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, 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 memória de 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.

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

Aos meus amigos, por me acompanharem sempre almejando o melhor, trazendo muitos momentos de alegria, tendo sido 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 que essas ameaças sejam filtradas.

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, was developed a *Web* tool that is able to identify the attack origins, generate reports and rules that can be used in Firewalls to these treats be filtered.

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

Sumário

Lista de Figuras

Lista de Tabelas

Lista de abreviaturas e siglas

API	Application Programming Interface
AUP	Agile Unified Process
CeWL	Custom Word List generator
CLI	Command Line Interface
CSS	Cascading Style Sheet
DoS	Denial of Service
FBI	Federal Bureau of Investigation
GNU	GNU Not Unix
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IP	Internet Protocol
LGPL	Lesser General Public License
MVC	Model View Controller
NIPC	National Infrastructure Protection Center
PHP	PHP: Hypertext Preprocessor
RUP	Rational Unified Process
SANS	SysAdmin, Audit, Network, Security
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, são 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 de uma ferramenta *Web* que busca por tentativas não autorizadas de acesso ao serviço SSH em servidores, a partir da análise de seus logs gerados, auxiliando a resolução deste problemas através da geração automática de um conjunto de regras para *Iptables*, que podem ser utilizadas para bloquear futuras tentativas de acesso por parte dos atacantes. Os capítulos são divididos da seguinte maneira:

- *Capítulo 1*: Descreve os ataques de dicionário, empregados contra os servidores da família *Unix* e que rodam o serviço *SSH*. Exemplifica este problema a partir do lançamento de um ataque bem sucedido contra o *host* local, preparado propositalmente para o teste.

Introduz a idéia de se ter uma ferramenta capaz de analisar os logs gerados a partir dessa modalidade de ataques.

- *Capítulo 2*: Apresenta a ferramenta que fora idealizada no Capítulo 1. Mostra um levantamento sobre a popularidade das tecnologias utilizadas em servidores *Web*, justificando a influencia que exerceram sobre o processo de criação da ferramenta. Exemplifica sua utilização e aborda suas características.

- *Capítulo 3*: Mostra a engenharia empregada no desenvolvimento do software e aborda aspectos sobre a metodologia de desenvolvimento utilizada.

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 $O(n^2)$, uma vez que trata-se do produto cartesiano dos elementos contidos na *Wordlist* consigo mesmo.

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 determinadas instituições. É também muito comum, que computadores comprometidos sejam integrados às redes zumbis e sejam controlados remotamente, de modo que lancem diversos tipos de *scans* em rede na busca de máquinas também comprometidas e/ou que rodem serviços 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*, padrão em muitos dos sistemas baseados na família *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 Análise de logs SSH

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 provenientes de indivíduos específicos, antes delatados por seus endereços de *IP*, registrados no *log* referido.

Foi escolhido como tema deste estágio, a criação de uma ferramenta capaz de fornecer aos administradores de sistemas, uma maneira simples e de fácil acesso aos ataques com base nos dados gerados pelo serviço *SSH*, armazenados em seus servidores, formatados como arquivos de log, optou-se também, pela possibilidade de se realizar upload de logs para o sistema *Web*, uma vez que os arquivos podem estar armazenados de maneira dispersa. Visando facilitar o acesso à ferramenta e suas funcionalidades, foi escolhido um conjunto de tecnologias voltadas para a *Web* em sua implementação e implantação. Uma vez disponível em formato *Hypertexto*, as consultas podem ser realizadas a partir de uma gama enorme de meios que suportem navegação, 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

Foi necessária a realização de um levantamento de algumas das tecnologias atuais, no contexto de aplicações *Web*, de modo que fora realizado um processo de conhecimento e experimentação das mesmas, para que as atividades relacionadas ao desenvolvimento pudessem ser realizadas.

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 no mercado, para a linguagem de *scripting PHP*, o *framework CakePHP* [9] prevaleceu, pois

[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/`
[9] Projeto CakePHP, disponível no site `http://cakephp.org/`

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*, *Netcraft* [11]. A figura 1 mostra um gráfico percentual de utilização global das tecnologias em Webservers, confirmando o servidor *Web Apache*, como sendo o mais amplamente adotado

[10] Incubadora de projetos Apache, disponível em `http://www.apache.org/`

[11] Instituto de pesquisas em *Internet Netcraft*, disponível em `http://netcraft.com/`

em sua categoria:

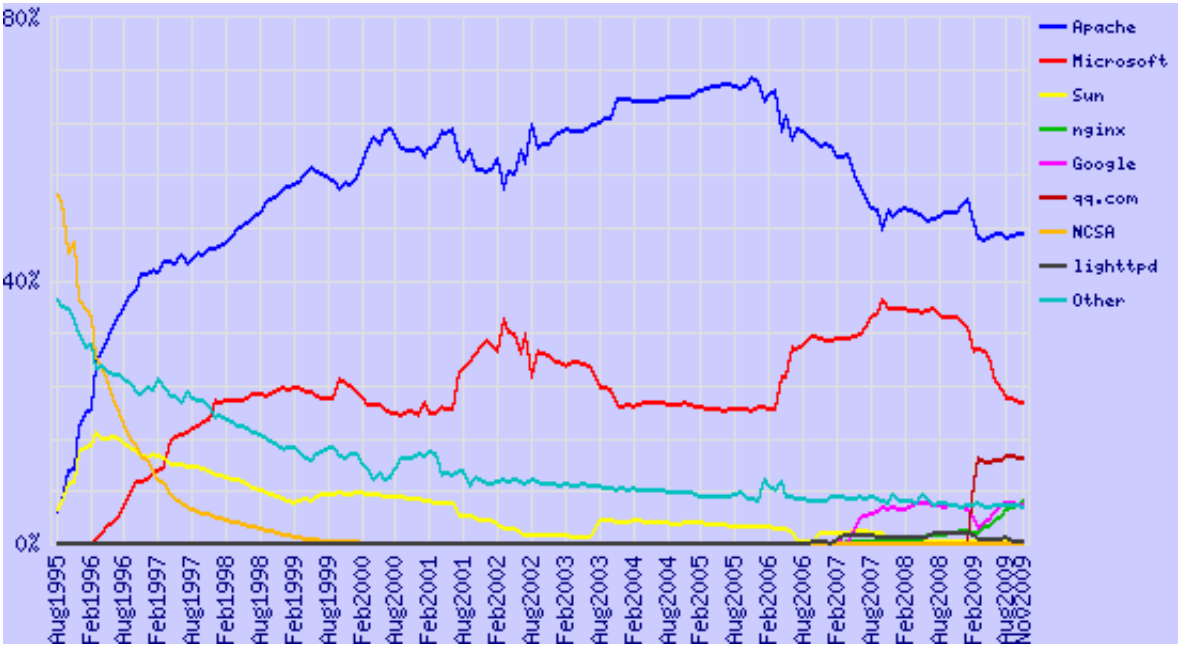

Figura 1: Utilização global de *Webservers* (NETCRAFT, 2009).

