UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

ESCOLA DE INFORMÁTICA APLICADA

Ataque de Homem do Meio em Aplicações de Realidade Virtual

Daniel Quintana de Andrade

Gabriel Castrillon Silva dos Santos

Orientador

Sidney Lucena

RIO DE JANEIRO, RJ – BRASIL

JULHO DE 2018

Catalogação informatizada pelo(a) autor(a)

A553 Andrade, Daniel Quintana de
Ataque de Homem do Meio em Aplicações de Realidade Virtual / Daniel Quintana de Andrade. -- Rio de Janeiro, 2018.
56

Orientador: Sidney Cunha de Lucena.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2018.

1. Virtual Reality. 2. Vulnerabilidades. 3. Segurança da Informação. 4. Ataque de Homem do Meio. I. Cunha de Lucena, Sidney, orient. II. Título.

Catalogação informatizada pelo(a) autor(a)

C355 Castrillon, Gabriel
Ataque de Homem do Meio em Aplicações de Realidade Virtual / Gabriel Castrillon. -- Rio de Janeiro, 2018.
57 p

Orientador: Sidney Cunha de Lucena.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2018.

1. Realidade Virtual. 2. Segurança da Informação. 3. Vulnerabilidades. 4. Ataque de Homem do Meio. I. Cunha de Lucena, Sidney, orient. II. Título.

Ataque de Homem do Meio em Aplicações de Realidade Virtual

Daniel Quintana de Andrade
Gabriel Castrillon Silva dos Santos

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovada por:

_____

Sidney Cunha de Lucena (UNIRIO)

_____

Leonardo Luiz Alencastro Rocha (UNIRIO)

_____

Maximiliano M. de Faria (UNIRIO)

RIO DE JANEIRO, RJ – BRASIL.
JULHO DE 2018

Agradecimentos

Daniel Andrade

Agradeço, primeiramente a Deus, por ter me concedido a oportunidade de entrar na Universidade, por ter me acompanhado durante o meu curso e me abençoado com todo o conhecimento que obtive durante o curso.

Agradeço também a minha família, por ser meu suporte, meu apoio, a tudo que precisei, por não desistirem de mim, mesmo quando eu já pensava em desistir e por me cobrarem mais resultados o que me motivou a melhorar e a me formar.

Agradeço a minha namorada por ser meu apoio, principalmente, quando achava que havia me saído mal em alguma prova ou quando achava que iria reprovar em alguma matéria, sempre me fortalecendo e me dando esperança a continuar.

Agradeço aos meus colegas por estarem comigo durante esses mais de quatro anos, sofrendo comigo e se divertindo comigo, pois também foi graças a eles, que me ouviram quando precisava e por toda a ajuda que me deram. Mais que colegas, vou me lembrar deles para a vida toda.

Aos meus professores, muito obrigado por toda a paciência, a vontade que tiveram de me ensinar o necessário para me tornar um bom bacharel, eu só tenho a agradecer pois foram parte importante na minha formação profissional.

Agradecimentos

Gabriel Castrillon

Agradeço, antes de tudo, a Deus. Agradeço por ter me dado forças, por não deixar que eu desanimasse durante esses 5 anos de faculdade.

Agradeço também aos meus pais. Eles foram fundamentais para meu êxito acadêmico. Eles acreditaram em mim, me deram suporte nos momentos mais difíceis. Esse apoio me levou a conseguir uma vaga numa das melhores faculdades do país.

Agradeço aos professores e colegas de curso, todos aqueles que me auxiliaram na caminhada, compartilhando seus conhecimentos.

Agradeço ao meu chefe por ser compreensivo, permitindo que em certos momentos eu pudesse priorizar a faculdade, especialmente durante a execução deste trabalho.

Também é válido agradecer a todos os colegas da área de T.I. que nos auxiliam em fóruns ao redor da internet.

RESUMO

Virtual Reality, conhecido pela abreviação "VR", em português, de Realidade Virtual, não é uma tecnologia nova, mas tem estado em evidência nos últimos anos. Empresas como a Google, Microsoft e Sony têm feito grandes investimentos em pesquisas sobre VR, colocando o assunto em destaque no meio tecnológico.

Sempre que uma ferramenta, ou tecnologia, é difundida e passa a ser amplamente usada, *hackers* procuram formas de explorar possíveis falhas de segurança no sistema, para ganho pessoal, político ou apenas por diversão. A *Internet of Things* (IoT) tem ampliado o campo em que indivíduos mal intencionados têm para atuar e os VRs não ficam de fora.

Também deve-se ressaltar a importância da segurança da informação. Após ataques como o WannaCry, em 2017, o assunto ganhou maior notoriedade, o que chama a atenção de mais pessoas.

Assim sendo, este trabalho tem por objetivo mostrar as possíveis fragilidades de segurança associadas a uma aplicação de VR. Em particular, será demonstrado como é possível interceptar o conteúdo a ser disponibilizado na tela, através de um ataque do tipo *Man in The Middle*, e com isso alterar as imagens mostradas ao usuário.

Palavras-chave: *Virtual Reality*, Vulnerabilidades, Segurança da Informação, Ataque de Homem do Meio.

ABSTRACT

The Virtual Reality, known as “VR”, is not a new technology, but it has been in evidence for the last years. Companies like Google, Microsoft and Sony have made big investments in research about VR, standing out this subject in technological sectors.

Every time a tool or technology becomes widespread and starts to build a user base, hackers search for ways to explore possible vulnerabilities in the systems, aiming for personal or political gain, or just for fun. The Internet of Things (IoT) has widened the field of bad willed people have to act and VR is now a new option for them.

It is also very important to emphasize the value of information security. After attacks like WannaCry in 2017, this kind of topic has gained traction, therefore drawing attention to the matter, all over the world.

The objective of this work is, so, to demonstrate potential security fragilities on a VR application. In particular, it will be shown how to intercept the content to be displayed on the app's screen using a Man in the Middle type of attack, and thus change the images that will be shown to users.

Keywords: Virtual reality, Vulnerability, Information Security, Man in The Middle Attack.

Índice

1. Introdução **12**
1.1 Motivação 12
1.2 Objetivo 13
1.3 Metodologia 13
1.4 Estrutura do Texto 14

2. Fundamentação Teórica **15**
2.1 História da Realidade Virtual 15
2.2 Tipos de Sistemas de Realidade Virtual 16
2.3 Protocolos e Segurança 19
2.3.1 Protocolos 20
2.3.2 Vulnerabilidades Conhecidas 21
2.3.3 Métodos de ataque 22

3. Descrição da Execução do Ataque **25**
3.1 Propostas de Execução do Ataque 25
3.2 Descrição da Arquitetura 26
3.3 A Aplicação 26

4. Ferramentas Adotadas e Tutoriais de Uso **32**
4.1 Ferramentas Utilizadas 32
4.1.1 Ferramentas do Kali Para Execução do Ataque 32
4.2 Requisitos Técnicos 34
4.3 Declaração de Exoneração de Responsabilidade 34
4.4 Arquitetura do Ataque 35
4.5 Tutorial de Ataque 36

5. Resultados **47**
5.1 No Computador: 47
5.2 No Celular: 49
5.3 Limitações e Dificuldades Encontradas 51

6. Conclusão **52**
6.1 Sugestões de Trabalhos Futuros 53

Índice de Figuras

Figura 1	Simulador de cockpits	14
Figura 2	Realidade virtual de projeção	15
Figura 3	Realidade virtual aumentada	16
Figura 4	Diagrama do ataque	19
Figura 5	Primeira tela em funcionamento	23
Figura 6	Código da função RandomlyTeleport	23
Figura 7	Código de requisição do nome da skybox	25
Figura 8	Código da função *start* de VRChangeColor	25
Figura 9	Código da função *start* de ChangeColorTerrain	26
Figura 10	Código de *update* da classe Texture	27
Figura 11	Conexão cliente-servidor normal	31
Figura 12	Conexão cliente-servidor alterada	32
Figura 13	Comando ifconfig na VM Kali	32
Figura 14	Comando ifconfig no macbook (vítima)	33
Figura 15	Rodando o nmap na subrede	33
Figura 16	Resultado após rodar o nmap	34
Figura 17	Edição *ip_forward*	35
Figura 18	Comandos arpspoof	36
Figura 19	Comando *dnsspoof*	37
Figura 20	Visualizando pacotes com o Wireshark	38
Figura 21	Configurações do Burp Suite	39
Figura 22	Configuração para mudar o destino da requisição automaticamente	40
Figura 23	Configuração para mudar o *host* da requisição automaticamente	40
Figura 24	Requisição interceptada	41
Figura 25	Edição da requisição	42
Figura 26	Site oficial	43
Figura 27	Site falso	44

Figura 28	Comparação entre os resultados	44
Figura 29	Funcionamento normal do aplicativo	45
Figura 30	Cenário *default* do aplicativo	45
Figura 31	Aplicativo hackeado	46

Glossário

API	-	*Application Programming Interface*
AR	-	*Augmented Reality*
ARP	-	*Address Resolution Protocol*
DNS	-	*Domain Name System*
GUI	-	*Graphical User Interface*
MAC	-	*Media Access Control*
MITM	-	*Man In The Middle*
SSL	-	*Secure Sockets Layer*
TLS	-	*Transport Layer Security*
VM	-	*Virtual Machine*
VR	-	*Virtual Reality*
OS	-	*Operational System*
URL	-	*Uniform Resource Locator*
IP	-	*Internet Protocol*

1. Introdução

Atualmente, existem diversas tecnologias de interação humano computador. Alguns dos dispositivos como *smartphones*, *smartwatches* e *smarTVs* já são bastante conhecidos. Porém, nos últimos anos, algumas empresas, como *Google*, *Facebook*, *Sony* e outras criaram o *Global Virtual Reality Association* (GVRA), uma organização sem fins lucrativos com o objetivo de "promover o desenvolvimento responsável e a adoção" da tecnologia ainda nascente, de acordo com *Business Insider Intelligence* [3], e vêm explorando as tecnologias de *Augmented Reality* (*AR*) e a *Virtual Reality* (*VR*), que são o foco deste trabalho.

1.1 Motivação

Cada vez que surge algo novo no mundo tecnológico, a atenção de consumidores, entusiastas, jornalistas e *hackers*, se voltam para tal. Algumas novidades acabam não vingando, o que reduz o interesse de indivíduos mal intencionados. Porém, quando uma tecnologia adquire uma base de usuários sólida e mostra sinais de crescimento, *hackers* passam a ter maior interesse.

Aplicações VR têm crescido ao redor do mundo, tendo uma previsão de grande aumento no número de usuários [1]. Visto isso, pode-se afirmar que a quantidade de indivíduos mal intencionados com interesse em explorar possíveis falhas deve aumentar cada vez mais. A facilidade em se obter acesso a essas aplicações por um *hacker* pode tornar essa ação cada vez mais constante.

A motivação para este trabalho surgiu a partir do momento em que pensamos no quão perigosa e economicamente impactante pode ser a tecnologia VR quando utilizada

sem as devidas providências. Em muitos momentos pode ser benéfica, porém, se houver falhas de segurança, pode ser prejudicial. Portanto, decidimos mostrar que deve existir uma forte segurança para evitar que, tanto usuários como responsáveis pelas aplicações, tenham que lidar com consequências a partir de uma falha.

Por hipótese, podemos utilizar como exemplo o jogo de realidade virtual aumentada *Pokemon GO*. Restaurantes, cafés e pequenos varejistas o utilizam para atrair clientes, propagando-se como *PokeStops* (lugar onde os jogadores podem pegar novos itens e aumentar o seu nível de poder dentro do aplicativo), ou utilizando recursos do aplicativo que atraem *Pokémons* para atrair jogadores ao estabelecimento [2]. Nessa hipótese, a exploração de uma falha de segurança poderia modificar a localização para outro estabelecimento, e o estabelecimento original ficaria prejudicado.

1.2 Objetivo

O objetivo do trabalho é demonstrar, por meio de experimentos, o quão importante é a segurança em aplicações de realidade virtual. Este trabalho propõe demonstrar essa premissa das seguintes maneiras: apontando as falhas dos protocolos, explicando maneiras de explorá-las e expondo os resultados obtidos. Também é um dos objetivos sugerir possíveis soluções para os problemas que serão citados.

1.3 Metodologia

A metodologia adotada pode ser dividida em quatro tópicos:

- Quanto à finalidade: Pesquisa Aplicada, onde se busca explorar um problema específico, nesse caso, demonstrar resultados ao explorar possíveis falhas de segurança quando são realizados ataques do tipo *Man in the Middle* com dispositivos em um ambiente real.
- Quanto aos objetivos: Pesquisa Exploratória, pois, neste caso, tentamos identificar melhor um fato, um possível problema de segurança em aplicações de realidade virtual realizada para *smartphones* e computadores, a pesquisa tem um cunho relevante para o entendimento do real problema.

- Quanto à abordagem: Abordagem Qualitativa, pois não há amostragem em nossa pesquisa, os resultados não são numéricos e sim valorativos.
- Quanto ao método: Método Indutivo, pois fazemos uma observação particular a partir de uma premissa. A premissa de que pode haver a falta de segurança e a observação de quais são as consequências principais dessa premissa. Como a pesquisa é feita em mais de um dispositivo, a indução ocorre para outros dispositivos comumente.

1.4 Estrutura do Texto

Este trabalho está organizado da seguinte forma:

- O Capítulo 2 traz a fundamentação teórica, onde serão mostrados os conceitos básicos de segurança.
- No Capítulo 3, encontram-se a descrição do problema, as propostas para a execução do ataque e alguns dos códigos que foram utilizados para a criação da aplicação.
- O Capítulo 4 apresenta detalhes sobre as ferramentas que serão utilizadas no ataque, além dos tutoriais, tanto para utilização do Google Cardboard quanto para a realização do ataque. Também dispomos da declaração de exoneração de responsabilidade e os requisitos técnicos para a realização do experimento.
- No Capítulo 5 são apresentados os resultados obtidos após a execução do experimento, assim como as limitações do experimento e as consequências do ataque.
- O Capítulo 6 apresenta a conclusão e sugestões para possíveis trabalhos futuros.

2. Fundamentação Teórica

Este capítulo descreve, de forma resumida, os principais conceitos usados durante o trabalho, assim como técnicas e tecnologias correlatas cujos entendimentos são importantes para a compreensão geral das implicações e limitações dos experimentos aqui relatados.

