FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Bruno Felippe Barbosa

Captura de Dados do Protocolo MySQL com Wireshark

Americana, SP

2018

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Superior de Tecnologia em Segurança da Informação

Bruno Felippe Barbosa

Captura de Dados do Protocolo MySQL com Wireshark

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Superior de Tecnologia em Segurança da Informação, sob a orientação do Prof. Especialista Edson Roberto Gaseta
Área de concentração: Segurança da Informação

Americana, SP.

2018

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

B195c BARBOSA, Bruno Felippe

Captura de dados do protocolo MySQL com wireshark. / Bruno Felippe Barbosa. – Americana, 2018.

66f.

Monografia (Curso de Tecnologia em Segurança da Informação) - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza

Orientador: Prof. Esp. Edson Roberto Gaseta

1. MySQL 2. Segurança em sistemas de informação I. GASETA, Edson Roberto II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana

CDU: 681.3.07

681.518.5

Faculdade de Tecnologia de Americana

Bruno Felippe Barbosa

Captura de Dados do Protocolo MySQL com Wireshark

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Segurança da Informação pelo Centro Paula Souza – FATEC Faculdade de Tecnologia de Americana.
Área de concentração: Segurança da Informação

Americana, 03 de Dezembro de 2018.

Banca Examinadora:

Edson Roberto Gaseta
Especialista
Faculdade de Tecnologia de Americana

Pedro Domingos Antoniolli
Doutor
Faculdade de Tecnologia de Americana

Maria Elizete Luz Saes
Mestre
Faculdade de Tecnologia de Americana

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus pois sem Ele não estaria onde estou hoje, a meu pai por sempre estar ao meu lado em todos os momentos, aos meus amigos, ao meu orientador por me guiar através desse caminho para que esse projeto fosse realizado e a todos os professores da faculdade com quem eu estive presente ao longo deste tempo.

DEDICATÓRIA

Dedico este trabalho a minha família e aos meus amigos, que sempre estiveram presentes em todos os momentos da minha vida.

RESUMO

Este trabalho apresenta, como objetivo principal, a importância de se estabelecer uma conexão de rede segura entre dois dispositivos baseado em um cenário real. Para tal demonstração foi utilizado uma máquina cliente, uma máquina servidor de banco de dados MySQL e utilização de senhas consideradas fortes por serviços verificadores de força de senha, como *How Secure Is My Password?, Kaspersky Lab Secure Password Check* e *Last Pass*. Para exemplificar esta hipótese, foram utilizados técnicas que são conhecidas como Ataques de Dicionário e *Sniffing*. Os ataques e métodos empregados foram testados em um pequeno laboratório controlado utilizando máquinas virtuais com os sistemas operacionais *Linux* e *Windows*, fazendo o uso das ferramentas *Wireshark*, um analisador de pacotes de rede, *HeidiSQL*, um visualizador gráfico de conexões de banco de dados, e *Hashcat*, um programa utilizado para recuperação e auditoria de senhas. O uso destas ferramentas foi capaz de demonstrar os riscos envolvidos com senhas conhecidas, que muitas vezes podem ser classificadas como fortes e conexões não criptografadas através do protocolo MySQL entre um cliente e um servidor, demonstrando que é possível um terceiro conhecer as credenciais de um banco de dados sem muito esforço. Este texto demonstra também, uma técnica que pode ser utilizada para evitar o comprometimento dos dados trafegados em rede e consequentemente invalidar o ataque proposto, como criptografia da conexão por meio do protocolo SSH, tornando, assim, o ataque em questão inutilizável, evitando-se que um possível atacante possa interceptar e decodificar os dados trafegados entre dois dispositivos que se comunicam através da mesma rede

Palavras Chave: rede de computadores; *wireshark*; *hashcat, mysql*; segurança de rede

ABSTRACT

This paper presents, as the main objective, the importance of establish a secure network connection between two hosts based on a real scenario. In order to demonstrate this, a client machine, a MySQL database server and considered strong passwords by services like How Secure Is My Password?, Kaspersky Lab Secure Password Check e Last Pass were used. To exemplify this hypothesis, technics known as dictionary attacks and sniffing were used. These attacks and methods were tested on a small controlled lab using virtual machines running Windows and Linux systems, using tools like Wireshark, a network packets analyzer, HeidiSQL, a graphical database connection visualizer and Hashcat, a program to recover and audit passwords. The use of these tools were capable of demonstrate the risks of known passwords, that sometimes may be classified as strong and non-encrypted connections of the MySQL protocol between a client and server, showing that a third-party is possible to discover the database credentials without much effort. This paper also shows a technic that can be used to avoid the compromising of the data running on the network and therefore, invalidating the proposed attack, like encrypted connection by SSH protocol, making it unusable, avoiding, that a possible attacker may be able to intercept and decode the data between two hosts that communicate with themselves on the same network

Keywords: *computer network, wireshark, hashcat, mysql, network security*

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

1 INTRODUÇÃO

Segundo uma pesquisa realizada pelos serviços *Hootsuite* e *We Are Social* (2018), existem mais de quatro bilhões de pessoas utilizando a internet ao redor do mundo, subindo 7% ano a ano.

Assim como já apontava Cansian (1997), com esse aumento expressivo, cresce também o número de preocupações em relação à segurança cibernética. Tentativas de invasão por indivíduos mal intencionados vão aumentando a cada dia, fazendo com que novas técnicas de segurança se tornem cada vez mais indispensáveis para qualquer sistema moderno.

> "A proliferação de computadores e redes com um custo mais acessível ampliou o problema de acesso não-autorizado, bem como a possibilidade de manipulação e ataque a dados e recursos. O crescimento da interconexão de redes não só possibilita o rápido acesso a uma maior variedade de computadores, como também garante acesso a dados que podem vir de qualquer lugar. Com frequência ocorrem situações onde os intrusos facilmente superam os mecanismos de autenticação de senhas, projetados para proteger os sistemas." (CANSIAN, 1997, p.1)

Ainda segundo o mesmo autor, as Intranets[1] de empresas, corporações ou instituições vem aumentando também. Assim, qualquer uso de sistema informatizado sem as precauções recomendadas e que esteja em rede com outros sistemas, pode se tornar vulnerável a ataques. Alguns casos exemplificados pelo autor remontam ao ano de 1994, quando as redes da General Eletric, IBM e NBC, sofreram ataques mal intencionados e consequentemente tiveram seus sistemas desabilitados por um determinado tempo.

Nesse sentido, Santos e Giavaroto (2013) ressaltam que, levando-se em conta as novas tendências e táticas de invasões, os administradores de rede devem ter em mente que indivíduos mal intencionados estão em evolução. Existem diversas técnicas de invasão atualmente e ter profissionais capacitados, treinados e preparados para lidar com um possível incidente que envolva a segurança da informação de uma organização é de fundamental importância para manter o ambiente seguro e consequentemente proteger os dados que o mesmo contém

[1] Espaço que é restrito a somente um determinado público, usado para compartilhamento de informações restritas e geralmente usado em servidores dentro de empresa (MULLER, 2011)

Assim sendo, o objetivo deste trabalho é apresentar quais são os perigos de se realizar uma conexão não criptografada entre um cliente e um servidor mesmo que estes, se encontrem em uma rede confiável, mostrando por meios de exemplos que informações sigilosas como senha, conta de acesso e informações pessoais e comerciais podem ser obtidas facilmente por um usuário mal intencionado, podendo, assim, prejudicar ou abusar dessas informações que obteve.

Como objetivos específicos este trabalho apresenta os seguintes:

a) Demonstrar que configurações padrões nem sempre são as mais seguras e podem apresentar vulnerabilidades

b) Determinar um método seguro para conexão entre um cliente e servidor para o protocolo SQL

c) Demonstrar de forma prática, mediante um laboratório de testes, todos os conceitos apresentados

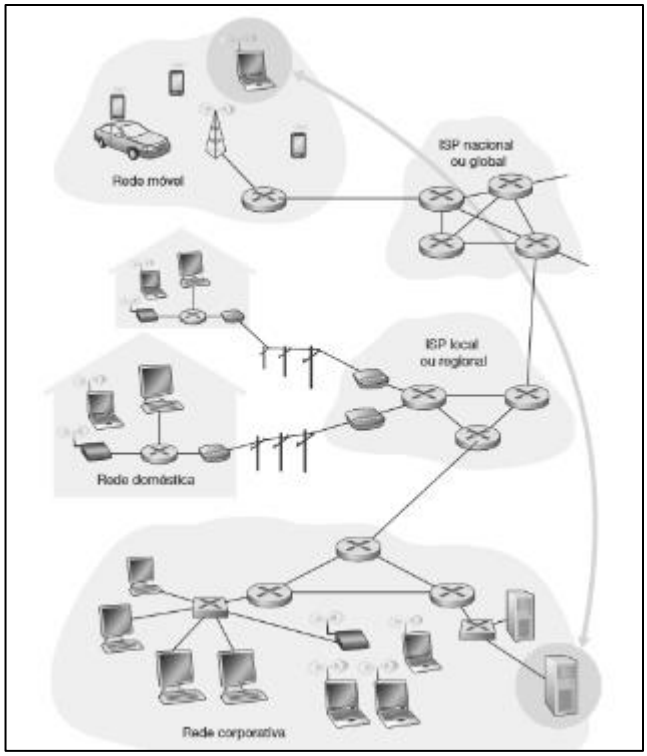
A fim de embasar o tema proposto deste trabalho, no segundo, terceiro e quarto capítulos, serão feitas breves explicações a respeito de conceitos que precisam ser entendidos para a metodologia apresentada. No quinto, será abordada a metodologia em si. No sexto é apresentado o ambiente de teste que será utilizado, bem como suas configurações. No sétimo serão colocados em prática os conceitos trabalhados nos capítulos anteriores fazendo o uso das ferramentas citadas a fim de demonstrar os perigos de uma conexão não criptografada para uma aplicação de banco de dados MySQL. No oitavo, será demonstrado uma medida protetiva em relação ao ataque demonstrado. No nono, será feita uma conclusão de acordo com os resultados coletados do método empregado.

Assim, este documento analisou alguns pontos de falha que se pode enfrentar quando se possui uma configuração mal realizada para conexão entre um cliente e um servidor de banco de dados, para demonstrar, através de maneira prática, como é importante que dados sigilosos sejam trafegados pela rede com uma criptografia forte, a fim de evitar escutas e interceptações não autorizadas de terceiros, bem como uma técnica que será capaz de tornar as atividades demonstradas aqui ineficientes para tornar o ambiente mais seguro.

2 CONCEITO DE REDES

Uma rede de computadores, segundo Tanembaum (2002, p.1), são computadores que estão conectados entre si por uma tecnologia única. Maya (2016) também define uma rede de computadores como sendo um grupo de sistemas que estão ligados um com o outro por canais de comunicação a fim de facilitar a própria comunicação e o compartilhamento de recursos entre diversos usuários. Para Kurose (2013, p.7) esses sistemas interligados são denominados hospedeiros, ou sistemas finais. Segundo o mesmo autor, no ano de 2011 haviam cerca de 850 milhões de sistemas finais que estariam conectados à Internet, sem contar os dispositivos que estão conectados de maneira intermitente como, notebooks ou smartphones por exemplo. Esses sistemas finais são conectados entre si pelo que Kurose chama de enlaces, ou links de comunicação e switches (comutadores de pacotes). A Figura 1 exemplifica esta conexão entre os diversos sistemas finais.

Figura 1 - Interação entre Sistemas Finais



Fonte: Kurose (2013, p.8)

Neste sentido, quando um sistema final precisa enviar alguma informação para outro ele irá dividir esses dados e adicionará bytes no cabeçalho de cada segmento resultado dessa divisão. Essa informação resultante é definida como pacote. Esse pacote por sua vez será encaminhado para o sistema final de destino onde poderão ser remontados e originarão os dados originais.

Kurose ainda diz que as aplicações que temos hoje em dia como, navegação web, VoIP (*Voice over IP*), *peer-to-peer*[2], televisão pela Internet, vídeos em tempo real, são denominadas aplicações distribuídas. Ele as define assim porque estas aplicações envolvem diversos sistemas finais ao mesmo tempo para envio e recebimento das informações

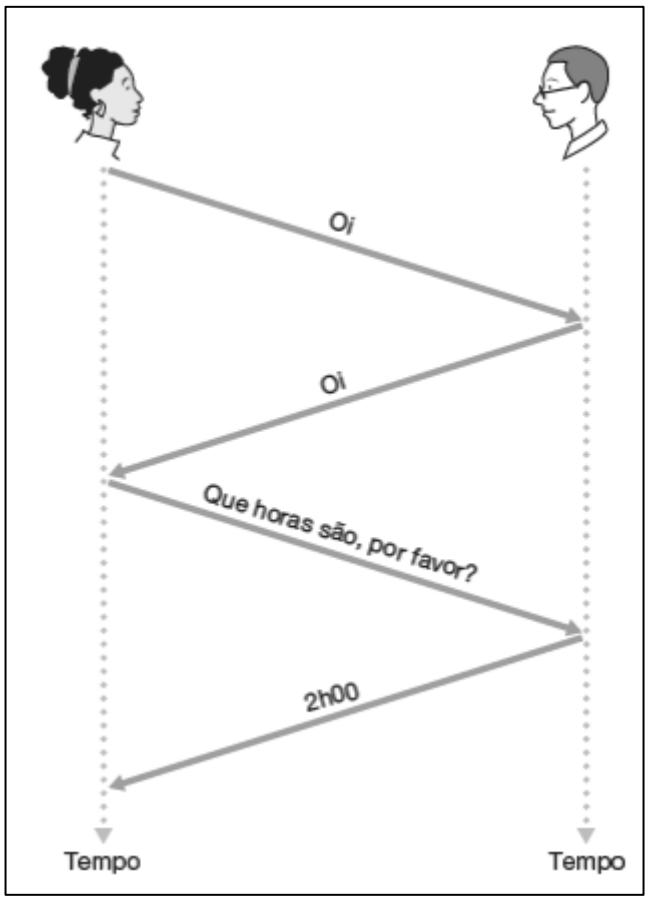
Os sistemas finais definidos por Kurose podem ainda ser divididos em duas categorias: clientes e servidores. Segundo o autor, clientes são PCs, notebooks, smartphones, etc. Já os servidores tendem a ser máquinas mais robustas e poderosas que são utilizados para armazenamento e distribuição de páginas da web, vídeos, e-mails e assim por diante.

[2] Forma de disposição de computadores interligados, onde cada um realizada funções de cliente e servidor ao mesmo tempo, sendo assim, descentralizado (MEYER, 2015)

2.1 PROTOCOLOS

Esta comunicação entre os computadores ou entidades, se dá através de protocolos. Kurose (2013, p.7), define um protocolo como o formato e a ordem em que a mensagens serão enviadas entre duas ou mais entidades que estão se comunicando, além das ações que serão feitas na transmissão e também no recebimento desta mensagem. Uma analogia interessante proposta pelo mesmo autor é a analogia humana para se perguntar a hora a alguém, ilustrada na Figura 2:

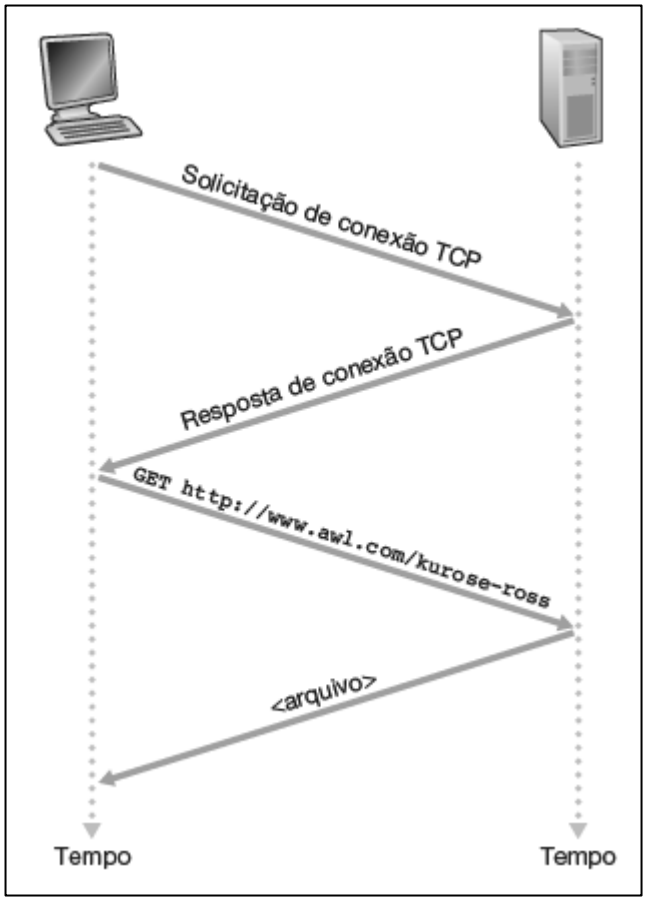
Figura 2 - Protocolo humano de pergunta de horas



Fonte: Kurose (2013, p.6)

Duas entidades conectadas em rede se comunicam da mesma forma, ilustrado na Figura 3:

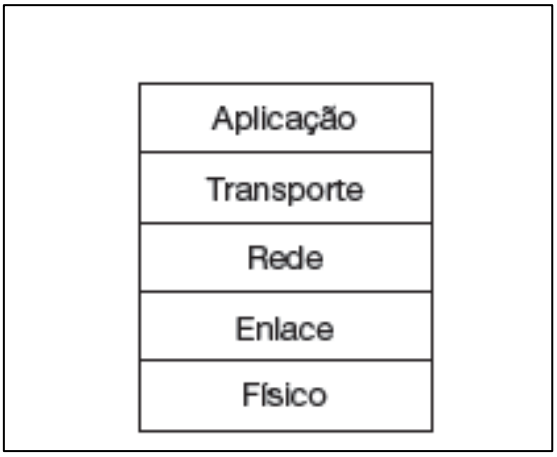
Figura 3 - Protocolo entre duas entidades em uma rede



Fonte: Kurose (2013, p.6)

Kurose divide estes protocolos em camadas de protocolo, onde cada uma terá uma função específica. Segundo o autor nós temos a seguinte divisão, demonstrada pela Figura 4.

Figura 4 - Pilha de protocolos da internet de cinco camadas



Fonte: Kurose (2013, p.37)

A tabela 1 define as camadas apresentadas, bem como exemplifica os tipos de protocolos mais comuns de cada uma, segundo Kurose (2013).

Tabela 1- Explicação das camadas da pilha de protocolos da internet

Aplicação	Camada onde ficam as aplicações de rede bem como seus protocolos. Ex: HTTP, SMTP, FTP e SSH.
Transporte	Sua função básica é acentuar dados da camada acima dela, dividi-los em unidades menores se necessário, repassar essas unidades a camada de rede e assegurar (ou não) que todos os fragmentos chegarão ao outro lado. Fornece serviços como controle de congestionamento e regula velocidade de transmissão. Exemplo de protocolos: TCP e UDP
Rede	É responsável pela movimentação, de uma máquina para outra. Implementa-se os serviços de Roteamento. Nessa camada se implementa o protocolo IP.

	A tarefa dessa camada é entregar os pacotes IPs (datagramas) onde eles devem chegar
Enlace	Ela é responsável por rotear um datagrama por meio de uma série de comutadores de pacotes (roteadores de internet) entre a origem e o destino
Física	Como a tarefa da camada de enlace é movimentar quadros de um elemento da rede para outro, a camada física é responsável por movimentar os bits que estão dentro do quadro de um nó para o seguinte. O protocolo desta camada depende do enlace, ou seja, do meio de transmissão que ele usa (fios de cobre, fibra ótica, etc.)

Fonte: Autor

O protocolo que será abordado neste documento será o MySQL. Este protocolo é utilizado por um gerenciamento de banco de dados homônimo desenvolvido, distribuído e suportado pela *Oracle Corporation*.

