

---

Aurelio Grott, Gabriel Dominico, Victor Lucas de M. Mafra

*Análise e solução de vulnerabilidades em ambiente LAMP  
baseada em experimentação com Kali Linux*

---

Joinville  
2016

**UNIVERSIDADE DO ESTADO DE SANTA CATARINA**  
**BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**Aurelio Grott, Gabriel Dominico, Victor Lucas de M. Mafra**

**ANÁLISE E SOLUÇÃO DE VULNERABILIDADES EM**  
**AMBIENTE LAMP BASEADA EM EXPERIMENTAÇÃO**  
**COM KALI LINUX**

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina  
como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

**Charles Christian Miers**  
**Orientador**

Joinville, Junho de 2016

# **ANÁLISE E SOLUÇÃO DE VULNERABILIDADES EM AMBIENTE LAMP BASEADA EM EXPERIMENTAÇÃO COM KALI LINUX**

Aurelio Grott, Gabriel Dominico, Victor Lucas de M. Mafra

Este Trabalho de Conclusão de Curso foi julgado adequado para a obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Ciência da Computação Integral do CCT/UDESC.

Banca Examinadora

---

Charles Christian Miers - Doutor (orientador)

---

Charles Christian Miers - Doutor

---

Charles Christian Miers - Doutor

## Agradecimientos

*“We must know - we will know!”*

*- David Hilbert*

## Resumo

O **SDN!** (SDN) é uma tecnologia recente que permite ao administrador de redes um maior controle sobre uma rede. Tal controle é obtido através da separação entre o *Control Plane* e *Data Plane*, o que caracteriza uma SDN. Neste trabalho são conceituados diversos pontos chaves relativos ao assunto, tais como *OpenFlow* e os planos do SDN. Em seguida é descrita a ferramenta de simulação de redes *Mininet* e no fim do trabalho é descrito dois *benchmarks* com o objetivo de coletar dados para uma análise de desempenho. Tendo em vista que os *hardwares* que suportam SDN são relativamente recentes, o seu custo é muitas vezes proibitivo para um pequeno grupo de pesquisa ou um pesquisador independente, de tal forma que o uso de simuladores se torna indispensável para o desenvolvimento científico e tecnológico na área. Este trabalho tem como objetivo realizar um comparativo entre a transferência de dados de tamanho médio dentro de um ambiente simulado no *Mininet* e um ambiente utilizando o modelo tradicional de rede (*Data Plane* e *Control Plane* acoplados).

Palavras-chaves: SDN, Openflow, Mininet, software livre, transferência

# Abstract

SDN is a new technology which gives the network administrator greater power over his network. Such control is given through the separation between the control plane and the data plane, which characterizes an SDN. In this paper it is conceptualized several key points relative to SDN, such as *OpenFlow* and the SDN planes. In the following section it is described the networking simulation tool *Mininet* following by the description of two benchmarks that will be used with the objective of data collection for later analysis. Since the hardware that supports SDN are relatively new, their costs are very often prohibitively high for a small research group or an independent researcher, so that the use of simulators becomes indispensable for the technological and scientific development in the field. This paper has as its major objective to do a comparative between the transferring of medium sized data in a Mininet simulated environment and a traditional networking model (i.e. coupled Data Plane and Control plane).