Sua integração com os Sistemas Operacionais baseados em famílias *Unix* é suave, pois foi projetado para que se integre perfeitamente sob sua arquitetura. No momento de sua instalação, muitos sabores de distribuições baseadas no *Kernel* [12] do *GNU/Linux* [13], por exemplo, incluem o *Apache* como pacote opcional junto aos demais serviços de rede, sendo assim, uma alternativa bastante plausível de utilização, devido seu suporte praticamente nativo, além de não exigir grandes esforços em 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

[12] O *Kernel*, trata-se do núcleo do Sistema Operacional. Entre outras funcionalidades, provê inúmeras camadas de abstração entre o *hardware* e o *software* empregados nos computadores.

[13] O *Kernel* do *GNU/Linux* é disponibilizado no site do projeto, 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 aliada ao 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 2, mostra um gráfico 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 no uso, 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] Linguagem de programação *Ruby*, disponível em `http://ruby-lang.org`
[16] Linguagem de programação *Pearl*, disponível em `http://perl.com/`
[17] Linguagem de programação *Python*, disponível em `http://python.org/`

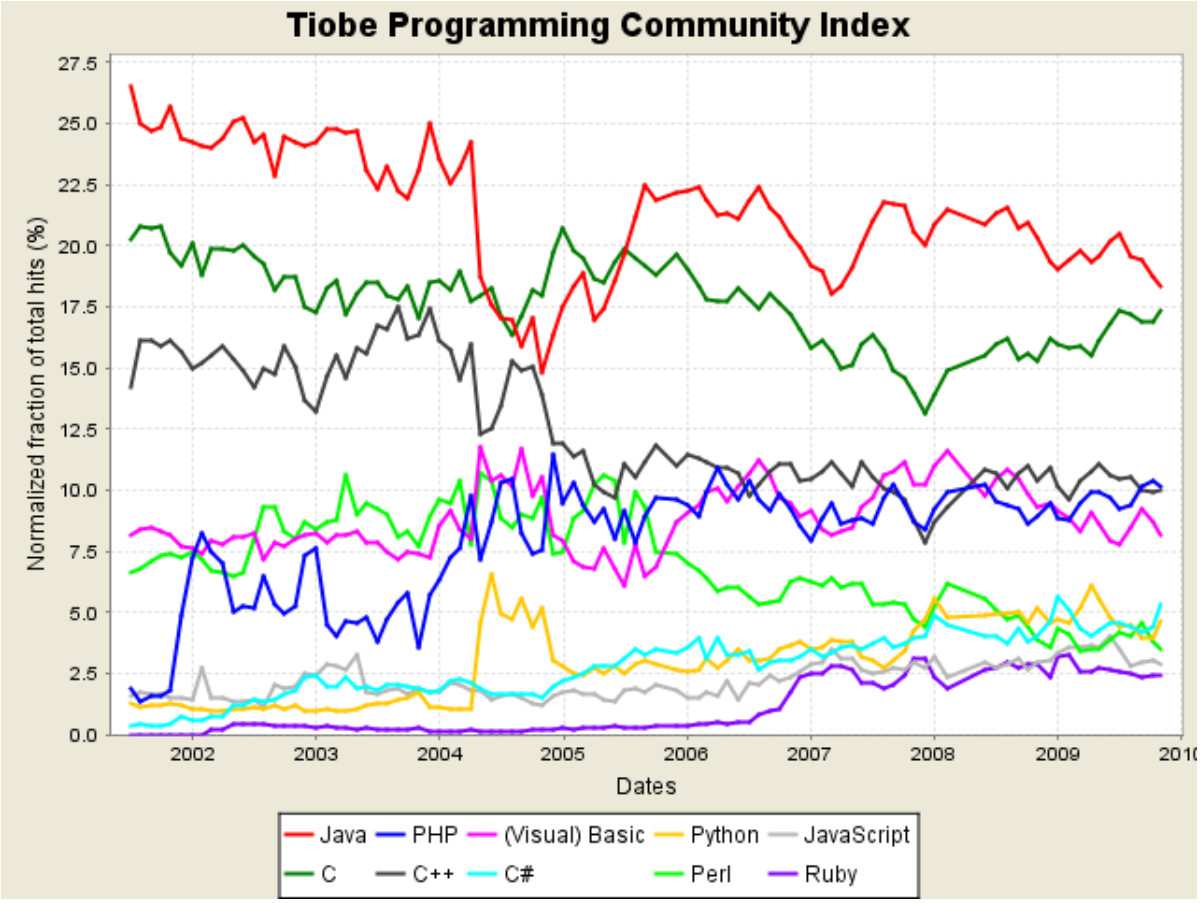

Figura 2: Utilização das linguagens de programação. (TIOBE. . . , 2009)

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* (ALBING *et al.*, 2004). Trata-se de um banco de dados relacional e que oferece suporte à multiplas conexões simultâneas. (AB, 2004)

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] O framework de desenvolvimento *Django*, proporciona o desenvolvimento rápido de aplicações *Web*. Baseado na linguagem de programação *Python*, segue o *design pattern MVC* na organização dos sistemas desenvolvidos. Suporta uma grande diversidade de bancos de dados. 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* (REENSKAUG, 2003), 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 3, apresenta de maneira esquemática, como se dá a ligação entre as camadas *MVC*:

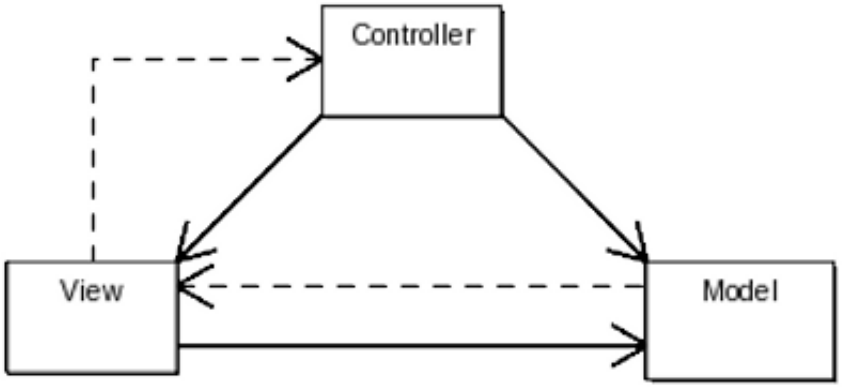

Figura 3: Diagrama representativo do *Design Pattern MVC*.

