UNIVERSIDADE
ESTADUAL DE LONDRINA

Relatório de estágio curricular obrigatório

**Rafael de Paula Herrera**

# Uma ferramenta para a análise de logs de firewall

**Londrina**

**2009**

**Rafael de Paula Herrera**

# Uma ferramenta para a análise de logs de firewall

Trabalho apresentado à Universidade Estadual de Londrina, como parte do requisito para a obtenção do título de Bacharel em Ciência da Computação, sob orientação do Prof. Dr. Mario Lemes Proença Jr.

Orientador:
Mario Lemes Proença Jr.

Universidade Estadual de Londrina

Londrina

2009

**Rafael de Paula Herrera**

# Uma ferramenta para a análise de logs de firewall

---

Prof. Dr. Mario Lemes Proença Jr.
Universidade Estadual de Londrina

---

Prof. Dr. Alan Salvany Felinto
Universidade Estadual de Londrina

---

Prof. Dr. Jacques Duílio Brancher
Universidade Estadual de Londrina

Londrina, 16 de Novembro de 2009

*À todas as noites que passei em claro, por conta de minha graduação.*

# Agradecimentos

Aos meus pais, Jonas e Maria Alice, pela oportunidade de passar por este planeta, chamado por nós seres humanos de Terra. Tenho plena consciência que sem seu apoio durante os momentos mais difíceis de minha vida, não teria a chance de buscar pelos meus sonhos, tão pouco teria condições de contar estruturalmente com bases sólidas para meu desenvolvimento sócio-cultural. A estes dois indivíduos dedico todo o meu amor, respeito e profunda admiração.

À minha irmã, Adriana, por ter sido, sempre, uma grande companheira ao longo de minha trajetória. Mesmo que desde o início de minha graduação nosso contato tenha sido frequentemente interrompido, busco em suas grandes obras artísticas a inspiração para que os dias não se façam rotineiros e que, sejam sempre repletos pela incessante busca do novo-criativo.

À minha avó paterna, Eurides, que em vida sempre apoiou minhas decisões, sendo um verdadeiro exemplo de perseverança, garra e força de opinião, fato que me impulsiona até os dias de hoje, na busca pelos meus objetivos.

Ao meu avô materno, Hélio, pelos seus sábios ensinamentos de vida, tendo ajudado a moldar minha personalidade tal como ela é e, acima de tudo, por ser uma das maiores amizades que construí até hoje, em toda minha vida.

Aos meus amigos, por me acompanharem sempre almejando o melhor, trazendo muitos momentos de alegria e serem escolhidos por mim, como membros genuínos de minha família.

Ao Bodão do passado, por me fazer orgulhoso e, não menos importante, ao Bodão do futuro, por realizar meus sonhos.

*"Miss Anders... I didn't recognize you with your clothes on"*

**James Bond - Moonraker (1979)**

# Resumo

Este trabalho expõe uma das grandes ameaças atuais aos servidores baseados nas famílias Unix, os ataques de força bruta que utilizam dicionários contra o serviço *Secure Shell* (*SSH*). Com tal premissa, foi desenvolvida uma ferramenta *Web* capaz de identificar as origens dos ataques, gerar relatórios e regras que podem ser utilizadas em Firewalls para o filtro destas ameaças.

**Palavras-chave: Segurança, Unix, Firewall, SSH, Força Bruta.**

# Abstract

This work exposes one of the actual greatest treats for the servers which are based on *Unix* families, the brute force attacks wich uses dictionaries against the *Secure Shell* (*SSH*) service. With such premisse, were developed a *Web* tool which can identify the attack origins, generate reports and rules that can be used in Firewalls to filter these treats.

**Keywords: Security, Unix, Firewall, SSH, Brute Force.**

# Sumário

# Lista de Figuras

# Lista de Tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| AUP | Agile Unified Process |
| CeWL | Custom Word List generator |
| CLI | Command Line Interface |
| CSS | Cascading Style Sheet |
| FBI | Federal Bureau of Investigation |
| GNU | GNU Not Unix |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IP | Internet Protocol |
| MVC | Model View Controller |
| NIPC | National Infrastructure Protection Center |
| PHP | PHP: Hypertext Preprocessor |
| RUP | Rational Unified Process |
| SGBD | Sistema Gerenciador de Banco de Dados |
| SSH | Secure SHell |
| URL | Uniform Resource Locator |
| WWW | World Wide Web |
| XML | Extensible Markup Language |

# Introdução

Desde o ano 2000, o *SANS Institute*, em conjunto com o *NIPC*, uma divisão pertencente ao *FBI*, vem compilando uma lista com as ameaças mais recorrentes aos computadores interconectados em rede, especialmente à *Internet*. A primeira lista fora classificada como *"As dez vulnerabilidades de Internet mais críticas"* (INSTITUTE, 2002), sendo expandida durante os anos seguintes para *"Os vinte maiores problemas, ameaças e riscos da Internet"* (INSTITUTE, 2007). Os esforços por parte de seus responsáveis, vêm sendo direcionados até os dias de hoje no intuito cobrir os pontos mais frágeis e comuns aos ambientes interconectados em rede.

Dentre tais ameaças, podem ser ressaltadas como mais marcantes, as que se relacionam aos seguintes tópicos:

- Sistemas operacionais podem apresentar algumas falhas que uma vez exploradas, podem acarretar na distribuição massiva de *worms* em sistemas conectados à Internet.
- Falhas são frequentemente encontradas em aplicações-cliente, tais como navegadores, ferramentas pertencentes à suites *Office* e reprodutores de mídia. Uma vez exploradas, podem levar à corrupção do sistema hospedeiro.
- Funcionários de corporações e instituições com acesso à Internet estão expostos à diversos riscos oferecidos por páginas da *WWW*, *World Wide Web*, maliciosamente manipuladas, sendo a porta de entrada para *malwares* nas redes privadas.
- Falhas encontradas tanto em sistemas *Web open-source* quanto em sistemas *Web* proprietários, podem conduzir à transformação de sites confiáveis em versões manipuladas maliciosamente pelos atacantes.
- Muitas configurações-padrão de sistemas operacionais e serviços permitem que tais ca-

racterísticas sejam exploradas, sendo comprometidos via ataques de força-bruta baseados em dicionários.

- Os interessados em atacar as instituições, obtém informações sensíveis via métodos cada vez mais inovadores, sendo uma tarefa essencial, que se estabeleça um controle mais aguçado de como as informações deixam os limites das organizações.

Neste trabalho, será abordada uma forma de ameaça específica relacionada aos ataques de força-bruta, os ataques de dicionário contra servidores baseados na família *Unix* e que possuem o serviço *SSH* rodando.

Tais ataques são promovidos com o auxílio de ferramentas automatizadas, que utilizam um conjunto de combinações de usuários e senhas comuns, catalogados em arquivos chamados *dicionários*. Quando reincidentes, essas tentativas são armazenadas em forma de *logs* nos servidores.

Observando a importância desta forma de ataque, foi proposto como estágio, o desenvolvimento de uma ferramenta *Web*, capaz de gerar relatórios que identifiquem as tentativas de acesso que foram negadas ao serviço *SSH*. A partir da análise dos dados identificados pela ferramenta, pode-se realizar uma melhoria nos filtros de *firewall* presentes nos servidores, através da elaboração de regras específicas de *Iptables*.