Keywords: SDN, Openflow, Mininet, open source, trasnference

## List of Figures



## List of Tables

## Lista de Siglas e Abreviaturas

**ASP** Apache Software Foundation

**BD** Banco de dados

**CGI** Common Gateway Interface

**CVE** Common Vulnerabilities and Exposures

**DNS** Domain Name System

**DoS** Denial of Service

**GNU** Gnu Not Unix

**GPL** General Public License

**HTTP** HyperText Transfer Protocol

**LAMP** Linux Apache MySQL PHP

**PHP** Hypertext Preprocessor

**SMTP** Simple Mail Transfer Protocol

**SQL** Structured Query Language

**UDESC** Universidade do Estado de Santa Catarina

# Contents

<b>List of Figures</b>	<b>5</b>
<b>List of Tables</b>	<b>6</b>
<b>Lista de Siglas e Abreviaturas</b>	<b>7</b>
<b>1 Introdução</b>	<b>10</b>
<b>2 Conceitos</b>	<b>11</b>
2.1 LAMP . . . . .	11
2.1.1 CONCEITO . . . . .	11
2.1.2 HISTÓRICO . . . . .	12
2.1.3 APLICABILIDADE . . . . .	12
2.2 FUNCIONAMENTO E COMPONENTES BÁSICOS . . . . .	12
2.2.1 Linux . . . . .	12
2.2.2 Apache . . . . .	13
2.2.3 MySQL . . . . .	13
2.2.4 PHP . . . . .	14
2.3 FUNDAMENTAÇÃO DE ATAQUE E SOLUÇÕES . . . . .	15
2.4 FRAMEWORKS E SOLUÇÕES PARA ANÁLISE DE VULNERABILIDADES . . . . .	15
2.5 NORMAS, RECOMENDAÇÕES E BOAS PRÁTICAS PARA ANÁLISE DE VULNERABILIDADES . . . . .	16
<b>3 Conclusão</b>	<b>18</b>



# 1 Introdução

[illegible]

## 2 Conceitos

### 2.1 LAMP

#### 2.1.1 CONCEITO

LAMP é um conjunto de soluções, ou seja, uma lista dos componentes principais que são usados para implementar uma solução. LAMP é provavelmente o conjunto mais comumente usado para hostear páginas e aplicações web. Especialmente entre pequenas e médias empresas, pois os principais componentes são de código aberto e não têm custo. Um bom exemplo para visualizar como cada componente do LAMP funciona seria um visitante entrando em um site pelo seu browser, que irá enviar um pedido para um servidor web com todos os componentes LAMP instalados e sendo executados que, por sua vez, irá executar alguns comandos e devolver para o browser um arquivo em HTML. Um servidor web é um tipo de computador, logo ele necessita de um sistema operacional que será a base na qual todos os outros componentes possam ser executados. Nesse caso é uma distribuição Linux, que é o "L" em LAMP. Agora, para receber e manipular todos os pedidos do visitante precisa-se de um servidor web que, nesse caso, é o Apache (o "A" em LAMP). Se o visitante faz a requisição de um arquivo, o Apache procura no servidor pelo o arquivo e envia devolta para o browser do visitante. O "P" significa PHP que também pode ser Perl ou Python, mas isso não é muito comum. PHP é uma linguagem de programação que é utilizada para gerar páginas dinâmicas. Quando um visitante faz a requisição de um arquivo em PHP o Apache não apenas pega esse arquivo e envia de volta para o browser como ele faz com outros tipos de arquivos. Em vez disso, ele primeiro envia o arquivo para o PHP, que executa o código no arquivo e gera um arquivo HTML para a página web. O HTML gerado pelo PHP volta para o Apache e depois para o browser do visitante. O "M" significa MySQL que é um SGBD usado para armazenar as informações de um site. Frequentemente os dados necessários para gerar o arquivo PHP depende em quem o visitante é, como o nome dele, número da conta ou histórico de pedidos. O banco de dados permite que essas informações fiquem armazenadas.

### 2.1.2 HISTÓRICO

O conceito principal em volta do LAMP (um servidor web sem custo) foi possível no início de 1995 quando a CERN httpd introduziu o conceito de CGI, que tornou possível aos servidores executar códigos para criar páginas dinâmicas. Linux, CERN httpd e linguagens de programação server-side como Perl estavam disponíveis gratuitamente, mas conseguir um banco de dados gratuito só foi possível mais tarde, no final daquele ano, com o lançamento de Postgre95.

Ainda em meados de 1995, o servidor HTTP Apache e PHP foram lançados, tornando o conjunto de aplicações LAPP (Linux Apache PostgreSQL e Perl) possível. Finalmente em 1996, MySQL foi lançado e, com ele, o conjunto LAMP completo fez-se possível. A popularidade do LAMP cresceu rapidamente nos anos 90, já que muitas empresas, por razões monetárias, utilizavam softwares de código aberto em suas páginas da web.

