Port Forwarding	Redirecionamento de tráfego de rede	Utilizado para acessar serviço interno na porta 8081
Dirbuster	Descoberta de arquivos e pastas no serviço web	Utilizado para descobrir páginas sensíveis

Vulnerabilidades

1. Divulgação de Versionamento:

Média

report.description

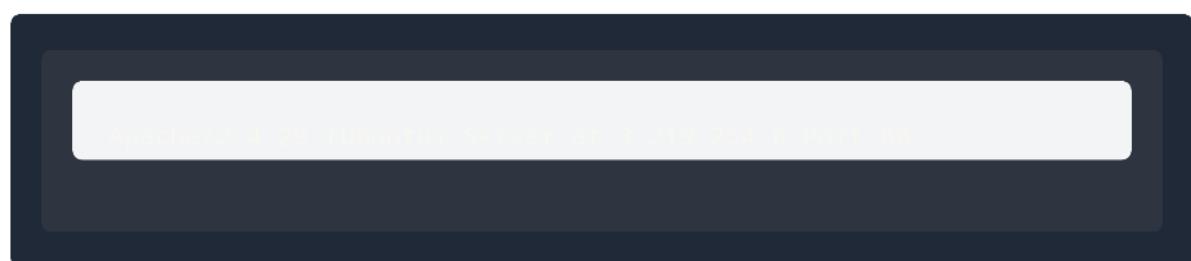
Tanto a aplicação web exposta ao público quanto o serviço de SSH divulgavam suas versões publicamente.

Ao entrar numa página onde o resultado era 404, constatamos que era possível obter o versionamento do Apache, bem como o sistema operacional de seu host. Ao usar a ferramenta netcat para conectar ao serviço de SSH, também obtivemos resultado similar.

Detalhes Técnicos

Servidor Web (Apache)

- Versão do Apache: Apache/2.4.29 (Ubuntu)
- Sistema Operacional do Host: Ubuntu
- Método de detecção: Acesso a uma página inexistente (erro 404)
- Exemplo de saída:

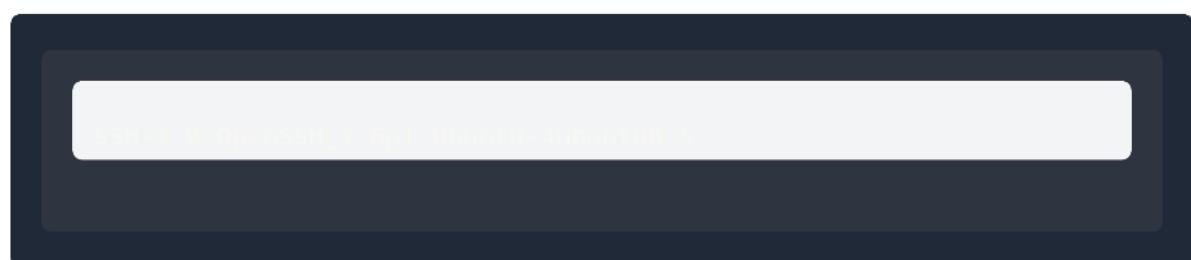


```
root@victims:~# curl -I http://192.168.1.111/index.html
HTTP/1.1 404 Not Found
Date: Mon, 12 Jul 2021 14:45:21 GMT
Server: Apache/2.4.29 (Ubuntu)
Content-Type: text/html; charset=UTF-8
Content-Length: 220
Connection: close

```

Serviço SSH

- Versão SSH: SSH-2.0-OpenSSH_7.6p1
- Método de detecção: Conexão via netcat (nc)
- Comando utilizado: `nc -v 3.219.254.0 22`
- Exemplo de saída:



```
root@victims:~# nc -v 3.219.254.0 22
OpenSSH_7.6p1, LibreSSL 2.6.5

```

Impacto

A exposição de informações de versão facilita a enumeração por parte de possíveis atacantes, permitindo:

- Pesquisa direcionada de vulnerabilidades conhecidas para as versões específicas detectadas
- Identificação precisa do sistema operacional e configurações
- Planejamento de ataques mais eficientes baseados nas tecnologias identificadas

Remediação

Para o Apache:

1. Edite o arquivo de configuração principal do Apache (normalmente

```
`/etc/httpd/conf/`
```

```
httpd.conf` ,
```

2. Adicione ou modifique as seguintes diretivas:

```
# Ocultar informações de versão do Apache
```

```
ServerSignature Off
```

3. Reinicie o serviço Apache:

```
sudo apachectl restart
```

ou

```
sudo service httpd restart
```

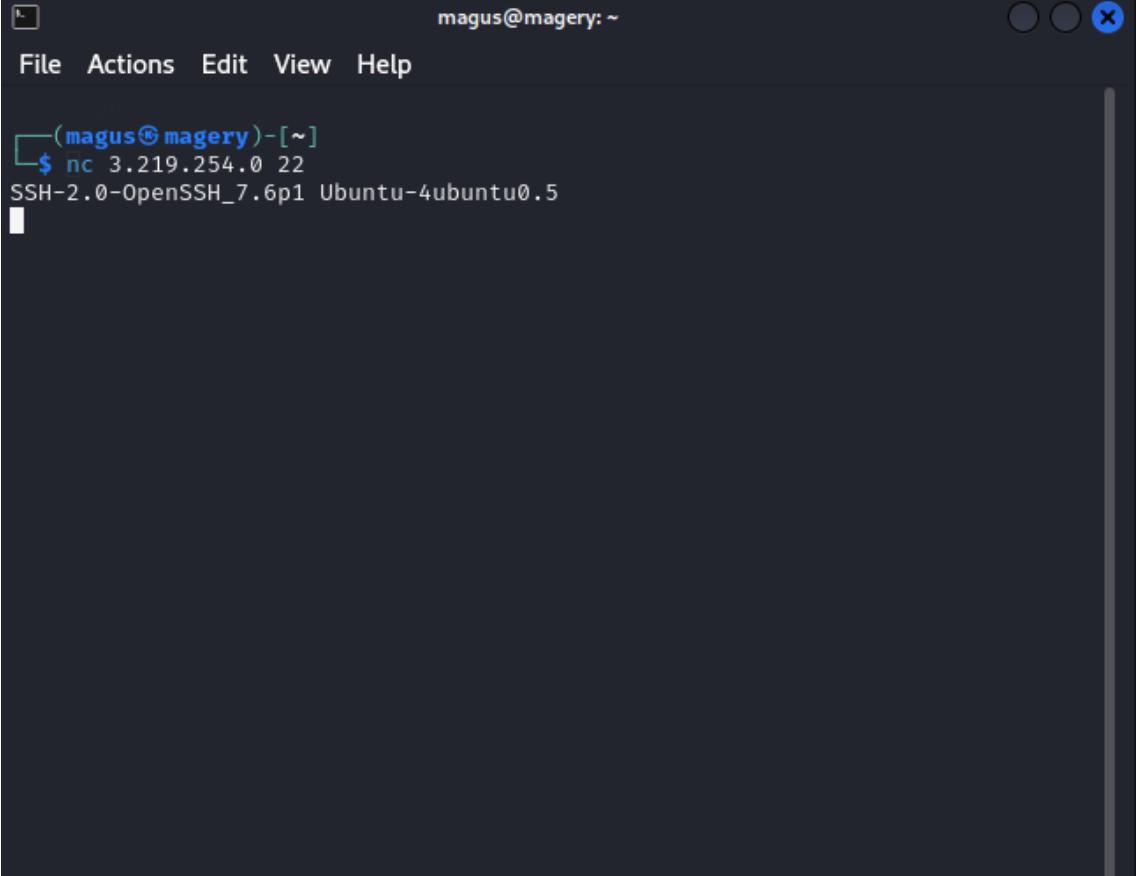
4. Edite o arquivo de configuração do SSH (``/etc/ssh/sshd_config``)

