UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

TECNOLOGIA EM ANALISE E DESENVOLVIMENTO DE SISTEMAS

JOSLEY ROSA

MAX LUIZ PFEFER FILHO

WILLIAN PESTANA

PROJETO DE UM PORTAL CATIVO SOCIAL PARA GERENCIAMENTO DE ACESSO AMIGÁVEL EM REDES IEEE 802.11

TRABALHO DE CONCLUSÃO DE CURSO

PONTA GROSSA

2013

JOSLEY ROSA

MAX LUIZ PFEFFER FILHO

WILLIAN PESTANA

PROJETO DE UM PORTAL CATIVO SOCIAL PARA GERENCIAMENTO DE ACESSO AMIGÁVEL EM REDES IEEE 802.11

Trabalho de Conclusão de Curso apresentado como requisito parcial à obtenção do título de Tecnólogo em Analise e Desenvolvimento de Sistemas, do Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná.

Orientador: Prof. Saulo Queiroz

PONTA GROSSA

2013

TERMO DE APROVAÇÃO

PROJETO DE UM PORTAL CATIVO SOCIAL PARA GERENCIAMENTO DE ACESSO AMIGÁVEL EM REDES IEEE 802.11

por

JOSLEY ROSA
MAX LUIZ PFEFFER FILHO
WILLIAN PESTANA

Este Trabalho de Conclusão de Curso (TCC) foi apresentado em 25 de março de 2013 como requisito parcial para a obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Msc. Saulo Queiroz
Prof. Orientador

_____
Dr. Daniel Costa de Paiva
Membro titular

_____
Dr. Lourival A. de Góis
Membro titular

_____
Drª Helyane Bronoski Borges
Responsável pelos Trabalhos
de Conclusão de Curso

_____
Drª Simone de Almeida
Coordenadora do Curso
UTFPR - Campus Ponta Grossa

RESUMO

PESTANA, Willian; PFEFFER, Max Luiz Filho; ROSA, Josley. **Projeto de um Portal Cativo Social para Gerenciamento de Acesso Amigável em Redes IEEE 802.11**. 2013. 50 f. Trabalho de conclusão de curso (Tecnologia em Analise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2013.

O processo de autenticação cumpre papel fundamental para o uso de recursos em uma rede sem fio IEEE 802.11. Um portal cativo social oferece uma nova abordagem em termos de autenticação amigável ao usuário leigo nestas redes. Este trabalho propõe um estudo sobre conceitos como tecnologia IEEE 802.11, métodos de autenticação em redes wireless e redes sociais, a fim de projetar e implementar um portal cativo social. A criação do sistema é detalhada neste trabalho. São identificados os problemas e dificuldades decorridas e possíveis melhorias.

Palavras-chave: Portal Cativo. Redes Sociais. Redes em Malha Sem Fio.

ABSTRACT

PESTANA, Willian; PFEFFER, Max Luiz Filho; ROSA, Josley. **Design of a Social Captive Portal for Friendly Access Management in IEEE 802.11 Networks**.2013. 50 f. Trabalho de conclusão de curso (Tecnologia em Analise e Desenvolvimento de Sistemas) - Federal Technology University. Ponta Grossa, 2013.

The authentication process performs a major role in the use of resources in an IEEE 802.11 wireless network. A social captive portal offers a new approach in terms of user-friendly authentication for layman users in these networks. This work proposes a study on concepts such as IEEE 802.11, methods of authentication in wireless networks and social networks services in order to design and implement a social captive portal. The creation of this project is detailed through this paper. It identifies problems, hindrances and possible improvements.

Keywords: Captive portal. Social Network Service.Wireless Mesh Networks.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS, SIGLAS E ACRÔNIMOS

AAA	*Authentication, Authorization, Accounting*
AP	*Access Point*
API	*Application Programming Interface*
CSS	*Cascading Style Sheets*
GNU	*Gnu's Not Unix*
IP	*Internet Protocol*
JS	*JavaScript*
LAN	*Local Area Network*
MAC	*Media Access Control*
MER	Modelo entidade relacionamento
OO	Orientadoaobjetos
OLSR	*Optimized Link State Routing Protocol*
ORM	*Object-relationalMapping*
PHP	*Hypertext PreProcessor*
QoS	*Quality of Service*
SQL	*Structured Query Language*
SSID	*Service Set Identifier*
URI	*Uniform Resource Identifier*
WAN	*Wide Wireless Network*
WDS	*Wireless Distribution System*
WEP	*Wired Equivalent Privacy*
WPA	*Wireless Protected Access*
YAML	*Yet Another Multicolumn Layout*

SUMÁRIO

1 INTRODUÇÃO

As tecnologias IEEE 802.11 (e.g. a, b, g, n) têm sido amplamente utilizadas para o estabelecimento de redes locais de acesso sem fio (do inglês *Wireless Local Area Network*, WLAN). Tal difusão pode ser explicada por fatores como baixo custo de seus equipamentos de conexão (e.g. Pontos de Acesso (AP), placas de rede), mobilidade, ampla aceitação pela indústria de dispositivos portáteis (e.g. celulares, *tablets*, *notebooks*) e possibilidade de operar em modo de acesso local ou de múltiplos saltos (i.e. redes em malha sem fio).

Não obstante as vantagens supracitadas, as tecnologias de acesso sem fio também apresentam desafios. Devido à natureza *broadcast* do meio sem fio, os recursos oferecidos por meio da referida tecnologia (e.g. conexão à Internet), estão sujeitos a serem acessados por usuários não autorizados. A solução clássica adotada no padrão IEEE (i.e. autenticação criptográfica) implica na necessidade de conhecimentos de configuração considerados avançados para usuários tipicamente leigos, o que constitui um fator limitante na segurança da tecnologia. Outro fator agravante decorre do fato de que nessa solução o usuário é responsável por gerenciar o compartilhamento de chaves de acesso entre os diferentes usuários da rede.

Diante das limitações de usabilidade associada à autenticação criptográfica, soluções baseadas em portais cativos têm sido propostas, como por exemplo, técnicas de prevenção de ataques ao endereço MAC (*Media Access Control*) (CHOI *et al*., 2011), soluções baseadas na arquitetura SWITCHaai/Shibboleth (KROPF; MONAKHOC; SCHILLER, 2011), gerenciamento do tráfego da rede (KOHT-ARSA;PHONPHOEM; SANGUAPONG, 2009), acesso à internet a baixo custo a longas distâncias (CALAFATE *et al*, 2007).

Em um portal cativo, um usuário recebe um *login* único, o qual não é compartilhado com nenhum outro usuário e, com este *login* particular, é garantido ao usuário acesso à rede. Algumas soluções de portais cativos são o WifiDog[1], Coova[2], AuthPuppy[3] e ChilliSpot[4]. Tendo em vista manter um controle sobre os usuários em um ambiente corporativo, foi desenvolvida esta tecnologia de portais cativos, pela qual é possível determinar quais funcionários poderão utilizar a rede, com base em um banco de dados mantido pela empresa. Este tipo de recurso pode ser encontrado em diversos ambientes, para prover acesso a pontos de acesso (CALAFATE *et al*, 2007).

[1] http://dev.wifidog.org/
[2] http://coova.org/
[3] http://www.authpuppy.org/
[4] http://www.chillispot.info/

Através de um portal cativo social, um usuário comum pode disponibilizar seus recursos (*i.e.,* acesso à internet) àqueles, com os quais possui uma relação em uma rede social. O usuário tem a possibilidade de determinar quais destes amigos poderão ter acesso de forma transparente e amigável, dispensando qualquer conhecimento avançado em redes.

1.1 OBJETIVO

Projetar e implementar um portal cativo que possibilite autenticação e autorização para uso dos recursos providos por um ponto de acesso sem fio (e.g. Internet) através das relações de amizades estabelecidas pelo proprietário do AP na rede social Facebook®[5].

1.2 OBJETIVOS ESPECÍFICOS

- Pesquisar e estudar soluções de autenticação em redes de acesso sem fio.
- Estudar o funcionamento de portais cativos.
- Pesquisar, estudar e selecionar uma implementação de portal cativo de código aberto.
- Projetar uma solução de portal cativo com autenticação baseada em relações sociais estabelecidas em redes de relacionamento online.
- Demonstrar o processo de autenticação social proposto.

[5]https://www.facebook.com/

2 REFERENCIAL TEÓRICO

2.1 REDES LOCAIS IEEE 802.11

A utilização de redes com cabos tem se mostrado muito útil ao longo do tempo, oferecendo baixo custo de manutenção e alta disponibilidade.Com o advento de novos dispositivos como *tablets*, *smartphones* e similares, existe a demanda para o acesso aos recursos da rede para estes dispositivos de modo a atender esses usuários e.g. troca de arquivos compartilhamento de internet.

Estes novos dispositivos na sua maioria não têm nenhuma interface LAN (Local area network) disponível, o que não permite conexão com cabos. Uma solução viável então é a idealização de uma rede *wireless*, que pode perfeitamente ser acoplada a rede LAN já existente. "*Uma rede sem fio é um sistema que interliga vários equipamentos fixos ou móveis utilizando ondas eletromagnéticas como meio de transmissão*" (IEEE 802.11a).

O padrão IEEE 802.11 especifica todo o funcionamento das redes locais sem fio (TANENBAUN,2006). Desenvolvido com auxilio de padrões proprietários foi estendido em padrão aberto devido às vantagens de interoperabilidade, baixo custo, confiabilidade de projeto, entre outras.

O IEEE802.11 define padrões nas camadas de enlace e física(TANENBAUN, 2006). A classificação das redes é definida pela área de cobertura e potência dos transmissores e receptores envolvidos no processo de comunicação. Na versão IEEE802.11b foram implementadas melhorias como aumento na taxa de transferência para de 1 e 2Mbps para 5 e 11Mbps.

Para atender determinados segmentos, são realizadas mudanças nesse padrão, como melhoramentos em taxas de transferência, diminuição de interferência, aumento do alcance dos pontos de acesso, compatibilidade entre os padrões, e melhoramentos nos mecanismos de segurança. Estas são organizadas através de subdivisões, diferenciadas por sufixos (e.g. IEEE 802.11a, IEEE 802.11n).

2.2 REDES SEM FIO DE MÚLTIPLOS SALTOS

Um cenário comum de redes sem fio consiste em APs, conectados à internet ou a alguma rede qualquer. Nesses cenários, o AP fica centralizado de forma que todos os dispositivos tem que solicitar o acesso diretamente a ele, e este terá que gerenciar os dispositivos que podem ter acesso e os que não (FLICKENGER, 2008). Em outra abordagem é possível utilizar a padrão 802.11 em um modo baseado em

ad-hoc (estrutura onde dispositivos se comunicam diretamente sem a intermediação do AP), a fim de se implementar redes em malha sem fio ou simplesmente redes *mesh* (ALBUQUERQUE *et al.,* 2006).

Uma rede em malha, embora baseada no modo *ad-hoc,* possui rotas dinâmicas. Nessa estrutura, todos os roteadores se tornam nós e não possuem um *gateway*(dispositivo que interliga duas redes distintas) fixo, pois, dependendo da condição do fluxo de dados e a situação dos roteadores, é plausível que uma requisição use rotas distintas a cada momento. São autoconfiguráveis e realizam a comunicação através de múltiplos saltos entre os nós. Estas redes vêm ganhando cada vez mais espaço e atenção por parte da comunidade cientifica em função de sua infraestrutura (LEAL; REIS, 2010). Na Figura 1 mostram-se vários nós em uma rede em malha, estes se conectam aos seus vizinhos.

Uma das principais vantagens da rede em malha é a possibilidade de estabelecer uma rede comunitária sem fio com conexão à internet (ALBUQUERQUE *et al.,* 2006). Uma rede em malha pode prover meios de diminuir custos, uma vez que nem todos os roteadores precisam ter acesso à internet como em uma estrutura centralizada, ao mesmo tempo em que os que possuem irão compartilhar com os outros (AGUIAR *et al.*, 2008).