3 BANCO DE DADOS

Para Elmasri e Navathe (2005, p.3) um banco de dados pode ser definido, de uma maneira genérica, como uma coleção de dados que estão relacionados entre si. Estes dados são fatos que poderão ser gravados e que possuem um significado.

Como exemplo o autor cita, nomes, números de telefone e endereços de pessoas. Estes tipos de informações podem ser escritas em agendas, planilhas ou em programas de computador. Assim sendo, estes dados são uma coleção e, portanto, podem ser definidos como um banco de dados.

Os autores ainda elaboram que um banco de dados pode ser mantido tanto manualmente quanto automatizado. Neste último caso utiliza-se os chamados SGDBs, ou Sistema Gerenciador de Banco de Dados. Um SGDB pode ser definido como um ou mais programas que dão a capacidade do usuário de criar e manter um banco de dados. Ainda segundo os mesmos autores, as funções dos SGDBs podem ser definidas como:

- Definição: Especificar quais serão os tipos de dados usados, suas estruturas e restrições;
- Construção: Processo para armazenar os dados em determinada mídia que será posteriormente usado pelo SGDB;
- Manipulação: Esta função incluiu algumas subfunções, como por exemplo, pesquisas no banco de dados para obter um dado específico, atualização de dados para refletir as mudanças e gerar um relatório dos mesmos;
- Compartilhamento: Permite que diversos usuários acessem, ao mesmo tempo, o banco de dados;

Outras funções ainda mencionadas pelo autor são a proteção e manutenção, que incluem medidas contra um mau funcionamento do sistema e segurança contra acessos não autorizados

Existem diversos SGDBs disponíveis no mercado hoje em dia, alguns que podem ser citados são SQLite, MariaDB, Microsoft Access, PostgreSQL e MySQL. Para a demonstração que será feita neste documento optou-se pelo uso do SGDB MySQL.

Segundo Anley (2004), o MySQL é o banco de dados *open source*[3] mais popular do mundo. Alguns motivos que o tornam tão popular são: o fato de ser gratuito, pode ser executado em uma gama de plataformas, é relativamente simples de usar, de fácil configuração e possui uma boa performance sobre uma carga significativa.

[3] Código de um software que deve ser disponível para redistribuição sem restrição ou cobrança e a licença deve permitir a criação de modificações e trabalhos derivativos (O'Reilly, 1999, p.32)

4 SISTEMAS SEGUROS

Segundo Garfinkel e Spafford (1996 *apud* CANSIAN, 1997) um sistema computacional seguro é aquele que se comporta da maneira com que se espere que ele se comporte. O nível de confiabilidade de um sistema se dá através da observação de seu comportamento esperado com aquele que ele realmente apresenta.

Cansian (1997) ainda aponta uma definição mais rígida para a segurança da informação. Esta definição se baseia na confidencialidade, integridade e disponibilidade das informações. Estes três aspectos fazem parte do que é denominado os três pilares da segurança da informação. A tabela 2 classifica estes pilares, segundo Morrissey (2010).

Tabela 2 - Os três pilares da Segurança da Informação

Pilar	Definição
Confidencialidade	Garantia de que a informação só é acessível para aqueles que possuem acesso a ela
Integridade	Garantia de que os dados estejam completos em sua estrutura e que não sofreram nenhum tipo de adulteração, todas as suas características devem corretas
Disponibilidade	Garantia de que a informação necessária esteja disponível para quem a requisitou, no momento em que ela for requisitada

Fonte: Morrissey (2010), adaptado pelo autor

Assim, um sistema que não forneça esses três aspectos, será tratado como não confiável, já que um ou mais dos aspectos não estarão presentes.

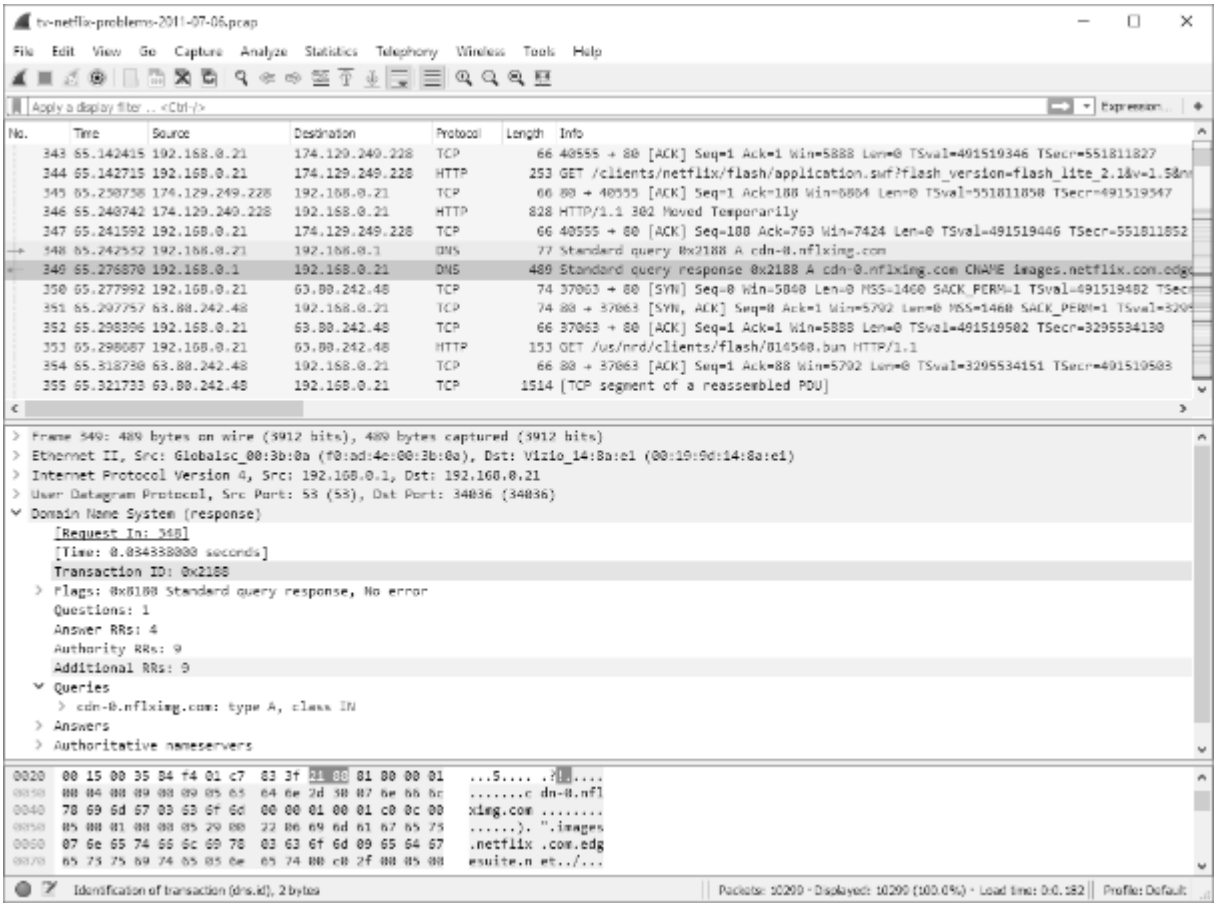
5 METODOLOGIA

A metodologia utilizada, se baseará em um cenário real enfrentado por uma empresa do ramo educacional, onde foi explorado com sucesso o acesso não autorizado aos dados através da captura do tráfego de rede.

Para demonstrar esta captura do protocolo *MYSQL* em uma rede, será utilizado o programa, também *open source*, *Wireshark*.

A *SecTools* [201-], um site que busca exibir as melhores ferramentas de segurança de rede, criado pelos mantenedores do projeto *Nmap*[4], define o *Wireshark* como um analisador de protocolos de rede que permite a visualização ao vivo de dados de uma rede específica, permitindo que o usuário possa interativamente navegar pelos dados capturados, detalhando cada um dos pacotes, a Figura 5 mostra a janela principal do programa.

Figura 5 - Janela principal do *Wireshark*



Fonte: *Wireshark User's Guide* (2018, p.6)

[4] Ferramenta de descoberta e auditoria de rede

A *SecTools*, no momento da escrita deste documento, lista esta ferramenta em primeiro lugar entre as 125 melhores ferramentas de segurança de rede.

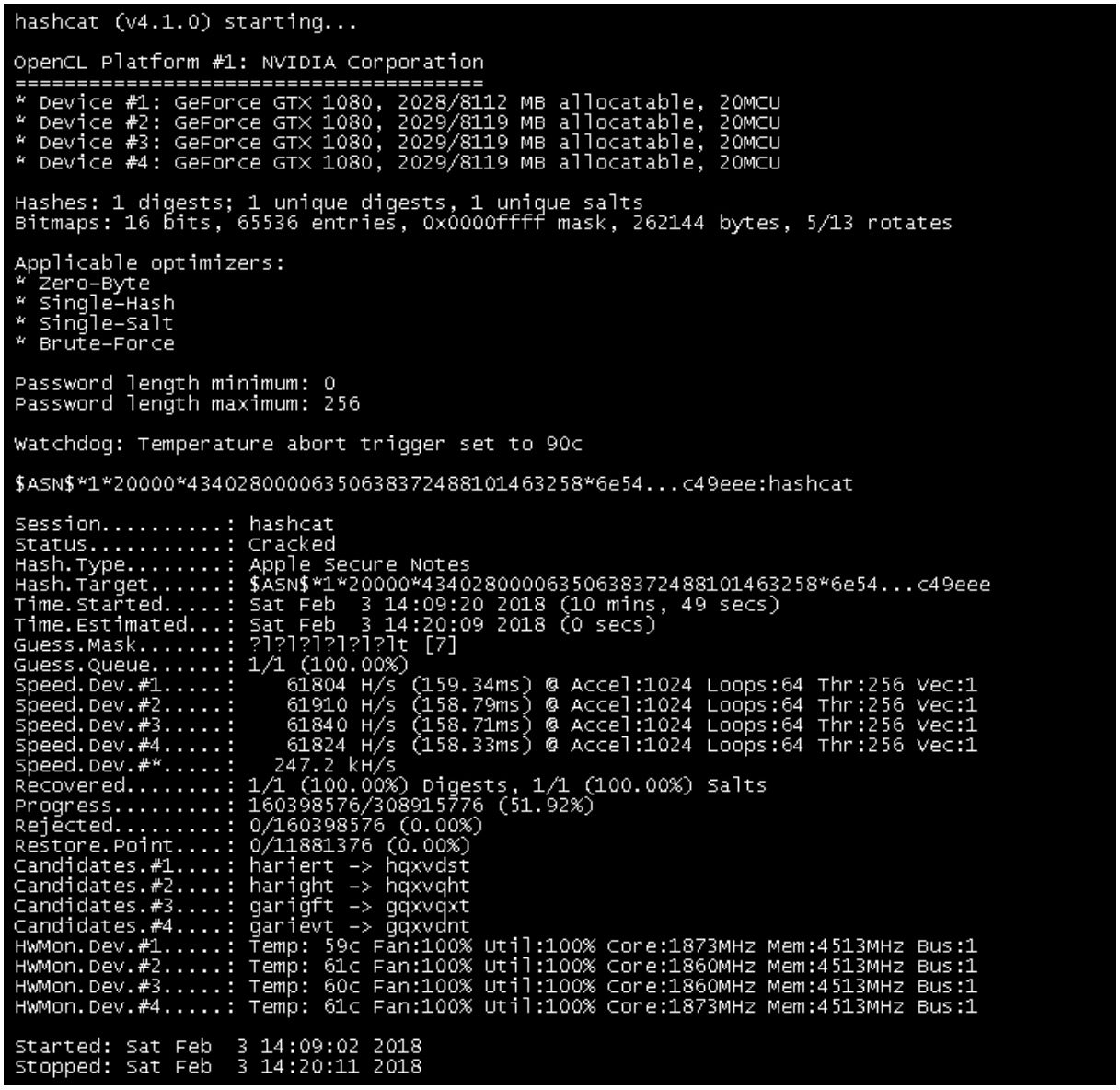
Algumas das funcionalidades deste programa incluem, mas não se limitam a:

- Inspeção profunda de centenas de protocolos
- Captura ao vivo e análise off-line
- Multiplataforma (Windows, Linux, Solaris, FreeBSD)
- Análise VoIP
- Interface gráfica para os dados capturados

As funcionalidades completas podem ser vistas no site oficial da ferramenta, em www.wireshark.org/about.html.

Com o tráfego de rede capturado pelo programa *Wireshark*, será mostrado que credenciais de conexão ao banco de dados podem ser recuperadas, mesmo não se sabendo quais são. O programa que será usado para isso é o *Hashcat*. Segundo o site original da ferramenta, ele é a utilidade de recuperação de senhas *open source* mais rápida do mundo (*HASHCAT*, [201-?]).

Figura 6 - Exemplo de saída do programa Hashcat



Fonte: Hashcat [201-?]

Este programa possui uma gama muito grande de algoritmos que podem processar diversos tipos de funções *hash*[5]. Além de diversos algoritmos, este programa possui diversos tipos de ataque, como por exemplo: ataque direto, combinação, força bruta e dicionário.

Como demonstração foi utilizado um pequeno laboratório de máquinas virtuais para simulação de um ambiente de rede baseado em um cenário real. A tabela 3 exemplifica quais serão os elementos constituídos pelo laboratório.

[5] Algoritmo matemático que é capaz de transformar um dado em uma sequência de caracteres que possuem um comprimento fixo (DONOHUE, 2014)

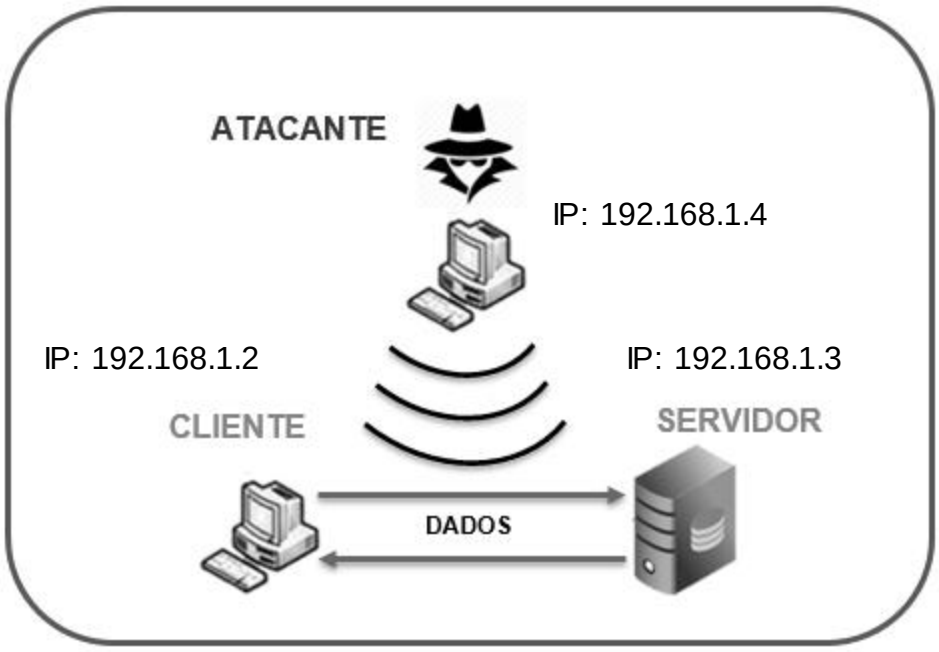
Tabela 3 - Elementos que comporão o ambiente simulado

Nome	Sistema Operacional
Cliente	Windows 7
Servidor	Linux Ubuntu
Atacante	Kali Linux

Fonte: Autor

A Figura 7, ilustra a topologia que será utilizada no ambiente simulado

Figura 7 - Topologia utilizada



Fonte: Autor

Existem diversas técnicas para invasão de redes, no entanto, não será o intuito deste trabalho apresentar ou demonstrar tais técnicas. Para todos os efeitos será assumido que o atacante obteve um ataque bem sucedido, já está presente na rede e que o mesmo pode se comunicar com qualquer outro componente que esteja também presente nela.

Nesse sentido, a topologia apresentada nos traz três elementos, um cliente, um servidor e o atacante, que interceptará os dados trafegados entre o cliente e o servidor.

Para Kurose (2013, p.8) um programa cliente é qualquer programa que solicita e recebe algum tipo de serviço de um programa servidor, assim, é o cliente que sempre iniciará

uma conexão para um servidor, que por sua vez, receberá esta conexão, a processará e devolverá a resposta ao cliente.

O atacante executará o programa de capturas de pacote *Wireshark* a fim de monitorar todo o tráfego na rede. Desta maneira, quando uma solicitação de conexão ao banco de dados do servidor for feita, todos os dados referentes a ela, como usuário e senha por exemplo, poderão ser interceptados. Vale lembrar que, analisadores de pacotes, como o *Wireshark*, não introduzem pacotes na rede, desse modo, eles são difíceis de detectar e provavelmente tanto o cliente como o servidor não saberão o que está ocorrendo e que a sua comunicação pode estar sendo interceptada (KUROSE, 2013, p.43).

> "Muitos usuários hoje acessam à Internet por meio de aparelhos sem fio, como laptops conectados à tecnologia Wi-Fi ou aparelhos portáteis com conexões à Internet via telefone celular [...]. Enquanto o acesso onipresente à Internet é extremamente conveniente e disponibiliza novas aplicações sensacionais aos usuários móveis, ele também cria uma grande vulnerabilidade de segurança – posicionando um receptor passivo nas proximidades do transmissor sem fio, o receptor pode obter uma cópia de cada pacote transmitido! Esses pacotes podem conter todo tipo de informações confidenciais, incluindo senhas, número de inscrição da previdência social, segredos comerciais e mensagens pessoais.[...] Os analisadores de pacotes podem estar distribuídos em ambientes de conexão com fio. Nesses ambientes [...] um analisador de pacote pode obter cópias de todos os pacotes enviados pela LAN" (KUROSE, 2013, p.43)

6 INTRODUÇÂO AO AMBIENTE DE TESTE

Como descrito brevemente nos capítulos anteriores, o ambiente utilizado neste teste será composto por um cliente, executando o sistema operacional *Windows 7 Ultimate*, Figura 8, um servidor banco de dados *MySQL*, executando o

sistema operacional *Ubuntu Linux* 16.04 LTS, Figura 9, e um atacante, executando o sistema operacional *Kali Linux*, Figura 10.

Figura 8 - Propriedades do Sistema Cliente

Fonte: Autor

Figura 9 - Propriedades do Sistema do Servidor

Fonte: Autor

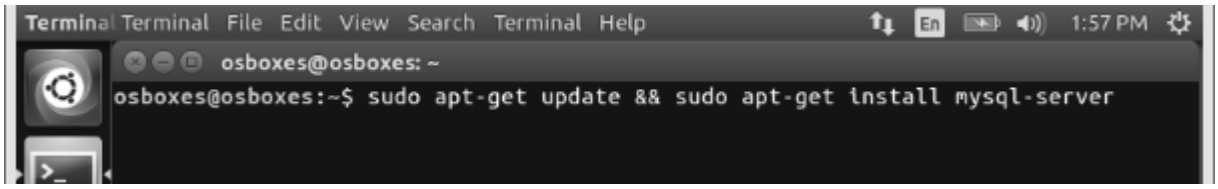
Figura 10 - Propriedades do Sistema do Atacante

Fonte: Autor

6.1 CONFIGURAÇÃO DO BANCO DE DADOS NO SERVIDOR

No sistema do servidor foi instalado o banco de dados *MySQL* versão 5.7.23. O processo para instalação do mesmo é bem simples e, segundo Virdó (2016), pode ser feito seguindo os passos demonstrados na Figura 11.

