FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Wesley Cesar Seguin

TÉCNICAS DE HARDENING EM SERVIDORES WEB APACHE

Americana, SP

2017

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Wesley Cesar Seguin

TÉCNICAS DE HARDENING EM SERVIDORES WEB APACHE

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Esp. Ricardo Kiyoshi Batori.

Área de concentração: Segurança da Informação.

Americana, SP.

2017

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

S459t SEGUIN, Wesley Cesar

Técnicas de hardening em servidores web Apache./ Wesley Cesar Seguin. – Americana: 2017.

60f.

Monografia (Curso de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp.Ricardo Kiyoshi Batori

1. Segurança em sistemas de informação I. BATORI, Ricardo Kiyoshi II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.518.5

Wesley Cesar Seguin

TÉCNICAS DE HARDENING EM SERVIDORES WEB APACHE

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área de concentração: Segurança da Informação.

Americana, 27 de Junho de 2017.

Banca Examinadora:

Ricardo Kiyoshi Batori (Presidente)
Especialista
Fatec Americana

Maria Cristina Luz Fraga Moreira Aranha (Membro)
Mestre
Fatec Americana

José Luis Zem (Membro)
Doutor
Fatec Americana

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus por me dar forças para chegar até aqui. Agradeço aos professores Ricardo Kiyoshi Batori e Maria Cristina Aranda pela grande ajuda prestada para a realização deste trabalho. E por fim a minha família pelo grande apoio demonstrado durante a longa jornada de estudos.

DEDICATÓRIA

Aos meus pais que sempre acreditaram em mim e me apoiaram durante a realização deste trabalho.

RESUMO

Nos dias de hoje em que tudo é feito através da Internet, desde uma simples apresentação institucional até grandes operações bancárias existe uma peça fundamental para que isso ocorra, o servidor *web*. Entre os servidores existentes, se destaca o servidor web Apache por ser o mais utilizado em todo o mundo, pois 49,7% de todos os servidores *web* ativos são Apache, que em sua configuração padrão apresenta diversas vulnerabilidades que podem colocar a aplicação e o próprio usuário em risco. Devido a estes problemas, foram criadas diversas técnicas para garantir um maior reforço á segurança desses servidores e informações neles contidas. Este trabalho tem o objetivo de apresentar uma técnica para melhorar o nível de segurança nesses servidores, blindando-os contra possíveis ataques e reduzindo suas vulnerabilidades. A técnica em questão é o *hardening* que através de bloqueios e restrições reforça a segurança do sistema onde o mesmo é aplicado. A aplicação do *hardening* será realizada no Apache 2.4.18 executado em um servidor *GNU/Linux* Ubuntu Server, escolhido por ser gratuito, possuir uma ampla comunidade de apoio e ser de maior familiaridade pelo autor. Para fazer a análise das vulnerabilidades, será utilizada a ferramenta Nikto.

Palavras Chave: Apache; *Hardening*; Servidor.

ABSTRACT

Nowadays everything is done through the Internet, a simple institutional presentation or large banking operations; there is a fundamental part for this to happen: the web server. Among them is the Apache web server is the most used in the world where 49.7% of all active web servers are Apache, which in its default configuration presents several vulnerabilities that can put the application and the user at risk. Because these problems, a number of techniques were created to ensure further enhancement of the security of these servers and information contained therein. This work aims to present a technique to improve the level of security on these servers, shielding them against possible attacks and eliminating their vulnerabilities. The technique in question is the hardening that through locks and restrictions strengthens the security of the system where it is applied. The application of hardening will be performed on Apache 2.4.18 running on a GNU / Linux server Ubuntu Server, chosen to be free, have a broad support community and be more familiar to the author. To do the vulnerability analysis, the Nikto tool will be used.

Keywords: *Apache; Hardening; Server.*

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

Atualmente garantir a segurança dos dados de uma empresa é a atividade que mais preocupa os administradores de um sistema de informação. Com o advento da Internet e dos sistemas corporativos *online* as empresas ficam cada vez mais expostas ao mundo virtual.

O simples fato da empresa operar através da rede ou da própria Internet, possibilita um ganho maior de produtividade pois os recursos são compartilhados, centralizados e permitem acesso a qualquer momento e de qualquer lugar.

O grande problema desta operação em rede é que muitos administradores após instalarem os servidores que serão os responsáveis por permitir que a operação aconteça, o configuram de maneira inapropriada, ou pior, utilizam as configurações padrões da aplicação, as quais deixam brechas de segurança e vulnerabilidades que mais tarde podem ser exploradas, por pessoas mal-intencionadas, a fim de obter acesso ilegal aos dados da corporação, interromper os serviços para prejudicar o andamento dos trabalhos ou até mesmo destruir a aplicação afim de, simplesmente, prejudicar a imagem da corporação junto ao seu público alvo.

Na tentativa de limitar ou eliminar ao máximo estes riscos e garantir uma maior segurança aos sistemas operacionais e às redes de computadores, surgiu uma técnica chamada *hardening*, que, traduzida para o português, significa endurecimento.

O objetivo geral deste trabalho foi implementar técnicas de *hardening* em um servidor Web Apache, apenas alterando configurações de maneira a contribuir para os três principais pilares da segurança da informação, protegendo o mesmo contra possíveis vazamentos de informações sigilosas (confidencialidade), inatividade por seus serviços estarem indisponíveis (disponibilidade) e possíveis ataques por exploração de vulnerabilidades (integridade).

Os objetivos específicos deste trabalho foram: identificar as possíveis vulnerabilidades, solucionar as possíveis vulnerabilidades, demonstrar a solução das vulnerabilidades, testar as soluções e por fim realizar um teste comparativo.

A justificativa para a realização deste trabalho deve-se ao fato do servidor Web Apache ser o mais utilizado em todo o mundo, conforme W3tech (2017) 49,7% dos servidores web ativos são Apache, e em sua configuração padrão existem diversas vulnerabilidades que o torna extremamente perigoso. O servidor web muitas vezes é colocado na rede sem nenhum tipo de proteção, tornando-se assim um dos serviços mais vulneráveis fornecendo informações que podem auxiliar novos ataques. Esses ataques podem causar um grande prejuízo para o proprietário da aplicação em execução no servidor atacado uma vez que a aplicação pode ter dados sigilosos furtados, ficar fora do ar e até mesmo denegrir a imagem da empresa e/ou do proprietário da aplicação.

Após o estudo do Servidor Web Apache e das técnicas de *hardening*, foi formulada a seguinte hipótese: A aplicação das técnicas de *hardening* em um servidor Web Apache, é capaz de mitigar os problemas de segurança da informação presente no mesmo quando utilizadas suas configurações padrões.

A metodologia utilizada é de caráter científico:

> "Traz informações redigidas de tal forma que um investigador competente e suficientemente especializado no mesmo campo poderia, baseando-se exclusivamente nas indicações contidas no texto: 1) repetir as experiências e obter os resultados descritos; 2) repetir as observações e julgar as conclusões do autor; 3) verificar a exatidão das análises e deduções que permitiram ao autor chegar a suas conclusões." (REY, 1987 apud ANDRADE, 1997, p.87).

E para Kerlinger (1973) "Pesquisa científica é uma investigação sistemática, controlada, empírica e crítica de proposições hipotéticas sobre as relações presumidas entre fenômenos naturais".

Esse trabalho utilizou segundo Lakatos e Marconi (1991) a pesquisa bibliográfica trata-se do levantamento, seleção e documentação de toda bibliografia já publicada sobre o assunto que está sendo pesquisado em livros, enciclopédias, revistas, jornais, folhetos, boletins, monografias, teses, dissertações e material

cartográfico. Pretende-se, assim, colocar o pesquisador em contato direto com todo material já escrito sobre o mesmo. Para fazer o embasamento teórico da aplicação, Apache, e da técnica, o *hardening*.

A pesquisa desse trabalho será aplicada, ainda segundo os autores é uma pesquisa que busca apresentar uma solução para o problema com base em uma hipótese. E nesse projeto tem como objetivo auxiliar na busca pela resposta se o *hardening* mitiga as vulnerabilidades do servidor Apache.

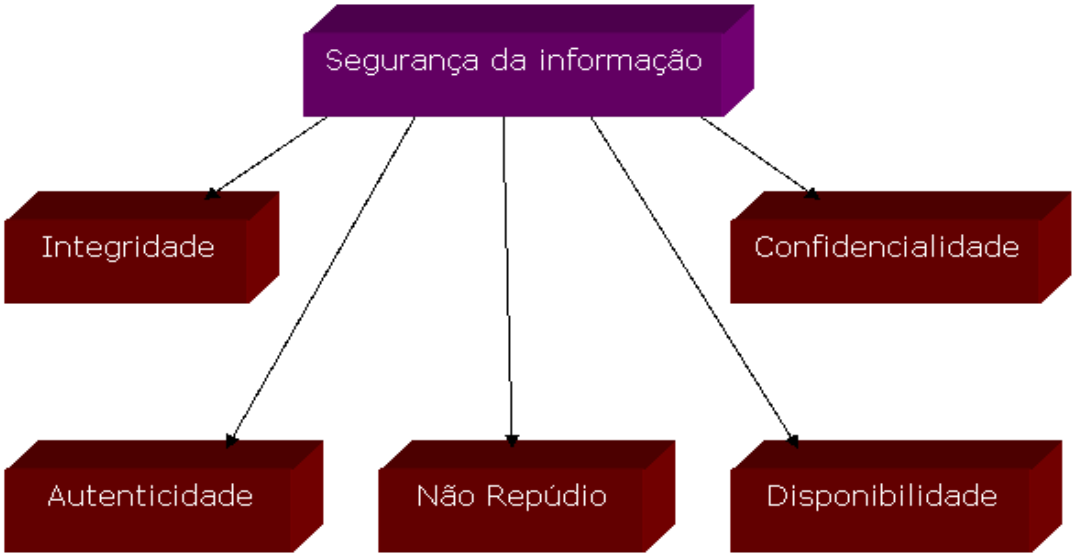
O trabalho foi estruturado em seis capítulos, sendo o primeiro esta Introdução, o segundo sobre Segurança da Informação e seus pilares, que servem para justificar esse trabalho já que visa deixar mais segura a aplicação, o terceiro capítulo vai descrever o que é o servidor Web Apache, o quarto o que é *hardening* e seus tipos, o quinto mostra a implementação prática deste trabalho e por fim o sexto as considerações finais.

2 SEGURANÇA DA INFORMAÇÃO

De acordo com Dantas (2011), a segurança da informação é a proteção da informação contra os tipos de ameaças e vulnerabilidades, para garantir que mesmo em caso de incidentes a continuidade do negócio não seja afetada, minimizando os riscos.

Silva (2004), diz que a Segurança da Informação compreende um conjunto de medidas que visam proteger e preservar informações e sistemas de informações, assegurando-lhes **integridade, disponibilidade, não repúdio, autenticidade e confidencialidade.**

Figura 1: Pilares da Segurança da Informação



Fonte: (SILVA, 2004)

Os pilares da Segurança da Informação, segundo os autores Goodrich e Tamassia (2013) são:

- **Integridade:** A integridade de uma informação tem como objetivo garantir que a mesma não irá sofrer alterações indevidas por usuários não autorizados, bem como impedir que usuários que porventura obtenham acesso a mesma realizem alterações não autorizadas. Para garantir a integridade de uma informação, existem algumas ferramentas como o '*checksum'*, por exemplo.

- **Disponibilidade:** Tem a função de garantir que uma determinada informação estará disponível sempre que seja necessário que a mesma seja consultada, editada ou simplesmente visualizada. Afim de garantir tal disponibilidade na maior parte do tempo, um sistema deve possuir redundância, ou seja, mais que uma fonte com o mesmo conteúdo porem em locais diferentes e assegurar a segurança física/lógica dos locais onde a informação está armazenada.

- **Confidencialidade:** A confidencialidade visa impedir a revelação de informações não autorizadas. Trabalhando com a proteção dos dados, ela garante o acesso a pessoas autorizadas e ao mesmo tempo impede o acesso de usuários não autorizados a esta mesma informação. Manter as informações em sigilo é a essência da segurança da informação e para garantir esse sigilo existem ferramentas como a criptografia, controle de acesso, autenticação entre outras.

- **Autenticidade:** Este termo é definido pelo dicionário Aurélio como: Qualidade do que é autentico, verdadeiro. Então neste caso, assegura que um usuário realmente é que se diz ser, que uma determinada informação é seguramente a original, e não uma cópia adulterada.

- **Não-repúdio:** É a garantia que um usuário não possa negar a autoria de alguma ação, ou seja, em algum local precisa ficar registrado os passos de um usuário dentro de um sistema, para que em caso de uma auditoria ou incidente o responsável por uma ação indevida seja facilmente identificado.

2.1 AMEAÇAS

A ameaça pode ser uma ação ou algum acontecimento que possa prejudicar um ativo, processo ou pessoa por meio de uma vulnerabilidade que irá gerar um impacto.

Sêmola (2003) diz que as ameaças podem ser classificadas quanto a intencionalidade e podem ser classificadas em grupos:

- **Naturais:** Ocorrem devido a fenômenos provocados pela natureza como: enchentes, terremotos, tempestades e incêndios naturais.

- **Involuntárias:** Causadas por acidentes ou por desconhecimento, um exemplo seria quando alguém acidentalmente tropeça em um cabo de energia desligando o mesmo da tomada.

- **Voluntárias:** Ameaças causadas de forma proposital por pessoas mal intencionadas como *crackers*, ladrões, incendiários, etc.

2.2 VULNERABILIDADES

A vulnerabilidade é o ponto fraco de algum ativo, pessoa, ambiente ou sistema de informação.

Para Sêmola (2003) as principais vulnerabilidades de um sistema de informação e de um ambiente são:

- **Naturais:** Onde fenômenos naturais desastrosos, por exemplo: furacões, inundações, terremotos, etc podem colocar em risco os sistemas de informação e todo seu conteúdo.

- **Físicas:** O ambiente onde são armazenadas as informações possui um ponto fraco que causa tal vulnerabilidade.

- **Armazenamento:** A má utilização ou a falta de segurança no local em que as mídias que armazenam as informações são guardadas gera uma vulnerabilidade, pois com a má utilização, pode ocorrer perda das informações armazenadas, pois a mídia pode vir a ser danificada e se a mesma for guardada em um local inseguro ela pode ser roubada ou destruída por algum agente de forma natural ou intencional.

- **Comunicação:** Esta vulnerabilidade se relaciona com o tráfego de informações, que podem ser realizados por meios guiados (cabos) ou não guiados (*wireless*). Seja qual for o meio escolhido para a transmissão dos dados o mesmo deve garantir que a informação alcance seu destino sem que a mesma seja interceptada ou bloqueada durante a transmissão.

- **Humanas:** Podem ser causadas por atos intencionais ou acidentais. As principais vulnerabilidades desta grupo são: uso de senhas fáceis de serem adivinhadas (senhas conhecidas como fracas), compartilhamento de dados de acesso a sistemas e locais, falta de um treinamento adequado para o usuário, o não conhecimento das boas práticas de segurança da informação e principalmente funcionários descontentes que podem ser uma ameaça a segurança da empresa.

3 APACHE

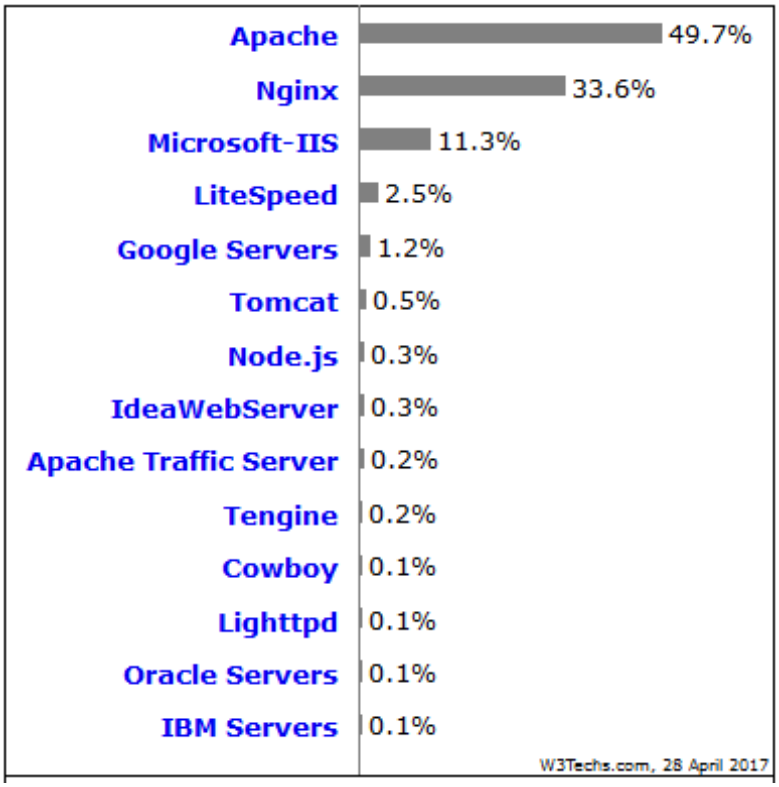
Para Alecrim (2006), um servidor web é quando uma pessoa acessa um site, comenta algo em algum fórum online ou mesmo envia um *email* através de um formulário *web*, por trás de tais endereços existe um servidor *web* para disponibilizar essas páginas e todos os demais recursos que precisem ser utilizados.

Ainda segundo o autor ainda reforça que um servidor *Web* se trata de um computador preparado para processar solicitações HTTP (*Hyper-Text Transfer Protocol*), o protocolo padrão de transferência de conteúdo da *Web*. O Apache também trabalha com outros protocolos como o HTTPS (versão segura do HTTP), por exemplo.

A definição dada por Laurie e Laurie (1999) para o servidor Web Apache é de: um programa executado dentro de um sistema operacional multitarefa, sendo chamado de *httpd* nos ambientes Linux/Unix; no ambiente Windows é denominado de *apache.exe* e normalmente executado em segundo.

Os mesmos autores em outra edição (2002, p. 1), o definiram como um servidor *Web* dominante na Internet atualmente, ocupando um lugar chave na infraestrutura da mesma.

Figura 2: Estatísticas de Uso do Servidor Web Apache



Fonte: W3Techs[1] (2017)

Baseado em dados obtidos de uma pesquisa no site w3techs[1], o Apache é o servidor *web* mais utilizado no mundo ocupando em abril de 2017, 49,7% do uso da Web. E para Alecrim (2006), tal colocação deve-se ao fato de sua excelente performance, segurança e compatibilidade com várias plataformas e todos os seus recursos.

3.1 CARACTERÍSTICAS DO APACHE

Algumas características do Apache são:

> O Apache Server é um software livre, o que significa que qualquer um pode estudar ou alterar seu código-fonte, além de poder utilizá-lo gratuitamente. É graças a essa característica que o software foi (e continua sendo) melhorado ao passar dos anos. Graças ao trabalho muitas vezes voluntário

[1] Disponível em: <https://w3techs.com/technologies/overview/web_server/all>. Acesso em: 01 mai. 2017

> de vários desenvolvedores, o Apache continua sendo o servidor Web mais usado no mundo. (ALECRIM, 2006).

De acordo com Alecrim (2006), além de estar disponível para o Linux, o Apache também pode ser instalado e executado em outras plataformas, tornando-o uma ótima opção para computadores obsoletos.

Ainda segundo Alecrim (2006), o Apache é capaz de interpretar e executar códigos HTML, PHP, Perl, Shell Script e até o ASP da Microsoft. Mas sua utilização mais conhecida é com o PHP utilizando o banco de dados MySQL.

A respeito dos requisitos de hardware Alecrim (2006) ressalta que são mínimos, podendo ser executado em um PC Pentium com 64 MB de memória RAM em pequenos ambientes corporativos.

Para Laurie e Laurie (1999), independente do sistema operacional onde o Servidor Web Apache for instalado, os autores afirmam que o diretório de uma instalação deve conter quatro subdiretórios sendo:

- **Conf:** Este subdiretório contém os arquivos de configuração, de todos os arquivos deste subdiretório o *httpd.conf* é o mais importante.

- **Htdocs:** Os scripts HTML estão armazenados neste subdiretório, e serão enviados para a máquina cliente quando ocorrer uma solicitação, todo o conteúdo armazenado neste subdiretório, também conhecido como *web-space*, é acessível para qualquer usuário e consequentemente a segurança pode ser colocada em risco quando gravados dados sigilosos neste subdiretório.

- **Logs:** Este subdiretório armazena as informações de acesso ao sistema e erros.

- **Cgi-bin:** Este subdiretório armazena os *scripts cgi* os *shell* scripts desenvolvidos pelo webmaster que podem ser executados pelo Web Apache.