Este trabalho cobre os fundamentos teóricos e práticos sobre a problemática envolvida, assim como a implementação da ferramenta que auxilia em sua resolução, tendo seus capítulos divididos da seguinte maneira:

- *Capítulo 1*: descreve ataques de dicionário, empregados contra os servidores que rodam o serviço *SSH*.
- *Capítulo 2*: apresenta a ferramenta proposta, em conjunto como seus aspectos tecnológicos.
- *Capítulo 3*: mostra a engenharia empregada no desenvolvimento do software.

# 1 Ataques de dicionário

Existem inúmeras formas de ataque empregadas contra servidores e computadores convencionais e muitas delas são descritas pelo *SANS Institute* em suas publicações periódicas (INSTITUTE, 2007).

Muitas das ameaças levantadas, são recorrentes exclusivamente aos sistemas *Desktop*, muitas vezes suscetíveis a variadas combinações de *worms*, *malwares* e *vírus*, o que afeta somente aos usuários finais dos sistemas. Também existem aquelas empregadas quase que exclusivamente contra os servidores, tais como ataques *Denial of Service* em rede, *Buffers Overflows* em rede, XSS e Ataques de dicionário por força bruta sempre estiveram listados entre os expressivos (INSTITUTE, 2002) (INSTITUTE, 2007).

Quando servidores são afetados por alguma forma de ataque, os resultados são muito mais drásticos, quando comparados aos causados contra computadores *Desktops*. Como sua finalidade é prover um ou mais serviços a uma variada gama de usuários e até mesmo a outros sistemas, quando os servidores são comprometidos os prejuízos podem ser, em muitas das vezes, incalculáveis. Devido sua importância, o tema escolhido para este trabalho, foram os Ataques de dicionário por força bruta lançados contra servidores da família *Unix*, restringindo-se aos que possuam o serviço *SSH*[1] rodando como *daemon*[2].

Caso o atacante deseje tomar o controle das operações em servidores, seu sucesso pode depender da exploração de algum serviço que seja capaz de disponibilizar acesso à linha de

[1] *SSH* é um acrônimo para **S***ecure* **SH***ell*. Sua variante mais utilizada atualmente, é o *OpenSSH*, sendo padrão na grande maioria das distribuições baseadas no *Kernel* do *GNU/Linux*. *Website* do projeto disponível em: `http://www.openssh.com/`

[2] *Daemon* é uma expressão frequentemente usada para referir-se a processos que rodam em *background*, mas que proveêm algum nível de interação com os clientes que utilizem seus protocolos de comunicação em rede, sendo para tanto, processos que são a base do funcionamento de tecnologias em servidores.

comando, *Shell*[3] do Sistema Operacional implantado. Uma vez que o acesso tenha sido garantido, deve-se avaliar os privilégios obtidos em sua investida e, caso estes sejam escassos, o passo seguinte para a tomada do sistema, certamente será a utilização de técnicas que possam escalar seus privilégios até às contas administrativas. Isso é possível, através da exploração de vulnerabilidades conhecidas em versões desatualizadas de programas e serviços que por ventura estejam em execução no sistema alvo, ou mesmo que estejam meramente disponíveis para utilização, mas que possam ainda sim, trazer riscos. (PROVOS *et al.*, 2003)

Historicamente, têm-se utilizado o serviço *SSH* para acesso remoto e controle de servidores. Seu uso foi favorecido em comparação ao antigo *Telnet*[4], por oferecer um canal seguro entre o modelo cliente/servidor, onde as informações trafegam criptografadas pela rede. No entanto, o serviço *SSH*, assim como todas as tecnologias, não é infalível à falhas de segurança, tendo como uma das principais ameaças, os ataques de dicionários por força bruta.

Existem inúmeras ferramentas que lançam, automaticamente, ataques de dicionário por força bruta via *SSH*, como a *SSH Brute Forcer*[5]. Um exemplo comum de saída padrão, gerado pela sua utilização contra um alvo arbitrário e sucetível a falhas, seria:

```
bode@bodacious:~/pacotes/seguranca$ ./sshbrute 127.0.0.1 test wordlist


          d3hydr8:darkc0de.com sshBrute v1.0
       ----------------------------------------


[+] Loaded: 7 words
[+] Server: 127.0.0.1
```

[3] O *Shell* faz parte da *CLI*, *Command Line Interface* do Sistema Operacional, sendo capaz de executar ações sem que seja necessário o modo gráfico, de maneira mais rápida e mais eficiente, com uso de um conjunto determinado de comandos e ferramentas padrão.

[4] *Telnet* é um acrônimo para **Tel***etype* **Net***work*, trata-se de um protocolo que permite a comunicação interativa bidirecional em rede, utilizado em conexões cliente/servidor. Os dados trocados não trafegam criptografados, sendo hoje considerada uma maneira insegura de se acessar informações em servidores remotos.

[5] *SSH Brute Forcer* é uma ferramenta que lança, automaticamente, ataques por força bruta via em servidores, levando em consideração um dicionário de palavras contendo as ocorrências mais comuns de usuários/senhas. Escrita na linguagem de programação *scripting Python*, pode ser encontrada em `http://packetstormsecurity.org/Crackers/sshbrute.py.txt`. Para seu funcionamento, é necessária a instalação do módulo *Python Pexpect*, disponível em `http://pexpect.sourceforge.net/`

```
[+] User: test
[+] BruteForcing...

Trying: test
Trying: aaaa
Trying: 123
Trying: 123456

uname -a
Linux bodacious 2.6.29.6-smp #2 SMP Mon Aug 17 00:52:54 CDT 2009 i686
AMD Athlon(TM) XP 2200+ AuthenticAMD GNU/Linux

        [!] Login Success: test 123456
```

Por meio deste exemplo simples, pôde ser observado o funcionamento básico de um típico ataque de dicionário para SSH, que iterativamente testa sob um usuário específico, várias senhas contidas em um arquivo específico, chamado *Wordlist*, que contém as recorrências mais comuns destas em sistemas vulneráveis. Este ataque foi lançado sob o computador local, que propositalmente continha o usuário *test*, cuja senha fora identificada como *123456* dentre as demais, contidas na *Wordlist*.

Com pouco esforço, pode-se automatizar ainda o processo envolvido nos ataques como o que o *SSH Brute Forcer* realiza. É possível, por exemplo, cruzar os dados de uma só *Wordlist* com os mesmos iterados, tendo assim, uma combinação completa entre usuários e senhas. Tem como vantagem, o teste exaustivo de todas as combinações, mas apresenta como principal desvantagem, um grande aumento no tempo de execução do processo, por ser definido em termos de uma função de ordem quadrática, denotada em notação assintótica, por $O(n^2)$.

## 1.1 Ataques de dicionário para SSH

Um ataque de dicionário, é assim classificado devido à utilização de um arquivo contendo inúmeras combinações de usuários e senhas para fins específicos, tal arquivo é comumente referido como *Wordlist*. No caso do serviço *SSH*, utilizam-se métodos automatizados para se obter o sucesso em ataques desse tipo.