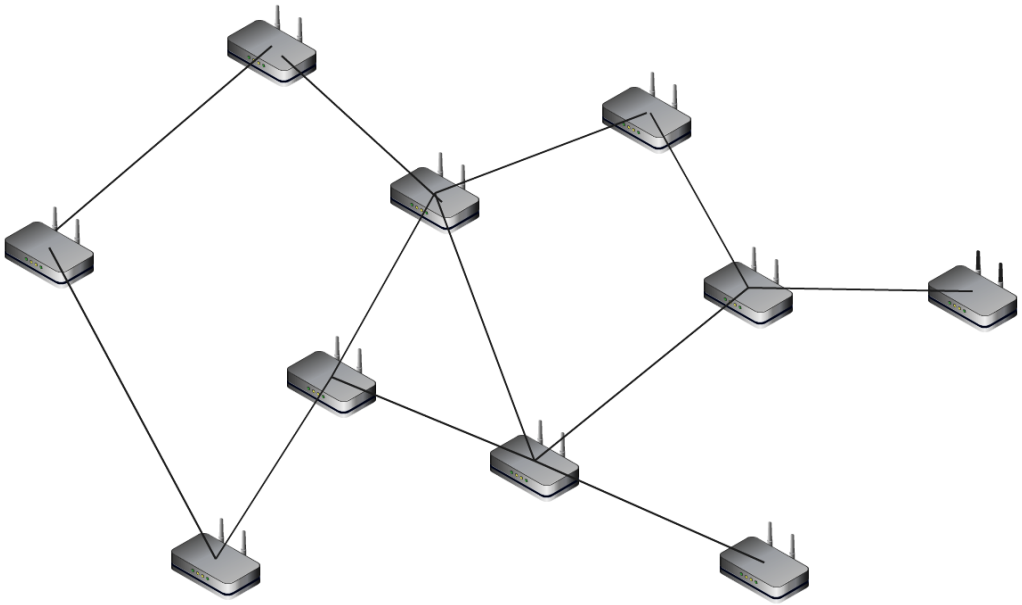

Figura 1 – Rede em malha sem fio
Fonte: Autoria própria

Em uma rede em malha comunitária, ao contrário de outras, é necessário que os usuários colaborem entre si para gerenciá-la e mantê-la funcionando. Devido ao baixo custo, o alcance dos pontos de acesso pode ser baixo, considerando uma vizinhança por exemplo. Portanto existe a necessidade de que os usuários colaborem e mantenham seus equipamentos sempre ligados e no melhor estado

possível, para evitar um eventual desmembramento de parte da rede.

Para ser estabelecida uma rede *mesh*, os roteadores precisam utilizar algum tipo de protocolo, como o OLSR (*Optimized Link State Routing Protocol*). Este protocolo pode traduzir-se como serviço otimizado para estado de conexão, suporta o protocolo IPV6 e é usado em diversas redes comunitárias pelo mundo (FLICKENGER *et al*., 2008).

O OLSR implementa um tipo de roteamento chamado de proativo, ou seja, enquanto está rodando o OLSR, o nó envia em intervalos determinados, uma mensagem por *broadcast*, assim os nós vizinhos podem identificar a sua presença e responder à mensagem, dessa maneira o OLSR consegue manter uma tabela de roteamento da rede (FLICKENGER *et al*., 2008).

2.3 MÉTODOS DE AUTENTICAÇÃO PARA ACESSO EM REDES SEM FIO IEEE 802.11.

Para um dispositivo se associar a um AP, primeiramente há a necessidade de se autenticar no mesmo, para só então utilizar os recursos disponibilizados e se comunicar com a rede. Alguns tipos de autenticação que serão explanados nesse trabalho são acesso aberto, WEP (*Wired Equivalent Privacy*), a mais utilizada por usuários comuns, e autenticação através de portais cativos.

2.3.1 ACESSO ABERTO

Utilizando esse modo, o AP permite que qualquer dispositivo se associe a rede, informando o SSID (*Service Set Identifier*). O dispositivo pode obter o nome da rede através de pacotes sem criptografia enviados por *broadcast* pelo AP. Esse modo de acesso permitirá a qualquer um utilizar os seus recursos.

A Figura 2 apresenta o processo de associação à uma rede de acesso aberto.

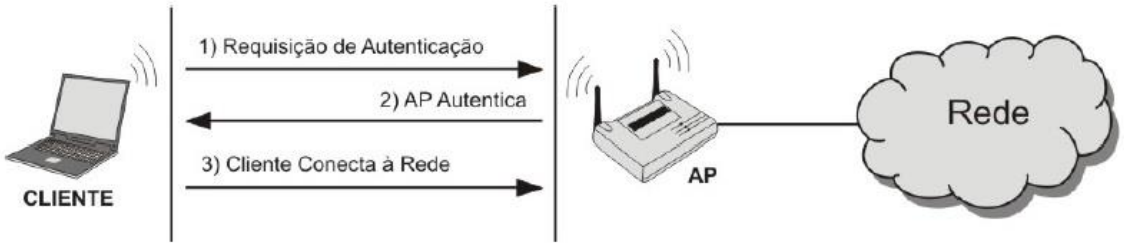

Figura 2 – Acesso Aberto
Fonte: GONÇALVES; LINHARES, 2006. p. 4.

2.3.2 WEP

A autenticação WEP é um protocolo de segurança do padrão IEEE 802.11, embora possua vulnerabilidades que podem ser facilmente exploradas como tamanho das chaves, além de ser estática e compartilhada por todos que utilizam o AP, ela possui apenas 40 bits, que é um tamanho suficiente para ser quebrada por força bruta (GONÇALVES; LINHARES, 2006). Apesar disso, ainda é o mais utilizado por usuários comuns (GONÇALVES; LINHARES, 2006).

Outros protocolos de autenticação tem o objetivo de suprir as desvantagens do WEP, como WPA e WPA2, mas mesmo assim, o WEP ainda é o mais usado. (GONÇALVES; LINHARES, 2006).Isso devido a não serem tão convenientes ao meio doméstico e ao usuário leigo e mais as redes corporativas, e mesmo em algumas destas, essa mudança pode implicar grandes custos em relação à aquisição de hardware e, portanto, certas vezes inviável para elas (GONÇALVES; LINHARES,2006).

Nesse modo de autenticação, o dispositivo, para se associar ao AP, deve possuir uma chave. O dispositivo envia uma solicitação e o AP envia um texto, chamado texto-desafio, sem criptografia. O dispositivo deve criptografar o texto e enviar novamente ao AP que irá descriptografar com sua própria chave. Caso os textos sejam equivalentes, é concedido o acesso.

Embora mais seguro que o acesso aberto, o modo de autenticação por chaves compartilhadas ainda não é adequado, pois tem a vulnerabilidade do tamanho de 40 bits, e outras oriundas de ser um tipo de autenticação WEP, onde o usuário não consegue disponibilizar com praticidade seus recursos a apenas a usuários previamente selecionados, pois para isso deve compartilhar sua chave com outros.

A Figura 3 apresenta o funcionamento do modo de autenticação em redes com compartilhamento de senhas e chaves compartilhadas.

Figura 3 – Chaves compartilhadas
Fonte: GONÇALVES; LINHARES, 2006. p.4.

2.3.3 PORTAIS CATIVOS

É uma técnica de autenticação que impõe que os usuários apresentem suas credenciais (normalmente usuário e senha) para conseguir ter acesso à rede (KOHT-ARSA; PHONPHOEM; SANGUANPONG, 2009). Genericamente, a rede é configurada como tendo acesso aberto, e como ferramenta de autenticação são usados os navegadores. Para ocorrer a autenticação o usuário deve, após selecionar a rede, abrir um navegador e informar seus dados, para então ter acesso.

Um portal cativo se propõe a implementar o conceito de AAA (*Authentication*, *Authorization*, *Accounting*).*Authentication* ou autenticação se refere à confirmação de que o usuário que solicitou os serviços é válido na rede requerida, isso é feito no momento que o ele informa suas credenciais; *Authorization* ou autorização se refere à concessão de funcionalidades especificas a um usuário baseando-se na autenticação; *Accounting* ou, literalmente, contabilidade, se refere à coleta de informações a respeito de um usuário e as funcionalidades por ele utilizadas(KROPF; MONAKHOC; SCHILLER, 2011).

Para o funcionamento desse tipo de autenticação, deve existir um AP específico na rede em questão, compatível com o portal cativo escolhido. Quando o usuário solicitar a sua associação à aquela rede, esse AP irá redirecioná-lo a um servidor onde será apresentada a página do portal, com o login, senha. Nesse servidor, ocorrerá a autenticação de acordo com os usuários cadastrados no mesmo ou regras propostas para que isso ocorra. O servidor pode ficar localizado na rede interna ou mesmo na internet.

Portais cativos podem ser encontrados como meio de acesso a internet em hotéis, restaurantes e estabelecimentos comerciais em geral, e também nos meios acadêmicos e corporativos, de forma a possibilitar acesso a alunos e funcionários. Existem soluções tanto abertas quanto comerciais quando se trata de portais cativos. Softwares como: Authpuppy, ChilliSpot, CoovaChilli, entre outros podem ser enumerados como soluções abertas, enquanto AirMarshall, LogiSense são comerciais.

O meio de autenticação por portais cativos permite acesso simples, intuitivo e controlado aos usuários autorizados, proporcionando uma ótima solução para, por exemplo, empresas e universidades. O problema é quando se aplica esta solução em um ambiente doméstico, embora permita que o dono da rede possa controlar quem utiliza seus recursos, ainda assim é necessário conhecimento, que normalmente está aquém de usuários leigos, para realizar o gerenciamento e controle de acesso da rede.

Desta maneira, sendo um portal cativo a solução mais próxima que se adapta ao problema proposto neste trabalho, existe a possibilidade de modificar um portal cativo, de forma que a customização de seu ponto de acesso se torne simples e intuitiva, ou seja, sem a necessidade de possuir conhecimentos técnicos.

2.4 REDES SOCIAIS ONLINE

Pode-se dizer que uma rede social online é basicamente um ambiente WEB onde um usuário pode criar um perfil público ou “semi-público”, uma lista de outros usuários e visualizar conexões existentes (BOYD; ELLISION, 2007). Essas conexões entre usuários podem ser denominadas genericamente como laços sociais, podendo variar de nomenclatura de acordo com o caso (e.g. amizade, contato, seguidor).

As redes sociais funcionam a partir de um perfil criado pelo usuário, onde ele pode disponibilizar informações pessoais, *hobbies*, escolaridade, informações profissionais ou qualquer coisa que seja permitida por essa determinada rede (SERRANO, 2011).Além do perfil, existem outros importantes elementos: atualizações, que permitem ao usuário descobrir conteúdo com praticidade, encorajando-o a explorar conteúdo compartilhado por seus amigos; comentários permitem opinar em determinado conteúdo; avaliações, muitas redes sociais permitem avaliar o conteúdo, como, por exemplo, a opção “curtir” que o Facebook® disponibiliza; favoritos, várias aplicações disponibilizam funcionalidades que permitem aos usuários organizar conteúdos permitindo à aplicação dar “sugestões” de conteúdo ao usuário; lista de mais populares, existem aplicações que selecionam por meio de dados estatísticos os conteúdos mais visualizados ou mesmo melhor

avaliados, assim como no Youtube®[6], onde os vídeos mais populares são exibidos com destaque; metadados, aplicativos como Youtube® disponibilizam conteúdo tipicamente associado a metadados, como *tags* e títulos, que são essenciais aos mecanismos de recuperação de conteúdo (ALMEIDA; BENEVENUTO; SILVA, 2011).

As redes sociais online são divididas em diversos temas, sendo, em sua maioria, genéricas, que tem o objetivo de criar conexões mesmo interesse, proporcionando ao usuário compartilhamento de conteúdo, bate-papo e diversas maneiras de socialização. Nessas redes pode-se dizer que o foco dos conteúdos e funcionalidades, estão voltados a um contexto lúdico, promovendo uma interação informal e recreativa entre os usuários (BARANAUSKAS *et al.*, 2009). Hoje os mais populares nessa categoria são o Facebook (atualmente, a rede social com mais usuário ativos), Google+®[7], Youtube® e Twitter®[8]·. Existem também redes sociais temáticas ou especializadas, onde o foco não é a socialização, embora possa ser uma consequência, e sim oferecer possibilidades de conexão com outros usuários e funcionalidades com base em uma temática particular, atendendo as necessidades de um determinado segmento (BARANAUSKAS *et al.*,2009). As mais comuns dentro desse segmento tem como temas a educação e empregos, vale ressaltar alguns exemplos como o LinkedIn®[9] que tem como temática contatos profissionais, também podemos citar o Livemocha®[10]que é voltado ao apoio ao ensino de idiomas.

