HOCHSCHULE FÜR UNIVERSITY OF
TECHNIK STUTTGART APPLIED SCIENCES

Bachelor Thesis

DESIGNING AN INTERCONNECTION CONCEPT OF AN ELASTIC COMPUTE CLOUD (EC2) ENVIRONMENT WITH LOCAL NETWORKS VIA VPN

by

Bernd Hietler

Course of Studies: Informatik (Computer Science)
Matriculation Number: 370921
January 2009

Reviewer at the University: Dorothee Koch
Prof. Dipl.-Math., MSc in Computer Science
Stuttgart University of Applied Sciences

Reviewer at the Company: Aleksey Aristov
Company: Weiglewilczek GmbH

ABSTRACT

DESIGNING AN INTERCONNECTION CONCEPT OF AN ELASTIC COMPUTE CLOUD (EC2) ENVIRONMENT WITH LOCAL NETWORKS VIA VPN

Bernd Hietler

Abstract **Elastic Compute Cloud (EC2) is an Amazon webservice which makes it possible to set up and run virtual machine servers in Amazon's cloud environment in a flexible manner within a few minutes. Today these machines are mostly used as part of a short-term solution for different kinds of Internet services. This text describes a design of an interconnection concept for the use of EC2 instances in the local network. Here these virtual machines can be used as replacements for high priced local servers which are expensive to purchase and to maintain. The most suitable solution in this case seems to be the concept of Virtual Private Networks (VPNs).**

ACKNOWLEDGMENTS

At first I would like to thank my examiner Dorothee Koch from HfT Stuttgart for the feedback and mentoring of this thesis and Aleksey Aristov for the topic idea, his support and advises.

I'd also like to thank Dr. Jörn Weigle and Dr. Stephan Wilczek from the Weiglewilczek GmbH for providing the Amazon Web Services account the office infrastructure, bearing the expenses for books and other support.

Furthermore I want to express my thanks to my friends and fellow students Frieder Bürzele and Levin Fritz for proofreading.

Without the financial support of my parents it would have been very difficult to finish my studies in this time.

Contents

Chapter 1

Introduction

1.1 Problem Definition

The so-called era of the Web 2.0 brought not only a change of the Internet but also of the way how we use our desktop computers. Today a big amount of applications and tasks are no longer executed in our desktop machines but in the Internet. Webservices are becoming more and more popular and important. This tendency is not only relevant for our desktop machines. Many smaller companies are moving servers or specific services to external large-scale datacenters as well. In many cases this is more efficient for the company, because it takes less effort to take care of power blackouts, network connectivity problems, hardware-related problems or to run backups and software updates. Nicholas Carr compares this development to the era of industrialization in the 19th century, where factories first ran their own small power generators and then switched more and more to the electric power grid [1]. Here he creates an analogy of the electric power grid and the broadband Internet, where the role of the electric power plants is compared to the role of large data centers running a large number of servers in an efficient way. Information Technology, he claims, is becoming more and more a service that can be obtained out of the network socket in the wall. This concept is known as utility computing. Cloud computing is a new technology, often required by utility computing services, that provides an environment where virtual servers can be launched and operated in a very flexible manner. This environment is designed to run servers for Internet applications like webservers during performance peaks. In cases where companies need to run servers in their local networks, for instance to add an temporary intranet application or only because they need more computing power for a short period, they still have to go back to the adding of local hardware to their networks. This is expensive and mostly means a lot of work. It would be an important benefit to build up a secure interconnection of cloud computing instances with local networks.