Ferramentas como *ssh-dictattack*[6] e *SSH Brute Forcer*, abordado anteriormente, fazem uso de dicionários, lançando na rede, sucessivas tentativas de se estabelecer conexões com servidores que dispõe de um *daemon SSH* rodando.

Os servidores são escolhidos através de *scans* previamente realizados nas redes (VYKOPAL *et al.*, 2009). Uma vez identificado o serviço *SSH* rodando em um servidor, este é imediatamente agregado à listas que contém os possíveis alvos. Também existe o caso onde o atacante escolhe cuidadosamente suas vítimas, existindo assim, interesse específico nas mesmas, em geral relacionado ao alto poder computacional de instituições públicas e privadas. É também muito comum, que computadores comprometidos sejam integrados às redes zumbis e controlados remotamente, de modo que lancem diversos tipos de *scans* em rede na busca de máquinas também comprometidas e/ou que rodem serviços que sejam interessantes aos atacantes, como é o caso do *SSH*.

Para que os dicionários sejam gerados, contendo em milhares de entradas de usuários e senhas, utiliza-se ferramentas intermediárias e totalmente automatizadas, como é o caso da chamada *CeWL*[7].

Uma vez que o serviço *SSH* esteja sofrendo com ataques de dicionário, são gravadas as informações relacionadas às fontes dos ataques, mostrando em logs no servidor[8], os endereços *IP* responsáveis pelos ataques e a data em que cada incidente ocorreu.

[6] O código-fonte da ferramenta *ssh-dictattack* está disponível em `http://ircsex.de/download/utils/ssh-dictattack.c`

[7] *CeWL* é um acrônimo para *Custom Word List generator*. É uma ferramenta para a geração automática de *Wordlists*, que podem ser utilizadas em diversos ataques baseados em dicionários. O site do projeto é disponível em: `http://www.digininja.org/projects/cewl.php`

[8] O referido log, é em geral, o *var/log/messages* como padrão para os sistemas baseados em *Unix*.

Em sistemas da família Unix, tais registros de log são armazenados, por padrão, no arquivo */var/log/messages*. Uma maneira nativa, de se obter informações sobre as investidas contra o sistema em questão, é através da interpretação dos dados armazenados, estabelecendo-se filtros baseados em expressões regulares. Desse modo, com a formação de uma expressão satisfatória, os dados filtrados correspondem exatamente às tentativas de entrada no sistema e, revelam na maioria das vezes, a presença de usuários não-cadastrados, mas encontrados nos dicionários dos atacantes.

## 1.2 Uma ferramenta auxiliar

É de grande valia, para os administradores de sistemas, que as informações referentes aos ataques baseados em dicionário para *SSH* possam ser visualizadas de maneira rápida e com o menor esforço possível. Desse modo, o monitoramento torna-se mais ágil e mediante a sucessivas tentativas de entrada não-autorizada no sistema, pode-se estabelecer políticas para que o *firewall* barre futuras tentativas de ataque, provenientes de indivíduos específicos, antes delatados por seus endereços de *IP*, registrados no *log* referido.

Para que essa tarefa possa ser cumprida, foi proposta a criação de uma ferramenta baseada em tecnologias *Web*, capaz de identificar as fontes de ataques, em forma de relatórios. Assim os administradores poderão consultar periodicamente o estado dos sistemas onde a mesma encontra-se implantada, afim de que parte das ameaças aos seus servidores, possa ser mitigada.

A escolha por uma tecnologia *Web*, se deu por motivos óbvios relacionadas à facilidade de acesso às informações. Uma vez disponível em formato *Hypertexto*, as consultas podem ser realizadas a partir de uma gama enorme de meios que suportem navegação na *Web*, variando desde computadores pessoais até dispositivos *mobile*.

A ferramenta foi batizada como *Crickets' little leg*, tendo como principais características, a facilidade de utilização e a simplicidade empregada em sua interface. Também foram levantados aspectos referentes à portabilidade da mesma, sendo assim, este requisito foi levado adiante até mesmo na escolha das tecnologias empregadas para o seu desenvolvimento, abrangendo

com sucesso, a maioria dos ambientes em que encontram-se servidores baseados na família *Unix*, o que traz como consequência positiva, um processo de implantação descomplicado e extremamente viável.

# 2 Ferramenta Crickets' little leg

A concepção dessa ferramenta foi idealizada pelos fatores apresentados no capítulo anterior, assim como a motivação para que sua plataforma escolhida fosse *Web*. No entanto, foi necessária a realização de um levantamento de algumas das tecnologias atuais disponíveis neste contexto, de modo que o processo de conhecimento e experimentação das mesmas tenha sido preciso, para que seu desenvolvimento pudesse ser de fato, concluído.

Como o público alvo que se deseja atingir são administradores que possuem sistemas baseados em famílias *Unix*, pode-se levantar características em comum, encontradas em seus sistemas. Dessa maneira, se obtém o perfil das tecnologias que trabalham em conjunto na maioria dos ambientes, o que porporciona maiores chances de adesão da ferramenta. A tabela 1 mostra quais são as tecnologias mais aderidas perante à comunidade, quando implantadas ferramentas em ambientes *Web*:

Tabela 1: Tecnologias mais empregadas em aplicações Web

| Propósito | Tecnologia |
|---|---|
| Sistema Operacional | Baseado em Família *Unix* |
| Servidor *Web* | *Apache*[1] |
| Servidor de Banco de dados | *MySQL*[2] |
| Linguagem de *Scripting* | *PHP*[3] |

É importante, também, que a ferramenta seja portável, de modo que sua implantação seja facilmente realizada em ambientes que atendam os requisitos tecnológicos básicos, levantados até então. Por isso, foi constatado que um bom *framework* de desenvolvimento *Web* que obedecesse este anseio devesse ser pesquisado. Para tanto, dentre os demais disponíveis

[1]Servidor Web Apache disponível na página do projeto `http://httpd.apache.org/`
[2]Servidor de banco de dados MYSQL disponível na página do projeto `http://mysql.com/`
[3]Interpretador de scripts PHP disponível na página do projeto `http://php.net/`

no mercado, para a linguagem de *scripting PHP*, o *framework CakePHP*[9] prevaleceu, pois tem como uma das características mais marcantes, a facilidade com que as aplicações são implantadas, não tendo exigência alguma quanto à modificações na estrutura padrão do Sistema Operacional, assim os administradores não precisam realizar operações custosas em seus sistemas, para suportar exclusivamente a ferramenta.

## 2.1 Tecnologias envolvidas

Foi realizado um levantamento inicial sobre as tecnologias envolvidas no processo de desenvolvimento da ferramenta, onde serão abordadas suas principais características. Alguns contrapontos foram encontrados em sua definição, principalmente com relação à linguagem de *scripting* empregada e seu *framework* escolhido, discutidos adiante.

Na busca por uma combinação ideal, foi necessário o entendimento sobre o ambiente escolhido para implantação e as implicações de se ter tal ferramenta disponibilizada em meio ao sistema.

### 2.1.1 Servidor Web

Como servidor *Web*, foi escolhido o *Apache*, devido sua popularidade e uso amplamente difundido entre a comunidade de desenvolvedores e administradores de sistemas. É um dos maiores projetos em Software Livre do mundo, sendo percursor na criação da incubadora de projetos da *Apache Software foundation*[10]. Tem sido desenvolvido desde o ano de 1995 e conta com uma comunidade extremamente ativa de colaboradores, provenientes de diferentes partes de todo o mundo.