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 fluxo de utilização do *Design Pattern MVC*, pelo framework de desenvolvimento *CakePHP* é exemplificado pela figura 4

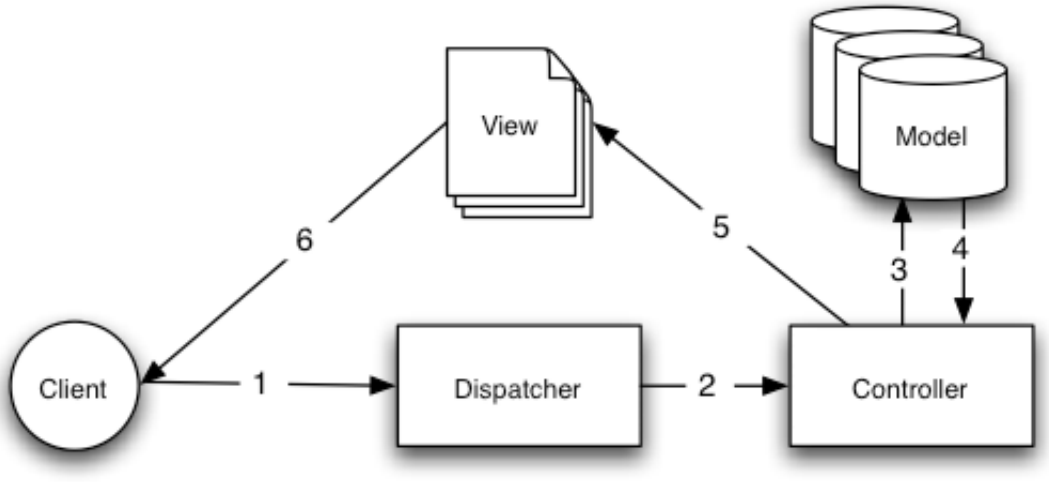

Figura 4: *CakePHP* e o *Design Pattern MVC*.

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.

[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.

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.

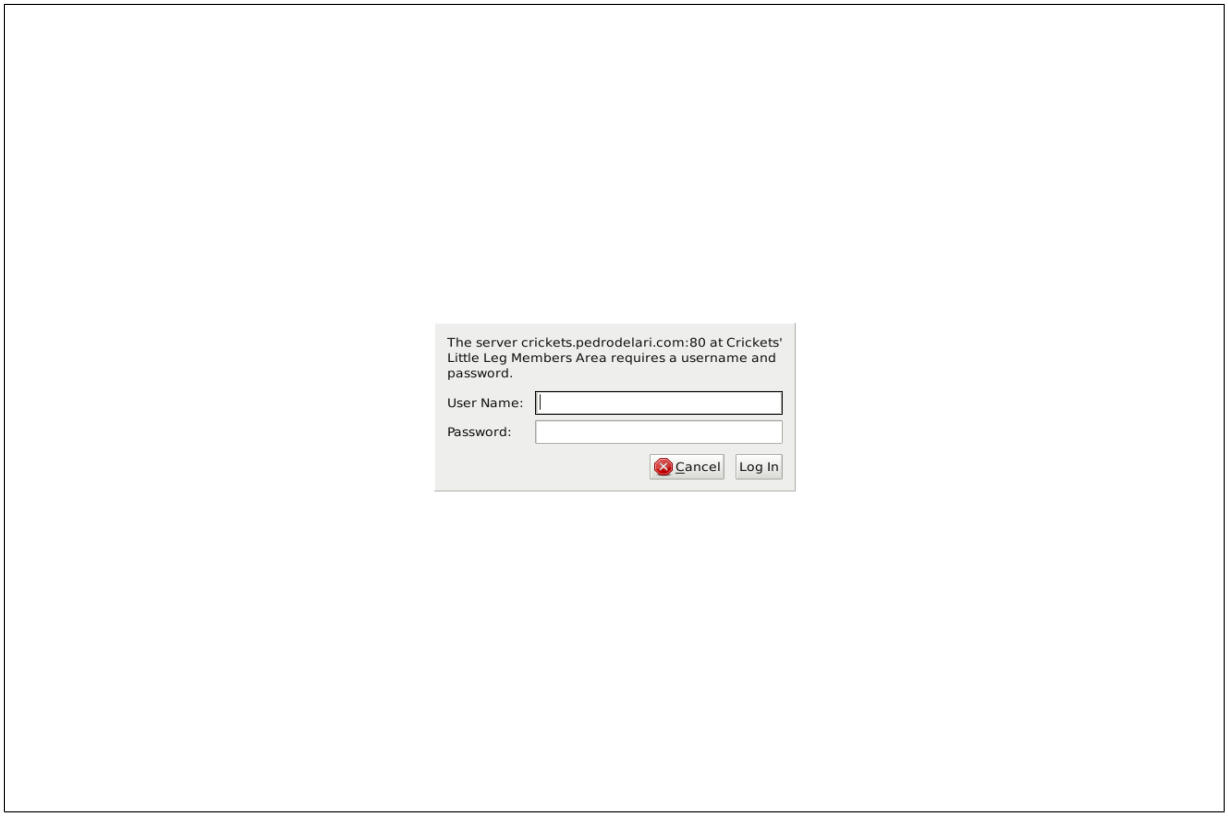

Figura 5: Autenticação básica *HTTP*.

- Visualização dos ataques de dicionário ao serviço *SSH* sofridos no servidor local, a partir da interpretação dos dados gravados em *log*, bastando informar o path onde estão localizados.