2.1 História da Realidade Virtual

O primeiro conceito de VR visto na história aconteceu em 1838 com o britânico Charles Wheatstone, com óculos estereoscópicos que usava um par de espelhos localizados a 45 graus dos olhos do usuário, cada um refletindo uma imagem localizada ao lado. As imagens causavam um efeito de volume e imersão com as imagens sobrepostas [3].

Em 1929, o americano Edward Link criou o primeiro simulador de voo comercial. Foi completamente eletromecânico e foi chamado de *Link Trainer*. O simulador de voo permitiu um treinamento mais seguro para os pilotos. Então os militares dos EUA compraram 6 desses aparelhos e, na segunda guerra mundial, mais de meio milhão de pilotos o usaram para treinamento e aprimoramento de suas habilidades de voo [3].

Em 1962, o americano Morton Heilig construiu um protótipo chamado Sensorama. Era uma espécie de simulador onde havia um gabinete de teatro de estilo *arcade* que estimulava todos os sentidos. Este dispositivo mecânico apresentava colunas estéreo, um visor 3D estereoscópico, ventoinhas, geradores de odores e uma cadeira vibratória. Heilig pretendia imergir completamente o homem dentro do filme. Ele também criou seis pequenos filmes para esse propósito. Cada um deles foi filmado, produzido e editado por ele mesmo [3].

Após essas primeiras invenções, houveram outras que permitiram à realidade virtual evoluir e progredir até a atualidade, onde diversas empresas como Facebook [4], Microsoft [5] e Sony [6], entre outras, investem em VR.

2.2 Tipos de Sistemas de Realidade Virtual

A VR de Simulação representa o tipo mais antigo de sistema de VR porque se originou com os simuladores de voo desenvolvidos pelos militares americanos depois da Segunda Guerra Mundial [7].

Um sistema de VR de Simulação basicamente imita o interior de um carro, avião ou jato, colocando o participante dentro de uma cabine com controles. Dentro dessa cabine, telas de vídeo e monitores apresentam um mundo virtual que reage aos comandos do usuário. Uma vez que o sistema de VR de Simulação não processa imagens em estéreo, as imagens aparecem de forma bastante rápida [8]. Em alguns sistemas as cabines são montadas sobre plataformas móveis [8], além de dispor de controles com feedback tátil e auditivo [8]. Um sistema de realidade virtual de simulação chamado de *Cockpit* BR [9], criado por brasileiros na *Campus Party* de 2012, pode ser visto na Figura 1.

Figura 1 : Simulador de cockpits - fonte: Tecmundo (2012), disponível em <https://www.tecmundo.com.br/campus-party-brasil-2012/19138-project-cockpit-br-um-simulador-de-voo-incrivel-e-100-nacional.htm>

Jacobson [10], em 1994, diz que a Realidade Artificial de Projeção foi criada na década de 70 por Myron Krueger, caracterizando-a pelo fato do usuário estar fora do mundo virtual, mas podendo se comunicar com personagens ou objetos dentro dele.

Krueger, nesta mesma época, cria um Sistema de VR de Projeção, ao qual denominou *Videoplace* (sala ou lugar de Projeção), que capturava imagens de um ou mais usuários e as projetava numa grande tela que representava um mundo virtual, onde era possível a interação desses usuários uns com os outros ou com objetos. O termo Realidade Virtual de Projeção, criado por Krueger, fora simplesmente para descrever o tipo de ambiente criado pelo seu sistema, que poderia ser utilizado sem a necessidade do participante usar dispositivos de entrada de dados, afirma Jacobson. Apesar de visto em um filme, um exemplo atual de realidade virtual de projeção pode ser visto no filme *Iron Man* (Figura 2), onde o personagem principal interage com uma projeção de armadura.

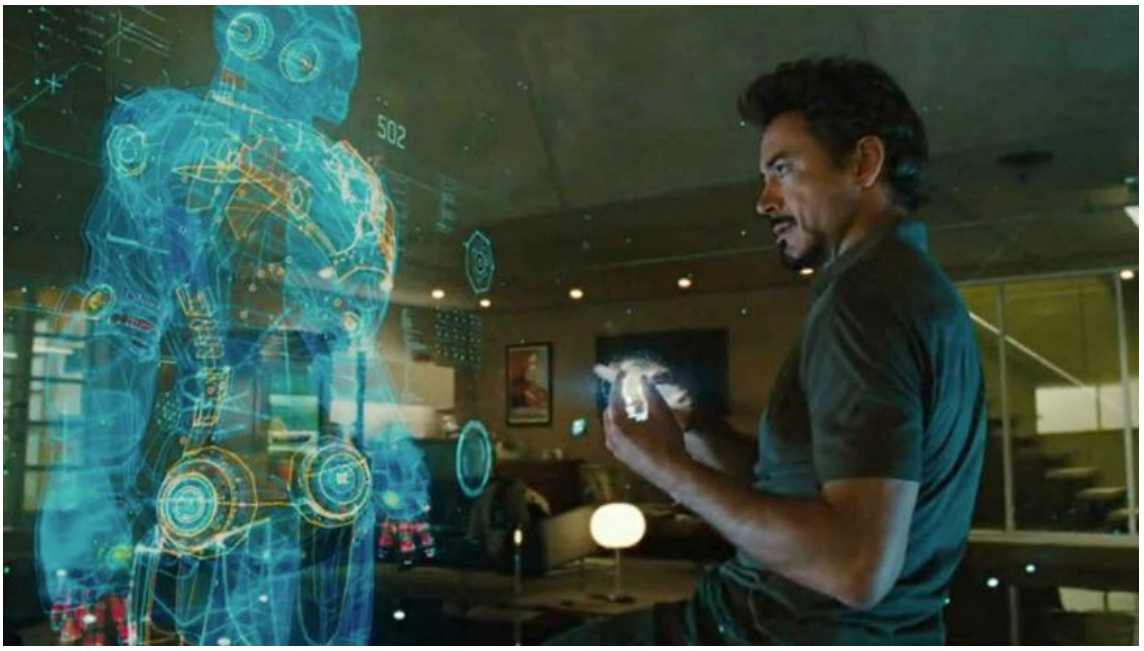

Figura 2 : Realidade virtual de projeção - fonte: Inhabitat (2015), disponível em: <https://inhabitat.com/elon-musk-unveils-his-iron-man-inspired-hand-manipulated-3d-holographic-technology/>

A *Augmented Reality* (AR), que pode ser traduzido como Realidade Aumentada, utiliza dispositivos visuais que permitem interagir com objetos virtuais no mundo real. Esses dispositivos podem ser óculos ou *smartphones*, entre outros. Eles causam um efeito de transparência a esses objetos projetados. O usuário pode ver dados, diagramas, animações e gráficos 3D sem deixar de enxergar o mundo real, tendo informações sobrepostas ao mundo real. Estes *displays* visuais de AR são chamados *Heads Up Displays* (HUDs) por permitirem essa visão através das informações geradas pelo computador. O usuário pode, por exemplo, estar consertando algo e visualizando nos óculos os dados necessários a esta operação [10]. Na Figura 3 podemos ver o jogo Pokémon *GO*, lançado em 2016, que utiliza a AR para permitir que os jogadores capturem *pokémons* no mundo real [11].

Figura 3: Realidade virtual aumentada - fonte: Slashgear (2017), disponível em <https://www.slashgear.com/pokemon-go-ar-plus-apple-arkit-augmented-reality-game-update-20512429/>

2.3 Protocolos e Segurança

Sabemos que seguir boas práticas de segurança é muito importante para reduzir ao máximo as chances de um ataque lograr êxito. No lado do servidor, recomenda-se a utilização de criptografia. Do ponto de vista do usuário, recomenda-se cuidado com quais tipos de aplicações e *sites* são utilizados.

Nesta seção vamos abordar os protocolos envolvidos no experimento que será explicado no Capítulo 4, assim como suas falhas e maneiras de explorá-las.

2.3.1 Protocolos

- HTTP [12]

  É um protocolo utilizado no nível da aplicação. Ele permite a transferência de dados entre redes de computadores, *browsers* e servidores pela *World Wide Web*. Os usos mais comuns desse protocolo são para transferências de páginas HTML, onde cada URL utiliza o prefixo "http://", e também para requisições *web,* disponíveis a diversos tipos de dispositivos. A versão mais utilizada deste protocolo é a 1.1.

- HTTPS [13]

  Este é um protocolo orientado ao envio de mensagens seguras. O HTTPS trabalha em conjunto com o HTTP e suporta um número de métodos para criptografia, como por exemplo o SSL e o TLS. Os *sites* acessados apresentam o prefixo "https://". Apesar disso, essas conexões não são totalmente seguras, sendo possível usar um método para burlar esses protocolos, conhecido como SSL *Strip* [14]. Esse método é frequentemente utilizado na prática de ataques *Man in The Middle*.

- HSTS [15]

  HTTP *Strict Transport Security*, ou HSTS, é um padrão de segurança SSL que força a utilização do HTTPS. Ele impede que o *site* seja executado no caso de não ser possível estabelecer uma conexão segura e, diferentemente do HTTPS, não permite que o usuário adicione exceções de protocolo.

  O método implementado por esse protocolo é uma das maneiras de solucionar o problema do HTTPS, como descrito acima.

- ARP [16]

  O ARP é um protocolo de resolução de endereço que permite uma máquina conhecer o endereço físico de uma placa de rede de outra máquina dentro da mesma rede local, onde esta placa de rede está atrelada a um dado

endereço lógico, o IP. Cada máquina ligada na rede possui um número único de identificação de 48 *bits* que é esse endereço físico, conhecido como MAC. Para fazer a correspondência entre MACs e IPs, o protocolo verifica as máquinas da rede local e cria uma tabela em memória. A máquina que desejar encaminhar pacotes para outra máquina vizinha deve verificar essa tabela para descobrir o MAC associado ao IP dessa máquina vizinha. Se não existir o endereço solicitado nessa tabela, o protocolo ARP emite um pedido na rede e a máquina que possui o endereço IP responde com seu endereço MAC. O requisitante armazena esse endereço na tabela, durante um tempo pré-determinado, e o usa para realizar a comunicação.

- DNS [17]

  O objetivo dos nomes de domínio é prover um mecanismo para nomear certos recursos, de forma que o mesmo possa ser utilizado por diferentes *hosts*, redes e protocolos.

  O DNS mapeia os nomes de domínio em endereços IP, que são usados para o roteamento dos pacotes de dados. Isso é feito através de um banco de dados distribuído composto por uma estrutura hierárquica de servidores DNS. Quando um usuário tenta visualizar uma determinada página, o *host* de origem acessa essa estrutura hierárquica do DNS para encontrar o servidor de DNS que contém o mapeamento do nome ao respectivo IP, permitindo assim que os pacotes de dados atinjam o *website* correto e estabeleçam uma conexão.

2.3.2 Vulnerabilidades Conhecidas

- HTTP

  O protocolo HTTP não assegura confidencialidade do conteúdo enviado e é vulnerável a vários tipos de ataques, dentre eles o DNS *spoofing* [18]. Essa é uma vulnerabilidade que permite um ataque de camada 7 (camada de aplicação).

Esse protocolo costuma utilizar a porta 80 e, segundo a *Exploit Database,* essa é a porta que detêm o maior número de explorações [19] registradas, tendo dez vezes mais que a segunda colocada.

- ARP

Este protocolo permite que qualquer *host* na subrede responda a requisições ARP, dando seu próprio endereço MAC como resposta [20]. Entretanto, a autenticidade do pacote não é verificada. Devido a essa falha, um *host* malicioso pode, por exemplo, responder todas as requisições em uma rede e dominá-la ao fingir deter o IP de todas os demais *hosts* da rede, tornando-o capaz de manipular todo o tráfego [21]. Dependendo do modo que for utilizado, esse ataque à tabela ARP pode causar um *Denial of Service* (DoS) na rede.

Essa vulnerabilidade permite um ataque de camada 2, no Enlace de Dados.

2.3.3 Métodos de ataque

- *Man In The Middle Attack* (MITM)

O "Ataque de Homem do Meio" tem como objetivo base inserir o *host* do *hacker* como um *proxy* anônimo, ou seja, um intermediador entre a conexão de dois dispositivos, como a Figura 4 exemplifica.

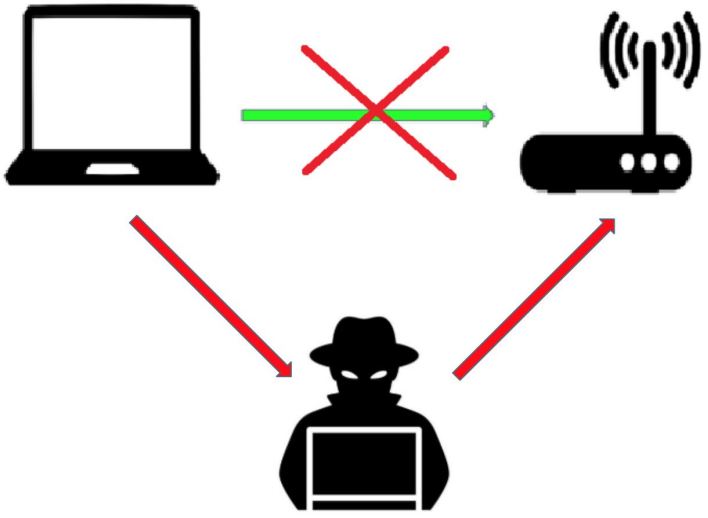

Figura 4: Diagrama do ataque

A motivação por trás desse tipo de ataque é de que, nessa conexão, podem trafegar dados relevantes do ponto de vista do atacante. O objetivo é "ouvir" o tráfego, a espera de algo relevante, como um *login* e senha, ou dados de cartão de crédito. Outro ponto de grande interesse é a possibilidade de adulterar as informação que saem ou chegam em um determinado *host*. Tudo isso sem ser detectado.

- ARP *Poison* / *Spoof*

  Esse método visa modificar a tabela ARP, fazendo com que o tráfego das vítimas seja roteado através da máquina do atacante. O ataque se aproveita do método no qual o protocolo associa IPs a MACs, uma vez que qualquer MAC pode assumir qualquer IP [22].

