UM CANAL ENCOBERTO PARA TRANSMISSÃO DE DADOS SOBRE O PROTOCOLO HTTP

Felipe Afonso Espósito

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientador: Luís Felipe Magalhães de Moraes

Rio de Janeiro

Agosto de 2013

UM CANAL ENCOBERTO PARA TRANSMISSÃO DE DADOS SOBRE O PROTOCOLO HTTP

Felipe Afonso Espósito

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____
Prof. Luís Felipe Magalhães de Moraes, Ph.D.

_____
Prof. Claudio Luis de Amorim, Ph.D.

_____
Prof. Márcio Portes de Albuquerque, Ph.D.

RIO DE JANEIRO, RJ - BRASIL
AGOSTO DE 2013

Espósito, Felipe Afonso

Um canal encoberto para transmissão de dados sobre o protocolo HTTP/ Felipe Afonso Espósito. – Rio de Janeiro: UFRJ/COPPE, 2013.

XV, 68 p.: il.; 29,7 cm.

Orientador: Luís Felipe Magalhães de Moraes

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2013.

Referências Bibliográficas: p. 56-61.

1. Canais Encobertos. 2. Segurança da Informação. 3. Extração de Dados. I. Moraes, Luis Felipe Magalhães de. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

"To live outside the law, you must be honest"
(Bob Dylan)

Agradecimentos

Primeiramente gostaria de agradecer ao meu pai, Alexandre Sérvulo L. Vaz Jr, que sempre me apontou com corretude o caminho certo a seguir, a minha mãe, Virginia Prata Afonso, que sempre me deu força todo o tempo, principalmente nos momentos difíceis que duvidei de tudo e de todos. Ao meu orientador e amigo, Luis Felipe M. de Moraes que por muitas vezes me tirou de enrascadas homéricas, onde amadureci e aprendi muitas coisas. A todos os meus professores que desde o jardim de infância me incentivaram e deram ferramentas para seguir em frente com os meus sonhos, em especial ao Prof. Henrique Cukierman que depositou um enorme voto de confiança em 2012. A Marita Maestrelli, que foi muito solicita e ajudou na obtenção dos dados que tanto enriqueceram esta dissertação. À FAPERJ e a REDERIO que financiaram parte dos meus estudos. Não poderia deixar de agradecer aos amigos que adquiri ao longo do percurso. Thiago Bemerguy, André Figueiredo, Vander Proença, Evandro Macedo e Cláudia Lima pelos momentos de descontração no laboratório, que foram vitais para manter a sanidade mental. Alejandra Klachkin, que embora seja de outro laboratório, sempre esteve presente e informando a galera sobre os acontecimentos na COPPE. Quero agradecer também a todos os meus amigos, Felipe Menezes, Felipe Fiorenza, Nairo Alves, Isadora Vianna, e em especial Patrícia Winiawer que ao longo destes anos insistiram no meu potencial, me ajudaram a manter a postura e não abaixar a cabeça nos momentos mais difíceis. Sem estas pessoas, este trabalho nunca teria sido concluído, e devo todo o meu sucesso a elas. Enfim a todos que colaboraram direta ou indiretamente para a produção deste trabalho, o meu mais profundo respeito e gratidão.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

UM CANAL ENCOBERTO PARA TRANSMISSÃO DE DADOS SOBRE O PROTOCOLO HTTP

Felipe Afonso Espósito

Agosto/2013

Orientador: Luis Felipe Magalhães de Moraes

Programa: Engenharia de Sistemas e Computação

Governos, empresas e pessoas, eventualmente, têm interesse em manter não só o conteúdo da comunicação secreto, como também a própria existência da mesma, já que em alguns casos a mera existência da comunicação é suficiente para levantar suspeitas e ocasionar ações mais contundentes. Canais encobertos (*Covert Channels*), ou dissimulados, tentam esconder a própria existência da comunicação, normalmente subvertendo outros canais de comunicação, de maneira diferente para a qual foram originalmente projetados. O HTTP ("*Hypertext transmission protocol"),* por ser um dos protocolos mais utilizados na internet, é um excelente candidato para a implementação de canais encobertos já que há muito tráfego usual para encobrir o tráfego encoberto. Os canais encobertos ativos de armazenamento no HTTP alteram propriedades e características básicas do cabeçalho, facilitando a sua identificação. Como o desenvolvimento de páginas web o caminho dos recursos não segue um padrão e podem ser explorados para a construção de canais encobertos. Esta dissertação propõe uma nova forma de canal encoberto utilizando recursos nativos do protocolo HTTP de forma a dificultar a criação de assinaturas de detecção.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A COVERT CHANNEL TO TRANSMISSION OF DATA USING HTTP PROTOCOL

Felipe Afonso Espósito

August/2013

Advisor: Luis Felipe Magalhães de Moraes

Department: Computer and Systems Engineering

Governments, companies and people eventually have an interest in keeping not only the contents of secret communication, but also the very existence of the same, since in some cases the mere existence of communication is enough to arouse suspicion and lead to more forceful actions. Covert, or concealed, try to hide the existence of communication, usually subverting other communication channels, in different ways for which they were originally designed. The HTTP ("Hypertext transmission protocol"), being one of the most widely used protocols on the Internet, is an excellent candidate for the implementation of covert channels as long to cover the usual traffic traffic overcast. The active storage covert channels in HTTP alter basic properties and characteristics of the header, facilitating their identification. As the development of web pages the way of resources does not follow a pattern and can be exploited for the construction of covert channels. This paper proposes a new form of covert channel using native features of the HTTP protocol in order to hinder the creation of detection signatures.

Sumário

Lista de Figuras

Lista de Tabelas

Abreviações

APT Advanced Persistent Threats

CCTT Covert Channel Tunneling Tool

CGI Common Gateway Interface

CPU Central Process Unit

DDoS Distributed Denial of Service

DoS Denial of Service

FIFO First In First Out

HTTP HyperTexting Transfer Protocol

IDS Instrusion Detection Systems

IP's Internet Protocol

IPCC InterPacket Covert Channel

IRC Internet Relay Chat

ISN Initial Sequence Number

LSB Least Significant Bit

MSB Most Significant Bit

OSI Open Systems Interconnection

PC Protocol Channel

PHCC Protocol Hopping Covert Channel

QoS Quality of Service

RFC Request For Comments

URI Uniform Resource Identifier

VPN Virtual Private Network

Capítulo 1

Introdução

A comunicação é um processo importante de troca de informações e quando o meio de transmissão não é confiável, é preciso garantir a segurança do que será transmitido. Confidencialidade, integridade e disponibilidade são características desejáveis para uma comunicação segura. Estas e outras características como autenticidade, irrefutabilidade e privacidade podem ser alcançadas, no caso de transmissões eletrônicas, com o emprego de técnicas como a assinatura digital, criptografia e esteganografia.

A criptografia visa por meio de uma transformação matemática, alterar forma da mensagem original de maneira que só possa ser recuperada por quem conhece a chave do secreta, enquanto a esteganografia visa camuflar a informação secreta dentro de uma outra informação aparentemente inofensiva aos olhos de quem não conhece o método para extrair a informação.

Antigamente a comunicação entre as partes se dava através de cartas e mensageiros. Hoje, a informação flui por meios eletrônicos e digitais, não só entre indivíduos, mas também entre processos e dispositivos eletrônicos espalhados ao redor do mundo, conectados a uma rede.

Quando se imagina uma comunicação segura é comum que se acredite que o processo criptográfico seja suficiente. No entanto, a criptografia só impede que o conteúdo original seja decodificado e compreendido por terceiros, baseando-se na complexidade para derivar a chave criptográfica, garantindo a propriedade da confi-

dencialidade da informação até que haja uma descoberta do algoritmo criptográfico ou poder computacional para descobrir a chave em um tempo mais eficiente. Governos, empresas e pessoas, eventualmente, tem interesse em manter não só o conteúdo da comunicação secreto, como também a própria existência da mesma, já que em alguns casos a mera existência da comunicação é suficiente para levantar suspeitas e ocasionar ações mais contundentes. Canais encobertos ou dissimulados tentam esconder a própria existência da comunicação, normalmente subvertendo outros canais de comunicação de maneira diferente o qual foi originalmente projetado.

O termo *"covert channel"* foi originalmente cunhado por Lampson [2] em 1973 no contexto de segurança de *mainframes*, quando dois processos em uma mesma máquina, um sendo executado em um nível de segurança mais alto, poderia vazar informações para um processo em um nível de segurança mais baixo, explorando recursos compartilhados como CPU, memória e armazenamento. Com o avanço das redes de computadores, tais processos já não se encontram em um único computador, mas sim em computadores interconectados.

1.1 Motivação e definição do problema

A informação adquiriu um valor estratégico e comercial inestimável nas últimas décadas e, em geral, é o objetivo primário de um ataque a sistemas computacionais. Em 2012, o roubo de dados foi considerado o principal temor entre os empresários de diversos setores, quando questionados sobre os danos causados por ataques cibernéticos[3].

Os canais encobertos podem ser utilizados de forma legítima para prover privacidade entre as partes comunicantes, como também podem ser utilizados de formas ilegais como a extração de dados provenientes de campanhas de espionagem digital, controle de máquinas zumbis ou computadores infectados por códigos maliciosos sob o controle de terceiros.

Canais encobertos ativos e de armazenamento atualmente descritos pela litera-

tura no protocolo HTTP utilizam *cookies*, caracteres invisíveis, redirecionamentos, alterações no cabeçalho ou no tempo de transmissão dos pacotes para codificar a informação secreta e até mesmo encapsular outros protocolos a fim de burlar as políticas de segurança da rede. Esses métodos são mais fáceis de serem desenvolvidos e em geral, por criar um padrão de alteração, são facilmente detectados e passíveis de serem eliminados com medidas como a normalização do cabeçalho e a eliminação de *cookies* entre outras, interrompendo o canal de comunicação e possivelmente expondo seus participantes.

Embora o tema seja de interesse em diversos aspectos econômicos e militares, a produção brasileira no que diz respeito aos canais encobertos é baixa. Em uma pesquisa realizada no Google Scholar[4], um buscador de artigos e produções científicas, pelas palavras chaves, "canais dissimulados", "canais ocultos" e "canais encobertos" foram obtidos apenas sete resultados que utilizam uma destas palavras chaves desde 2005. A busca foi realizada utilizando os três termos encontrados na literatura brasileira, por que não há um consenso da tradução a ser utilizada pelos pesquisadores.

Na tabela 1.1 podemos ver a quantidade de resultados da busca por *"Covert Channel"* em todas as línguas (primeira linha). Apenas em Chinês Tradicional[1] e em Português do Brasil (destacado em negrito) do ano de 2005 a 2012. A produção científica dos últimos anos principalmente por países como China e Estados Unidos sobre o tema mostram a preocupação, crescente sobre a utilização do ciberespaço como o quinto ambiente de guerra.

Idioma/Ano	2005	2006	2007	2008	2009	2010	2011	2012
Total	206	238	262	229	341	307	405	529
Chinês(zh-TW)	1	3	7	1	10	5	1	5
Português(Pt-BR)	**2**	**1**	**2**	**0**	**0**	**0**	**0**	**0**

Tabela 1.1: Produção literária sobre canais encobertos entre os anos de 2005 e 2012

A incidência destes tipos de ataques vem aumentando e serão necessárias novas

[1]Não foram consideradas outros dialetos. A busca foi feita pelo termo "*covert channels*" em inglês encontrando-os no *abstract*.

formas de proteger a informação. A criação de técnicas para a construção de canais encobertos ainda é um desafio devido às inúmeras possibilidades e cenários diversificados. Entender como estes canais são construídos e, como podem ser detectados é de suma importância para manter um estado seguro em uma rede de computadores e/ou aumentar o arsenal cibernético de uma nação.

1.2 Objetivos

O objetivo principal deste trabalho é criar um novo método de canal encoberto, utilizando apenas os recursos e características nativas do protocolo de *Hyper Texting Transfer Protocol (HTTP)* de forma que seja muito difícil a identificação e eliminação deste canal. Além disso, pretende-se implementar este novo método a fim de realizar medições do seu funcionamento em uma rede real. Sendo assim ficam expostos os seguintes objetivos específicos:

1. Propor um canal encoberto, que aproveita das características nativas do protocolo HTTP e que seja de difícil identificação e eliminação.
2. Realizar experimentos para obtenção de métricas quanto às principais características de canais encobertos: capacidade, discrição e robustez.
3. Avaliar o seu comportamento em um ambiente real.
4. Comparar a técnica desenvolvida com outras técnicas exploradas por ferramentas conhecidas, apontando as principais vantagens e desvantagens.

1.3 Contribuições

Através da elaboração deste trabalho, as seguintes contribuições serão alcançadas:

1. Uma nova técnica para a criação de canais encobertos no protocolo HTTP.

2. Obtenção de métricas e características deste novo método, possibilitando a sua identificação no futuro.

3. Vantagens e desvantagens da utilização desta técnica em contraposição as técnicas já conhecidas.

1.4 Organização do texto

No Capítulo 2, é feita uma revisão bibliográfica sobre canais encobertos. O capítulo também apresenta uma breve discussão sobre a terminologia empregada nesta dissertação, mostrando o estado da arte no que diz respeito aos canais encobertos em protocolos de rede e sua utilização na camada de aplicação com o protocolo HTTP.

No Capítulo 3 apresenta-se a nova técnica para a a construção de canais encobertos, a proposta de codificação , o projeto da aplicação e as considerações teóricas do novo canal utilizando o protocolo HTTP;

No Capítulo 4 são apresentados os resultados obtidos através da avaliação do método proposto obtendo medidas para caracterizá-lo e a comparação com outras ferramentas semelhantes.

Por último, no Capítulo 5 são tecidas as considerações finais sobre o canal proposto, uma breve revisão sobre os resultados, contribuições obtidas e perspectivas de trabalhos futuros que poderão ser realizados em virtude das contribuições aqui apresentadas.

Capítulo 2

Revisão da Literatura

Este capítulo apresenta os conceitos de canais encobertos e o dilema do prisioneiro, que é o problema base para canais encobertos, em quais cenários eles podem ser utilizados, possíveis aplicações e o estado da arte da utilização de canais encobertos em protocolos de rede, inclusive no protocolo HTTP.

2.1 Canais encobertos

Lampson definiu canais encobertos como "canais que não foram projetados para transferência de informação de forma alguma" [2]. O *Trustec Computer Systems Evaluation Criteria*, criado pelo departamento de defesa dos Estados Unidos, também conhecido como "*Orange Book*", define canais encobertos como "[...] qualquer canal de comunicação que pode ser explorado por um processo para transferir informação de maneira que comprometa a política de segurança do sistema"[5].

A terminologia utilizada no Brasil não é unificada. Dos poucos documentos produzidos na literatura brasileira, Chaves e Montes em [6] traduziram o termo *"Covert Channels"* para "Canais dissimulados", Geus por sua vez traduziu o termo como "Canais ocultos" ou "Canais cobertos" em [7]. Do dicionário em inglês temos a definição de *"Covert":*

"Covert" - Adjetivo: Encoberto, dissimulado escondido.

A palavra “Encoberto” do dicionário Michaelis, disponível online [8] diz:

Encoberto en.co.ber.to adj (part de encobrir) 1 Escondido, tapado, oculto. 2 Disfarçado. 3 Clandestino. 4 Incógnito. 5 Enevoado (o tempo).

A palavra “*encoberto*” dá a entender melhor o modelo de como são construídos estes tipos de canais que utilizam o tráfego existente para se esconder, cobrindo-se e camuflando ao tráfego existente na rede. Estar oculto é uma propriedade inerente ao canal e assim também pode ser entendido, mas para o bem do entendimento faz-se necessário adotar uma tradução única, sendo assim, “canal encoberto” foi o termo escolhido nesta dissertação como tradução de *“covert channels”*.