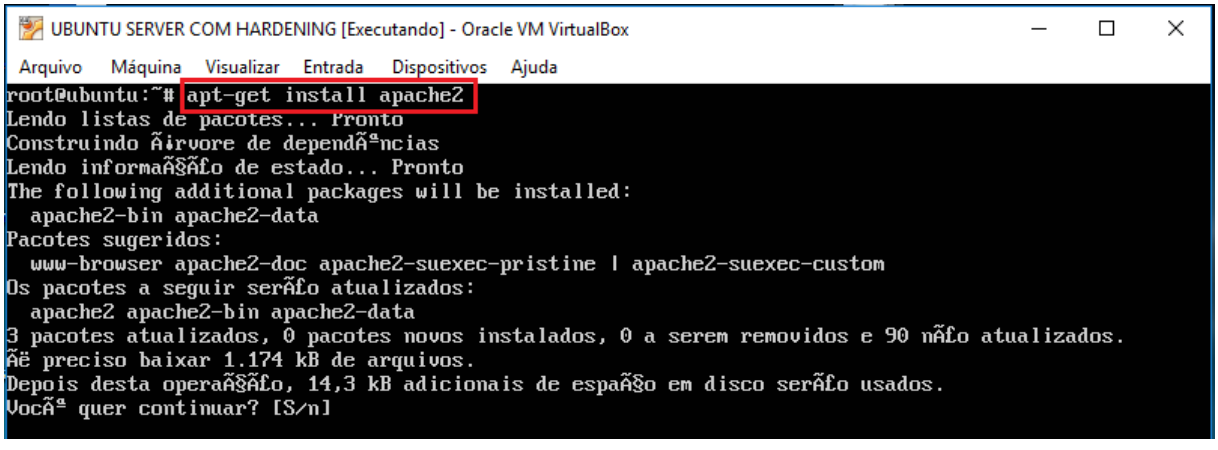
Por questões de segurança não é recomendado que este subdiretório esteja localizado no *web-space.*

Os autores Laurie e Laurie (1999) dizem ainda que, no estado ocioso, o Servidor Web Apache não executa tarefa alguma, mas continua a ouvir os endereços de IP e portas TCP especificadas no seu arquivo de configuração, httpd.conf. Quando há uma requisição de HTTP em uma porta válida no Servidor Web Apache existe um analise de cabeçalho, são aplicadas as regras encontradas no arquivo de configuração e são executadas as ações de aceite ou recusa da solicitação.

3.2 INSTALANDO O APACHE

De acordo com informações encontradas na documentação oficial do Ubuntu, para instalar o Apache versão 2 precisa-se estar autenticado como *root* e no terminal de comando digitar: **apt-get install apache2**. Após pressionar a tecla **enter**, o Apache será instalado automaticamente no sistema e estará pronto para ser configurado.

Figura 3: Instalando o servidor Web Apache



Fonte: Próprio autor**.**

Após a instalação é necessário checar periodicamente se o Apache está seguro ou não. Tal cheque, se realizado manualmente pode vir a ser muito complicado, mas

facilitar o processo, existem algumas ferramentas para Unix com o objetivo de testar a segurança de seu servidor. Para Mobily (2004, p.15), são elas:

- **Nessus:** Esta é provavelmente a mais conhecida e poderosa ferramenta de análise conhecida nos dias de hoje.

- **Nikto:** Um *scanner* que se concentra exclusivamente em servidores Web.

3.3 TIPOS DE ATAQUES

Nem todos os ataques ao Apache são a mesma coisa, de modo geral para Mobily (2004, p.14), eles podem ser divididos em duas categorias:

- **Remote Shell**: consiste em habilitar um *shell* para executar comandos arbitrários em um computador remoto. Para um cracker, obter um *shel*l remoto é apenas o início, pois partir deste ponto o mesmo pode executar comandos como um usuário normal. Isto é extremamente perigoso uma vez que o atacante pode conseguir acesso total ao servidor.

- **Ataque de negação de serviço (*Denial Of Service*):** Esse ataque visa abater seu servidor, ou seja, fazer o mesmo parar de responder e sair fora do ar.

Entre os dois ataques, DOS provavelmente é o menos danoso que o Shell remoto, mas é mais comum e pode causar maiores problemas.

3.4 VULNERABILIDADES DO APACHE

Serão apenas algumas as mencionadas neste tópico, uma vez que novos problemas podem ser encontrados a cada dia. Para Mobily (2004, p. 44), são elas:

- Codificação de vulnerabilidade de Fragmentação: O Apache falha ao calcular o tamanho do *buffer* necessário para armazenar a informação fragmentada. Portanto, um usuário malicioso pode fazer uma requisição que corrompe a memória do servidor e em alguns casos executar um código aleatório no sistema.

- Requisições podem causar listagem do diretório a ser exibido: Explorando esta vulnerabilidade, um atacante pode a lista de todos os arquivos encontrados em um diretório, mesmo que esteja contido um *index.html,* a listagem pode ocorrer se um número absurdo de barras (/) for enviado ao Apache durante a requisição.

- Problema de estouro de *buffer*: Em outubro de 2002 surgia um novo verme (*worm*) da Internet. O verme é baseado em uma vulnerabilidade no OpenSSL, onde o problema é encontrado no processo de confirmação da conexão, ao enviar uma chave deformada para o servidor pode causar um estouro no *buffer* e aceitar que um atacante executar códigos arbitrários no maquina alvo do ataque.

- O ‘problema maior’: Usando as vulnerabilidades do SSL acima apresentadas, um atacante pode escrever um programa que possibilite:

    - Atacar o servidor Web.

    - Instalar-se no servidor usando a vulnerabilidade do SSL.

    - Tornar-se parte de um rede ponto-a-ponto para executar ataques DDOS
      (*Distributed Denial Of Service*).

4 HARDENING

Skoudis (2012), define *hardening* como um processo para um mapeamento das ameaças afim de atenuar os riscos e executar atividades corretivas, focando na infraestrutura e com o objetivo de preparar um sistema ou ambiente para enfrentar possíveis ataques ou violação de segurança.

As principais características desta técnica são: Eliminação de riscos afim de assegurar a Confiabilidade, Integridade e Disponibilidade dos Sistemas Operacionais; Reduzir os riscos de haver uma invasão; Desativar serviços desnecessários no Sistema Operacional; Criar as chamadas senhas fortes e preparar o ambiente caso venha a ocorrer um ataque.

Júlio, Reis e Verbena (2011) acreditam que *hardening* se trata de uma técnica genérica, uma vez que pode ser aplicada em qualquer Sistema Operacional. No caso do *GNU/Linux* consegue obter um alto nível de segurança ao implementar a técnica. Ao aplica-las, deve-se considerar três variáveis: Segurança, Risco e Flexibilidade.

Cabe ao profissional de Segurança da Informação que irá implementar a técnica de *hardening*, mensurar de forma correta essas variáveis para obter o maior nível de segurança possível sem afetar a produtividade do Sistema.

Domingos (2006) fala que, *hardening* são ajustes finos realizados em um sistema após a instalação do Sistema Operacional e Melo (2014) diz se tratar da proteção do sistema por meio da redução de suas possíveis vulnerabilidades.

4.1 TIPOS DE HARDENING

De acordo com o autor Turnbull (2005), podem ser considerados dois tipos de *hardening*, são eles:

- *Hardening* de Sistema;
- *Hardening* de Serviço;

Ainda segundo o autor, o *hardening* de sistemas, que diz respeito ao Sistema Operacional em si é dividido em várias etapas e configurações:

- Segurança no sistema de arquivos;
- Arquivos com *suid bit* ativos;
- Segurança no terminal;
- Gerenciamento de privilégios;
- Procura por senhas fracas;
- *Check-lists* no sistema de arquivos.

4.1.1 SEGURANÇA NO SISTEMA DE ARQUIVOS

Melo (2014), diz que as boas práticas de instalações de um Sistema Operacional Linux, aconselham particionar e colocar os principais diretórios em partições separadas.

Com isso é possível proporcionar maior segurança uma vez que cada partição terá sua tabela separada e uma regra de montagem exclusiva para a mesma.

Ainda segundo o autor, a segurança de um particionamento não se resume a isso. É necessário que antes de instalar um sistema Linux, o administrador leia a documentação para conhecer os recursos de montagem oferecidos.

Exemplo de montagem: **mount -o remount,rw,noexec /home**

Acima pode-se observar que a partição “/home” foi montada utilizado a opção “**noexec**”, sendo assim nesta partição esta impossibilitada a execução de qualquer arquivo binário ou executável. Melo (2014) acredita que, essa opção pode ser aplicada a todos os diretórios que contenham binários necessários as funções do sistema.

mount -o remount,rw,nosuid /home

Acima pode-se observar que a partição “/home” foi montada utilizando a opção **nosuid**, assim binários com permissão de *suid bit* não terão efeito na partição a qual estão definidos. Abaixo encontra-se uma explicação mais detalhada sobre *suid bit.*

4.1.2 ARQUIVOS COM *SUID BIT* ATIVOS

Suid bits (*Set owner User ID up on execution*), é um tipo especial de permissão dado a um arquivo. Para Melo (2014), essa permissão possibilita que um determinado binário possa ser executado por outros usuários com o mesmo direito do “usuário proprietário”.

Mas nem todos os usuários precisam dessa permissão, uma vez que um usuário comum não precisa executar tarefas administrativas. Baseado nessa ideia, a CIS SECURITY (Center for Internet Security), criada em 2000 com a missão de “melhorar a prontidão da segurança cibernética e de resposta das entidades públicas e privadas, com um compromisso com a excelência através da colaboração”, recomenda a remoção da permissão *suid bit* dos arquivos binários que a possuem.

Antes de remover as permissões, é necessário encontrar no sistema quais são esses arquivos. Para esta tarefa basta usar o comando **find** com o parâmetro “4000”, que representa a permissão *suid*. Finalmente para remover essas permissões basta usar o comando **chmod –s –R /**, onde o **chmod** altera a permissão do arquivo, **-s** retira a permissão de *suid bit* e **-R /** remove recursivamente do **/** para baixo.

4.1.3 SEGURANÇA NO TERMINAL

Para Turnbull (2005, p. 15), no mundo Linux, através do terminal é permitido que o administrador execute comandos, tarefas e funções que o mesmo não conseguiria de outros locais. Para Melo (2014), existem vários tipos de vulnerabilidades que permitem ameaças através do terminal.

A seguir encontra-se algumas recomendações para aumentar a segurança dos terminais Linux:

- Desativar o uso de CTRL + ALT + DEL

  Melo (2014) fala que, deixar o “CTRL+ALT+DEL” desabilitado pode ser uma boa ideia, pois um administrador distraído pode se confundir com um terminal Windows e acabar acidentalmente reiniciando o servidor Linux desnecessariamente.

- Limitar uso de terminais em modo texto

  Ainda segundo o autor, em alguns casos é interessante deixar habilitado o *login* em todos os terminais em modo texto, mas quando é necessário impedir o *login* nesses terminais, podemos editar o arquivo encontrado em: **/etc/inittab** e adicionar a seguinte linha: **4:23:respawn:/sbin/getty 38400 tty4,** onde

4 representa o terminal de número 4.

- Usar a variável TMOUT

  Segundo Domingos (2008), a variável TMOUT controla quanto tempo leva para uma sessão ser encerrada por tempo de inatividade.
  A definição de um valor padrão para a variável TMOUT pode ser feita dentro do arquivo global de variáveis do sistema **/etc/profile**, basta utilizar o seguinte comando: **echo "export TMOUT=30" >> /etc/profile**, neste caso a sessão se encerrara após 30 segundos de inatividade.

- Usar o programa Vlock (*Virtual console lock program*)

  Melo (2014) diz que, o Vlock utilizado com o parâmetro "-a", bloqueia o uso de todos os terminais, liberando os mesmos somente após a autenticação do mesmo usuário que os travou.

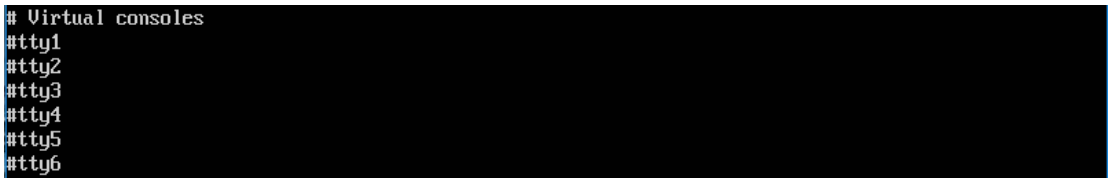
4.1.4 GERENCIAMENTO DE PRIVILÉGIOS

Melo (2014) diz que o usuário mais comum em sistemas Unix/Linux é o root, portanto é o mais visado no momento de um ataque ao sistema, tanto por crackers quanto por usuários mal-intencionados. Afim de evitar tais ataques recomenda-se:

- Bloquear *login* do usuário *root*.

  Segundo Melo (2014), é recomendável deixar desabilitado o login do usuário root em todos os terminais, forçando o *login* com usuários normais e apenas quando necessário de tornar root com o comando "su". Uma das maneiras pelas quais podemos bloquear o login do root é editar o arquivo "/etc/ securetty" e comentar as seguintes linhas:

Figura 4: Terminais



Fonte: Próprio autor.

Onde:
tty1, tty2, tty3, tty4, tty5 e tty6, representam os terminais.

- Determinar datas de expiração para as contas de usuário.

  Melo (2014) garante que em um sistema com uma quantidade elevada de usuários se faz importante determinar o tempo de validade das contas de usuário, pois se os usuários as utilizam todos os dias quando for solicitada uma troca de senha o mesmo efetuará, e caso a conta esteja inativa a mesma irá expirar, dando a segurança que ela não será utilizada por pessoas mal intencionadas.
  Para fazer essa configuração, é utilizado o comando **chage**.

  Exemplo: **chage -M 30 -W 5 -I 2 nome_conta**

  Onde:
  -M é o tempo (em dias) máximo de validade da conta;
  -W é o tempo (em dias) de aviso que a mesma irá expirar;

  -I é o tempo (em dias) antes da conta ser desativada.

- Remover *shells* válidas de usuários que não precisam delas.

  Para Melo (2014), usuários que não precisam usar a *shell* – usuários de sistema ou cliente de serviço – exceto aqueles que precisam fazer acesso FTP, SSH e/ou TELNET, não precisam tê-la ativada em suas contas.

4.1.5 UTILIZAÇÃO DO PAM *(PLUGGABLE AUTHENTICATION MODULE*)

PAM para o autor Turnbull (2005, p. 46) é um sistema projetado pela Sun Mycrosystems para oferecer um *framework* de autenticação que é fortemente utilizado e desenvolvido no mundo Linux, existindo um grande número de módulos disponíveis para serem utilizados. E como explica Melo (2014), um módulo PAM pode ser uma ou todas as quatro interfaces possíveis, são elas: *account, auth, password* e *session*.

Onde as interfaces:

- *Account*: verifica de uma conta tem permissão para acessar o sistema.
- *Auth*: autentica os usuários.
- *Password*: verifica e define a autenticação de senha
- *Session*: configura e gerencia as sessões do usuário.

Mas para o uso seguro do PAM, são necessários alguns ajustes e configurações:

- Limitar horário de *logins*;
- Limitar quantidade de *logins* por usuários;
- Definir tamanhos mínimos de senhas;
- Limitar quais usuários podem ter acesso *root*.

4.1.6 BUSCAR POR SENHAS DE BAIXA SEGURANÇA

De acordo com Melo (2014), as senhas dos usuários não devem ser conhecidas nem mesmo pelo administrador do sistema. Quando um usuário é criado é atribuído ao mesmo uma senha padrão que deverá ser trocada no primeiro *login.*

Quando não é definido um critério de senha forte, um usuário pode criar uma senha “fraca” utilizando sequências numéricas obvias, datas de nascimento e o próprio nome de usuário como parte da senha. Senhas desse tipo são facilmente “quebradas” por programas como o *John the Ripper*, por exemplo.

4.1.7 *CHECKLIST* NOS SERVIÇOS DO SISTEMA

Segundo Melo (2014), quando um Sistema Operacional Linux é instalado alguns serviços já serão instalados por padrão e após a instalação já estarão funcionando. Mas é necessário rever as configurações desses serviços antes de colocá-los em produção, uma vez que uma configuração padrão pode colocar em risco a segurança do sistema.

Abaixo encontra-se uma lista de ferramentas que auxiliam a revisão das configurações desses serviços, as quais estão apresentadas no manual do comando no terminal Linux:

- **Netstat**

  Através do comando **man netstat** no terminal de comando da distribuição Ubuntu, o Netstat mostra as conexões de rede, tabelas de roteamento, estatísticas de interface e conexões mascaradas. Quando utilizado sem nenhum argumento, seu comportamento padrão será mostrar o estado das conexões de rede através da listagem dos sockets abertos.

- **Nmap**

  O software **Nmap** pode ser obtido através do comando **sudo apt-get install nmap**, e segundo sem manual encontrado através do comando **man nmap** no terminal de comando da distribuição Ubuntu, o Nmap é uma ferramenta de exploração e auditoria de segurança de Sistemas Operacionais que faz o escaneamento de todas as portas do sistema.

- **Hping**

O software Hping3 pode ser obtido através do comando **sudo apt-get install hping3**, e segundo o seu manual encontrado através do comando **man hping3** no terminal de comando da distribuição Ubuntu, o hping3 é uma ferramenta de rede capaz de enviar pacotes com tamanhos personalizados e mostrar a resposta assim como o programa ping faz através do protocolo ICMP. Mas usando o Hping3, você pode testar regras de firewall, realizar um escaneamento de portas avançado, testar a performance da rede utilizando diferentes protocolos e tamanhos de pacotes.

- **lsof**

O software lsof pode ser obtido através do comando **sudo apt-get install lsof**, e segundo o seu manual encontrado através do comando **man lsof** no terminal de comando da distribuição Ubuntu, o lsof utilizado sem nenhum parâmetro é utilizado para mostrar os arquivos que estão abertos no sistema.

4.2 *HARDENING* DE SERVIÇO

Melo (2014) diz que o *hardening* de serviço, deve ter o foco em eliminar ou modificar os recursos disponíveis por padrão no serviço, uma vez que o padrão e conhecido e comumente usados por atacantes.

Hardening do serviço SSH:

- Ainda de acordo com o autor, uma possibilidade de controle seria o uso de módulos do PAM, tratando-se do SSH existem vários parâmetros que podem somar ainda mais na segurança do serviço. Abaixo será apresentada baseada no mesmo autor, uma lista de melhores práticas para tornar o serviço mais seguro:

- Alterar a porta padrão do serviço (no caso do SSH a porta 22), para uma porta alta não utilizada pelo sistema, para encontrar as portas abertas pode-se utilizar o *software* Nmap;
- Configurar o serviço para escutar somente o IP (*Internet Protocol*) definido como permitido no arquivo de configuração;
- Proibir o *login* como *root*;
- Utilizar somente o protocolo na versão 2;
- Liberar acesso somente a usuários específicos;
- Não permitir usuários sem senha;
- Possuir controle contra os ataques de força bruta.

5 EXPERIMENTO REALIZADO

Para o experimento realizado foi criado um ambiente contendo três maquinas virtuais, onde duas serão destinadas para teste das técnicas de *hardening* e uma será a que realizará os testes (ataques) (Figura 5).

Utilizando-se destas máquinas serão criados dois cenários, aos quais um será sem a aplicação das técnicas e o outro com as técnicas de *hardening* devidamente aplicadas. No primeiro momento a fim de esclarecimento será demonstrada as configurações das máquinas e suas características e por fim serão realizados os testes para obter os resultados que serão apresentados.

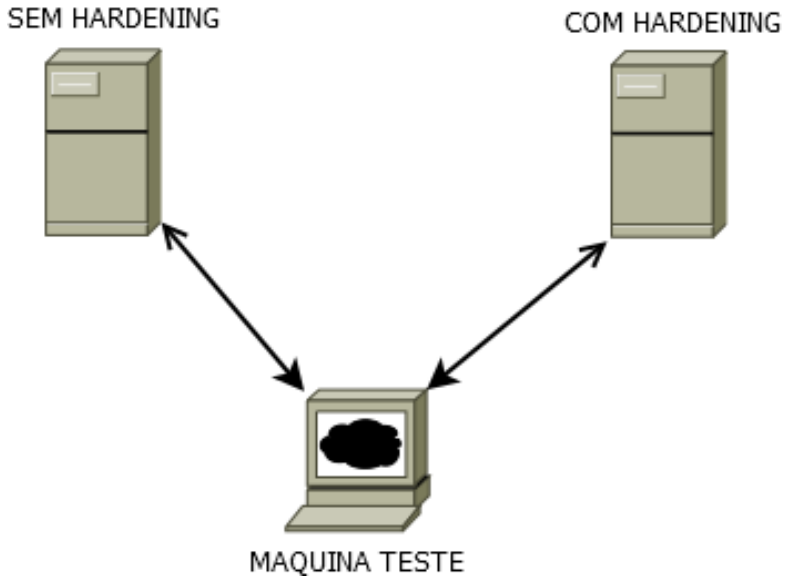
5.1 AMBIENTE

Para a realização do experimento foi utilizado um ambiente contendo três máquinas virtuais, onde a máquina chamada UBUNTU SERVER SEM *HARDENING*, é a máquina que contém uma instalação padrão do Servidor Web Apache (Figura 6). A denominada UBUNTU SERVER COM *HARDENING* por sua vez sofreu a aplicação das técnicas de *hardening* em sua instalação do Servidor Web Apache (Figura 7) e por fim, a máquina MAQUINA TESTE foi a responsável por executar os testes aqui demonstrados (Figura 8). As especificações técnicas tanto de hardware quanto de software da máquina hospedeiro são: Processador Intel i5 de 2.60Ghz, Memória RAM 8GB DDR3 e 500GB de disco rígido. Para software é utilizado o sistema operacional Windows 10 64 Bits, onde foi instalado o Oracle VM Virtual Box versão 5.0.24 r108355, utilizado para a virtualização e criação dos ambientes (Figura 9).