  Ambos, *poison* e *spoof*, visam enganar o protocolo, fazendo com que o mesmo acredite que o *host* do atacante é o da vítima. Em síntese, o ataque consiste no "**envenenamento**" da tabela ARP com um MAC *address* "**forjado**" [23] [24].

- DNS *Spoof*

  O DNS *Spoof* é usado com o objetivo de corromper o *cache* do sistema de nomes de domínio, forçando o sistema a retornar um valor errado. A utilização do *cache*, nessa situação, oferece um ganho de performance, porém abre uma porta para ataque.

  Esse método é muito utilizado em ataques que utilizam de *sites* forjados. Normalmente, nesse cenário o atacante visa extrair dados de uma vítima que está tentando acessar uma página. Dentro do escopo deste trabalho, tal método será utilizado para transferir a vítima de um *site* **x** para outro **y**, sem que a vítima do ataque perceba.

3. Descrição da Execução do Ataque

O problema abordado está dentre aqueles cobertos pela falta de segurança em aplicações *mobile* e conexões de rede. Por isso, serão mostradas vulnerabilidades nas conexões não criptografadas em redes locais sem fio. Um dos objetivos é expor o fato de que a má utilização da rede facilita o sucesso de ataques como o MITM. Além disso, procuramos expor os efeitos a respeito do impacto do ataque na vítima.

3.1 Propostas de Execução do Ataque

Foi feito um estudo a fim de decidir qual a melhor forma de ataque para os testes. Desse estudo, obteve-se as opções descritas abaixo:

1. Substituição do aplicativo normal por um manipulado pelo atacante. Para atingir tal objetivo, devem ser usadas técnicas complexas de engenharia social, o que foge do escopo do trabalho.

2. Colocar um vírus no aplicativo. Essa opção se mostrou interessante, porém descartada apesar de seu escopo mais técnico, pois mais uma vez teríamos que utilizar métodos de engenharia social.

3. Realizar o ataque enquanto o usuário estiver jogando, o que traz um efeito significativo para a vítima. Para tal é necessário "entrar" na rede da vítima, redirecionando todas as requisições do aplicativo para um *site* falso ou mudando as imagens a serem recebidas uma a uma.

Após analisarmos as opções, a opção 3 foi escolhida para o experimento deste trabalho, a ser descrito no Capítulo 4.

3.2 Descrição da Arquitetura

No cenário deste trabalho, o *hacker* utiliza uma máquina virtual "rodando" o sistema operacional Kali Linux, usada como atacante, tendo uma máquina Windows e um dispositivo Android na versão 7.1 Nougat como vítimas. Todos ficam conectados na mesma rede Wi-Fi.

3.3 A Aplicação

A ideia do jogo é simples. O local escolhido foi o de uma floresta, ambiente típico para muitos jogos. A ideia é permitir que o jogador mexa a cabeça e possa observar esse cenário, onde se encontra uma caixa. Ao mirar na caixa, ela desaparece, para então reaparecer em outro lugar aleatório.

A aplicação foi feita na ferramenta de desenvolvimento Unity, onde foram desenvolvidos jogos como *Monument Valley 2* para dispositivos móveis e *Cuphead,* um jogo exclusivo para a plataforma *Xbox One* que relembra os jogos clássicos de tiro de 8 e 16 bits, como *Contra*, e um estilo das animações da década de 1930 [25]. Foi utilizada a última versão do *Software Development Kit* (SDK) do Google para realidade virtual no Unity. Essa versão contém um conjunto de ferramentas que permitem a criação dessas aplicações [26].

A Figura 5 ilustra a primeira versão do aplicativo, apresentando a floresta e o plano de fundo de montanha.

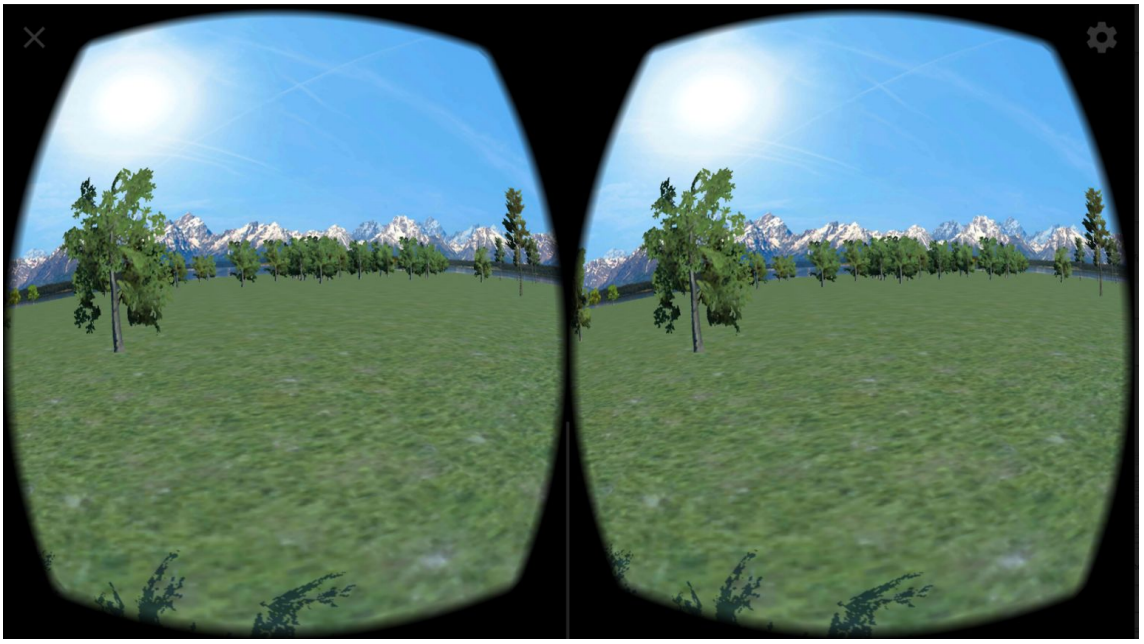

Figura 5 : Primeira tela em funcionamento

Para o bom funcionamento do objeto “caixa” no jogo, foi criada uma função que permitiu a ela sumir e reaparecer em outro lugar dentro do *range* do jogador. A Figura 6 representa esse código.

```
public void RandomlyTeleport() {
      gameObject.transform.position = new Vector3(Random.Range(240, 250),
Random.Range(0.5f, 2), Random.Range(165, 185));
   }
```

Figura 6: Código da função RandomlyTeleport

Para realizar nossos ataques decidimos modificar a aplicação para fazer requisições a um determinado servidor *web*. Utilizamos o *000WebHost* [27], que permite o registro gratuito de dois domínios. Criamos então dois *sites*: um que dispõe as informações a serem chamadas quando a aplicação for iniciada e outro similar, para ser utilizado durante o ataque.

O plano de fundo do jogo é um objeto chamado *skybox*. Esse objeto, próprio do Unity, permite inserir uma imagem previamente criada pelo *Unity* como fundo, ou importar essa *skybox* [28] como uma extensão da loja do *Unity*. A princípio, o plano de fundo *default* é apenas uma tela vazia de cor cinza. Para carregar o plano de fundo, importamos algumas *skybox* da loja e inserimos no projeto. Até o fim do nosso experimento, não foi possível concluir se é possível inserir qualquer imagem da Internet como *skybox*. Utilizamos uma classe *controller* auxiliar chamada *MaterialsController*, que é uma classe onde são feitas as chamadas do tipo *GET* para requisição do nome da *skybox*, e uma classe principal chamada *VRChangeColor*, que é uma classe com funções próprias do Unity como *Start*. Essa classe é chamada quando a aplicação é carregada pela primeira vez e, posteriormente, usa-se a função *Update*, que é chamada a cada *frame* por segundo (unidade de medida da cadência de um dispositivo audiovisual) da aplicação. Nessa classe são feitas as associações do nome da *skybox*, obtido após sua requisição, ao respectivo objeto criado anteriormente como plano de fundo.

Para ilustrar, apresentamos a Figura 7, referente à classe *MaterialsController*. A função *Start* é mostrada na Figura 8. Fizemos também um contador para evitar que fossem feitas requisições ao servidor a cada *frame* por segundo da aplicação, o que poderia resultar em uma sobrecarga de requisições.

```
public string DownloadCurrent()
{
        HttpWebRequest webRequest =
(HttpWebRequest)WebRequest.Create("http://dgtcc.000webhostapp.com/");
        webRequest.Method = "GET";
        webRequest.Timeout = 3000;
        HttpWebResponse response = (HttpWebResponse)webRequest.GetResponse();

        string htmlContent = "";
        var encoding = ASCIIEncoding.ASCII;
        using (var reader = new System.IO.StreamReader(response.GetResponseStream(),
encoding))
        {
           htmlContent = reader.ReadToEnd();
        }

        Regex regex = new Regex("Using:(?<valor>.*?)</p>",RegexOptions.IgnoreCase |
RegexOptions.Singleline);
        Match matValue = regex.Match(htmlContent);
        string valor = matValue.Groups["valor"].Value.Trim();

        return valor;
    }
```

Figura 7: Código de requisição do nome da *skybox*

```
  void Start() {
     nameUsing = matController.DownloadCurrent();
     escolheSkybox(nameUsing);
  }

  void escolheSkybox(string nameUsing) {
     if   (nameUsing == "mountain")   { RenderSettings.skybox = mountain; }
     else if (nameUsing == "galaxy")   { RenderSettings.skybox = galaxy;     }
     else if     (nameUsing == "hell")   { RenderSettings.skybox = hell;         }
  }
```

Figura 8: Código da função *start* de *VRChangeColor*

Já em relação ao terreno, fizemos requisições para *download* de uma imagem qualquer da Internet, sendo que, em nosso caso, usamos um *site* criado por nós. Essa imagem é então associada como textura principal do objeto terreno. Para tanto, decidimos criar outra classe *controller* chamada de *TextureController*. Ela faz basicamente a mesma requisição que a classe *MaterialsController*, sendo que a única diferença é que ela retorna a URL (caminho) da imagem que é associada ao objeto terreno na classe principal desse objeto, chamada *ChangeColorTerrain*.

Nessa classe principal, como vemos abaixo na Figura 9, foi utilizada uma função do *Unity* chamada de *Coroutine* que permite a criação de uma rotina baseada no objeto do *Unity WWW* [29], que serve justamente para trabalhar com URLs e a utilização de imagens ou vídeos encontrados na Internet. A função que faz essa associação entre o objeto e nova imagem é a função *LoadAsset*, que recebe a URL da imagem e a associa à textura principal.

```
void Start () {
    gameobject = GameObject.Find("Terrain");

    terrainData = GetComponent<Terrain>().terrainData;
    string urlSaved = textureController.DownloadCurrent();
    StartCoroutine(LoadAsset(urlSaved));
}
```

Figura 9: Código da função *start* de *ChangeColorTerrain*

Na função *Update* é feita a atualização da textura a todas as partes do objeto terreno que já havia sido criado anteriormente. Para isso, é feito um vetor com o tamanho da propriedade *SplatPrototype* [30] (propriedade de textura utilizada pelo objeto do terreno), como visto abaixo na Figura 10, e a propriedade *texture* recebe o *texture* principal já atualizado.

```
void Update () {
    count++;
    if (count >= 1000)
```

```
    {
        string urlSaved = textureController.DownloadCurrent();
        StartCoroutine(LoadAsset(urlSaved));
        count = 0;
    }

    SplatPrototype[] splatPrototype = new SplatPrototype[terrainData.splatPrototypes.Length];
    for (int i = 0; i < terrainData.splatPrototypes.Length; i++)
    {
        splatPrototype[i] = new SplatPrototype();
        splatPrototype[i].texture = (Texture2D)texture;
        splatPrototype[i].tileSize = new Vector2(terrainData.splatPrototypes[i].tileSize.x,
terrainData.splatPrototypes[i].tileSize.y);
        splatPrototype[i].tileOffset = new Vector2(terrainData.splatPrototypes[i].tileOffset.x,
terrainData.splatPrototypes[i].tileOffset.y);
    }
    terrainData.splatPrototypes = splatPrototype;
  }
```

Figura 10 : Código de update da classe Texture

Com isso, conseguimos o efeito esperado. Ao inicializar o aplicativo, conseguimos reproduzir a imagem da floresta e da *skybox*. A caixa aparece no local aleatório e o jogador consegue mexer a cabeça, sem nenhum indício de ataque.

4. Ferramentas Adotadas e Tutoriais de Uso

Abaixo descrevemos as ferramentas utilizadas, tanto para criação da aplicação quanto para a realização dos ataques, com exemplos de comandos e figuras para um maior entendimento das mesmas.

4.1 Ferramentas Utilizadas

Para esse laboratório, estaremos utilizando o Kali Linux. Essa é uma distribuição unix, baseada em Debian, que visa oferecer o melhor ambiente possível para o que chamamos de segurança ofensiva. É desenvolvida pela *offensive security*, uma empresa que oferece treinamentos na área de segurança da informação. Neste sistema operacional temos pré instaladas todas as aplicações que serão utilizadas durante o ataque.

Quase todas as *tools* utilizadas durante a exploração das vulnerabilidades são de código aberto, a exceção é o Burp Suite, que exige a compra de uma licença anual. Entretanto, a Port Swigger, desenvolvedora do Burp, oferece uma versão básica sem custo.

4.1.1 Ferramentas do Kali Para Execução do Ataque

- **Nmap**

  O *Network Mapper* é uma ferramenta de exploração de redes e auditoria de segurança, famoso por ser uma das ferramentas mais utilizadas na fase de

reconhecimento. Consegue escanear redes de grande porte com eficiência. O Nmap é utilizado para verificar:

- quais hosts na rede estão acessíveis;
- quais serviços esses *hosts* estão oferecendo (nome da aplicação e versão);
- qual sistema operacional está rodando na máquina; e
- que tipo de filtro e/ou *firewall* está sendo utilizado.

Para o escopo desse projeto, utilizaremos as opções que auxiliam na descoberta de *hosts* ativos e identificação de sistema operacional.

- **Arpspoof**

Essa é a ferramenta base do ataque. O Arpspoof é direto e simples: ele redireciona o tráfego destinado a um *host* para a máquina do *hacker*. Isso é feito através de respostas ARP forjadas.