2.1.1 O dilema do prisioneiro

Considere que duas pessoas Alice e Bob estão encarcerados e pretendem fugir. Para criar um plano de fuga ambos precisam se comunicar, mas na prisão há um guarda que monitora todo tipo de mensagem.

Se o guarda encontrar qualquer indício de mensagens suspeitas, irá aprisionar Bob e Alice na solitária, tornando a fuga impossível. Alice e Bob devem trocar mensagens inocentes contendo informações escondidas que, por sorte o Guarda não irá notar. O guarda pode adotar três posturas ressaltadas por Zander [9]:

- O Guarda pode ser passivo, apenas observando a troca de mensagens, mas sem alterá-las.
- O Guarda pode se ativo, modificando ligeiramente o conteúdo das mensagens, mas sem alterar a semântica.
- O Guarda pode ser malicioso, podendo alterar o conteúdo das mensagens sem ser punido.

Handel *et al.* *apud* [1] estenderam este cenário para redes de computadores onde Alice e Bob usam a rede de computadores para comunicação trocando mensagens à primeira vista inofensivas, mas contendo um canal encoberto escondido.

Alice e Bob compartilham um segredo usado para determinar os parâmetros do canal encoberto e cifrar/autenticar a mensagem escondida. Alice se conecta com Bob através da rede utilizando um canal normal de comunicação seguindo um protocolo de rede. Este canal (aberto) é utilizado para encobrir o tráfego secreto entre Alice e Bob que decodificam o conteúdo encoberto a fim de engendrar seu plano de fuga.

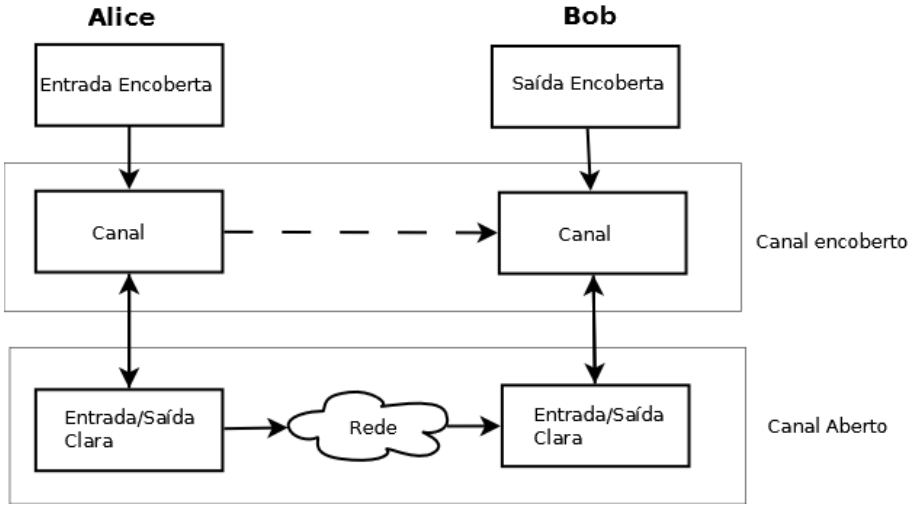


Figura 2.1: Modelo de canal encoberto adaptado de [8].

Para fins práticos Alice e Bob podem ser a mesma pessoa, por exemplo, um invasor extraindo informações confidenciais. O Guarda é o administrador da rede e monitora o tráfego em busca de alterações ou canais encobertos.

2.2 Cenários de comunicação

Uma comunicação secreta pode ocorrer em diferentes cenários, dependendo da atuação de Alice e Bob. Se Alice também é a emissora da comunicação aberta, ela pode manipular parâmetros para alterar, por exemplo, a capacidade ou a discrição do canal encoberto, embora, às vezes, ela não possa criar o canal aberto a bel prazer, ou pode escolher não fazê-lo para aumentar a discrição. Neste caso, Alice age como um intermediário, manipulando um canal previamente estabelecido. No lado oposto, Bob também pode atuar de duas formas, como o receptor e como um intermediário. Como um receptor sua tarefa é decodificar a informação secreta, mas, para aumen-

tar a discrição, Bob pode ser um intermediário decodificando a informação secreta e removendo-a do canal antes de passar adiante para o destinatário final. A Figura 2.2 ilustra as possibilidades de localização.

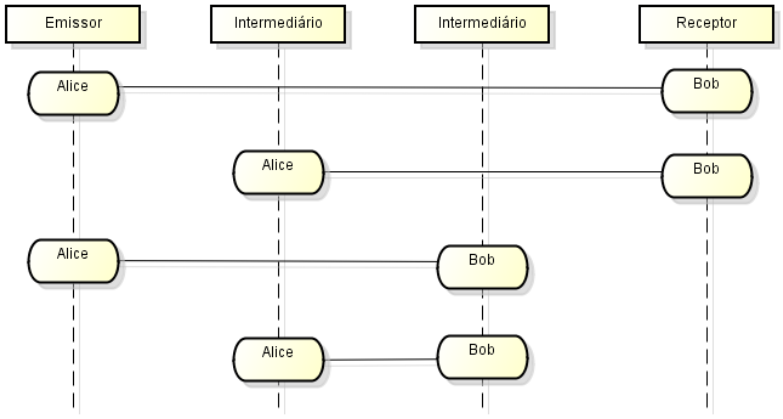


Figura 2.2: Possíveis cenários de comunicação, adaptado de [8].

Estar em uma posição intermediária não necessariamente significa que Alice ou Bob estão fisicamente separados do emissor ou receptor da comunicação. Alice pode, por exemplo, ser um módulo de *kernel*, agindo como um intermediário na linha de comunicação, manipulando a pilha de rede. A localização depende da aplicação que é utilizada para encobrir o canal. Por exemplo, se o canal é utilizado para extração de dados, utilizar Alice e Bob como intermediários torna o canal mais discreto e Bob poderia ser um roteador de borda da rede comprometida desviando as informações extraídas para o atacante.

2.3 Aplicações

Em alguns casos, canais encobertos podem ser utilizados para melhorar a segurança da rede. Yuan e Lutz por exemplo, construíram um canal encoberto modificando o protocolo TFTP para fazer fazer o download de um arquivo e do *checksum* simultaneamente, tornando possíveis a detecção de erros de transmissão[10]. Um canal de comunicação encoberto pode ser utilizado para a comunicação secreta entre duas partes, esta comunicação pode ocorrer desde um simples bate-papo para fugir dos

mecanismos de censuras, controle de uma rede de botnet, transferência de arquivos e roubo de informações.

2.3.1 *Mallware Callback*

É de interesse do atacante ser notificado quando um software malicioso se instala em um dispositivo com sucesso avisando que o dispositivo foi comprometido. Este processo é chamado de "*callback*".

O *callback* ou *call home* é o método que um processo malicioso tem para avisar ao seu controlador que o mesmo infectou o alvo, podendo enviar informações como endereço IP, que outros softwares estão instalados, versão do antivírus, velocidade de acesso à INTERNET entre outras informações que o controlador julgue interessante. Estas informações podem ajudar o atacante a criar uma *botnet* (Redes de computadores zumbis) ou utilizar esta máquina para o envio de spam (*e-mails* enviados sem o pedido do destinatário) e outras atividades em geral maliciosas [11].

2.3.2 Extração de dados

Extração de dados é o processo de extração de informações de um computador ou rede alvo. Na maioria dos casos, o acesso a uma informação privilegiada é o objetivo principal em um ataque cibernético e o principal temor dos empresários em 2012[3]. Planos estratégicos, estudos de reservas petrolíferas, informações pessoais como e-mail, nome, CPF, números de cartão de crédito estão entre os dados que podem ser revertidos em lucro pelos atacantes. O *modus operandi* genérico de uma invasão a uma rede de computadores se assemelha ao apresentado na Figura 2.3[12].

Figura 2.3: *Modus operandi* genérico de um ataque a uma rede de computadores.adaptado de [12]

Da Figura 2.3 temos:

1) *Footprinting* - Nesta etapa, o objetivo do atacante é obter o máximo de informações possíveis sem "tocar" no alvo utilizando empresas parceiras, balanço financeiro, termos técnicos, compras de *softwares*, sistemas operacionais, empregados. Tudo o que for possível descobrir através da INTERNET e/ou engenharia social.

2) Enumeração - Nesta etapa o principal objetivo é enumerar possíveis brechas, indivíduos, *sites*, endereços IP acessíveis na INTERNET e confirmar os dispositivos vulneráveis.

3) Ataque - Nesta etapa o atacante de fato obtém acesso não autorizado ao alvo, aprende o máximo possível sobre os sistemas alvo, e invade outras máquinas caminhando horizontalmente na rede alvo.

4) Manutenção do ataque - Na fase de manutenção o atacante comumente limpa os rastros do seu acesso, instala *backdoors* e *rootkits* e, possivelmente, até corrige as falhas que ele encontrou para evitar que outros invasores consigam acesso e prejudiquem as suas atividades.

5) Extração de dados - Nesta fase o objetivo do atacante é reunir a maior quantidade possível de informações tais como: manuais, senhas, dados pessoais , cartões de crédito e extrair estes dados da rede da companhia sem chamar a atenção dos administradores de rede.

Neste ponto a utilização de canais encobertos é um diferencial. Para um atacante, manter-se escondido é essencial para garantir que possa extrair toda a informação e controlar o alvo sem ser identificado pelo administrador.

Advanced Persistent Threats

Advanced Persitent Threat (APT) refere-se a um grupo de ataques digitais, geralmente patrocinados por governos ou grandes corporações, com a capacidade e intenção de perpetrar atividades maliciosas em um ou mais alvos específicos. Um dos grandes diferenciais destes ataques, é justamente a capacidade do atacante de obter

informações de inteligência, manter uma rede de peritos em diversas áreas de conhecimento e coordenar as invasões por períodos longos[13]. Casos como o *malware Stuxnet,* um *worm* descoberto em junho de 2010 e desenvolvido especialmente para infectar computadores Windows de controladoras das centrífugas de enriquecimento do programa nuclear Iraniano.

Outros ataques tem sido reportados por grandes corporações, como o episódio que ficou conhecido como "*Operation Aurora*", do qual empresas como Google, Yahoo, Adobe entre outras tiveram as suas redes comprometidas seguidas de roubo de informações confidenciais. A Mandiant, expôs em seu relatório sobre APTs em 2012, que alguns *backdoors* estão utilizando técnicas de canais encobertos que imitam o tráfego de aplicações HTTP legítimas como o MSN Messenger, GMail e o Calendar do Google para o controle remoto de servidores[13].

2.3.3 *IP Traceback*

O objetivo de um metodo de IP Traceback e identificar o endereço de origem real do fluxo de pacotes ou reconstruir o caminho do ataque na rede, especialmente em casos de ataques de negação de serviço e ataques de negação de serviços distribuídos (DoS e DDoS) porque permite que o tráfego seja filtrado a partir de uma origem em comum facilitando a criação de regras para prevenir/diminuir a carga do ataque. Vários pesquisadores desenvolveram formas de rastrear pacotes de dados[14] utilizando inclusive, técnicas de canais encobertos [15].

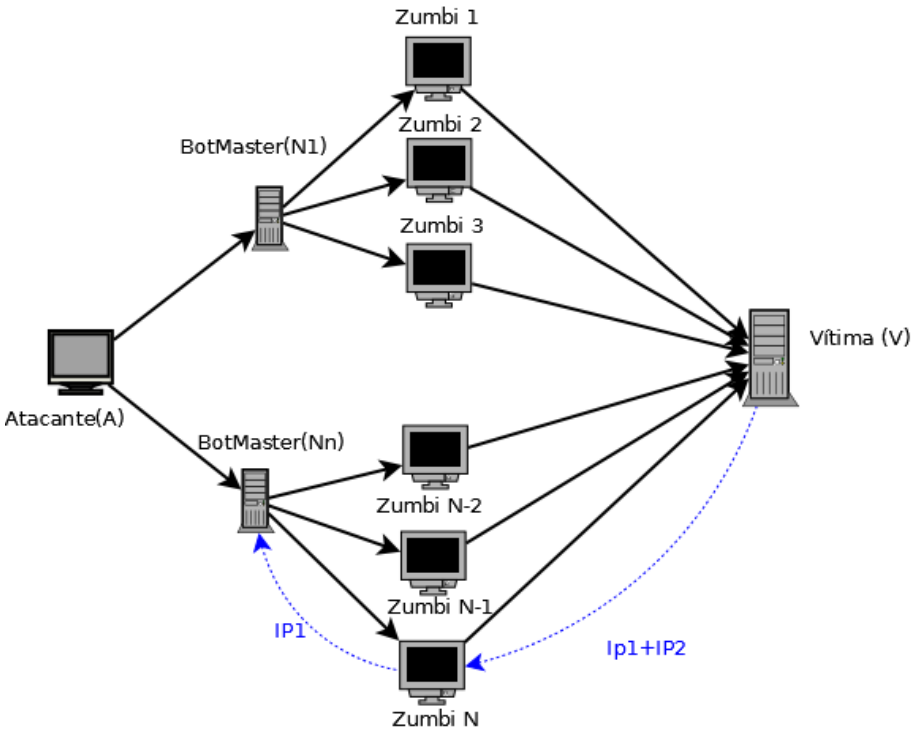


Figura 2.4: IPTraceback adaptado de [15]

Na Figura 2.4, temos um atacante(A) controlando uma *botnet* composta por dois *botmaster*s, cada *botmasters* controla um conjunto de computadores Zumbis. Ao ordenar um ataque contra a vítima, o comando é primeiramente enviado ao *botmaster*, que replica para o exército de zumbis. Estes por sua vez atacam a máquina da vítima.

Considerando que ao menos em uma dessas redes seja utilizado canais encobertos para manter dentro dos pacotes informações que digam por quais nós da rede este tráfego passou (Azul). É possível encontrar uma origem comum ao ataque e criar uma regra no *firewall* para bloquear qualquer informação que tenha, por exemplo, passado por um *botmaster* em comum a outros zumbis, diminuindo o impacto do ataque [15].

2.4 Características

Para caracterizar canais encobertos três características foram propostas por Zander na sua tese de doutorado [9] são elas: robustez, discrição e capacidade.

2.4.1 Robustez

Essa característica define o quão resistente um canal é à sua eliminação e/ou diminuição de sua capacidade por interferências possivelmente introduzidas intencionalmente por um terceiro [1].

2.4.2 Discrição (Stealth)

A discrição é a característica que diz o quão fácil é detectar o canal encoberto. Uma forma é comparar as características do tráfego com o canal encoberto e sem o canal encoberto [16, 1]. Um canal encoberto, completamente discreto, não apresenta nenhuma diferença ao tráfego normal, sendo semelhante em suas propriedades estatísticas e de comportamento. Uma métrica proposta na literatura para a quantificação desta propriedade consiste no módulo do coeficiente de correlação entre o tráfego encoberto e o tráfego real, que varia de 0, nenhuma, à 1 onde há correlação total entre as características avaliadas.

2.4.3 Capacidade

A capacidade é determinada pela taxa com que a informação é transmitida sem erros. Em geral a capacidade é medida em bits por segundo, mas para canais encobertos também pode ser expressa em bits por pacote, já que muitas vezes a capacidade não está diretamente relacionada com o meio de comunicação, mas sim ao protocolo de rede em unidades fechadas chamadas de pacote.

A capacidade em bits por pacote é definida por $C = \frac{Q_b}{Q_p}$, onde Q_b é a quantidade de bits a serem transmitidos e Q_p é a quantidade de pacotes necessários para a transmissão dos bits.

As três características divergem entre si, pois não é possível maximizá-las. Por exemplo, não é possível aumentar a capacidade sem que haja perda na discrição uma vez que ao transferir uma enorme quantidade de informação, o canal encoberto irá se diferenciar do tráfego que o encobre. De forma análoga, para aumentar a robustez de um canal, é necessário incluir mecanismos de confiabilidade o que em geral implica em mecanismos para retransmissão e verificação, gastando bits a mais e, portanto diminuindo a capacidade do canal [9, 17].