Para as máquinas virtuais foram configurados dois servidores, sendo eles com sistema operacional *GNU/Linux* Ubuntu Server 64 bits versão 14.04.2 com 1GB de memória RAM. Para a máquina virtual de teste, foi utilizado o sistema operacional

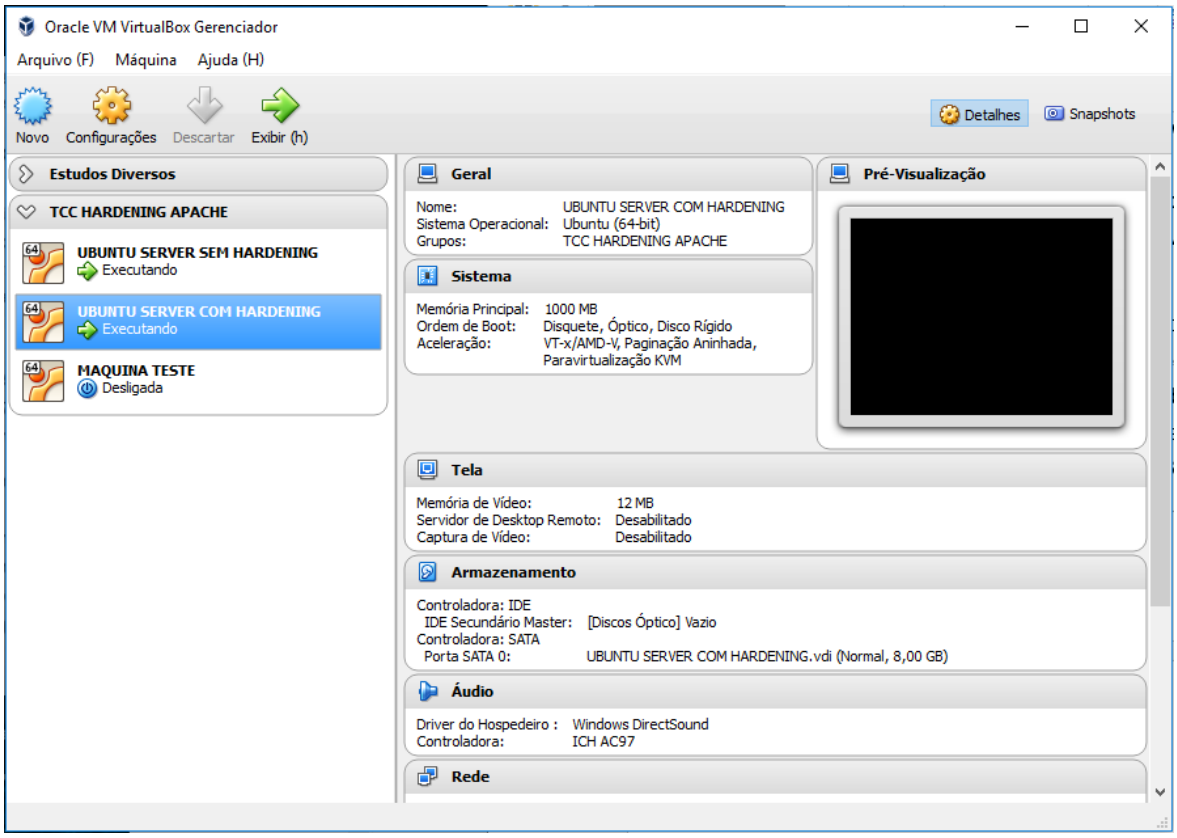
LXLE versão 16.04.2 de 64 bits, por conter interface gráfica a mesma foi atribuído 2GB de memória RAM.

Figura 5: **Cenário do Experimento**



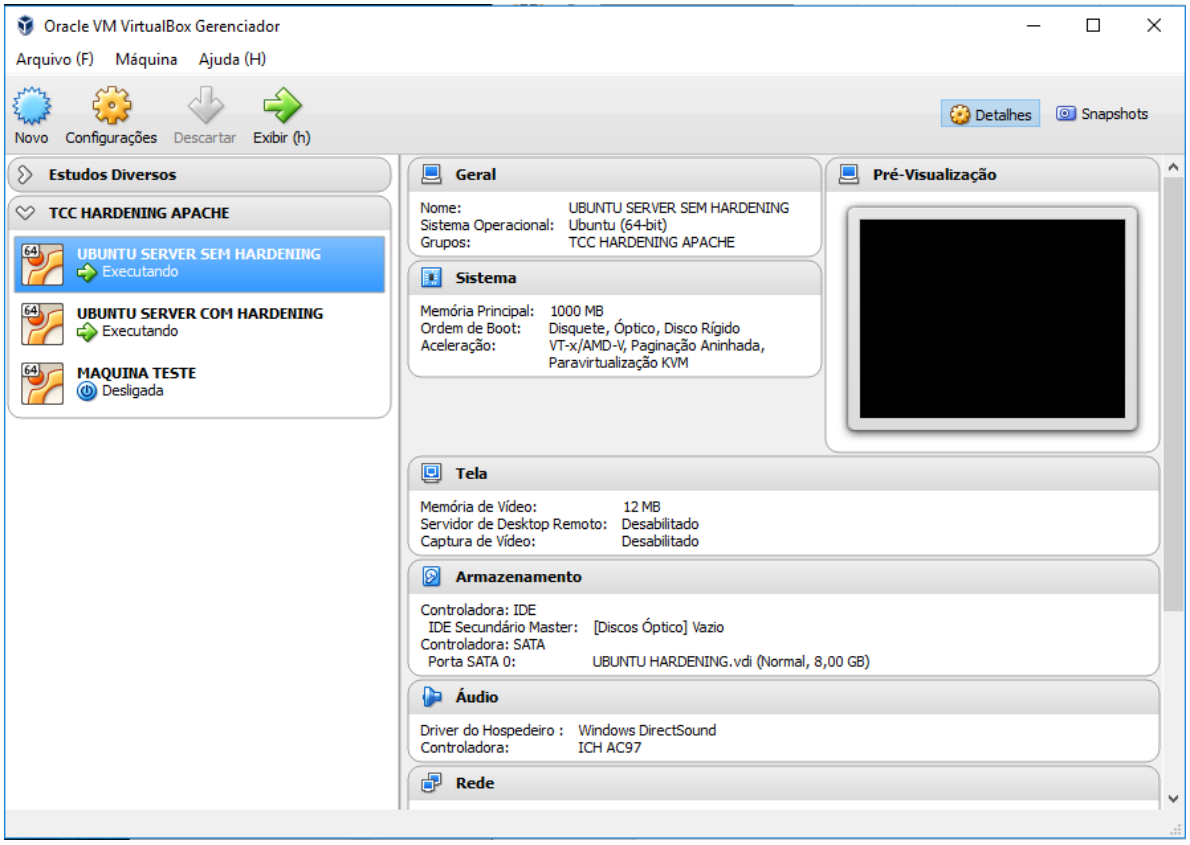
Fonte: Próprio autor.

Figura 6: **Servidor com *Hardening***



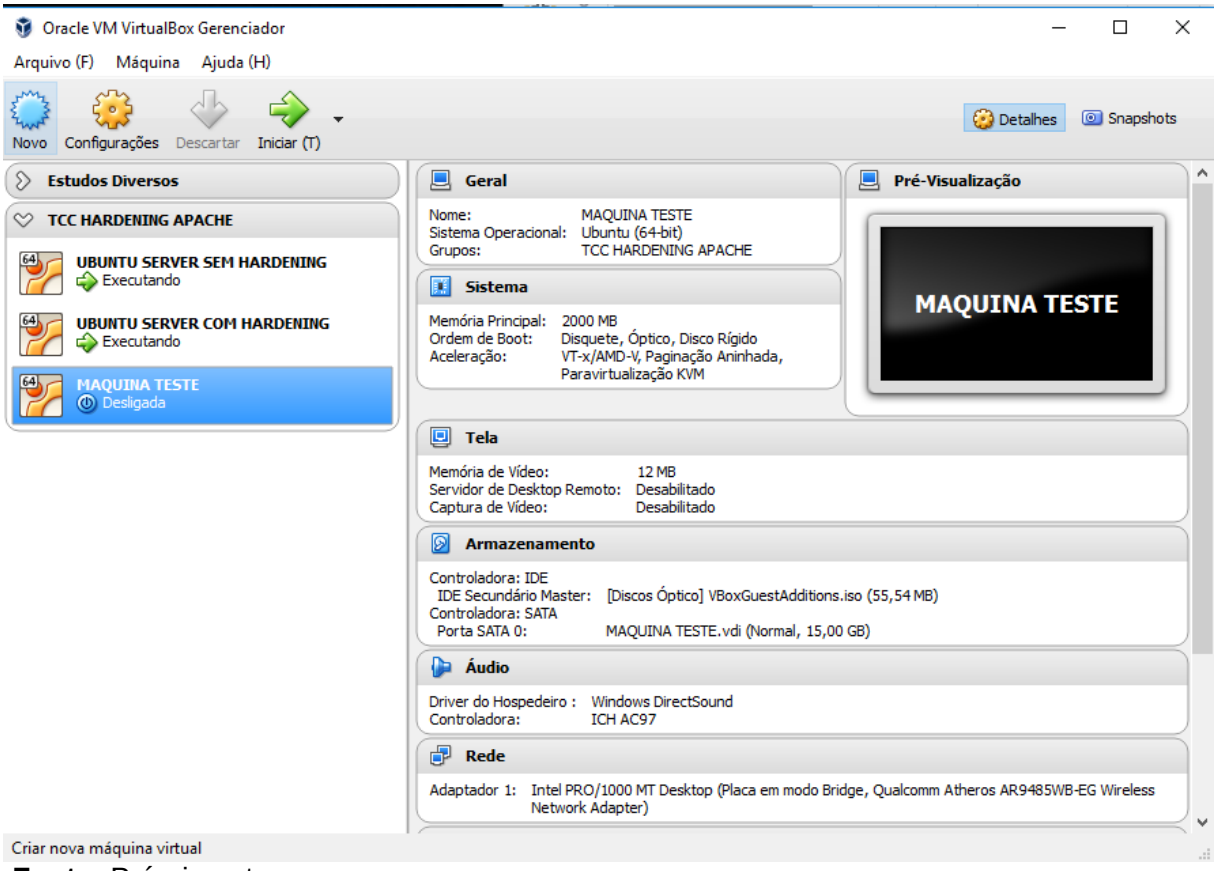
Fonte: Próprio autor.

Figura 7: Servidor sem *Hardening*



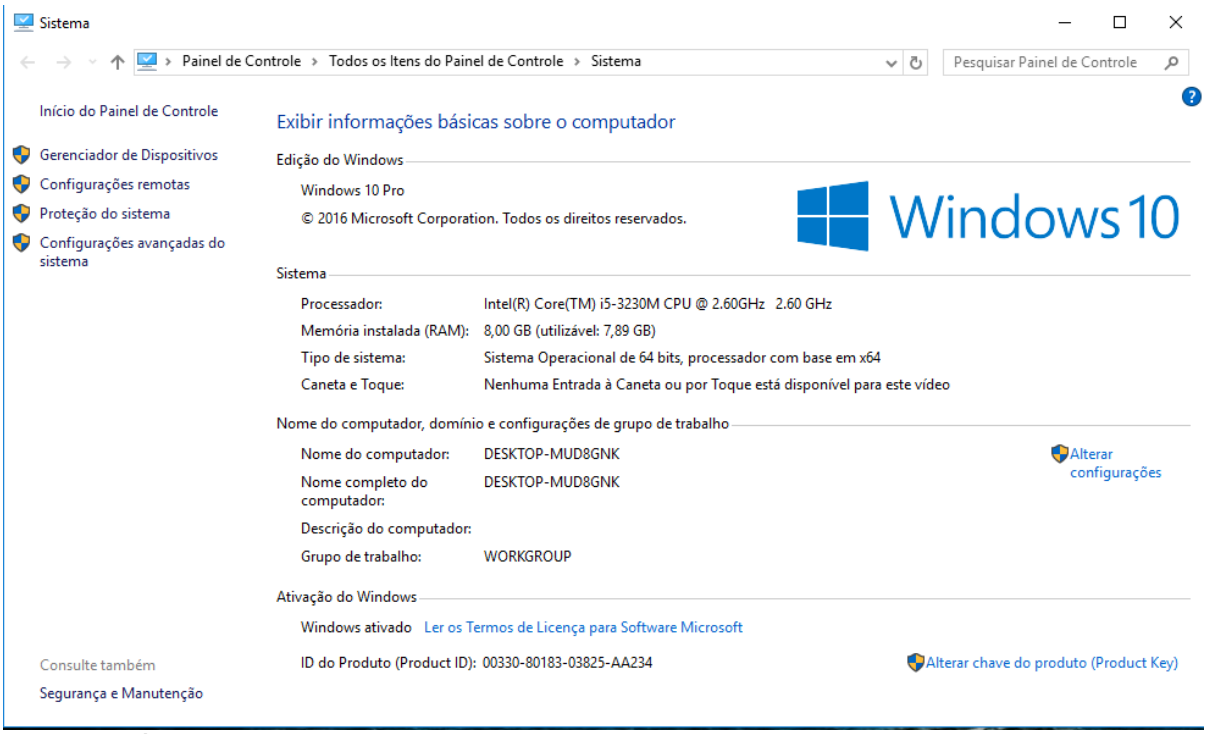
Fonte: Próprio autor.

Figura 8: Máquina teste



Fonte: Próprio autor.

Figura 9: Hospedeiro



Fonte: Próprio autor.

5.2 DESCRIÇÃO DO EXPERIMENTO

O experimento prático foi dividido em 6 etapas, sendo elas:

- Vazamento de informações;
- Autorização;
- Web Application Security;
- SSL;
- Mod Security;
- Configurações gerais.

Inicialmente em ambas as instalações foi criado um subdiretório chamado '*hardening*' em */var/www/html*, e foi também criado um arquivo chamado de texto chamado **arquivo.txt**.

5.2.1 VAZAMENTO DE INFORMAÇÕES

Em sua instalação padrão a configuração do Servidor Web Apache permite a divulgação de informações sigilosas, que podem ser utilizadas para executar um ataque ao mesmo. Identificar e não divulgar tais informações é uma das tarefas mais importantes para o administrador.

Para auxiliar no processo de identificação dessas vulnerabilidades, foi instalada uma extensão no *web browser* Mozilla Firefox chamada **Firebug**.

Figura 10: Obtendo o *Firebug*

Fonte: Próprio autor.

5.2.1.1 REMOÇÃO DO BANNER DE VERSÃO DO SERVIDOR

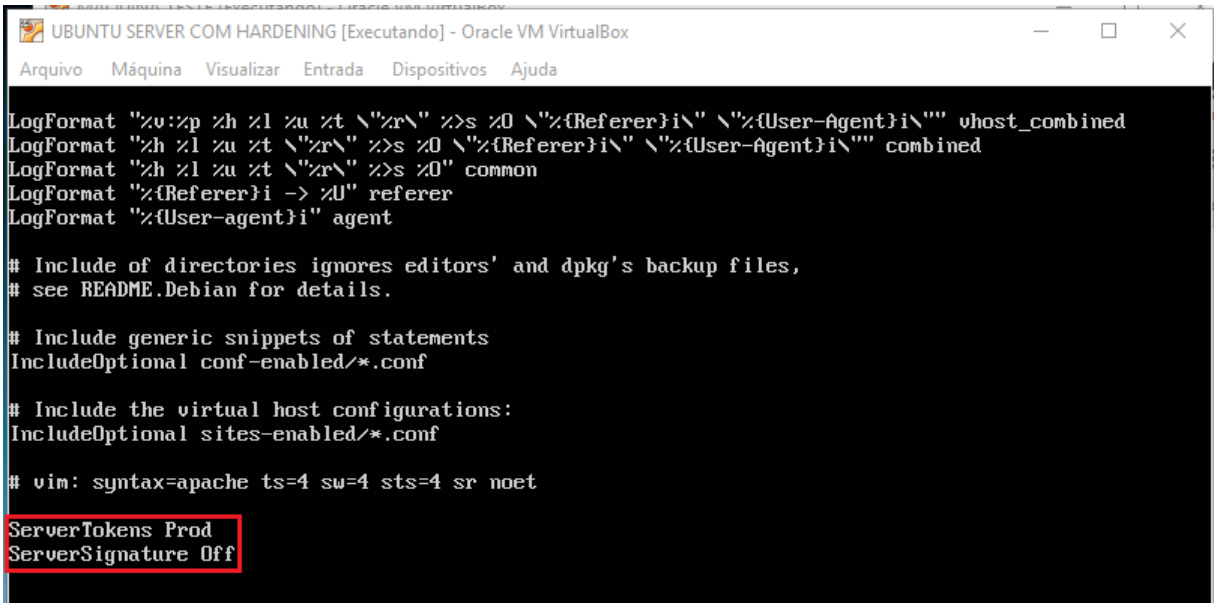
Com o auxílio da máquina de teste, foi realizado o primeiro acesso ao servidor de teste que não contém as técnicas aplicadas, e pode-se notar que foram apresentadas algumas informações a respeito do servidor como o Sistema Operacional e sobre o Apache foi exibida a sua versão. O fato de expor a versão do *software* utilizado ajuda um possível *hacker* a planejar seu ataque de forma mais rápida e certeira.

Figura 11: *Banner* de versão

Fonte: Próprio autor.

Afim de ocultar essas informações, foram adicionadas duas novas instruções no arquivo de configuração do Servidor Web Apache. A figura a seguir (Figura 12) demonstra as alterações realizadas.

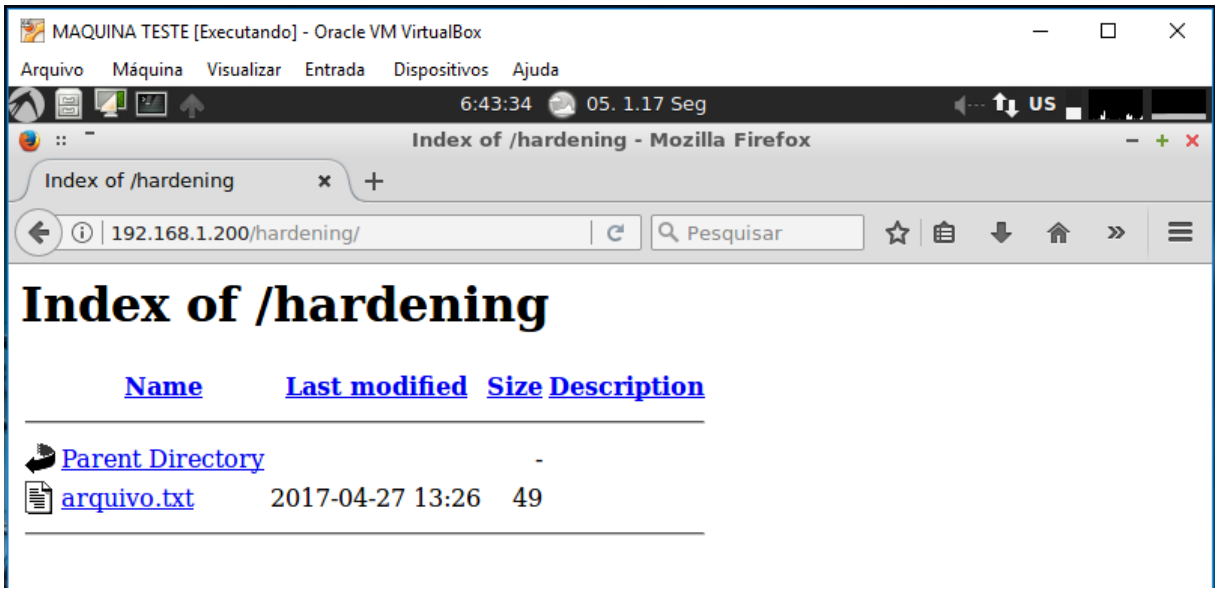
Figura 12: Removendo *Banner*



Fonte: Próprio autor.

O serviço foi reiniciado através do comando ***service apache2 restart*** e um novo acesso foi realizado.