5. Adicione ou modifique a seguinte linha:

```
# Desabilitar ou personalizar o banner SSH
```

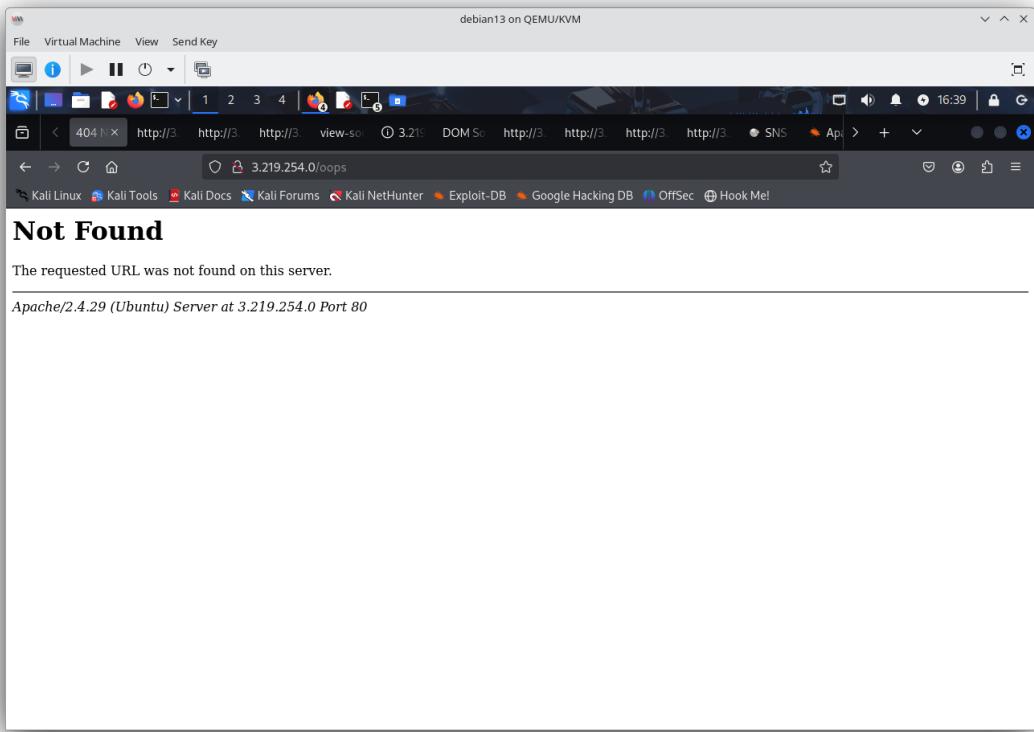
Evidência

Screenshot_20250309_174047.png



A screenshot of a terminal window titled "magus@magery: ~". The window has a dark theme with light-colored text. At the top, there is a menu bar with "File", "Actions", "Edit", "View", and "Help". Below the menu, the terminal prompt shows "(magus@magery)-[~]". The command entered is "\$ nc 3.219.254.0 22", which is a netcat command to connect to port 22 of the IP address 3.219.254.0. The response from the server is "SSH-2.0-OpenSSH_7.6p1 Ubuntu-4ubuntu0.5". The rest of the terminal window is blank.

404.png



report.description

Foi identificada uma vulnerabilidade crítica de Server-Side Request Forgery (SSRF) no arquivo `info.php`. Esta vulnerabilidade permite a um atacante acessar recursos internos do servidor, incluindo arquivos confidenciais contendo credenciais. A gravidade desta falha é considerada **Crítica** (CVSS 10.0).