1.2 Cloud Computing

The term cloud computing is often associated with the technology grid computing and the business model utility computing. Grid computing is a kind of distributed computing on computer clusters. The typical grid computing environment is not limited to a single data center. It consists of several data centers which are located in different places, sometimes even on different continents, connected via high speed data cables. Grid computing was primarily designed for special compute-intensive tasks e.g in science and research. Due to falling prices for network and server hardware, today, large computer grids are operated for economic purposes as well [2]. Cloud computing, usually relying on a grid computing infrastructure, is a good example for this development. Franco Travostino, a system architect at Ebay, describes cloud computing as more related to the Web 2.0 mindset, where there has to be simplicity in contrast to grid computing, which is out of supercomputing, where people are used to have a look at very complex things [3]. This simplicity is represented by easy to handle tools and programming APIs to manage these virtual servers. Cloud computing can be understood as the possibility to set up, start, administer and shut down virtual servers via mouse clicks or simple commands. If cloud computing is offered as a service to customers it fits with the term utility computing. Today there are different technology types and vendors of cloud computing services e.g. Sun, Oracle and Amazon. This thesis describes an interconnection with Amazon's cloud computing service Elastic Compute Cloud.

1.3 Virtual Private Networks (VPN)

There is no consistent terminology for the Virtual Private Network concept, but the book "Building GNU/Linux virtual private networks" [4] gives a reasonable definition. Usually a VPN is considered as a local network, where other networks or single machines are connected via encrypted tunnels using an already existing network connection. A tunnel is a network connection where one network protocol is transported within another protocol. In this case it is used to encapsulate and encrypt network packets that are sent through an insecure environment (usually the Internet). Thereby they are unreadable for others than the receptor. Hence different participants are forming a new "virtual" network with a "private" nature. While the term "network" should be clear, the individual interpretation of "virtual" and "private" may vary depending on their viewing perspectives. "Virtual" networks for instance can be considered as different parts of an entire network which are using different network protocols. Also a combination of different connected networks or single clients can be called a virtual network. The

term “privacy” can be understood in a physical manner, e.g. a connection between private networks which might not necessarily be encrypted. It can also be regarded as an encrypted connection of participants which aren't necessarily located in private networks [4] [5]. Private networks consist of IP addresses which cannot be allocated in the Internet and are reserved for the use in internal networks of companies or at home like e.g. the network 192.168.0.0 [6]. The Virtual Private Network concept in this text will be an encrypted connection between a group of server instances, which are running in EC2's private network, and a local private network, where they can interact with local server machines. During the evaluation of different VPN technologies it turned out that a plain site-to-site VPN, in which the cloud instances don't need to be modified, is not possible. This is caused by specific cloud computing network issues, mainly the lack of an accessible standard gateway for the EC2 instances. Therefore each instance has to act like a VPN client, connecting to VPN server in the cloud.

Chapter 2

The Technology's State of the Art

For the design of a specific network solution it is necessary to start with the description and analysis of the different technologies which are used or could be used for the design of this VPN concept. The first part of this chapter covers Amazon's EC2 service. The second part deals with the VPN aspects and their reconcilability with EC2.

2.1 Elastic Compute Cloud (EC2)

2.1.1 The Elastic Compute Cloud Service

The Elastic Compute Cloud (EC2) is a cloud computing environment, run by Amazon, which is based on the Xen virtualization technology. It is part of Amazon's Webservices (AWS), a collection of remote computing services offered over the Internet. EC2 makes it possible to run and manage predefined or self built virtual server instances in this environment and access them via the Internet. They are based on predefined images called AMI, which means Amazon Machine Image, and can be handled via command line tools or via programming APIs. EC2 runs as a paid webservice from Amazon which requires a valid Amazon S3 netstorage-account as well. The EC2 network consists of several availability zones, which are hosted in different datacenters. Currently there are three availability zones `us-east-1a`, `us-east-1b`, `us-east-1c` in the USA and since December 2008 `Eu-west-1` in the EU as well. For security reasons the exact location is kept secret and a strong security concept is underlying these data centers [7]. The EC2 service is still labeled Beta, which means the service's features are still evolving quickly and there is still a small risk of problems [8].