### 2.1.3 APLICABILIDADE

## 2.2 FUNCIONAMENTO E COMPONENTES BÁSICOS

### 2.2.1 Linux

Linux poderia ser descrito como um sistema operacional similar a qualquer outro, como Windows e OS X. Porém tem algo que o destaca dos demais, desde sua origem em 1991, e que é o motivo do sistema ter crescido e ganhado uma grande força na computação, atualmente presente em lugares desde a bolsa de valores de Nova York e supercomputadores à telefones celulares e computadores pessoais, o Linux é um software livre desenvolvido de maneira colaborativa (PROFFITT, 2009). Mais de 1.000 desenvolvedores de pelo menos 100 diferentes companhias, contribuíram para cada versão do kernel sob a licença General Public License (GPL) que é baseada em quatro liberdades (FSF, 2016):

- A liberdade de executar o programa como quiser, para qualquer propósito;
- A liberdade para estudar como o programa funciona, e alterá-lo para que ele execute como você queira. Ter acesso ao código fonte é necessário para tal;

- A liberdade para redistribuir cópias para ajudar o próximo; e
- A liberdade para distribuir cópias de suas versões modificadas para outros. Fazendo isso você concede à comunidade a chance de se beneficiarem de suas alterações. Ter acesso ao código fonte é necessário para tal.

Por esses motivos o Linux tem sido bem-sucedido, particularmente como plataforma de servidor: até mesmo em organizações que confiam veemente em sistemas operacionais comerciais como Microsoft Windows, o Linux aparece frequentemente em papéis infraestruturais, como em *gateways* de Simple Mail Transfer Protocol (SMTP) e servidores Domain Name System (DNS) devido a sua confiança, segurança, baixo custo e a qualidade excepcional das aplicações do servidor (BAUER, 2005).

### 2.2.2 Apache

Atuando como servidor HyperText Transfer Protocol (HTTP) no sistema Linux Apache MySQL PHP (LAMP), o Apache é o servidor *web* mais popular na Internet desde Abril de 1996 (ASF, ). O Apache é um projeto de código livre (sob a licença GPL) da Apache Software Foundation (ASF), o qual tem como objetivo manter um seguro, eficiente e extensível servidor que provê serviços HTTP de acordo com os padrões HTTP atuais.

### 2.2.3 MySQL

O Banco de dados (BD) MySQL, foi projetado com base no mSQL, o qual tinha muitos problemas, como não ser rápido e flexível o suficiente para o uso dos usuários, com isso a necessidade de um novo BD foi aumentando e com base nesse conceito foi desenvolvido o que hoje conhecemos como MySQL, por isso é o BD do servidor LAMP.

Um BD pode ser definido como uma coleção de dados. Porém para conseguir acessar os dados armazenados nesse sistema, teve-se a necessidade de criar algum tipo de gerenciador, sendo o MySQL um dos mais usados. Algumas características (MySQL, 2013a) desse sistema podem ser vistas abaixo:

- **Banco de dados relacional:** a principal diferença desse tipo de BD para os outros é que os dados são guardados em pequenas tabelas de uma forma que seu acesso seja da forma mais eficiente o possível.



- **Open Source:** esse termo corresponde que qualquer pessoa pode modificar o *software* do jeito que preferir, podendo ajustá-lo conforme a sua necessidade.
- **Rápido, confiável, escalável e fácil de usar:** como foi criado para atender a grandes quantidades de dados de uma forma mais rápida que seus concorrentes, foi apenas lógico que se tornasse um dos mais rápidos BD. Portanto começou a ser utilizado em grande escala, consequentemente a segurança foi aumentando juntamente com sua escalabilidade para atender a demanda de usuários.

Contudo, mesmo com medidas de seguranças sendo tomadas, precisamos ainda tomar algumas atitudes para dificultar que seu BD seja acessado por pessoas não autorizadas, alguns métodos básicos que ajudam a proteger são descritas abaixo (MYSQL, 2013b):

- Não prover acesso a ninguém para a tabela usuário do BD MySQL.
- Não guardar senhas sem algum tipo de função *hash* (algoritmo usado para transformar sua senha para uma *string* ilegível).
- Crie senhas aleatórias, porém de fácil memorização.
- Invista em um *firewall*, protegem pelo menos 50% dos ataques feitos contra seu *software*.
- Sempre criptografe os dados que precisam ser enviados pela internet.

