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

Agradecimentos

"We must know - we will know!"

- David Hilbert

Resumo

resumo resumo

Palavras-chaves:

Abstract

abstract abstract

Keywords:

Lista de Figuras

Lista de Tabelas

Lista de Siglas e Abreviaturas

ASP Apache Software Foundation

BD Banco de dados

CERN Conseil Européen pour la Recherche Nucléaire

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

InP Infrastructure Provider

LAMP Linux Apache MySQL PHP

LAPP Linux Apache PostgreSQL PHP

PHP Hypertext Preprocessor

SGBD Sistema de Gerenciamento de Banco de Dados

SMTP Simple Mail Transfer Protocol

SQL Structured Query Language

UDESC Universidade do Estado de Santa Catarina

Sumário

1 Introdução

Introdução introdução

2 Conceitos

2.1 LAMP

2.1.1 CONCEITO

Linux Apache MySQL PHP (LAMP) é conjunto de soluções em forma de uma lista dos componentes centrais usados na implementação uma aplicação web. Seu uso é predominante entre pequenas e médias empresas, já que tais componentes são de código aberto e não têm custo. LAMP pode ser definido pelo seus componentes, sendo eles: **L**inux (sistema operacional), **A**pache (serividor web), **M**ySQL(software de banco de dados) e **P**HP (linguagem de programação), que também pode se referir a Perl ou Python, apesar de não ser muito comum (mais detalhes serão vistos no tópico 2.2).

Um exemplo do funcionamento de cada componente do LAMP em um servidor *web* se dá por um visitante entrando em um *site* pelo seu navegador, o qual irá enviar um pedido para o servidor, onde todos os componentes LAMP estão instalados e sendo executados. Este servidor pode então prover ao usuário, interação com banco de dados através de *scripts* Hypertext Preprocessor (PHP) através de formulários enviados ao navegador pelo protocolo HyperText Transfer Protocol (HTTP).

2.1.2 HISTÓRICO

O conceito principal em volta do LAMP (um servidor *web* sem custo) foi possível no início de 1995 quando a Conseil Européen pour la Recherche Nucléaire (CERN) httpd introduziu o conceito de Common Gateway Interface (CGI), que tornou possível aos servidores executar códigos para criar páginas dinâmicas (ROBINSON, 2013). Linux, CERN httpd e linguagens de programação do lado do servidor como Perl estavam disponíveis gratuitamente, mas conseguir um banco de dados gratuito só foi possível mais tarde, com o lançamento de Postgre95.

Ainda em meados de 1995, o servidor HTTP Apache e PHP foram lançados, compondo então o conjunto de aplicações Linux Apache PostgreSQL PHP (LAPP). Final-

mente 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 *web*.

2.1.3 APLICABILIDADE

Sistemas LAMP podem ser encontrados em diversos níveis de aplicações. Quem utiliza um sistema operacional com kernel Linux em seu computador pessoal e deseja desenvolver uma aplicação *web* pode fazê-la através da instalação dos componentes faltantes, ou seja, o Apache como servidor HTTP, MySQL como banco de dados local e PHP como linguagem de *script* de servidor local. Após essas configurações o sistema está pronto para exibir *web sites* criados localmente, os quais poderam conter formulários a serem processados e enviados a outro servidor ou armazenados na própria máquina. Com o ajuste de alguns detalhes (configurações de *firewall* e permissões), este *site* desenvolvido localmente pode se tornar acessível à usuários externos, fazendo com que o sistema se assemelhe à um servidor contratado.

Não se limitando a projetos de desenvolvimento local, o LAMP se encontra frequentemente em servidores providos por Infrastructure Provider (InP)s, onde a principal diferença de um sistema local é a falta de interface gráfica, para alguns usuários experientes isso não é um problema e então a instalação dos componentes se dá de forma muito parecida, através de linhas de comando.

2.2 FUNCIONAMENTO E COMPONENTES BÁSICOS

Para garantir o bom funcionamento do sistema, é necessário a boa comunicação e integração entre todos os componentes do LAMP. Apesar de serem serviços independentes, juntos eles constituem um servidor *web* capaz de executar *scripts* (do lado do servidor) e armazenar dados. Detalhes dos componentes do sistema se darão a seguir.

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 licensa 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 papeis 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 execepcional das aplicações do servidor (BAUER, 2005).

2.2.2 Apache

Atuando como servidor HTTP no sistema 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 licensa GPL) da Apache Software Foundation (ASP), 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, um Sistema de Gerenciamento de Banco de Dados (SGBD) que viabiliza programação em Structured Query Language (SQL) presente no 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 desse sistema podem ser vistas abaixo (MYSQL, 2013a):

- **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 o BD seja acessado por pessoas não autorizadas, alguns métodos básicos que ajudam a proteger são descritos 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 aleátorias, 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 PHP foi criado em 1994 por Rasmus Lerdof, o projeto inicial era um simples conjunto de CGIs 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 a 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 deles é o *Nikto*, o qual é uma ferramenta de segurança específica para *website*, podendo verificar: mais de 6.000 ameaças em potencial localizadas em arquivos ou programas, mais de 1.200 versões desatualizadas de servidores e mais de 270 problemas em servidores específicos (SULLO; LODGE,). 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 detalhadas 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 permite que uma pessoa não autorizada ganhe 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, ocorre a verificação em um 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, esses escaneadores atuam de forma local. Ao identificar de maneira isolada cada falha e vulnerabilidade muitas vezes não passam a proporção real do problema, que se daria em um único contexto com todas as falhas identificadas. Ou seja, várias pequenas vulnerabilidades podem, juntas, proporcionar um grande risco ao servidor.

Outro meio de se prevenir contra ataques pode ser feito com uma simples revisão dos códigos fontes quando estes estão sendo implementados, não apenas nos seus estados finais. Também é viável realizar testes unitários para verificar sua consistência.

3 Conclusão

Conclusão conclusão

Referências Bibliográficas

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. (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).

ROBINSON, D. *The Common Gateway Interface (CGI) Version 1.1*. RFC Editor, mar. 2013. RFC 3875. (Request for Comments, 3875). Disponível em: <https://rfc-editor.org/rfc/rfc3875.txt>.

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).

SULLO, C.; LODGE, D. *Nikto2 — CIRT.net*. https://www.cirt.net/Nikto2. (Accessed on 04/11/2016).