Antes de iniciar o ataque, é importante checar se a funcionalidade *ip forwarding*, do sistema operacional da máquina atacante, está ativo.

- **Dnsspoof**

Essa ferramenta forja respostas de endereço DNS arbitrárias na LAN. É útil na implementação de ataques de MITM.

- **Wireshark**

O Wireshark é uma ferramenta GUI para análise de tráfego e protocolos de rede. Ela é simples de usar e auxilia o usuário a ter uma ampla visão dos pacotes que estão passando pela rede. Sua utilização, em conjunto com os filtros disponíveis, dão uma percepção muito mais profunda do tráfego, auxiliando na análise de padrões, entre outras vantagens.

- **Burp Suite**

O Burp Suite é um *framework* para teste de invasão. É uma ferramenta muito utilizada tanto por profissionais de segurança como por *hackers* mal intencionados. Ele possui um conjunto de funcionalidades, onde as duas

principais são a análise de vulnerabilidades e o *proxy web* (visível e invisível).

O Burp implementa algumas técnicas muito úteis para certos tipos de ataques. Quanto a MITM, com ele podemos pausar pacotes *on the fly*, alterá-los e até impedi-los de prosseguir na rede. Dado o exposto, podemos afirmar que essa é ferramenta ótima para o caso de teste apresentado mais à frente.

- **Ettercap**

O Ettercap é uma ferramenta completa para ataques MITM. É fácil de usar e possui uma interface amigável. É capaz de fazer boa parte das tarefas que as ferramentas listadas acima fazem, mas com baixo nível de customização.

Essa é uma ótima ferramenta para quem não gosta da linha de comando ou tem pouca experiência. Possui bom desempenho bem em ataques de baixa complexidade.

4.2 Requisitos Técnicos

Para fazer este experimento, utilizaremos:

- um Macbook Pro com processador i5 e 8gb de memória RAM;
- uma máquina virtual Kali Linux *build* 2018.2, configurado em modo *bridge*, com dois núcleos de processamento e 4 GB de RAM;
- um *smartphone* Android (versão 7.1 Nougat) com o aplicativo VR instalado;
- uma rede *wireless* pública, fornecida por meio de um roteador comum.
- duas páginas *web* na Internet que irão funcionar como servidores *web*:
    - Servidor Real: http://dgtcc.000webhostapp.com
    - Servidor Falso: http://dgtccsecond.000webhostapp.com

4.3 Declaração de Exoneração de Responsabilidade

Neste capítulo estaremos explicando, passo a passo, como reproduzir o ataque em uma rede local. Devemos **enfatizar** que este procedimento tem o objetivo de **mostrar vulnerabilidades nas redes e chamar a atenção para os perigos do uso imprudente delas**.

Esse laboratório **não foi feito com intuito de obter vantagens ou extrair dados reais de quaisquer indivíduos**, portanto desencorajamos o uso do conhecimento aqui disposto para tal finalidade.

Todos o processo é realizado de acordo como o código de ética *hacker* do EC-Council [31]. Além disso, todos os testes na rede da universidade foram feitos com o aval dos responsáveis.

4.4 Arquitetura do Ataque

O Macbook, o Android e o roteador serão utilizados como as vítimas do ataque. Já o Kali, será o atacante, o *hacker*. Vale a observação de que o resultado foi atingido em um ambiente não controlado, o mais próximo possível de uma situação real.

Todas as ferramentas utilizadas estão descritas na Seção 4.1 deste capítulo. Nas Figuras 11 e 12 vemos, respectivamente, exemplo de uma conexão normal e o que se espera de uma conexão durante o ataque.

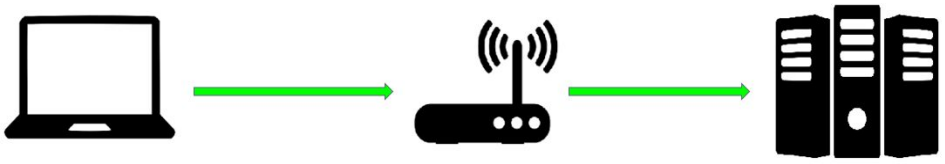

Figura 11: Conexão cliente-servidor normal

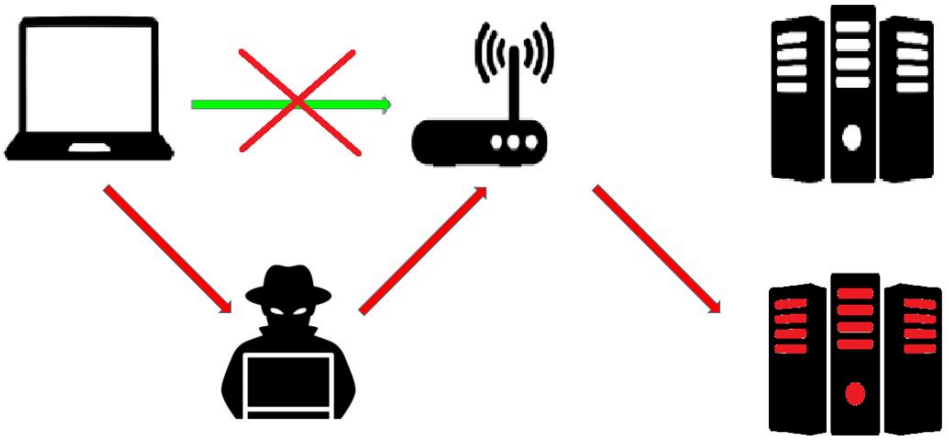

Figura 12: Conexão cliente-servidor alterada

4.5 Tutorial de Ataque

Seguem abaixo os passos para a efetivação do ataque:

- Primeiro devemos confirmar que ambas as máquinas estão conectadas na mesma rede *wireless*. Demonstramos isso nas Figuras 13 e 14.

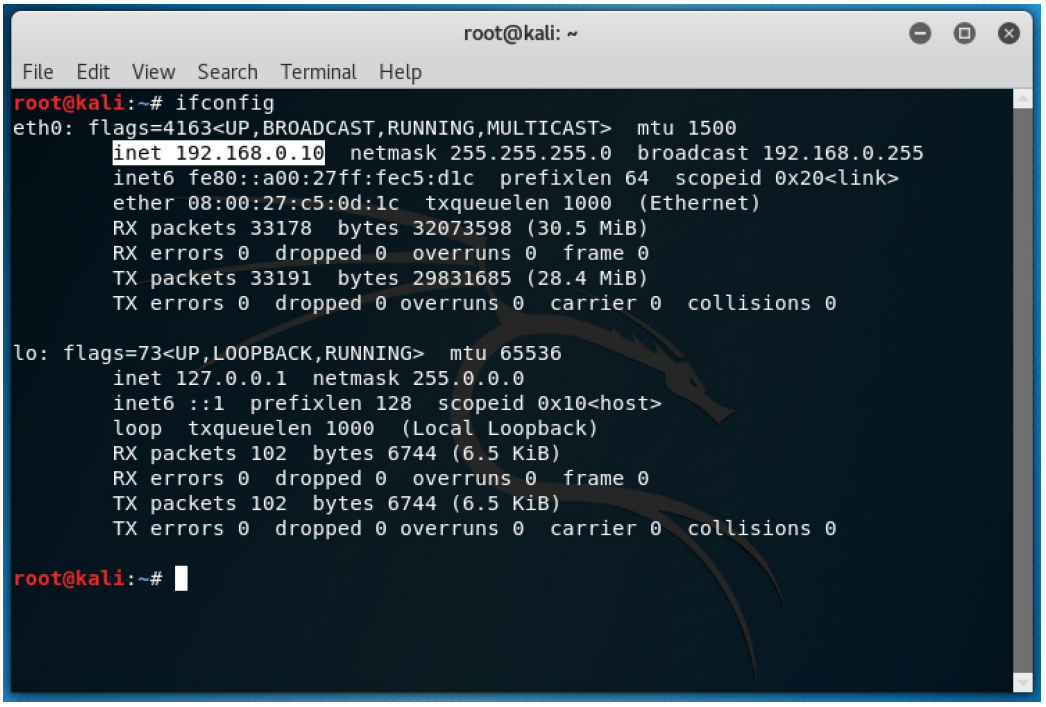

Figura 13: Comando ifconfig na VM Kali

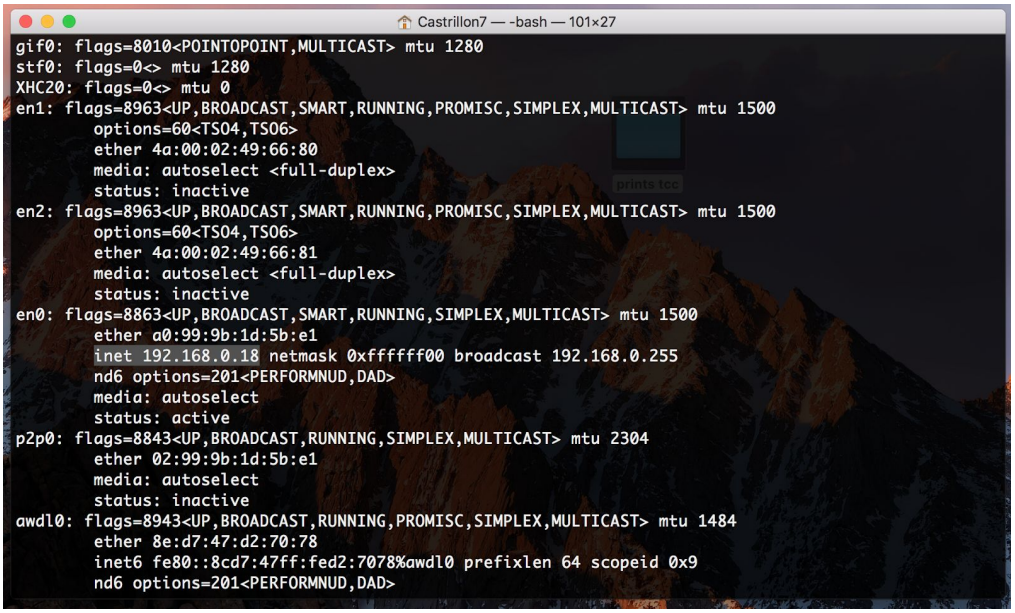

Figura 14: Comando ifconfig no macbook (vítima)

- Em um cenário real, não saberíamos os IPs das máquinas na sub-rede. Sendo assim, fazemos o mapeamento da rede usando o Nmap. Varremos toda a subrede com a opção *-A* (Figura 15), que checa o sistema operacional (OS) de cada *host* encontrado. Essa opção nem sempre retorna o OS com precisão, mas sua utilização é válida.

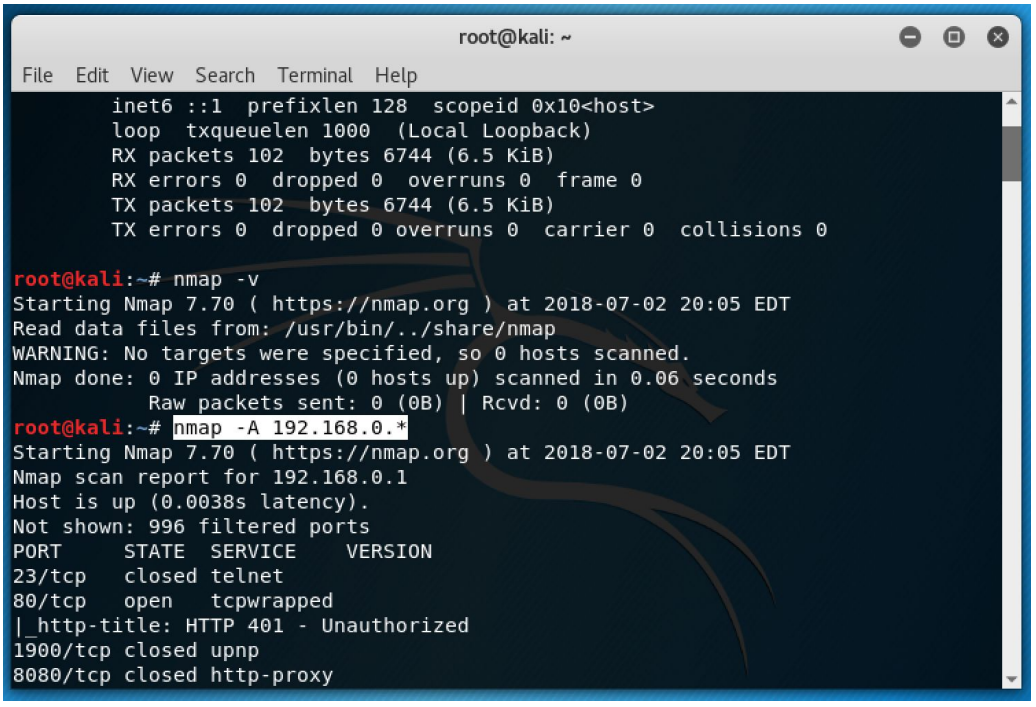

Figura 15: Rodando o nmap na subrede

- Vemos que a varredura da rede trás vários *hosts* conectados ao roteador. Analisando os resultados, achamos o *host* a ser atacado, que, neste caso, é o Macbook. Vemos, na Figura 16, que o mesmo foi identificado pelo endereço MAC.