### 2.2.4 PHP

O Hypertext Preprocessor (PHP) foi criado em 1994 por Rasmus Lerdof, o projeto inicial era um simples conjunto de Common Gateway Interface (CGI)s binários escritos na linguagem de programação C, usados para rastrear as visitas ao seu *site*. Com o tempo, otimizações foram sendo feitas e funcionalidades adicionadas. Sendo lançado em 1998, o PHP 3.0 foi a primeira versão que contém traços do PHP de hoje em dia, incluindo o suporte a programação orientada a objeto. Porém essa versão tinha muita dificuldade em processar aplicações complexas, foi com base nessa premissa que foram lançadas as versões 4.0 e 5.0 (Julho de 2004), principalmente para melhorar seu antecessor e acrescentar dezenas de novos recursos.

Usado principalmente para desenvolvimento *web*, é um *script open source* de uso geral. Podemos especificar em quais áreas os *scripts* PHP são mais utilizados (PHP, 2016), como:

- **Scripts no lado do servidor.** Podendo acessar os resultados do seu programa com um navegador web.
- **Scripts de linha de comando.** Executar os scripts sem um servidor ou navegador, apenas necessita de um interpretador PHP.
- **Escrever aplicações desktop.** Não é a melhor linguagem para se desenvolver aplicações desktop, porém para um programador experiente o PHP tem alguns recursos avançados que permitem escrever esse sistema.

Uma característica é a escalabilidade que o PHP possui, podendo ser utilizado na maioria dos sistemas operacionais e servidores *web*. Com isso ele vem sendo aplicado cada vez mais em servidores LAMP, por suas várias extensões que facilitam a conectividade com diversos banco de dados.

## 2.3 FUNDAMENTAÇÃO DE ATAQUE E SOLUÇÕES

Testes de penetração de servidores *web* não devem ser confundidos com ataques maliciosos. Apesar de possuírem rotinas parecidas, possuem objetivos distintos. Ataques maliciosos são realizados para roubar informações, causar indisponibilidade de serviços (Denial of Service (DoS)) ou qualquer outro evento indesejado ao responsável pelo servidor. Já um teste de penetração consiste em um processo autorizado, programado e sistemático onde se faz uso de vulnerabilidades conhecidas para realizar tentativas de invasão à um servidor, rede ou conteúdos de aplicações.

Testes de penetração podem ser executados de mais de uma maneira para propor mais de um ponto de vista sob a mesma organização. Para tal, existem dois tipos de testes (não exclusivos) que podem ser conduzidos (SANS, 2002).

**Teste interno de penetração** Realizado com o objetivo de identificar vulnerabilidades com acesso físico ou exposição a engenharia social. Servem para determinar quais

vulnerabilidades existem no sistema interno, acessível somente à pessoas autorizadas com acesso a rede interna da organização.

**Teste externo de penetração** Realizado com o objetivo de identificar vulnerabilidades presentes através de conexões que foram estabelecidas através da conexão entre a organização e a Internet (através do *firewall* ou *gateway*).

## 2.4 FRAMEWORKS E SOLUÇÕES PARA ANÁLISE DE VULNERABILIDADES

*Frameworks* para análise de vulnerabilidades são ferramentas para facilitar a detecção de vulnerabilidades em determinado sistema/página *web*.

Uma delas é o *Nikto*, o qual é uma ferramenta de segurança específica para *website*, podendo verificar mais de 6.000 testes. Alguns dos principais problemas que podem ser achados são: Arquivos perigosos, serviços mal configurados, *scripts* vulneráveis, entre outros.

Outra ferramenta que pode ser citada é o *Vega*, com objetivos semelhantes ao *Nikto*, porém com dois modos de operação (SUBGRAPH, 2014):

- **Scanner automático:** rastreia automaticamente por *websites*, extraindo links, e executa módulos em possíveis pontos de vulnerabilidades.
- **Proxy de interceptação:** permite análises detalhados sobre a interação navegador-aplicação.

Pode-se citar ainda o OpenVAS, uma ferramenta para varredura e gerenciamento de vulnerabilidades. Podendo ser subdividido em duas arquiteturas (OPENVAS, 2016):

- **Rastreados OpenVAS:** executa os testes de vulnerabilidades.
- **Gerenciador OpenVAS:** consolida a varredura de vulnerabilidade em uma solução completa de gerenciamento de vulnerabilidade.