Perante o mercado e às outras soluções de servidores *Web*, o *Apache* ocupa uma porção de uso correspondente a aproximadamente 47%, segundo revelam as pesquisas (NETCRAFT, 2009) disponibilizadas periodicamente pela companhia de serviços relacionados à *Internet*,

[9] Disponível em `http://cakephp.org/`
[10] `http://www.apache.org/`

*Netcraft*[11]. A figura **??**

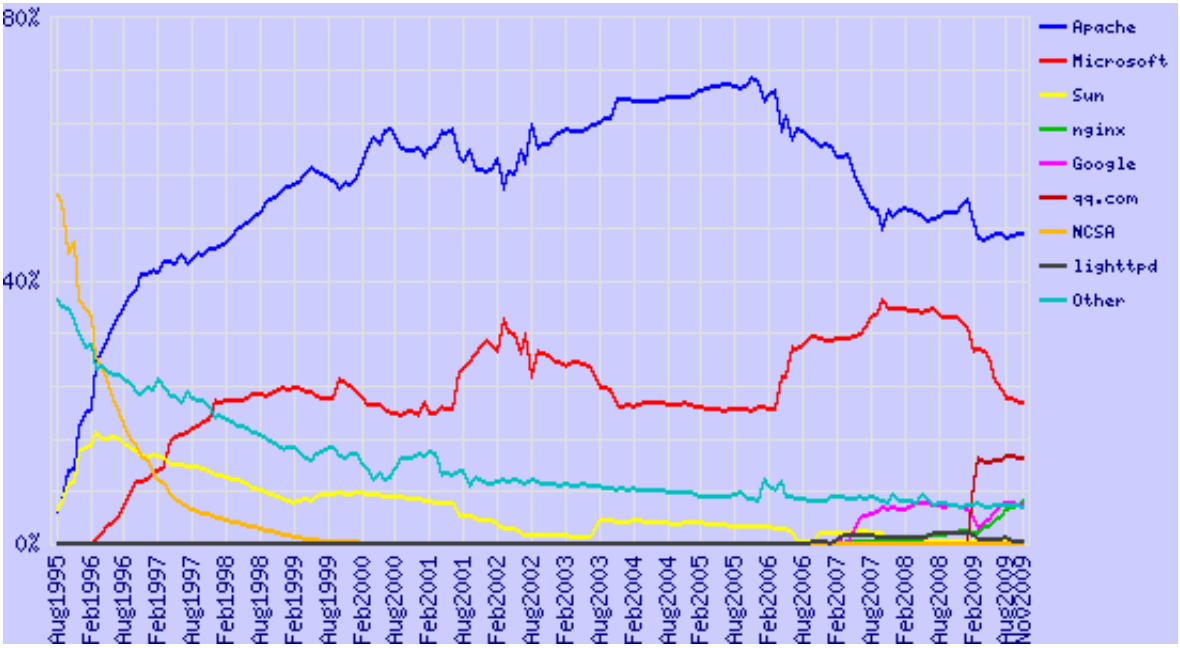


Figura 1: Utilização global de *Webservers* (NETCRAFT, 2009).

Sua integração com os Sistemas Operacionais baseados em famílias *Unix* é suave, pois foi projetado para que seja integrado perfeitamente sob sua arquitetura. Muitos sabores de distribuições baseadas no *Kernel*[12] do *GNU/Linux*[13], por exemplo, incluem o *Apache* como pacote opcional no momento de sua instalação, junto aos demais serviços de rede, sendo assim, uma alternativa bastante plausível para que seja utilizado, além de não exigir grandes esforços no momento de sua configuração para que entre em ambiente de produção.

### 2.1.2 Linguagem de Scripting

Para que a ferramenta pudesse ser desenvolvida em ambiente *Web*, foi definido que se utilizaria uma linguagem de *scripting*, devido a agilidade e quantidade de recursos que simplificam muitos processos de desenvolvimento, comuns a este gênero de linguagens de programação.

Sua concepção, assume que as linguagens de *scripting*, não sejam utilizadas na construção

[11] Netcraft: `http://netcraft.com/`

[12] O *Kernel*, trata-se do núcleo do Sistema Operacional, entre outros, provê inúmeras camadas de abstração entre o *hardware* e o *software* empregados em um computador.

[13] Disponível em `http://kernel.org/`

de aplicações desde toda sua base, mas que reutilizem uma série de componentes e bibliotecas previamente implementadas, possivelmente em diferentes linguagens de programação, ou ainda que controlem outros *softwares* através de chamadas específicas (OUSTERHOUT, 1998). Por este motivo, são amplamente empregadas nos servidores *Web*, que implementam todo o processo de receber requisições *HTTP* de seus clientes e repassá-las para filtros especiais, podendo estes ser interpretadores de código, e que ainda no lado do servidor, possam responder aos clientes com conteúdo estatico ou dinâmico à tais requisições, de acordo com sua entrada e a um determinado conjunto de estados mapeados pela aplicação requisitada.

O servidor *Web Apache*, utiliza-se de módulos para suportar a integração com outras tecnologias, sendo que existem muitos que são implementados para suportar diversas linguagens de *scripting*. O módulo mais difundido e bem suportado pelo *Apache*, para este tipo de operação, é o *mod_php*, que provê acesso às funcionalidades que a linguagem de programação *PHP* oferece. É distribuido juntamente aos pacotes binários e fontes do *Apache*, sua configuração é praticamente nativa, não necessitando mais que pouquíssimas alterações no arquivo de configuração principal[14] do Apache, para que tal suporte esteja ativo.

A figura seguinte, mostra um gráfico onde existe um comparativo entre o uso das linguagens de programação, sendo que é notória a importância da linguagem de scripting PHP, já que em sua categoria, lidera em primeiro lugar, frente aos seus maiores concorrentes: Ruby[15], Pearl[16] e Python[17].

[14] Em instalações padrão, este arquivo é encontrado em */etc/httpd/httpd.conf*
[15] Disponível em `http://ruby-lang.org`
[16] Disponível em `http://perl.com/`
[17] Disponível em `http://python.org/`