Brown *et al.* descrevem outras caraterísticas como:

- Mecanismo, que especifica como o canal é construído e o que ele faz para esconder e carregar a informação. O mecanismo utilizado para esconder a informação é o que diferencia um canal do outro e influencia em outras características como o tipo e a discrição [10];

- Tipo é como são classificados os canais encobertos. Algumas taxonomias foram propostas e é uma prática comum classificar os canais nos seguintes tipos: Armazenamento, Temporais, Híbridos, e Protocolos que serão melhor explicados na seção 2.5;

- Prevenção, cada canal também pode ser avaliado quanto a sua habilidade de ser prevenido ou interrompido. A prevenção se diferencia da robustez. Enquanto a robustez é a habilidade do canal persistir a circunstâncias naturais, a prevenção, por outro lado, é a habilidade do canal permanecer quando alguma ação explicita é tomada para romper ou degradar o canal encoberto [10].

Embora haja diferença da taxonomia proposta por Brown e por Zander, nesta dissertação, a definição da característica de robustez será a definida por Zander, que incorpora as definições de prevenção e robustez definidas pro Brown.

2.5 Taxonomia de canais encobertos

Canais encobertos podem ser classificados quanto ao mecanismo utilizado para encobrir o tráfego oculto, estes mecanismos diferenciam entre si e são descritos a seguir:

2.5.1 Armazenamento

Os canais encobertos que utilizam um ou mais campos do cabeçalho de um protocolo para esconder a informação são chamados de canais de armazenamento ou *storage (storage covert channels)[17, 18, 19].*

Há na literatura uma discussão entre as definições de esteganografia e canais encobertos no âmbito de redes de computadores. Em geral, esconder a informação na carga de um pacote pertence ao escopo da esteganografia e canais encobertos são, em geral, caracterizados por esconder a informação nos campos do cabeçalho, no tempo de transmissão dos pacotes ou em outras características como o tamanho dos pacotes[20][21]. Nair *et. al.* utiliza o termo esteganografia de rede como sinônimo para canais encobertos. Nesta dissertação, o termo esteganografia será utilizado quando o conteúdo secreto for transmitido alterando o conteúdo da carga do pacote, concordando com a definição de Zander [1]. Por exemplo, o protocolo IP possui um campo de cabeçalho *Fragment offset* que é utilizado para ordenar um pacote que foi fragmentado, e que pode ser preenchidos com treze bits de dados. Neste caso dá-se o nome de canal encoberto. Utilizar a carga do protocolo IP para encapsular um conteúdo disfarçado atende ao domínio da esteganografia.

Obviamente nem todos os campos são passíveis de serem utilizados para esconder dados, não é plausível alterar o conteúdo do campo endereço de origem (IP.src) com trinta e dois bits de informação e esperar o retorno correto da conexão. Um administrador de rede, observando uma grande quantidade de cabeçalhos IP com o *fragment offset* repleto de bits logo suspeitará que algo está errado no ambiente de rede. Por estes motivos, desenvolvedores de *Covert Channels* tendem a escolher

um campo com dados aparentemente aleatórios. Em *"Embedding Covert Channels into TCP/IP"* [22] Murdoch e Lewis investigaram diversos campos no TCP/IP que seriam interessantes para armazenar dados, podemos destacar no protocolo IP o campo ID cujo valor é pseudo-aleatório e no TCP os campos *Sequence Number* e *TimeStamp* que podem ser visualizados na figura2.5. Outros campos, embora passíveis de mudança seriam facilmente detectados como por exemplo o campo de *flags* com uma ou várias ativadas sem necessidade.

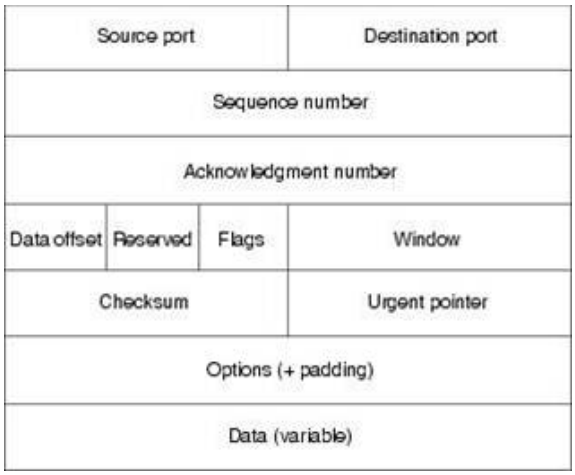


Figura 2.5: Diagrama Protocolo TCP

Outros protocolos e propriedades também podem ser utilizados para a criação de canais encobertos. Por exemplo, Lucena *et. al.* [23] investigaram os campos do protocolo IPv_6 que podem ser utilizados para a construção de canais encobertos.

Nair *et. al.* utilizaram outro método: modular o tamanho dos pacotes e aplicaram esta técnica no protocolo UDP. Para os experimentos, os autores utilizaram um programa de bate-papo (*chat*) que troca mensagens de texto através do protocolo UDP, é importante frisar que todo o tratamento de erro e recuperação é feito pela aplicação de bate-papo. A razão da utilização de um programa de bate-papo é que o tamanho dos datagramas são aleatórios por natureza, ou seja segundo a literatura sobre esteganografia, sequências relativamente aleatórias são melhores para esconder mensagens já que a aleatoriedade pode esconder a distorção causada pelas informações inseridas [21].

2.5.1.1 Nushu

O Nushu é um canal encoberto do tipo armazenamento, criado pela pesquisadora de segurança Joanna Rutkowska em [24] apresentado no congresso *"Chaos Communication Congress"*, um congresso informal organizado por hackers do *Chaos Computer Club* [25] e estudado em diversos artigos acadêmicos [22, 24, 26].

O Nushu é um módulo do *kernel* do Linux. Quando uma conexão TCP é iniciada, o *kernel* do Linux gera um valor aleatório para o campo ISN (*Initial Sequence Number*) para a realização do *three-way handshake* do protocolo. O Nushu, intercepta este valor, antes de enviar para o dispositivo de rede, substituindo o valor deste campo por outro valor contendo a informação secreta [24, 26]. Para manter a aparente aleatoriedade do campo e aumentar a segurança da informação contida no canal, o Nushu criptografa o conteúdo da mensagem com o algoritmo DES aumentando assim a aleatoriedade e a segurança do canal encoberto.

2.5.2 Temporais

Os canais encobertos temporais (*timing*) manipulam o intervalo de tempo entre os pacotes para a transferência de informações. Um exemplo é o *Inter-packet covert channels*(IPCC) que modula a informação ser transmitida de acordo com o tempo de envio e chegada dos pacotes [27, 28]. Considere que pacotes que chegam ao endereço de destino em um intervalo de tempo t são sinalizados com o bit 1, e para pacotes que cheguem com o intervalo de tempo igual a $2t$ são sinalizados com o bit 0. Para a transmissão da informação oculta, o emissor da mensagem manipula o tempo de transmissão entre os pacotes de acordo com a informação a ser transmitida. Na Figura 2.6 podemos ver um exemplo de um canal do tipo *timing* no protocolo HTTP, enviando uma sequência de bits. Para o correto funcionamento deste tipo de canal, emissor e o receptor devem combinar previamente o intervalo de tempo para que saibam a diferença que caracteriza cada tipo de bit e suas variações (devido a condições da rede) [27].

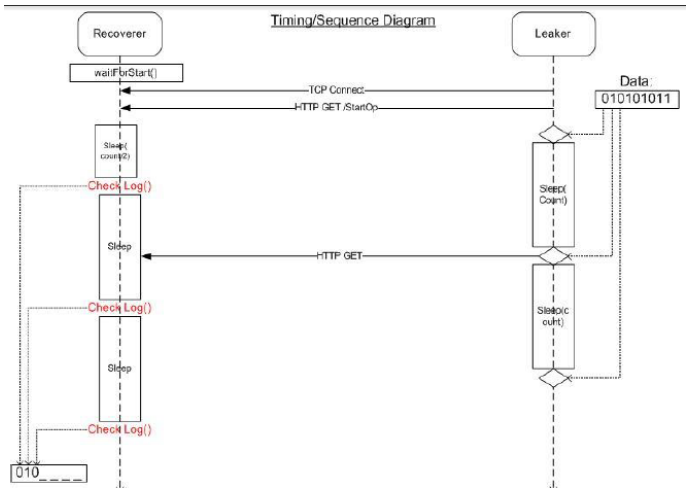


Figura 2.6: HTTP *timing covert channel* de [10]

Os canais temporais estão sujeitos às condições da rede. Se o tempo acordado entre as partes não for bem definido, o canal pode dessincronizar, ou seja, a informação não chegará ao destino de maneira correta [29]. Em [29] Liu *et. al.* observam que as condições da rede como *jitter* e atraso podem diminuir a capacidade deste tipo de canal ou causar a sua ruptura.

Em redes de computadores também podem ocorrer a perda de pacotes em algum segmento. Essa perda prejudica o desempenho de canais encobertos baseados na diferença de tempo entre os pacotes, pois se um pacote é perdido o intervalo do tempo de chegada entre dois pacotes é alterado sem que o receptor saiba, dessincronizando o canal encoberto [30].

2.5.3 Protocol channels

Canais encobertos do tipo protocolo são capazes de alterar o protocolo de rede utilizado para encobrir o conteúdo secreto. A primeira prova de conceito foi encontrada em 1997, uma ferramenta chamada LOKI2 disponibilizada na em uma revista chamada Phrack. O LOKI2 era capaz de alterar entre o ICMP e o UDP *apud* [31]. Wendezel e Keller descrevem dois métodos de canais encobertos, o chamado Protocol Switching Covert Storage Channels também conhecidos como Protocol Hopping Covert Channels (PHCC) e o Protocol Channels (PC).

O PHCC foi apresentado pela primeira vez na revista norte americana de segurança Hackin9[32]. Estes canais transferem a informação usando campos de protocolos de redes diferentes para não chamar atenção, dividindo o comportamento peculiar em mais de um protocolo, diminuindo a variação estatística em um único protocolo, por exemplo utilizar o *User-Agent* do HTTP com o RETR do POP3 (*e-mail*) para transferência de informações.

Os chamados Protocol Channels (PC), alternam entre dois ou mais protocolos para a transmissão da informação. O principal objetivo é que o pacote não seja alterado, tornando mais dificil a sua detecção. Os protocolos utilizados estão atrelados a um valor secreto, por exemplo, na Figura 2.7 o HTTP tem valor "01", o POP3 "10" e para transmitir por exemplo a sequência 010110 o emissor emitiria dois pacotes HTTP e um POP3. A capacidade deste canal é geralmente limitada a algumas centenas de bits/segundo, no entanto é rápido o suficiente para transferir alguns nomes de usuário e senhas.

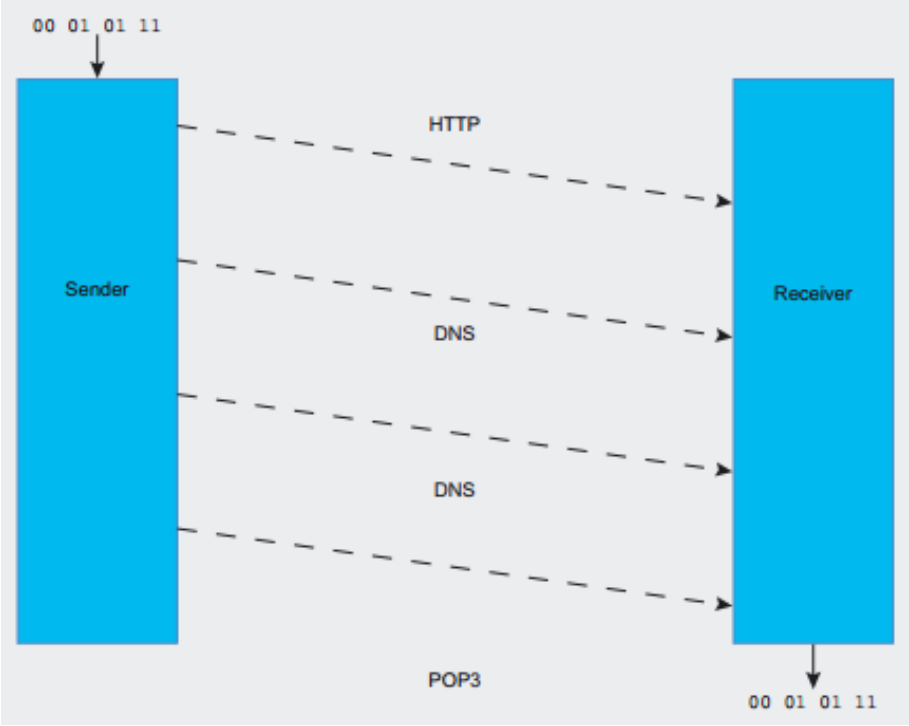


Figura 2.7: Canal encoberto utilizando "Protocol channel" para transferência de bits [32]

2.5.4 Híbridos

Wendzel e Keller introduzem em [31] o conceito de canais encobertos híbridos que combinam duas ou mais classes de canais encobertos em um único, argumentam ainda que canais híbridos podem ser mais difíceis de serem detectados já que as possíveis combinações entre tipos de *covert chanels* aumenta a dificuldade de identificação de um canal [31, 33].

2.5.5 Ativos e Passivos

Existem dois cenários de comunicação para transmissão de informação através de canais encobertos em uma rede real. No primeiro cenário, chamado canal encoberto ativo, o emissor da informação secreta está posicionado na fonte dos pacotes gerando o seu próprio tráfego. No segundo, canal passivo, o emissor da informação secreta está posicionado em um nó intermediário utilizando um canal de comunicação já estabelecido para inserir informações encobertas, que serão decodificadas em outro ponto por um receptor que está manipulando o tráfego passante para recuperar a informação secreta como visto na Figura 2.8.

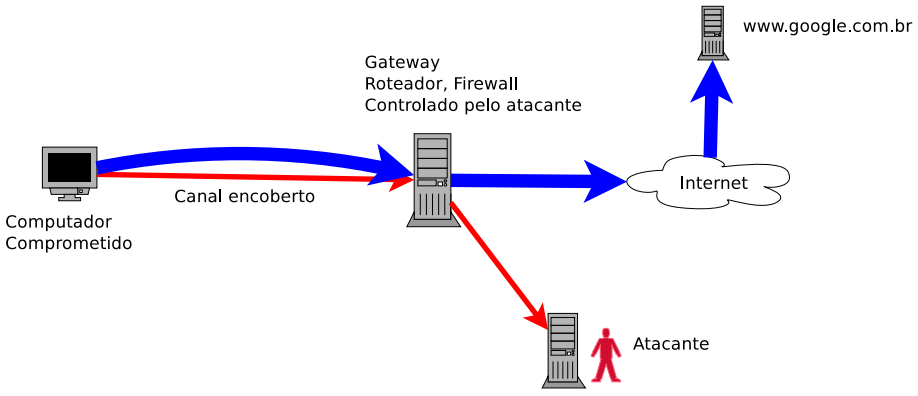


Figura 2.8: Exemplo de canal coberto passivo adaptado de [1]

Canais encobertos passivos tendem a ser mais discretos que os canais ativos já que, não geram seu próprio tráfego utilizando o tráfego já existente na rede gerados por processos ou dispositivos para inserir a sua informação. Todavia, para a extração

de dados de uma rede, os canais passivos exigem que o atacante possua o controle de um *gateway* próximo ao alvo [28].