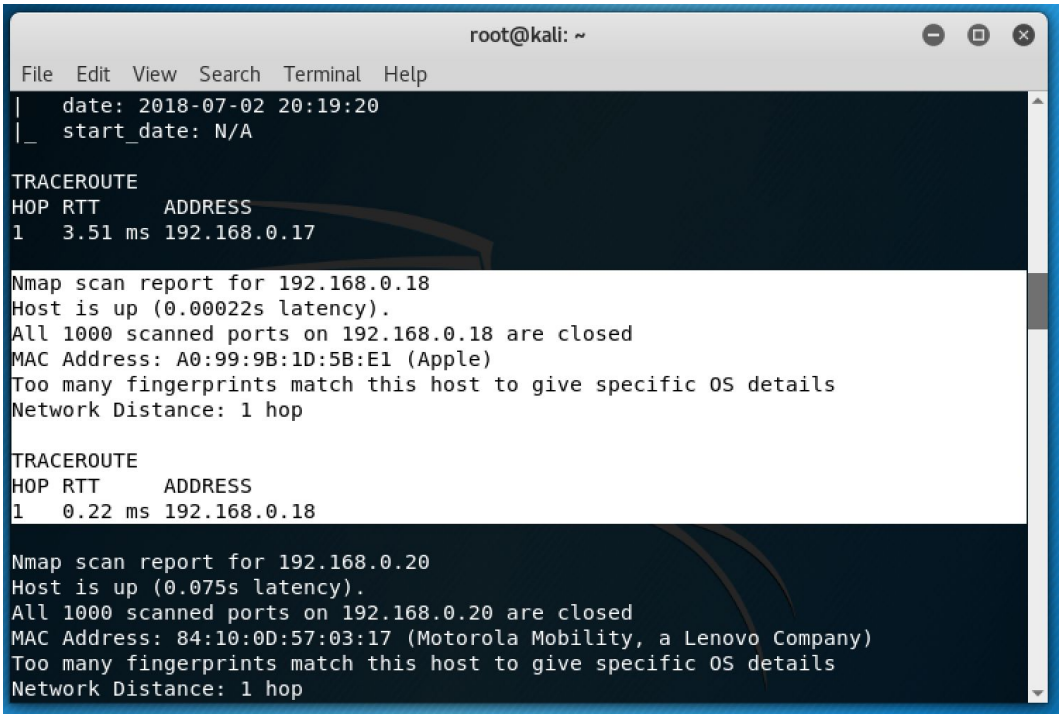

Figura 16: Resultado após rodar o nmap

- Após identificar a vítima, passamos para a parte de configuração do ataque. Como explicado na Seção 2.3.3, faremos um ataque de MITM. Sendo assim, a máquina Kali funcionará como um "roteador", ou seja, um *proxy* invisível. Dessa forma, é necessário que ***ip forward*** seja ativado. Basta alterar o arquivo citado de 0 para 1, como exemplificado na Figura 17.

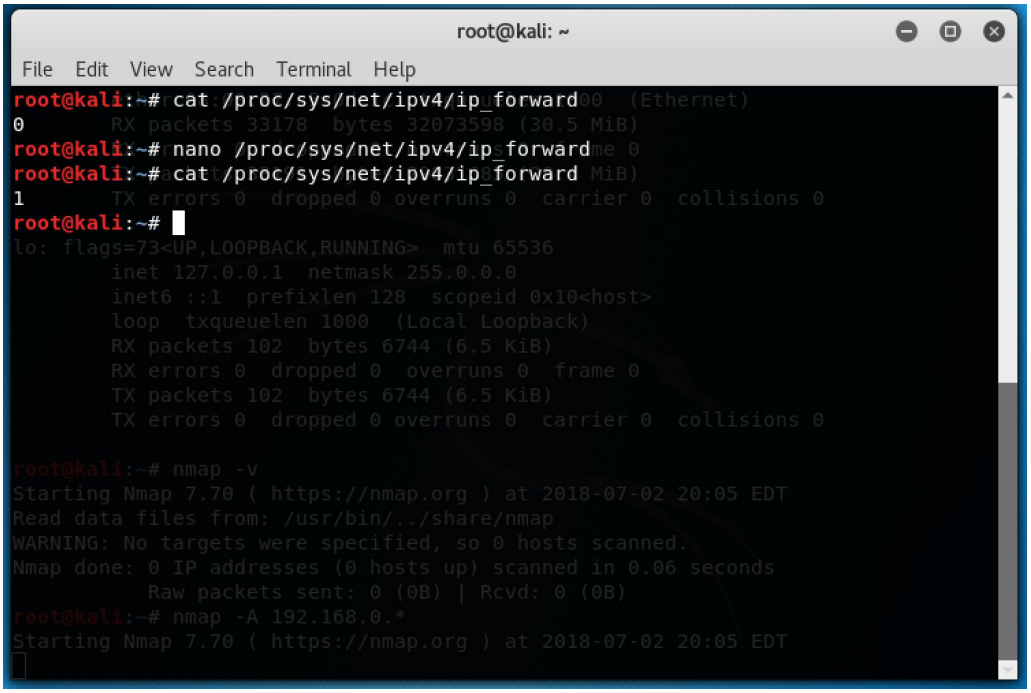

Figura 17: Edição *ip_forward*

- Tendo configurado o roteamento, iniciaremos o ataque. Podemos fazer o *arpspoof*, interceptar os pacotes, nos seguintes sentidos:
    1. Vítima → *Gateway* & *Gateway* → Vítima;
    2. Vítima → *Gateway*, apenas, e desse modo só é interceptada a requisição;
    3. *Gateway* → Vítima, apenas, e desse modo só é interceptada a resposta do servidor.

    Neste laboratório, aplicamos a opção 1 (Figura 18), assim podemos alterar tanto requisição como resposta no Burp.

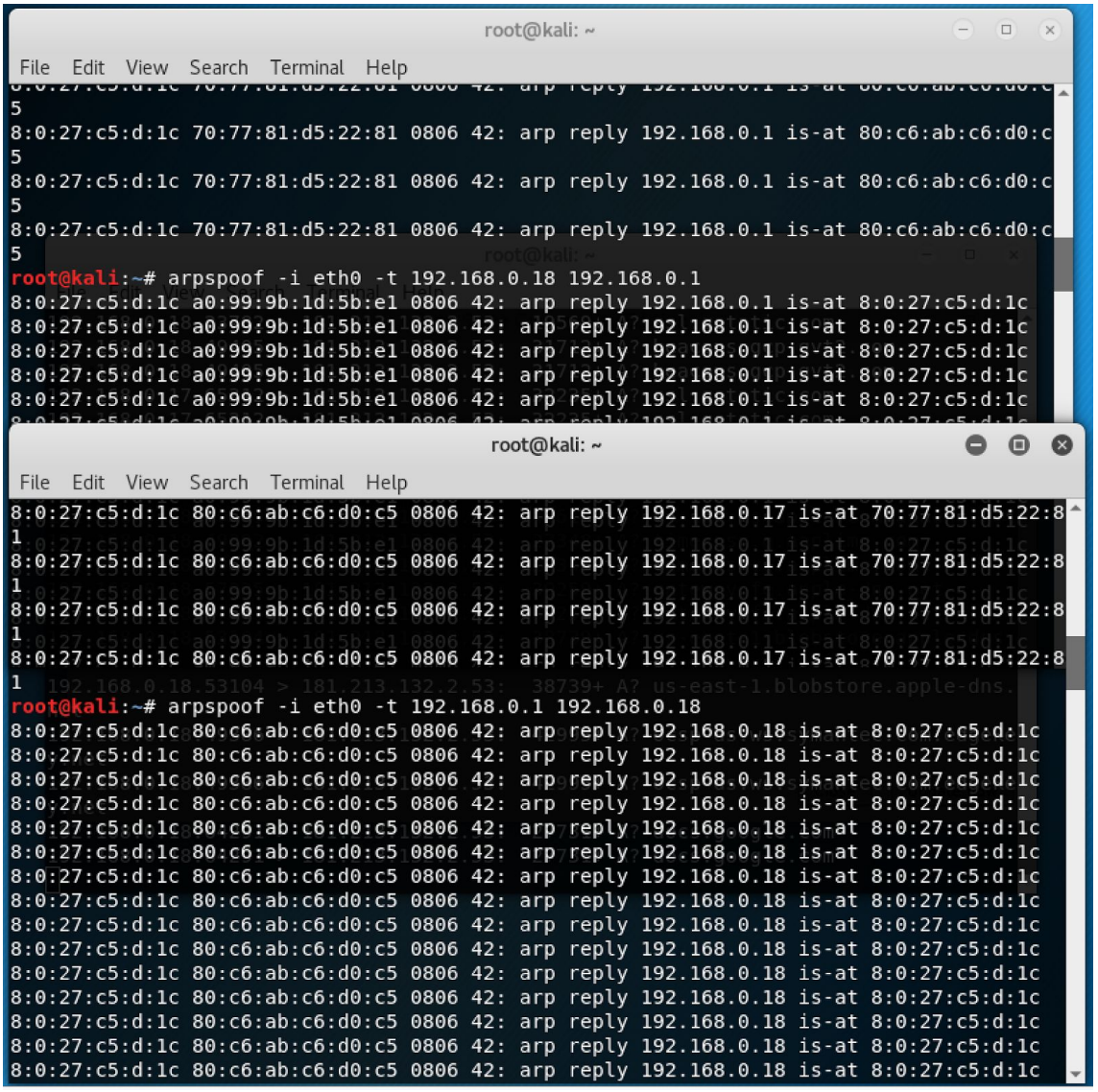

Figura 18: Comandos *arpspoof*

- O próximo passo é acionar o *dnsspoof,* assim não haverá problemas quando forem feitas as mudanças de servidor no Burp. Vemos na Figura 19 que o *site* desejado foi lido pela aplicação.

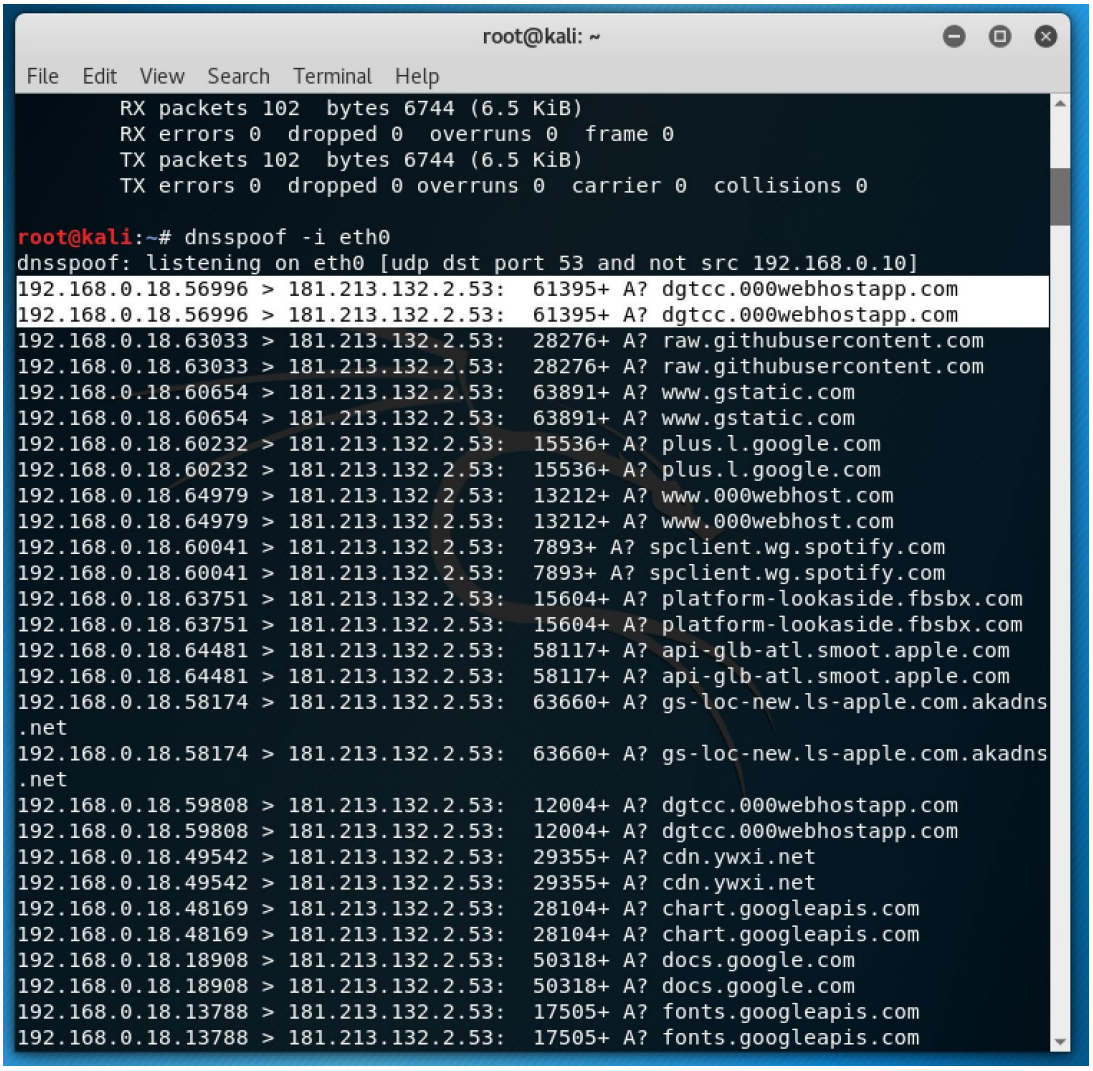

Figura 19: Comando *dnsspoof*

- Para ratificar que o tráfego está sendo roteado através da VM (confirmando que o ataque está tendo sucesso), utilizamos o Wireshark. Dessa maneira, podemos visualizar os pacotes que estão trafegando, partindo do princípio que a vítima está utilizando seu aparelho. Nesse caso, conforme mostrado na Figura 20, com o filtro utilizado vemos apenas pacotes HTTP com destino / origem na máquina da vítima.