Figura 13: Servidor sem *banner*

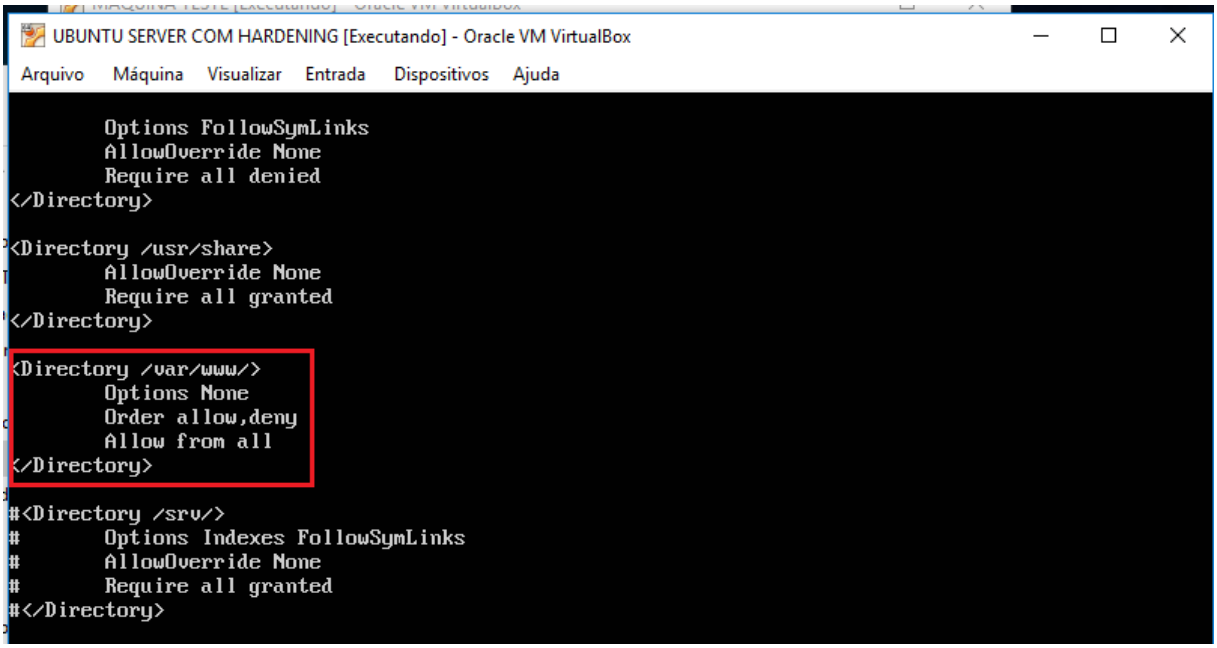


Fonte: Próprio autor.

5.2.1.2 REMOÇÃO DO LISTAGEM DE ARQUIVOS E DIRETÓRIOS

O Servidor Web Apache, em sua instalação padrão exibe ao visitante uma listagem dos arquivos e diretórios contidos no *web-space* como foi observado na figura anterior (Figura 13). Afim de não exibir mais essa listagem de arquivos e diretórios, foram feitas as seguintes alterações no arquivo de configuração do servidor web Apache (Figura 14).

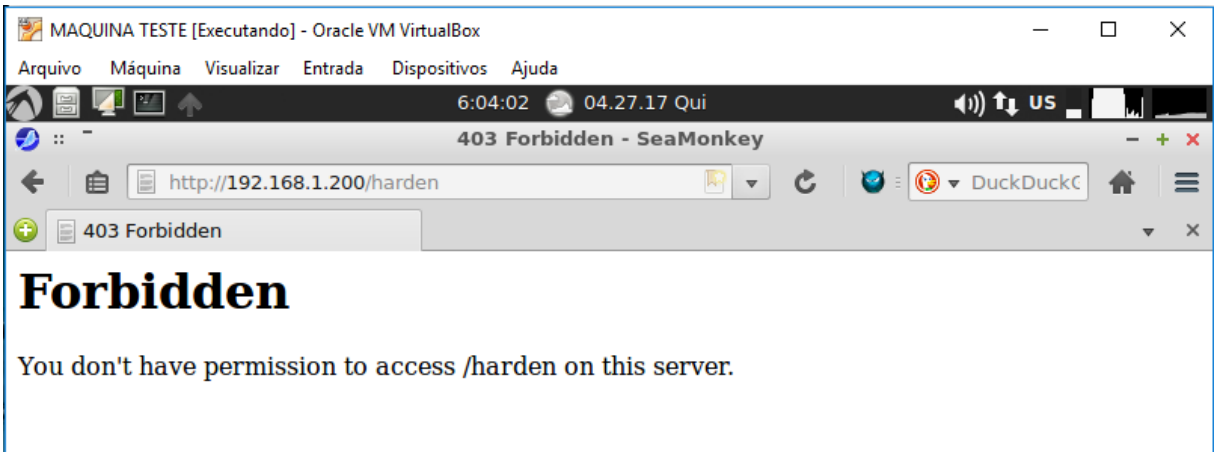
Figura 14: Removendo a listagem



Fonte: Próprio autor.

Após o serviço ser reiniciado através do comando ***sevice apache2 restart***, foi realizado um novo acesso, retornando a seguinte tela (Figura 15).

Figura 15: Listagem removida

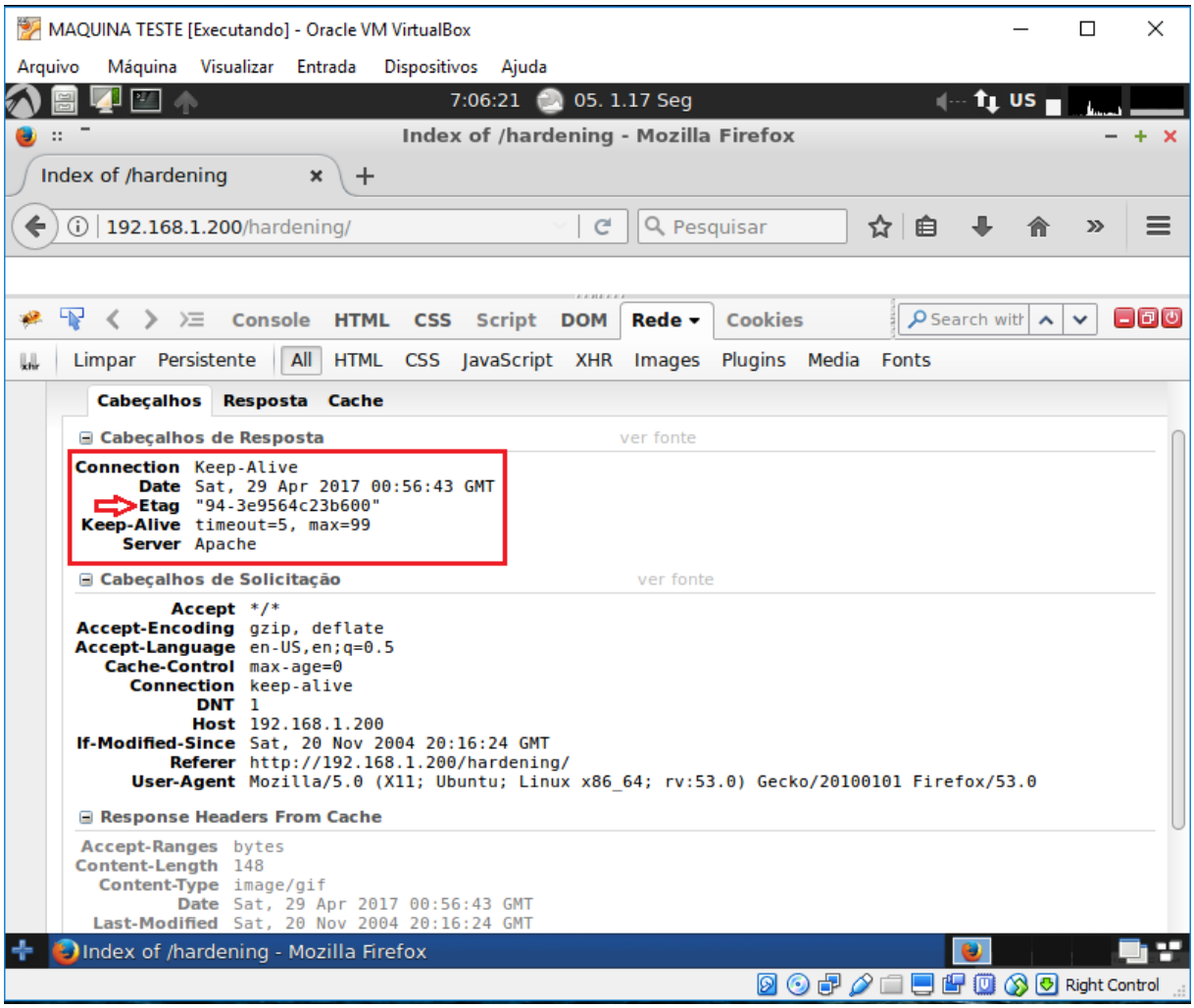


Fonte: Próprio autor.

5.2.1.3 ETAG

Ao utilizar o **firebug** na busca de novos vazamentos foi constatado que a Etag ainda era exibida. A Etag permite possíveis atacantes obterem informações confidenciais como o número do *inode*, limite MIME *multipart* e processos filhos através desse cabeçalho.

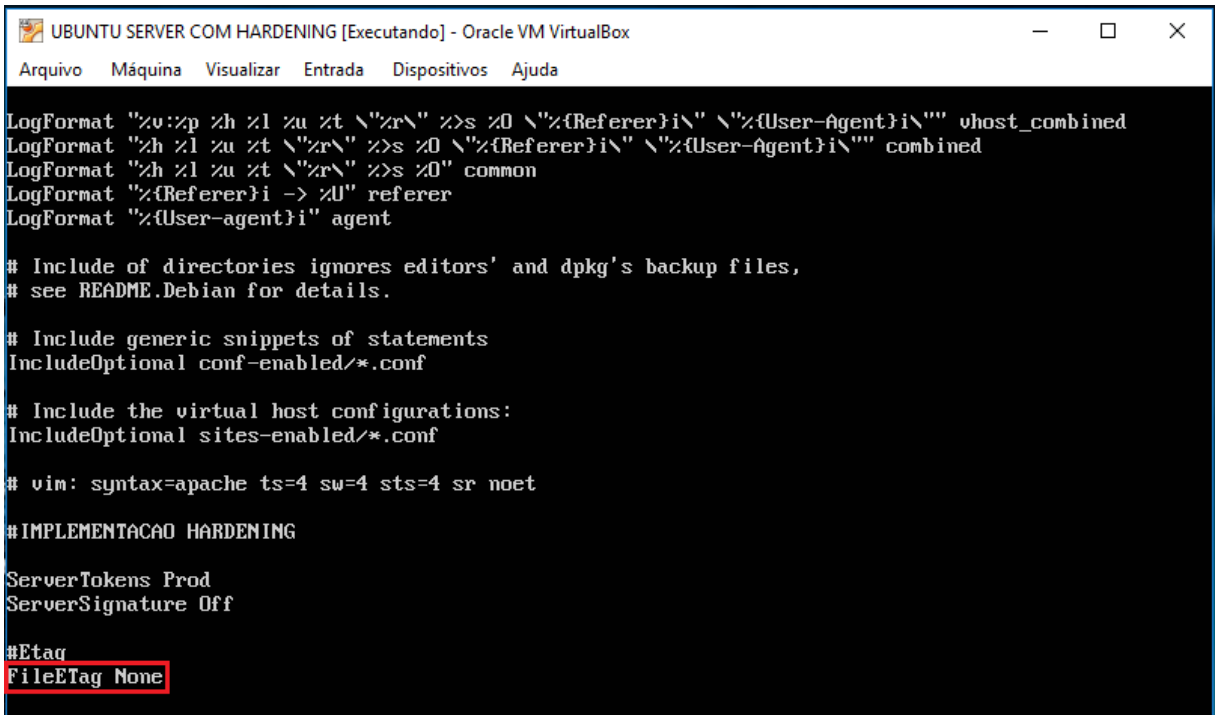
Figura 16: Exibição da Etag



Fonte: Próprio autor.

Para eliminar a vulnerabilidade encontrada na figura anterior (Figura 16), a seguinte linha foi adicionada ao arquivo do servidor web Apache (Figura 17) e mais uma vez o serviço foi reiniciado.

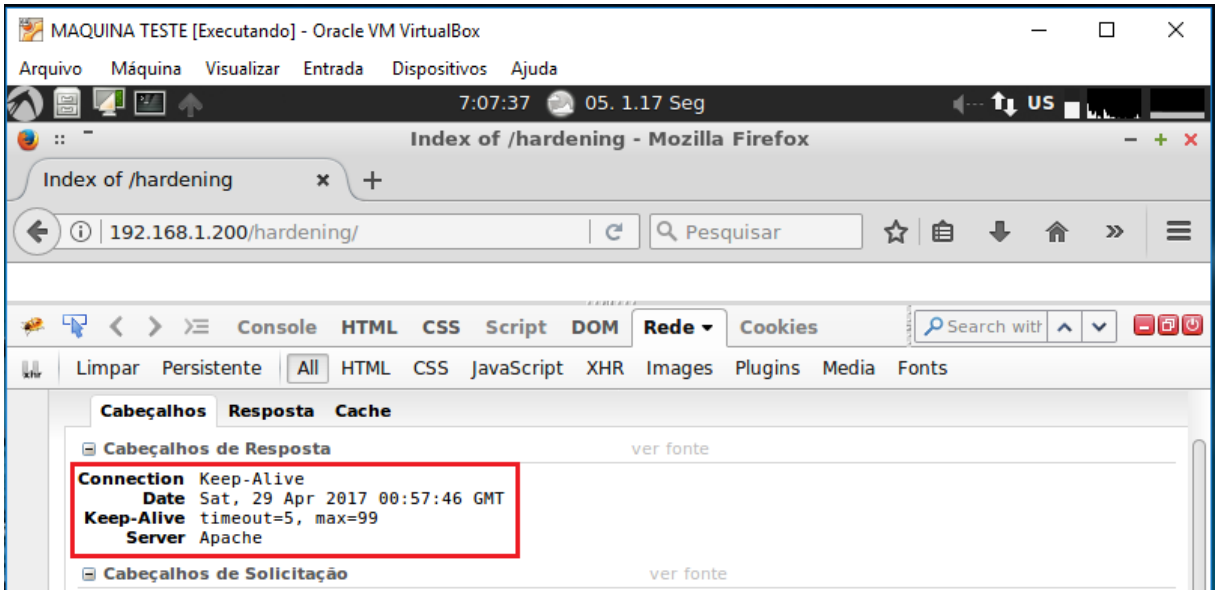
Figura 17: Removendo Etag



Fonte: Próprio autor.

Para confirmar que o vazamento da Etag foi resolvido, utilizando a máquina de teste foi realizado mais um acesso (Figura 18), comprovando o sucesso do procedimento e finalizando a etapa de vazamento de informações.

Figura 18: Etag removida



Fonte: Próprio autor.

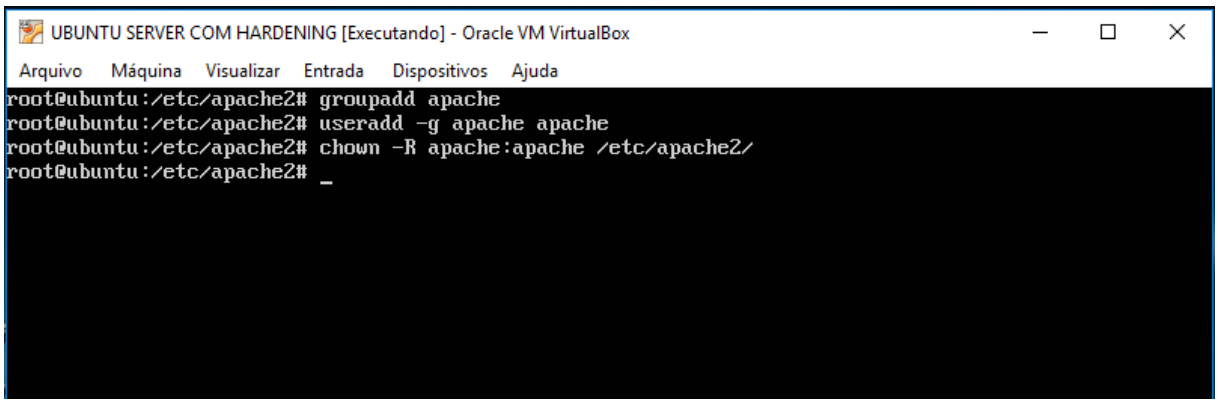
5.2.2 AUTORIZAÇÃO

A autorização é o processo de decidir que usuário tem acesso a acessar determinado recurso. Neste caso em especifico, será definir e determinar quais usuários poderão acessar o Servidor Web Apache.

5.2.2.1 EXECUTAR O APACHE COM UMA CONTA NÃO PRIVILEGIADA

A configuração padrão do Servidor Web Apache é para não necessitar de um usuário especifico ou executar simplesmente como um *deamon.* Neste caso é recomendado utilizar um usuário especifico e não privilegiado para o Servidor web Apache, pois caso ocorra algum problema de segurança os outros serviços estarão protegidos.

Figura 19: Adicionando usuário e grupo

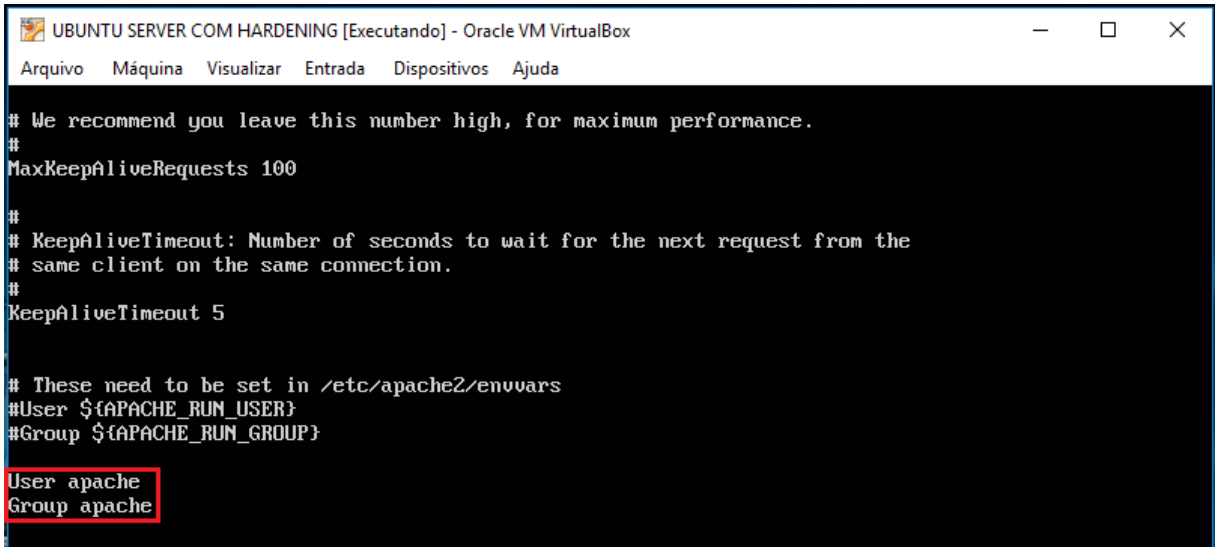


Fonte: Próprio autor.

Como pode ser observado na figura anterior (Figura 19), foi criado um grupo e um usuário chamado “apache” e posteriormente a propriedade do diretório da instalação do Servidor Web Apache foi alterada para este usuário não privilegiado.

Após estas alterações, as diretivas “Usuários e Grupos” do arquivo de configuração do Apache foram alteradas para a conta não privilegiada criada anteriormente.

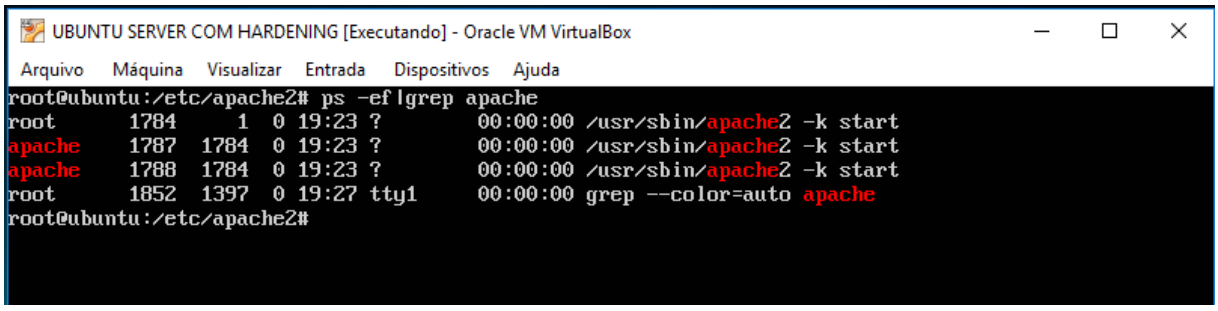
Figura 20: Alterando o usuário e o grupo



Fonte: Próprio autor.

Após o Servidor Web Apache ser reiniciado, foi verificado se a alteração havia funcionado através do comando **ps –ef | grep apache** (Figura 21).

Figura 21: Verificando o usuário

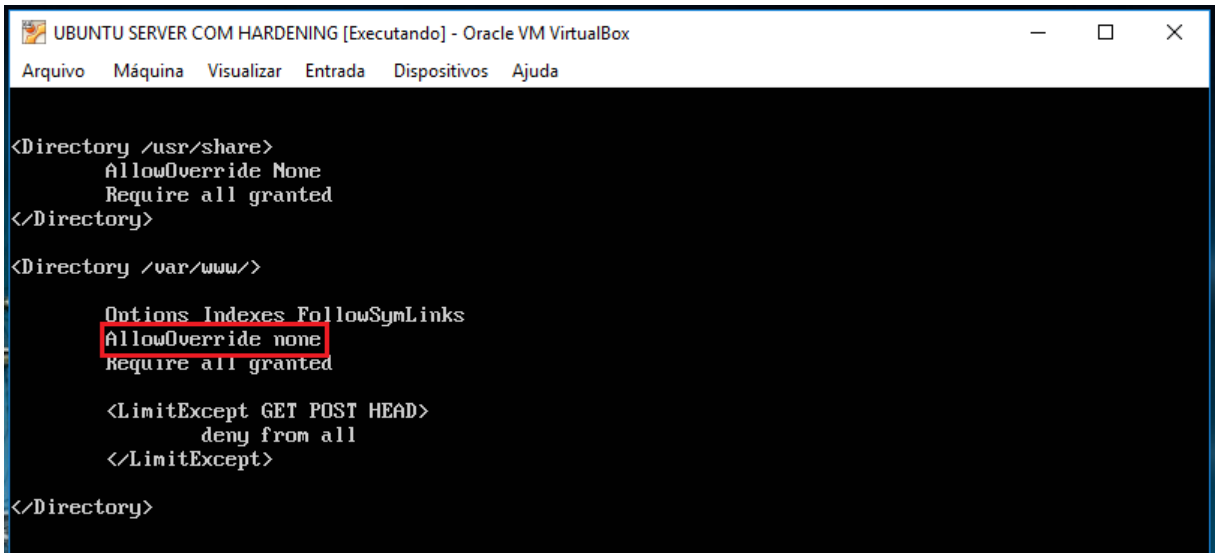


Fonte: Próprio autor.