A transmissão de informações através de canais encobertos passivos deve ser feita de maneira cautelosa. A variação do tráfego e das condições da rede são inevitáveis, e o emissor do canal secreto está sujeito a duas situações extremas. A primeira ocorre quando os pacotes chegam a um intervalo de tempo muito curto, se o canal é temporal e a diferença de tempo acordada é alta por exemplo, força os pacotes a serem armazenados na memória (*buffer*) do dispositivo de rede e quando não puder mais armazenar os pacotes entra em estado de estouro de memória (*buffer overflow*) e passa a descartar os pacotes que continuam chegando. Em contraposição, pode ocorrer a exaustão de pacotes (*starving*) quando a taxa de chegada dos pacotes no dispositivo é menor que a taxa de saída, esvaziando a memória do dispositivo forçando a ficar um tempo maior que o esperado sem enviar pacotes dessincronizando a estrutura do canal encoberto temporal. Este fenômeno também pode ocorrer em canais passivos do tipo armazenamento, porém o efeito é menos prejudicial pois diminui a capacidade do canal em bits por segundo enquanto o canal espera por um pacote para continuar a transmissão secreta de informação e no pior dos casos, a transmissão incompleta da informação.

2.5.5.1 Nushu

O Nushu se comporta de maneira passiva, ou seja, ele não gera o seu próprio tráfego, utiliza o tráfego gerado no *kernel* e o intercepta antes de chegar à placa de rede atuando como um intermediário no lado emissor. Para manter a conectividade do TCP junto ao kernel, ao interceptar o ISN, o Nushu guarda o ISN original em uma variável (ISN_orig), criptografa parte da informação a ser enviada e gera um novo valor de ISN, enviando para a rede. Quando um pacote TCP é recebido, o Nushu intercepta novamente o pacote, subtrai do valor do número de sequência (SEQ#) o valor do ISN gerado pelo Nushu e substitui por um SEQ# que teria sido gerado pelo ISN original, mantendo o estado da conexão TCP para o kernel e consequentemente

a aplicação [26].

2.5.6 Ruídosos e sem ruídos

Os ambientes onde canais encobertos podem ser utilizados também podem ser divididos canais sem e com ruídos. Um canal de comunicação dito sem ruído é aquele onde a informação enviada é a mesma recebida pelo receptor com probabilidade igual a um, em um canal de comunicação onde a probabilidade de recepção do canal é diferente de um, é necessário que os participantes façam o tratamento do ruído que pode gerar retransmissão e diminuir a capacidade do canal. Os ruídos são facilmente observados em canais temporais, o atraso e o *jitter* prejudicam a capacidade e a robustez do canal encoberto. Um canal encoberto do tipo armazenamento, dependendo do campo e do protocolo utilizado pode ser considerado com um canal sem ruído. Tomemos como exemplo o protocolo ICMP, que possui um campo TTL (*Time To Live*) que é decrementado a cada nó da rede que passa. Um canal encoberto que utilize este campo para transmitir a informação poderá sofrer ruído caso a rota do pacote seja alterada e tenha de passar por mais nós da rede do que o originalmente previsto.

2.6 Canais encobertos no HTTP

Por mais restritivo que um ambiente seja, é bem provável que uma empresa libere o tráfego HTTP para a internet, permitindo o acesso a sites. Por ser um protocolo muito utilizado em aplicações web o HTTP é um protocolo conveniente para a implementação de canais encobertos porque há muito tráfego para encobrir. Diversas ferramentas foram criadas utilizando o HTTP para encapsular outros protocolos ou contornar medidas de segurança ou censura, por exemplo, Feamster propôs um canal encoberto chamado Infranet [34] para transpor mecanismos de censura na web. Os servidores que participam da Infranet recebem requisições escondidas em imagens

aparentemente inocentes criando um túnel criptografado que fornece confidencialidade para os participantes. [34].

Existem outros métodos para criar canais encobertos utilizando o HTTP, por exemplo os mecanismos de redirecionamentos previstos na RFC, normalmente é utilizado para redirecionar um cliente a uma nova URL. Quando a nova URL é um *script* CGI, é possível a inclusão de parâmetros na em uma *string* de consulta para a nova URL (ex: WWW.SITE.COM.BR/INDEX.PHP?Q=CONSULTA). Estes parâmetros especificados na consulta são o conjunto de dados que são secretamente transferidos de um servidor web para outro através de um cliente. Quando um redirecionamento é enviado para um cliente (*Browser*) muitas vezes irá reconhecer e redirecionar para a nova URL que pode inclusive ser mostrada na barra de endereços do *browser*. Um usuário observador e com conhecimento do sistema pode notar a presença de parâmetros adicionais na *string* de consulta e alertar o administrador. Também é bastante comum sistemas de detecção de intrusão analisarem as URLs aumentando a probabilidade de detecção[35].

Uma outra funcionalidade comumente utilizada na construção de aplicações web, é a utilização de *cookies,* estes foram projetados para melhorar a experiência de um usuário na internet lembrando algumas informações. Os valores armazenados podem ser acessados não apenas pelo servidor que os criou, mas também por qualquer outro servidor no mesmo domínio que foi definido junto ao *cookie*. Serviços de DNS dinâmicos como o dydns.org permitem diversos servidores serem localizados no mesmo domínio, permitindo a troca de *cookies* entre eles, o que pode ser explorado para a criação de um canal encoberto[35]. Uma aplicação conhecida que utiliza *cookies* é o Google Analytics este serviço provê estatísticas de acesso sobre *websites*. O Analytics esta presente em mais de 12 milhões de sites, incluindo 57% dos top 10.000 sites visitados[36] o que torna um bom meio para um *covert channel* já que uma grande quantidade do tráfego web contem *cookies* do Google Analytics. É difícil criar uma base de conhecimento de valores esperados para os *cookies* do Google Analytics. Para detectar o canal encoberto proposto por Huba et al. uma

forma seria a proibição da utilização de cookies, o que poderia prejudicar outras aplicações legítimas.

Outro método utilizado para subverter o protocolo HTTP é a utilização do atributo *"Referer"*. Requisições HTTP do tipo POST podem conter o atributo opcional "*Referer*", que informa ao servidor a página anterior ao qual o novo recurso se refere, e que pode ser manipulado para conter informações arbitrárias em vez da página referida.

Também é possível utilizar elementos de Metatag, para a criação de canais encobertos. Nesta técnica, elementos do Metatag enviados ao *browser* de um servidor podem ser usados para especificar uma nova requisição para outro servidor web e assim passar informações para outro servidor dentro desta requisição. O exemplo comum é a utilização da tag <META HTTP-EQUIV> que pode ser incluída no corpo de uma mensagem HTTP. A capacidade deste canal é alta, limitada apenas pela quantidade de elementos inseridos. Poucos ou talvez nenhum navegador por padrão removem elementos Metatag no conteúdo do HTTP. Impedir a utilização deste canal secreto é relativamente complicado, sendo necessário analisar os pacotes HTTP em tempo de execução, o que é custoso e aumenta o tempo de resposta para o usuário[37].

Em 1999, Estrada *et. al.* desenvolveram um canal encoberto na camada de aplicação para evitar filtros de pacotes, analisadores de rede e motores de busca. Para tal utilizaram o protocolo HTTP, modificando ligeiramente o valor do campo "*User-Agent*", que informa a versão do *browser* utilizada. Para que o servidor possa reconhecer que o cliente não é um *browser* comum mas sim um modificado com o canal encoberto. Existem outras técnicas, como a alteração de ordem dos campos no cabeçalho, utilização de letras em caixa alta, manipulação dos dados utilizando caracteres invisíveis para o usuário como espaço, quebras de linha(\n) e tabulação (\t).[37].

Também é possível a implementação de canais temporais no protocolo HTTP o primeiro foi desenvolvido por Brown(2007) e provavelmente e o único canal en-

coberto baseado no tempo utilizando o HTTP. No entanto pode ser considerado como um caso especial do TCP/IP reportado em outros artigos.O método consiste em definir um tempo de espera entre duas requisições, a diferença de tempo entre estas requisições podem ser interpretadas como 0 e 1. Neste modelo, a informação só pode ser transferida unidirecionalmente, do cliente para o servidor. Tão qual no protocolo TCP/IP canais encobertos baseados no tempo, estão sujeitos às condições de rede[35].

Bauer ainda propôs criar um canal encoberto de acordo com o padrão de navegação de um usuário que ao navegar em um site acessa páginas de um site em um determinado padrão para obter notícias, imagens vídeos do seu interesse. Se for definida para cada página visitado um valor em binário, a alteração no padrão de visitação das páginas pode criar um canal de comunicação. Este canal em teoria é possível mas até a data atual não há notícias de sua implementação. Um aspecto interessante deste canal é que nenhum software especial é necessário para o seu funcionamento, apenas o conhecimento da estrutura do site e o valor previamente atribuído a cada página.

Aplicações WEB e Redes Sociais também podem ser utilizadas para a criação de canais encobertos. O FaceCat por exemplo, foi desenvolvido por José Selvi como prova de conceito e utiliza recursos do Facebook para criar um canal encoberto, se passando por um usuário normal, podendo inclusive controlar remotamente um computador através de um *trojan*[38].

2.7 Contra medidas

Segundo Zander [1], as seguintes contra medidas podem ser utilizadas contra canais encobertos:

- Eliminação do Canal
- Limitar a capacidade do canal

- Detectar e auditar o canal
- Documentar o canal

Estas medidas visam aumentar a segurança do ambiente e podem ser tomadas nas estações de trabalho e no ambiente de rede.

Estação de trabalho Aumentar o nível de segurança das estações de trabalho não remove canais encobertos, mas pode dificultar a exploração em alguns cenários [9]. Por exemplo, manter serviços seguindo a política de menos privilégios possíveis, se um atacante ao explorar uma vulnerabilidade em uma máquina não puder ter todas as permissões para por exemplo, instalar um modulo de kernel, o mesmo não poderá instalar ferramentas que utilizem *covert channels* diminuindo a superfície de exploração.

Segurança de Rede Uma medida a ser tomada é a restrição de protocolos ativos na rede. Se não existem serviços utilizando ICMP por exemplo ou IPv_6, estes protocolos não devem ser permitidos e podem ser filtrados em um *firewall*, diminuindo a quantidade de vetores a serem explorados por canais encobertos. O projeto de rede também deve ser levado em consideração quanto à proteção da informação. Servidores e serviços com informações de alta confidencialidade não devem ter comunicação direta com servidores externos.

2.7.1 Métodos de detecção

Existem diversas técnicas para a eliminação dos canais encobertos [39], todavia é mais interessante do ponto de vista da segurança a detecção antes mesmo da eliminação. Primeiramente, detectar a utilização de um canal indevido evidencia o comprometimento de uma ou mais máquinas. Segundo que após a sua correta identificação, pode-se comprometer a privacidade dos participantes e até mesmo o conteúdo embora, muitas vezes, além de oculto este tráfego pode estar criptografado.

O terceiro ponto importante é que se a comunicação ocorre entre dois pontos, podemos determinar tais pontos e, com sorte, achar os responsáveis, tomando medidas preventivas.

2.7.1.1 Assinatura

Após a análise do tráfego de rede de um *covert channel* conhecido e documentado, pode ser possível a criação de uma assinatura a ser utilizada para a sua futura identificação. Esta assinatura deve ser utilizada para a criação de regras de sistemas de detecção ou prevenção de intrusão e se a assinatura for detectada dentro da rede, tais sistemas podem inibir ou emitir um alerta para o administrador avisando sobre a existência do canal no ambiente de rede.

2.7.1.2 Estatísticas

É possível distinguir o trafego legítimo de um tráfego encoberto através de medidas estatísticas. Uma das maneiras de se realizar esta comparação entre um tráfego normal e o encoberto é realizando o teste Kolmogorov-Smirnov. O teste é independente da distribuição, e é aplicável a diferentes tipos de tráfego com diferentes distribuições e já foi empregado com sucesso na detecção de canais encobertos baseados no tempo [27].

Nushu Como visto anteriormente, o Nushu altera o valor do ISN e do SEQ# no protocolo TCP para carregar informações confidenciais. Lewis e Murdoch, ao analisar estatisticamente os valores do ISN gerados pelo *kernel* do Linux não modificado e comparar aos valores gerados pelo Nushu, descobriram que os valores gerados pelo Nushu seguem uma distribuição mais uniforme que o *kernel* original, tornando o tráfego do canal diferenciado e permitindo a sua identificação.

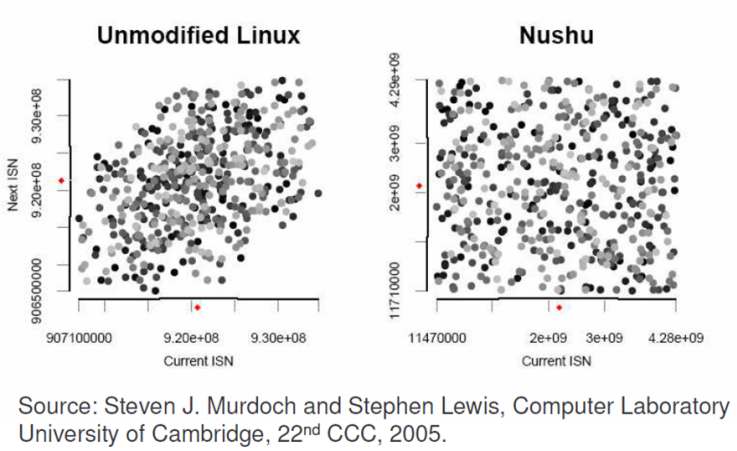


Figura 2.9: Distribuição dos valores de ISN gerados pelo kernel do Linux 2.6.x(esquerda) e por um kernel modificado pelo NUSHU (direita) fonte [22]

2.7.1.3 Redes Neurais

Uma rede neural é capaz de criar um modelo a partir de dados empíricos, mesmo sem nenhuma informação prévia do algoritmo de geração destes dados [26]. As redes neurais também possuem a capacidade de generalizar, respondendo corretamente a um resultado se a entrada for parecida com o treinamento, corrigindo erros mesmo que o dado de entrada esteja corrompido ou contenha erros. Estas propriedades são obtidas durante a fase do treinamento que é um processo interativo e muitas vezes supervisionado, o que pode levar a uma demora no processo de aprendizado dependendo do conjunto de treinamento.

Nushu Tumoian [26] propôs a utilização de redes neurais em um sistema de detecção baseado em identificação no algoritmo de geração do ISN, de fato o sistema pode ser treinado para reconhecer a presença de um canal escondido. Foram utilizados no conjunto de treinamentos capturas de pacotes com o TCPDump de 20.000 pacotes do Fedora Core2, Fedora Core2 modificado com o Nushu, Fedora Core3 , Windows 2000 e Windows XP. Cada arquivo contém 20.000 entradas de cada sistema operacional, sendo 1500 utilizados no conjunto de treinamento da rede neural e o restante utilizado para teste. Foi utilizada a rede neural recorrente de Elman em diferentes tamanhos (30x32, 40x32, 50x32,100x32) [22].

Quantity of ISN	< 3	< 50	< 100	> 100
False channel detection	50%	5%	0.1%	$< 0.1\%$
False channel missing	50%url	8%	6%	5%
OS detection error [1]	50%	5%	1%	$< 1\%$

Figura 2.10: Quantidade de falso positivo após o treinamento de uma rede neural para a detecção do NUSHU fonte [26]

Na Figura 2.10 é possível ver os resultados obtidos por Tumoian em seus experimentos. A rede de melhor desempenho foi a 30x32. Com uma pequena quantidade de ISN's não é possível garantir a precisão, com o ISN menor que três não é possível nem mesmo identificar o sistema operacional utilizado. Com menos de cem números de ISN é possível descobrir com 99% de precisão o sistema operacional, e detectar um canal encoberto com 99,9% de precisão. De acordo com os experimentos, para o Windows XP SP2 a probabilidade de identificar o NUSHU é de 60%, já que o mecanismo de geração do Windows se aproxima da distribuição uniforme. O experimento demonstrou que o método proposto identifica canais encobertos com alta precisão e, além disso, constrói modelos de geração de ISN automaticamente.