2.1.2 Amazon Machine Images

There are ready defined AMI images provided by Amazon, which can be taken as template to modify, configure and register as own customized images. It is possible as well to build own images from scratch via loopback file or purchase AMI images for special tasks. There are predefined AMIs for different GNU/Linux distributions and for OpenSolaris[1]. During the writing of this thesis Amazon added predefined Windows AMIs as well. This thesis however only describes the work with GNU/Linux based AMIs. The following schema shows the steps that are required to run an EC2 instance:

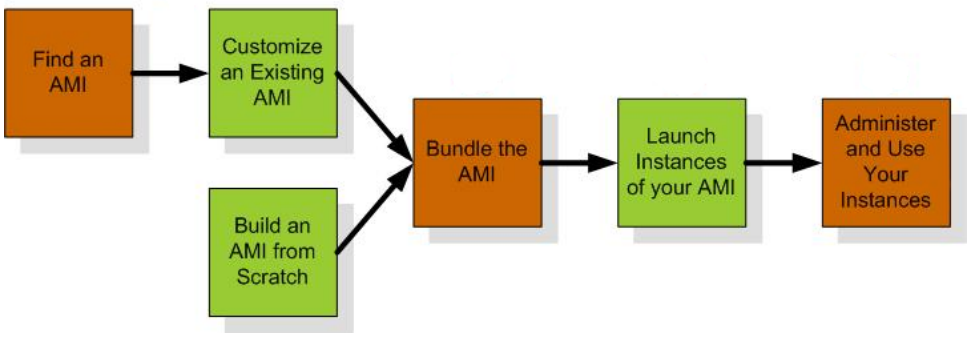

Figure 2.1 The basic workflow to use EC2 [9].

Basically an AMI image can is an image of an usual Xen guest host. The characteristics of AMI images are the environment where they are running, the possibilities of management with EC2's tools and utilities and the special network aspects. There are five different sizes of AMIs with different prices. The standard size is called m1.Small and has 1 EC2 Compute Unit (1 virtual core with 1 EC2 Compute Unit) 1.7 GB RAM 160 GB Storage 32-bit. It is possible to choose different AMI variations with different CPU power, image storage or memory up to the largest AMI size which is called c1.xlarge. It has 20 EC2 Compute Units (8 virtual cores with 2.5 EC2 Compute Units each), 7 GB Memory and 1.6 TB Storage.

2.1.3 The Amazon Simple Storage Service

The Simple Storage Service (S3) is Amazon's online storage webservice. It can be managed through a simple web interface. Every customer can allocate storage space without any limitation. Within EC2, S3 is used to store self built or reconfigured AMI images which are split up into so-called bundles. This means a snapshot of an instance's root file system is split up to smaller packets that are compressed, encrypted and signed. It is possible to build

[1] www.opensolaris.org

AMI images that use S3 Storage as persistent storage, using a so-called Elastic Block Store (EBS) volume. The data will then remain in S3 after shutting down the instance. Additional to this feature, EC2 offers the option to do snapshots of an EBS volume at a particular point in time and store them on S3. An EBS can be mounted from an EC2 instance like a usual storage device, like shown in figure 2.2. S3 stores files as objects which are organized in buckets with global names. Each bucket can be given or denied read or write access for other AWS users via ACLs. For the unique identification of objects within buckets there is a third component called key. For the use of S3 there is a SOAP and a REST programming API as well. The use of S3 is described in the Simple Storage Developer Guide [10].