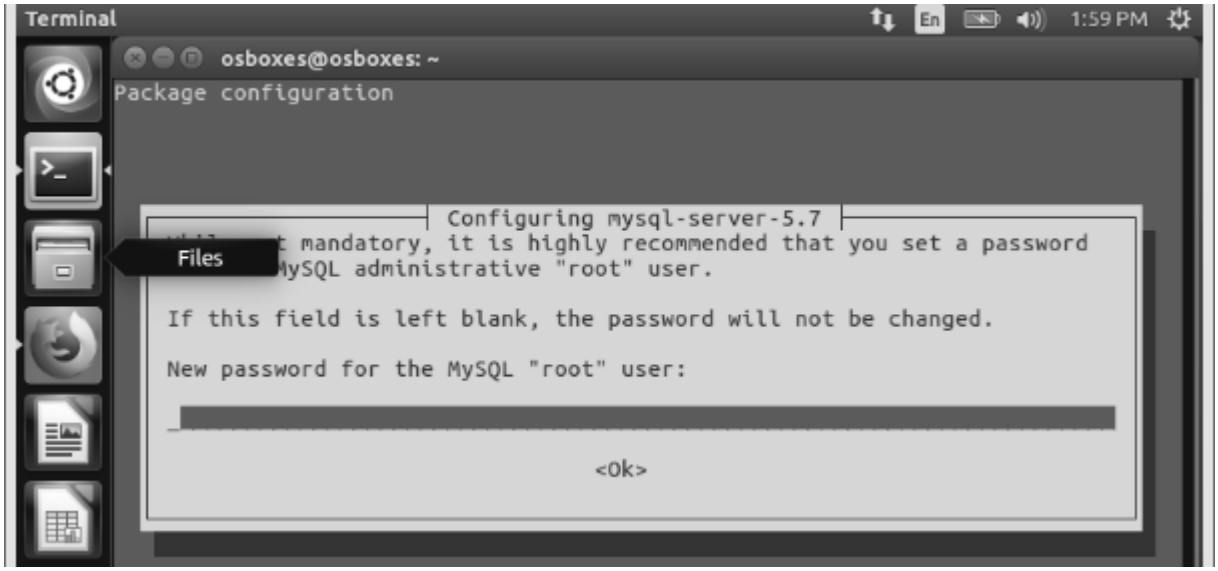
Figura 11 - Comando para instalação do servidor *MySQL*



Fonte: Autor

Durante a instalação será preciso escolher uma senha para o usuário *root* do banco de dados, assim como mostra a **Figura 12**.

Figura 12 - Solicitação de senha do usuário root do servidor de banco de dados



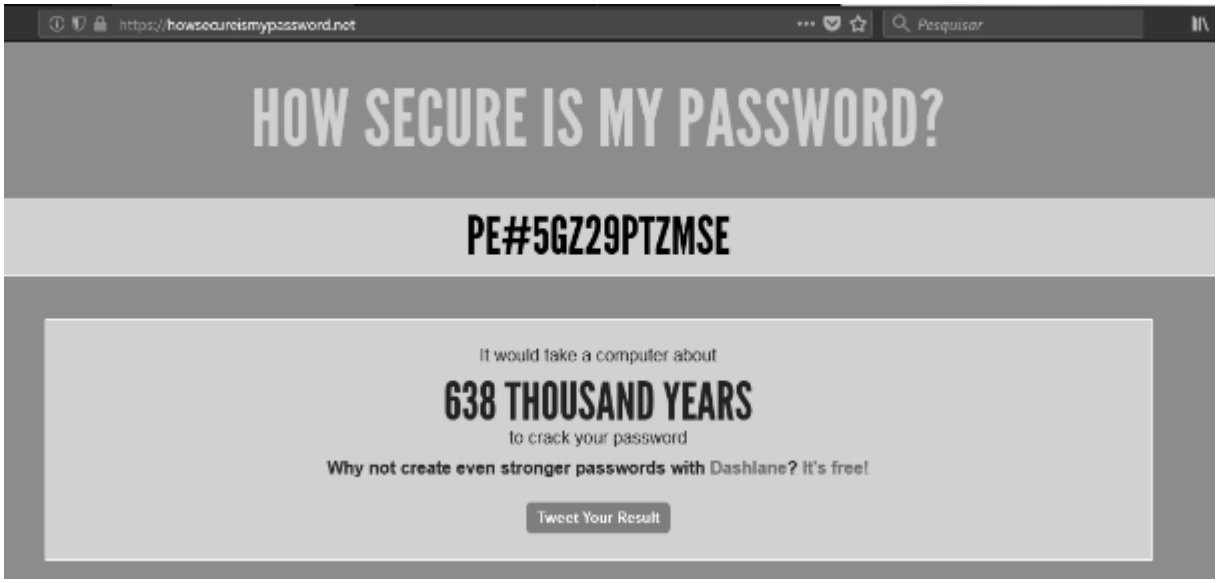
Fonte: Autor

Para demonstrar que uma senha considerada forte nem sempre é segura, será utilizado neste trabalho a seguinte senha para o usuário root do banco de dados: “PE#5GZ29PTZMSE”.

Esta senha foi utilizada em alguns analisadores de força de senhas online para validar que se trata de uma senha forte. Os serviços utilizados foram “*How Secure Is My Password?” (*HOW SECURE IS MY PASSWORD?, *2016), Kaspersky Lab Secure Password Check (*KASPERSKY LAB, 2018*)* e *“How Secure Is My Password?” (*LASTPASS, 2018*)*. Todos esses serviços são gratuitos e podem ser utilizados por qualquer pessoa para verificação de suas senhas.

A Figura 13, Figura 14 e Figura 15 demonstram quanto tempo, aproximadamente, levaria para decifrarem a senha que será usada neste laboratório, por cada serviço utilizado.

Figura 13- Análise de força da senha do "How Secure Is My Password?"

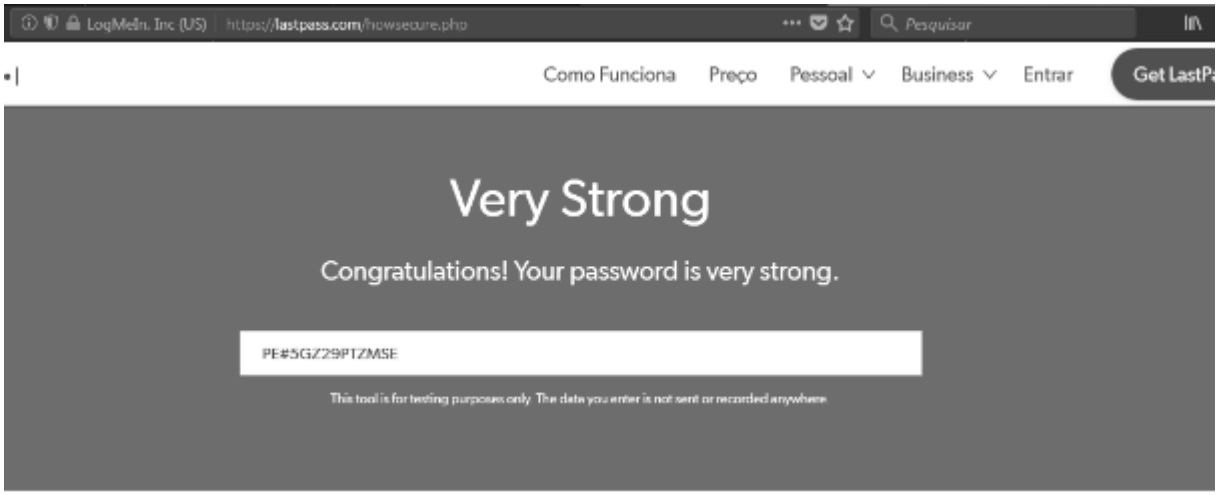


Fonte: Autor

Figura 14- Análise de força da senha do "Kaspersky Lab"

Fonte: Autor

Figura 15 - Análise de força da senha do "How Secure Is Your Password?"

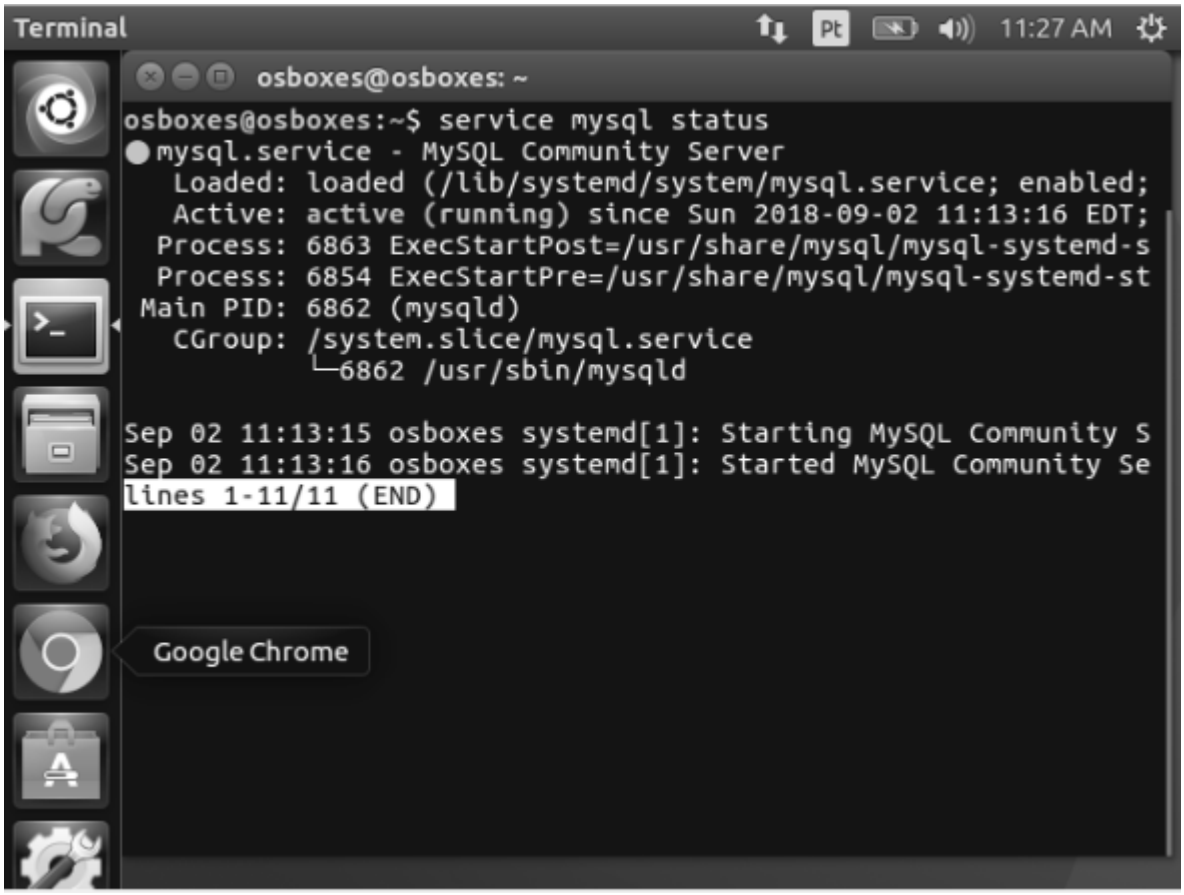


Fonte: Autor

Como pode-se observar, levariam cerca de 638 mil anos para um computador conseguir quebrar a senha, segundo o primeiro serviço e 327 séculos, segundo o serviço do Kaspersky. Para o terceiro serviço, ele aponta somente como “Muito forte”.

Após a definição da senha para o usuário *root*, a instalação continuará automaticamente. Para verificar se tudo foi instalado com sucesso, podemos rodar o comando “*service mysql status*” no servidor. A Figura 16 demonstra que o serviço MySQL foi instalado e está sendo executado corretamente.

Figura 16 - Resultado do comando executado no servidor

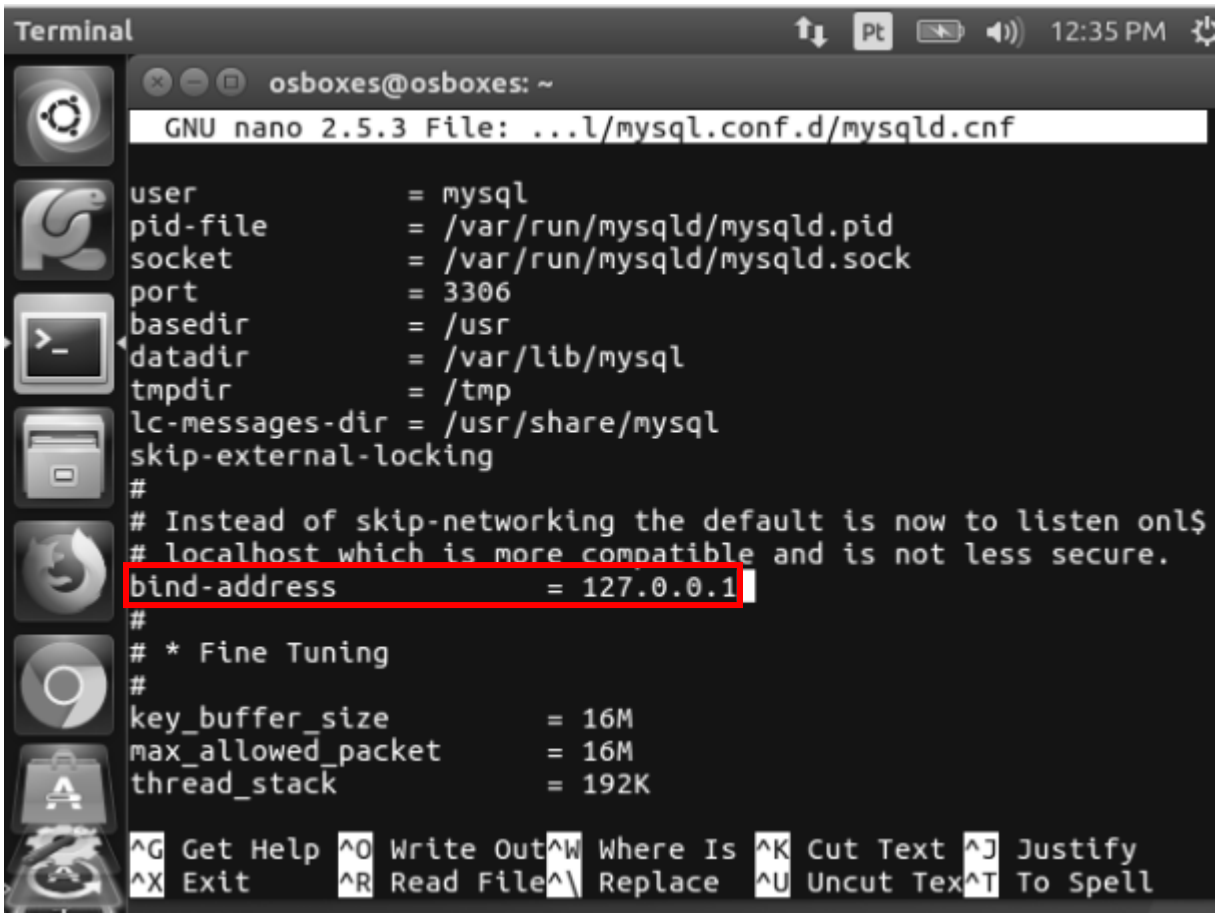


Fonte: Autor

Durante a instalação padrão do serviço MySQL, não é possível realizar conexões remotas para o mesmo, ou seja, conexões vindas de outras máquinas não são permitidas por padrão. Para permitir essa funcionalidade é necessário alterar as configurações do serviço no caminho "*/etc/mysql/mysql.conf.d/mysqld.cnf*" do servidor (WALLEN, 2017).

Como demonstra a Figura *17*, dentro deste arquivo existem diversas configurações, dentre elas a opção "*bind-address = 127.0.0.1*".

Figura 17 - Arquivo contendo as configurações do serviço MySQL

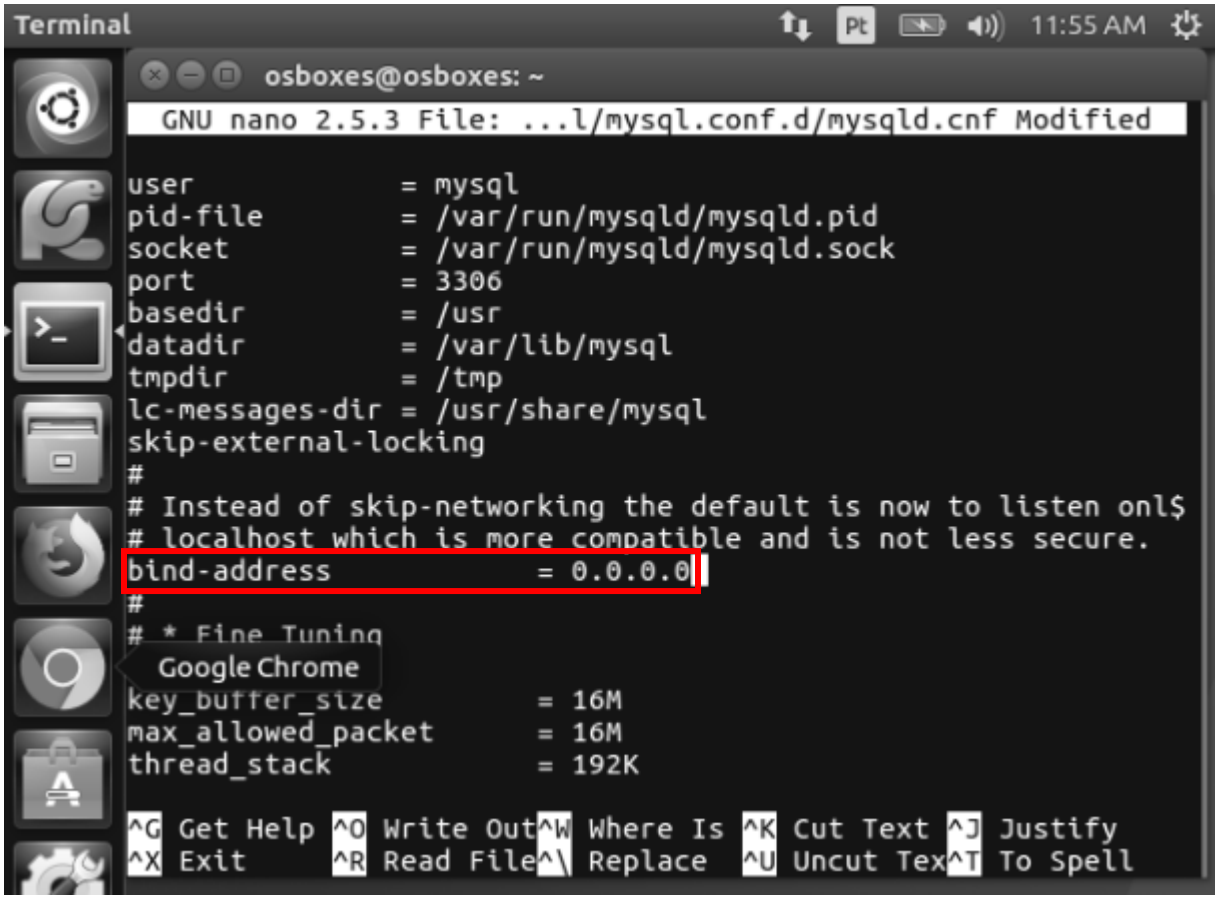


Fonte: Autor

Basicamente, esta opção permite que o servidor MySQL aceite conexões somente para o endereço permitido, ou seja, 127.0.0.1, conhecido também como *localhost*, ou o próprio servidor, para qualquer outra máquina esta conexão será negada.