2.7.2 Métodos de Eliminação

É possível eliminar a existência e a exploração de um canal encoberto mesmo que este não tenha sido previamente identificado [1]. Por exemplo, canais encobertos temporais podem ser rompidos ou ter o seu desempenho prejudicado apenas inserindo atrasos aleatórios na rede, estes atrasos irão aumentar a latência e prejudicar o desempenho da rede e das aplicações, no entanto pode ser uma alternativa a sistemas que lidam com informações extremamente confidenciais e sensíveis. Os canais do tipo armazenamento, podem ser interrompidos por um roteador ou intermediário que grave todas as sessões e integridade de todos os protocolos e troque o cabeçalho do pacote, normalizando os protocolos que estão sendo utilizados. Um equipamento

com esta capacidade, deve ter memória e agilidade para não degradar a rede e, portanto, é custoso. Outras medidas como reordenar os pacotes, alterando o algoritmo de entrada e saída FIFO (*First in First Out*) dos equipamentos de rede devem impedir canais encobertos baseado na alternância de protocolos.

Uma medida contundente é não permitir a utilização de protocolos desconhecidos ou não utilizados, se não há a necessidade de permitir por exemplo o protocolo IPv_6 para nenhum dos serviços, proibir este protocolo de trafegar na rede é suficiente para eliminar canais ocultos baseados neste protocolo.

Se um canal não pode ser eliminado, a prática recomendada pelo *"Orange book"* é que a sua capacidade seja reduzida. O valor a ser reduzido depende da quantidade de informação que está vazando e quão crítica ela é, por exemplo, se um *token* de acesso para o sistema de autenticação remoto expira a cada dois minutos, o canal encoberto deve ser reduzido para que o mesmo não consiga vazar o valor do *token* a tempo de poder ser utilizado por um atacante [5]. Canais encobertos que não são eliminados, devem ser auditados, para que possa ser detectado com confiança, diminuindo a probabilidade de falsos positivos. Canais encobertos com uma capacidade muito baixa para serem significativos devem ser documentados para que outros saibam e estejam cientes da sua possível existência [1].

Capítulo 3

Projeto

O protocolo HTTP é um dos protocolos mais utilizados na INTERNET atualmente [36]. Em adição ao formato tradicional, o HTTP provê a habilidade de estender a sua funcionalidade através de aplicações *web*, e por isso é um excelente candidato para a implementação de canais encobertos já que há muito tráfego normal para encobrir o tráfego encoberto. Atualmente uma enorme variedade de *malwares* utilizam o *Hyper Text Transfer Protocol* para avisar suas centrais de controle e comando sobre o comprometimento de uma máquina *(callback)*. Todavia, o padrão com que os *softwares* maliciosos se comunicam com seus criadores, facilita a criação de regras para estes *malwares* desde que seja identificado o modo de comunicação através de uma análise reversa do *malwar*e, prática comumente feita por empresas de antivírus para a criação de vacinas.

A literatura sobre a construção de canais encobertos assim como a esteganografia versa em procurar um campo cujo valor seja aparentemente aleatório, para esconder a distorção, e utilizar um algoritmo (LSB, MSB) para codificar a informação secreta. No entanto, o protocolo HTTP não possui muitos campos que aparentam ser pseudoaleatórios, há a possibilidade da utilização de *cookies* e outras técnicas, mas a maioria destes campos são utilizados para verificar a integridade de um arquivo e uma anormalidade invalidaria o arquivo e exporia o canal encoberto.

Para a construção de melhores canais encobertos, observar as características

como capacidade, discrição e robustez vistas no capítulo anterior são de suma importância. No HTTP, para aumentarmos a robustez, diminuindo a possibilidade de um canal ser destruído ou desconectado não é recomendável a utilização de *cookies*, uma vez que um usuário pode optar por bloqueá-los ou simplesmente excluí-los do sistema interrompendo o canal de comunicação.

Quanto a discrição, quanto menos modificar a estrutura padrão do HTTP, mais semelhante é o tráfego encoberto comparado com o real. Portanto, a inclusão de caracteres invisíveis, reordenação de campos e utilização de *"user-agents"* diferentes, como vistos nas técnicas anteriores, tornam o canal encoberto mais suscetível à detecção. Ao reduzir as possibilidades para esconder o tráfego encoberto, acabamos por diminuir a capacidade do canal.

Uma característica interessante do protocolo HTTP é que ele funciona como um conjunto de perguntas e respostas. Ao entrar em um *website*, um cliente (*browser*) estabelece uma conexão TCP e, através do HTTP, faz uma requisição de um recurso. O servidor interpreta esta requisição e responde com um código. Este código avisa o cliente se o servidor poderá atender a sua requisição com sucesso (200 - OK), se o recurso não foi encontrado (404 – NOT FOUND) ou se a sua localização foi alterada (3XX).

O protocolo foi especificado dessa forma porque, *a priori*, não há como o cliente saber que recursos o servidor pode prover e não sabe se no restante da conexão TCP virá a transferência de um recurso ou se o servidor está deliberadamente negando. Mas, considere que tanto o cliente quanto o servidor possuam uma lista de recursos que obrigatoriamente estão disponíveis no servidor. O que acontece se o servidor escolher negar um recurso dessa lista, enviando a resposta 404 - NOT FOUND?!. Poderíamos dar um significado a mais para tal fato já que o servidor não teria negar este recurso.

Como visto no dilema do prisioneiro (Capítulo 2) os envolvidos em uma comunicação secreta, precisam previamente compartilhar um segredo. Este segredo é a forma com o qual o canal encoberto é codificado e decodificado permitindo a

comunicação pelos participantes. No exemplo anterior, a codificação é a possibilidade do servidor negar ou não um recurso que deveria estar sempre disponível. Negar ou não pode ser interpretado de forma binária como zero (0) ou um (1) para estabelecer a comunicação secreta. No entanto, apenas essa codificação não é suficiente pois o cliente não sabe o motivo pelo qual o servidor está negando o recurso. Também é necessária, no momento em que os participantes troquem o segredo da codificação, compartilhar, sem custo, uma lista de recursos que deverão estar sempre disponíveis. Esse compartilhamento é dito sem custo, porque não envolve um passo extra e é inerente à codificação do canal. Para melhor entendermos é necessário uma rápida visão do protocolo HTTP.

3.1 Protocolo HTTP

O protocolo HTTP (*HyperText Transfer Protocol*) é um protocolo no nível da aplicação na camada OSI, para a distribuição de conteúdo através da Web. O HTTP tem sido usado pela *World Wide Web* desde 1990 a sua primeira versão foi definida na RFC 1945 e foi o primeiro protocolo na Internet a permitir a troca de mensagens e transferência de arquivos MIME e metadados [40]. A versão que será utilizada para esta dissertação é a versão mais atual 1.1 disponível na RFC 2616[40].

O protocolo HTTP é baseado em pares de requisições e respostas. As requisições, são originadas de um cliente (*user-agent*) para um servidor e as respostas seguem do servidor para um cliente. Este processo é exemplificado na Figura 3.1 através de uma captura realizada com o analisador de protocolo Wireshark [41].

```
GET /states/rj/riodejaneiro.js HTTP/1.1
Host: geoip.home.uol.com
Connection: keep-alive
Accept: */*
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.17 (KHTML, like Gecko)
Ubuntu Chromium/24.0.1312.56 Chrome/24.0.1312.56 Safari/537.17
Referer: http://www.uol.com.br/
Accept-Encoding: gzip,deflate,sdch
Accept-Language: pt-BR,pt;q=0.8,en-US;q=0.6,en;q=0.4
Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3
If-Modified-Since: Wed, 20 Feb 2013 01:00:05 GMT

HTTP/1.1 200 OK
Server: nginx
Date: Wed, 20 Feb 2013 02:20:39 GMT
Content-Type: application/javascript; charset=UTF-8
Last-Modified: Wed, 20 Feb 2013 02:00:06 GMT
Transfer-Encoding: chunked
Connection: close
Expires: Wed, 20 Feb 2013 03:20:39 GMT
Cache-Control: max-age=3600
Cache-Control: must-revalidate
Content-Encoding: gzip
```

Figura 3.1: Exemplo de uma requisição(Vermelho) e uma resposta(Azul) no protocolo HTTP

Na Figura 3.1 em vermelho podemos observar uma requisição pelo método GET do recurso /STATES/RJ/RIODEJANEIRO.JS realizada pelo USER-AGENT MOZZILA seguida da a resposta do servidor (azul) respondendo com o status 200 - OK informando que será possível atender a requisição do *browser*.

Requisição - Uma requisição é uma mensagem do cliente para o servidor que inclui na primeira linha o método para ser aplicado ao recurso, o identificador do recursos e a versão utilizada do protocolo.

A tabela 3.1 contém os principais métodos utilizados pelo HTTP para requisições.

Método	**Definição**
OPTIONS	Permite o cliente a determinar as opções e ou requerimentos.
GET	Pede um recurso ao servidor.
POST	Envia um recurso ao servidor.
PUT	Requisita que o recurso enviado seja salvo.
DELETE	Requisita que o recurso seja deletado.

Tabela 3.1: Principais métodos do protocolo HTTP e suas definições.

Após receber e interpretar uma requisição do cliente, o servidor deve responder com uma mensagem de resposta.

Resposta - O código de status é um inteiro de três dígitos, resultantes da tentativa de entender e satisfazer a requisição.

Os principais códigos de status podem ser vistos na tabela 3.2:

Código	Descrição
1xx	Informação.
2xx	Esta classe indica o sucesso de uma requisição.
200	Requisição pode ser atendida com sucesso.
3xx	Série 3XX é um redirecionamento de ação.
302	O recurso foi movido temporariamente
304	O arquivo não foi modificado.
4xx	Série 4XX é algum erro requisição.
404	Quando o servidor não encontra o recurso.

Tabela 3.2: Resumo dos principais códigos de resposta do protocolo HTTP

3.2 Prova de Conceito

Como dito na introdução, canais encobertos podem ser utilizados para a extração de informação e outras atividades que podem ser consideradas ou não ilícitas. No Brasil, a lei brasileira promulgada em dezembro de 2012, que tipifica os crimes cibernéticos discrimina a criação de códigos que possam ser utilizados de maneira errônea, o que acaba criminalizando a criação de provas de conceito.

> "[CAPUT] invadir dispositivo informático alheio, conectado ou não à rede de computadores, mediante violação indevida de mecanismo de segurança e com o fim de obter, adulterar ou destruir dados ou informações sem autorização expressa ou tácita do titular do dispositivo ou instalar vulnerabilidades para obter vantagem ilícita.
>
> § 1º Na mesma pena incorre quem produz, oferece, distribui, vende ou difunde dispositivo ou programa de computador com o intuito de permitir a prática da conduta definida no caput." (Lei Nº 12.737)

A lei deixa à cargo do magistrado o entendimento de conceitos subjetivos. O intuito, como descrito, se não explicitado previamente pelo pesquisador de segurança da informação ao desenvolver uma prova de conceito deixa margem a interpretação errônea da atividade de pesquisa. Para estar de acordo com a Lei Nº 12.737, e

ainda assim provar o conceito exposto nesta dissertação, o autor decidiu por encapsular o protocolo IRC com o método desenvolvido para o canal encoberto. O IRC (INTERNET Relay Chat) é um protocolo de bate-papo que permite que várias pessoas conectadas a um servidor, possam trocar mensagens de texto entre si, explicitando assim, que a prova de conceito desenvolvida tem por objetivo único e exclusivamente educacional, não podendo ser utilizado para fins ilícitos ou moralmente questionáveis.

Como visto na seção 3.1, o protocolo HTTP é baseado em requisições e respostas. Quando um recurso é pedido ao servidor o mesmo retorna com uma resposta e o recurso é transferido em uma outra conexão TCP. Porém, um cliente (*user-agent*) não tem a certeza de qual *status* a sua requisição será respondida, e portanto não sabe se o recurso existe e por algum motivo o servidor não deseja entregar.

Baseando-se neste princípio, se o cliente possui uma lista de objetos existentes, uma lista de objetos não existentes, é possível determinar se o servidor está "mentindo" para o cliente. Normalmente um servidor HTTP legítimo não mentiria, mas é possível utilizar este conceito para modular um bit em um par de requisições. Por exemplo, se o *user-agent* tem a certeza da existência do recurso e faz a seguinte requisição ao servidor: GET /images/logo.png e o mesmo responde com um HTTP/1.1 200 Ok, sabemos que o servidor não mentiu e portanto podemos interpretar que recebeu um bit FALSE (em relação a mentira) caso o servidor retornasse com HTTP/1.1 404 Not Found, o cliente sabe que este servidor está mentindo e portanto recebeu um bit TRUE.

A capacidade deste canal é baixa apenas $\frac{1}{2}$bit/pacote já que para o *user agent* obter um único bit são necessários dois pacotes um de requisição e outro de resposta. Para aumentar a capacidade é necessário alterarmos a forma como codificamos este canal. No exemplo anterior, utilizamos duas possíveis respostas do HTTP para estipular se o servidor está mentindo, para aumentar a capacidade do canal basta acrescentarmos mais um método. A requisição pode ser através do método GET, POST entre outros métodos do HTTP. Com isso receberíamos 1bit/pacote.

Adicionando mais possíveis respostas teríamos adição de bits por pacotes no canal aumentando a sua capacidade, mas reduzindo a sua discrição. Vale ressaltar que o autor não encontrou na literatura nenhum outro canal encoberto que se baseia no princípio que um Servidor pode estar mentindo.

3.2.1 Codificação (Cod#1)

A codificação é o segredo do canal encoberto, variando a forma de codificar é possível aumentar a capacidade, explorar uma codificação eficiente depende da aplicação que utilizará o canal.

O sistema foi dividido em duas partes seguindo o paradigma cliente-servidor. Tanto o cliente quanto o servidor foram desenvolvidos em Python, utilizando a biblioteca TwistedMatrix [42] para estabelecer uma conexão HTTP. A codificação do canal é baseada em um conhecimento *a priori*, e precisam compartilhar uma tabela de arquivos existentes. O HTTP permite que as requisições sejam feitas basicamente em dois métodos, GET e POST. Ao escolher um destes métodos podemos acrescentar mais uma informação na forma com o qual o canal se codifica. Com isso é possivel enviar dados do cliente para o servidor. O cliente sabe a resposta do servidor, escolher um método a ser utilizado para requisitar este recurso, e juntamente com a resposta do servidor obter os dados do servidor. Na tabela 3.3 temos um exemplo simplificado da tabela de codificação da resposta do servidor dado um método de requisição.

Método	Valor	Existe	Valor	Resposta
GET	0	0	00	200
GET	0	1	01	302
POST	1	0	10	301
POST	1	1	11	404

Tabela 3.3: Tabela codificação binária de requisições.

3.2.1.1 Servidor

O servidor proposto foi desenvolvido em Python utilizando o motor TwistedMatrix que é uma *engine* movida a eventos, escrita em Python de código aberto e sob a licença de uso do MIT[42] fornece classes e bibliotecas para o desenvolvimento de aplicativos em python. O *software* desenvolvido atua como um servidor HTTP semelhante a outros de sua categoria (Apache, nginX, IIS) modificado para responder as requisições de acordo com a codificação das mensagens a serem enviadas ao cliente. Por exemplo: ao ser requisitada uma página, o servidor analisa todos os objetos desta página, altera de acordo com a resposta a ser enviada ao cliente para que as respostas às requisições sejam compatíveis, decodifica e descomprime a mensagem no canal encoberto e a repassa para o cliente de IRC.