5.2.2.2 PROTEÇÃO DAS CONFIGURAÇÕES DO SISTEMA

Em uma instalação padrão do Servidor Web Apache os usuários podem alterar as configurações do mesmo através do arquivo **.htaccess**. Para que isto não ocorra, basta realizar uma alteração no arquivo de configuração do web Apache (Figura 22).

Figura 22: Proteção do sistema



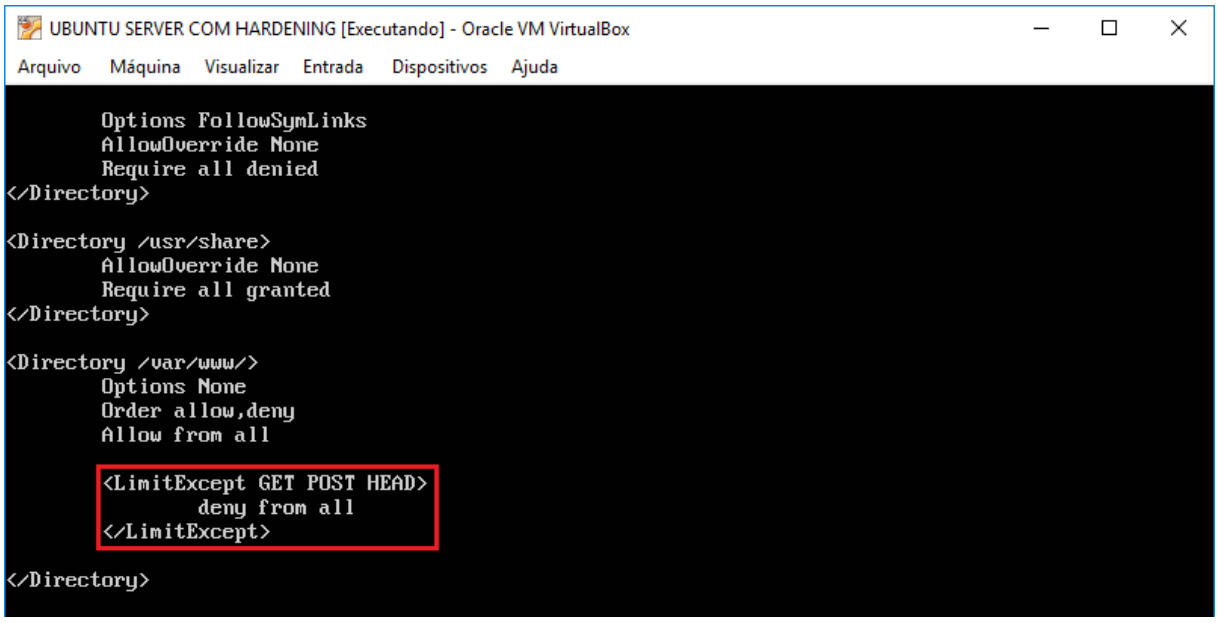
Fonte: Próprio autor.

5.2.2.3 MÉTODOS DE REQUISIÇAO HTTP

O protocolo HTTP 1.1 suporta muitos métodos de requisição que por vezes são desnecessários e podem oferecer um risco em potencial. Normalmente em uma aplicação Web são necessários apenas os métodos GET, POST e HEAD que podem ser configurados nas diretivas de diretórios. Em sua configuração padrão o Servidor Web Apache suporta utilizando protocolo HTTP 1.1 requisições nos métodos OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE e CONNECT.

A figura a seguir (Figura 23) demonstra como foram desabilitados os métodos de requisição desnecessários.

Figura 23: Desabilitando métodos de requisição



Fonte: Próprio autor.

5.2.3 WEB APPLICATION SECURITY

O Servidor Web Apache se configurado de maneira incorreta ou sem a aplicação das técnicas de *hardening*, pode permitir a exploração da aplicação web em execução no mesmo.

5.2.3.1 *COOKIES*

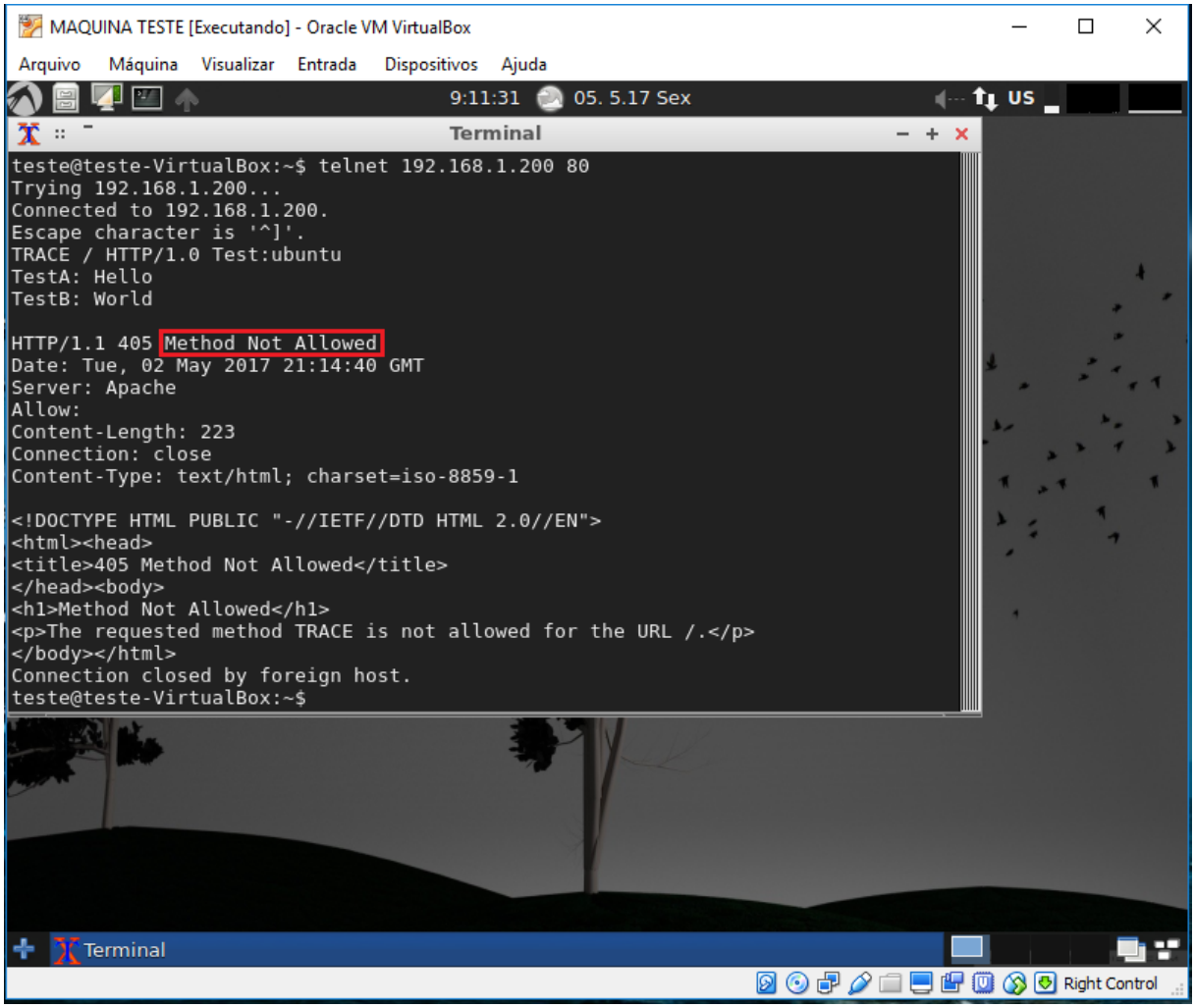
Segundo informações do site Mozilla, um cookie é um pequeno arquivo de dados que um servidor envia para o navegador cliente, que por sua vez pode armazená-lo e envia-lo junto na próxima solicitação para o mesmo servidor.

5.2.3.1.1 DESABILIAR A REQUISIÇÃO HTTP DE RASTREAMENTO

Por padrão o método trace está habilitado no Servidor Web Apache e isso pode permitir um ataque chamado *Cross Site Tracing,* oferecendo a possibilidade de um *hacker* através da linguagem Javascript, ler informações de um *cookie* para um possível sequestro de sessão.

No entanto, na versão em questão (2.4.18) esta opção veio desabilitada como comprova o teste demonstrado na figura a seguir (Figura 24).

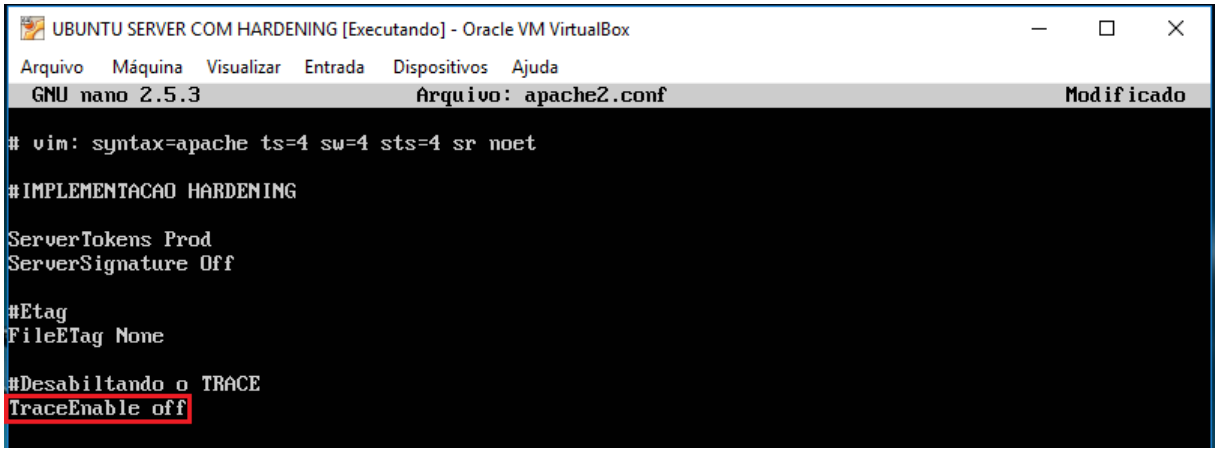
Figura 24: Trace Teste



Fonte: Próprio autor.

Caso esta opção esteja habilitada, basta ir até o arquivo de configuração do Servidor Web Apache e adicionar a seguinte linha de acordo com a próxima figura (Figura 25).

Figura 25: Desabilitando o trace



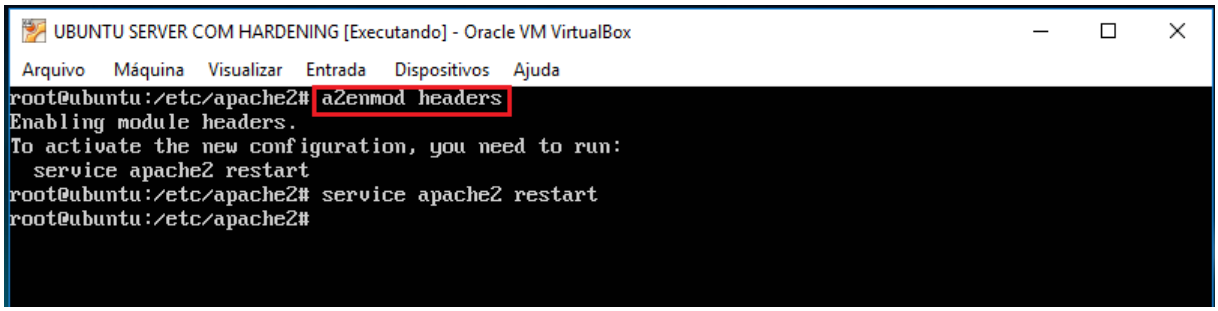
Fonte: Próprio autor.

5.2.3.1.2 DEFINIR *COOKIE* COM HTTPONLY E SECURE FLAG

Pode-se mitigar a maior parte do ataque chamado *Cross Site Scripting* utilizado *HttpOnly* e *flag* segura em um cookie. Sem o *HttpOnly* e o *secure flag*, é possível roubar ou até manipular a sessão de criada por uma aplicação web.

Para ativar o *HttpOnly* e *secure flag*, é necessário acessar o arquivo de configuração do Servidor web Apache e verificar se *o mod_headers* está ativado, caso não esteja, basta utilizar o seguinte comando (Figura 26).

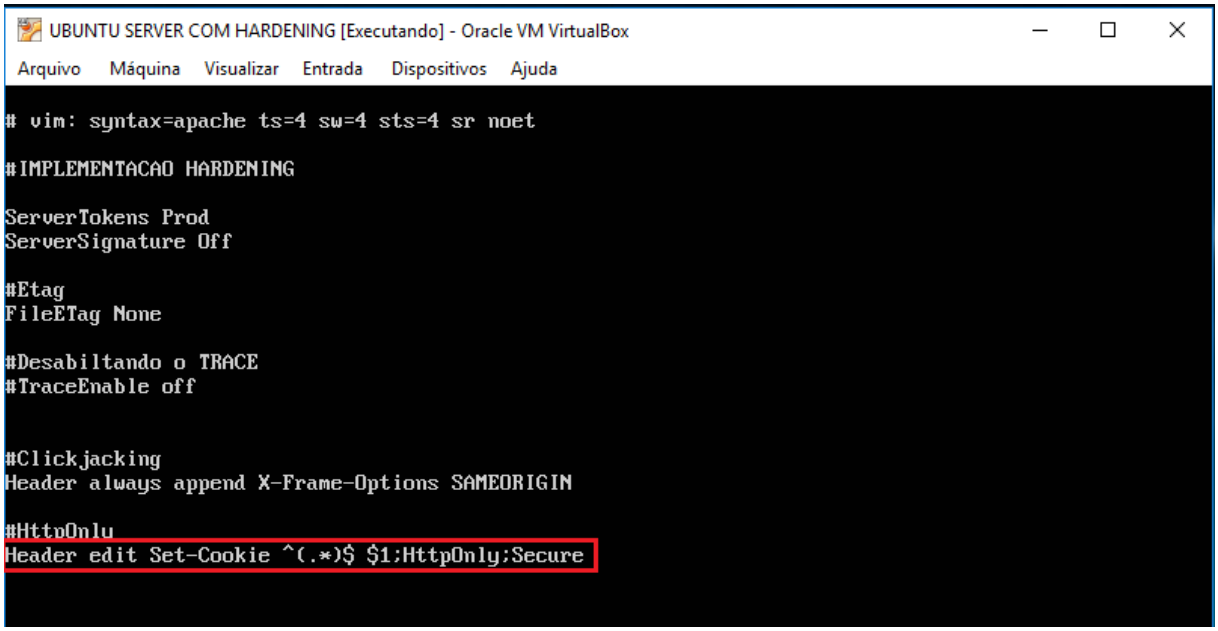
Figura 26: Ativação *Mod Headers*



Fonte: Próprio autor.

Após ativado o *mod_header*, foi adicionada a seguinte linha ao arquivo de configuração do Servidor Web Apache (Figura 27).

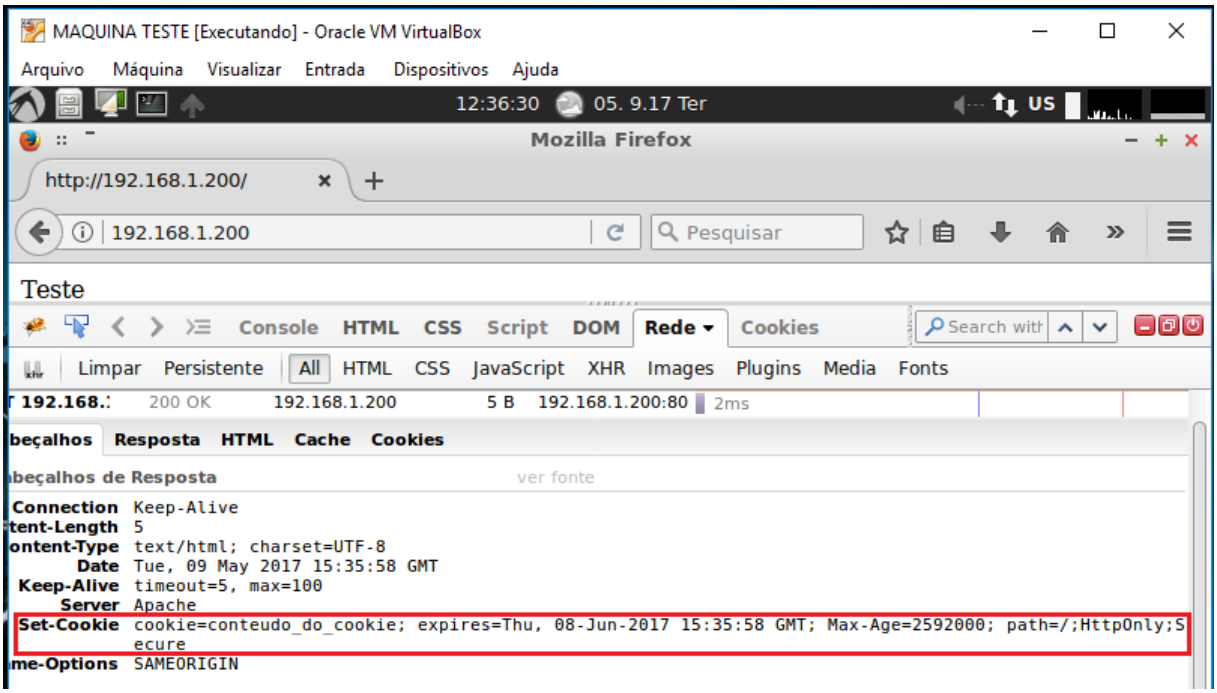
Figura 27: Http Only



Fonte: Próprio autor.

Após feita a devida correção, foi efetuado um novo teste e foi obtido o seguinte resultado (Figura 28).

Figura 28: Teste HttpOnly



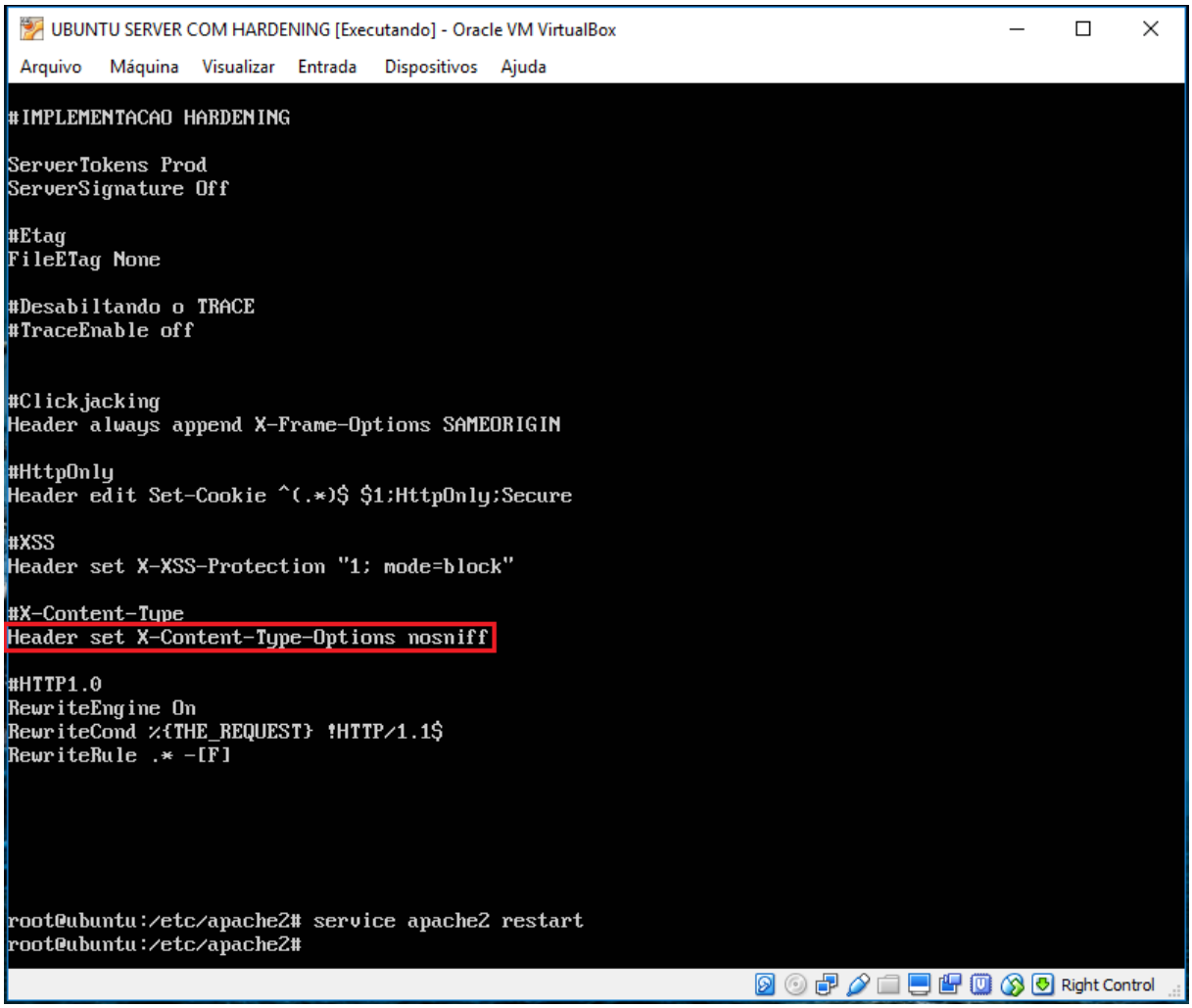
Fonte: Próprio autor.