Assim, para permitirmos que conexões de outras máquinas sejam autorizadas, devemos alterar a linha mencionada para “bind-address = 0.0.0.0”, fazendo com que qualquer máquina, que possua as credenciais do banco de dados, consiga acessar o servidor sem problemas. A Figura 18 mostra como o arquivo ficará, após as alterações serem feitas.

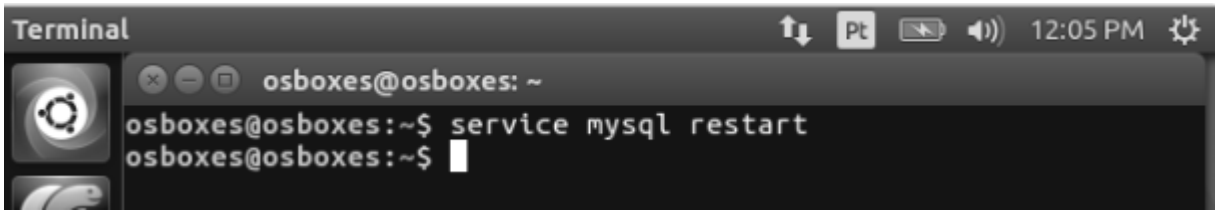
Figura 18 - Arquivo com a configuração correta



Fonte: Autor

Com o arquivo configurado da maneira correta, assim, como mostra a Figura 19, basta reiniciar o serviço com o comando "*service mysql restart*" para que as alterações entrem em vigor.

Figura 19 - Comando para reinicialização do serviço

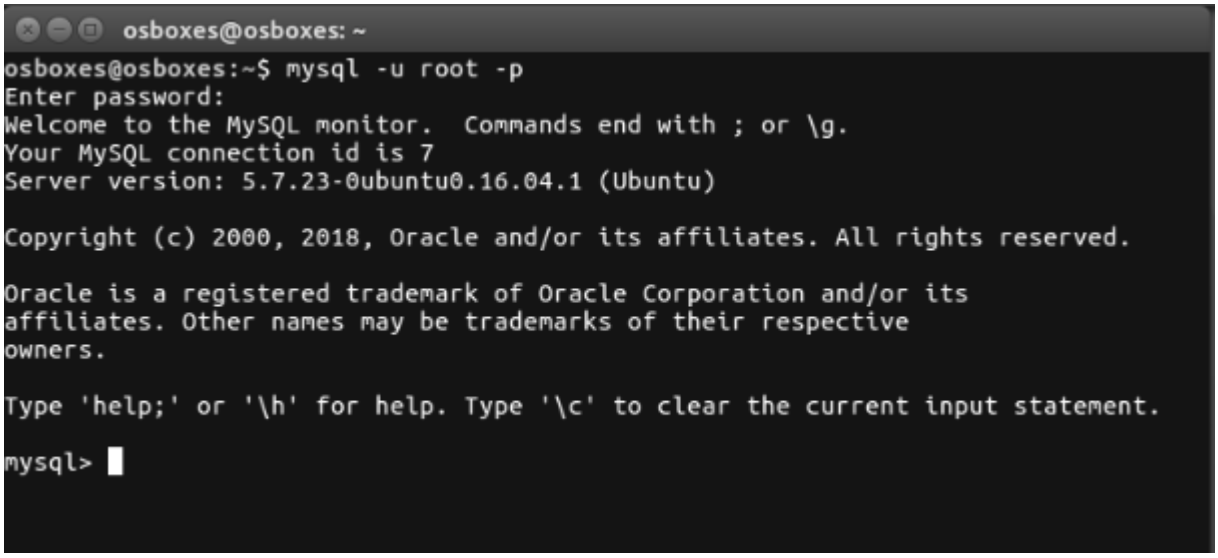


Fonte: Autor

Após o comando de reinicialização, deve-se configurar um usuário com permissões de conexão remota, neste caso utilizaremos o próprio usuário *root*. Para realizar este comando devemos conectar ao banco de dados utilizando o comando "mysql –u root -p".

Ao pedir a senha deve-se digitar a senha definida durante a instalação, no caso deste laboratório, PE#5GZ29PTZMSE. A Figura 20 mostra este conexão sendo realizada.

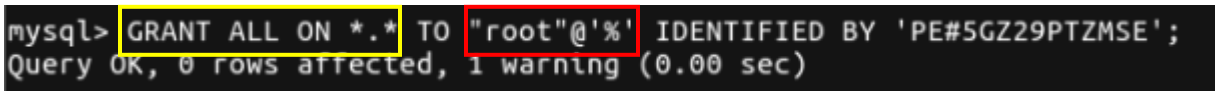
Figura 20 - Conexão realizada com o banco de dados



Fonte: Autor

A seguir deve-se alterar as permissões do usuário para habilitar o login remoto. O comando que será utilizado será o demonstrado pelo Figura 21.

Figura 21 - Comando para mudar as permissões de login remoto



Fonte: Autor

Os pontos importantes deste comando são as partes destacadas na Figura 21, utilizando o símbolo “%” estamos dizendo que o usuário *root* poderá aceitar conexões de qualquer lugar, ou seja, qualquer máquina. Já o comando “GRANT ALL ON *.*”, é usado para dar acesso total a todo o banco de dados ao usuário passado (*root*).

Vale ressaltar que, assim como é dito pelo SANS Institute (2003), é necessário implementar uma política de menor acesso para qualquer sistema. Isto significa que os usuários somente irão ter e manter as permissões e privilégios

necessários para suas tarefas diárias. Como exemplo pode-se citar um usuário comum com permissões de escrita e leitura em um banco de dados. Caso as tarefas delegadas a ele sejam somente para extrair dados do banco, não haveria necessidade da permissão de escrita para ele. Se, por acaso, a conta deste usuário for comprometida por uma pessoa mal intencionada, ela poderá fazer alterações indesejadas dentro do banco de dados, como por exemplo, deletando configurações, alterando ou modificando informações.

No entanto, muitas empresas acabam esquecendo ou não tendo tempo para esse tipo de configuração de menor acesso e acabam não configurando seus serviços da maneira mais correta. É comum também as empresas permitirem este tipo de acesso remoto para facilitar o suporte remoto aos seus servidores, de modo que não será necessário estar fisicamente presente no local para fazer uma configuração ou resolver um problema.

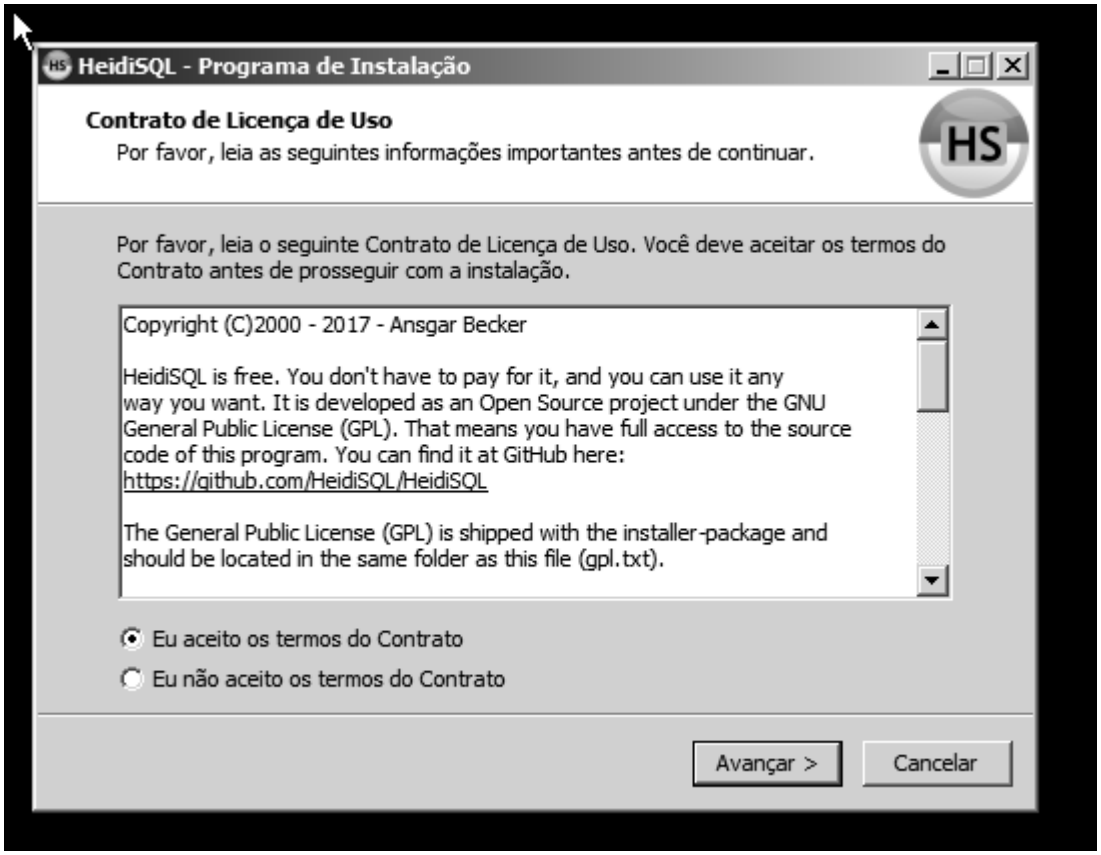
Assim, após o comando executado, o servidor já estará preparado para receber conexões remotas através do usuário *root*. A seguir será demonstrado uma conexão entre o cliente e o servidor

6.2 CONFIGURAÇÃO DO CLIENTE

Para o cliente não será necessário muita configuração, bastando apenas do programa *HeidiSQL* para visualização do banco de dados no servidor de maneira mais amigável. Este programa está disponibilizado de maneira gratuita no site oficial, https://www.heidisql.com/. Ele permite que você edite dados, crie tabelas, *views*, procedimentos, *triggers* e eventos de uma maneira mais fácil e confiável (HEIDISQL WEBSITE, 2018).

A instalação é feita de maneira simples, bastando apenas seguir o assistente da instalação, a Figura 22 mostra a tela inicial deste assistente.

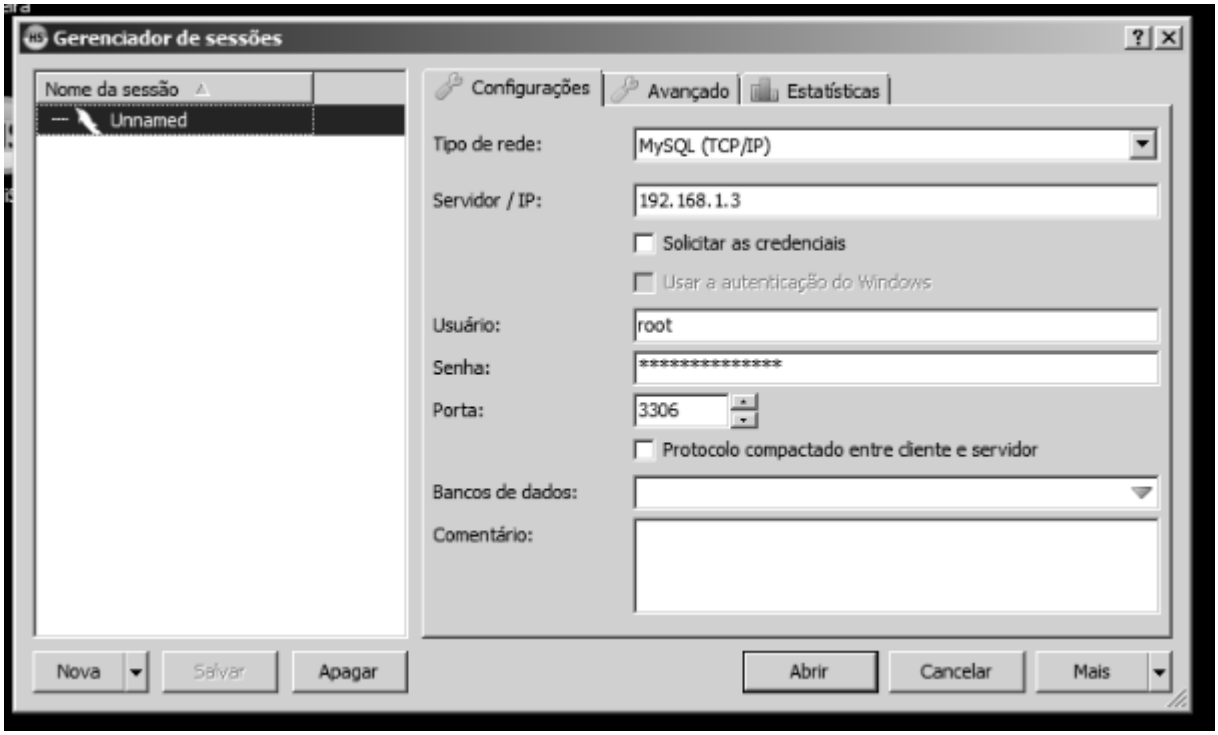
Figura 22 - Tela inicial do assistente de instalação do *HeidiSQL*



Fonte: Autor

Após a instalação, basta executar o programa, sua tela inicial é representada pela Figura 23.

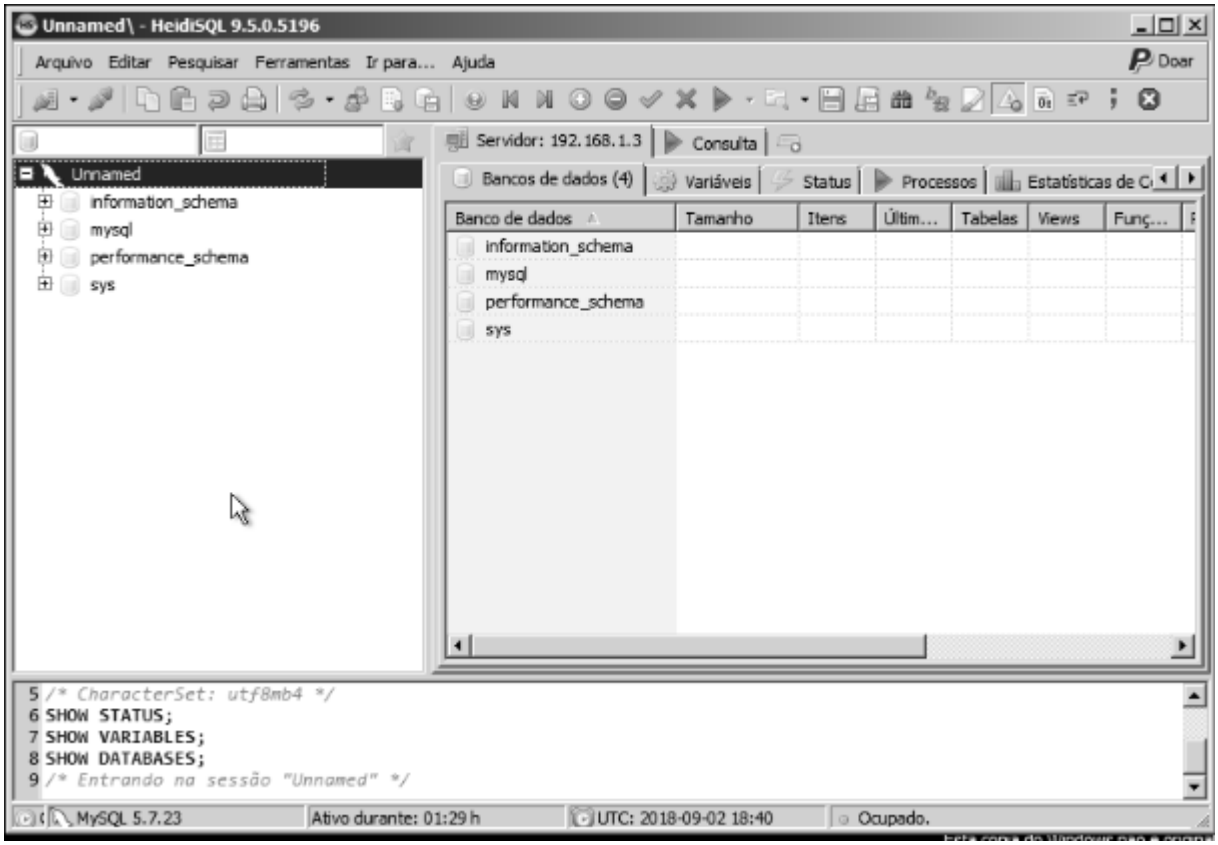
Figura 23 - Tela inicial do programa *HeidiSQL*



Fonte: Autor

Nesta tela deverá ser configurado o tipo de conexão que será feita para o banco de dados. Deve-se colocar o IP do servidor, que no caso deste laboratório é 192.168.1.3, o usuário no qual se deseja fazer a conexão (*root*) e a senha deste usuário (PE#5GZ29PTZMSE). Assim como mostra a Figura 24, clicando em “Abrir” a conexão será feita e será possível ter acesso ao banco de dados presente no servidor e consequentemente, acesso a toda a sua estrutura.

Figura 24 - Conexão bem sucedida com o banco de dados, mostrando os bancos disponíveis



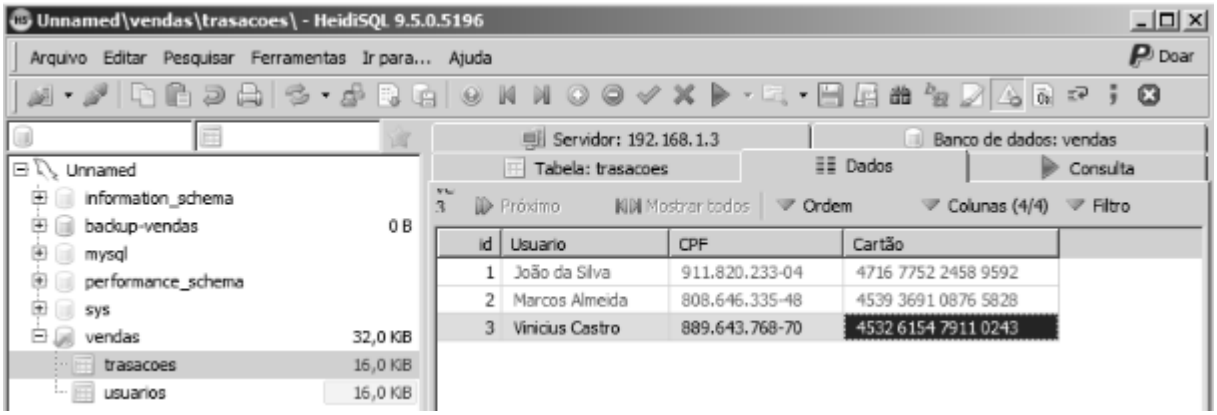
Fonte: Autor

Foram criados alguns registros neste banco de dados para demonstração deste laboratório, eles foram os seguintes:

- Dois bancos de dados chamados, “vendas” e “backup-vendas”
- Dentro do banco de vendas, foram criadas duas tabelas: transações e usuários
- Dentro da tabela de transações foram colocados alguns dados, como ID, usuário, CPF e número do cartão
- Dentro da tabela de usuários, foram colocados dados como ID, Usuário e Senha

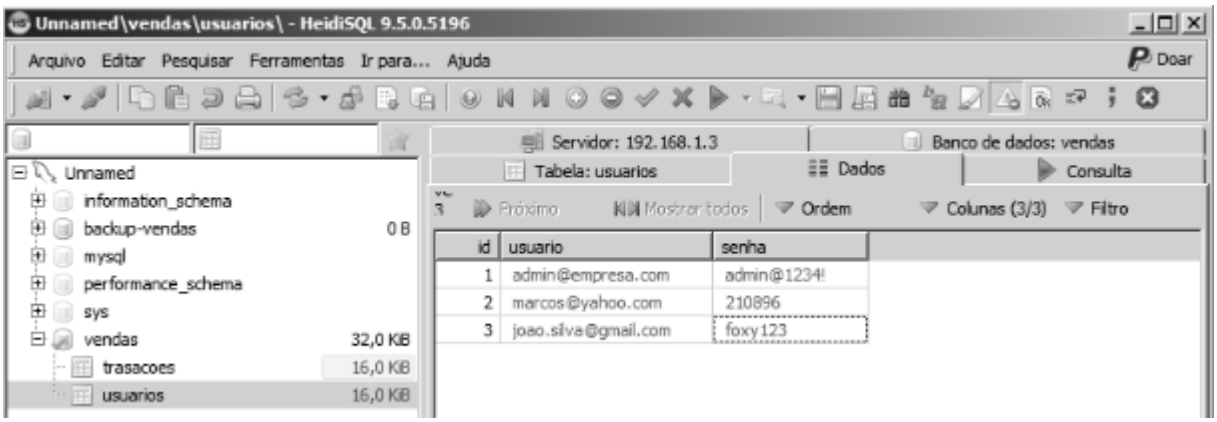
As Figura 25 e Figura 26 ilustram esses registros

Figura 25 - Conteúdo da tabela de transações



Fonte: Autor

Figura 26 - Conteúdo da tabela de usuários



Fonte: Autor

Com o banco de dados e os registros que o compõem devidamente criados, será demonstrado no capítulo seguinte a simulação do ataque em si.