Além das funcionalidades nativas, as redes sociais mais populares permitem o desenvolvimento de aplicativos que podem ser integrados a elas, por isso nestas podem ser encontrados os mais variados tipos de aplicações. Jogos como AngryBirds[11], FarmVille [12] possuem versões para o Facebook®, e se aproveitam deste meio social para expandir suas funcionalidades e proporcionar ao usuário modos cooperativos e competitivos. Aplicativos como TravelMap®[13] no Facebook® permite aos usuários criarem mapas de suas viagens e compartilhar com seus amigos.

Existe uma gama muito grande de aplicativos com diversas funcionalidades que podem, por exemplo, usar informações de conexões entre usuários para criar aplicações sociais diferenciadas.

[6] http://www.youtube.com
[7]https://plus.google.com
[8]https://twitter.com/
[9] http://linkedin.com/
[10] http://livemocha.com/
[11] http://www.angrybirds.com/
[12]https://www.facebook.com/FarmVille
[13]https://www.facebook.com/TravellerspointTravelMap

2.5 PORTAL CATIVO PARA GERENCIAMENTO DE ACESSO AMIGÁVEL

O objetivo deste trabalho é utilizar as tecnologias supracitadas para desenvolver um método de gerenciamento de redes IEEE 802.11 de forma a possibilitar ao usuário, uma experiência amigável, deixando detalhes técnicos transparentes durante o uso da rede.

Com o emprego destas tecnologias, foram alteradas características fundamentais do processo de autenticação em uma rede wireless utilizando portais cativos. Para ingresso na rede através desse método, é explorada uma relação em uma rede social online que possibilita a entrada de um novo individuo na rede com base na sua relação com dono da mesma, sem a necessidade de pré-cadastrar usuários.

Da mesma forma que não precisamos de pré-cadastro para novos indivíduos, quando existe a inclusão de novos APs , existe a possibilidade da rede se acomodar sem a necessidade de interferência humana utilizando redes mesh. Dentro dessa rede em malha as rotas em que ocorrerem os saltos são definidas com base em APs possuídos por indivíduos aos quais esta relacionado, com acesso a internet, e não apenas através da menor a rota.

Assim é possível se solicitar uma conexão a um AP informando suas credenciais (*login* e senha) de uma rede social online, e ser autenticado a ela caso tenha alguma relação online com o dono desse equipamento.

3 FERRAMENTAS PARA AUTENTICAÇÃO SOCIAL

Como ferramentas para o trabalho é possível citar o OpenWRT (*firmware* GNU/Linux para um AP), um AP compatível com uma versão do OpenWRT, um portal cativo de código aberto e compatível com o firmware e uma API (*Application Programming Interface*) para recuperação de dados públicos em redes sociais, no caso a API do Facebook®.

3.1 *FIRMWARE* GNU/LINUX (OPENWRT)

Os fabricantes de pontos de acesso desenvolvem *firmwares* para que seus hardwares funcionem corretamente para a função que foram projetados, porém não permitem o acesso ao código destes *firmwares* tornando a customização impossível.

Pensando nisso algumas comunidades desenvolveram *firmwares* em Linux capazes não somente de controlar de forma mais eficiente os pontos de acesso, mas também com código aberto possibilitando que sejam modificados. Uma das vantagens de se ter Linux como base é que existem outros pacotes implementados que instalados removidos ou ainda modificados. Outra vantagem é que podemos remover limitações colocadas pelo fabricante.

A versão escolhida para esse trabalho é a 10.03, conhecida como Backfire, que é a mais recente do OpenWRT, que não seria indicada para pontos de acesso mais antigos por sua pequena memória e seu processador com baixo poder de processamento, tornaria o sistema impraticável mesmo que conseguisse colocar dentro da memória flash.

Na Figura 4 pode ser observado o histórico do OpenWRT ao longo de suas versões.

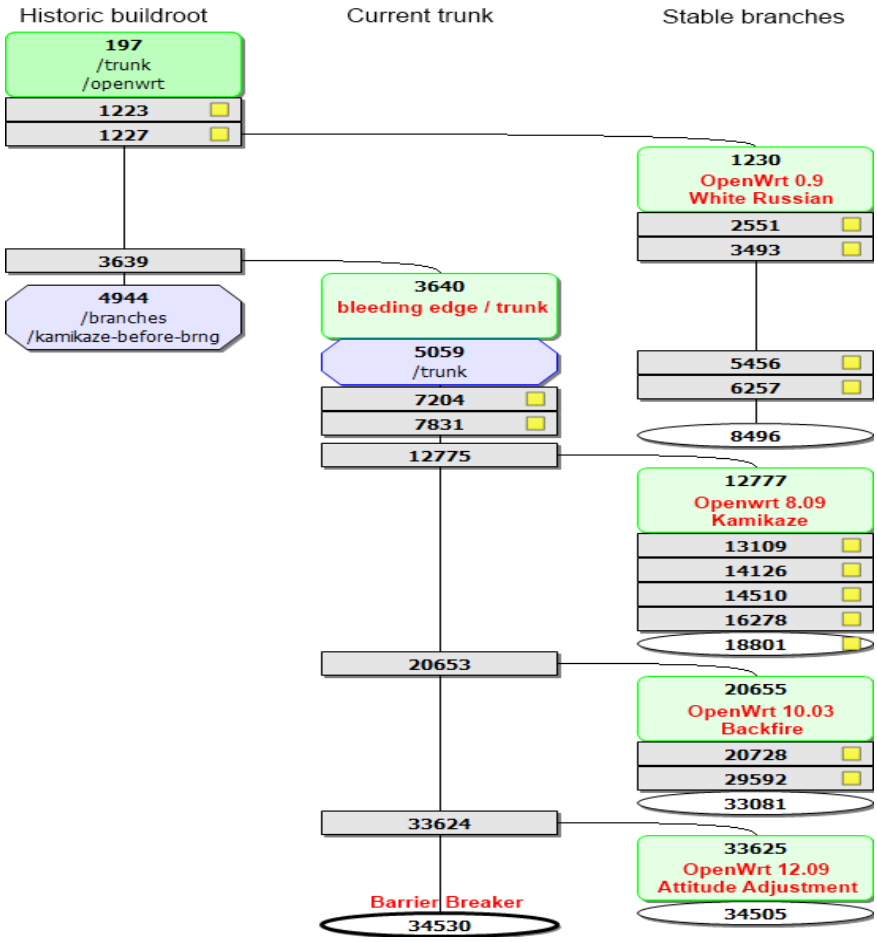

Figura 4 - Evolução do OpenWRT
Fonte: wiki.openwrt.org/about/history

A interface gráfica desta versão é conhecida como *LuCi,* mostrado na Figura 5.

.

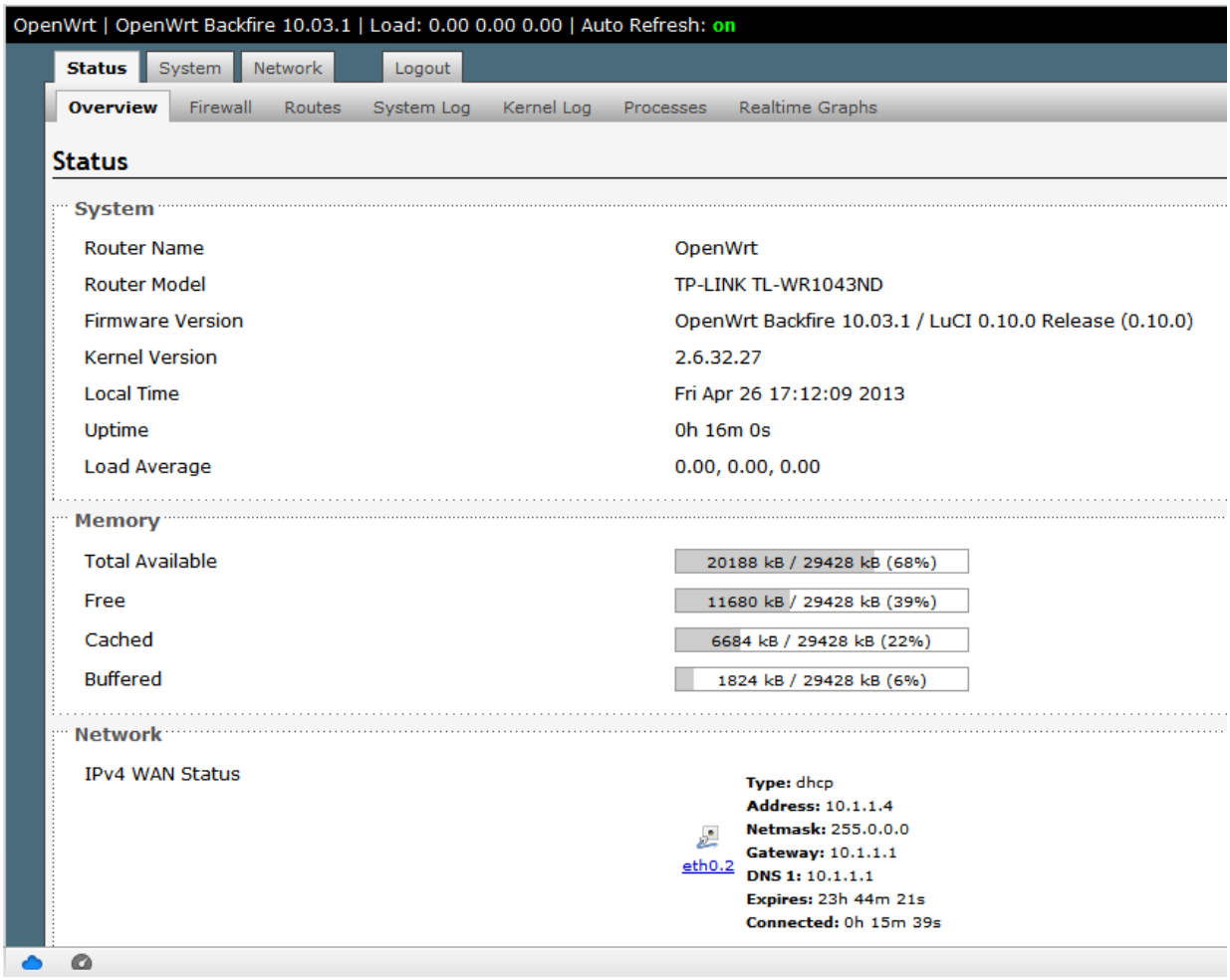

Figura 5 - Interface LuCi
Fonte: Interface do firmware OpenWRT

3.2 ROTEADOR TP-LINK WR1043ND

Para criar um processo de autenticação que não use compartilhamento de senha, temos que ter um hardware que atenda nossas expectativas quanto à memória e processamento e compatibilidade com o firmware escolhido.

O modelo escolhido foi TP-Link WR1043ND por ser o melhor custo beneficio entre os pré-selecionados possuindo 16MB de memória, e um processador que proporciona ao sistema instalado um desempenho adequado.

3.3 AUTHPUPPY

Para a criação de um modo de autenticação social com gerenciamento amigável, foi necessário selecionar, entre várias opções, um portal cativo já existente e que possa ser mais facilmente adaptável às necessidades deste trabalho, de código aberto e compatível com o firmware OpenWRT, anteriormente citado. Com esse intuito foram analisadas várias aplicações, dentro destas, foi selecionada uma que atende as expectativas do projeto.

Wifidog[14]: esse portal cativo consiste em duas partes: uma que fica no AP, escrita em linguagem C, que redireciona o usuário até a sua segunda parte, que fica no servidor, onde está o portal e os métodos de autenticação, este escrito em PHP com banco de dados PostgreSQL. Embora seja funcional, é pouco flexível, e, por ter sido descontinuado em 2006, sua documentação é escassa e sua comunidade é pouca ativa.