5.2.3.1.3 X-CONTENT-TYPE-OPTIONS HEADER

O cabeçalho *X-Content-Type-Options* é utilizado para proteger o servidor web contra vulnerabilidades do tipo MIME *sniffing.* Elas ocorrem quando um site permite que seus visitantes façam *upload* de conteúdo para o site, mas algum usuário mal intencionado disfarça um tipo de arquivo específico como se fosse outro. Isso pode dar a oportunidade ao atacante de comprometer a aplicação.

Para remover esta vulnerabilidade, basta ativar o cabeçalho *X-Conntent-Type-Options* no arquivo de configuração do servidor web Apache como demonstrado a seguir (Figura 29) uma vez que o *mod_header* esteja ativado.

Figura 29: Ativando *X-Content-Type-Option*



Fonte: Próprio autor.

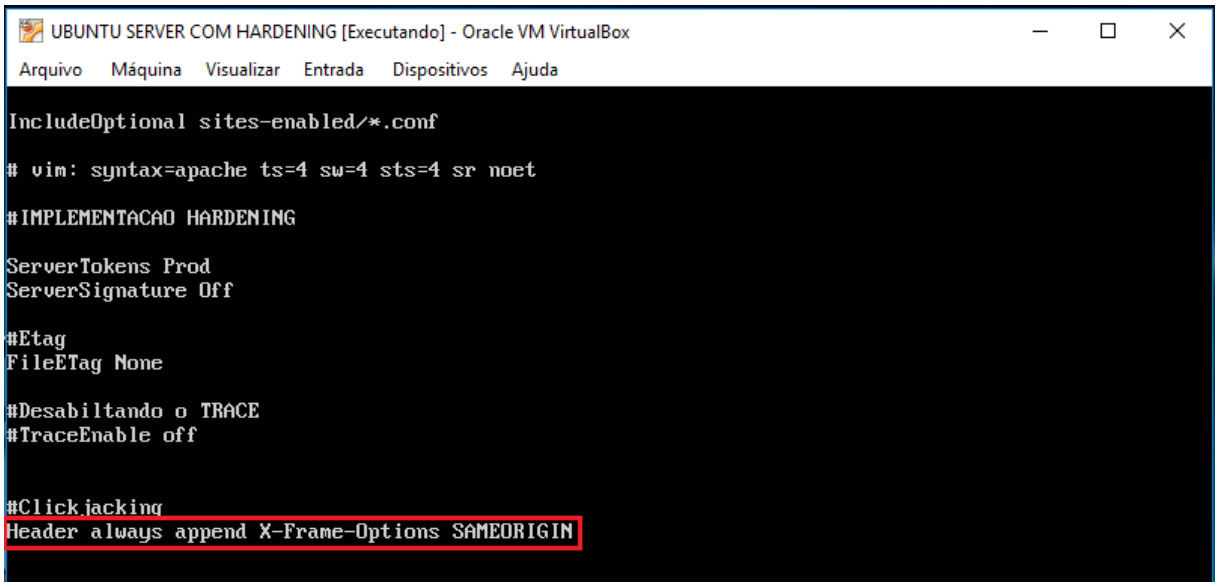
5.2.3.1.4 ATAQUE CLICKJACKING

Este ataque pode ter seu nome traduzido como furto de clique e consiste em infectar os botões de um site legitimo e o usuário ao clicar no botão acaba baixando inconscientemente arquivos maliciosos para sua máquina ou pode ser redirecionado para uma página maliciosa.

O cabeçalho de resposta X-Frame pode ser utilizado para indicar se um navegador deve ou não abrir uma página em *iframe*, isso impedira que algum conteúdo incorporado de outro site seja exibido.

Para sanar tal vulnerabilidade, foi adicionada a seguinte linha ao arquivo de configuração do servidor Web Apache (Figura 30).

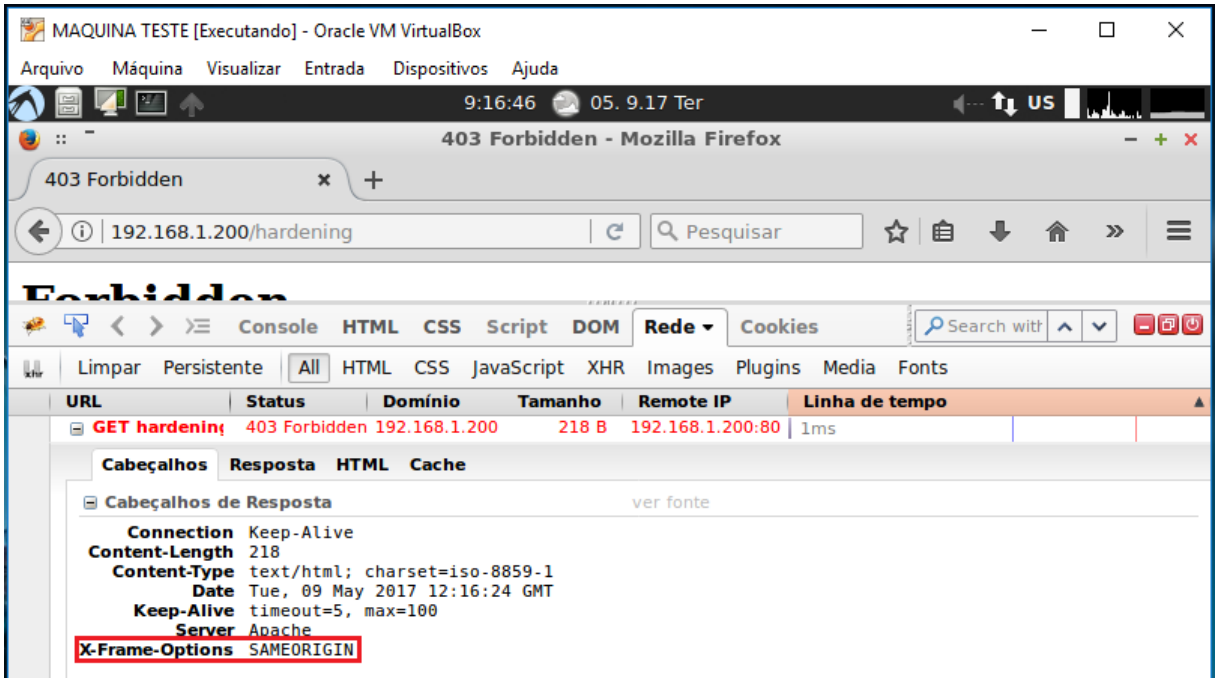
Figura 30:Removendo *clickjacking*



Fonte: próprio autor.

Para comprovar que a vulnerabilidade foi eliminada, foi realizado um acesso ao servidor e os dados foram analisados através da ferramenta **firebug** (Figura 31).

Figura 31: *Clickjacking* resolvido



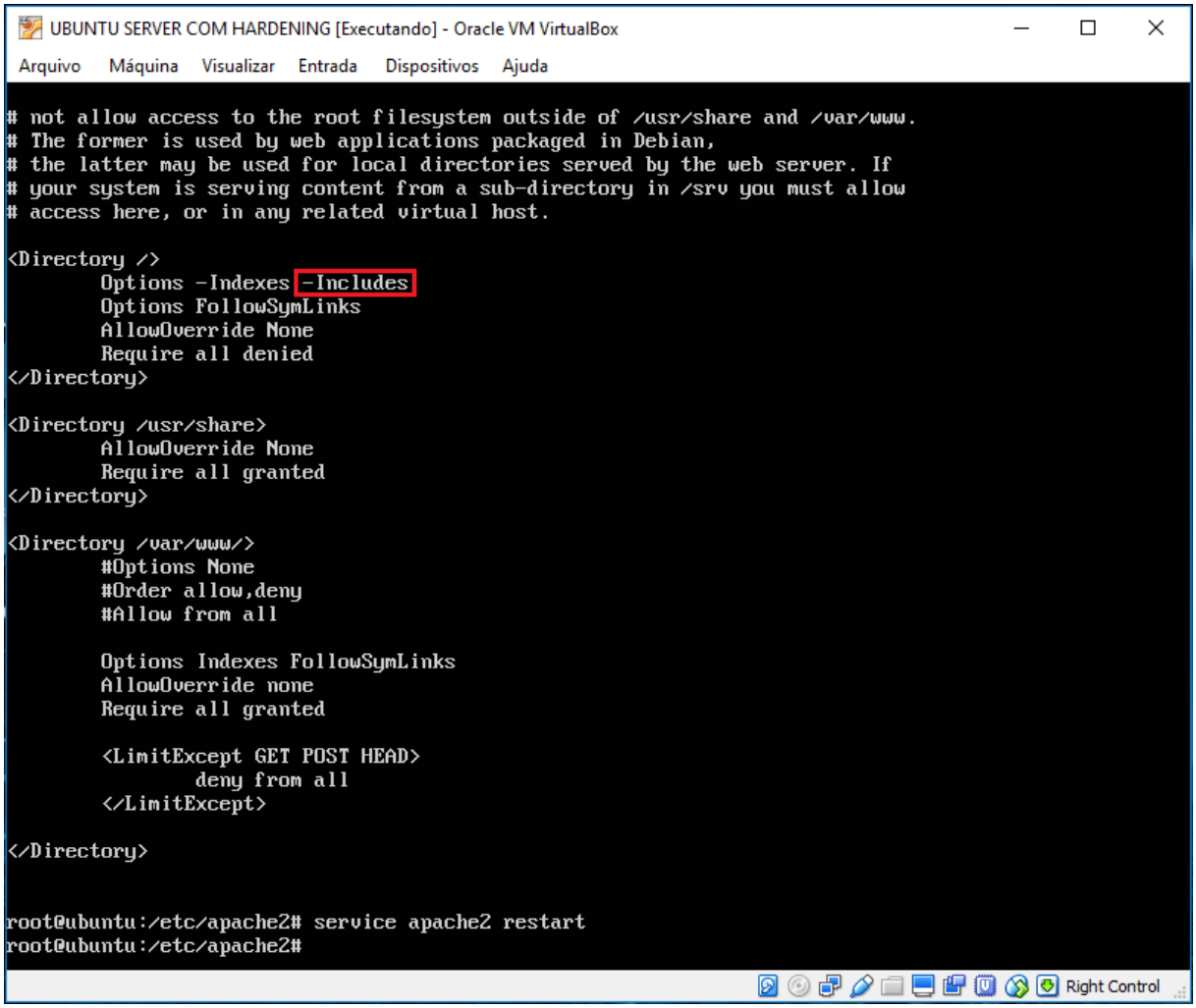
Fonte: Próprio autor.

5.2.3.1.5 SERVER SIDE INCLUDE

Server Side Include ou simplesmente SSI é uma parte de código que pode estar incluído em uma página HTML e é interpretada pelo servidor após a página ser enviada ao navegador da máquina cliente e possui um risco de aumentar a carga no servidor. Se o ambiente é compartilhado com aplicações de tráfego pesado, deve ser considerado a ativação do SSI. O ataque SSI permite a exploração de um aplicativo Web injetando *scripts* em páginas HTML ou executando códigos remotamente.

Para realizar esta correção, foi adicionada a seguinte opção no arquivo de configuração do servidor Web Apache (Figura 32).

Figura 32: Solução SSI



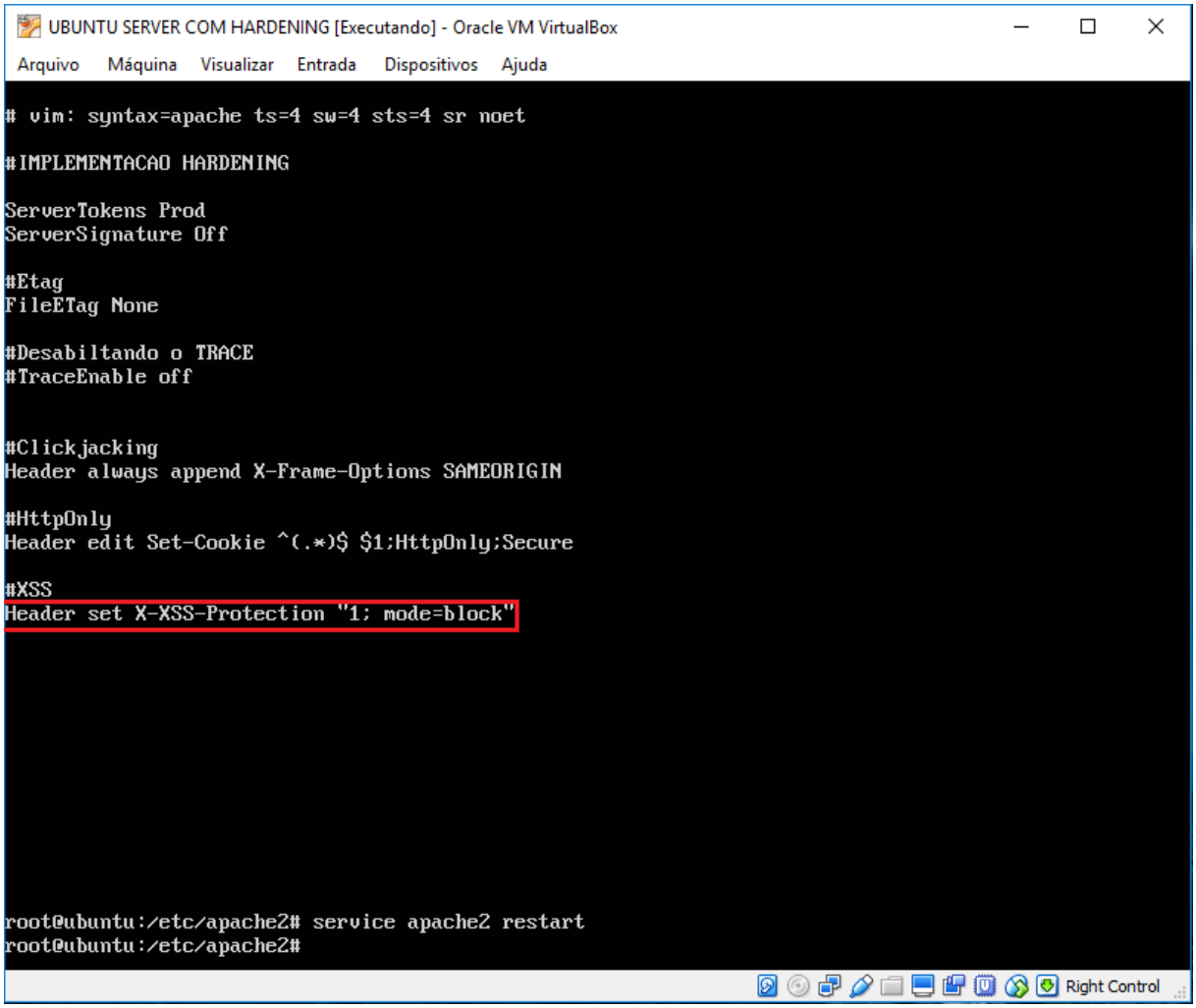
Fonte: Próprio autor.

Após a alteração, o serviço foi reiniciado através do comando **service apache2 restart**, e o mesmo voltou a funcionar sem nenhum problema.

5.2.3.1.6 PROTEÇÃO X-XSS

A proteção Cross Site Scripting (XSS) pode ser ignorada em muitos navegadores, mas ela pode ser aplicada em uma aplicação Web mesmo que ela tenha sido desativada pelo usuário. Para ativa-la, basta adicionar a seguinte linha do arquivo de configuração do Apache (Figura 33).

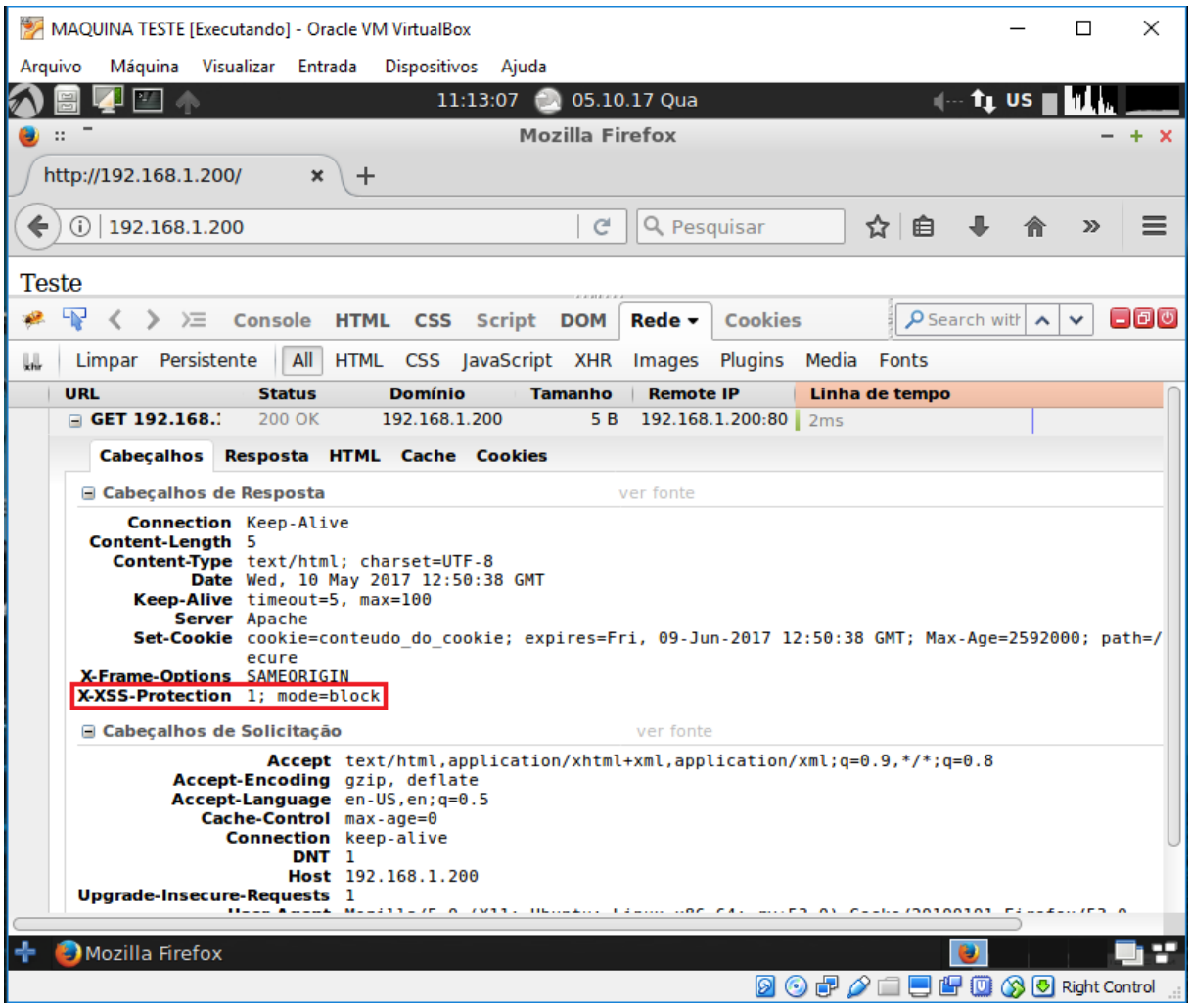
Figura 33: Ativando proteção XSS



Fonte: Próprio autor.

Após esta implementação, foi realizado um acesso ao servidor e com o auxílio da ferramenta **firebug**, foi obtido o resultado a seguir (Figura 34).

Figura 34: Teste XSS



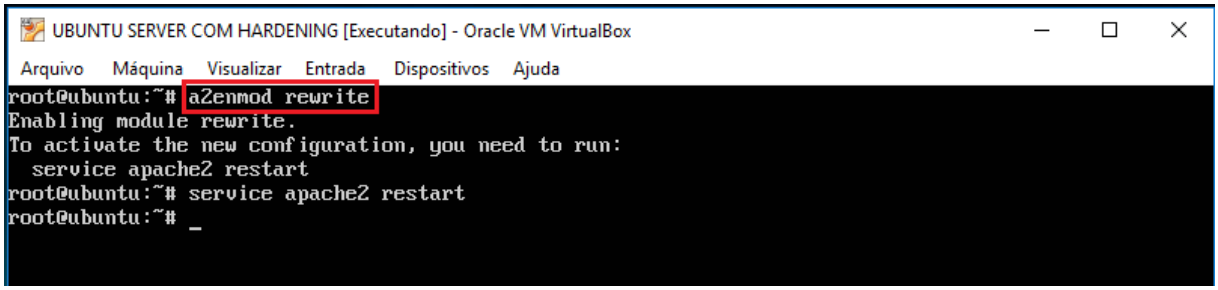
Fonte: Próprio autor.