7 SIMULAÇÃO DO ATAQUE

O tipo de ataque que será demonstrado neste documento será uma combinação de *sniffing* e ataques de dicionário.

Assim como ressalta Petracioli (2008), o ataque do tipo *sniffing*, que significa farejar, em tradução livre, consiste em uma prática em que se intercepta o tráfego de dados e decodifica o seu conteúdo que foi trocado entre computadores de um determinada rede, ou seja, esta técnica é utilizada para interceptar a “conversa” entre duas ou mais máquinas em uma mesma rede. Programas que utilizam esta técnica são denominado *sniffers*, ou farejadores de pacotes em português. O programa *Wireshark* é um deles.

O ataque do tipo dicionário se trata de uma técnica usada para se conseguir a senha de um determinado sistema em que o atacante tentará, através de uma lista fixa de palavras, adivinhar a senha correta. Este tipo de ataque é similar ao tipo de força bruta, porém possui vantagens como o tempo que pode ser poupado para o atacante ao tentar adivinhar a senha, já que com o ataque de força bruta, todas as possibilidades de senhas serão tentadas, caractere por caractere (CZAGAN, 2013). O tempo para se adivinhar uma senha utilizando força bruta aumenta consideravelmente ao se usar caracteres especiais nela como, @, # ou $, por exemplo. Utilizando o tipo de ataque de dicionário, será utilizado uma lista já pronta de palavras, o que resulta em um tempo mais rápido para se saber a senha. Uma desvantagem deste tipo de ataque é que nem sempre a senha estará na lista de palavras usadas e isso fará com que o ataque não seja bem sucedido.

Existem diversas listas prontas disponíveis na Internet. Essas listas contém nomes de usuários e senhas que foram mais utilizadas ou que acabaram sendo expostas em virtude de algum ataque a uma determinada empresa. Alguns exemplos que podem ser citados foram da empresa *Yahoo* em 2013, com cerca de três bilhões de contas de usuários comprometidas (SPORCK, 2017).

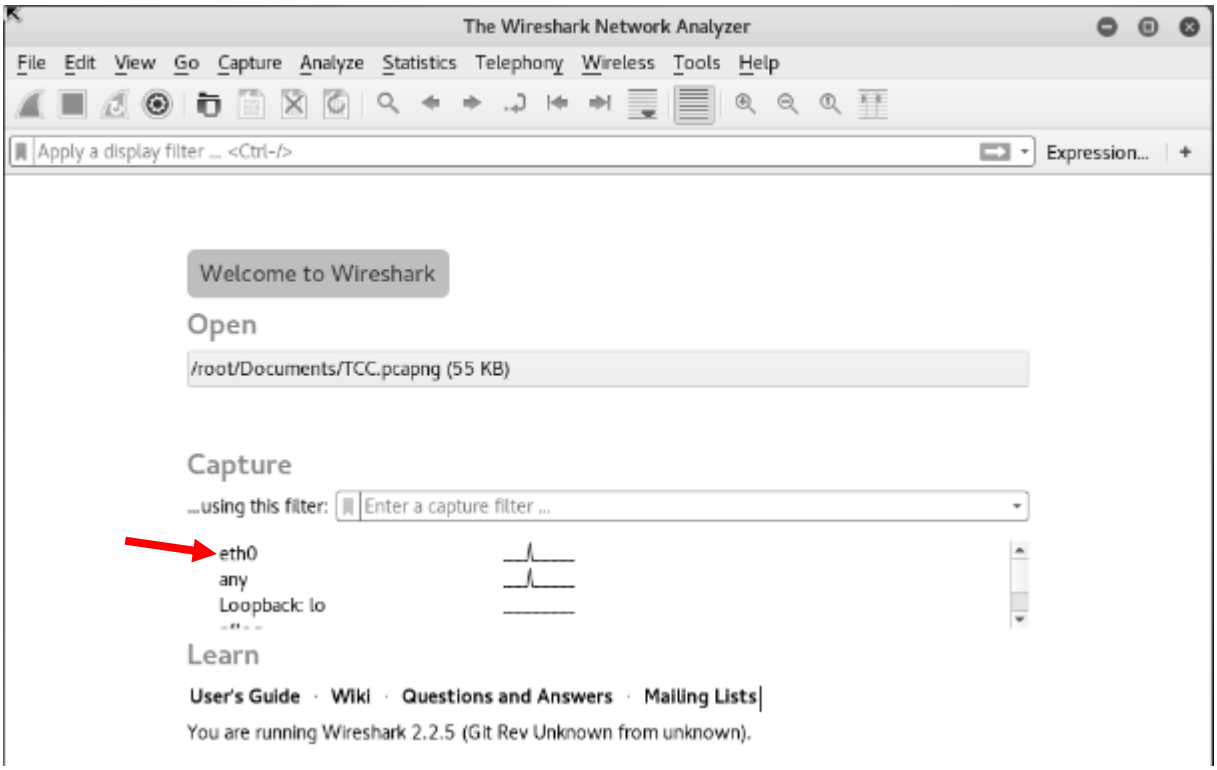
Esses dados expostos são compartilhados por diversos *sites* na internet, resultando em listas de credenciais que podem ser usadas em ataques de dicionário para determinar a senha de algum serviço, algo que será demonstrado neste teste.

7.1 CAPTURA DOS DADOS

Com o atacante presente na mesma rede do cliente, o programa *Wireshark* poderá ser executado para começar a capturar os pacotes que trafegam na rede.

Ao iniciar o programa, deve-se selecionar a placa de rede, clicando duas vezes, em que será feita a captura, na maioria das vezes esse placa de rede é denominada “eth0”, assim como mostra a Figura 27.

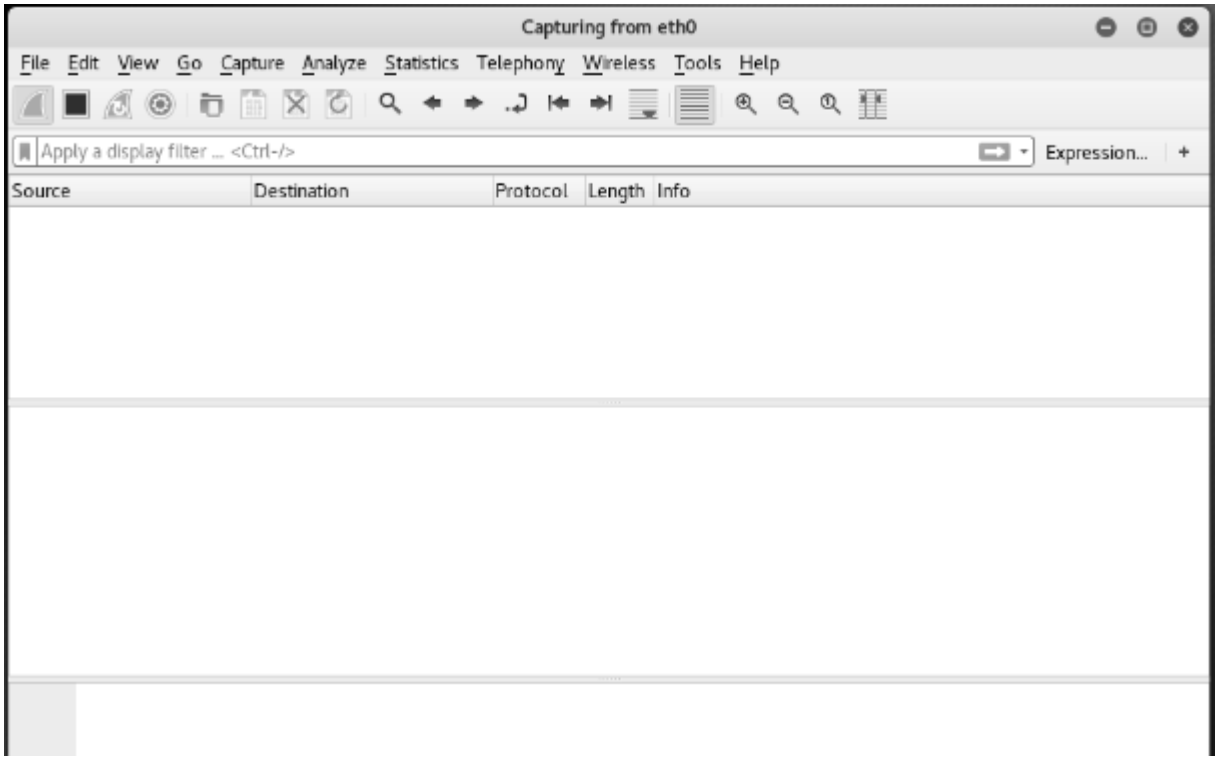
Figura 27 - Tela inicial do programa *Wireshark*



Fonte: Autor

Após isso, o *Wireshark* começará a capturar todo o tráfego que passará pela rede, a Figura 28 mostra a tela de captura do programa.

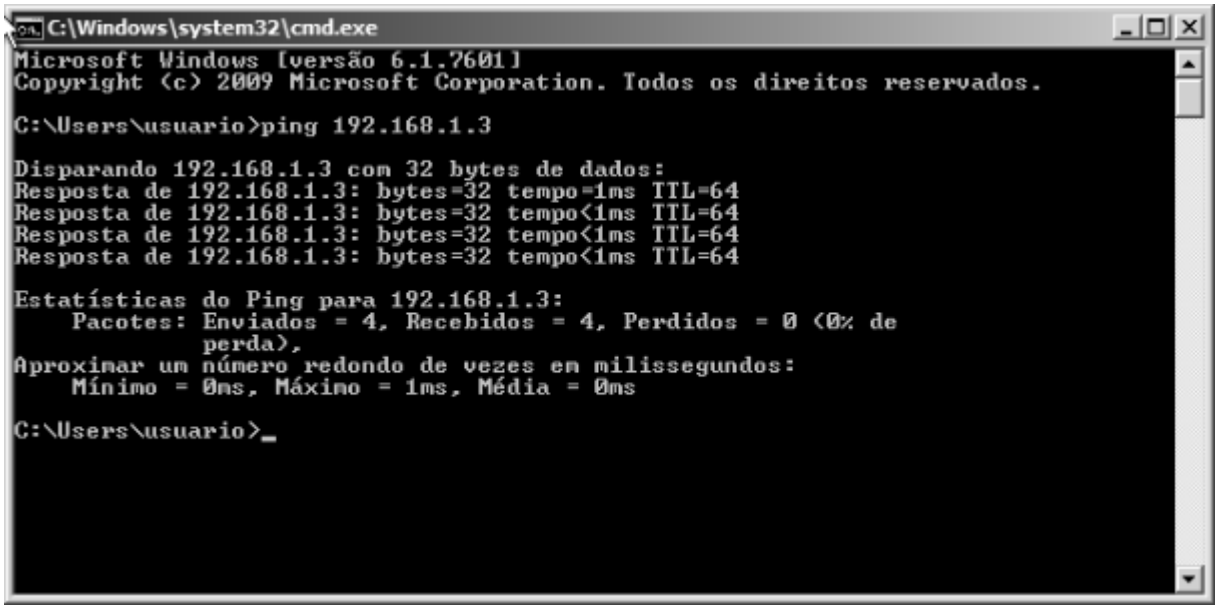
Figura 28 - Tela de captura de dados do programa *Wireshark*



Fonte: Autor

A partir desse ponto, qualquer comunicação entre o cliente e o servidor será interceptada pelo atacante. Dependendo do tipo de comunicação entre eles, os dados poderão ser facilmente decodificados. Como demonstração que as máquinas estão se comunicando entre si, foi executado um comando denominado *ping* do cliente para o servidor. Este comando serve para checar se um dispositivo consegue se comunicar com outro, a Figura 29 mostra este comando sendo executado, bem como a resposta do servidor.

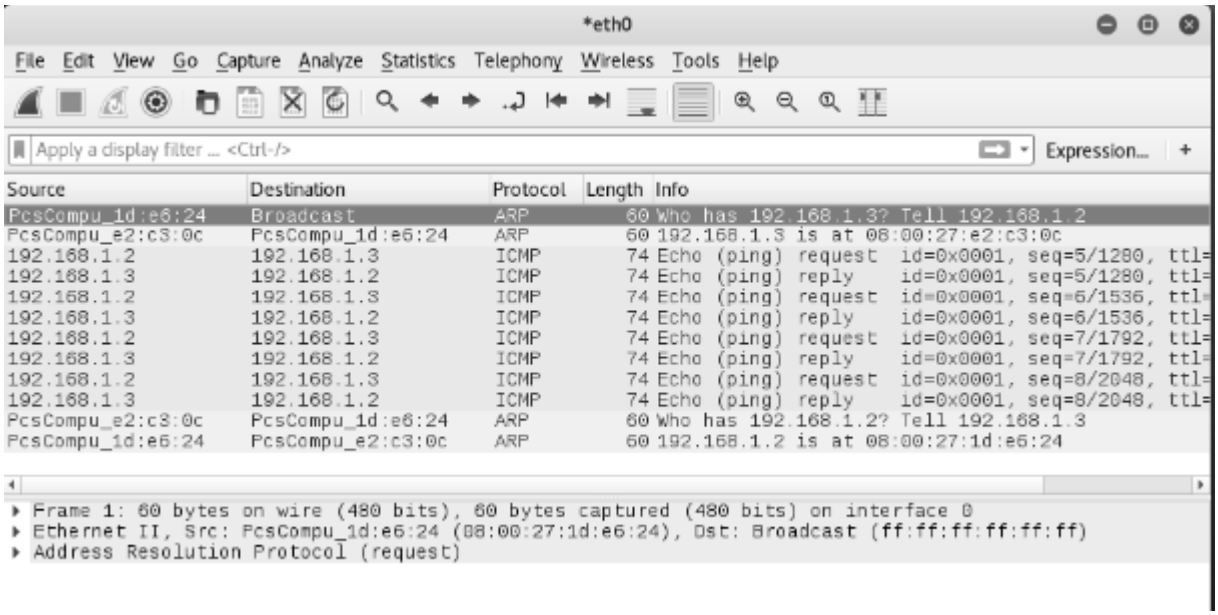
Figura 29 - Comando *ping* sendo realizado do cliente para o servidor



Fonte: Autor

Assim como mostra a Figura 30, o dispositivo atacante é capaz de interceptar o comando realizado, mostrando a origem dos dados, que partem do endereço IP 192.168.1.2 (Cliente) e possuem o destino para 192.168.1.3 (Servidor), além de outras informações como o tipo de protocolo, comprimento e informações adicionais.

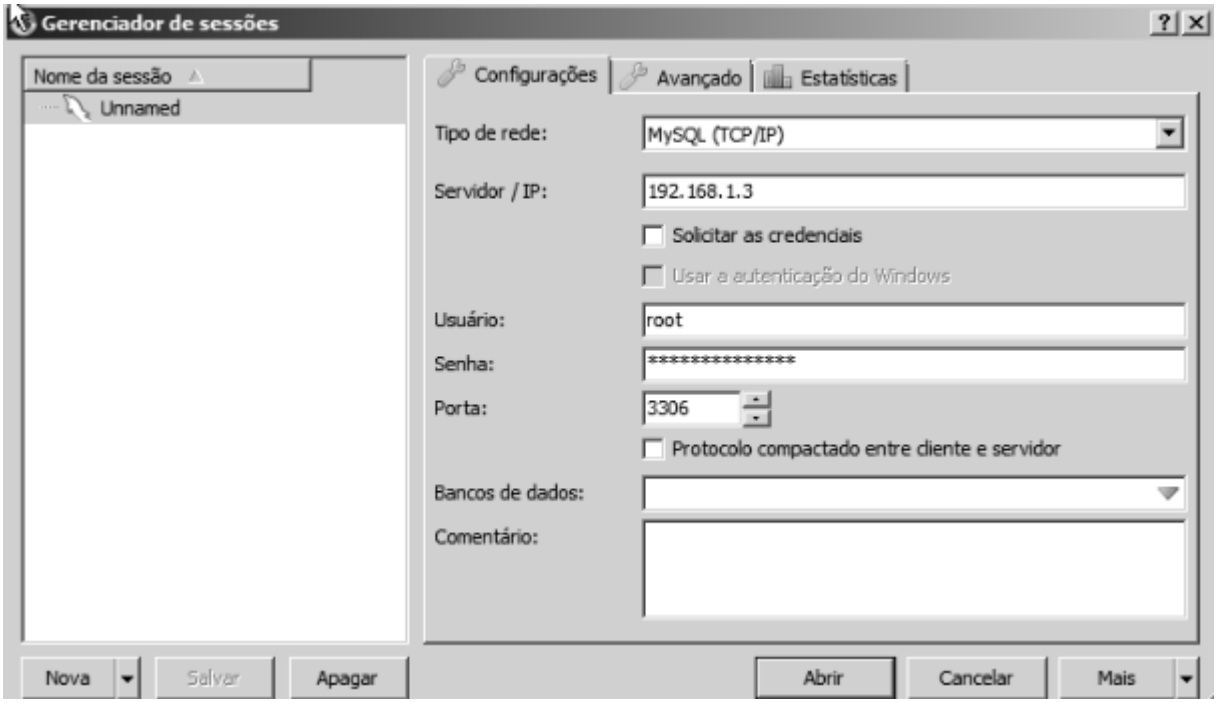
Figura 30 - O comando *ping* sendo capturado pelo atacante



Fonte: Autor

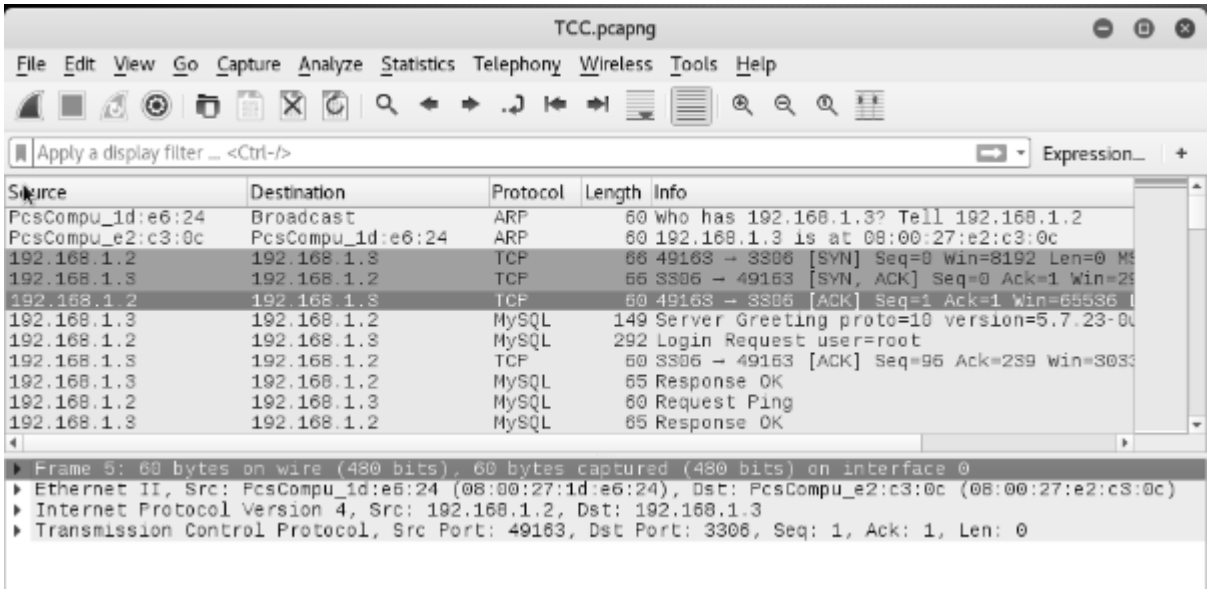
Nesse sentido, quando o cliente requisitar uma conexão para o banco de dados presente no servidor, os dados também serão interceptados pelo atacante, a Figura 31 mostra o cliente iniciando uma conexão para o banco de dados e a Figura 32 mostra a interceptação desta conexão.