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

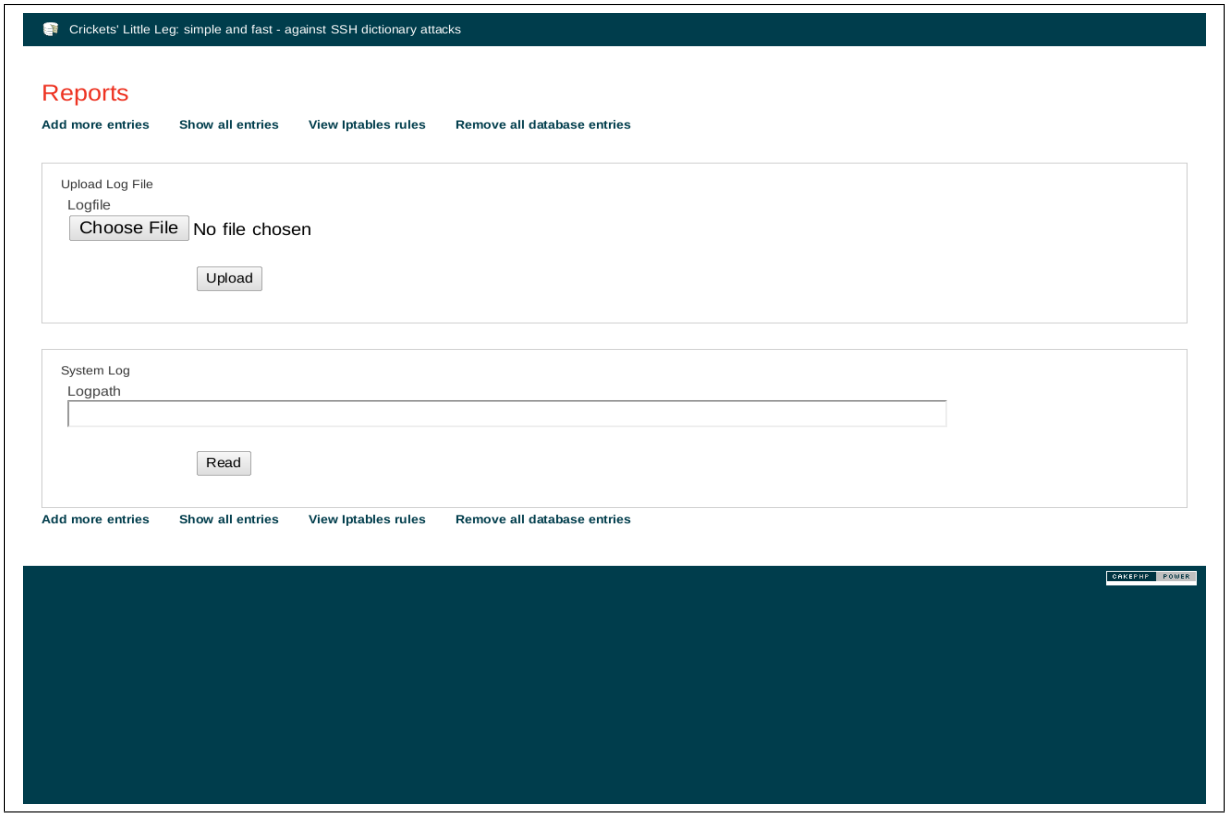

Figura 6: Seleção da fonte de dados.

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

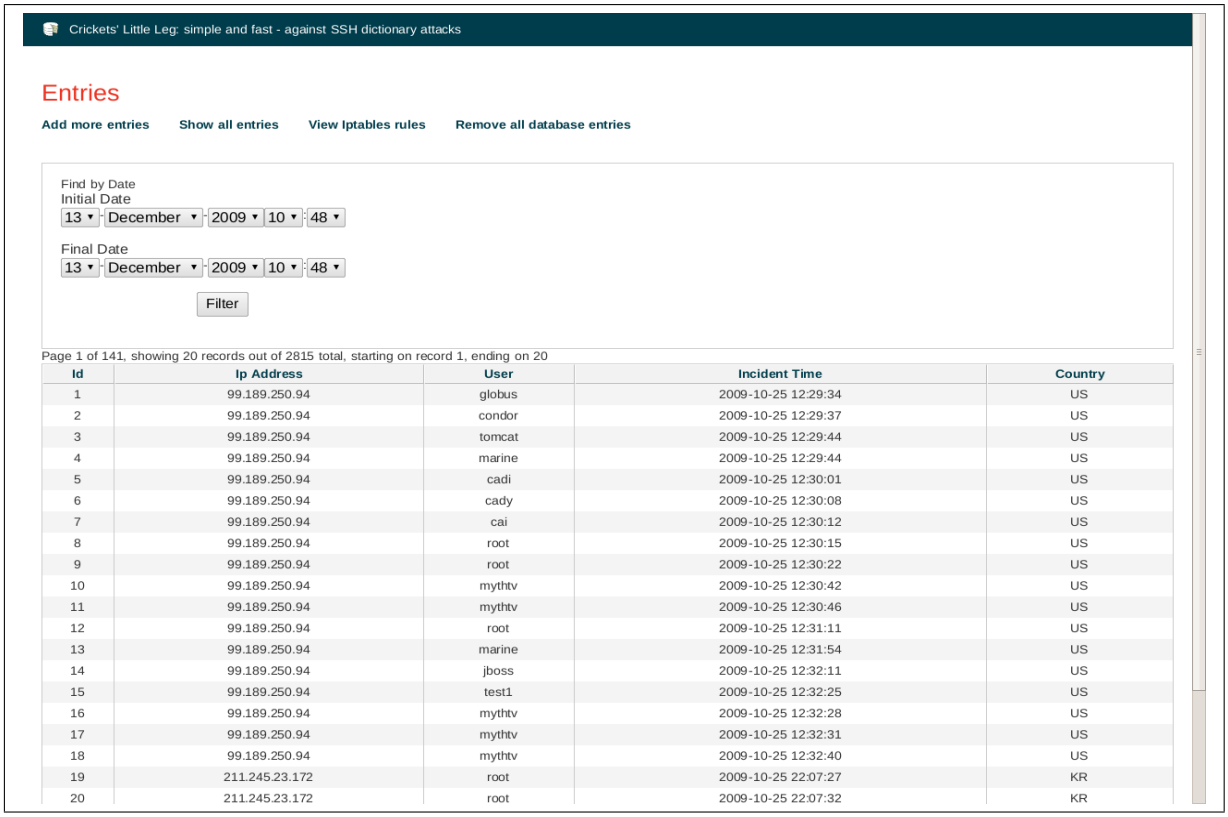

Figura 7: Dados extraídos dos logs.

- 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*.