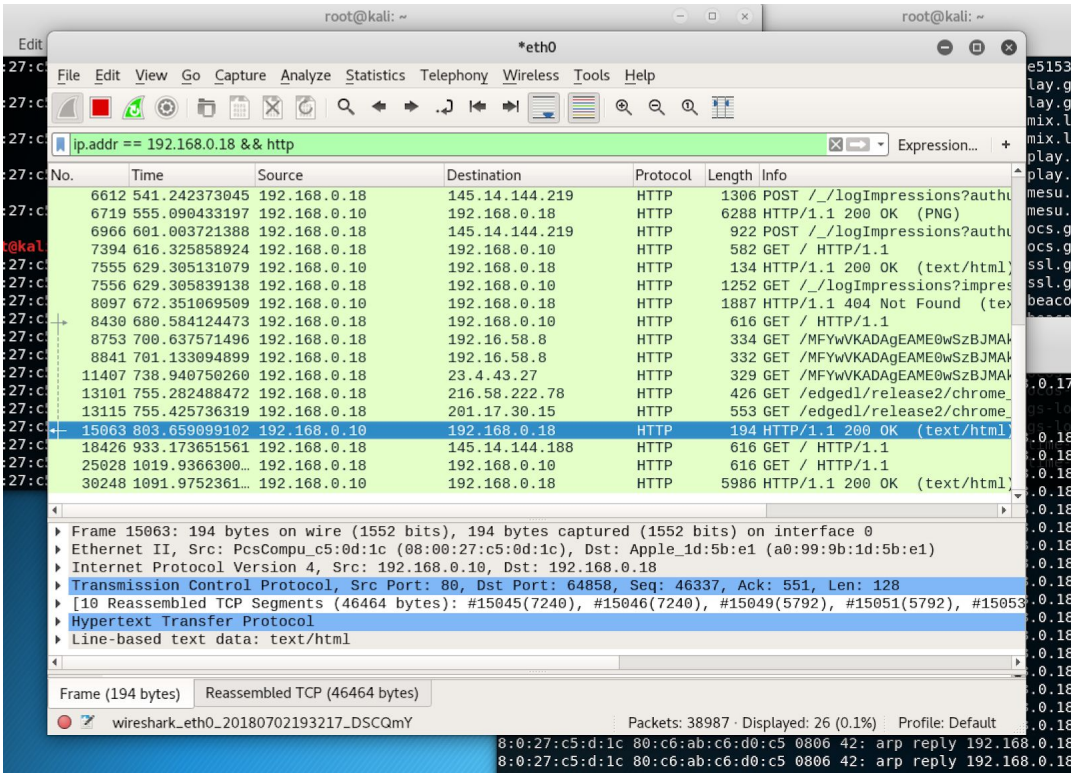

Figura 20: Visualizando pacotes com o Wireshark

- Essa é a parte mais importante do experimento. A configuração do Burp pode ser complexa, como vemos na Figura 21, mas a mesma é essencial para o sucesso do ataque. Elas ficam divididas em 3 partes:

    1. *Proxy Listeners*

        Por usarmos o *proxy* de maneira invisível, sem a ciência da vítima, devemos fazer com que o Burp "ouça" as portas por onde passarão os pacotes a serem alterados. Nesse caso, focaremos na porta 80, correspondente ao HTTP. Vale enfatizar que se deve marcar a opção "*invisible*", por causa da natureza do ataque.

    2. *Intercept Client Request*

        Essa opção deve ser ativada quando o objetivo é analisar ou alterar as requisições do cliente - no caso, a vítima. Fazemos então com que a aplicação intercepte os métodos HTTP *GET* e *POST*, assim nos permitindo editá-las.

    3. *Intercept Server Response*

        Essa opção deve ser ativada quando o objetivo é analisar ou

alterar a resposta do servidor. É necessário configurá-la para capturar todos os pacotes que retornam o código HTTP 200, que significa OK (resposta bem sucedida).

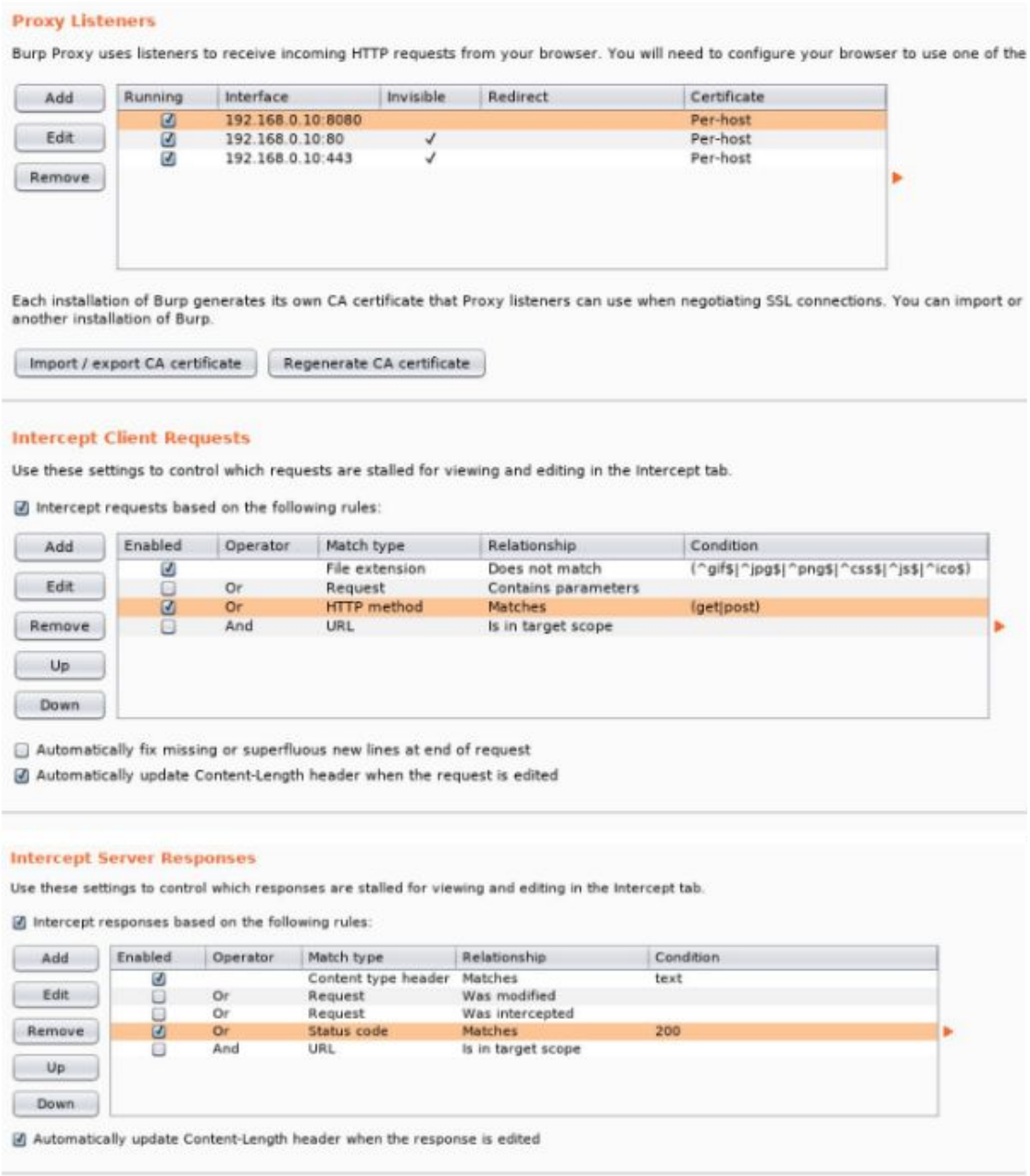

Figura 21: Configurações do Burp Suite

- Uma configuração opcional, conforme mostrado nas Figuras 22 e 23, é fazer as alterações nos pacotes automaticamente. Utilizando esse método, o atacante ganha tempo e simplifica a tarefa de modificação dos pacotes. No entanto, a desvantagem desse método é que, uma vez configurado, o Burp redirecionará

todo o tráfego da porta escolhida, no caso a 80, para o *site* desejado. Isso não afeta o experimento em questão, mas pode atrapalhar em outros casos.

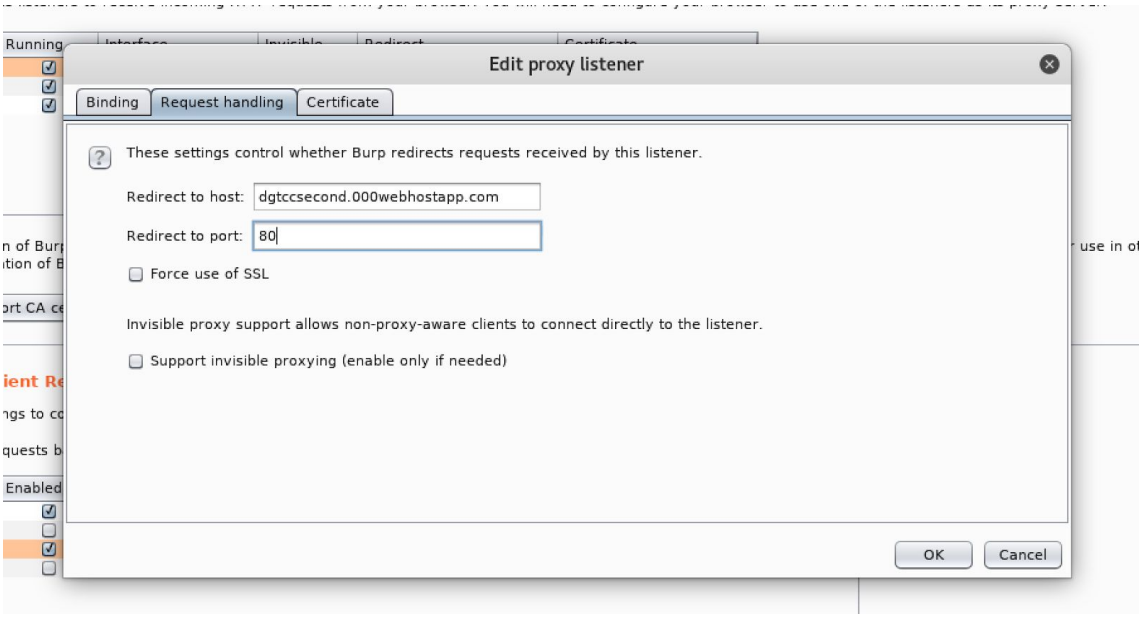

Figura 22: Configuração para mudar o destino da requisição automaticamente

Figura 23: Configuração para mudar o *host* da requisição automaticamente

- Na Figura 24 vemos a requisição *GET* interceptada. A partir desse ponto podemos fazer diversas alterações no pacote, que logo após será transmitido ao servidor. Durante esse processo, no *host* Windows, o navegador fica a espera da resposta. Já no aplicativo Android, sucessivas requisições são feitas, a espera de uma resposta.

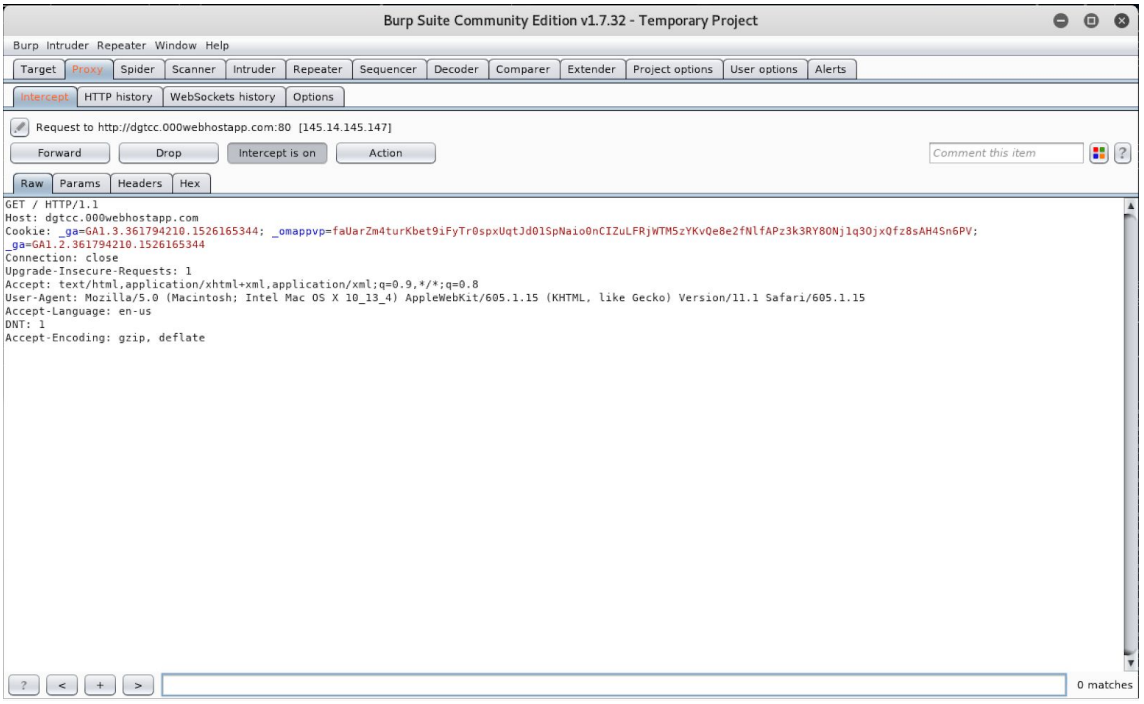

Figura 24: Requisição interceptada

- A Figura 25 mostra a edição de 2 campos:
    1. *Request to*, que é o destino da requisição. A mudança fará com que a *request* seja feita para o *site* falso, http://dgtcc.000webhostapp.com.
    2. *Host,* que nesse cenário é o mesmo que o do ponto anterior. É importante mudar ambos para que não haja conflitos, o pacote não se perca e / ou a conexão seja derrubada.

Os resultados obtidos estão apresentados no próximo capítulo.

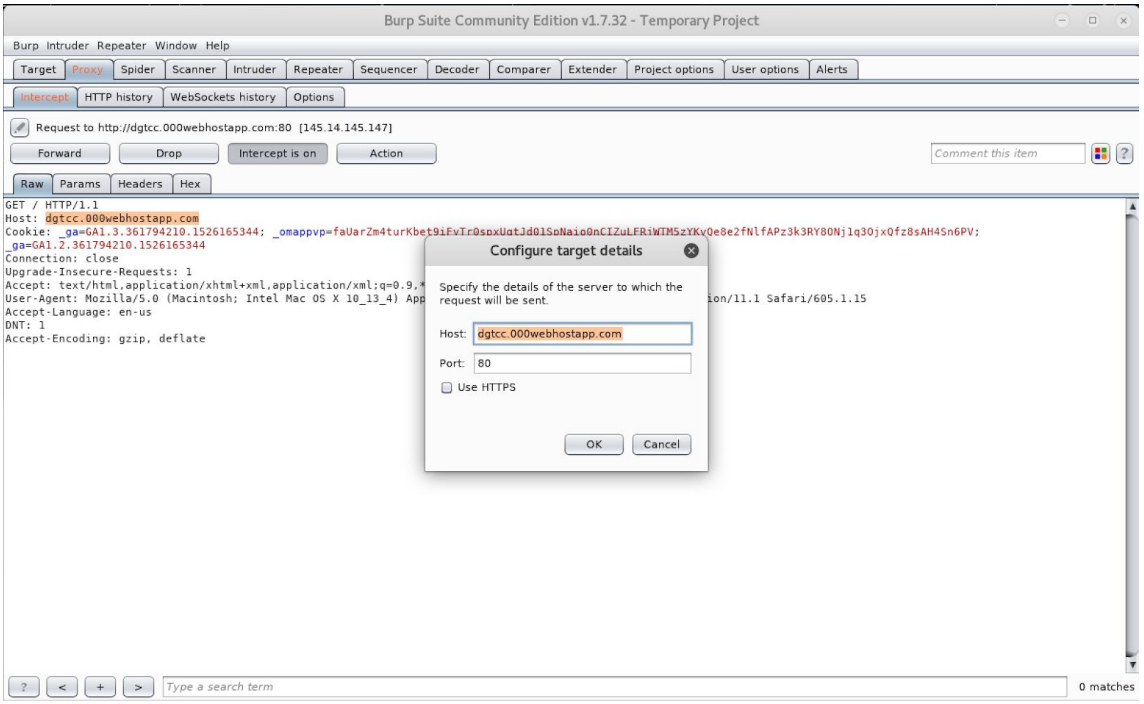

Figura 25: Edição da requisição

5. Resultados

Ao fim da execução do ataque descrito no capítulo anterior podemos analisar seus resultados e consequências. Abaixo estão expostas imagens que mostram os resultados antes, durante e depois da exploração.