Figura 31 - Conexão a ser aberta pelo cliente ao servidor



Fonte: Autor

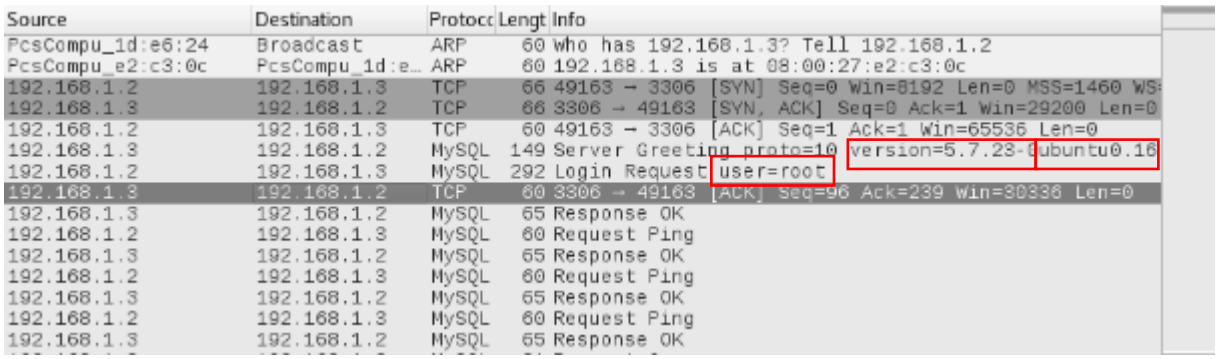
Figura 32 - Conexão *MySQL* interceptada pelo atacante



Fonte: Autor

Com a conexão interceptada, o atacante pode visualizar os dados capturados a procura de informações sensíveis, como os demonstrados na Figura 33, onde pode-se observar a versão do banco de dados, o sistema operacional em execução e o mais interessante, o usuário em que foi feita a autenticação, ou seja, o atacante já teria neste caso, 50% das informações necessárias para a invasão ao banco de dados.

Figura 33 - Informações extraídas da conexão

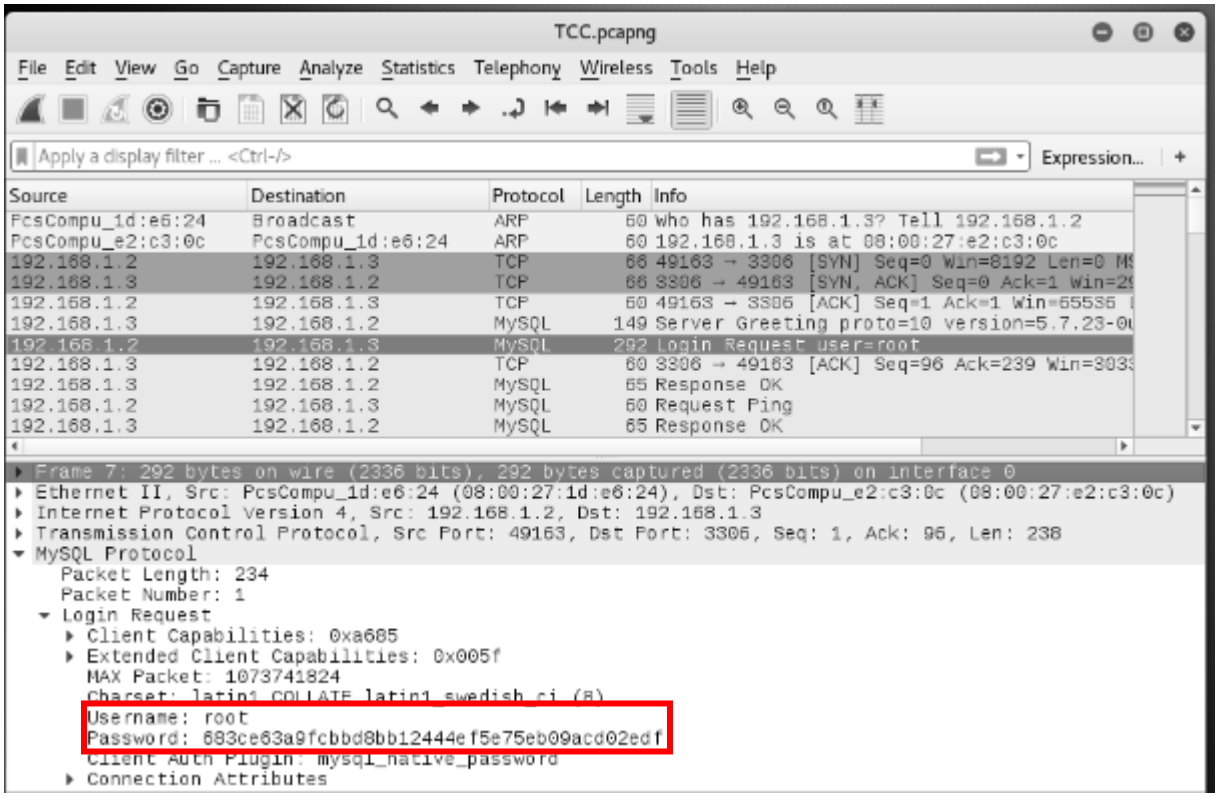


Fonte: Autor

O *Wireshark* é capaz também de inspecionar o conteúdo dos dados capturados através da janela de detalhes do pacote. Com o pacote de requisição de *login* selecionado pode-se obter mais informações a respeito do mesmo. Como

ilustra a Figura 34, foi capturado não somente o usuário utilizado para *login*, mas também sua senha.

Figura 34 – Destaque do pacote mostrando usuário e senha criptografada do banco de dados



Fonte: Autor

A senha capturada está criptografada, ou seja, foi aplicado um algoritmo matemático que faz com que a senha passada pelo cliente, para se conectar ao banco de dados, se transforme em uma sequência de letras e números totalmente diferente da original, justamente para dificultar o acesso não autorizado. Esta forma como a senha é apresentada é conhecida como *hash.* Dependendo do sistema implementado o algoritmo usado será diferente, no caso do protocolo *MySQL* ele se trata do *SHA-1*. De acordo com o manual do *MySQL* (2018) a maneira como é feita a autenticação da senha é mostrado pela Figura 35, quando, de maneira simples, aplica-se o algoritmo duas vezes à senha utilizada para gerar um *hash*, este, por sua vez, será aplicado ao algoritmo junto com uma sequência randômica de caracteres gerado pelo servidor e, por fim, será feita uma operação lógica XOR com o próprio *hash* gerado da senha.

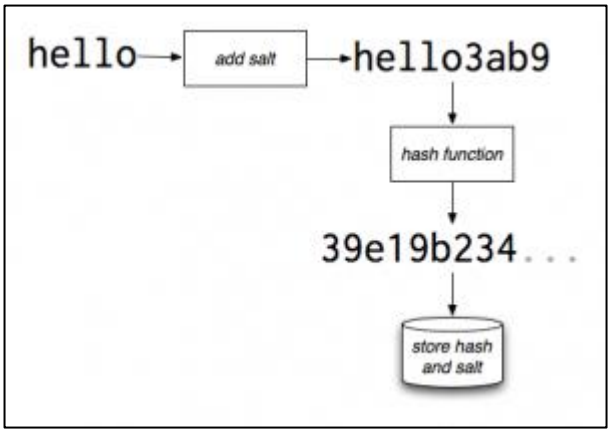
Figura 35 - Algoritmo usado pelo *MySQL* para verificação da senha

```
SHA1( password ) XOR SHA1( "20-bytes random data from server" <concat> SHA1( SHA1( password ) ) )
```

Fonte: *MySQL Internals Manual* (2018)

Os dados gerados randomicamente pelo servidor, como mostrado na Figura 35, são denominados *salt*. Segundo Thadani (2012) o *salt* é uma sequência de caracteres que será incorporado à senha antes do processo de *hash* tornando-a, assim, mais segura. A Figura 36 demonstra este processo, onde o *salt* (3ab9) foi adicionado a senha (*hello*) e então gerado o *hash* (39e19b234...).

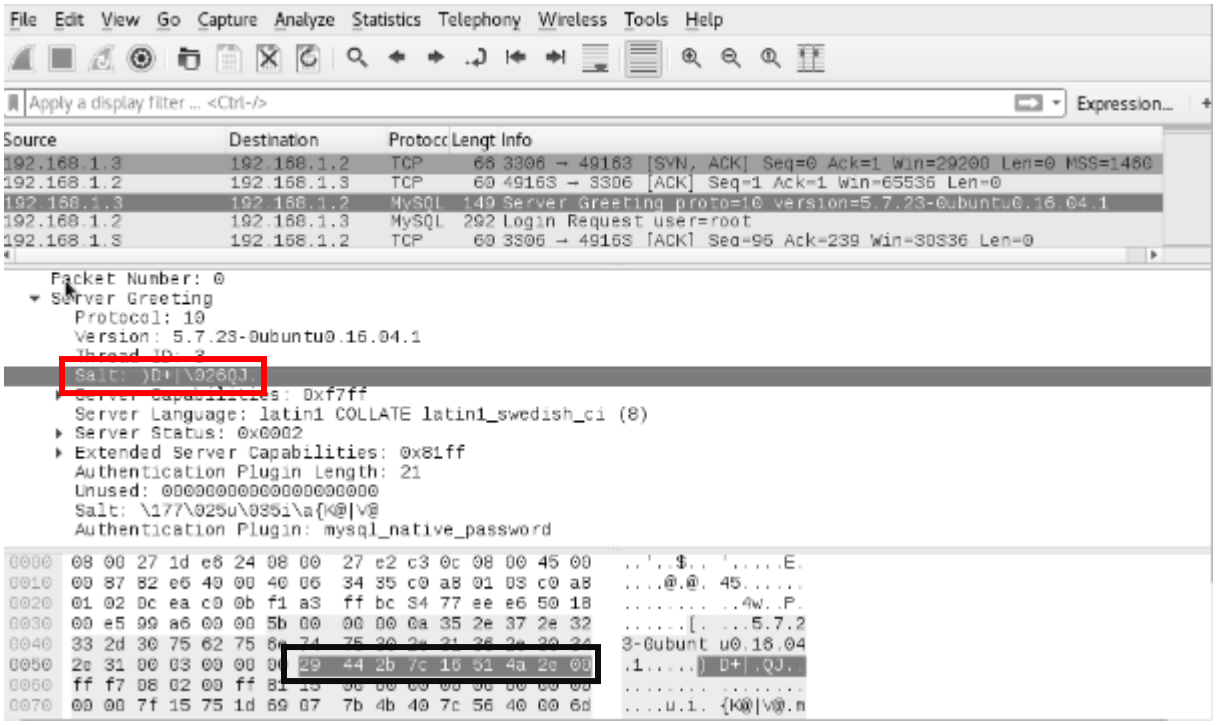
Figura 36 - Processo de adição do Salt a uma senha



Fonte: Thadani (2012)

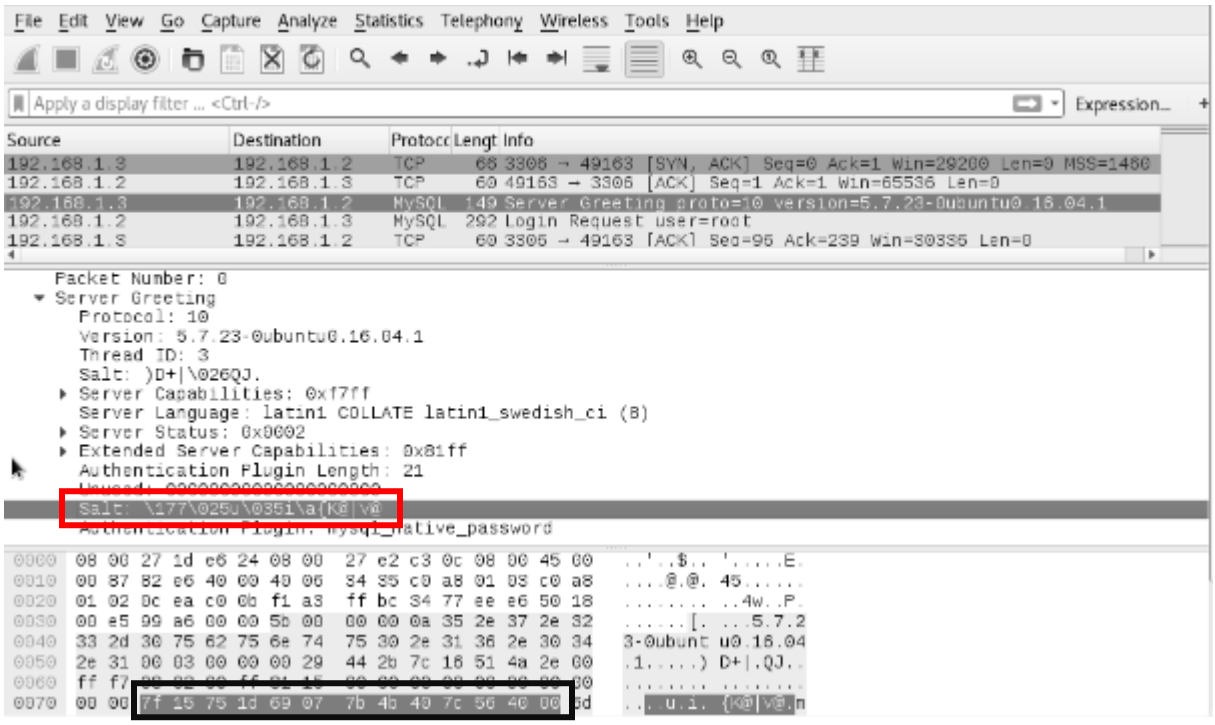
Deste modo, o *Wireshark* também foi capaz de capturar o *salt* gerado para adicionar a senha, como mostra a Figura 37 e 38, fazendo com que a senha em texto puro, seja o único elemento dentro do algoritmo de criptografia do MySQL que não é conhecido e é onde será concentrado o ataque de força bruta.

Figura 37- Primeira parte do salt e sua representação em bytes



Fonte: Autor

Figura 38 - Segunda parte do salt, bem como sua representação em bytes



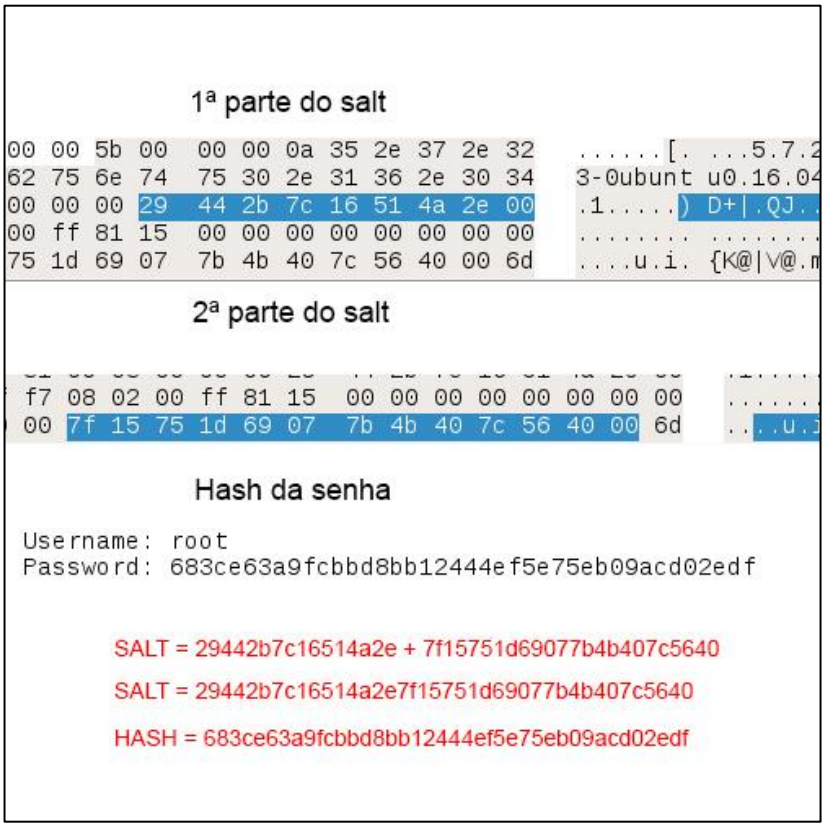
Fonte: Autor

A Figura 37 e 38 mostram também, na parte inferior, a representação do *salt* em bytes. O que será necessário mais adiante para o ataque de força bruta pelo *Hashcat*.

7.2 EXECUÇÃO DO ATAQUE

A partir dos dados capturados pode-se tirar as seguintes informações, demonstradas na Figura 39, o *salt* gerado será a concatenação entre a primeira e segunda parte, o *hash* utilizado será o mesmo que foi capturado pelo *Wireshark* (*FOLLOW THE WHITE RABBIT*, 2016).

Figura 39 - Organização das informações coletadas



Fonte: Autor

Com essas informações devidamente organizadas, basta introduzi-las em um arquivo de texto que será lido pelo programa *Hashcat*.

O *Hashcat* possui diversos tipos de algoritmos que podem ser usados para um ataque de força bruta, dentre eles o MySQL, assim como mostra a Figura 40.

Figura 40 - Exemplos de algoritmos usados pelo *Hashcat*

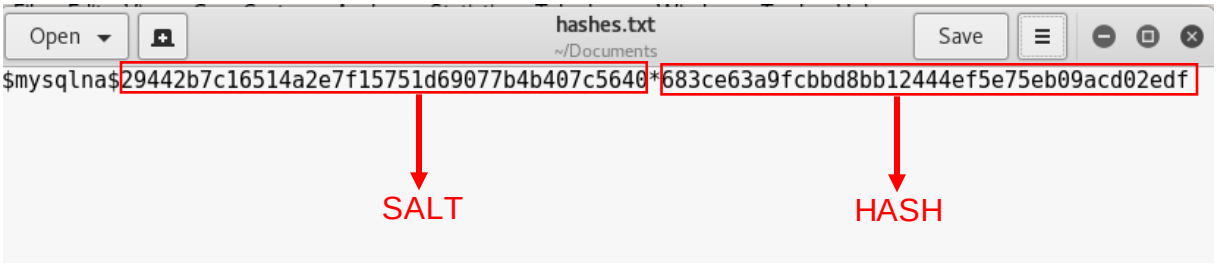
11100	PostgreSQL CRAM (MD5)	$postgres$postgres*f0784ea5*2091bb7d4725d1ca85e8de6ec349baf6
11200	MySQL CRAM (SHA1)	$mysqlna$1c24ab8d0ee94d70ab1f2e814d8f0948a14d10b9*437e93572f18ae44d9e779160c2505271f85821d
11300	Bitcoin/Litecoin wallet.dat	$bitcoin$96$d011a1b6a8d675b7a36d0cd2efaca32a9f8dc1d57d6d01a58399ea04e703e8bbb44899039326f7a(
11400	SIP digest authentication (MD5)	sip*192.168.100.100*192.168.100.121*username*asterisk*REGISTER*sip*192.168.100.121**2b01df0b*
11500	CRC32 [5]	c762de4a:00000000

Fonte: (*Hashcat*, [201-?])