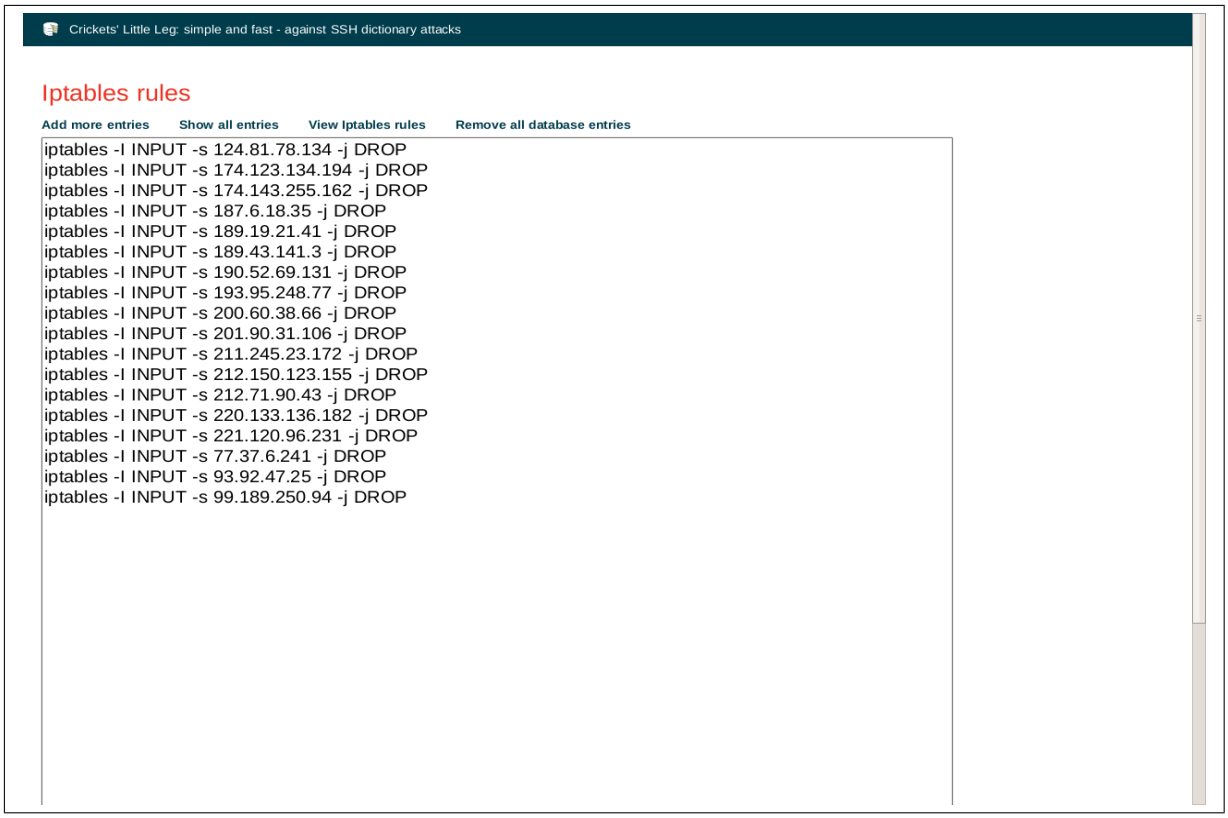

Figura 8: Regras de Iptables geradas a partir dos dados extraídos pela ferramenta.

Um exemplo reduzido do log extraído de um caso real de ataques por dicionário, realizado para a elaboração da ferramenta, é mostrado no anexo A

Para a elaboração da ferramenta, utilizou-se um computador que fora exposto a ataques de dicionário ao serviço *SSH*. Os dados coletados a partir dos ataques foram armazenados em logs, tendo sido utilizados como base para a identificação de padrões nos arquivos. A identificação dos dados foi realizada por meio de expressões regulares *taggeadas*. Cada campo relevante nas linhas escolhidas do arquivo de logs, recebeu uma marca correspondente, sendo extraído e armazenado posteriormente, no banco de dados.

[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.

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.

2.4 Localização do atacante

A ferramenta é capaz de localizar os países de onde partiram os ataques ao servidor analisado. Isso é possível graças à integração realizada entre a ferramenta e um *Webservice* da família de ferramentas *GeoIP*, específicas para esta tarefa.

O acesso a tal *Webservice*, se dá através do uso de um componente para o *CakePHP* chamado *GeoIPComponent* [21], criado por *Wayne Khan* e cuja licensa é a *GNU Lesser General Public License*, mais conhecida como simplesmente *LGPL*.

O *Webservice* é acessado por meio de uma *API* publica, que fora integrada ao *PHP*, sendo utilizada pelo componente *GeoIPComponent*. Tal *API* pública é, disponibilizada pela empresa *MaxMind* [22].

Para cada entrada reconhecida no arquivo de logs, a ferramenta *Cricket's Little Leg* dispara uma consulta à *API* pública da *MaxMind*, requisitando o código da região, relativo ao *IP* informado, que no horário em que o ataque fora registrado, pertence obviamente, ao seu autor. A *MaxMind*, dotada de um sistema específico para a identificação de *IPs*, retorna o código de região. O código da região ao qual o endereço de *IP* consultado pertence, é comparado com um banco de dados local de códigos de regiões [23], e então a sigla do país em questão é retornada, sendo assim possível interpretar satisfatoriamente o resultado desta

[21] O componente *GeoIPComponent* para *CakePHP* é disponibilizado em `http://github.com/kzhiwei/cakephp-geoip`.

[22] Site da empresa MaxMind: `http://www.maxmind.com/`.

[23] O banco de dados local de possíveis regiões, pode ser obtido em `http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz`

busca, que é armazenado no banco de dados da ferramenta, para fins de exibição ao usuário em consultas posteriores.

A figura 9 ilustra todo esse processo, compreendido desde a identificação das entradas e a busca pelo país de origem de cada ataque identificado:

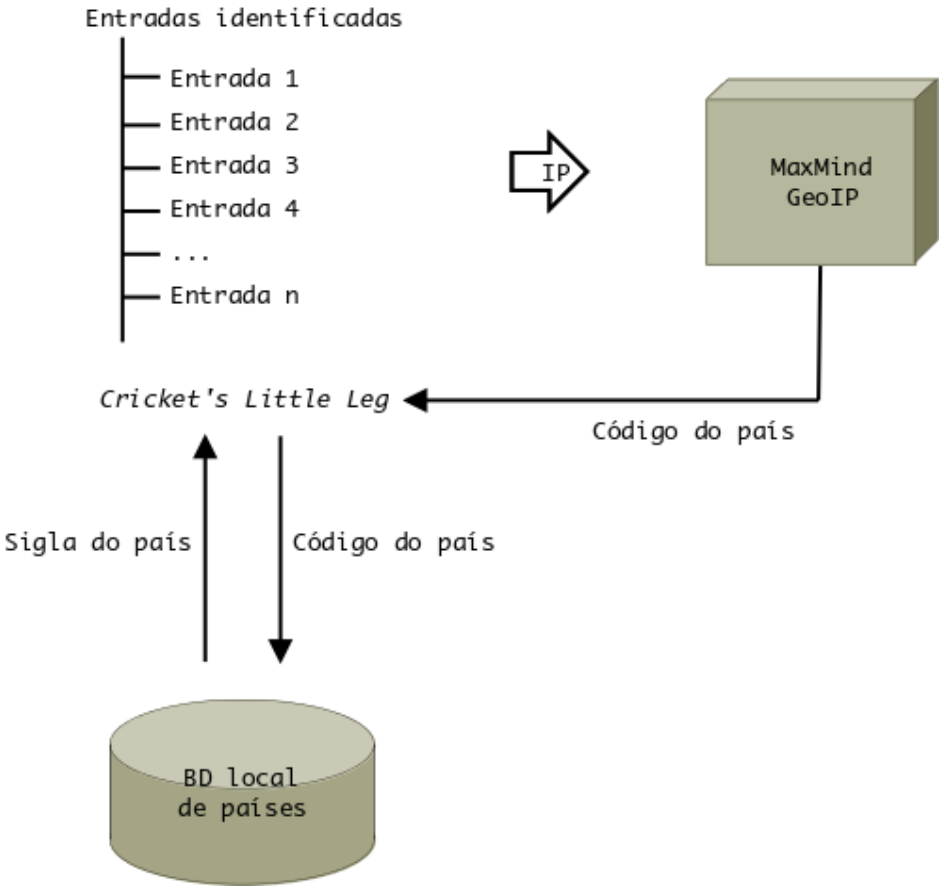

Figura 9: Identificação da procedência dos ataques.