Detalhes Técnicos

O arquivo `info.php` contém uma implementação insegura que permite a execução de comandos `curl` com entrada de usuário não sanitizada adequadamente. O código verifica apenas se o input contém a string "file://" e se o tamanho do input é menor que 50 caracteres, mas falha em implementar restrições adequadas sobre quais recursos podem ser acessados.

```

<?php
    include '../control/validate.php';
    $type = "";
    $title = "Info";
    $icone = "fas fa-radiation";
    include "../include/menu.php";
    $dominio= $_SERVER['HTTP_HOST'];
    if(md5($dominio) == "421aa90e079fa326b6494f812ad13e79" ||
       md5($dominio) == "f528764d624db129b32c21fbca0cb8d6" )
    {
?
<div class="p-3">
    <div class="col-sm-12 container">
        <form action=<?php echo $_SERVER['PHP_SELF']; ?>" method="GET">
            <h3 class="title-access"> </h3>
            <div class="form-group">
                <input class="form-field field" type="text" id="conteudo" name="conteudo">
            </div>
            <div class="form-group text-center">
                <button class="btn btn-info col-md-6 p-2" name="pesq" value="pesquisar" type="submit"> </button>
            </div>
        </form>
<?php
    $conteudo = filter_input(INPUT_GET, 'conteudo', FILTER_SANITIZE_STRING);
    if($conteudo){
        $termo = 'file://';
        $pos = strpos($conteudo, $termo);
        $tamanho = strlen($conteudo);
        if ($pos === false) {
            echo
                '<div class="alert alert-danger alert-dismissible fade show m-3" role="alert">
                    <strong>Formato inválido</strong>
                </div>';
        } else {
            if($conteudo > 50) {
                echo
                    '<div class="alert alert-danger alert-dismissible fade show m-3" role="alert">
                        <strong>Formato inválido</strong>
                    </div>';
            } else {
                exec('curl -v -s "'.$conteudo.'"', $a);
                foreach ($a as $value) {
                    echo $value;
                }
            }
        }
    }
}

```

Problemas Críticos Identificados

1. **Injeção de Comando:** O código utiliza a função `exec()` diretamente com entrada do usuário, permitindo a execução de comandos arbitrários no servidor.
2. **Verificação de Domínio Insegura:** A aplicação utiliza hashes MD5 hardcoded para verificar o domínio:

```
if(md5($dominio) == "421aa90e079fa326b6494f812ad13e79" ||  
    md5($dominio) == "f528764d624db129b32c21fbca0cb8d6" )
```

- Hash `421aa90e079fa326b6494f812ad13e79` = "localhost"
 - Hash `f528764d624db129b32c21fbca0cb8d6` = "123"
3. **Bypass de Autenticação:** A página possui um mecanismo de autenticação fraco que pode ser contornado via acesso direto.
 4. **Validação Insuficiente:** O código verifica apenas se o input contém "file://" e se tem menos de 50 caracteres, não implementando restrições adequadas sobre os recursos acessíveis.

Método de Exploração

A vulnerabilidade foi explorada usando a seguinte sintaxe:

```
file:///etc/passwd
```

Impacto

Impacto

A exploração desta vulnerabilidade permitiu:

1. `/home/`
`ti/password.txt` , expondo credenciais do usuário "ti".`
2. **Enumeração de Usuários:** A vulnerabilidade possibilitou a enumeração de usuários internos do servidor.
3. **Acesso Não-Autenticado:** O arquivo ``info.php`` pode ser acessado mesmo sem autenticação adequada, através do acesso direto a ``admin/view.php``.

Referências

1. [OWASP - Server Side Request Forgery Prevention Cheat Sheet](#)
2. [CWE-918: Server-Side Request Forgery \(SSRF\)](#)
3. [PortSwigger - Server-side request forgery \(SSRF\)](#)
4. [OWASP - Command Injection](#)
5. [PHP Security - User Data Validation](#)