Authpuppy [15]: foi feito para substituir o Wifidog, apresenta as mesmas características quanto às duas partes da aplicação, porém o Authpuppy é muito flexível, e é possível adicionar novas funcionalidades através da criação de *plug-ins*. Sua documentação é detalhada e a comunidade prestativa.

ChilliSpot[16]: feito em linguagem C e, assim como o WifiDog, também foi descontinuado, portanto, sua comunidade é fraca e sua documentação deixa a desejar.

ChilliCoova [17]: feito para substituir o ChilliSpot, possui documentação e comunidade prestativas.

A principio foi escolhido o WifiDog para este trabalho, porém, houve inúmeras dificuldades em relação à documentação e *feedback* da comunidade. Por essa razão a escolha passou a ser do Authpuppy, que possui as vantagens anteriormente descritas.

Foi desenvolvido através do *framework* PHP *Symfony*e banco de dados *PostgreSQL* e se encontra em uma versão estável. É uma solução de software com o objetivo de fornecer um portal cativo, possuindo autenticação e autorização, além de funcionalidades para monitoramento de uma rede sem fio.

O Authpuppy foi desenvolvido de uma forma a dar flexibilidade ao desenvolvedor para criar novas funcionalidades. Por este motivo, o Authpuppy não fornece nada mais que uma interface de autenticação sem senha, não identificando quem está se conectado à internet. Todas as outras funcionalidades são descritas através de *plug-ins*que são desenvolvidos por pessoas ativas na comunidade. Para a criação de novos *plug-ins* e funcionalidades, é necessário ao usuário

[14] http://dev.wifidog.org/
[15] http://www.authpuppy.org/
[16] http://www.chillispot.info/
[17] http://coova.org/

conhecimentos em PHP para desenvolvimento através do framework Symfony assim como em banco de dados PostgreSQL.

3.4 API DO FACEBOOK

O Facebook é uma rede social criada em 4 de fevereiro de 2004 por Mark Zuckerberg e hoje é a mais popular rede social existente na internet com mais de 900 milhões de usuários, sendo que o Brasil ocupa o segundo lugar em contas existentes, com aproximadamente 46 milhões de usuários. A interface disponibilizada pelo software também possui instrumentos, fornecendo facilidades ao desenvolvedor.

A API é o principal meio de recuperar dados de um perfil do Facebook e é considerada uma das mais completas existentes entre as redes sociais. Nesta API, alguns dados dos usuários são considerados públicos, como, por exemplo, seu nome completo e a foto principal do perfil, portanto não há necessidade de saber a senha do perfil do usuário para alguém recuperar estes dados. Entretanto, outros dados somente podem ser recuperados com o consentimento do usuário, ou seja, são liberados somente após a inserção da senha do perfil. Por outro lado, existem também dados que não são liberados pela API, como o endereço eletrônico (*i.e* e-*mail*). Além disso, é possível inserir e apagar objetos de dentro de um perfil, sem a necessidade de estar na página do Facebook.

Para ter acesso a esta API é necessário se cadastrar no que é chamado de *Facebook developers*. Após o cadastro o usuário deverá criar um aplicativo e este possuirá duas chaves chamadas *API Key* e *API Secret*, e é com o uso desta a única maneira de garantir o acesso de um aplicativo externo a API.

4 ARQUITETURA DA SOLUÇÃO PROPOSTA

Os elementos básicos que compõem o sistema de autenticação social são: um servidor AAA incluindo o banco de dados correspondente, um ponto de acesso ou roteador com cliente um cliente AAA, a base de dados de uma rede social existente e os usuários da rede sem fio. Em particular para este cenário de teste e desenvolvimento esses elementos foram baseados nas seguintes tecnologias: servidor AAA (Linux Mint 13 e portal cativo AuthPuppy com banco de dados PostgreSQL), AP (TP-Link WR1043ND com OpenWRT), base de dados do Facebook. A Figura 7 apresenta graficamente a arquitetura do projeto.

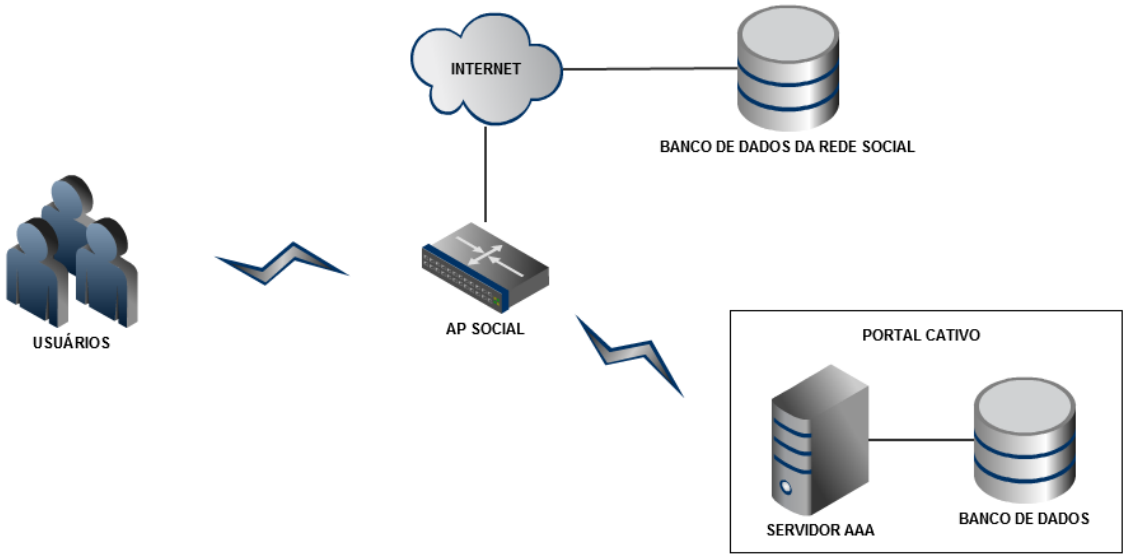

Figura 6 – Arquitetura do projeto
Fonte: Autoria própria

O roteador ficará sempre ligado e com o Wifidog ativado. No momento que o usuário faz uma requisição ao roteador, o Wifidog verifica que dado usuário (neste caso, o hardware) não está autenticado e exibirá o portal cativo proveniente do servidor. O usuário será redirecionado a pagina de autenticação do Facebook. Neste cenário, o Facebook é a única página que possui liberdade de acesso, mesmo sem autenticação. Após a autenticação no Facebook, o servidor retorna uma mensagem ao Wifidog, dando permissão de acesso ao usuário.

5 PROJETO E IMPLEMENTAÇÃO DO PORTAL CATIVO SOCIAL

Para a implementação da solução proposta foi necessário usar um roteador compatível com o *firmware* OpenWRT e, após uma análise de custo-benefício entre os roteadores mais recomendados pela comunidade, foi decidido pela aquisição do TP-LINK WR1043ND (Figura 8), este apresentando total compatibilidade com o firmware mencionado anteriormente. Este firmware permite seu uso com WDS (*Wireless Distribution System*), múltiplas interfaces sem fio virtuais, estatísticas, acesso SSH, QoS, dentre outras funcionalidades, dos quais o *firmware* original não disponibiliza ao usuário.

Figura 7 – Roteador TP-Link WR1043ND
Fonte: www.tp-link.com.br, acesso em 18/03/2013.

Após a instalação do *firmware* OpenWRT (como descrito no Apêndice A), foi averiguado que a porta WAN não estava funcionando. Após pesquisas, foi visto que este é um problema que já havia sido informado por outros usuários, afetando somente a versão 1.8 deste roteador. A solução proposta pela comunidade é realizar um *downgrade* do firmware original e então instalar o *firmware* OpenWRT.

Para continuar o projeto, foi necessário instalar o Wifidog. Isto é feito pela interface do OpenWRT, que pode ser acessada através do IP 192.168.1.1 no navegador. A configuração do Wifidog é simples, sendo necessário somente editar um arquivo chamado wifidog.conf. Neste arquivo é preciso informar o domínio de acesso ao servidor para que o Wifidog saiba onde está o Authpuppy. Caso o projeto esteja em executado localmente, este não estará hospedado em nenhum servidor, ou seja, não possui um domínio. Isto é facilmente resolvido criando um domínio local através do arquivo *hosts* no OpenWRT. Neste arquivo deverá ser informado o IP do servidor, portanto este deverá ter um IP fixo. Com isso, toda a configuração necessária no roteador está concluída.

O banco de dados usado neste projeto é o PostgreSQL. Após criar o banco de dados que será usado pelo Authpuppy, é preciso instalá-lo no servidor. Será

possível acessar o portal através de *localhost* no navegador. A primeira página para qual o navegador será redirecionado será a instalação propriamente dita do Authpuppy (descrito no Apêndice B), na qual será necessário dar permissão de acesso/leitura a algumas pastas específicas do portal (descrito no Apêndice B) e informações do banco de dados anteriormente criado, como nome do banco e senha de acesso. No final da instalação serão criadas várias tabelas no banco de dados, essenciais para o funcionamento do portal cativo.

As funções do Authpuppy são descritas através de *plug-ins*. Portanto, para não afetar o núcleo do portal e seu banco de dados, foi preciso criar um *plug-in* (descrito no Apêndice F), com o objetivo substituir o atual *plug-in* de autenticação do Authpuppy. Este novo *plug-in* utilizará uma conta já existente do Facebook para efetuar o login na rede e fornecer acesso à internet. Além disso, novas tabelas foram criadas no banco de dados. No entanto, a criação desta foi feita de forma automática pelo framework, não sendo necessário cria-las manualmente.

O novo *plug-in* de autenticação irá adicionar quatro novas tabelas ao banco de dados do Authpuppy: usuário, facebook, roteador, usuário_roteador. A tabela usuário armazenará um código do usuário que fará o relacionamento entre o perfil do usuário da rede social e os roteadores que esta possui. A tabela Facebook irá guardar a identificação de todos os perfis que possuírem um ou mais roteadores associados. A tabela roteador será usada para armazenar o IP que cada roteador possui e, a quarta tabela, usuário_roteador, fará o relacionamento das tabelas usuário e roteador. Esta última tabela foi necessária, por que um usuário pode possuir um ou mais roteadores e um roteador pode possuir um ou mais donos, definindo-se, portanto, um relacionamento de muitos-para-muitos, para uma descrição mais detalhada das tabelas (ver o Apêndice D). O MER (Modelo entidade relacionamento) projetado para esse trabalho pode ser observado na Figura 9.

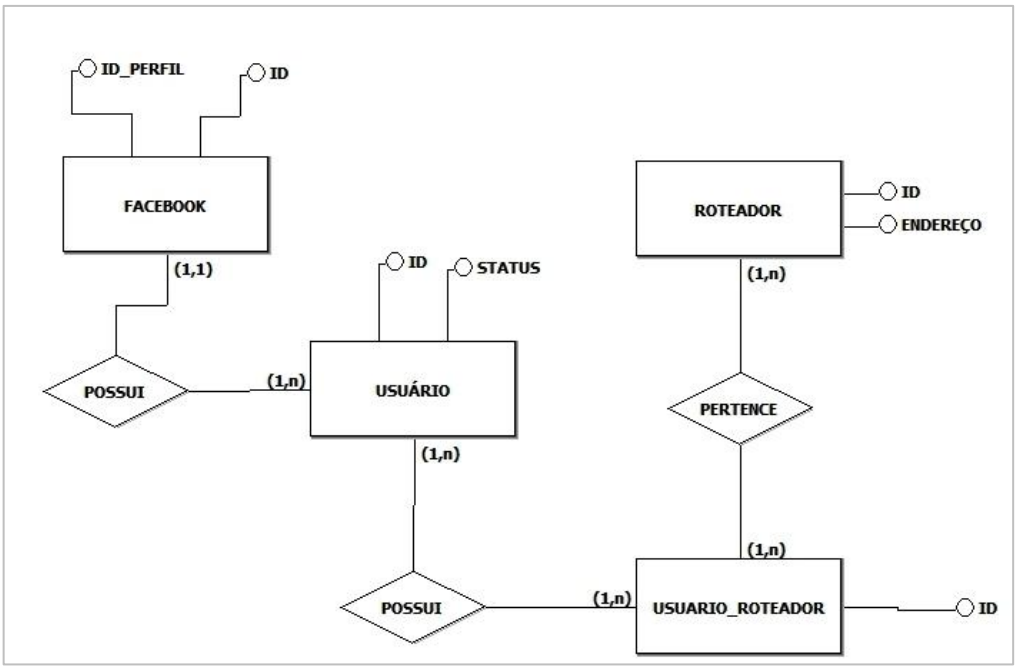

Figura 8 – Modelo entidade relacionamento para o projeto
Fonte: Autoria própria

Por ser um *framework* orientado à objetos, é possível realizar alterações no banco de dados utilizando classes e objetos sem qualquer uso de SQL. Para tal, é necessário criar um arquivo YAML chamado *schema* (descrito no Apêndice E).