Para melhorar a capacidade do canal encoberto, sabendo que as informações trafegadas no canal serão frases de bate-papo, foi escolhido de acordo com o Apêndice E o algoritmo Smaz para a compressão do texto antes de ser enviado ao canal, diminuindo a quantidade de requisições.

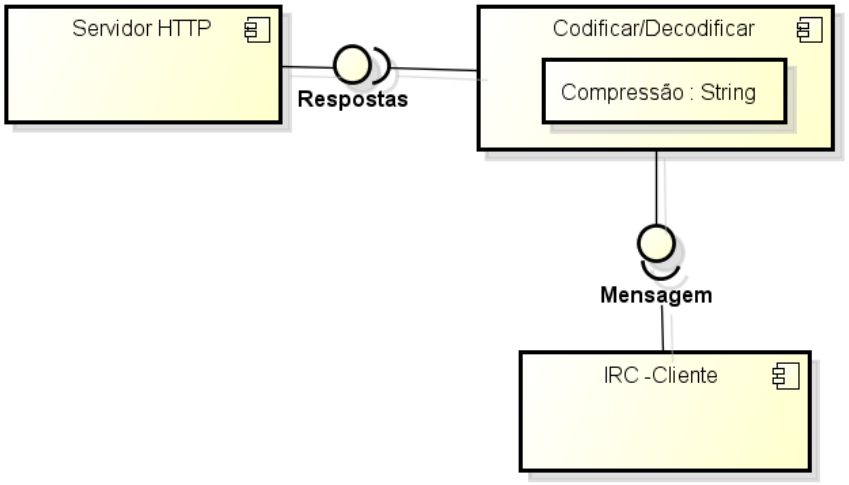


Figura 3.2: Diagrama de componentes do servidor proposto.

Na Figura 3.2 temos o diagrama de componentes do servidor proposto, o servidor HTTP escuta as requisições na porta 80, imitando o funcionamento dos servidores HTTP normais e fornece uma interface com o valor das respostas para o

componente capaz de decodificar e codificar as requisições e recuperar a mensagem, informando ao cliente IRC a ação a ser tomada.

3.2.1.2 Cliente

O cliente é um script em Python que simula um *browser* comum (Google Chrome, Firefox, INTERNET Explorer), requisitando páginas web e codificando e decodificando as mensagens a serem trocadas com o servidor. O diagrama de pacotes do cliente é ilustrado na Figura 3.3 mostra o pacote principal atuando como um cliente, ao enviar requisições a um servidor e o pacote "Codificar/Decodificar" que provê uma interface para o cliente utilizar o canal encoberto.

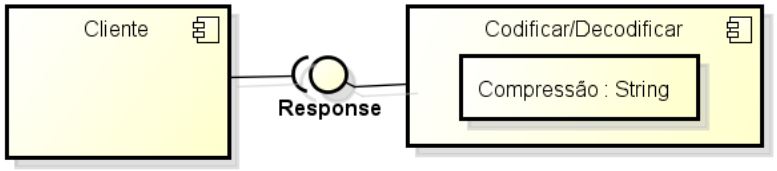


Figura 3.3: Diagrama de componentes do Cliente proposto.

A comunicação entre o cliente e o servidor pode é ilustrado no diagrama de sequencia na Figura 3.4. A interação entre o cliente e o servidor neste exemplo mostra a troca de mensagens necessária para a obtenção de um byte de informação.

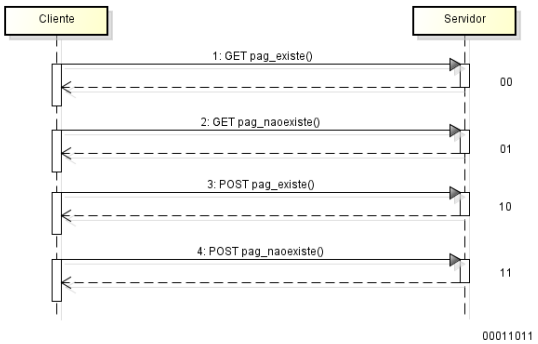


Figura 3.4: Diagrama de sequência das Requisições e Respostas entre o cliente e o servidor no esquema proposto.

3.3 Características do canal

Podemos inferir algumas características do canal encoberto proposto, são elas:

3.3.1 Capacidade.

Como visto no capítulo 2 a capacidade é definida pela quantidade de informação útil que o canal consegue transportar, e a codificação influencia na capacidade do canal. Sabendo que o canal encoberto proposto irá transferir strings curtas, frases em um bate papo e que o canal não terá ruído podemos calcular a capacidade do canal em 2 bits/pacote HTTP. A capacidade em bits/segundo vai depender exclusivamente da velocidade do meio e a taxa de transmissão do emissor.

3.3.2 Robustez.

Prevenir a incidência deste tipo de canal é extremamente difícil já que utiliza métodos autênticos do protocolo HTTP. Proibir ou alterar qualquer mecanismo de requisição e resposta do protocolo, através de um *proxy* modificado poderia prejudicar o real funcionamento de outras aplicações/web sites.

3.3.3 Discrição

Até o momento, pelo que se é conhecido na literatura, não há documentação de um canal encoberto que utilize este tipo de codificação. Um administrador ao analisar o tráfego HTTP poderia perceber apenas uma variação na quantidade de métodos utilizados (GET e POST) e de respostas do tipo 404, 302 e 304, e possivelmente não desconfiaria de um canal encoberto, julgando o *website* ser a fonte de problemas. Tentativas de detecção deste canal através da assinatura de strings geraria um enorme quantidade de falso positivo impossibilitando a sua detecção segura. No

entanto, tendo conhecimento do canal encoberto, foi possível observar que a distribuição das respostas não segue um padrão. O que é incomum em páginas web, que até mesmo se construídas dinamicamente não costumam apresentar uma variação tão grande nos *status* de resposta, podendo este ser utilizado para a possível detecção inicial do canal.

A discrição de um canal também está ligada com a frequência com que o canal é utilizado, é possível diminuir a frequência de utilização e distribuir as requisições em diversos *websites*. Como existe uma grande quantidade de tráfego normal dada a popularidade do protocolo HTTP, se a frequência de utilização deste canal for baixa é possível aumentar a sua discrição.

3.4 Possíveis contra medidas

Proxies HTTP se atuarem de maneira correta, seguindo a RFC do protocolo HTTP não podem atrapalhar a codificação do canal encoberto proposto nesta seção. No entanto, caso um *covert channel* fosse detectado, é possível alterar o funcionamento de um *proxy* para distorcer o valor das respostas do servidor HTTP, desconectando o canal.

Também é possível através de serviços de garantia de qualidade (QoS) diminuir a capacidade do canal, limitando a largura de banda e consequentemente a quantidade de requisições e respostas HTTP para a internet, reduzindo a capacidade do canal.

Capítulo 4

Metodologia e Experimentação

Para a modelagem do tráfego real, foram realizadas noventa e oito capturas de cinco minutos, a cada meia hora entre os dias 7 e 8 de julho de 2013 totalizando 6.7GB de dados, no servidor HTTP da RedeRio localizado no Centro Brasileiro de Pesquisas Físicas (CBPF). Para o procedimento de captura de forma automatizada foi criado um *script* de linha de comando disponível no Apêndice C.

Como o conteúdo da captura dos pacotes provenientes do servidor HTTP do CBPF é sensível, podendo conter senhas, *tokens* de sessão além de informações pessoais como quais sites estão sendo acessados durante o momento da captura, termos de busca e conversas, expondo informações possivelmente comprometendo a privacidade dos usuários, optou-se por criar um segundo *script* (Apêndice D) que obtém apenas os dados estatísticos do HTTP de interesse para a dissertação tais como:

- Método utilizado para a requisição
- Tamanho em bytes da Requisição
- Método de resposta

Este *script* foi auditado por outros dois alunos da Universidade Federal do Rio de Janeiro (UFRJ)(vide apêndice A) para garantir que a sua execução não colete nenhuma informação sigilosa ou comprometa a privacidade de uma ou mais pessoas.

A primeira vista, a codificação proposta no capítulo anterior é semelhante ao tráfego real, como parte dos esforços para tornar o canal mais discreto, procurou-se uma forma de diferenciar o tráfego real do tráfego encoberto com a codificação proposta, além de formas de impedir ou dificultar a utilização deste canal encoberto, tal qual um Guarda do dilema do prisioneiro faria:

- Um guarda passivo teria grande dificuldade para diferenciar um tráfego legítimo de um tráfego encoberto, sem o prévio conhecimento sobre o funcionamento do canal encoberto.
- Um guarda ativo e malicioso, alterando o tráfego, normalizando cabeçalho, não é capaz de desconectar o canal encoberto sem afetar outras aplicações legítimas.

Com os dados capturados no CBPF foi possível obter a distribuição empírica do tamanho das requisições HTTP e a distribuição dos métodos de requisição, com isso foi possível levantar mais informações para aprimorar o canal encoberto quanto a capacidade e discrição da seguinte maneira:

Distribuição de frequência dos métodos utilizados nas requisições HTTP - A primeira codificação utiliza os métodos GET e POST, no entanto, como pode ser visto na Figura 4.1 o método POST quase não foi tão utilizado nos dados do tráfego capturado, então, se a primeira codificação for utilizada para a transferência de muita informação em um curto espaço de tempo, há de se notar uma variação na utilização do método POST em comparação com o tráfego real o que diminui a discrição do canal.

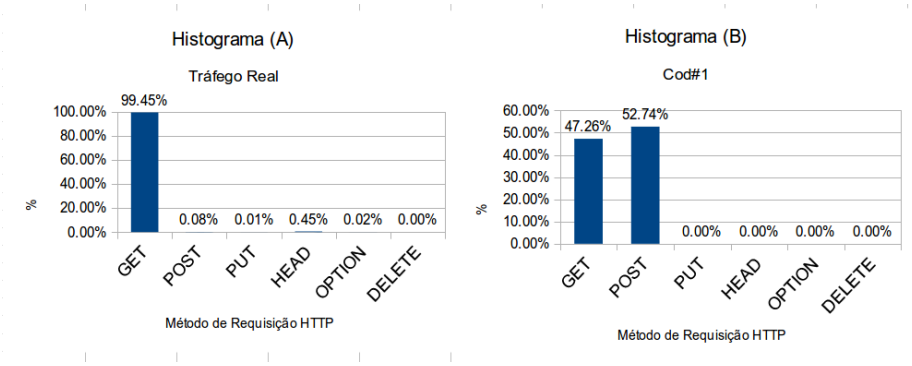


Figura 4.1: Comparação entre os métodos de requisição do protocolo HTTP (A) tráfego obtido no CBPF e (B) tráfego gerado a partir da primeira codificação.

Podemos ver na Figura 4.1(A) que a quantidade de requisições HTTP do tipo GET chega a 99,45% e POST 0,08% enquanto na Figura 4.1(B) o método POST é utilizado em mais de 50% das requisições, portanto, o comportamento do tráfego do canal encoberto proposto difere do tráfego real e esta diferença de comportamento poderia alertar um administrador.

Distribuição de frequência do tamanho das requisições HTTP - A primeira codificação (Cod#1) realizava requisições apenas com o nome do arquivo existente ou não, estes nomes variavam apenas entre 8 e 15 caracteres como pode ser visto na figura4.2(B) em contra partida, a variação do tamanho das URI no CBPF(A) é maior, chegando a 120 caracteres.

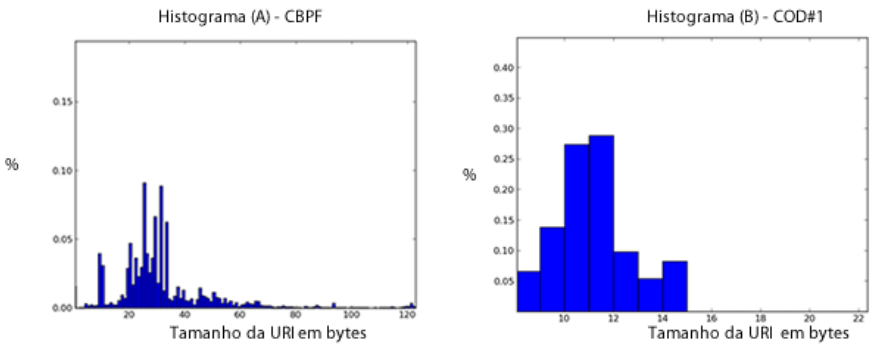


Figura 4.2: Comparação dos tamanhos das URI (A) CBPF e (B) gerado pelo COD#1

Embora o protocolo HTTP não especifique o tamanho máximo de uma requisição na RFC[40] este fator é deixado a cargo dos servidores retornarem um

status de erro caso a URI seja muito grande. A maioria dos *browsers* e servidores suporta mais de dois mil bytes, mas na prática não é comum visualizarmos requisições tão grandes. A primeira codificação utilizava URIs com pouca variação de tamanho e cada requisição transporta ocultamente 2bits. Se é interessante aumentar a variação do tamanho da URI visando imitar a distribuição de frequência do tráfego real é possível utilizar essa variação para aumentar não só a discrição como também a capacidade do mesmo. Fazendo o tamanho da requisição variar e seguir uma distribuição próxima a do real, podemos incorporar mais bits no canal encoberto.

Para a utilização deste canal encoberto em um ambiente real, é interessante alterar a codificação proposta (Cod#1), aumentar a sua capacidade em termos de bits por pacote e fazer com que a mesma tenha um comportamento estatisticamente mais próximo do tráfego HTTP da rede, aumentando a discrição.

4.1 Codificação (Cod#2)

Como visto nas seções anteriores, a primeira codificação proposta para o canal encoberto sobre o protocolo HTTP não é tão discreta podendo ser melhorada quanto a duas características: capacidade e discrição. Para alcançar tais objetivos esta seção apresenta a segunda codificação (COD#2) que visa melhorar a codificação do canal encoberto proposto.

Para tal, a segunda codificação não deve utilizar o método POST com a mesma frequência que a primeira codificação, para solucionar o problema a segunda codificação utilizará somente o método GET, o que implica em um bit a menos na requisição. Em contra partida, o tamanho da requisição pode ser aumentado para estar de acordo com a distribuição de tamanho de requisição vistos na Figura 4.2,ao variar o tamanho da requisição, bits podem ser adicionados neste contexto.

O algoritmo básico para a criação de uma nova requisição é:

Algoritmo:

```
Tam_max = sorteia o tamanho que a requisição deve ter a partir da coleta do trá
Enquanto tamanho da requisição < Tam_max {
verifica o bit a ser adicionado
se bit = 1:
requisição = + adiciona caminhos do tipo um na requisição
se bit = 0:
requisição = + adiciona caminhos do tipo zero na requisição.
}
retorna requisicao
```

A saída deste algoritmo retorna uma requisição a ser realizada com o método GET como pode ser visto no exemplo da Figura 4.3. Uma vez que o tamanho desta requisição segue a distribuição de probabilidade de tamanhos obtida através da rede real, a distribuição do tamanho das requisições irá se aproximar ao esperado a medida que mais requisições forem feitas ao longo do tempo.

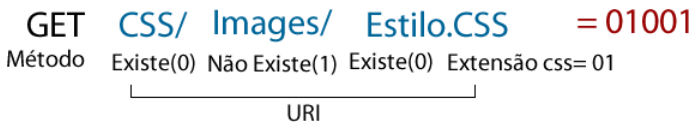


Figura 4.3: Exemplo de Requisição HTTP gerada a partir do algoritmo da segunda codificação.

O servidor para todos os casos deverá retornar 200 ou 304 (NOT MODIFIED). Caso o servidor queira enviar dados para o cliente, a última requisição deverá ser respondida com o código 302 (*Redirect*) seguido de uma URL contendo um código HTML pré-produzido com a mensagem a ser enviada. Neste caso, o cliente ao realizar as requisições irá salvar a sequência de arquivos que serão requisitados ao servidor permitindo a emissão de dados também pelo servidor.