A primeira coluna corresponde ao modo do *hash*, para especificação dentro do programa, a segunda coluna corresponde ao nome do *hash* e a terceira, um exemplo de como deve ser passado o mesmo ao programa para leitura e interpretação.

Nesse sentido, para este documento, será utilizado o modo 11200, que corresponde ao *MySQL Challenge-Response Authentication Mechanism* (CRAM), bastando apenas organizar os dados que serão passados. A Figura 41 exemplifica como ficará o arquivo para introdução no programa.

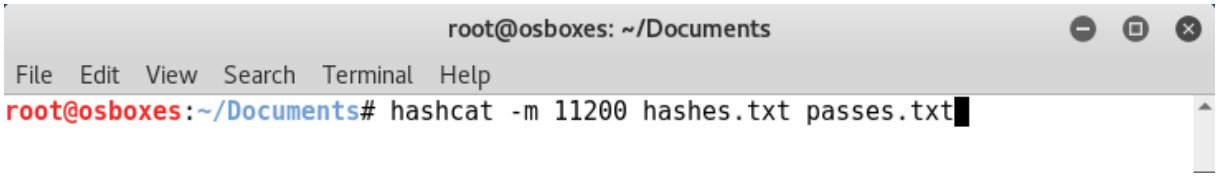
Figura 41- Arquivo configurado com o *salt* e o *hash* usados para a força bruta



Fonte: Autor

Com o arquivo devidamente configurado, o próximo passo é executar o *Hashcat* com os parâmetros necessários para realizar o ataque, a Figura 42 mostra o comando usado, em seguida é dada uma breve explicação de cada parâmetro.

Figura 42 - Comando usado para executar o Hashcat

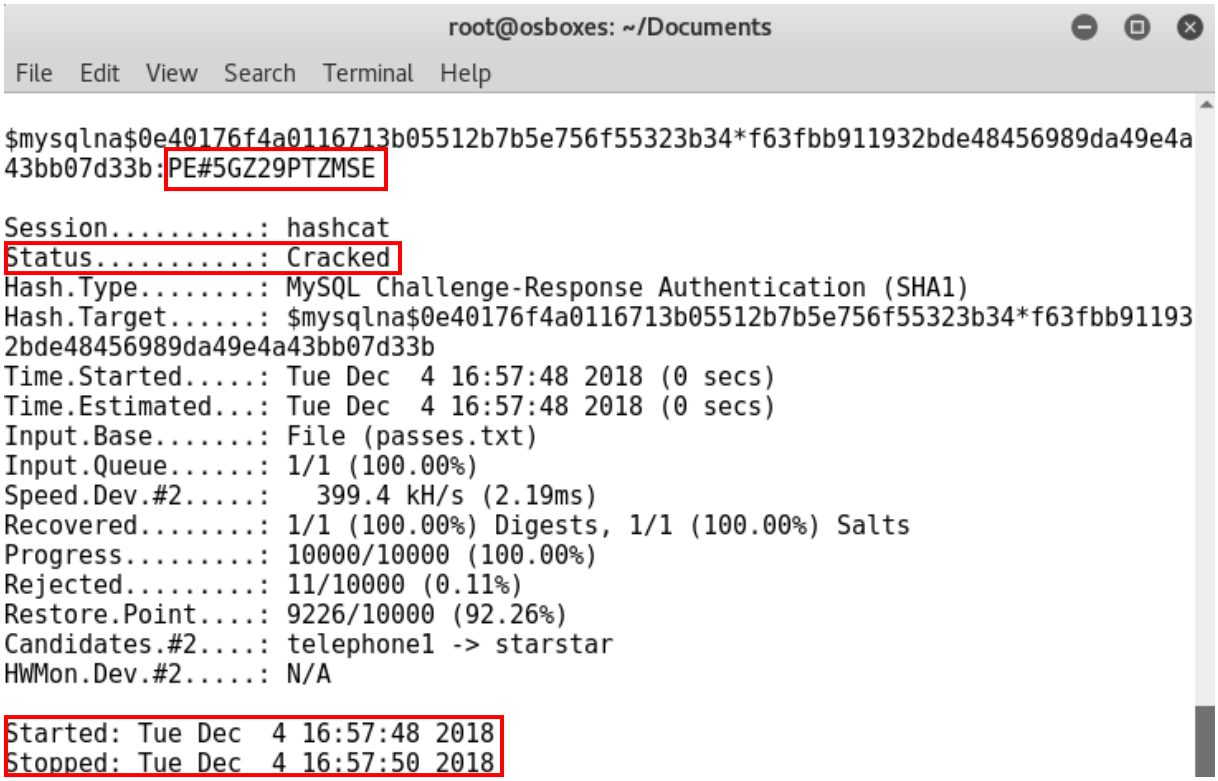


Fonte: Autor

Primeiramente deve-se iniciar o programa com o comando "*hashcat*", o parâmetro "-m" diz o modo em que o ele irá trabalhar, como dito anteriormente 11200 corresponde a *hashes* do tipo MySQL, "hashes.txt" corresponde ao arquivo configurado previamente e por fim, "passes.txt" corresponde a uma lista de possíveis senhas que será usada para realizar o ataque do tipo de dicionário. Vale lembrar que, estas listas estão disponíveis gratuitamente em diversos *sites* relacionados à segurança e auditoria de senhas, muitos deles mantém estas listas atualizadas com diversos tipos de senhas que vão sendo descobertas a cada dia.

Nesse sentido, ao executar o comando, basta aguardar o término de sua execução. Este pode ser um processo demorado, dependendo da capacidade do computador em que o *Hashcat* está sendo executado e do tamanho da lista de senhas passada. Para este teste foi utilizado uma lista contendo 10000 senhas, extraídas no período de 2017 mantidas por Haddix e Meissler (2012). A Figura 43 mostra a saída do programa e em seguida é feito uma breve explicação sobre ela.

Figura 43 - Saída do programa Hashcat, indicando que a senha foi descoberta com sucesso



Fonte: Autor

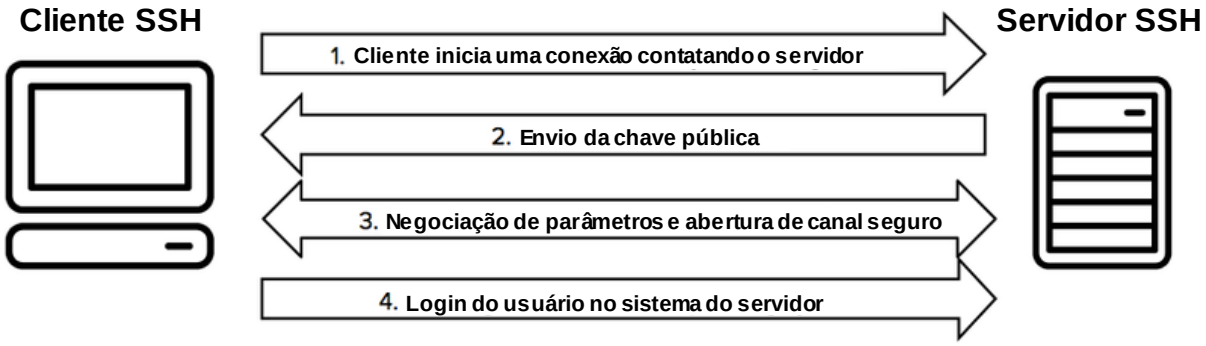
A Figura 43 demonstra que a senha foi descoberta com sucesso e foi mostrada na tela ao lado dos *hashes* usado no arquivo. Outro dado interessante e que vale a pena ser mencionado é o tempo levado para descobrir a senha, cerca de 2 segundos, um tempo expressivamente menor do que os estipulados pelos serviços de verificação de força da senha usados neste documento e também o fato desse tipo de ataque ser totalmente passivo, ou seja, não gerará nenhum ruído na rede onde estará o alvo e dificilmente será detectado.

No próximo capítulo será apresentado uma maneira de tornar essa conexão segura a fim de evitar que um atacante possa interceptar a conexão e realizar o ataque.

8 EXECUÇÃO DO ATAQUE EM CENÁRIO PROTEGIDO

Para demonstração de uma conexão segura será utilizado o protocolo SSH (*Secure Shell*). Segundo Lewis (2001, p.2), este protocolo é comumente usado em servidores de aplicação para proteger dados em transmissão entre dispositivos. Diferentemente de outros protocolos como o Telnet, o SSH utiliza criptografia para conexão e autenticação, a Figura 44 exemplifica o funcionamento deste protocolo.

Figura 44 - Exemplificação do funcionamento do SSH



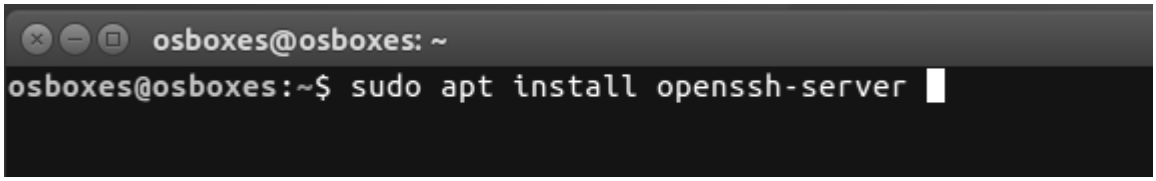
Fonte: Adaptado de SSH Protocol (2018)

Assim como explica Ellingwood (2014), este protocolo funciona em um modelo de cliente-servidor, em que o cliente inicia o processo com o servidor para se estabelecer uma conexão segura e verificar sua identidade. Uma sessão SSH possui duas etapas, a primeira é onde será acordado entre os dispositivos o estabelecimento de uma conexão segura entre eles para proteger as comunicações seguintes e a segunda para autenticar o usuário e verificar se o mesmo possui acesso ao servidor.

8.1 CONFIGURAÇÃO SSH NO SERVIDOR

No servidor será necessário instalar o programa *OpenSSH Server* que será responsável por permitir conexões do tipo SSH. De acordo com a documentação oficial do *Ubuntu* [201-?], a instalação é bem simples e pode ser feita com o comando “sudo apt install openssh-server”, como mostrada na Figura 45.

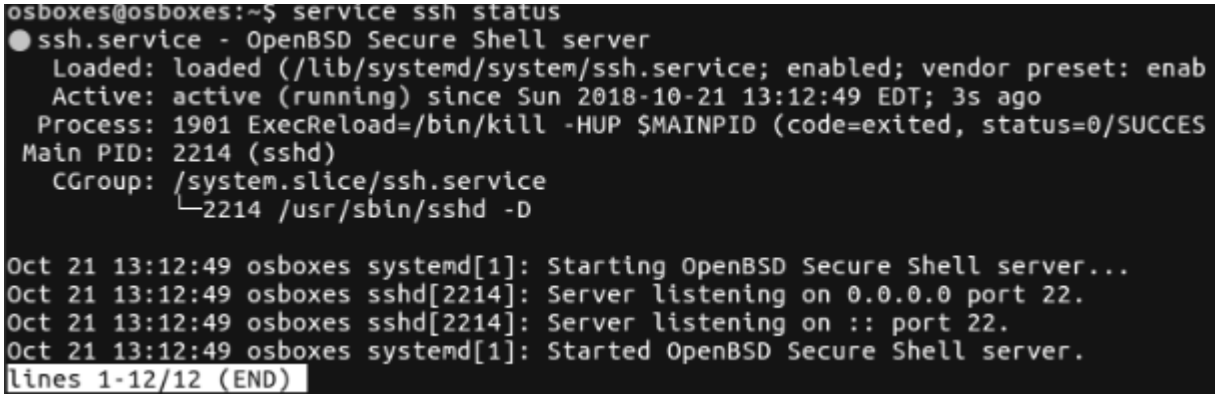
Figura 45 - Comando de instalação do servidor SSH



Fonte: Autor

Após sua instalação, o mesmo já estará em funcionamento, como mostrado na Figura 46.

Figura 46 - Servidor SSH ativo



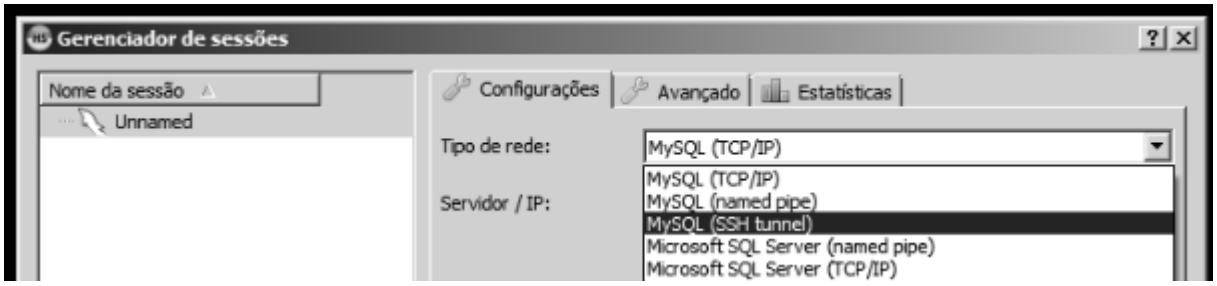
Fonte: Autor

8.2 CONFIGURAÇÃO SSH NO CLIENTE

Para o cliente também deverá ser baixado um programa para o gerenciamento da conexão SSH, além de outras configurações que serão detalhadas a seguir.

Ao utilizar o programa *HeidiSQL*, é possível configurar o tipo de conexão que se deseja com o banco de dados, a Figura 47 mostra algumas opções disponíveis, dentre elas, o SSH.

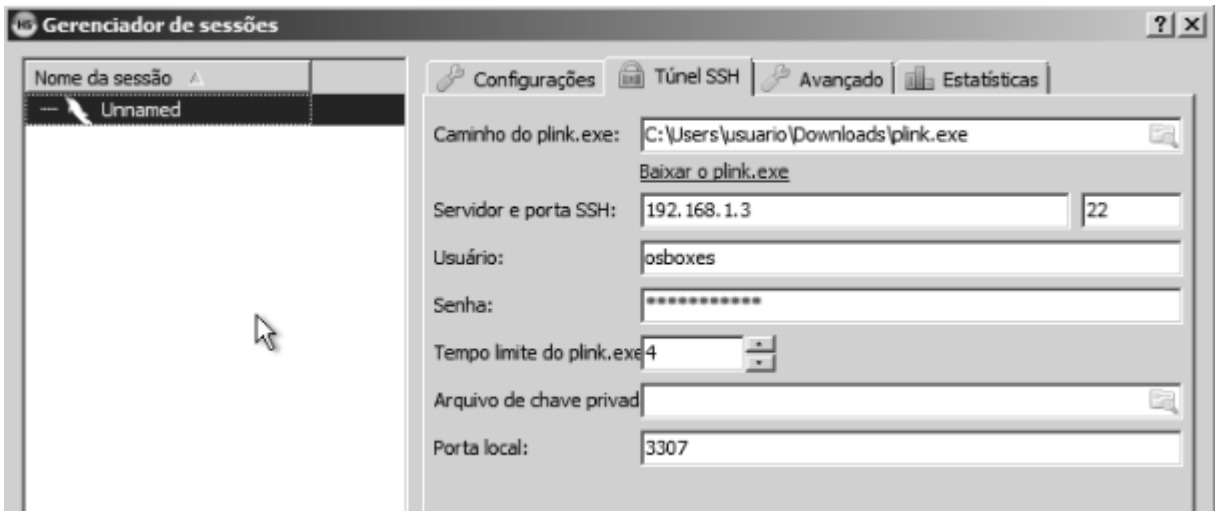
Figura 47- Opções de conexão do programa *HeidiSQL*



Fonte: Autor

Selecionando a opção para o SSH, uma nova aba de configuração será habilitada no programa, chamada Túnel SSH, a Figura 48 mostra essa aba.

Figura 48 - Aba com configurações para conexão SSH



Fonte: Autor

Para a configuração correta, segundo Van de Merwe (2017), deve-se fazer o download o utilitário *plink*, que será o programa que permitirá a conexão via SSH para o servidor. O próprio *HeidiSQL*, disponibiliza o *link* para download, ou poderá ser adquirido pelo site oficial da ferramenta em https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html.

Basta preencher o caminho para o arquivo baixado na tela de configuração, junto com o endereço IP do servidor (192.168.1.3), a porta em que o SSH estará operando (normalmente porta 22) e um usuário e senha para acesso ao servidor. Vale ressaltar que o usuário e senha para conexão, não são aqueles referente ao

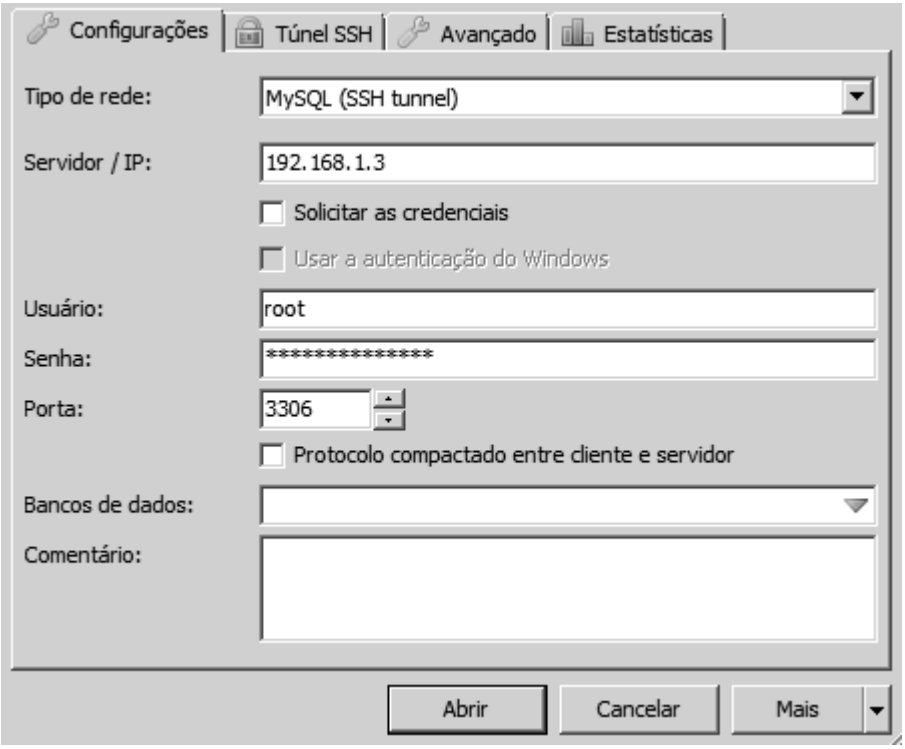
banco de dados e sim aqueles que estão presentes no sistema operacional do servidor.

8.3 TENTATIVA DE EXECUÇÃO DO ATAQUE

Com o cliente e servidor devidamente configurados, será repetido o cenário apresentado no capítulo 6, onde o atacante irá tentar interceptar os dados da conexão.