Agora, para criar as tabelas que serão usadas pelo *plug-in*, é necessário criar arquivos chamados *Migrations*. Estes arquivos definem, em linguagem PHP, como serão as tabelas e seus respectivos atributos. Com o *plug-in* e as tabelas criadas, foi iniciada a implementação com o primeiro objetivo de recuperar dados de um perfil do Facebook®. Isso foi feito usando um *plug-in* disponível na comunidade chamado sfMelody. Este *plug-in* foi originalmente criado para permitir o acesso a API de algumas redes sociais a projetos de *websites* desenvolvidos usando o *framework* Symfony. Este permitiu a recuperação de dados de um perfil do Facebook® autenticado, tais como ID, nome e lista de amigos.

A autenticação padrão do Authpuppy funciona capturando a URI (*Uniform Resource Identifier*) do *request,* usando como valor para o botão de autenticação. Este URI informa de qual roteador veio a requisição de acesso e qual é a página requisitada pelo usuário, é criado pelo Wifidog e é entregue ao Authpuppy no momento que este é convocado pelo Wifidog. Compreendendo este aspecto, o *plug-in* foi feito para que a página de autenticação redirecione o usuário para a página de autenticação do Facebook. Quando a autenticação é bem sucedida, o usuário é redirecionado novamente o portal e seus dados de perfil são recuperados. Então, através da mensagem que o Wifidog enviou ao portal, é verificado de qual IP veio a requisição. Desta forma é possível saber qual é o roteador usado, visto que os roteadores em uma rede sempre possuem endereços distintos. A partir disto, é

verificado se este roteador está cadastrado e possui um proprietário relacionado no banco de dados. Se não possui, um novo usuário e perfil da rede social são gravados no banco de dados e é feito o relacionamento destes em outra tabela. Com isso, o usuário recebe autorização de acesso à internet. Caso já possuir um proprietário relacionado e o usuário autenticado não for o próprio, é analisado se o código de identificação no Facebook do proprietário existe na lista de amigos do autenticado. Caso afirmativo, é garantido o acesso à internet através do roteador. A Figura 10 apresenta um fluxograma descrevendo todo o processo de autenticação social.

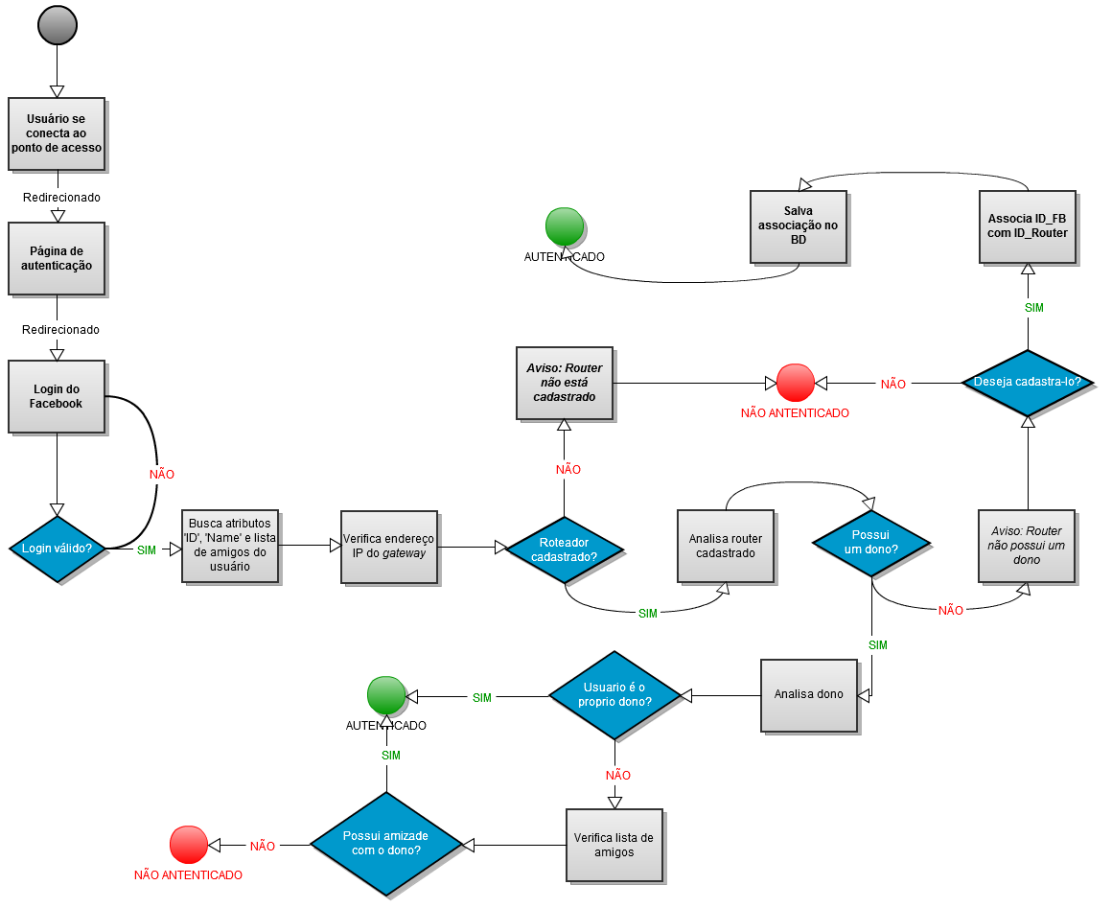

Figura 9 – Funcionamento da autenticação social
Fonte: Autoria Própria

6 APRESENTAÇÃO DO PROTÓTIPO

O desenvolvimento deste trabalho levou a um protótipo cujas telas principais estão apresentadas a seguir.

A figura 11 mostra a tela que será exibida ao usuário assim que este tenta utilizar o recurso da internet, através de um navegador WEB. Ao clicar no ícone do Facebook®, o usuário é redirecionado a página de autenticação do mesmo.

Figura 10 – Página inicial do protótipo
Fonte: Autoria Própria

A seguir, na figura 12, para autenticar-se, o usuário deve informar suas credenciais para realizar uma tentativa de acesso à rede.

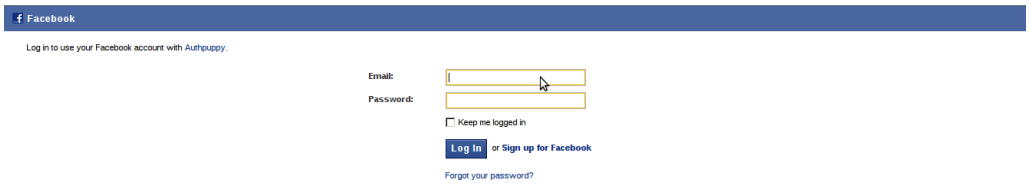

Figura 11 – Autenticação pelo Facebook
Fonte: Autoria Própria

Caso o usuário tenha permissão de acesso, ou seja, tenha um relacionamento no Facebook® com o proprietário do roteador, o acesso é garantido a ele. A página exibe a imagem de perfil do Facebook® e o botão “Navegar” para o processo final de autenticação, como mostra a figura 13.

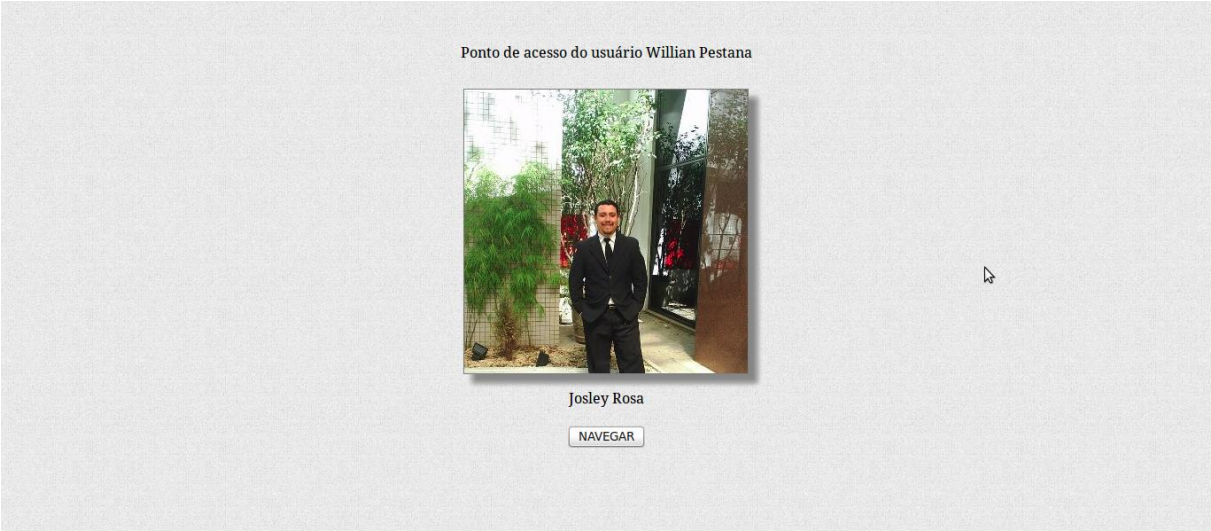

Figura 12 – Usuário com permissão de acesso
Fonte: Autoria Própria

Caso contrário, será exibida uma mensagem informando que não é permitido o acesso.

Figura 13 – Usuário sem permissão de acesso
Fonte: Autoria Própria

6 CONCLUSÕES E TRABALHOS FUTUROS

A ideia proposta atingiu seus objetivos. O sistema resultante deste trabalho consegue realizar autenticação de usuários em uma rede tendo como base as suas credenciais do Facebook® e evitando a autenticação por chaves compartilhadas, o sistema possibilita que apenas amigos do proprietário da rede possam ter acesso a ela. Outra característica resultante é que o usuário não precisa de conhecimento técnico em portais cativos.

Durante o processo de integração com o Facebook®, foi constatada a necessidade de permitir acesso para autenticação do usuário informando o endereço IP do mesmo ao Wifidog, porém foi visto que esse IP era alterado constantemente. Esse empecilho foi resolvido adicionando ao Wifidog uma lista contendo todos os endereços que o Facebook® pode assumir. Outro problema com a integração com o Facebook® foi a necessidade de informar um domínio ao qual o acesso da API seria liberado, devido ao fato deste projeto ser executado localmente, isso não foi possível. O problema foi sanado criando um domínio local definido no próprio roteador. Mais um problema encontrado foi o fato de o Facebook® usar um servidor alheio para armazenar imagens e arquivos CSS (*Cascading Style Sheets*) e JS (*JavaScript*). Este servidor não possui um endereço fixo e não foi possível encontrar quais endereços abrangem este servidor. Devido a isto, a página de autenticação demorava cerca de 2 minutos para carregar. Sabendo que este servidor usa o protocolo HTTPS, a solução foi liberar a porta 443 para o tráfego de dados, permitindo assim o carregamento normal da página de autenticação.

Durante o processo de construção do projeto, alguns problemas foram resolvidos com a ajuda da comunidade do Authpuppy, pode-se constatar o quão ativa esta comunidade. Alguns contatos feitos com alguns membros dessa comunidade estão transcritos e disponibilizados no Apêndice I.