5.1 No Computador:

Acessando o servidor (*site*) oficial antes do ataque, a página é encontrada como visto na Figura 26. Esse é o resultado esperado, antes do ataque.

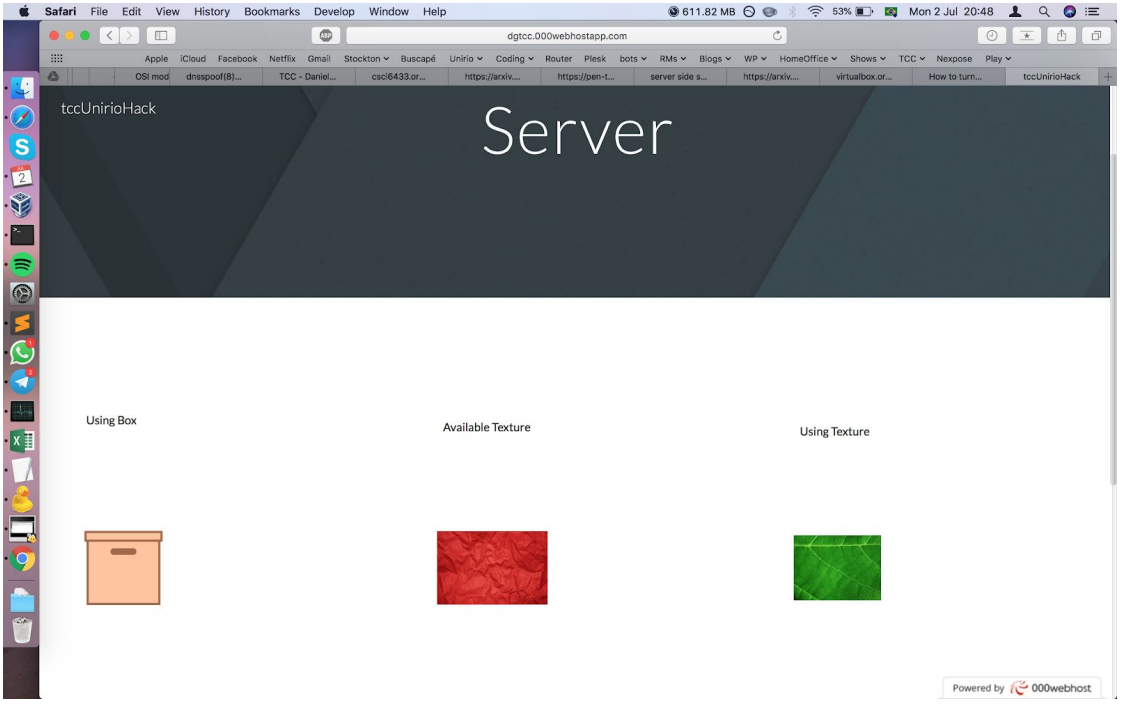

Figura 26: Site oficial

Acessando esse mesmo *site* durante o ataque, o usuário foi redirecionado para a página falsa, como mostrado na Figura 27.

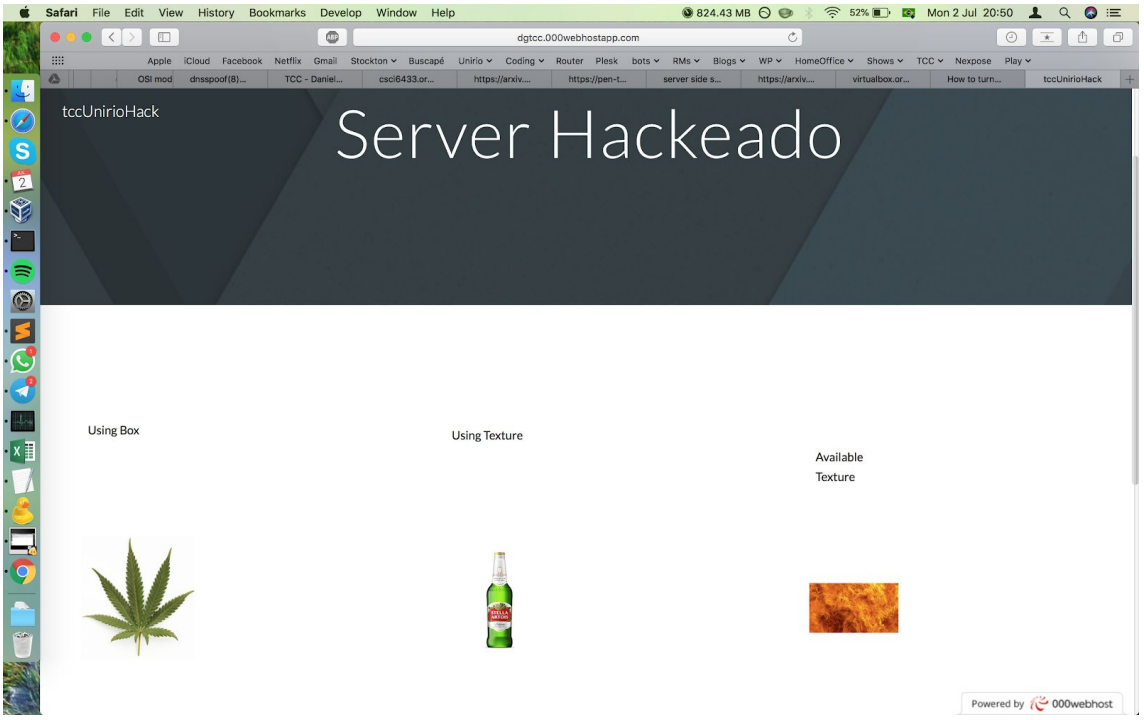

Figura 27: Site falso

Vale ressaltar que, durante o ataque, um dos objetivos era o de que a vítima não tivesse ciência das alterações feitas. Isto é exemplificado na Figura 28, onde se verifica que o conteúdo muda, porém a URL se mantém a mesma.

Figura 28: Comparação entre os resultados

5.2 No Celular:

Na Figura 29 podemos ver o cenário obtido quando acessado o *site* oficial.

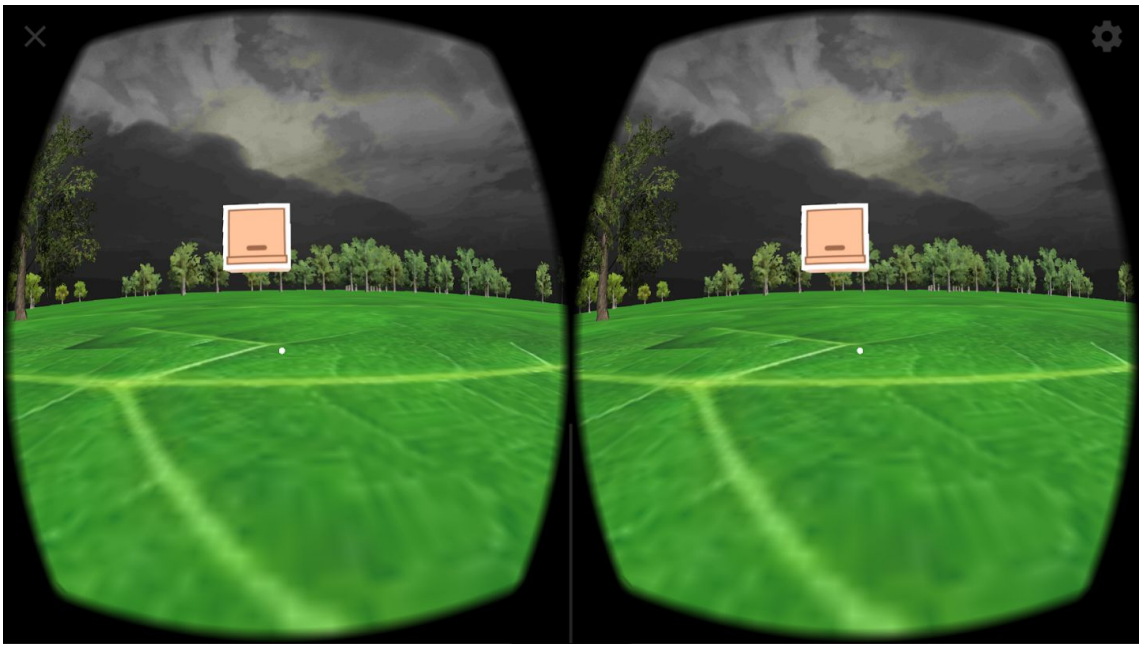

Figura 29: Funcionamento normal do aplicativo

Na Figura 30 vemos uma imagem do cenário *default* obtido quando não há conexão ou quando a requisição ao *site* oficial falha. Este cenário pode ocorrer quando o ataque não tem sucesso total, pois apenas impede a vítima de utilizar a rede.

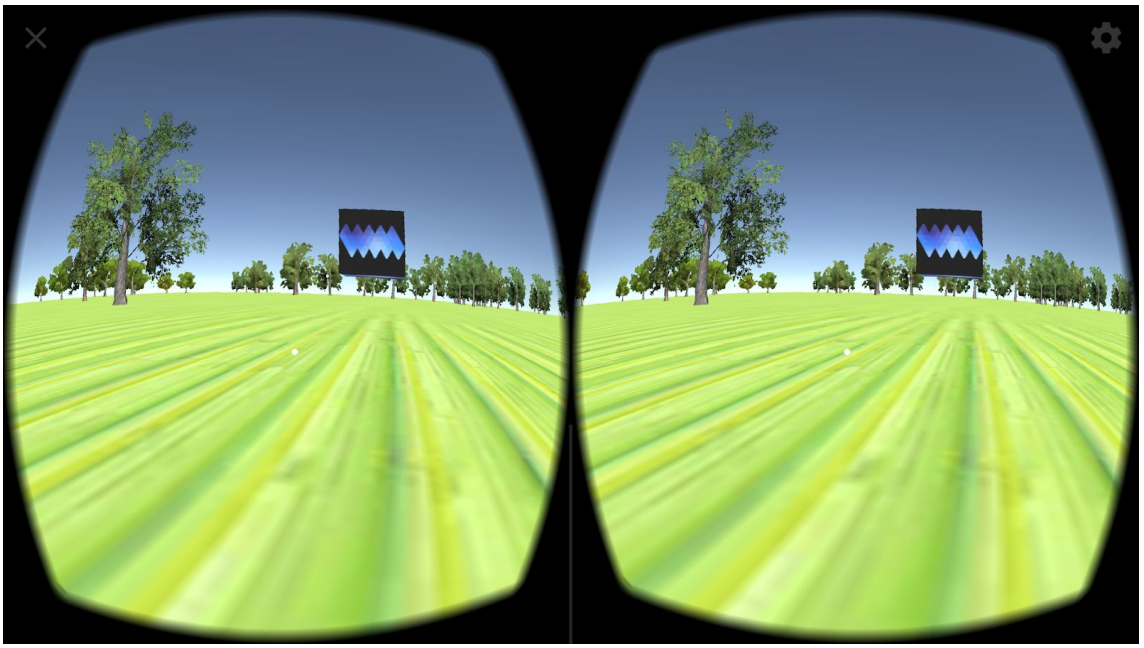

Figura 30: Cenário *default* do aplicativo

Já na Figura 31, vemos o cenário onde o ataque obtêm sucesso. Nesse caso o céu, a caixa e o chão foram alterados.

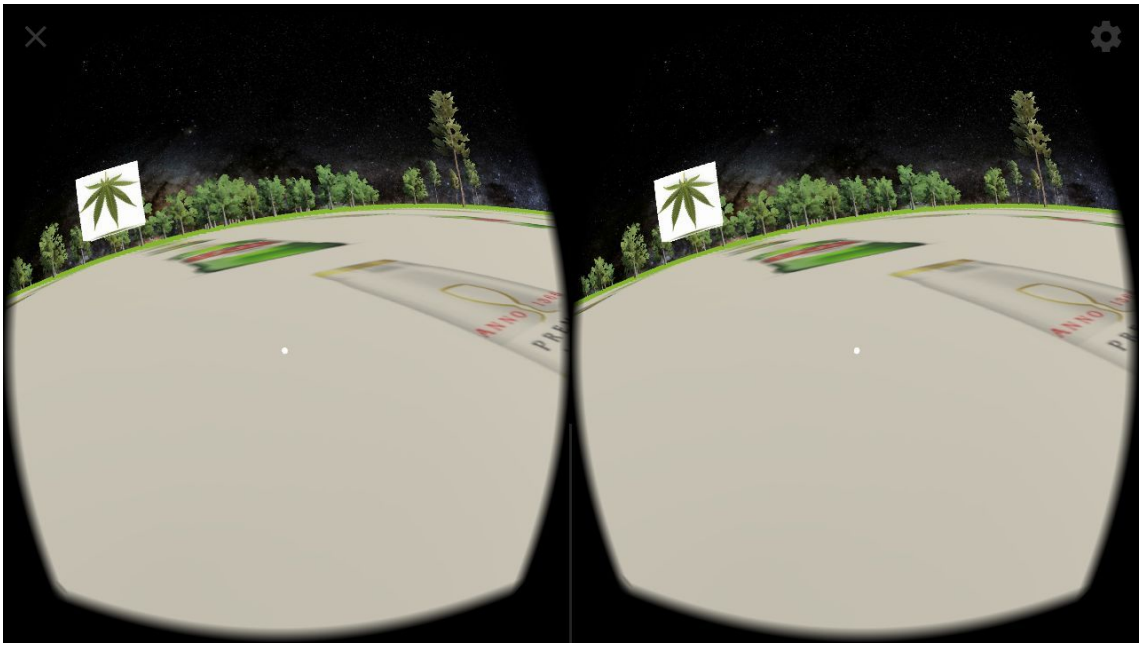

Figura 31: Aplicativo hackeado

5.3 Limitações e Dificuldades Encontradas

Durante o processo de construção da aplicação e execução do ataque, foram encontradas algumas barreiras que foram vencidas em sua maioria.