4.2 Metodologia e Experimentação

Para avaliar a primeira (COD#1) e a segunda codificação proposta (COD#2) quanto a sua discrição, foram separados três conjuntos de conversa IRC com 10, 100 e 1000 frases escolhidas de maneira aleatória. Estas conversas são públicas e foram gravadas em canais de IRC abertos. A quantidade de frases, foi escolhido de forma arbitraria para imitar uma conversa curta, com pouca exposição do canal encoberto, uma conversa média e uma conversa maior, que ocorre por mais tempo expondo mais o canal. Estes conjuntos foram testados uma rede virtual criada com o software *VMware Workstation*[43] versão 9. Foram criadas três máquinas virtuais com as características de hardware e softwares descritos no Apêndice B. A estrutura da rede virtual pode ser vista na Figura 4.4.

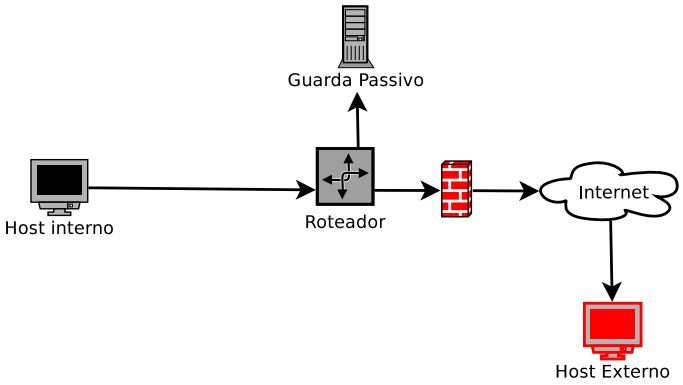


Figura 4.4: Diagrama de rede virtual utilizada para os testes.

No *host* "interno" foi instalado a versão cliente do canal encoberto, enquanto no *host* "externo" foi instalada a versão servidor. O objetivo é que uma vez estabelecida a comunicação entre estes dois *hosts*. O externo pode redirecionar as mensagens para um canal de IRC que antes não era acessível pelo "host interno". O "roteador" faz a interconexão entre a rede interna e a rede externa, capturando pacotes e analisando-os com o sistema de detecção de intrusão Snort.

Os seguintes critérios foram avaliados:

1. Capacidade.

2. Quantidade de alertas gerados nas regras no IDS.

4.2.1 Capacidade

Para avaliar a capacidade das duas codificações foram criados três arquivos de texto com 10, 100 e 1000 frases, respectivamente. Estas frases foram obtidas de maneira aleatória de canais de batepapo, o tamanho do arquivo foi medido em bytes(Q_b) e foi enviado para o servidor. O roteador contou a quantidade de pacotes transmitidos(Q_p) para então obter a capacidade média do canal encoberto em bits por pacotes de acordo com a definição da seção 2.4.3.

Os resultados são comparados a seguir:

10	**10**	**100**	**1000**
COD#1	2571(≈4.5bits/pkt)	12976(≈4.6bits/pkt)	127392(≈4.5bits/pkt)
COD#2	1119(≈11bits/pkt)	5254(≈12.2bits/pkt)	60985(≈14bits/pkt)

Tabela 4.1: Tabela comparativa de capacidade entre a COD#1 e COD#2.

Ambas as codificações, utilizam o algoritmo de compressão de strings e por isso puderam obter uma maior capacidade. A segunda codificação, por ter um tamanho de URL variável, foi utilizada na tabela 4.1 a média de dez transmissões com 10, 100 e 1000 frases.

4.2.2 Detecção por assinatura.

O Snort é uma ferramenta de detecção de Intrusão (IDS) baseado em assinatura, *opensource* e disponível no endereço www.snort.org, é possível baixar regras diretamente do site do Snort ou criar novas regras[44]. O Snort gera um alerta toda vez que um ou mais pacotes, aderem a assinatura.

4.2.2.1 Ferramentas

É possível encontrar na internet ferramentas criadas para evadir políticas de segurança e criar canais encobertos utilizando o protocolo HTTP. Nesta seção é feita

uma análise de três ferramentas encontradas na internet, que empregam técnicas de canais encobertos para encapsular outros protocolos

A primeira delas é o *Covert Channel Tunneling Tool* (CCTT) que foi desenvolvido pelos pesquisadores do *Gray-World Team* e disponibilizado ao público em junho de 2003. O CCTT encapsula protocolos como o TCP/IP, UDP e DNS dentro do HTTP. Esta ferramenta ainda possui a capacidade de inclusão de *addons* que aumentam as suas funcionalidades inclusive permitindo que o mesmo funcione como um *proxy* reverso[45]. O *Firepass* também foi desenvolvido pela equipe do *Gray-World* e foi projetado para encapsular protocolos TCP e UDP a partir de requisições HTTP do tipo POST, com o objetivo de burlar *firewalls* e sistemas de detecção de intrusão. Foi escrito em Perl e o servidor roda como um CGI[46]. O Wsh (*Webshell*) permite através de requisições HTTP do tipo POST o controle de um servidor remoto que esteja rodando o *script* CGI[47].

O funcionamento destes softwares são semelhantes, encapsulando outros protocolos em requisições HTTP. O Wsh foi escolhido para a comparação com o canal encoberto proposto por ser relativamente simples de alterar o seu funcionamento.

A segunda codificação (COD#2) assim como qualquer método recém criado não possui assinatura nos sistemas de detecção. O Snort permite que seus administradores criem regras e as submetam ao repositório da comunidade.

Dos canais encobertos, o Wsh não possui uma assinatura conhecida para o Snort na versão 2.9.2 utilizando as regras *default*. Para fins de comparação foi criada uma assinatura para a detecção do Wsh (A) e outra (B) para a tentativa de detecção do COD#2.

```
A)alert tcp any any -> any 80 (msg:"Wsh HTTP CovertChannel";\
    content:"X_KEY"; http_raw_header; \
  classtype:policy-violation;sid:5000983;)
```

```
B)alert tcp any any -> any 80 (msg:"COD#2 HTTP CovertChannel";\
  content:"/css/images"; http_raw_header; \
   classtype:weird-webdesign;sid:5000984;)
```

Ao analisar os pacotes provenientes do Wsh, foi identificado um campo "extra" no cabeçalho HTTP chamado X_KEY que transporta a chave de autorização, para que o servidor permita a conexão remota, como não é um campo padrão especificado no protocolo HTTP, foi utilizado para a criação da assinatura (A). Todavia, o código fonte desta ferramenta é aberto e um atacante mal intencionado poderia facilmente alterar o campo X_KEY invalidando a assinatura criada. Quanto ao COD#2 durante a execução dos testes, o Snort foi capaz de identificar menos de 20% dos pacotes, já que nem todos os pacotes formados pelo algoritmo terão a assinatura.

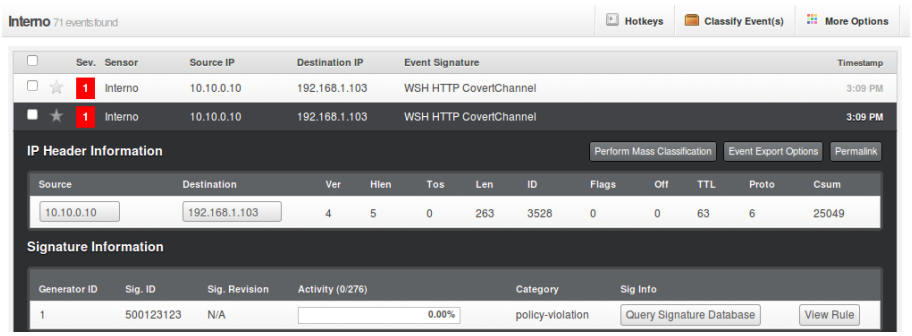


Figura 4.5: Alerta gerado pelo Snort detectando a assinatura do Wsh sendo visualizado no Snorby

Houve uma enorme dificuldade para a criar assinaturas eficientes para o COD#2 devido as requisições de URL serem semelhantes À URL's encontradas em sites na internet. Embora não seja uma prática comum colocar uma pasta "imagens" dentro de uma pasta "css" (folhas de estilo), nada impede que o *webdesigner* o faça criando

a possibilidade de obter uma grande quantidade de falso positivos impossibilitem a utilização da técnica de detecção por assinatura. Uma vez que os caminhos (*paths*) das requisições podem ser facilmente alterados para seguir um modelo próximo ao real, a criação de regras de detecção/prevenção aparentemente não é uma alternativa viável para a detecção do COD#2 já que ainda não foi possível encontrar uma forma de diferenciar de uma requisição normal justamente por utilizar as características originais do HTTP.

Para a identificação desta segunda codificação seria necessário a utilização de outras técnicas como outras características estatísticas ou análise de comportamento. Para isso, é necessário investigar melhor a modelagem do comportamento do HTTP na rede, buscando outras métricas que podem ser utilizadas para diferenciar o comportamento do canal encoberto proposto do tráfego real.

Uma possível vertente não explorada nesta dissertação é o comportamento ao longo do tempo. É sabido que o protocolo funciona em rajadas, onde são feitas muitas requisições em um curto período de tempo (visita de uma página de um usuário por exemplo) e um período de pouca ou nenhuma atividade leitura da página até que um outro *hyperlink* leve à um novo conjunto de requisições.

Capítulo 5

Conclusão

A utilização de canais encobertos traz uma maior privacidade para a segurança da comunicação, e o aumento da sua utilização aparenta ser a evolução natural para reforçar os sistemas de segurança, ainda mais no que condiz ao ambiente *web* com a utilização do protocolo HTTP.

Com o método proposto foi possível testar duas codificações para a criação de canais encobertos no protocolo HTTP, avaliá-las quanto suas características.

As principais dificuldades encontradas ao longo do desenvolvimento do presente trabalho foram relativos à falta na literatura, de métricas para caracterizar a discrição que é a característica do canal encoberto de não chamar atenção, e a robustez, ou seja, o quão difícil é evitar ou eliminar o canal. Embora alguns autores tenham realizado medidas quanto à estes critérios, foi particularmente difícil adaptá-los para o modelo proposto.

Nesta dissertação foram apresentadas duas codificações. A primeira é utilizada para ilustrar o conhecimento *a priori* no qual é fundamentado o modelo de canal encoberto proposto, e não deve ser utilizado em um ambiente real já que, como visto no capítulo 3, suas características diferenciam das características estatísticas observadas, possibilitando assim a sua detecção. A segunda codificação é mais coerente com o comportamento do protocolo HTTP e poderia ser utilizado em um ambiente real. Embora a capacidade seja relativamente baixa se comparado com outros canais

encobertos ativos e de armazenamento, pode ser usado para o transporte de quantidades de informação limitadas, no ponto de vista de um atacante por exemplo, pode ser utilizado para transportar algumas credenciais válidas para posterior acesso remoto (legítimo). Em casos de ataques como APT (*Advanced persistent threats*) que na maioria dos casos os atacantes controlam a rede alvo durante meses, a taxa de transferência baixa é compensada pelo tempo e discrição que o canal proporciona. Outra vantagem desta técnica, na extração de dados é que a maioria dos servidores Web salvam registros de suas atividades. Uma vez que um atacante não tem acesso direto a um site permitido pela rede alvo, o mesmo pode enviar suas requisições para um site permitido e posteriormente recuperar os registros daquele período.

Cabe salientar que uma vez expostos os mecanismos para a criação e codificação de um canal encoberto, o mesmo fica passível de sofrer sanções que eliminem ou limite a sua capacidade. A codificação com troca de conhecimento *à priori* baseado em protocolos do tipo "pergunta e resposta", embora tendo tido o mecanismo explicitado e documentado no capítulo 3 (Cod#1) e melhorado no capítulo 4(Cod#2) ainda torna dificil a diferenciação de um canal encoberto e um tráfego HTTP usual.

5.1 Considerações finais

A exemplo de uma utilização errônea, a baixa capacidade de bits em contra partida com a alta discrição poderia ser utilizado para o transporte de logins e senhas, que poderiam dar acesso posterior a sistemas de rede privada(VPN) ou outro tipo de conexão remota válida.

A exemplo de uma aplicação benigna do canal encoberto proposto, seria utilizar o canal encoberto para transportar informações que validem o servidor de origem e por exemplo impedir ataques de *"man-in-the-middle"* e *phishing* que assolam o sistema bancário brasileiro. A ideia de utilizar canais encobertos no HTTP para prevenir ataques do homem do meio foi exposta por Brown *et. al.*[10] para canais

encobertos no HTTP, que até onde se tem conhecimento não foram implementados.

5.2 Trabalhos futuros

Futuramente, analisar que outras características podem limitar a capacidade do canal encoberto, tal qual investigar novas codificações baseadas no conhecimento *à priori* que permitam um melhor *tradeoff* entre as características inerentes à como a capacidade, robustez e discrição. Investigar possíveis utilizações de canais encobertos que possam melhorar a segurança de uma rede de computadores incluindo a possibilidade de identificação de ataques do tipo *"man-in-the-middle"* em ambientes WEB e LAN a fim de aumentar as possíveis utilizações deste tipo de canal encoberto melhorando aspectos de segurança como a privacidade, integridade e confidencialidade.

Referências Bibliográficas

[1] ZANDER, S.; BRANCH, P.; ARMITAGE, G. A survey of Covert Channels and Countermeasures in Computer Network Protocols. *ieee communications surveys*, v. 9, n. 3, p. 44–57, 2007. xi, 7, 14, 16, 21, 26, 30, 31

[2] LAMPSON, B. W. A note on the confinement problem. *Commun. ACM*, ACM, New York, NY, USA, v. 16, n. 10, p. 613–615, out. 1973. ISSN 0001-0782. Disponível em: <http://doi.acm.org/10.1145/362375.362389>. 2, 6

[3] WEF. *Global Risks 2012 Seventh Edition An Initiative of the Risk Response Network*. [S.l.], 2012. Disponível em: <http://reports.weforum.org/global-risks-2012/>. 2, 10

[4] GOOGLE. *Google Scholar*. february 2013. Disponível em: <http://scholar.google.com>. 3

[5] DEFENSE, U. S. D. of. *Trusted Computer System Evaluation Criteria (Orange Book)*. [S.l.], 1985. 6, 31

[6] CHAVES, C.; MONTES, A. Detecção de Backdoors e Canais Dissimulados. *V Workshop dos cursos de Computação ...*, p. 311 –323, august 2005. 6

[7] GEUS, P.; NAKAMURA, E. *Segurança de Redes em Ambientes Cooperativos*. 1. ed. São Paulo, Brasil: Berkeley, 2002. 6

[8] MICHAELIS. *Dicionário Michaelis*. october 2012. Disponível em: <http://michaelis.uol.com.br>. 7

[9] ZANDER, S. *Performance of Selected Noisy Covert Channels and Their Countermeasures in IP Networks*. Tese (Doutorado) — Centre for Advanced Internet Architectures Faculty of Information and Communication Technologies Swinburne University of Technology, Melbourne, May 2010. Disponível em: <http://caia.swin.edu.au/cv/szander/thesis/thesis.html>. 7, 14, 15, 27

[10] BROWN BO YUAN, D. J. E.; LUTZ, P. Covert channels in the http network protocol: Channel characterization and detecting man-in-the-middle attacks. In: . [S.l.: s.n.], 2010. p. 726 –730. 9, 15, 54

[11] STONE-GROSS, B. et al. Analysis of a botnet takeover. *Security Privacy, IEEE*, v. 9, n. 1, p. 64 –72, jan.-feb. 2011. ISSN 1540-7993. 10

[12] MCCLURE, S.; SCAMBRAY, J.; KURTZ, G. *Hacking Exposed: Network Security Secrets and Solutions, Fourth Edition*. 4. ed. New York, NY, USA: McGraw-Hill, Inc., 2003. ISBN 0072227427. 10