Potenciais melhorias podem ser descritas para esse projeto. Existe a possibilidade de inserir o projeto em uma rede em malha, de forma a prover acesso à internet através de comunicação por múltiplos saltos através do uso do protocolo OLSR; uso de certificações digitais para associar um AP ao seu proprietário; estender o suporte do sistema a outras redes sociais; oferecer ao usuário possibilidade de gerenciamento de seus recursos.

O sistema apresentado nesse trabalho é um passo primordial no que diz respeito à autenticação social por portais cativos, é possível afirmar que a possibilidade de incrementações às funcionalidades do sistema proposto é muito grande, pode-se utilizar a API de uma rede social para buscar diversas informações públicas e inseri-las de alguma maneira no contexto de portais cativos, além do mais, o suporte para *plug-ins* oferecido pelo portal cativo AuthPuppy, permite que essas melhorias possam ser inseridas com muita praticidade.

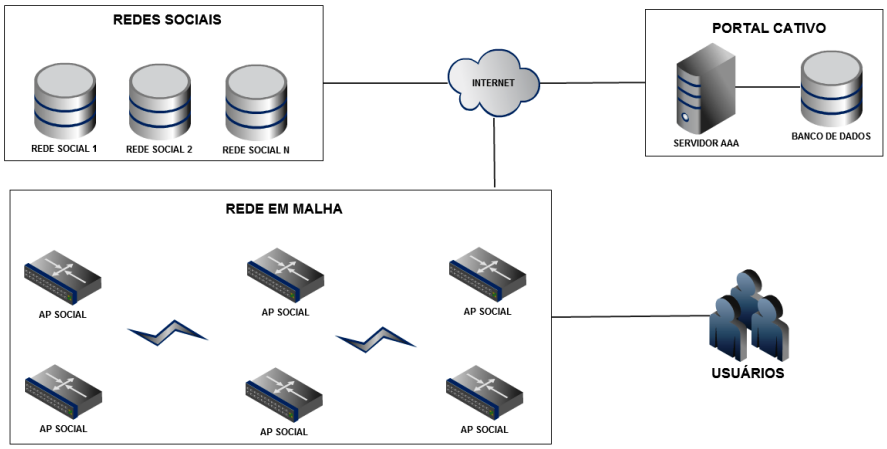

Figura 14 – Arquitetura do projeto com potenciais melhorias
Fonte: Autoria própria

7 REFERÊNCIAS

AGUIAR, Rui; ANTONIADIS, Panayotis; SATSIOU, Anna. Community Building over Neighborhood Wireless Mesh Networks. IEEE Technology and Society, Special Issue on Potentials and Limits of Cooperation in Wireless Communications, Março. 2008.

ALBUQUERQUE, Célio V N; CARRANO, Ricardo C; GOMES, Arthur G; MAGALHÃES, Luiz C S; SAADE, Débora C M; TAROUCO, Liliane R. Multihop MAC: Desvendando o Padrão 802.11s. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, Rio de Janeiro, 2008.

ALMEIDA, Jussara; BENEVENUTO, Fabrício; SILVA, Altigran S. Explorando Redes Sociais Online: Da Coleta e Análise de Grandes Bases de Dados às Aplicações. In: XXIX SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES E SISTEMAS DISTRIBUÍDOS, Campo Grande, 2011.

ALVES, Luiz F S. Um aplicativo baseado em inteligência coletiva para compartilhamento de rotas em redes sociais. Graduação em Tecnologia em Sistemas para Internet - UTFPR, Campo Mourão, 2011, 43p.

ARRUDA, Fernanda W. Estudo da Tecnologia, do Desenvolvimento e da Utilização das Redes Mesh. Graduação em Engenharia de Telecomunicações – FURB, Blumenau, 2010,50p.

BARANAUSKAS, Maria C; MELO-SOLARTE, Diego S; MIRANDA; Leonardo C; NERIS, Vânia P A; SANTANA, Vagner F. Redes Sociais Online: Desafios e Possibilidades para o Contexto Brasileiro. In: XXIX CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (CSBC), Porto Alegre, 2009. p.339-353.

BOYD, Danah; ELLISSON, Nicole. Social Network Sites: Definition, History and Scholarship.Journal of Computer-Mediated Communication, 2008.

CHANG, Sang-Yoon; CHOI, Jihyuk; HU, Yih-Chun; KO, Diko. Secure MAC-Layer Protocol for Captive Portals in Wireless Hotspots. In: IEEE INTERNACIONAL CONFERENCE, 978-1-61284-233-2, 2011, Kioto. Anais. Urbana/EUA: Universidade de Illinois, 2011. p.1 – 5.

COSTA, Rogério D. Por um novo conceito de comunidade: redes sociais, comunidades pessoais, inteligência coletiva. Interface - Comunicação, Saúde, Educação, v. 9, 2005, p.235-248.

ALBUQUERQUE, Célio V N; DA FONSECA, José E M; LEITE, Julius; MAGALHÃES, Luiz C S; NEVES, Luciana E; SAADE, Débora C M; TEIXEIRA, Douglas V. RT1 - Termo de referência e estado da arte. GT-Mesh, Rio de Janeiro, 2008. p. 2-79.

FLICKENGER, Rob. Redes sem fio no Mundo em Desenvolvimento: Um guia prático para o planejamento e a construção de uma infra-estrutura de telecomunicações – 2006/2008. 411p.

GONÇALVES, Paulo A da S; LINHARES, André G. Uma Análise dos Mecanismos de Segurança de Redes IEEE 802.11: WEP, WPA, WPA2 e IEEE 802.11w. Pós-graduação em Segurança de Redes de Computadores – UFPE, Recife, 2006, 17p.

CALAFATE, Carlos T; CANO, Juan-Carlos. HORTELANO, Jorge; MANZONI, Pietro. A Wireless Mesh Network-based System for Hotspots Deployment and Management. In: THIRD INTERNATIONAL CONFERENCE ON NETWORKING AND SERVICES, 0-7695-2858-9, 2007, Athenas. Valencia Polytechnic University of Valencia, 2007.

IEEE Doc. IEEE P802.11-96/49C. "802.11 Tutorial – 802.11 MAC Entity: MAC Basic Access Mechanism Privacy and Access Control". U.S.A., 2006.

IEEE.Std 802.11-2012 (Revision of IEEE Std 802.11-2007). IEEE Standard for Information technology-- Part 11: Wireless LANMAC and PHY Specifications Amendment 5: Enhancements for Higher Throughput. Março . 2012. p.1-2793.

KOTH-ARSA, Kasom; PHONPHOEM, Anan; SANGUANPONG, Surasak.Architectural design for large-scale campus-wide captive portal. In: 43rd ANNUAL INTERNATIONAL CARHNAGAN CONFERENCE: 2009, Tailândia. p.72 – 76

KROPF, Peter; MONAKHOV, Alexey; SHILLER, Eryk. Shibboleth based Authentication, Authorization, Accounting and Auditing in Wireless Mesh Networks. In: 11th IEEE INTERNATIONAL WORKSHOP ON WIRELESS LOCAL NETWORKS, 978-1-61284-928-7, 2011, Bonn. Nechatel: UniversityofNeuchatel, 2011. p.918 – 926.

LEAL, Arthur D; REIS, Mário H. Simulação de Protocolos de Roteamento em uma RedeMesh Interurbana. Engenharia da Computação em Revista, Nazaré, 2009.

TANENBAUM, Andrew S. Redes de Computadores. 4. Ed. Rio de Janeiro: Campus, 2003. 968p.

APÊNDICE A – INSTALAÇÃO DO OPENWRT

Requerimentos:

- *Firmware* TP-LINK (2011/2012)
- *Firmware* TP-LINK (2010)
- *Firmware*OPENWRT (2012)

Caso esteja usando Windows, será necessário também este dois softwares abaixo:

- WinSCP
- Putty

(Caso o OPENWRT ainda não tenha sido instalado no roteador, pule para o 4º passo.).

1º passo**:** É necessário voltar ao *firmware* padrão do roteador, portanto primeiro é necessário fazer o download do firmware no site oficial.

A partir de agora, são necessários os seguintes dados:

1. IP do roteador
2. *User_name* (normalmente 'root')
3. *Password* (definido pela interface web do OPENWRT)

2º passo**:** Agora transfira o *firmware* para roteador usando protocolo SCP.

Linux:

scpfirmware.bin<usuário>@<ip_do_computador><path_do_firmware><ip_do_router>:/tmp

Windows:

No WinSCP, digite o IP do roteador, usuário e a senha, e em *File Protocol* selecione SCP. Clique em *Login*. Ignore os avisos. Com a interface carregada, arraste o *firmware* para dentro da pasta tmp do roteador. Em opções, escolha transferência "*binary*". Clique em OK e espere a transferência terminar.

3º passo: Para instalar o *firmware stock* é necessário usar o protocolo de rede SSH.

1. Abra o Putty, digite o IP do roteador, selecione SSH e clique em *Open*.
2. Digite o *User_name* e o *Password*.
3. Digite ls e tecle <ENTER>, caso nada apareça, digite cd ..
4. Digite mtd -r write /tmp/<nome_do_firmware>.bin firmware
5. Espere a instalação e o *reboot* do roteador terminar. Feche o Putty.
6. Vá a um *browser* qualquer, digite o IP do roteador e verifique se aparece a interface do TP-LINK.

4º passo: Agora é necessário instalar o *firmware* TP-LINK (2010). Na interface *web* entre em *System tools e Firmware update.* Clique em *browse, selecione* o *firmware* e inicie a instalação. Espere a instalação e o reinicialização do roteador terminar.

5º passo: Repita o 4º passo, mas agora selecionando o *firmware* OPENWRT.

A partir de 2011, o firmware TP-LINK veio com um novo *bootloader* que é incompatível com o OPENWRT, fazendo com que a porta WAN seja desabilitada durante o boot do roteador. A solução foi substituir o firmware 2011/2012 por um firmware 2010, e então instalar o OPENWRT.

APÊNDICE B – INSTALAÇÃO DO AUTHPUPPY

A instalação foi feita sob o sistema Linux Mint 13, ambiente MATE 1.2. Os softwares requeridos e seus respectivos processos de instalação serão descritos a seguir.

O primeiro passo é a instalação de um servidor HTTP, como o Apache. O Mint sendo um sistema baseado na plataforma Debian é possível usar o seguinte comando no terminal para executar o download e a instalação automática do pacote Apache:

apt-getinstall apache2

É estritamente necessário ativar o módulo *mod_rewrite* no Apache. Este módulo reescreve a URL em nível de servidor a tornando mais “amigável” e mais seguro para o servidor. Por exemplo:

URL requisitada ao servidor:

http://www.sitequalquer.com/principal/login/login.php?user=eu&passwd=26189

URL retornada ao usuário:

http://www.umsitequalquer.com/principal/login

Sem este módulo ativado, o apache não irá encontrar a página de login do Authpuppy e, consequentemente, o autenticador não irá funcionar.

Para ativar tal módulo, escreva o seguinte comando no terminal do sistema:

a2enmodrewrite

O banco e a interface web para a configuração do mesmo podem ser instalados com os seguintes comandos:

apt-get install PostgreSQL
apt-get install phppgadmin

O banco de dados que será usado pelo autenticador deverá ser criado pelo usuário. Usando a interface web do *phppgadmin* isto é facilmente alcançável. No entanto, devido a segurança desta interface, é impossível fazer o login ao *PostgreSQL*. Portanto devemos fazer algumas pequenas modificações para isto ser possível.

O arquivo *pg_hba.conf*, localizado no pasta *'main'* dentro do diretório do postgresql, deverá ser modificado. No final do arquivo, todas as linhas que possuem no parâmetro *method*os valores MD5, *peer* ou *ident* devem ser comentadas usando o caractere sustenido (#). Após estas linhas serem comentadas, a linha abaixo deverá ser adicionada no final do arquivo:

local sameuser postgres trust

Salve e feche o arquivo. Esta linha torna possível a modificação do usuário padrão *postgres* em um ambiente local. Para tal modificação ser aplicada é necessário reiniciar o PostgreSQL com o seguinte comando no terminal:

servicePostgreSQLrestart

Agora usando o terminal do sistema, definiremos uma senha para este usuário *postgres*. Isso só será possível após a instalação do pacote *pgsql*usando o seguinte comando:

apt-getinstall php5-pgsql

Após a instalação do pacote, será usado um comando para se conectar ao usuário *postgres* e outro para alterar sua senha:

psql -U postgres -d postgres
ALTER USER PostgreSQL WITH ENCRYPTED PASSWORD “NOVASENHA”;

Também é necessário modificar a linha “*Extra_Login_Security*” no arquivo *config.inc.php* localizado no diretório */etc/phppgadmin* de *true*para *false*. Com esta mudança, contas como *root*, *postgres*, *administrator*, *pgsql* são aceitas pelo *phppgadmin*. Portanto, é recomendado que, após a criação de uma nova conta no *phppgadmin*, seja atribuído o valor *true* a esta linha novamente.