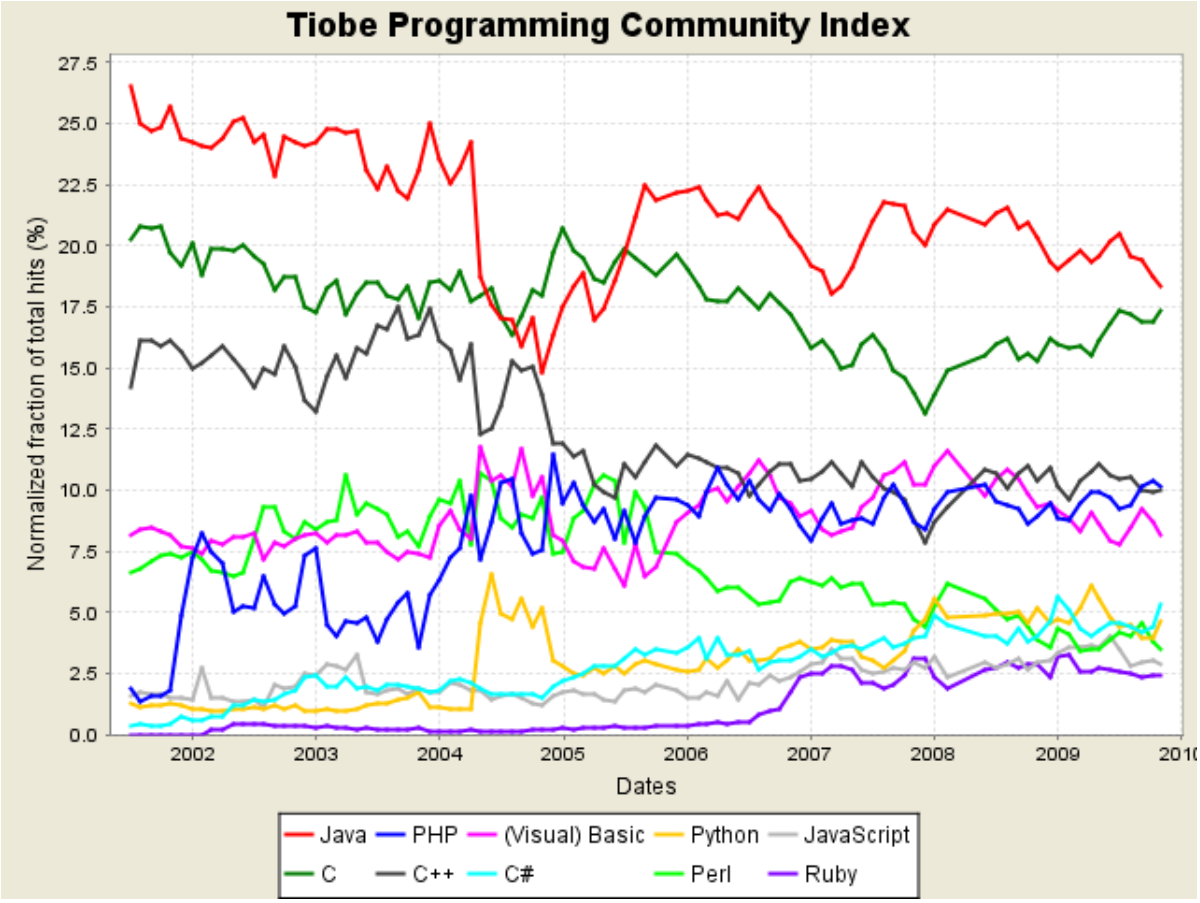


Figura 2: Utilização das linguagens de programação. (TIOBE. . . , 2009)

### 2.1.3 Banco de dados

Em aplicações *Web*, utilizando-se o servidor *Web Apache* em conjunto com a linguagem de *scripting PHP*, existe um ótimo suporte ao *SGBD MySQL*. Trata-se de um banco de dados relacional e que oferece suporte à multiplas conexões simultâneas.

Assim como as tecnologias *Apache* e *PHP*, também encontra-se como pacote de instalação opcional ao conjunto de serviços de rede, nas distribuições baseadas no *Kernel* do *GNU/Linux*. Sua aceitação é grande, em meio aos desenvolvedores *Web*. De maneira similar, suporte dado tanto pela sua comunidade de desenvedores e pela sua base de usuários é extremamente vasto, o que implica em uma adesão maior pela maioria das organizações que oferecem aplicações *Web*.

### 2.1.4 Framework de desenvolvimento Web

Para o desenvolvimento de aplicações *Web*, é amplamente utilizado o auxílio que os *frameworks* oferecem, pois muitos processos repetitivos são automatizados, fazendo com que o desenvolvimento se dê de maneira mais rápida e esteja menos suscetível a erros de implementação, oferecendo aos desenvolvedors níveis consideráveis de abstração de seus problemas, permitindo que se concentrem integralmente na lógica de negócios envolvida em seus projetos.

Levando-se em considerações o conjunto de tecnologias envolvidas, optou-se pela utilização do *framework CakePHP*. Essa decisão foi guiada após o estudo de outra tecnologia, proposta anteriormente ao início de seu desenvolvimento.

Fora proposto no plano de estágio, a utilização da linguagem de programação *Python*, em conjunto ao *framework* de desenvolvimento *Web Django*[18]. No entanto, como requisito principal de funcionamento, tal framework necessita obrigatoriamente que seja instituído um processo que envolve modificações na hierarquia do Sistema Operacional, através de sua instalação. Isso nos remete à uma das características desejadas para a ferramenta *Cricket's little leg*: que a portabilidade seja facilitada ao máximo, de acordo com o perfil mais comum entre ambientes de produção e desenvolvimento *Web*. Como nem todos os sistemas administrados requerem que a linguagem de *scripting Python* seja utilizada em conjunto ao ambiente *Web* e, ainda sim, é incomum que nestes ambientes sejam empregadas tecnologias como *Django*, optou-se pela substituição deste *framework* por algum que seja tecnologicamente equivalente e que atenda ao requisito de portabilidade, desejado desde o início da concepção da ferramenta.

O *CakePHP* é um *framework* que atende perfeitamente esta necessidade específica, sendo equivalente até mesmo em termos de arquitetura empregada na base do *framework*. Tem como vantagem principal, relativa à portabilidade, de poder ser empacotado junto à aplicação desenvolvida, e desempacotado dentro do diretório que o *Apache* utiliza para servir às requisições realizadas por seus clientes, sem que seja necessária a instalação de softwares adicionais para o seu pleno funcionamento em ambiente de produção.

[18] Disponível em `http://djangoproject.com/`

Pelas circunstâncias e motivos apresentados, justifica-se a troca de *frameworks*, pois esta foi realizada em prol da satisfação do requisito de portabilidade, o que visa estimular ainda mais a adesão da ferramenta por parte dos administradores de sistemas que possuem, de maneira praticamente nativa, um ambiente propício e compatível ao apresentado até aqui.

## 2.2 Estrutura

O *Crickets' little leg* foi projetado para se comportar de acordo com os padrões estabelecidos pelo *Design Pattern MVC*, em conjunção às convenções empregadas no *framework* de desenvolvimento *Web CakePHP*.

O *Design Pattern MVC* consiste de três camadas principais:

- **M**odel: responsável pelos dados da aplicação, são compreendidos pelo modelo, bancos de dados, arquivos em disco, dados em memória, enfim, toda e qualquer forma de fonte que contenha os dados consultados e gravados pela aplicação.
- **V**iew: a camada de apresentação, trata-se de como os dados serão dispostos. Em uma aplicação *Web*, consiste de uma página contento elementos *HTML*, *CSS*, possivelmente *Javascript* e *XML*, em conjunto aos dados processados no lado do servidor por uma linguagem de scripting, que no caso é a *PHP*.
- **C**ontroller: resposável pelo fluxo de execução da aplicação, desde as requisições realizadas pelos navegadores dos clientes, até o processamento interno das mesmas, passando pela camada de modelo e pela disposição final dos dados possivelmente processados, na camada de visão.