[13] MANDIANT. *APT1 Exposing One of Chinas Cyber Espionage Units*. [S.l.], 2012. 12

[14] MARTINS, D. V. *BGP TRACEBACK: Um novo método para Identificação de caminhos de ataques na Internet*. Dissertação (Mestrado), 2005. 12

[15] QU, H.; CHENG, Q.; YAPRAK, E. Using covert channel to resist dos attacks in wlan. In: YANG, L. T.; ARABNIA, H. R.; WANG, L.-C. (Ed.). *ICWN*. CSREA Press, 2005. p. 38–44. ISBN 1-932415-55-6. Disponível em: <http://dblp.uni-trier.de/db/conf/icwn/icwn2005.html>. 12, 13

[16] RAY, B.; MISHRA, S. A protocol for building secure and reliable covert channel. *Privacy, Security and Trust, 2008. PST'08. ...*, p. 246–253, 2008. Disponível em: <http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4641292>. 14

[17] OKJRAVI, H.; BAK, S.; T.KING, S. Design, implementation and evaluation of covert channel attacks. In: *Proceedings of IEEE Conference on Technologies for Homeland Security*. [S.l.: s.n.], 2010. 15, 16

[18] NEWMAN, R. C. Covert computer and network communications. In: *Proceedings of the 4th annual conference on Information security curriculum development*. New York, NY, USA: ACM, 2007. (InfoSecCD '07), p. 12:1–12:8. ISBN 978-1-59593-909-8. Disponível em: <http://doi.acm.org/10.1145/1409908.1409922>. 16

[19] EGGERS, K.; MALLETT, P. Characterizing network covert storage channels. In: *Aerospace Computer Security Applications Conference, 1988., Fourth*. [S.l.: s.n.], 1988. p. 275 –279. 16

[20] ZI, X. et al. Evaluating the transmission rate of covert timing channels in a network. *Computer Networks*, Elsevier B.V., v. 55, n. 12, p. 2760–2771, ago. 2011. ISSN 13891286. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S138912861100185X>. 16

[21] NAIR, A. et al. Length based network steganography using udp protocol. In: *Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on*. [S.l.: s.n.], 2011. p. 726 –730. 16, 17

[22] MURDOCH, S. J.; LEWIS, S. Embedding covert channels into tcp/ip. In: *Information Hiding: 7th International Workshop, volume 3727 of LNCS*. [S.l.]: Springer, 2005. p. 247–261. 17, 18, 29

[23] LUCENA, N.; LEWANDOWSKI, G.; CHAPIN, S. Covert channels in ipv6. In: DANEZIS, G.; MARTIN, D. (Ed.). *Privacy Enhancing Technologies*. Springer Berlin / Heidelberg, 2006, (Lecture Notes in Computer Science, v. 3856). p. 147–166. Disponível em: <http://dx.doi.org/10.1007/1176783110>. 17

[24] RUTKOWSKA, J. The implementation of passive covert channels in the linux kernel. In: *Chaos Communication Congress*. [S.l.]: Chaos Communication Congress, 2004. 18

[25] HOLLAND, W. *Chaos Computer Club*. 2004. Hacker group. Disponível em: <http://www.ccc.de/>. 18

[26] TUMOIAN, E.; ANIKEEV, M. Network Based Detection of Passive Covert Channels in TCP/IP. In: *Proceedings of the The IEEE Conference on Local Computer Networks 30th Anniversary*. Washington, DC, USA: IEEE Computer Society, 2005. (LCN '05), p. 802–809. ISBN 0-7695-2421-4. Disponível em: <http://dx.doi.org/10.1109/LCN.2005.92>. 18, 23, 29

[27] LIU, Y. et al. Hide and seek in time - robust covert timing channels. In: BACKES, M.; NING, P. (Ed.). *Computer Security - ESORICS 2009*. [S.l.]: Springer Berlin / Heidelberg, 2009, (Lecture Notes in Computer Science, v. 5789). p. 120–135. ISBN 978-3-642-04443-4. 18, 28

[28] SELLKE, S. H. et al. TCP/IP Timing Channels: Theory to Implementation. In: *INFOCOM 2009, IEEE*. [S.l.: s.n.], 2009. p. 2204–2212. ISSN 0743-166X. 18, 22

[29] LIU, G. et al. Covert timing channel with distribution matching. In: *Proceedings of the 2009 International Conference on Multimedia Information Networking and Security - Volume 01*. Washington, DC, USA: IEEE Computer Society, 2009. (MINES '09), p. 565–568. ISBN 978-0-7695-3843-3. Disponível em: <http://dx.doi.org/10.1109/MINES.2009.280>. 19

[30] ZI, X. et al. Implementing a passive network covert timing channel. *Computers & Security*, Elsevier Ltd, v. 29, n. 6, p. 686–696, set. 2010. ISSN 01674048. Disponível em: <http://linkinghub.elsevier.com/retrieve/pii/S0167404809001485>. 19

[31] WENDZEL, S.; KELLER, J. Low-attention forwarding for mobile network covert channels. *Communications and Multimedia Security*, p. 122–133, 2011. Disponível em: <http://link.springer.com/chapter/10.1007/978-3-642-24712-5_10>. 19, 21

[32] WENDZEL, S. Protocol Channels. *Hackin9*, v. 6, p. 38–40, 2009. 20

[33] JADHAV, M.; KATTIMANI, S. Effective detection mechanism for tcp based hybrid covert channels in secure communication. In: *Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on*. [S.l.: s.n.], 2011. p. 1123 –1128. 21

[34] FEAMSTER, N. et al. Infranet: Circumventing web censorship and surveillance. In: *11th USENIX Security Symposium*. San Francisco, CA: [s.n.], 2002. 23, 24

[35] BAUER, M. New covert channels in HTTP. *Proceeding of the ACM workshop on Privacy in the electronic society - WPES '03*, ACM Press, New York, New York, USA, p. 72, 2003. Disponível em: <http://portal.acm.org/citation.cfm?doid=1005140.1005152>. 24, 26

[36] HUBA, W. et al. A HTTP cookie covert channel. In: *Proceedings of the 4th international conference on Security of information and networks*. New York, NY, USA: ACM, 2011. (SIN '11), p. 133–136. ISBN 978-1-4503-1020-8. Disponível em: <http://doi.acm.org/10.1145/2070425.2070447>. 24, 32

[37] ESTRADA, N.; FEAMSTER, N.; FREEDMAN, M. An Application Layer Covert Channel: Information Hiding With Chaffing. p. 1–15, 1999. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.58.4318&rep=rep1&type=pdf>. 25

[38] SELVI, J. Covert Channels over Social Networks. *Sans Institute*, mar. 2012. 26

[39] ZANDER, S.; BRANCH, P.; ARMITAGE, G. Capacity of Temperature-Based Covert Channels. *Communications Letters, IEEE*, v. 15, n. 1, p. 82–84, jan. 2011. ISSN 1089-7798. 27

[40] FIELDING, R. et al. *Hypertext Transfer Protocol – HTTP/1.1 (RFC 2616)*. 1999. Request For Comments. Disponível em: <http://www.ietf.org/rfc/rfc2616.txt>. 34, 45

[41] WIRESHARK. *Wireshark*. 2013. Disponível em: <http://www.wireshark.org>. 34

[42] TWISTEDLANS. *Twisted*. Disponível em: <http://twistedmatrix.com/>. 38, 39

[43] VMWARE, I. *Download VMware Workstation*. 2013. Disponível em: <https://my.vmware.com/web/vmware/info/slug/desktop_end_user_computing/vmware_workstat 48

[44] SOURCEFIRE. *Snort*. 2012. Disponível em: <http://www.snort.org/>. 49

[45] CASTRO, S. *CCTT*. 2006. Disponível em: <http://www.gray-world.net/pr_cctt.shtml>. 50

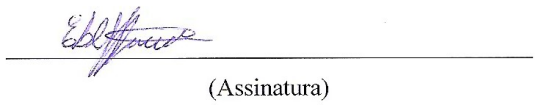
[46] CHAVES, C.; MONTES, A. Detecção de Backdoors e Canais Dissimulados. *V Workshop dos cursos de Computação ...*, 2005. Disponível em: <http://bibdigital.sid.inpe.br/rep-/sid.inpe.br/MTC-m13@80/2006/02.14.16.47>. 50

[47] DYATLOV, A.; CASTRO, S. *WSH*. 2003. Disponível em: <http://gray-world.net/pr_wsh.shtml>. 50

[48] SANFILIPPO, S. *Smaz*. Disponível em: <https://github.com/antirez/smaz>. 68

Apêndice A

Eu Evandro Luiz Cardoso Macedo, aluno de Mestrado do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ. Atesto que o código desenvolvido por **Felipe Afonso Espósito**, para a dissertação "UM CANAL ENCOBERTO PARA TRANSMISSÃO DE DADOS SOBRE O PROTOCOLO HTTP" presente no **Apêndice D,** intitulado "*grab-statistics.py*", não obtém informações pessoais tais quais: nomes de usuário, senhas ou comprometa de alguma forma a privacidade de outrem.

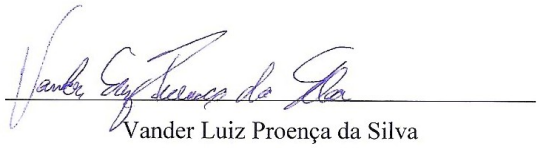


(Assinatura)

Rio de Janeiro

Agosto de 2013

Eu, Vander Luiz Proença da Silva aluno de Mestrado do Programa de Engenharia de Sistemas e Computação, COPPE/UFRJ. Atesto que o código desenvolvido por **Felipe Afonso Espósito**, para a dissertação "UM CANAL ENCOBERTO PARA TRANSMISSÃO DE DADOS SOBRE O PROTOCOLO HTTP" presente no **Apêndice D,** intitulado "*grab-statistics.py*", não obtém informações pessoais tais quais: nomes de usuário, senhas ou comprometa de alguma forma a privacidade de outrem.



Vander Luiz Proença da Silva

Rio de Janeiro

Agosto de 2013

Apêndice B

Todas as máquinas virtuais dispõem das seguintes configurações de hardware e sistema operacional:

- Um processador 64 bits
- 512MB de memória RAM
- Disco rígido de 20GB
- Uma placa de rede conectada a rede de teste(exceto a máquina virtual do atacante).
- Sistema operacional Ubuntu 12.04 64bits.

Além das ferramentas de canais encobertos, foram instalados os seguintes softwares:

Roteador:

- Wireshark
- Snort IDS
- IPtables

Apêndice C

Script criado para realizar a captura de tráfego TCP em intervalos de quinze minutos.

——- capture.sh ——————

```
1  #Para incluir no Crontab
2  #00,15,30,45 * * * * /path/to/capture.sh
3
4  #!/bin/bash
5
6
7  #finaliza todos os processos de captura
8  killall tcpdump
9
10 #obtem a data e hora local para salvar o nome do arquivo formato 01122013-1220.pcap
11 var=$(date +%d%m%G-%k%M)
12
13 #executa o TCPDUMP com filtros na porta 80 TCP na interface de saÃda para a Internet salvando o arquivo como PCAP
14 /usr/sbin/tcpdump 'tcp port 80'-i eth0 -C 10 -w /home/user/$var.pcap
```

Apêndice D

Programa desenvolvido para retirar métricas HTTP de um arquivo do tipo PCAP (*Packet Capture*).

```
#!/usr/bin/env python
import os
try:
    import scapy.all as scapy
except ImportError:
    import scapy
try:
    scapy.load_contrib('HTTP')
except ImportError:
    print "FAILED"

dir = "/var/log/felipe/"
dir_save = "/var/log/tmp-ravel/"

total_arq = 0 #conta o numero de arquivos do tipo pcap em um diretorio
files = os.listdir(dir)
for file in files:
        if file.endswith(".pcap"):
                total_arq+=1

print "num# de arquivos na pasta: "+ str(total_arq)
#cria uma pasta temporaria para armazenar os arquivos
if not os.path.exists(dir_save ):
    os.makedirs(dir_save )
#abre os arquivos com permissao de escrita
try:
        distPktSize =  open(dir_save+'dist_PktSize.txt', 'w')
        distRequestMethod =  open(dir_save+'dist_requestMethod.txt', 'w')
        distRequestMethodSize =  open(dir_save+'dist_requestMethodSize.txt', 'w')
        distRequestPaths =  open(dir_save+'dist_requestPaths.txt', 'w')
except:
        print "COULD NOT OPEN ONE OR MORE FILES"


n=1
for file in files:
        print str(n)+")"+file
        pacotes = scapy.rdpcap(dir+file) #usando o scapy para ler o pacote pcap
        distPktSize.write(str(len(pacotes))+'\n')
        httpRqst = pacotes.filter(lambda(s): HTTPrequest in s) #filtrando por HTTP Request

        ReqMethod =[i.Method[:4] for i in httpRqst]
        for w in ReqMethod:
                print w
                distRequestMethod.write(w) #Method
        ReqMethodS =[str(len(i.Method[4:len(i.Method)-11]))+'\n' for i in httpRqst]
        for u in ReqMethodS:
                print u
                distRequestMethodSize.write(u) #Method
        ReqMethodP =[str(i.Method[4:len(i.Method)-11].count('/'))+'\n' for i in httpRqst]
        for v in ReqMethodP:
                print v
                distRequestPaths.write(v) #Method

        n=n+1

#fechando os arquivos
distPktSize.close()
distRequestMethod.close()
distRequestMethodSize.close()
distRequestPaths.close()

#abre os arquivos com permissao de escrita
try:
        distResponse =  open(dir_save+'dist_response.txt', 'w')
```

```
66  except:
67          print "COULD NOT OPEN ONE OR MORE FILES"
68  n=1
69  for file in files:
70          print str(n)+')'+file
71          pacotes = scapy.rdpcap(dir+file) #usando o scapy para ler o pacote pcap
72          httpRsp = pacotes.filter(lambda(s): HTTPresponse in s)
73          rsp = [str(j.StatusLine[9:12])+'\n' for j in httpRsp]
74          for u in rsp:
75                  distResponse.write(u) #Method
76          n=n+1
77
78  distResponse.close()
79
```

Apêndice E

Para a escolha do algoritmo de compressão a ser utilizado, foi feita uma pequena comparação. Para tal foram separadas dez frases aleatoriamente, e realizado uma comparação com as ferramentas de compressão do sistema operacional linux: gzip, zip e bzip2.

Tabela comparativa:

Amostra	**(ASCII 7bits)**	**ZIP**	**GZIP**	**TAR.GZ**	**SMAZ**
f0.txt	60	217	80	10240	32
f1.txt	33	195	30	10240	28
f2.txt	119	263	126	10240	76
f3.txt	132	260	123	10240	32
f4.txt	26	188	53	10240	18
f5.txt	84	302	165	10240	38
f6.txt	189	302	165	10240	10
f7.txt	20	182	47	10240	14
f8.txt	70	229	92	10240	29
f9.txt	55	216	79	10240	30
Tamanho médio(bits)	79	229	92	10240	20

Figura 5.1: Tabela com o tamanho final em bits das frases comprimidas por diferentes algoritmos.

O Smaz é um algoritmo leve de compressão para pequenos textos, registrado sob licença de uso BSD [48] Smaz vez não é bom para a compressão de dados de uso geral, mas pode comprimir texto de 40-50% no caso da média (funciona melhor com o Inglês) e é capaz de executar um pouco de compressão para HTML e urls bem. O ponto importante é Smaz que é capaz de comprimir até sequências de dois ou três bytes!

Os algoritmos de compressão gzip, zip e bzip2 aumentam o tamanho do arquivo porquê junto aos dados, salva também informações para a descompressão e ou o tamanho do bloco a ser comprimido é maior do que o conteúdo em si.

Portanto o algoritmo de compressão escolhido foi o Smaz, que possui melhor índice de compressão de strings pequenas.