5.2.3.1.7 DESABILITANDO O PROTOLO HTTP 1.0

O protocolo HTTP em sua versão 1.0, possui uma vulnerabilidade já conhecida relacionada ao sequestro de sessão. Para resolver este problema basta desabilitar o mesmo utilizando o módulo *mod_rewrite.*

Para habilitar o *mod_rewrite* basta acessar o terminal de comando do servidor e digitar o seguinte comando (Figura 35).

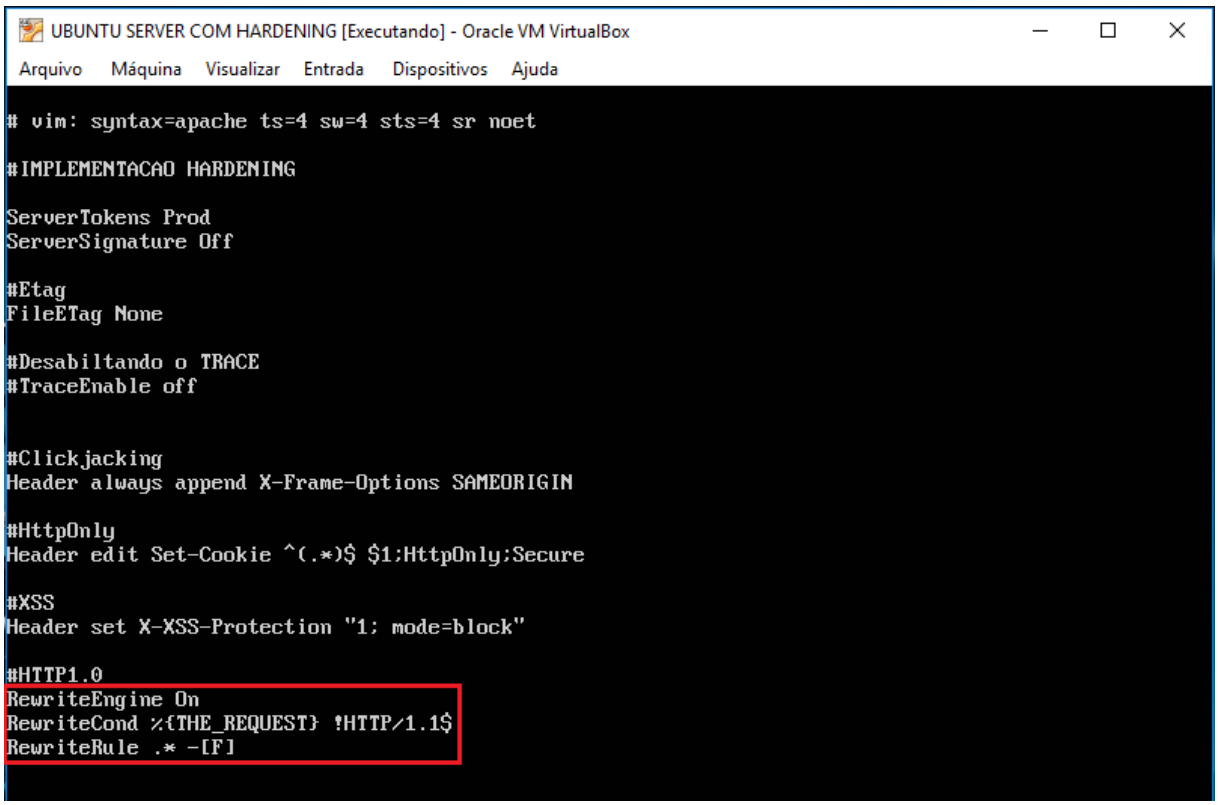
Figura 35: Habilitando Mod_Rewrite



Fonte: Próprio autor.

Após a ativação do *mod_rewrite*, basta ir ao arquivo de configuração do servidor Web Apache e realizar a adição das linhas demonstradas na figura a seguir (Figura 36).

Figura 36: Habilitando o rewrite



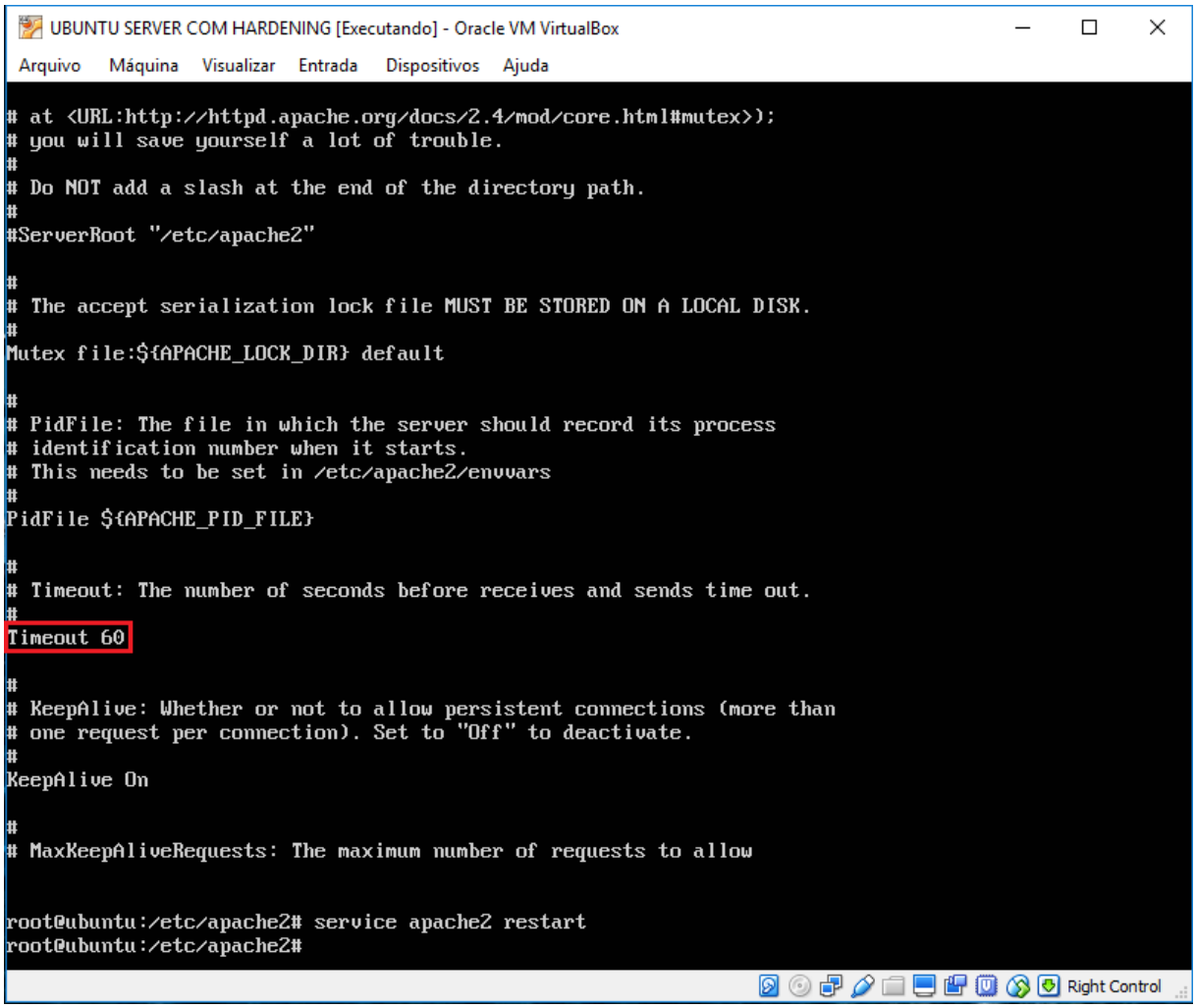
Fonte: Próprio autor.

5.2.3.1.8 CONFIGURAÇÃO DO VALOR DO *TIMEOUT*

Por padrão, o valor do *timeout* (tempo limite) do servidor Web Apache é de 300 segundos, o que pode torná-lo uma vítima de DDoS. Para mitigar esta ameaça, pode-se diminuir o valor do tempo limite para por exemplo 60 segundos.

Para isto, basta editar o arquivo de configuração do Apache e alterar o tempo limite como na figura a seguir (Figura 37).

Figura 37: Alterando o *TIMEOUT*



Fonte: Próprio autor.

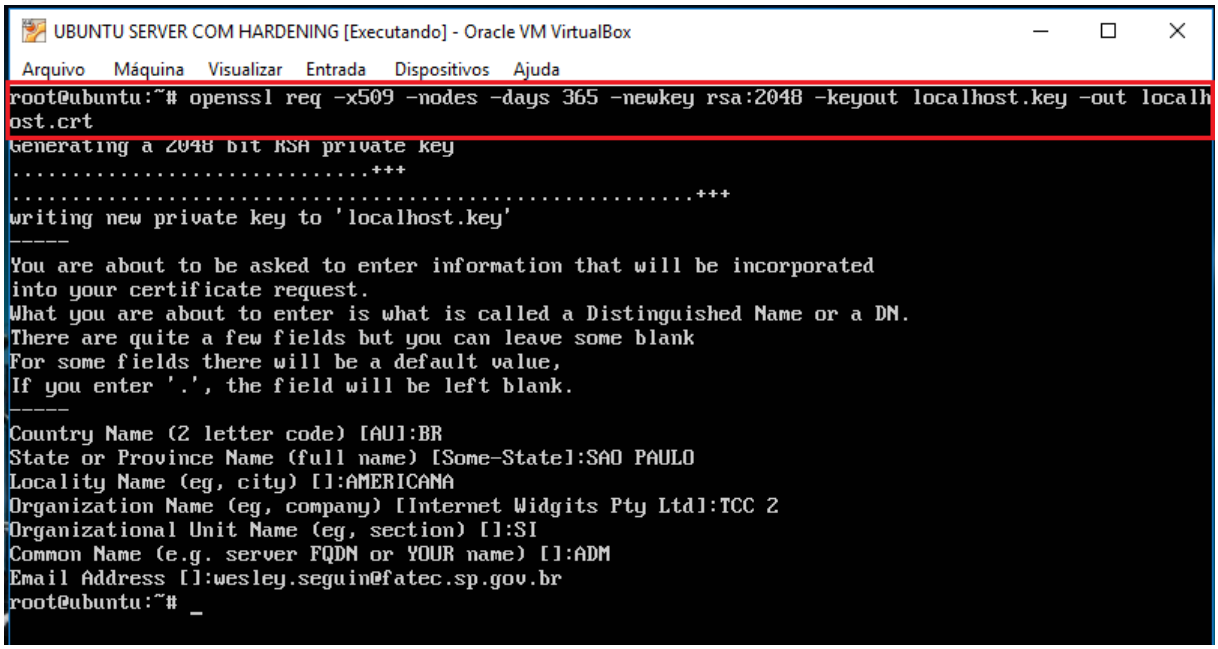
5.2.3.2 SSL

O SSL é uma camada adicional de segurança na aplicação Web. Mas a sua configuração padrão possui determinadas vulnerabilidades que devem ser consideradas.

5.2.3.2.1 CHAVE SSL

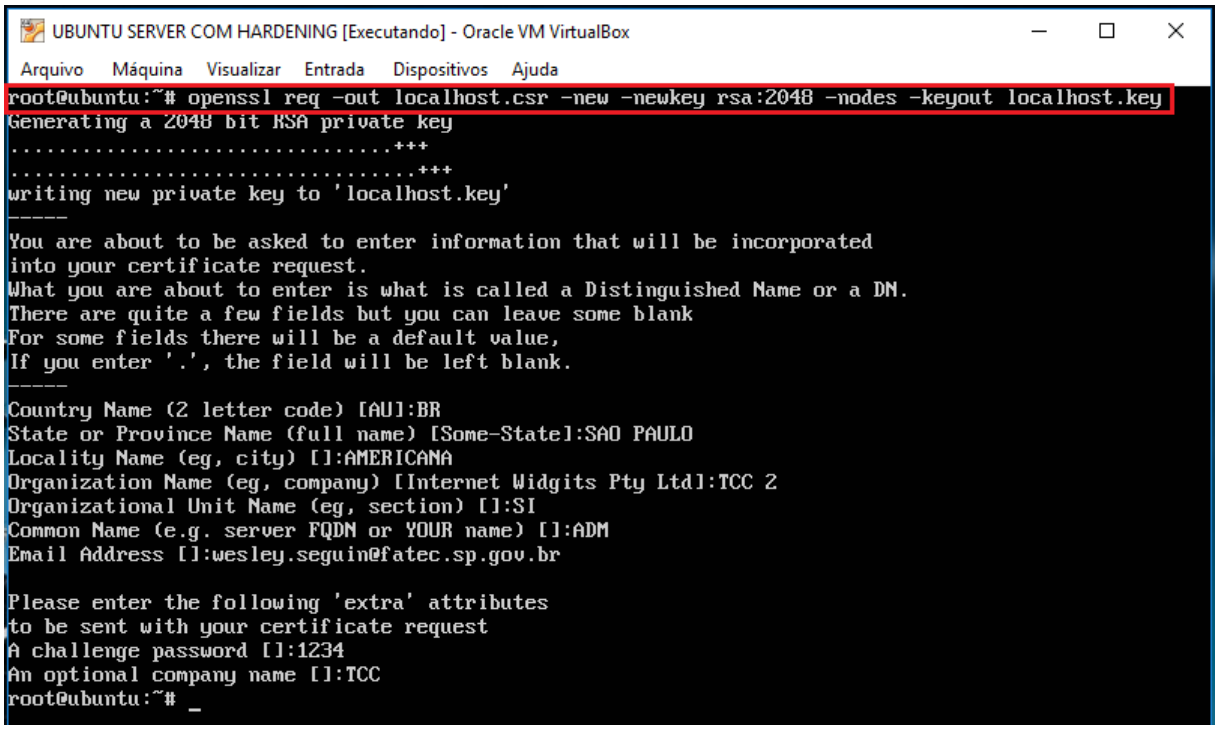
A violação da chave SSL é difícil, mas não impossível. Para quebrar uma chave SSL é necessário poder computacional e tempo. Portanto, quanto maior o comprimento da chave, mais difícil fica para quebrá-la. A seguir será demonstrado como foi gerado o certificado auto assinado com uma chave SSL de tamanho 2048 bit (Figura 38) e como foi gerada uma nova CSR e chave primária (Figura 39).

Figura 38: Gerando o certificado



Fonte: Próprio autor.

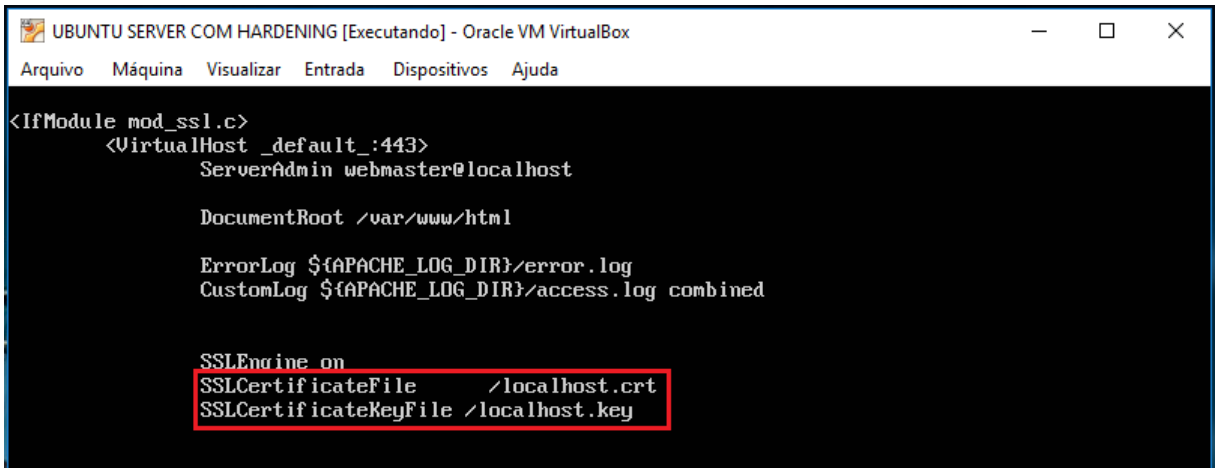
Figura 39: Gerando a chave



Fonte: Próprio autor.

Agora basta indicar no arquivo de configuração localizado em /etc/apache2/sites-avaliable/default-ssl.conf onde se encontram os arquivos do certificado e da chave SSL (Figura 40).

Figura 40: Indicando o certificado e a chave SSL

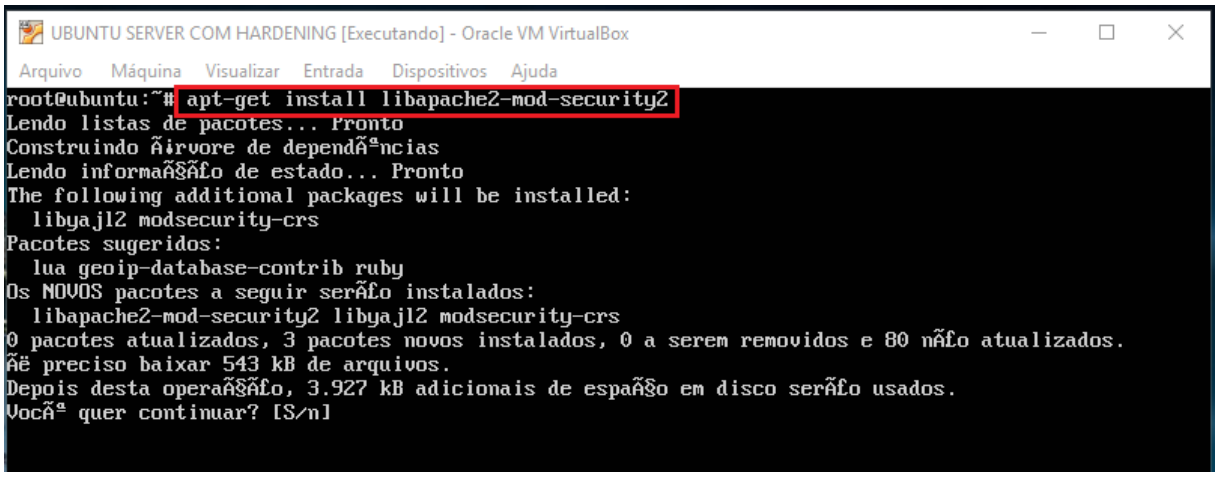


Fonte: Próprio autor.

5.2.3.3 MOD SECURITY

O Mod Security funciona como um *firewall* para aplicações web de código aberto. Ele também auxilia na proteção dos sites e do próprio servidor web contra ataques de força bruta. Para instalar, ele pode ser compilado ou simplesmente baixado pelo gerenciador de pacotes através do comando mostrado na Figura 41.

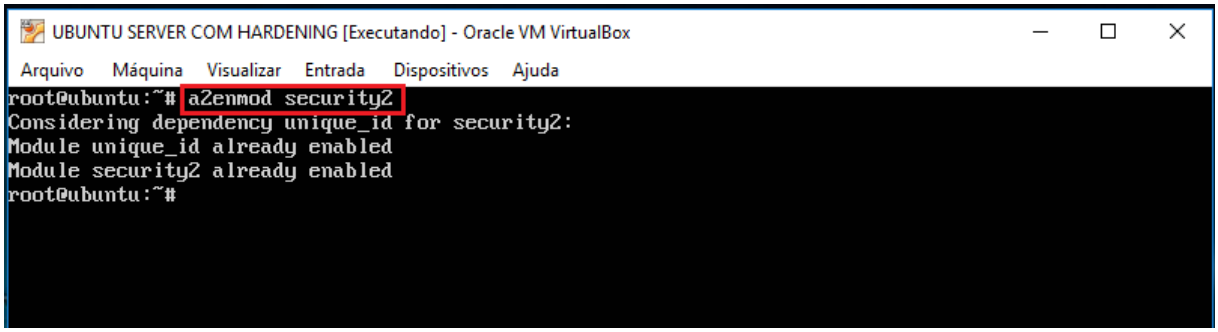
Figura 41: Instalando *Mod Security*



Fonte: Próprio autor.

Após instalado o Mod Security, o mesmo precisa ser ativado para funcionar. Para isto basta acessar o terminal do servidor e inserir no terminal o comando apresentado na figura 42.

Figura 42: Habilitando *Mod Security*



Fonte: Próprio autor.