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* [24]. 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 [25], 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*

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

na Gerência são partes típicas dos requisitos de *Gerenciamento de Esforço* (AMBER, 2009), sendo parte da disciplina *Modelo*. Esta linha de pensamento é ilustrada na figura 10:

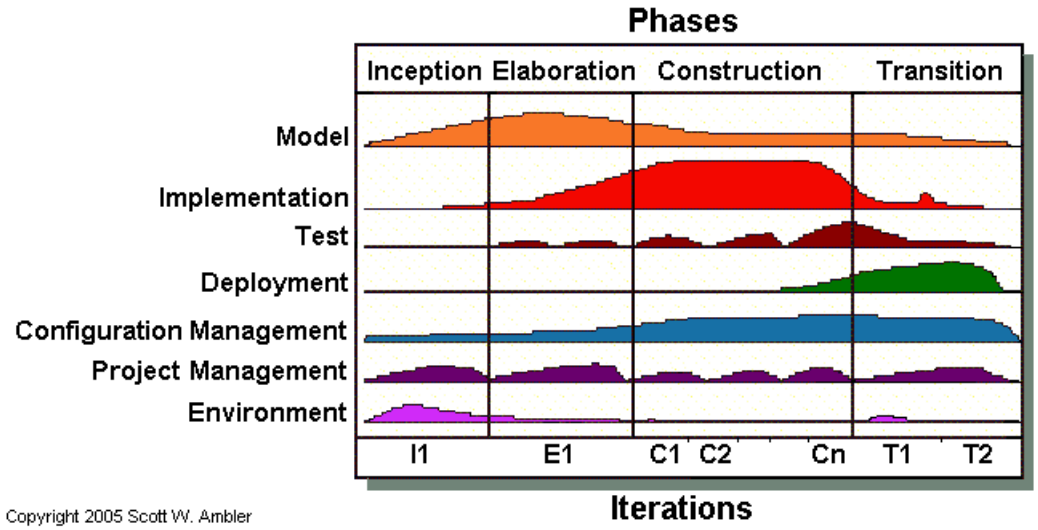

Figura 10: Ciclo de vida da *AUP*. (AMBER, 2009)

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 Diagramas de Casos de Uso

Os Diagramas de Casos de Uso foram elaborados, como parte auxiliar do projeto da ferramenta, obedecendo aos requisitos funcionais e não-funcionais, compreendidos por todas as características da ferramenta, previamente apresentadas no capítulo 2.

O Diagrama de Caso de Uso relativo ao usuário, ilustrado pela figura 11, mostra quais tipos de interação que o usuário pode ter com a ferramenta.

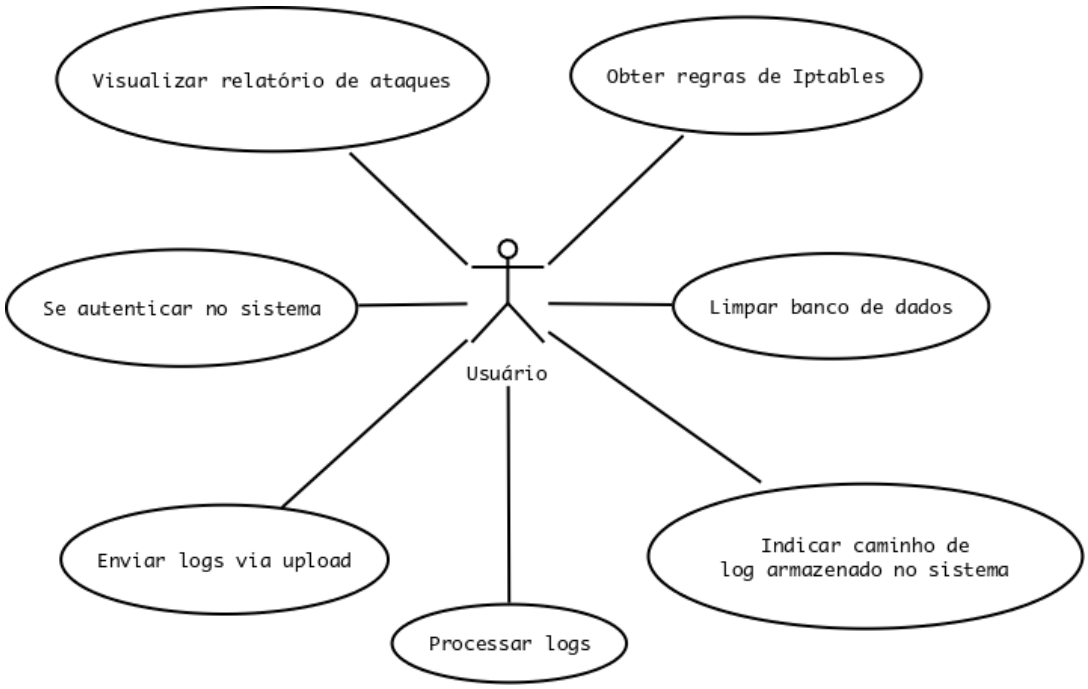

Figura 11: Diagrama de Caso de Uso relativo ao usuário.

Diagrama de Caso de Uso relativo ao *Crickets' little leg*, ilustrado pela figura 12, mostra os requisitos não-funcionais aos quais a ferramenta encontra-se atrelada.