A figura a seguir, apresenta de maneira esquemática, como se dá a ligação entre as camadas *MVC*:

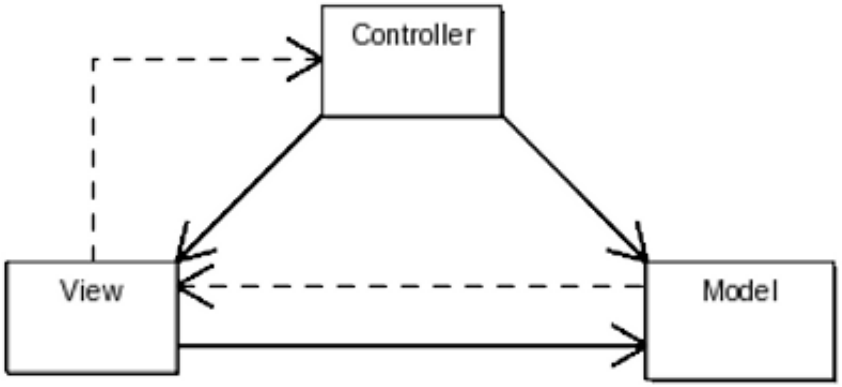


Figura 3: Diagrama representativo do *Design Pattern MVC*.

Com esta abordagem, a construção de aplicações *Web* torna-se extremamente simplificada, fornecendo diretivas suficientes para que sua construção seja rápida e que tragam ao desenvolvedor um processo bem definido em sua codificação, possibilitando uma concentração mais aplicada à resolução da lógica proposta pelo problema que se deseja resolver.

O *CakePHP* utiliza o *Design Pattern MVC* do mesmo modo que é ilustrado na figura seguinte:

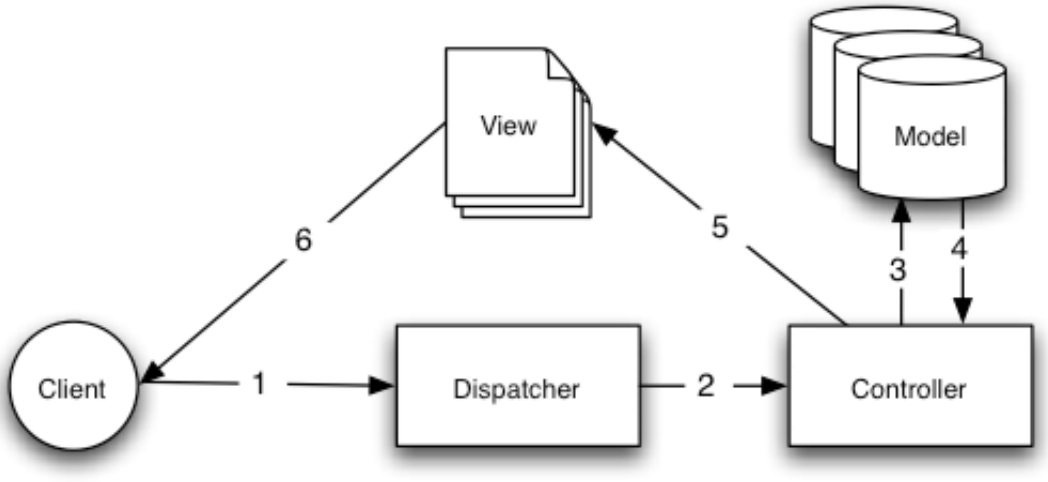


Figura 4: *CakePHP* e o *Design Pattern MVC*.

Através dos ítens numerados, pode-se traçar um perfil padrão para todas as requisições realizadas para as aplicações que fazem uso do *CakePHP*:

1. O cliente realiza a requisição.

2. O *dispatcher*[19] verifica a *URL* e seus dados, acionando uma chamada em um método de um objeto específico, instanciado como controlador.

3. O controlador escolhido pelo *dispatcher*, passa o fluxo de execução, para o modelo designado pela chamada recebida, com os dados necessários para sua execução, caso sejam necessários.

4. O modelo responde ao controlador com os dados requisitados, processados após serem extraídos da fonte de dados padrão da aplicação.

5. O controlador encaminha os dados recebidos à visão, aplicando filtros para ajustes finais, caso sejam necessários.

6. A visão renderiza os dados, junto aos elementos comuns, encontrados em páginas *Web*, como tabelas e campos de formulários, sendo assim visualizados no navegador do cliente, que antes fizera a requisição inicial.

## 2.3 Características

As características principais da ferramenta são compreendidas pelos ítens denominados a seguir:

- Utilização do mecanismo básico de autenticação (FRANKS *et al.*, 1999), implementado nativamente pelo *Apache* e suportado pela grande maioria dos navegadores *Web* existentes no mercado, no intuito de facilitar a implementação da parte de *logon* no sistema, visando acima de tudo, a simplicidade por meio da reutilização de componentes.

- Visualização dos ataques de dicionário ao serviço *SSH* sofridos no servidor local, a partir da interpretação dos dados gravados em *log*.

[19] O mecanismo chamado *dispatcher*, permite que o processo de passagem de uma mensagem para uma sequência específica de código, ou método, seja realizado em tempo de execução.

- Visualização dos ataques de dicionário ao serviço *SSH* sofridos em qualquer outro servidor, a partir do upload de seus arquivos de *log*.

- Tela provida com paginação básica, contendo os dados relativos aos ataques sofridos, após o processamento dos arquivos de *log*.

- Possibilidade de geração de regras de *Iptables*[20], capazes de barrar ataques deste gênero, com base nos endereços de *IP* pertencentes aos atacantes identificados pela interpretação dos registros contidos no arquivo de *log*.

Uma questão abordada durante a fase de idealização do projeto, era a capacidade de gerar alertas aos administradores do sistema. Como a ferramenta teve seu desenvolvimento voltado exclusivamente para a visualização das informações em forma de relatório, essa característica foi eliminada, por fugir ao escopo de sua atuação. Tal tarefa seria exclusiva à Sistemas de Detecção de Intrusão e por este motivo, foi considerada semanticamente incompatível com o propósito do projeto, justificando assim, o motivo pelo qual tal característica fora eliminada do processo de desenvolvimento.

[20] *IPtables* é uma ferramenta que permite aos administradores de sistemas baseados em famílias *Unix*, que configurem as tabelas de *firewall* encontradas no *Kernel* do Sistema Operacional.

# 3 Engenharia empregada no Software

É desejável, que para todo software, seja empregada uma metodologia em seu desenvolvimento. Esse passo é essencial para que se obedeça tenha um fluxo constante de melhorias em suas novas características e em correções de eventuais problemas, relatados pela sua utilização.

Para o desenvolvimento do *Crickets' little leg*, foi adotada a metodologia *AUP* de Desenvolvimento. Sendo altamente recomendada para pequenos projetos e que envolvam uma pequena quantidade de desenvolvedores.

A *AUP*, ou *Agile Unified Process é uma versão simplificada da IBM Rational Unified Process* (*RUP*), desenvolvida por *Scott Ambler*[21]. Descreve uma abordagem simples de se entender o desenvolvimento de aplicações, utilizando técnicas ágeis e conceitos remanescentes da natureza encontrada em seu predecessor, o *RUP*. São encontradas nessa metodologia, técnicas como o desenvolvimento orientado a testes[22], modelagem ágil, gerência ágil e refatoração de base de dados no intuito de se melhorar a produtividade.