5.2.3.4 CONFIGURAÇÕES GERAIS

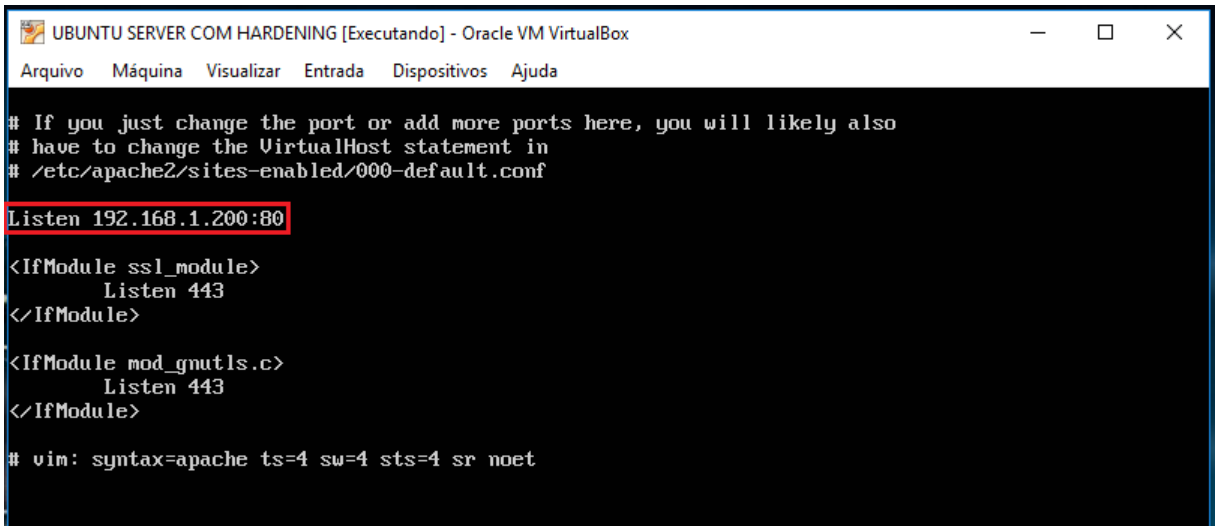
Aqui serão apresentadas as melhores práticas a respeito das configurações gerais do servidor web Apache.

5.2.3.4.1 CONFIGURANDO O LISTEN

Quando há múltiplas interfaces e IP'S em um único servidor, é recomendado possuir uma diretiva configurada com o IP e o número da porta. Quando a configuração do servidor web Apache é para escutar todos os IP'S com algum número de porta, pode haver um problema de encaminhamento de solicitação HTTP para outro servidor web.

Para solucionar este problema, basta alterar a seguinte linha no arquivo *ports.conf*, que é instanciado no arquivo de configuração do servidor web Apache adicionando o IP do servidor em questão (Figura 43).

Figura 43: Configuração IP e porta

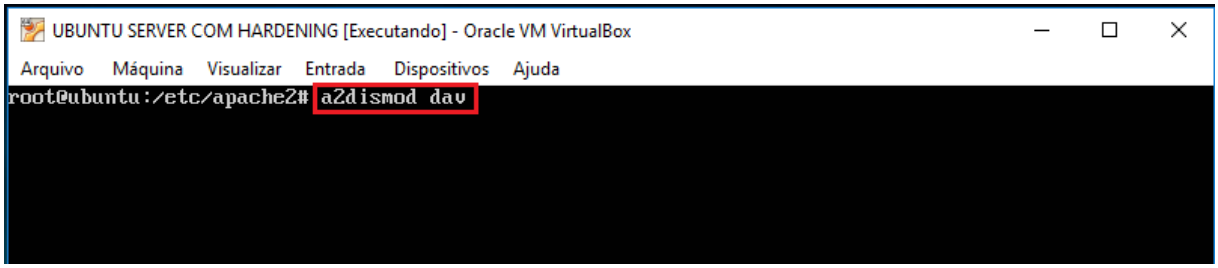


Fonte: Próprio autor.

5.2.3.5 DESABILITANDO O MÓDULOS DESNECESSÁRIOS

Se foram instalados todos os módulos, existe uma grande chance de haver algum desnecessário sendo carregado no servidor web Apache. Neste caso, o melhor a se fazer é configurar o Apache apenas com os módulos necessários para a aplicação web em execução. O módulo em especial **WebDAV**, permite que clientes remotos manipulem arquivos no servidor e que o mesmo fique sujeito a ataques de negação de serviço. Na instalação em questão, o mesmo veio desativado, caso contrário para desativá-lo seria necessário digitar o seguinte comando (Figura 44).

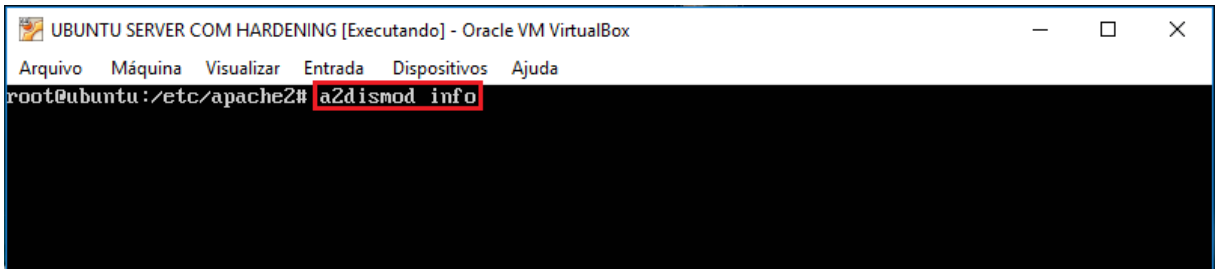
Figura 44: Desabilitar WebDAV



Fonte: Próprio autor.

Outro módulo que pode ser desabilitado é o *info_module* pois o mesmo pode fornecer informações sensíveis através do *.htaccess*. Este módulo também veio desativado nesta instalação, mas se estivesse ativado, bastaria digitar o seguinte comando (Figura 45).

Figura 45: Desabilitar info_module

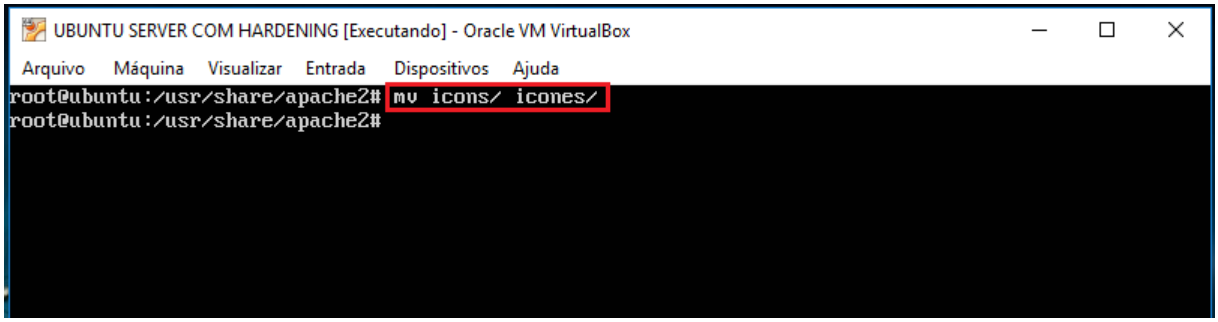


Fonte: Próprio autor.

5.2.3.6 ALTERAR O DIRETÓRIO ICONS

Quando instalado, por padrão o servidor web Apache cria um diretório chamado *icons*, que contém os ícones padrão do Apache que por vezes não são necessários a aplicação em execução no servidor. Então recomenda-se ao menos renomear este diretório para retirá-lo da configuração padrão e ser um recurso a menos conhecido pelo atacante (Figura 46).

Figura 46: Alterando diretório icons



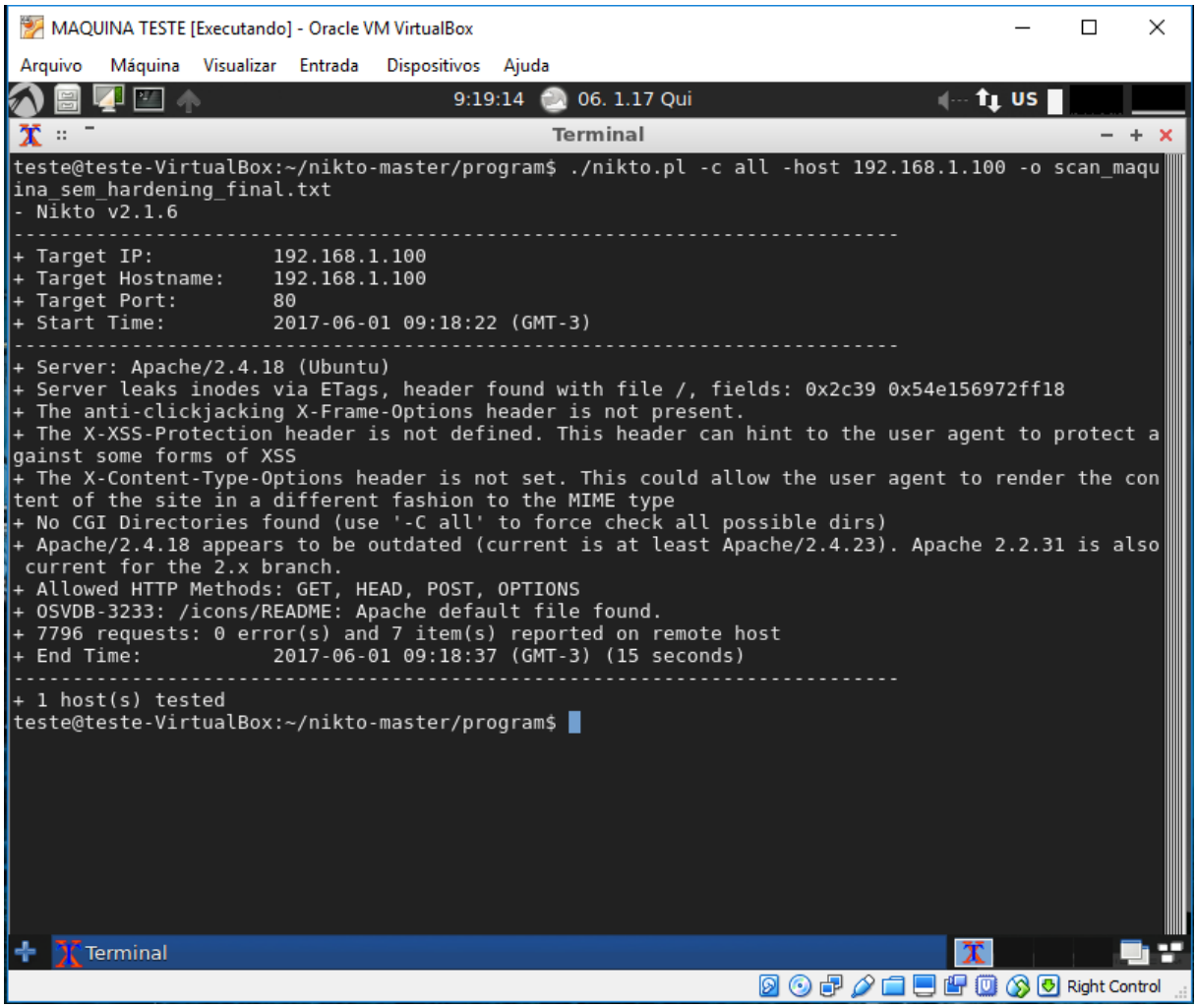
Fonte: Próprio autor.

5.2 RESULTADO DO EXPERIMENTO

Afim de checar se realmente o *hardening* foi realmente eficaz em eliminar as vulnerabilidades encontradas no servidor web Apache, foi realizado um *scanner* de vulnerabilidades em duas maquinas. A ferramenta escolhida para esta tarefa foi o Nikto, por ser gratuito, de simples utilização e ser criada especificamente para servidores web.

O teste a seguir foi realizado na máquina denominada UBUNTU SERVER SEM *HARDENING*, que foi deixada em suas configurações padrões, e foi obtido o seguinte resultado (Figura 47).

Figura 47: Teste da máquina sem *hardening*

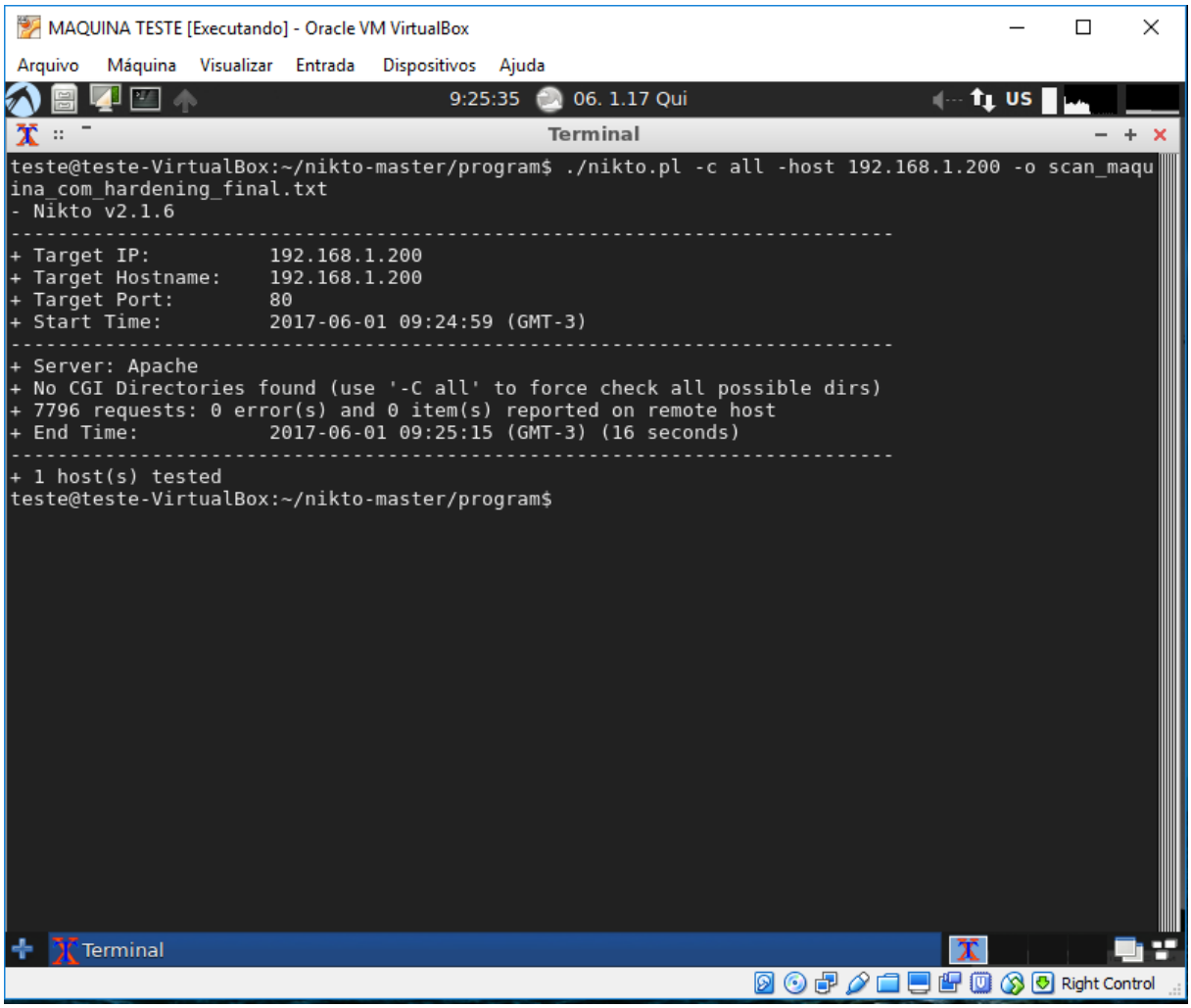


Fonte: Próprio autor.

O teste apresentado na figura anterior (Figura 47), revela que a ferramenta Nikto foi capaz de encontrar apenas uma parte das vulnerabilidades presentes no servidor Web Apache, neste caso, foram reportados 7 itens, deixando por conta do administrador a tarefa de encontrar as demais vulnerabilidades através de testes e análises das respostas obtidas em cada solicitação feita ao servidor.

Por fim a máquina chamada UBUNTU TESTE COM *HARDENING* assim como a anterior, foi testada com o auxílio da ferramenta Nikto e foi obtido o resultado a seguir (Figura 48) comprovando o sucesso da aplicação das técnicas de *hardening* em servidores web Apache.

Figura 48: Teste da máquina com *hardening*



Fonte: Próprio autor.

O teste realizado na figura anterior (Figura 48), mostrou que as técnicas de *hardening* foram capaz de mitigar as vulnerabilidades encontradas no teste anterior (Figura 47), após o fim da procura por vulnerabilidades com o auxílio ferramenta Nikto, foi obtido um resultado onde nenhuma vulnerabilidade foi reportada, comprovando o sucesso das técnicas de *hardening* aplicadas ao servidor Web Apache.

6 CONSIDERAÇÕES FINAIS

Ao final deste trabalho foi possível concluir a veracidade da hipótese formulada no capítulo de introdução, a qual diz que, a aplicação das técnicas de *hardening* em um servidor Web Apache, é capaz de mitigar os problemas de segurança da informação presente no mesmo quando utilizadas suas configurações padrões, como pode ser observado no último teste (Figura 48), o qual comprovou que o *hardening* de fato mitigou todas as vulnerabilidades encontradas nas configurações padrões do servidor Web Apache.

Durante o desenvolvimento deste trabalho, os objetivos específicos propostos inicialmente foram cumpridos pois, foram identificadas as vulnerabilidades presentes na aplicação, as mesmas foram solucionadas, os processos para a resolução foi demonstrado de forma detalhada, as soluções foram testadas e por fim, foi realizado um teste comparativo afim de comprovar a eficácia das técnicas de *hardening* aplicadas em um servidor Web Apache.

O objetivo geral também foi cumprido, uma vez que as técnicas de *hardening* foram implementadas em um servidor Web Apache apenas alterando suas configurações padrões para contribuir com os três principais pilares da segurança da informação (Disponibilidade, Confidencialidade e Integridade).

Algumas ferramentas adicionais também foram instaladas como por exemplo o *Mod Security* o que fez o servidor Web Apache alcançar um nível mais alto de segurança.

Para dar continuidade a esta pesquisa e alcançar níveis mais altos de segurança da informação no servidor Web Apache, recomenda-se o estudo e a aplicação das técnicas de *hardening* não apenas no servidor Web Apache, mas também na máquina a qual o mesmo esteja sendo executado.

REFERÊNCIAS BIBLIOGRÁFICAS

ALECRIM, Emerson. Conhecendo o Servidor Apache (HTTP Server Project). Disponível em: <http://www.infowester.com/servapach.php>. Acesso em 28 de maio de 2016.

DANTAS, M. L. **Segurança da Informação:** uma abordagem focada em gestão de riscos. Olinda - PE: Livro Rápido, 2011.

DOMINGOS, Cesar. **BS779:** da tática á pratica em servidores Linux. Rio de Janeiro. Alta Books. 2006, p.240.

HILL, R. **Getting started with SSH security and configuration**. Disponível em: <https://www.ibm.com/developerworks/aix/library /ausshsecurity/>, 2011. Acesso em: 1 jun. 2016.

KERLINGER, Fred N. **Foundations of behavioral research**. New York: Holt, Rinehart &Winston, 1973.

LAKATOS, Eva Maria; MARCONI, Marina de Andrade. **Metodologia do trabalho científico**. São Paulo: Atlas, 1991.

LAURIE, Ben; LAURIE, Peter. **Apache:** the definetive guide. 2ª Ed. Bostom. O'Reilly, 1999, p. 536.

LAURIE, Ben; LAURIE, Peter. **Apache:** the definetive guide. 3ª Ed. Bostom. O'Reilly, 2002, p. 536.

MELO, Sandro. **Hardening em Linux.** Rio de Janeiro. Rede Nacional de Ensino a Pesquisa, 2014.

MOBILY, Tony. **Hardening Apache.** 1ª Ed. Estados Unidos. A Press Media, 2004.

Mozilla. Disponivel em: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies> Acesso em: 01 maio 2017.

NESSUS. Disponível em: < http://www.nessus.org > Acesso em: 7 jun. 2016

NIKTO. Disponível em: < http://cirt.net/code/nikto.shtml > Acesso em: 7 jun. 2016

REIS, F. A; VERBENA, M. F; JULIO, E. P. Hardening. Disponível em: < http://www.devmedia.com.br/hardening-artigo-revista-infra-magazine-1/20818>, 2011. Acesso em: 06 jun. 2017.

REY, Luis. **Planejar e Redigir Trabalhos Científicos**. Rio de Janeiro: Edgar Blucher, 1987.

SÊMOLA, Marcos. **Gestão da Segurança da Informação:** Uma visão executiva. São Paulo: Campus. 2003.

SILVA, Antonio Mendes. Segurança da Informação: Sobre a Necessidade de Proteção de Sistemas de Informações. **Revista Espaço Acadêmico**. Nº 42, 2004. Disponível em: < http://www.espacoacademico.com.br/042/42amsf.htm > Acesso em 15 maio 2016.

SKOUDIS, E. Step-by-step guide to computer attacks and effective defenses. 2ªed.Canada: CRAM101. 2012. THC.ORG.
THC

TAMASSIA, Roberto; GOODRICH, Michael. **Introdução à segurança de computadores.** 1ª Ed. Porto Alegre. Bookman, 2013, p. 568.

TURNBULL, James. **Hardening Linux.** Estados Unidos: Apress, 2005, p. 552.

Web Technology Surveys. Disponível em: < https://w3techs.com/technologies/overview/web_server/all> Acesso em: 01 maio 2017.