Voltando ao arquivo *pg_hba.conf*, devemos modificar o parâmetro *trust* da linha adicionada anteriormente para *md5* e adicionada outra linha no final arquivo, ficando desta forma:

local sameuser postgres md5
host all all 127.0.0.1 255.255.255.255 trust

O PostgreSQL deve ser reiniciado novamente. Outras bibliotecas importantes também devem ser instaladas:

apt-get install php5 php5-curl

O primeiro passo para instalação do Authpuppy é baixar o pacote do site oficial. O arquivo deve ser descompactado e a pasta Authpuppy deve ser movida para o diretório */var/www* usando o comando *mv* pelo terminal do sistema.

Link para download: https://launchpad.net/authpuppy/+download

Depois de movido devemos iniciar a instalação. Inicie um navegador qualquer e digite na barra de endereços:

http://localhost/authpuppy/web/preinstall.php

Importante: caso aconteça algum problema após a instalação e seja necessário fazer a reinstalação do Authpuppy novamente, será preciso apagar o arquivo installed.txt localizado em /var/www/authpuppy/web. A existência desse arquivo define que a instalação já foi concluída e, dessa forma, não será mais possível acessar a página de instalação.

A primeira página contém somente algumas informações e não é necessária a descrição desta aqui neste relatório. A segunda página de instalação contém a descrição de vários requisitos que devem ser alcançados antes de continuar, desde a instalação de pacotes necessários, modificação de arquivos, até permissões de pastas. Cada permissão necessária deve ser feita usando o comando *chmod* pelo terminal do sistema.

chmod 777 /config/authpuppy.yml
chmod 777 /config/databases.yml
chmod777 cache
chmod777 web
chmod777 log
chmod777 data

Após alcançar todos os requerimentos, na terceira página deverão ser informados dados de acesso ao banco anteriormente criado no *phppgadmin*. E, finalmente, na quarta página, contém a criação da conta administrador do Authpuppy.

Existe uma página de debug no Authpuppy, qualquer erro que aconteça durante a interação com a interface será descrito em detalhe ao administrador. Para tal acesse: *localhost//frontend_dev.php.*

APÊNDICE C – FRAMEWORK PHP SYMFONY

Um framework torna o desenvolvimento mais fácil e rápido, uma vez que transforma um pacote de complexas operações em simples funções, acrescenta estrutura ao código, fazendo com que o desenvolvedor escreva um código mais simples e legível.

O Symfony é um framework completo projetado para aperfeiçoar o desenvolvimento de *websites*, através de várias características. Uma destas características é o uso de padrões de projeto, separando o desenvolvimento entre modelo, visão e controle. Contém diversas ferramentas e classes que visam reduzir o tempo de desenvolvimento de uma complexa aplicação web. Além disso, este framework automatiza tarefas comuns, para que o desenvolvedor possa se concentrar inteiramente no principal objetivo da aplicação que está sendo desenvolvida. O reuso de código é o resultado final das vantagens deste framework, fazendo com que o desenvolvedor não precise se preocupar com a programação de módulos banais, que é o que torna o desenvolvimento lento e ineficiente.

Este framework foi escrito totalmente em PHP5 e foi testado em vários projetos de alta demanda em sites de negócio. É compatível com a maioria das bases de dados disponíveis, incluindo o Mysql, Postgresql, Oracle e Microsoft SQL Server. É compatível com as plataformas Windows e Unix.

Um projeto Symfony é formado por aplicações, cada aplicação é dividida em módulos. O Symfony é capaz de gerar automaticamente um módulo de um dado modelo com os recursos básicos de manipulação. Cada módulo possui um arquivo PHP chamado *action*, onde, como o próprio nome diz, são definidas as ações possíveis para um respectivo módulo. Nestas classes, cada ação deve ser sempre possuir o sufixo *execute*, fazendo com que todas as ações sejam padronizadas.

Em cada aplicação existe um arquivo YAML que faz o controle de todas as rotas de todos os módulos. Chamados de routing, neste as rotas são criadas usando uma sintaxe simples, onde é necessário somente definir o classe da rota, endereço que será usada, módulo usado e sua respectiva ação.

Originalmente este framework foi criado para desenvolver aplicações web complexas, com lógica de negócio pesadas, cenário onde o uso do PHP puro é algo quase inviável. No entanto, independentemente se o projeto é simples ou não, este framework traz a vantagem de um ambiente controlado, dentro de padrões de projeto, tornando o desenvolvimento eficaz e o código confiável.

APÊNDICE D – DESCRIÇÃO DAS TABELAS

A seguir está descrita a função de cada atributo criado para as tabelas

Tabela usuário:

- *ID:* atributo do tipo inteiro, primário, com valor único, não nulo, que terá o propósito de identificar cada usuário dentro do banco de dados.
- Status: atributos do tipo inteiro, não nulo, com o propósito de definir se este usuário em questão está ativo ou não. Pode possuir somente dois valores inteiros, 0 e 1.

O atributo ID da tabela usuário agirá como chave estrangeira nesta tabela.

Tabela facebook

- *ID:* atributo do tipo inteiro, primário, com valor único, não nulo, que terá o propósito de identificar cada perfil dentro do banco de dados.
- ID_Perfil*:* atributo do tipo texto, com valor único, não nulo, com o propósito de armazenar o número de identificação de cada perfil proveniente do Facebook.

O atributo ID da tabela usuário agirá como chave estrangeira nesta tabela.

Tabela roteador

- *ID:* atributo do tipo inteiro, primário, com valor único, não nulo, que terá o propósito de identificar cada roteador dentro do banco de dados.
- IP_roteador: atributo do tipo texto, com valor único, não nulo, com o propósito de armazenar o IP proveniente de cada roteador conectado à rede.

O atributo ID da tabela usuário agirá como chave estrangeira nesta tabela.

Tabela usuário_roteador

- *ID:* atributo do tipo inteiro, primário, com valor único, não nulo, que terá o propósito de identificar cada relacionamento entre usuários e roteadores dentro do banco de dados.

- Os atributos ID da tabela usuário e ID da tabela roteador agirão como chaves estrangeiras nesta tabela.

APÊNDICE E – MAPEAMENTO DE OBJETOS

Para fazer o mapeamento dos objetos que o framework irá usar, é necessário transcrever toda a modelagem do seu banco de dados para um arquivo que o *Doctrine*[18]seja capaz de compreender. Esse arquivo é chamado de *schema* e é feito usando formato YAML. Nesse arquivo é definido as tabelas, suas relações e seus atributos, e deve ser colocado no diretório /config/doctrine.

O *doctrine* é um software que se baseia na indentação do arquivo, portanto o *schema* deve ser organizado com base na hierarquia de cada linha. Contudo não deve possuir nenhuma tabulação, porque esta não é identificada de corretamente pelo software. O padrão estabelecido para indentação é usar dois espaços para diferenciar da hierarquia anterior.

Hierarquia1
>>Hierarquia2
>>>>Hierarquia3

Exemplo:
nome_da_relação:
tableName: nome_da_tabela
columns:
name:
type: string(50)
age:
type: integer
status:
type: integer,
default: 0

Talvez você tenha notado que o atributo primário não foi definido no exemplo, isso porque a criação automática deste atributo é uma convenção do ORM *Doctrine*, mas isso não o impede de definir este atributo no arquivo.

APÊNDICE F – CONSTRUÇÃO DE UM *PLUG-IN*

Para gerar o *plug-in* e suas classes básicas é necessário seguir alguns passos simples. Antes de tudo, é necessário instalar um *plug-in* chamado *sfTaskExtra* através do seguinte comando:

(Todos os comandos devem ser executados na raiz do projeto)

phpSymfonyplug-in:installsfTaskExtraPlug-in

Este *plug-in* torna mais fácil a criação de *plug-ins* e a geração de seus módulos. Depois de instalado é possível criar um *plug-in* com o seguinte comando:

phpSymfonygenerate:plug-in<nome_qualquer>

O *plug-in* será gerado dentro da pasta *plug-ins*, no entanto não possui praticamente nenhuma classe usável. Para tal primeiro precisamos criar um módulo, que armazenará as ações/funções e templates do projeto. Isso é possível com o seguinte comando:

phpSymfonygenerate:plug-in-module<nome_do_plug-in><nome_qualquer>

Após o módulo ser gerado, devemos gerar as classes básicas do *form*, *model* e *filter*. Isso é possível se o arquivo *schema.ym*l existir, que não é o caso no momento, portanto crie este arquivo antes de continuar. (Leia o documento '**Mapeamento de Objetos**' para saber do que se trata e como criar um para seu projeto). Além disso, é necessário habilitar o *plug-in* no arquivo ProjectConfiguration.class, localizado dentro da pasta *config* do projeto. Adicione a seguinte linha no final da função *setup()*:

$this->enablePlug-ins('<nome_do_plug-in>');

Com o schema.yml pronto e com o *plug-in* habilitado, faça um backup da pasta *lib* do projeto antes de continuar copiando a pasta para outro diretório, ou usando o *Bazaar*. Isso é necessário porque o próximo comando modifica arquivos que não devem ser modificados, o que resulta em um mau funcionamento do projeto. Após isso, execute o comando abaixo:

./Symfonydoctrine:build --all-classes

Agora vá até o diretório /lib/[filter][model][form]/doctrine/<seu_*plug-in*> do projeto e transfira estes arquivos para:

/plug-ins/<seu_*plug-in*>/[filter][model][form]/doctrine/autogen.

(É necessário criar cada pasta *autogen*) Após isso, restaure a pasta *lib* do projeto. A pasta *plug-in* também foi modificada, mantenha a modificada.

Todas as classes bases do *plug-in* estão prontas para uso. Agora a última parte é a atualização do banco de dados, e para fazer isso é necessário criar arquivos chamados de arquivos de migração.

Pelo Authpuppy é possível atualizar o banco de dados pela interface web do administrador, no menu *Manage Plug-ins*. Contudo, para que o *plug-in* recém-criado seja visível na interface web é necessário adicionar um função abaixo no arquivo *Plug-inConfiguration.class*, localizado na pasta *config* do *plug-in*:

```
public static function isAuthPuppyPlug-in() {
returntrue;
}
```

Navegue até a opção *manage plug-ins* e note que o *plug-in* agora aparece na lista de *plug-ins* instalados no AuthPuppy. Se o(s) arquivo(s) de migração foi criado corretamente, deverá aparecer a opção de atualização (upgrade) do banco de dados. Clicando neste link, o banco será atualizado com as novas tabelas.