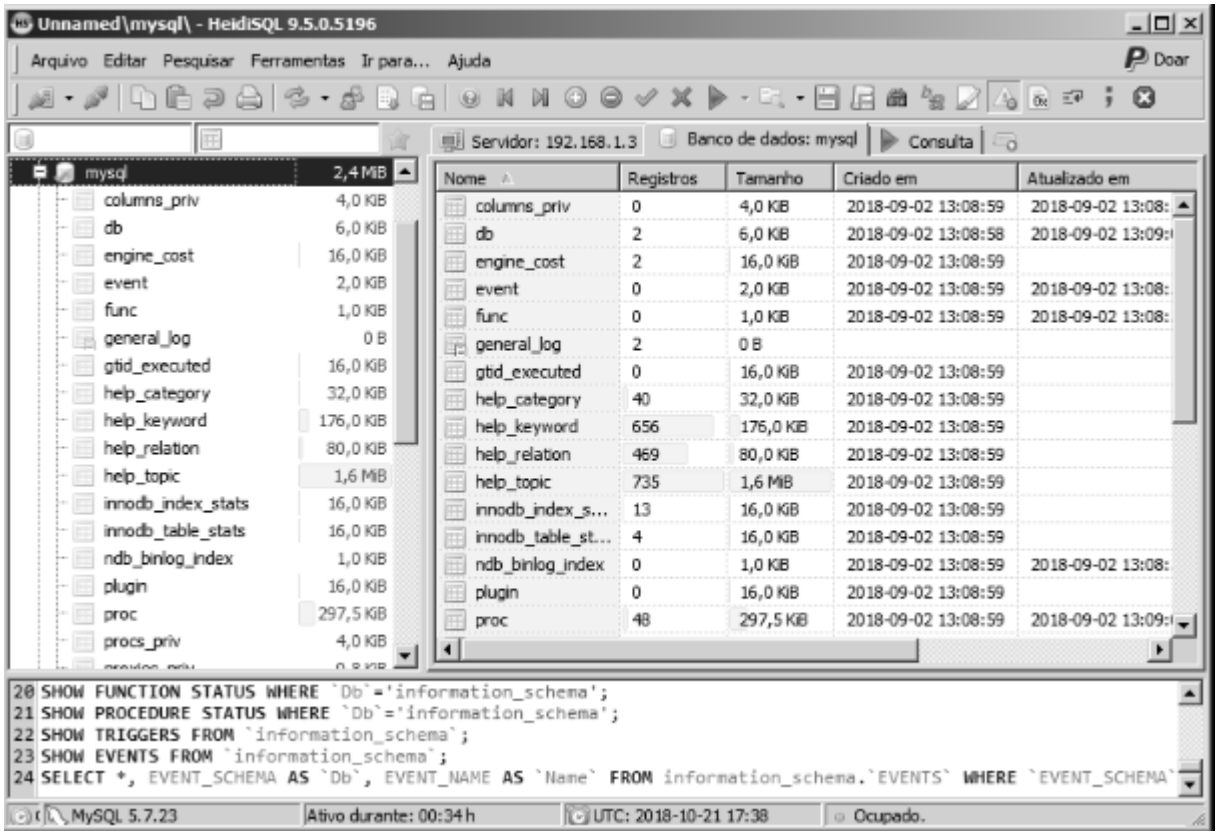
A Figura 49 e 50, mostram as configurações prontas para a conexão e a própria conexão bem sucedida do cliente com o servidor.

Figura 49 - Conexão pronta para ser realizada



Fonte: Autor

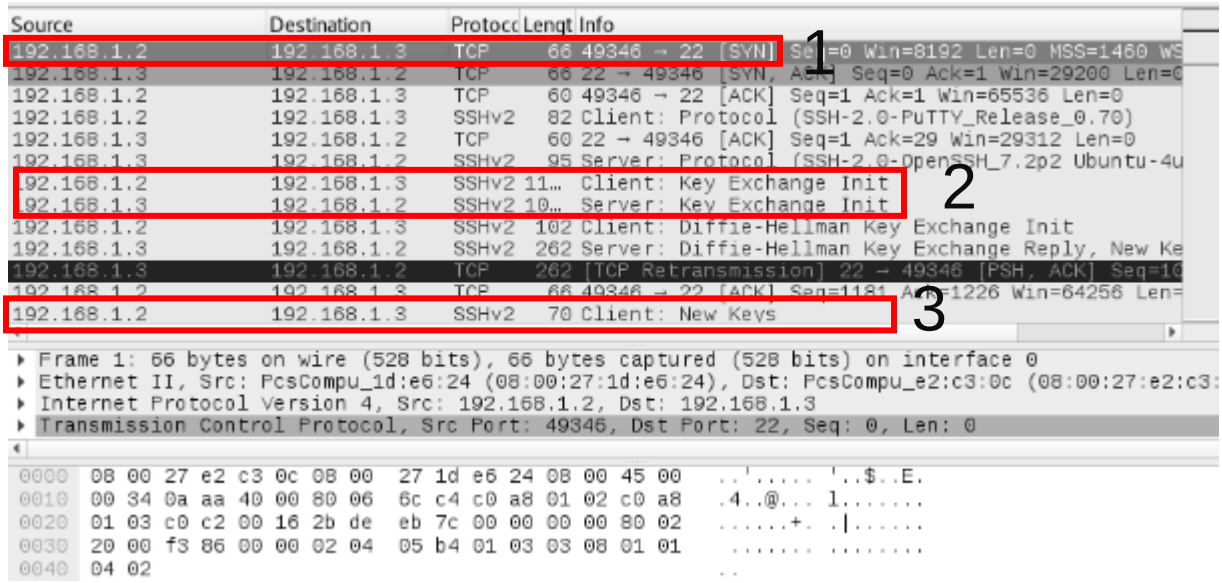
Figura 50 - Conexão bem sucedida, abrindo o banco de dados



Fonte: Autor

No lado do atacante, utilizando o *Wireshark*, pode-se perceber que o tipo de conexão mudou totalmente, alguns pontos importantes são demonstrados na Figura 51.

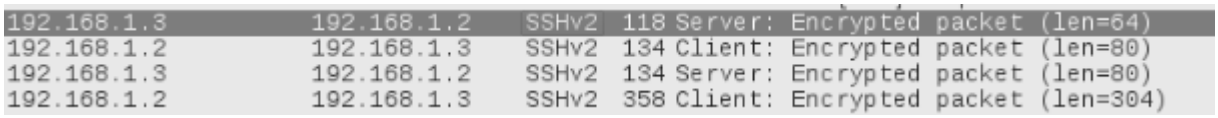
Figura 51 - Dados do SSH capturados pelo *Wireshark*



Fonte: Autor

Destacado pelo número um, tem-se o início da conexão pelo cliente e em dois, tem-se a troca de chaves sendo iniciada entre o cliente e o servidor e, em três, a troca bem sucedida. A partir deste ponto na conexão, como mostra a Figura 52, qualquer interação entre o cliente e o servidor aparecerá como encriptado para o atacante.

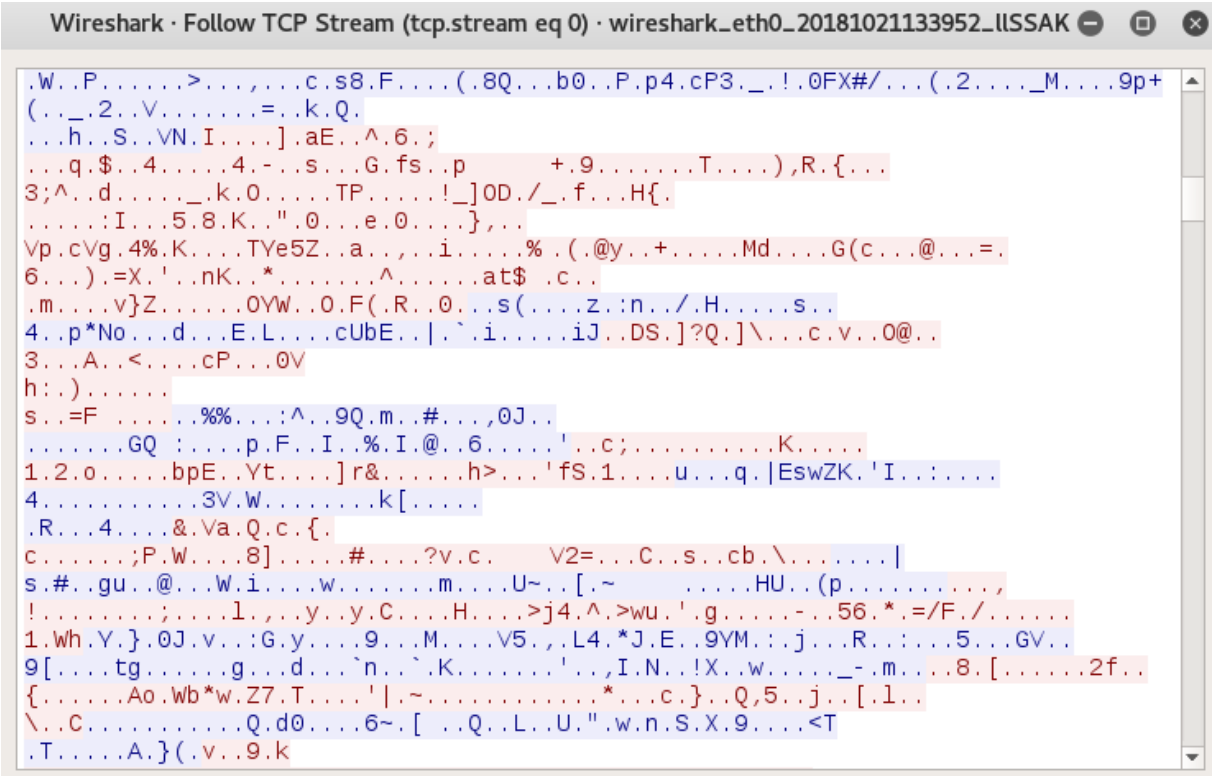
Figura 52 - Pacotes encriptados no *Wireshark*



Fonte: Autor

Qualquer tentativa de observar os dados resultará em diversos caracteres sem sentido, que não possuem correlação nenhuma e, portanto, são ilegíveis para o atacante. A Figura 53 mostra os dados encriptados.

Figura 53 - Detalhamento dos pacotes encriptados



Fonte: Autor

As informações obtidas anteriormente, como versão do banco de dados, usuário, *hash* da senha e comandos executados no banco, ficam agora criptografados, impossibilitando o atacante de continuar e realizar a força bruta para conseguir o acesso ao servidor, tornando assim, a conexão segura entre os dispositivos.

9 CONSIDERAÇÕES FINAIS

A partir da apresentação e análise dos dados, observa-se que, conexões inseguras entre dispositivos podem ser facilmente interceptadas e consequentemente exploradas por terceiros, podendo resultar no roubo de informações sigilosas.

Primeiramente mostrou-se que, mesmo senhas consideradas fortes podem ser facilmente descobertas com ferramentas e técnicas conhecidas, o que pode aumentar o risco para uma organização, já que diversos tipos de documentação estão acessíveis para qualquer tipo de pessoa.

Mostrou-se também que o tipo de ataque explorado neste documento é considerado passivo do ponto de vista da rede, ou seja, uma vez dentro da rede, dificilmente o atacante seria descoberto capturando os dados, já que este tipo de ataque não gera ruídos dentro da rede. Após a captura destes, o trabalho em se adivinhar a senha ficará localmente na máquina do atacante, também não gerando ruído nenhum.

Nesse sentido, a aplicação de criptografia forte através do protocolo SSH para conexão entre dispositivos em uma rede se provou eficiente para mitigar o cenário explorado neste documento, onde o atacante “ouviria” a conexão para explorá-la, resultando, assim, em uma conexão confiável onde os dados podem trafegar com segurança da origem ao destino, sem serem interceptados por terceiros.

Vale ressaltar também que, na área da segurança da informação, novas ferramentas e técnicas para invasão vão sendo descobertas a cada dia, as medidas preventivas descritas aqui podem um dia não serem mais úteis, cabe as empresas se manterem atualizadas técnica e humanamente nesse quesito para evitarem o roubo de suas informações, que, se exploradas com sucesso, podem causar prejuízos milionários.

REFERÊNCIAS

ANLEY, Chris. **Hackproofing MySQL**. 2004. Disponível em <https://www.nccgroup.trust/globalassets/our-research/uk/whitepapers/hackproofing_mysql.pdf>. Acesso em abril de 2018.

CANSIAN, Adriano Mauro. **Desenvolvimento de um sistema adaptativo de detecção de intrusos em redes de computadores**. 1997. Disponível em: <http://www.teses.usp.br/teses/disponiveis/76/76132/tde-25032015-111336/publico/AdrianoCansianD.pdf>. Acesso em fevereiro de 2018.

CZAGAN, Dawid. **Online Dictionary Attack with Hydra**. 2013. Disponível em: <https://resources.infosecinstitute.com/online-dictionary-attack-with-hydra/>. Acesso em julho de 2018.

DONOHUE, Brian. **Hash: o que são e como funcionam**. 2014. Disponível em: <https://www.kaspersky.com.br/blog/hash-o-que-sao-e-como-funcionam/2773/>. Acesso em maio de 2018.

ELLINGWOOD, Justin. **Understanding the SSH Encryption and Connection Process**. 2014. Disponível em: <https://www.digitalocean.com/community/tutorials/understanding-the-ssh-encryption-and-connection-process>. Acesso em setembro de 2018.

ELMASRI, Ramez; NAVATHE, Shamkant B. **Sistemas de Banco de Dados**. 2005. Disponível em: <http://www.rclick.com.br/prime/BD/Sistema_de_banco_de_dados_Navathe.pdf>. Acesso em março de 2018.

FOLLOW THE WHITE RABBIT. **Crackeando credenciales de MySql obtenidos por un MITM like a Sir!** 2016. Disponível em: <https://www.fwhibbit.es/crackeando-credenciales-de-mysql-obtenidos-por-un-mitm-like-a-sir>. Acesso em agosto de 2018.

HASHCAT. **Hashcat - Advanced password recovery**. [201-?]. Disponível em: <https://hashcat.net/hashcat/>. Acesso em maio de 2018.

HEIDISQL WEBSITE. **HeidiSQL - MySQL, MSSQL and PostgreSQL made easy**. 2018. Disponível em: <https://www.heidisql.com/>. Acesso em julho de 2018.

HOW SECURE IS MY PASSWORD. **How Secure Is My Password?** 2016. Disponível em: <https://howsecureismypassword.net/>. Acesso em junho de 2018.

KASPERSKY LAB. **Kaspersky Lab: Secure Password Check**. 2018. Disponível em: <https://password.kaspersky.com/br/>. Acesso em junho de 2018.

KUROSE, James F. **Redes de computadores e a internet uma abordagem top-down**, São Paulo: Pearson Education. 2013.

LASTPASS. LastPass - **How Secure is Your Password?** 2018. Disponível em: <https://lastpass.com/howsecure.php>. Acesso em junho de 2018.

LEWIS, Shawn. **A Discussion of SSH Secure Shell**. 2001. Disponível em: <https://www.sans.org/reading-room/whitepapers/vpns/paper/729>. Acesso em setembro de 2018.

MAYA, Alcides. **O que são redes de computadores?** 2016. Disponível em: <http://www.alcidesmaya.com.br/blog/o-que-sao-redes-de-computadores/>. Acesso em março de 2018.

MEYER, Maximiliano. **O que é P2P e como ela funciona?** 2015. Disponível em: <https://www.oficinadanet.com.br/post/14046-o-que-e-p2p-e-como-ela-funciona>. Acesso em março de 2018.

MIESSLER, Daniel; HADDIX, Jason. **GitHub - danielmiessler/SecLists: SecLists is the security tester's companion. It's a collection of multiple types of lists used during security assessments, collected in one place**. 2012. Disponível em <https://github.com/danielmiessler/SecLists/tree/master/Passwords>. Acesso em agosto de 2018.

MORISSEY, Joe. **Three Pillars of Information Security**. 2010. Disponível em: <http://www.infosecisland.com/blogview/4504-Three-Pillars-of-Information-Security.html>. Acesso em abril de 2018.

MULLER, Nicolas. **Internet, intranet e extranet o que são, e quais as diferenças?** 2011. Disponível em: <https://www.oficinadanet.com.br/artigo/1276/internet_intranet_e_extranet_o_que_sao_e_quais_as_diferencas>. Acesso em março de 2018.

MYSQL INTERNALS MANUAL. **MySQL :: MySQL Internals Manual :: 14.3.3. Secure Password Authentication**. 2018. Disponível em: <https://dev.mysql.com/doc/internals/en/secure-password-authentication.html#packet-Authentication::Native41>. Acesso em agosto de 2018.

O'REILLY, Tim. **Lessons from open source software development**. Disponível em: <http://faculty.salisbury.edu/~xswang/Research/Papers/SERelated/OpenSource/p32-o_reilly.pdf>. Acesso em abril de 2018.

PETRACIOLI, Fernando. **Sabe o que são sniffing e wardriving?** 2008. Disponível em: <http://pcworld.com.br/dicas/2008/02/27/sabe-o-que-e-sniffing-e-wardriving/>. Acesso em julho de 2018.

SANS INSTITUTE. **Implementing Least Privilege at your Enterprise**. 2003. Disponível em: <https://www.sans.org/reading-room/whitepapers/bestprac/implementing-privilege-enterprise-1188>. Acesso em julho de 2018.

SANTOS, Gerson Raimundo dos; GIAVAROTO, Sílvio César Roxo. **Backtrack Linux Auditoria e teste de invasão em redes de computadores**. 2013. Disponível em: <http://site.livrariacultura.com.br/imagem/capitulo/30757884.pdf>. Acesso em fevereiro de 2018

SECTOOLS. **Top 125 Network Security Tools**. [201-?]. Disponível em: <https://sectools.org/>. Acesso em abril de 2018.

SPORCK, Lauren. **11 of the Largest Data Breaches of All Time (Updated)**. 2017. Disponível em: <https://www.opswat.com/blog/11-largest-data-breaches-all-time-updated>. Acesso em agosto de 2018.

SSH PROTOCOL. **SSH Protocol – Secure Remote Login and File Transfer | SSH.COM**. 2018. Disponível em: <https://www.ssh.com/ssh/protocol/>. Acesso em setembro de 2018.

TANENBAUM, Andrew S. **Redes de Computadores**, São Paulo: Pearson Education, 2011.

THADANI, Rahul. **Password Security: A dash of 'salt' and little of 'hash' to go please!** 2012. Disponível em: <https://blogs.quickheal.com/password-security-a-dash-of-salt-and-little-of-hash-to-go-please/>. Acesso em agosto de 2018.

UBUNTU DOCUMENTATION. **OpenSSH Server**. [201-?]. Disponível em: <https://help.ubuntu.com/lts/serverguide/openssh-server.html.en>. Acesso em outubro de 2018.

VAN DE MERWE, Johan. **How to connect to a MySQL database over a SSH tunnel with HeidiSQL**. 2017. Disponível em: <https://www.enovision.net/mysql-ssh-tunnel-heidisql/>. Acesso em outubro de 2018.

VIRDÓ, HAZEL. **How To Install MySQL on Ubuntu 16.04**. 2016. Disponível em: <https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-16-04>. Acesso em junho de 2018.

WALLEN, Jack. **How to set up MySQL for remote access on Ubuntu Server 16.04**. 2017. Disponível em: <https://www.techrepublic.com/article/how-to-set-up-mysql-for-remote-access-on-ubuntu-server-16-04/>. Acesso em junho de 2018.

WE ARE SOCIAL, HOOTSUITE. **Digital in 2018: World's internet users pass the 4 billion mark - We Are Social**. 2018. Disponível em: <https://wearesocial.com/blog/2018/01/global-digital-report-2018>. Acesso em fevereiro de 2018.

WIRESHARK USER'S GUIDE. **Wireshark User's Guide Version 2.9.0**. 2018. Disponível em: <https://www.wireshark.org/download/docs/user-guide.pdf>. Acesso em maio de 2018.