## 3.1 Ciclo de vida

A disciplina *Modelo*, engloba o modelo de negócios encontrado na modelagem *RUP*, assim como as disciplinas de *Requisitos*, *Análise* e *Projeto*. O *Modelo* é uma parte importante da *AUP*, mas não domina o processo como um todo, pois parte da premissa de que devem ser gerados somente os modelos e documentos que sejam suficientemente necessários. As disciplinas de *Configuração* e *Mudança na Gerência* são suprimidas em favor da disciplina Gerência de Configuração. No processo de desenvolvimento ágil, as atividades de *Mudanças*

[21] Página oficial da AUP disponível em `http://www.ambysoft.com/unifiedprocess/agileUP.html`
[22] *Test Driven Development*, ou *TDD*.

*na Gerência* são partes típicas dos requisitos de *Gerenciamento de Esforço*, sendo parte da disciplina *Modelo*. Esta linha de pensamento é ilustrada na figura seguinte:

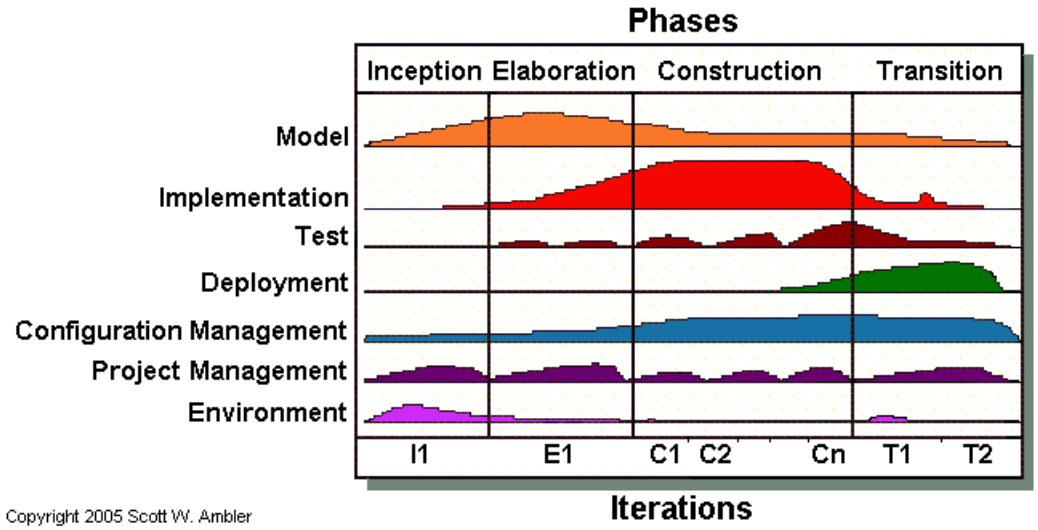


Figura 5: Ciclo de vida da *AUP*.

A natureza serial da AUP é capturada em quatro fases:

1. Início: Identificar o escopo inicial do projeto, arquitetura potencial do sistema e obtenção de fundos iniciais advindos dos interessados em seu desenvolvimento.

2. Elaboração: Discutir a arquitetura do sistema, buscando a prova de que seus conceitos são bons.

3. Construção: Construir um software funcional, com bases incrementais e regulares, capazes de atender às mais altas prioridades estabelecidas pelos interessados em seu desenvolvimento.

4. Transição: Validar e implantar o sistema em um ambiente de produção.

Sobre as disciplinas envolvidas, temos sua descrição como:

- Modelo: entender as regras de negócio, o domínio que especifica os problemas que devem ser atacados pelo projeto e identificar soluções que os resolvam.

- Implementação: transformar o modelo, em código, além de realizar testes básicos, em particular, testes unitários.

- Test: realizar avaliações objetivas que garantam a qualidade do software.

- Entrega: planejar a entrega do sistema, de modo que seja executado um plano capaz de fazer com que seus usuários finais estejam confortáveis com sua adoção/utilização.

- Configuração do Projeto: gerenciar o acesso aos artefatos do projeto, incluindo o rastreamento de suas versões ao longo do tempo e gerenciar mudanças entre si.

- Gerenciamento do Projeto: direcionar atividades relacionadas à gerência do projeto, tais como a gerêcia de riscos, delegação de tarefas e manipulação de recursos fora do escopo do desenvolvimento do projeto, para que seja garantida sua entrega.

- Ambiente: garantir que os recursos necessários para que seu desenvolvimento e implantação estejam disponíveis ao time de desenvolvimento.

## 3.2 Diagrama de Casos de Uso

Os seguintes Diagramas de Casos de Uso foram elaborados, como parte auxiliar do projeto da ferramenta:

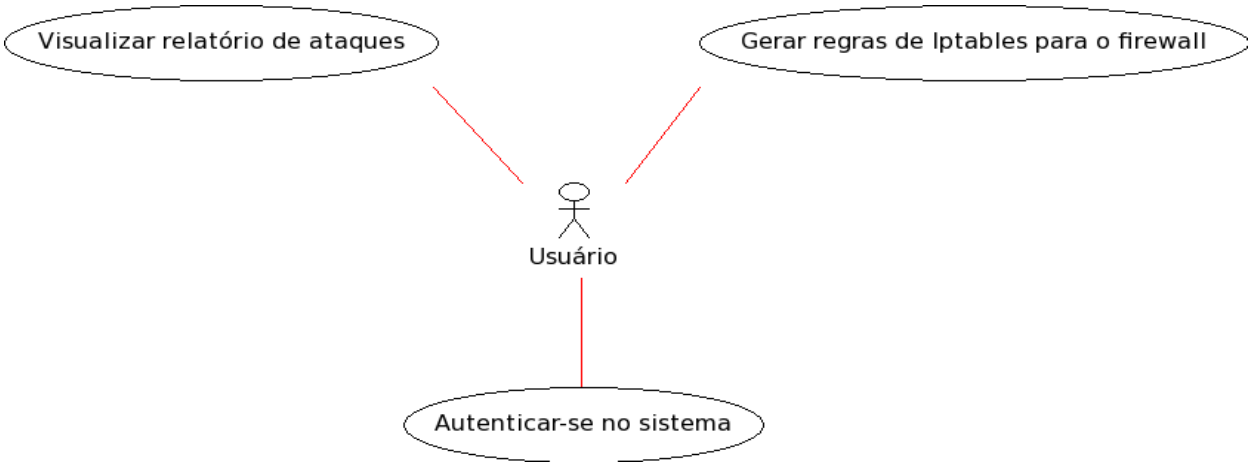


Figura 6: Diagrama de Caso de Uso relativo ao usuário.