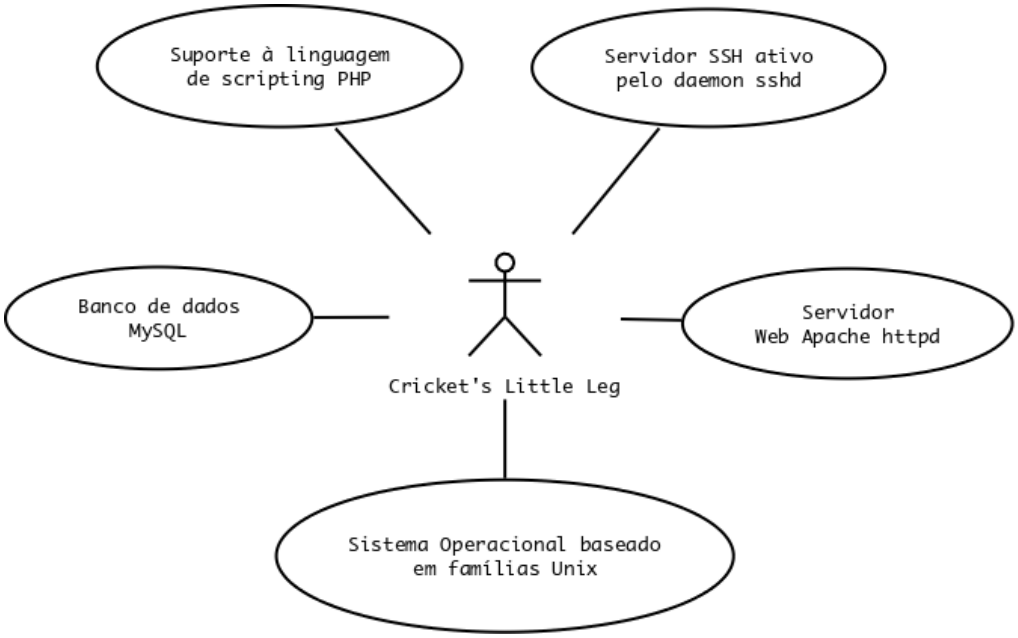

Figura 12: Diagrama de Caso de Uso relativo ao *Crickets' little leg*.

3.3 Diagrama de classes

Seguindo as convenções do design *pattern MVC*, adotado como padrão pelo *framework* de desenvolvimento *CakePHP*, que fora utilizado para o desenvolvimento da ferramenta, foram elaboradas, de acordo com suas categorias enumeradas, as seguintes classes:

1. Classe de Modelo
    - *Entry*
2. Classes de Controladores
    - *EntriesController*
    - *PagesController*

Para as classes apresentadas, foi elaborado um Diagrama de Classes, ilustrado pela figura 13, como parte auxiliar no projeto da ferramenta. É importante atentar-se ao fato de que as classes apresentadas, têm como herança, classes advindas do *framework CakePHP*. Desta maneira, tais superclasses tiveram seus atributos e métodos suprimidos no diagrama em questão.

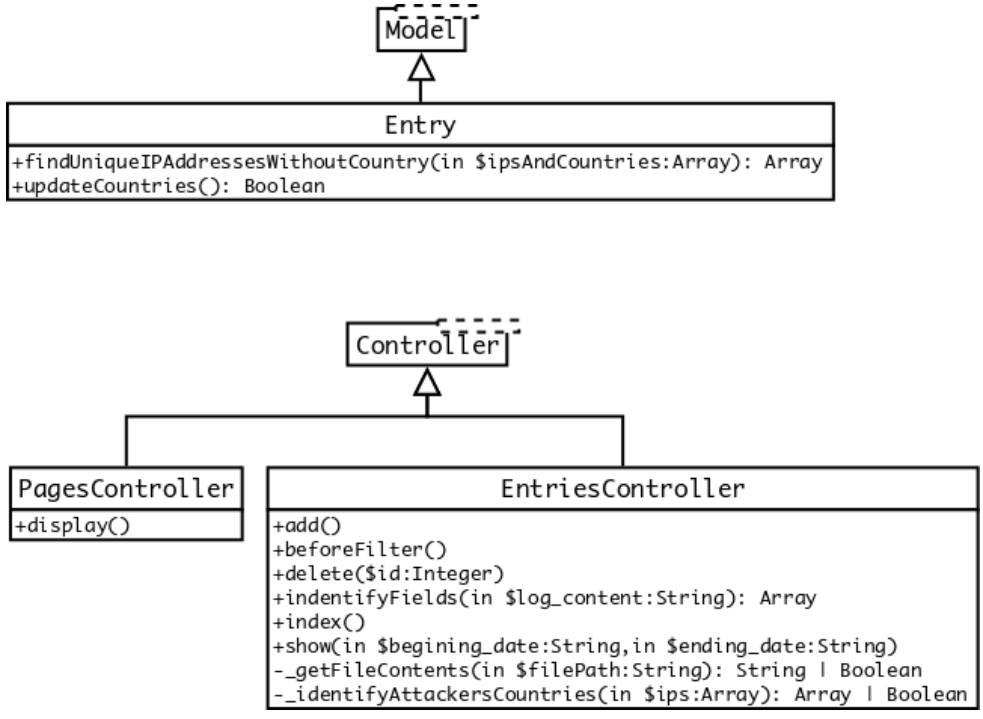

Figura 13: Diagrama de Classes geral do *Crickets' little leg*.

3.4 Modelo Entidade Relacionamento

O banco de dados da ferramenta é muito simples, tendo em vista que os únicos dados armazenados são os resultados das capturas obtidas, através da interpretação dos logs especificados pelo usuário. Todas as entradas, ficam armazenadas em uma tabela, chamada *Entry*. Os países identificados em cada entrada, são armazenados na tabela *Country*. Cada entrada, se relaciona com um país por meio da chave "country_id", presente em *Entry*, sendo correspondida em *Country* por sua chave primária, ou simplesmente "id". O Modelo Entidade Relacionamento da ferramenta é ilustrado pela figura 14:

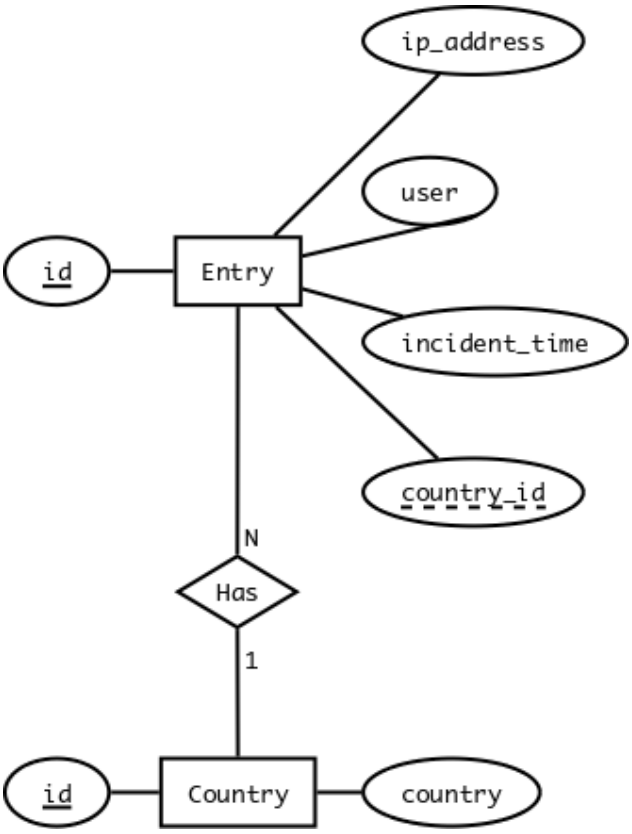

Figura 14: Modelo Entidade Relacionamento geral do *Crickets' little leg*.