Remediação

Ações Recomendadas

1. **Remover ou Restringir a Funcionalidade:**
 - Idealmente, remover completamente esta funcionalidade se não for essencial para a operação
 - Caso seja necessária, limitar o acesso apenas a administradores autenticados
2. **Implementar Lista Branca de Recursos:**
 - Definir uma lista explícita de recursos aos quais o aplicativo pode acessar
 - Utilizar um mapeamento indireto para os recursos (não usar o input diretamente)
3. **Corrigir a Validação de Input:**

```
// Definir uma lista de recursos permitidos
$allowed_resources = [
    'internal_resource_1',
    'internal_resource_2'
];

// Verificar se o recurso solicitado está na lista permitida
if (in_array($conteudo, $allowed_resources)) {
    // Processar a requisição de forma segura
} else {
    echo '<div class="alert alert-danger">Recurso não
permitido</div>';
}
```

4. **Eliminar Uso de exec() com Input do Usuário:**

- Nunca usar `exec()`, `shell_exec()` ou funções similares com dados controlados pelo usuário

6. Reestruturar a Verificação de Domínio:

- Não confiar em hashes MD5 fixos para validação de domínio
- Implementar verificação baseada em configuração segura do servidor

7. Implementar Sanitização Adequada:

- Validar e filtrar rigorosamente todas as entradas do usuário
- Implementar escapamento apropriado para comandos de sistema

3. Bypass de Autenticação

Média

report.description

O sistema apresenta falhas críticas no mecanismo de proteção das páginas administrativas, permitindo acesso direto mesmo sem autenticação. O redirecionamento implementado no servidor pode ser facilmente contornado com requisições HTTP diretas.

Impacto

Diversas páginas de acesso restrito estão completamente desprotegidas:

Isto permite que qualquer usuário, mesmo sem credenciais válidas, acesse informações e funcionalidades restritas da aplicação.

Página/Arquivo	Problema de Segurança	Informações Expostas	Método de Acesso
`/admin/home.php`	Bypass de autenticação	Painel administrativo, opções de navegação	Acesso direto via GET ignorando redirecionamento
`/admin/code.php`	Bypass de autenticação	Código: K9801-982, Usuário: XAMARIM	Acesso direto via curl ignorando redirecionamento
`/admin/view.php`	Vulnerabilidade SSRF	Interface para acesso a URLs e arquivos do sistema	Acesso direto e posteriormente exploração
`/admin/doc.php`	Bypass de autenticação	Documentos sensíveis (098-01K.JPG, etc.)	Acesso direto via curl ignorando redirecionamento

Página/Arquivo	Problema de Segurança	Informações Expostas	Método de Acesso
`/admin/consultant.php`	Bypass de autenticação	Código: ZETA9890, Sala: 876K	Acesso direto via curl ignorando redirecionamento

Remediação

1. Implementação de Controle de Sessão Robusto:

- Verificar autenticação em todas as páginas administrativas no início de cada script PHP
- Implementar verificação de sessão antes de processar qualquer requisição

```
<?php
session_start();
if (!isset($_SESSION['authenticated']) ||
    $_SESSION['authenticated'] !== true) {
    header('Location: /login.php');
    exit();
}
// Continuar apenas se autenticado
?>
```

2. Uso de Middleware de Autenticação:

- Implementar uma camada de middleware que valide autenticação para todas as rotas administrativas
- Considerar frameworks como Laravel ou Symfony que oferecem proteção de autenticação robusta

3. Validação de Tokens Anti-CSRF:

- Implementar tokens anti-CSRF para todas as transações administrativas
- Verificar a validade do token em cada requisição administrativa

4. Headers de Segurança:

- Implementar headers de segurança adicionais para prevenir acesso não autorizado:

1. [OWASP Top 10 - Broken Authentication](#)
2. [OWASP Authentication Cheat Sheet](#)
3. [CWE-287: Improper Authentication](#)
4. [CWE-352: Cross-Site Request Forgery \(CSRF\)](#)
5. [PHP Session Security Best Practices](#)

Descobertas

Nenhuma descoberta adicionada ainda.

CONCLUSÃO FINAL

Nenhuma conclusão fornecida.