## 2.5 NORMAS, RECOMENDAÇÕES E BOAS PRÁTICAS PARA ANÁLISE DE VULNERABILIDADES

Vulnerabilidade pode ser definida como um erro no *software* que permita ao *hacker* ganhar acesso ao sistema ou a rede interna (CVE, 2016). Primeiro monitora-se com qual frequência determinada vulnerabilidade pode ocorrer, e o quão prejudicial é para o programa. Quando é descoberto alguma vulnerabilidade com os testes realizados, verifica-se em algum banco de dados de vulnerabilidades, sendo o Common Vulnerabilities and Exposures (CVE) uma boa fonte para conhecer um pouco mais sobre o problema encontrado.

Existem também os escaneadores de vulnerabilidade, os quais são *softwares* que ajudam na identificação de possíveis problemas que podem facilitar a corrupção do sistema, identificando, como por exemplo: versões de *softwares* desatualizadas, configurações falhas. Segundo (STANDARDS; TECHNOLOGY, 2008), esses escaneadores podem:

- **Verificar políticas de segurança.**
- **Prover informações sobre alvos para testes de penetração.**
- **Fornecer informações sobre como diminuir as vulnerabilidades descobertas.**

Contudo, mesmo com esses facilitadores, ainda existem alguns problemas a serem vencidos, como quando pequenos riscos de vulnerabilidade são analisados de uma forma isolada, porém quando observados como um conjunto podem trazer um grande problema ao sistema, isso pode ser dito como uma das principais falhas desses escaneadores, pois conseguem apenas analisar ameaças individuais e não em um escopo específico.

Outro meio de se prevenir contra ataques pode ser feito com uma simples revisão do código fonte quando está sendo implementado, não apenas no seu estado final. Também é viável realizar testes unitários para verificar a consistência presente no código.



# Bibliography

ASF. *The Apache HTTP Server Project*. <https://httpd.apache.org/>. (Accessed on 04/08/2016).

BAUER, M. *Linux server security*. Sebastapol, CA Cambridge: O'Reilly, 2005. ISBN 978-0-596-00670-9.

CVE. *CVE - Frequently Asked Questions*. 2016. <http://cve.mitre.org/about/faqs.html>. (Accessed on 04/08/2016).

FSF. *What is free software? - GNU Project - Free Software Foundation*. January 2016. <http://www.gnu.org/philosophy/free-sw.en.html>. (Accessed on 04/08/2016).

MYSQL. *MySQL :: MySQL 5.7 Reference Manual :: 1.3.1 What is MySQL?* April 2013. <https://dev.mysql.com/doc/refman/5.7/en/what-is-mysql.html>. (Accessed on 04/09/2016).

MYSQL. *MySQL :: MySQL 5.7 Reference Manual :: 6.1.1 Security Guidelines*. April 2013. <http://dev.mysql.com/doc/refman/5.7/en/security-guidelines.html>. (Accessed on 04/09/2016).

OPENVAS. *OpenVAS - About OpenVAS Software*. 2016. <http://www.openvas.org/software.html>. (Accessed on 04/09/2016).

PHP. *PHP: O que o PHP pode fazer? - Manual*. March 2016. [http://php.net/manual/pt\\_BR/intro-whatcando.php](http://php.net/manual/pt_BR/intro-whatcando.php). (Accessed on 04/09/2016).

PROFFITT, B. *What Is Linux: An Overview of the Linux Operating System — Linux.com — The source for Linux information*. April 2009. <https://www.linux.com/learn/what-linux-overview-linux-operating-system>. (Accessed on 04/09/2016).

SANS. *Penetration Testing - Is it right for you?* 2002. <https://www.sans.org/reading-room/whitepapers/testing/penetration-testing-you-265>. (Accessed on 04/09/2016).

STANDARDS, N. I. of; TECHNOLOGY. *Technical Guide to Information Security Testing and Assessment*. september 2008. <http://csrc.nist.gov/publications/nistpubs/800-115/SP800-115.pdf>. (Accessed on 04/08/2016).

SUBGRAPH. *About Vega*. 2014. <https://subgraph.com/vega/documentation/about-vega/index.en.html>. (Accessed on 04/09/2016).