Conclusão

Foi apresentada neste trabalho a ferramenta *Crickets' little leg*, concebida no intuito de auxiliar os administradores de sistemas baseados em famílias *Unix* a identificar informações sobre ataques de dicionários lançados contra o serviço *SSH*, a partir dos *logs* registrados. Através de uma interface *Web* simples e intuitiva, as seguintes informações são recuperadas:

- Os endereços de *IP* dos responsáveis pelos ataques
- Os horários em que os ataques ocorreram
- Os países de onde foram lançados os ataques

Além do processo de extração de informações, a ferramenta permite a geração automática de regras para *Iptables*, que uma vez aplicadas nos servidores, bloqueiam efetivamente os pacotes de rede provenientes dos enderços de *IP* considerados suspeitos.

A ferramenta foi desenvolvida no intuito de atender a demanda da maioria dos ambientes onde se pode hospedadar aplicações *Web*, não exigindo assim, requisitos adicionais, que por ventura possam inviabilizar sua implantação.

Como trabalhos futuros, a ferramenta poderá:

- Agregar análises relacionadas a diferentes formas de ataque, como por exemplo, a identificação de ataques *DoS* específicos, bastando que sua configuração seja modificada e seus módulos sejam programados, se integrando facilmente ao modelo *MVC* empregado na arquitetura do projeto.
- Oferecer suporte um banco de dados de localidades ainda mais específicas, revelando não somente os países cujos ataques foram lançados, mas também, as cidades de onde estes partiram.

Referências Bibliográficas

AB, M. *MySQL Reference Manual*: Mysql ab table and history of mysql. 2004.

ALBING, C. *et al. PHP 5 Power Programming*. Setembro 2004.

AMBER, S. W. *The Agile Unified Process (AUP)*. 2009. `http://www.ambysoft.com/unifiedprocess/agileUP.html`. Disponível em: <http://www.ambysoft.com-/unifiedprocess/agileUP.html>.

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.

REENSKAUG, T. The model-view-controller (mvc). In: *Its Past and Present*. [s.n.], 2003. Disponível em: <http://heim.ifi.uio.no/~trygver/2003% -/javazone-jaoo/MVC_pattern.pdf>.

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.

ANEXO A – Exemplo de log filtrado

Oct 25 12:28:40 bodacious sshd[4068]: Did not receive identification string from 99.189.250.94
Oct 25 12:29:34 bodacious sshd[4069]: Invalid user globus from 99.189.250.94
Oct 25 12:29:34 bodacious sshd[4069]: Failed password for invalid user globus from 99.189.250.94 port 49795 ssh2
Oct 25 12:29:37 bodacious sshd[4071]: Invalid user condor from 99.189.250.94
Oct 25 12:29:37 bodacious sshd[4071]: Failed password for invalid user condor from 99.189.250.94 port 54088 ssh2
Oct 25 12:29:44 bodacious sshd[4073]: Invalid user tomcat from 99.189.250.94
Oct 25 12:29:44 bodacious sshd[4073]: Failed password for invalid user tomcat from 99.189.250.94 port 56590 ssh2
Oct 25 12:29:44 bodacious sshd[4075]: Invalid user marine from 99.189.250.94
Oct 25 12:29:44 bodacious sshd[4075]: Failed password for invalid user marine from 99.189.250.94 port 58660 ssh2
Oct 25 12:30:01 bodacious sshd[4081]: Invalid user cadi from 99.189.250.94
Oct 25 12:30:01 bodacious sshd[4081]: Failed password for invalid user cadi from 99.189.250.94 port 53588 ssh2
Oct 25 12:30:08 bodacious sshd[4086]: Invalid user cady from 99.189.250.94
Oct 25 12:30:08 bodacious sshd[4086]: Failed password for invalid user cady from 99.189.250.94 port 64762 ssh2
Oct 25 12:30:12 bodacious sshd[4088]: Invalid user cai from 99.189.250.94
Oct 25 12:30:12 bodacious sshd[4088]: Failed password for invalid user cai from 99.189.250.94 port 56785 ssh2
Oct 25 12:30:15 bodacious sshd[4090]: Failed password for root from 99.189.250.94 port 63330 ssh2
Oct 25 12:30:22 bodacious sshd[4093]: Failed password for root from 99.189.250.94 port 54174 ssh2
Oct 25 12:30:42 bodacious sshd[4098]: Invalid user mythtv from 99.189.250.94
Oct 25 12:30:42 bodacious sshd[4098]: Failed password for invalid user mythtv from 99.189.250.94 port 57022 ssh2
Oct 25 12:30:46 bodacious sshd[4101]: Invalid user mythtv from 99.189.250.94
Oct 25 12:30:46 bodacious sshd[4101]: Failed password for invalid user mythtv from 99.189.250.94 port 60520 ssh2
Oct 25 12:31:11 bodacious sshd[4104]: Failed password for root from 99.189.250.94 port 59894 ssh2
Oct 25 12:31:54 bodacious sshd[4110]: Invalid user marine from 99.189.250.94
Oct 25 12:31:54 bodacious sshd[4110]: Failed password for invalid user marine from 99.189.250.94 port 50196 ssh2
Oct 25 12:32:11 bodacious sshd[4108]: Invalid user jboss from 99.189.250.94
Oct 25 12:32:11 bodacious sshd[4108]: Failed password for invalid user jboss from 99.189.250.94 port 58370 ssh2
Oct 25 12:32:25 bodacious sshd[4114]: Invalid user test1 from 99.189.250.94
Oct 25 12:32:25 bodacious sshd[4114]: Failed password for invalid user test1 from 99.189.250.94 port 50998 ssh2
Oct 25 12:32:28 bodacious sshd[4117]: Invalid user mythtv from 99.189.250.94
Oct 25 12:32:28 bodacious sshd[4117]: Failed password for invalid user mythtv from 99.189.250.94 port 63387 ssh2
Oct 25 12:32:31 bodacious sshd[4119]: Invalid user mythtv from 99.189.250.94
Oct 25 12:32:31 bodacious sshd[4119]: Failed password for invalid user mythtv from 99.189.250.94 port 54749 ssh2
Oct 25 12:32:40 bodacious sshd[4121]: Invalid user mythtv from 99.189.250.94
Oct 25 12:32:40 bodacious sshd[4121]: Failed password for invalid user mythtv from 99.189.250.94 port 60707 ssh2