Sobre o ataque de MITM em si, dificuldades foram encontradas quanto à constância e estabilidade dos resultados. Por muitas vezes o resultado obtido era um DoS (do inglês *Denial of Service*), onde a vítima foi impedida de utilizar a rede. Essa dificuldade foi mitigada após um melhor entendimento da configuração do Burp, o que consumiu parte considerável do tempo.

De um ponto de vista mais técnico, a maior dificuldade esteve em efetivamente modificar os pacote. Ataques de MITM geralmente focam no roubo de credenciais, o que fez do experimento um desafio maior uma vez que não havia rotinas conhecidas e já empregadas para o que aqui se propôs. Conforme mostrado, acabamos utilizando o Burp Suite para fazer as modificações, entretanto, há escassa documentação a respeito de como usá-lo para a finalidade pretendida. Após diversos ajustes finos, chegamos a uma configuração que nos permitiu atingir o objetivo desejado.

No que diz respeito à criação da aplicação, apesar de tanto o Unity quanto o SDK do Google Cardboard apresentarem bom suporte ao desenvolvedor, houve um grande número de funcionalidades a serem aprendidas, o que tomou grande parte do tempo de desenvolvimento.

6. Conclusão

Neste trabalho discorremos sobre como VR é uma tecnologia em ascensão. Falamos da sua história e de como criar um aplicação. Também afirmamos que existem vulnerabilidades nos métodos de implementação do VR, o que nos trouxe a idéia de executar esse projeto. Expomos como funcionam os protocolos envolvidos na exploração das vulnerabilidades, assim como suas falhas conhecidas. Foi então desenvolvido um aplicativo capaz de raptar uma conexão de VR e, nesta interceptação, retornar ao usuário imagens forjadas, provenientes de um servidor falso, sem que o usuário perceba a falsificação.

Ao final da criação do aplicativo usado nesse trabalho, do estudo sobre segurança e da execução do ataque, concluímos que aplicações VR podem ser atacadas utilizando MITM. O ataque obteve sucesso ao demonstrar falhas nos protocolos HTTP e ARP, alterando o conteúdo exposto ao usuário final.

Outro objetivo atingido foi o de demonstrar a importância de se adotar medidas de segurança ao se usar sistemas de VR pela rede. Devido à simplicidade do ataque, mostramos efetivamente que práticas básicas de segurança devem ser implementadas em quaisquer tipo de conexão. A utilização de criptografia e protocolos seguros se mostram essenciais nesse quesito.

Verificamos que esse tipo de ataque em uma aplicação VR pode ter um grande impacto. Tomemos como exemplo o jogo Pokemon *Go*, como mencionado no Capítulo 1, que utiliza locais reais como ponto de encontro de jogadores [32], onde alguns desses locais são, por exemplo, redes de *fast food* que pagam para terem suas lojas "anunciadas" no jogo. Caso um indivíduo mal intencionado altere esse local para um concorrente, a empresa responsável pelo *game* pode se encontrar em uma situação complicada, perdendo credibilidade no mercado.

Para os problemas expostos neste trabalho, existem diversos métodos para melhorar a segurança. Para servidores, é o caso de forçar a utilização de protocolos seguros, e para o usuário, evitar utilizar redes públicas, não acessar páginas que não

sejam seguras e não baixar aplicações de procedência duvidosa são exemplos de procedimentos seguros. Já para a raiz do problema, apontada como sendo o protocolo ARP, pode-se implementar algumas soluções, como o ARP estático, impedindo que o *hacker* envie respostas forjadas e assim alterar a tabela ARP [33]. Outro método seria utilizar o S-UAR [34], que implementa criptografia para a verificação da autenticidade do *host*.

Por mais que não tenha sido implementada, vale ressaltar que existe uma biblioteca em Python chamada Scapy [35]. Essa *library* permite que façamos as alterações de uma forma ainda mais customizada, o que tornaria o uso do Burp obsoleto.

Uma coisa é certa: nenhuma defesa garante segurança total. Com dedicação suficiente, qualquer sistema pode ser invadido, mas é melhor estar sempre preparado para o pior.

6.1 Sugestões de Trabalhos Futuros

Encorajamos trabalhos que vão além do que expomos aqui, implementando métodos que não foram testados. Alguns exemplos são:

- aplicar o mesmo conceito, porém na plataforma iOS;
- tentar fazer o ataque em cima de requisições que utilizam protocolos de segurança, quebrando a criptografia ou burlando os protocolos;
- adicionar uma camada extra de segurança, com certificações no próprio código da aplicação, e verificar se é possível repetir o mesmo resultado atingido aqui;
- utilizar o Scapy, uma biblioteca em Python, para reproduzir o funcionamento do Burp de uma forma mais otimizada e eficiente;
- por último, mas definitivamente não menos importante, implementar projetos que visem achar vulnerabilidades em sistemas operacionais *mobile* e em protocolos de segurança que sejam as referência no mercado.

Obviamente, encorajamos que esses projetos sejam executados de acordo com as *guidelines* de um *hacker* ético.

Referências Bibliográficas

[1] LAMKIN, PAUL. VR And AR Headsets To Hit 80 Million By 2021. Forbes, 2017. Disponível em: <https://www.forbes.com/sites/paullamkin/2017/09/29/vr-and-ar-headsets-to-hit-80-million-by-2021/>. Acesso em: 23 maio 2018.

[2] REUTERS.: "'Pokémon Go' pode ser ferramenta de marketing para varejistas". g1.globo.com, 2016. Disponível em: <http://g1.globo.com/economia/negocios/noticia/2016/07/pokemon-go-pode-ser-ferramenta-de-marketing-para-varejistas.html />. Acesso em 23 jul, 2018.

[3] DYBSKY, D.: "The History of Virtual Reality: Ultimate Guide. Part 1", Teslasuit.io, 2017. Disponível em: <https://teslasuit.io/blog/history-of-virtual-reality-ultimate-guide >. Acesso em: 23 jul. 2018.

[4] KOJIKOVSKI , G.: "O Facebook mergulha, de vez, na realidade virtual", Exame.abril.com.br, 2017. Disponível em: <https://exame.abril.com.br/tecnologia/o-facebook-mergulha-de-vez-na-realidade-virtual/ >. Acesso em: 23 jul. 2018.

[5] SPENCE , E.: "Microsoft HoloLens Review: Winning The Reality Wars", Forbes.com, 2017. Disponível em: <https://www.forbes.com/sites/ewanspence/2017/01/14/microsoft-hololens-review-experience-review/#265e4e94eabc >. Acesso em: 23 jul. 2018.

[6] WALLACE , M.: "I Finally Tried PSVR And It's Nothing Like I Expected", Forbes.com, 2017. Disponível em: <https://www.forbes.com/sites/mitchwallace/2017/05/09/i-finally-tried-psvr-and-its-nothing-like-i-expected/#1c6a2171305f >. Acesso em: 23 jul. 2018.

[7] JACOBSON, L, "Virtual Reality": A Status Report, AI Expert, pp. 26-33, 1991.

[8] PIMENTEL, K. & BLAU,B.: "Teaching Your System to Share, IEEE Computer Graphics & Applications", 14(1): 60-65.

[9] HAMANN, R.: “Project Cockpit BR: um simulador de voo incrível e 100% nacional”, Tecmundo.com.br, 2012. Disponível em: <https://www.tecmundo.com.br/campus-party-brasil-2012/19138-project-cockpit-br-um-simulador-de-voo-incrivel-e-100-nacional.htm>. Acesso em: 23 jul. 2018.
[10] GROZDANIC, L.: “Elon Musk uses Iron Man-inspired holographic 3-D user interface to print a rocket part”, Inhabitat.com, 2015. Disponível em: <https://inhabitat.com/elon-musk-unveils-his-iron-man-inspired-hand-manipulated-3d-holographic-technology/ >. Acesso em: 23 jul. 2018.
[11] DAVIES, C.: “Pokemon GO AR+ update gives iPhone Trainers an edge”. Slashgear.com, 2017. Disponível em: <https://www.slashgear.com/pokemon-go-ar-plus-apple-arkit-augmented-reality-game-update-20512429 />. Acesso em 23 jul, 2018.
[12] IETF. Hypertext Transfer Protocol -- HTTP/1.1. [s.l.: s.n.], 1999. Disponível em: <https://tools.ietf.org/html/rfc2616>. Acesso em: 5 jun. 2018.
[13] IETF. The Secure HyperText Transfer Protocol. [s.l.: s.n.], 1999. Disponível em: <https://tools.ietf.org/html/rfc2660>. Acesso em: 5 jun. 2018.
[14] MARLINSPIKE, MOXIE. sslstrip. Moxie.org. Disponível em: <https://moxie.org/software/sslstrip/>. Acesso em: 5 jun. 2018.
[15] IETF. HTTP Strict Transport Security (HSTS). [s.l.: s.n.], 2012. Disponível em: <https://tools.ietf.org/html/rfc6797>. Acesso em: 5 jun. 2018.
[16] IETF. An Ethernet Address Resolution Protocol. [s.l.: s.n.], 1982. Disponível em: <https://tools.ietf.org/html/rfc826>. Acesso em: 5 jun. 2018
[17] IETF. DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. [s.l.: s.n.], 1987. Disponível em: <https://tools.ietf.org/html/rfc1035>. Acesso em: 5 jun. 2018.
[18] IETF. Hypertext Transfer Protocol -- HTTP/1.1. [s.l.: s.n.], 1997. Disponível em: <http://www.ietf.org/rfc/rfc2068.txt>. Acesso em: 5 jun. 2018.
[19] Exploit Database Statistics. Exploit-db.com. Disponível em: <https://www.exploit-db.com/exploit-database-statistics/>. Acesso em: 5 jun. 2018.
[20] IETF. IPv4 Address Conflict Detection. [s.l.: s.n.], 2008. Disponível em: <https://tools.ietf.org/html/rfc5227#section-5>. Acesso em: 5 jun. 2018.
[21] XI, HUIXING. Research and Application of ARP Protocol Vulnerability Attack

and Defense Technology Based on Trusted Network. [s.l.: s.n.], 2017. Disponível em: <https://aip.scitation.org/doi/pdf/10.1063/1.4977403>. Acesso em: 5 jun. 2018.
[22] GANGAN, SUBODH. A Review of Man-in-the-Middle Attacks. [s.l.: s.n.], 2015. Disponível em: <https://arxiv.org/pdf/1504.02115.pdf>. Acesso em: 5 jun. 2018.
[23] SANS INSTITUTE. Real World ARP Spoofing. [s.l.: s.n.], 2003. Disponível em: <https://pen-testing.sans.org/resources/papers/gcih/real-world-arp-spoofing-105411>. Acesso em: 5 jun. 2018.
[24] NACHREINER, COREY. Anatomy of an ARP Poisoning Attack. [s.l.: s.n.], 2012. Disponível em: <http://csci6433.org/Papers/Anatomy%20of%20an%20ARP%20Poisoning%20Attack%20_%20WatchGuard.pdf>. Acesso em: 5 jun. 2018.
[25] GILBERT, B.: “Cuphead: Bringing 1930s style to 21st century games ”, Engadget.com, 2014. Disponível em: <https://www.engadget.com/2014/07/10/cuphead-interview/ >. Acesso em: 23 jul. 2018.
[26] SOFTWARE DEVELOPMENT KIT. In: Wikipédia: a enciclopédia livre. Disponível em: <https://en.wikipedia.org/wiki/Software_development_kit> Acesso em: 23 jun 2018.
[27] 000WEBHOST. Free Web Hosting. Disponível em: <https://www.000webhost.com/>. Acesso em: 12 jul. 2018.
[28] UNITY DOCUMENTATION. Skybox. Disponível em: <https://docs.unity3d.com/Manual/class-Skybox.html>. Acesso em: 12 jul. 2018.
[29] UNITY DOCUMENTATION. WWW. Disponível em: <https://docs.unity3d.com/ScriptReference/WWW.htm >. Acesso em: 12 jul. 2018.
[30] UNITY DOCUMENTATION. Splat Prototype. Disponível em: <https://docs.unity3d.com/ScriptReference/SplatPrototype.html >. Acesso em: 12 jul. 2018.
[31] Code Of Ethics - EC-Council. EC-Council. Disponível em: <https://www.eccouncil.org/code-of-ethics/>. Acesso em: 5 jun. 2018.
[32] OLSON, P. Pokémon GO's McDonald's Partnership Points To A Promising Business Model. Forbes, 2016. Disponível em: <https://www.forbes.com/sites/parmyolson/2016/07/20/pokemon-go-mcdonalds-japan-nintendo-revenue>. Acesso em 05 jul, 2018.

[33] ARSLAN, YUKSEL. A SOLUTION FOR ARP SPOOFING: LAYER - 2 MAC AND PROTOCOL FILTERING AND ARPSERVER. [s.l.: s.n.], 2017. Disponível em: <https://arxiv.org/pdf/1708.01302.pdf>. Acesso em: 5 jun. 2018.

[34] ISSAC, BIJU. Secure ARP and Secure DHCP Protocols to Mitigate Security Attacks. [s.l.: s.n.], 2008. Disponível em: <https://arxiv.org/pdf/1410.4398.pdf>. Acesso em: 5 jun. 2018.

[35] SECDEV. Scapy.net. Disponível em: <https://scapy.net/>. Acesso em: 5 jun. 2018.