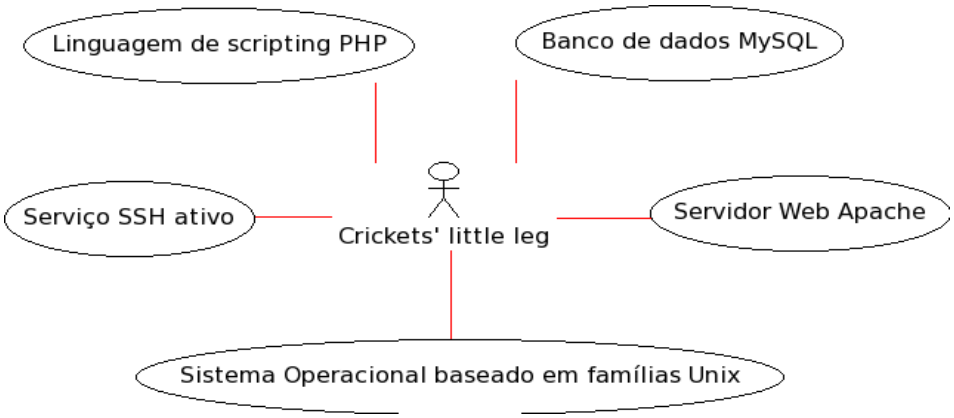


Figura 7: Diagrama de Caso de Uso relativo ao *Crickets' little leg*.

## 3.3 Diagrama de classes

Os seguintes Diagramas de Classes foram elaborados, como parte auxiliar do projeto da ferramenta:

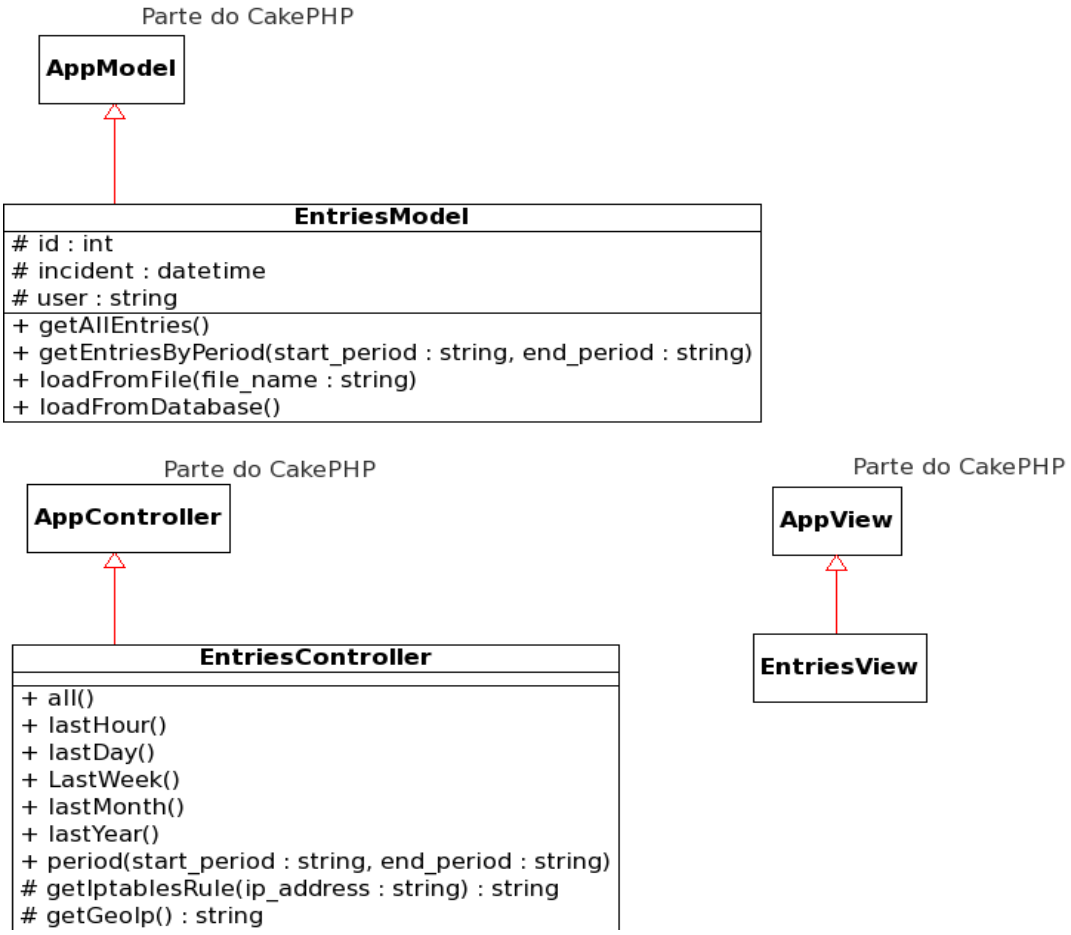


Figura 8: Diagrama de Classes geral do *Crickets' little leg*.

## 3.4 Modelo Entidade Relacionamento

O seguinte Modelo de Entidade Relacionamento, como parte auxiliar do projeto da ferramenta:

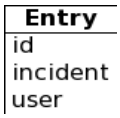


Figura 9: Modelo Entidade Relacionamento geral do *Crickets' little leg*.

# Conclusão

Foi apresentada a ferramenta chamada *Crickets' little leg*, concebida no intuito de auxiliar os administradores de sistemas baseados em famílias *Unix*, a verificarem quais são os endereços de *IP* provenientes de seus atacantes, assim como a data dos incidentes, existindo a possibilidade de terem como resultado, regras de *Iptables*, capazes de se integrar aos *Kernels* de seus Sistemas Operacionais e com isso, bloquearem o acesso não-autorizado aos indivíduos que fazem uso mal intencionado do serviço provido.

Seu desenvolvimento foi guiado no intuito de gerar uma ferramenta que fosse extremamente simples e que atendesse a demanda da maioria dos ambientes onde se encontram hospedadas aplicações *Web*, não exigindo assim, requisitos adicionais capazes de inviabilizar sua implantação.

Como trabalhos futuros, a ferramenta poderá agregar mais análises, relativas a diferentes formas de ataques, bastando que sua configuração seja modificada e mais módulos sejam programados, integrando-se facilmente ao modelo *MVC* empregado no seu projeto.

# Referências Bibliográficas

FRANKS, J. *et al. HTTP Authentication: Basic and Digest Access Authentication*. [S.l.], Junho 1999. Disponível em: <http://tools.ietf.org/html/rfc2617>.

INSTITUTE, S. T. *The Top 10 Most Critical Internet Security Threats*. [S.l.], Maio 2002. Disponível em: <http://www.sans.org/top20/2000/>.

INSTITUTE, S. T. *Top 20 Internet Security Problems, Threats and Risks*. [S.l.], Novembro 2007. Disponível em: <http://www.sans.org/top20/>.

NETCRAFT. *November 2009 Web Server Survey*. [S.l.], Novembro 2009. Disponível em: <http://news.netcraft.com/archives/2009/11/10/november_2009_web_server_survey.html>.

OUSTERHOUT, J. K. Scripting: Higher level programming for the 21st century. *IEEE Computer magazine*, Março 1998. Disponível em: <http://home.pacbell.net/ouster-/scripting.html>.

PROVOS, N. *et al.* Preventing privilege escalation. In: *SSYM'03: Proceedings of the 12th conference on USENIX Security Symposium*. Berkeley, CA, USA: USENIX Association, 2003. p. 16–16.

TIOBE Programming Community Index for November 2009. [S.l.], Novembro 2009. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>.

VYKOPAL, J. *et al.* Network-based dictionary attack detection. In: *ICFN '09: Proceedings of the 2009 International Conference on Future Networks*. Washington, DC, USA: IEEE Computer Society, 2009. p. 23–27. ISBN 978-0-7695-3567-8.