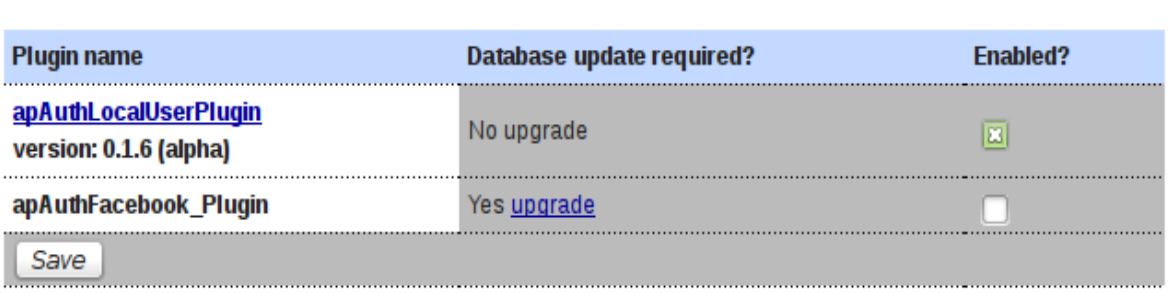

Figura 15 – *Plug-ins*
Fonte: Interface do Authpuppy

APÊNDICE G – ROTEAMENTO

Rota	*melody_connect*	Função que faz o redirecionamento para a página de *login* do Facebook usando como argumento a chave de acesso localizada no arquivo app.yml. Após o login, executa o call-back para facebook_data.
Módulo	*melody*	
Ação	*executeConnect*	
Template	-	

Rota	*auth_provided*	Redireciona para a página de autenticação final. Com os argumentos provenientes do WifiDog executa a autenticação.
Módulo	*melody*	
Ação	*executePostAuth*	
Template	*postAuthSuccess*	

Rota	*form_mac*	Função que executa a criação de um objeto formulário para cadastro de um roteador. Somente acessado pela administração.
Módulo	*melody*	
Ação	*executeFormMac*	
Template	*formMacSuccess*	

Rota	*set_mac*	Função usada para cadastro de um roteador. *Após a execução, é feito o* redirecionamento para a rota *form_mac*. Somente acessado pela administração.
Módulo	*melody*	
Ação	*executePushMac*	
Template	-	

Rota	*get_mac*	Função que verifica se o roteador pelo qual o usuário está conectado existe no banco de dados. Se existir, é redirecionado para *get_owner*, caso contrário é redirecionado para *no_such_mac*.
Módulo	*melody*	
Ação	*executeGetMac*	
Template	-	

Rota	*facebook_data*	Função que recupera os dados do usuário logado pelo Facebook, tais como *ID*, *nome* e lista de amigos. Redirecionado para get_mac.
Módulo	*melody*	
Ação	*executeFacebook*	
Template	-	

Rota	*no_such_mac*	Função que redireciona o usuário para uma página que informa que o
Módulo	*melody*	

Ação	*executeNoSuchMac*	roteador que está sendo usado não existe.
Template	*noSuchMacSuccess*	

Rota	*get_owner*	Função que verifica se o roteador em uso possui um dono. Redirecionado para owner_who ou no_owner.
Módulo	*melody*	
Ação	*executeGetOwner*	
Template	-	

Rota	*no_owner*	Redireciona para uma página onde pergunta se o usuário deseja cadastrar o roteador em seu nome. Redirecionado para get_user (sim)
Módulo	*melody*	
Ação	*executeNoOwner*	
Template	*noOwnerSuccess*	

Rota	*get_user*	Verifica se o usuário logado está ou não cadastrado. Redirecionado para set_user ou set_owner.
Módulo	*melody*	
Ação	*executeGetUser*	
Template	-	

Rota	*set_user*	Cadastra o ID do Facebook do usuário no banco de dados
Módulo	*melody*	
Ação	*executeSetUser*	
Template	-	

Rota	*set_owner*	Associa uma conta do Facebook a um roteador usando as chaves primárias do usuário e do roteador previamente cadastrados. Redirecionado para get_owner.
Módulo	*melody*	
Ação	*executeSetOwner*	
Template	-	

Rota	*owner_who*	Verifica quem é o dono do roteador. Caso seja o próprio usuário é redirecionado para auth_provided, caso contrário é redirecionado para is_friend.
Módulo	*melody*	
Ação	*executeOwnerWho*	
Template	-	

Rota	*owner_who*	Verifica quem é o dono do roteador. Caso seja o próprio usuário é redirecionado para auth_provided,
Módulo	*melody*	
Ação	*executeOwnerWho*	

Template	-	caso contrário , para is_friend.

Rota	*is_friend*	Verifica se o dono do roteador faz parte da lista de amigos do usuário logado. Caso sim é redirecionado para auth_provided, caso contrário, para not_friend
Módulo	*melody*	
Ação	*executeIsFriend*	
Template	-	

Rota	*not_friend*	Avisa que o usuário não possui amizade com o dono do roteador e não tem permissão de acesso.
Módulo	*melody*	
Ação	*executeNotFriend*	
Template	*notFriendSuccess*	

Rota	*login*	Página principal de login
Módulo	*node*	
Ação	*executeLogin*	
Template	*loginSuccess*	

APÊNDICE H – DESCRIÇÃO DAS VARIÁVEIS

executeFacebook

- me: recebe dados do usuário que se autenticou no Facebook
- $_SESSION['friends']: recebe a lista de amigos do usuário que logo no Facebook
- $_SESSION['id_facebook']: recebe o ID do Facebook do usuário está autenticado

executeFormac

- form: objeto do tipo *Plug-in*routersForm

executeGetMac

- arp: recebe o resultado do comando arp
- userIp: recebe o IP do gateway
- routerTable: recebe a tabela routers
- ipAddr: recebe o resultado da SQL

executeSetMac

- form: objeto do tipo *Plug-in*routersForm
- request: recebe o request do form
- params: recebe o MAC digitado pelo administrador no form

executeGetOwner

- routerId: recebe a chave primária ID do roteador cadastrado no banco
- relTable: recebe a tabela fbUserRouter
- owner: recebe o resultado da SQL que verifica se o roteador possui um dono
- $_SESSION['id_owner']: recebe o ID do dono do roteador (chave primária da tabela fbUsers)
- $_SESSION['hash']: recebe uma hash md5

executeGetUser

- relTable: recebe a tabela fbUser
- user: recebe o resultado da SQL que busca um usuário tendo como parâmetro o ID do Facebook
- $_SESSION['id_user']: recebe a chave primária ID do usuário encontrado

executeSetUser

- form: recebe a tabela *Plug-in*fbUserForm
- params: recebe o ID do Facebook do usuário

executeSetOwner

- form: recebe a tabela *Plug-in*fbUserRouterForm

- params: recebe a chave primária ID do roteador e a chave primária ID do usuário

executeOwnerWho

- id_owner: recebe o ID do dono do roteador (não do facebook)
- id_facebook_login: recebe o ID do facebook do usuário autenticado
- relTable: recebe a tabela fbUser
- user: recebe o resultado da SQL que busca por um usuário

executeIsFriend

- $_SESSION['friends']: session que possui a lista de amigos do usuário autenticado

APÊNDICE I – CONTATOS COM A COMUNIDADE DO AUTHPUPPY

Devido ao problema de liberação de acesso livre a rede social, para que fosse possível usá-la para autenticação no portal social, foi visto a necessidade de buscar apoio da comunidade do Authpuppy. Abaixo está uma transcrição dos contatos realizados, e estes podem ser acessados nos endereços descritos abaixo:

Contato 1: *https://answers.launchpad.net/authpuppy/+question/219083*

Contato2: *https://answers.launchpad.net/authpuppy/+question/223686*

Contato 1:

AuthPuppy authentication server for Wifidog networks

Allow access without authentication

Asked by Notheros on 2013-01-13

Hello!

I have a question. Please, imagine the following scenario:

I have a network with WifiDog (OpenWRT) and AuthPuppy. The Person X is connected to my access point, but he is not authenticated to it. He want to access, for example, his Gmail, however he doesn't has as account on Authpuppy.

There's a way to allow access to a specify website without an actually authentication?

Question information

Language: English

Status: Solved

For: AuthPuppy

Assignee: No assignee

Solved by: Robin Jones

Solved: 2013-02-13

Last query: 2013-02-13

Last reply: 2013-01-19

Robin Jones (robin-networkfusion) said on 2013-01-14: #1

It’s called a walled garden.

As far as I know the only way to do this is in the wifidogconfig file, however it will only let you enter IP addresses, so you will need to know what IP the host resolves to and hope they don't change it. However I don't know where you will stand with DNS resolution?!

Notheros (Notherospestana-l) said on 2013-01-14: #2

Thank you, Robin!

I was thinking about this solution just a few minutes ago. That will solve my problem, but I don't know exactly how to do that.

Anyway, I will try to make it work.

gbastien (gbastien02) said on 2013-01-16: #3

Hello,

That reminds me. I made a *plug-in* for a client that was exactly that: a server-side walled garden (proxy). I gave him a few months head start working with it, but now is about time I publish the code, hopefully this week.

Notheros (Notherospestana-l) said on 2013-01-16: #4

Interesting. I'm looking forward to try it.

gbastien (gbastien02) said on 2013-01-19: #5

Hi, I just released the *plug-in*. You'll need to pull it from bazaar for now, it is not yet downloaded directly from authpuppy:

https://code.launchpad.net/~alliancecsf-dev/authpuppy/apProxy*Plug-in*

Notheros (Notherospestana-l) said on 2013-01-19: #6

Thanks gbastien! I already downloaded it. I will take a look on it later. This *plug-in* will, definitely, be useful to many people.

Notheros (Notherospestana-l) said on 2013-02-13: #7

Thanks Robin Jones, that solved my question.

Contato 2:

Walled garden - Slow loading

Asked by Notheros on 2013-03-07

Hello guys,

Earlier, I asked a question about how allow access to a website without authentication and it worked, thank you very much!

However, try to access this website (in this case, Facebook) is a patience exercise, slow and sometimes is really slow. Not to mention that it never loads properly, I mean, only text appears, without CSS whatsoever.

On the server it's slow, about 15seg. But with another computer, it takes about 2min to load the page, and sometimes even more.

I've no idea why this is happening. Can someone give me a hint?

Thanks!

Question information

Language:	English
Status:	Open
For:	AuthPuppy
Assignee:	No assignee
Last query:	2013-03-12
Last reply:	2013-03-07

gbastien (gbastien02) said on 2013-03-07: #1

Hi,

Are you using the Authpuppy proxy *plug-in*?

About the CSS now showing properly, you probably need to add the server to the list of white listed sites. Use firebug while querying the web page (normally, without AuthPuppy). In the Net tab, you'll see all the queries that are made to fetch the page. You need to make sure all requested urls are whitelisted.

As for why it is slow, I don't know. If I recall, it was ok when I used it. Try to see what takes so long. Is it the time on the server? The request to Facebook from the

AuthPuppy server? Maybe a circular request, if the Facebook page requests something that is not in the walled garden, it will be redirected to the login page, which might be Facebook, who will again request...? Try solving the css problem first and see if the issue is still there then.

Maybe you could try adding some error_log in the *plug-in*'s php code and see what might take so much time. Debugging with http://<your server>/frontend_dev.php could also give you some hints.

Notheros (willianpestana-l) said on 2013-03-08: #2

apProxy plug-in? No, I tried, but I couldn't "turn it on". It should have appeared on the *Plug-in* management menu, right? But it didn't.

I did what you said, and, indeed, there's a server which I wasn't aware of. However, this domain doesn't have a static IP. Facebook doesn't have either, but I found a list with all possibles IP's, and that's why it's working now.

I think you might be right about it. Probably if I add this other server to the walled garden, it will work flawlessly.

Thank you very much, gbastien! I will be back here to give you a feedback. I just need to find out the IP range of this other server. (however, I don't know how to do that)

gbastien (gbastien02) said on 2013-03-08: #3

The proxy *plug-in* does not work? Weird. Are you sure you put it at the right path, that <authpuppy>/plug-ins/apProxy*Plug-in*/ contains directly the *plug-in* files, amongh which ainfo.yml file used to get the *plug-in*'s information?

But the proxy server would be the way to go. As you found out, a domain with many possible IP is quite an hassle to configure in the walled garden.

Notheros (willianpestana-l) said on 2013-03-09: #4

Yes, exactly. I also took a look on the apProxy*Plug-in*Configuration.class file to check if the isAuthPuppy*Plug-in*() function was correct. Perhaps, I'm forgetting something. It need any specific configuration to work?

gbastien (gbastien02) said on 2013-03-09: #5

Did you clear the cache?

Notheros (willianpestana-l) said on 2013-03-10: #6

Yes, many times. I will try again tomorrow.

Notheros (willianpestana-l) said on 2013-03-12: #7

I got the plug-in to work. I forgot to give permission to the folder. But I think I didn't understand how it works. Can you explain to me? How your plug-in is able to redirect to a specify website, if the Wifidog doesn't allowed it without authentication? I took off the Facebook from walled garden, and wrote 'www.facebook.com' on your *plug-in*, but it doesn't work. Maybe I misunderstood it?