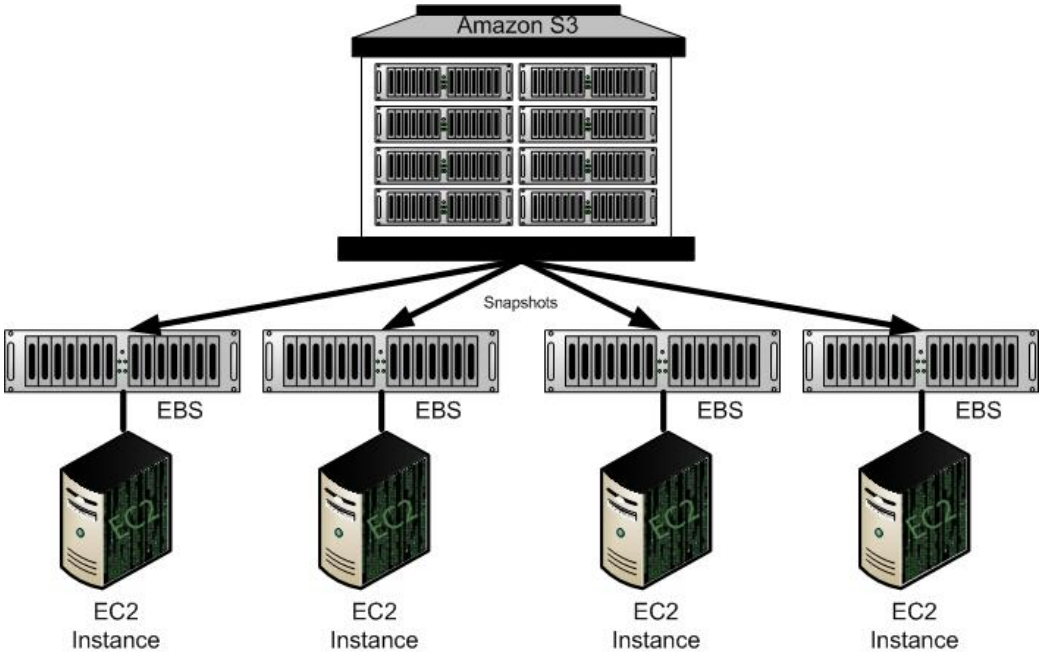

Figure 2.2 EC2 instances using EBS volumes. [9]

2.1.4 EC2 Security Aspects

For the registration of an Amazon Webservice account a credit card and email address is needed. This section describes the AWS Security concept and how a secure access to the single services and EC2 instances is realized.

Keys and Certificates for Amazon Webservices

During the signing up process for an Amazon Web Services account at the AWS home page [11] the password is set with the email address as username. AWS uses two different procedures for identification and authorization. The

first uses an HMAC algorithm to sign each request with a 40-character secret key. The second is based on X.509 certificates. Amazon creates a secret key and X.509 keypair for each user which can be downloaded from the AWS homepage's user dashboard. It is also possible to upload X.509 certificates from other sources. These certificates are then trusted as well. Thus after signing up, the user gets a X.509 certificate, an access key, a secret key, an account ID and a private key.

Secure Shell Keypairs

While instances based on self created images can be accessed using the root password that has been set up during the creation, the instances based on public AMIs must be accessed via SSH keypairs. A keypair can be created using the ec2-api-tools described on the following pages. The creation is done by the following command:

```
$ ec2-add-keypair <keypair-name>
```

This command returns the key which should be saved in a file with the keypair name. After the generation of a key pair, the public key is stored in Amazon EC2. The public key is copied to the instance's metadata if an instance is launched using the key pair name. This makes it possible to access the instance securely using the private key.

Critique

Amazon allows the resending of forgotten AWS passwords. Therefore full access to the AWS services is possible with the knowledge of the username (the customer's email address) and a password that can be resent via this email. Hence eavesdropping of the the email account's password leads to full access to the AWS services. The Harvard University's Center for Research on Computation and Society considered this issue in an evaluation as one of the most critical security aspect [12].

2.1.5 EC2 Network Aspects

Every launched instance gets an internal network address via DHCP and an internal DNS entry. This internal network address is located in the private network [6] 10.0.0.0/8 and is translated via NAT to an external IP address. The instance can be reached through this external IP address, which is bound to an EC2 specific DNS entry. This external DNS entry contains the external IP address in its name. For example an instance with the IP address 123.123.123.123 can be reached via the following external DNS entry:

`ec2-123-123-123-123.compute-1.amazonaws.com`

An internal DNS name is bound to the MAC address of the instance's virtual ethernet device and can only be reached within the EC2 environment. A DNS entry with a MAC address 12:31:39:02:ED:08 would look like the following example:

```
domU-12-31-39-02-ED-08.compute-1.internal
```

Usually the external IP addresses are allocated dynamically from Amazon's IP pool. It is possible to purchase static IP addresses as well and bind them to the instances if needed. These IP addresses are called Elastic IP addresses. They are static IP addresses which are associated with the customer's account, not with a single instance. The customer can remap these elastic IP addresses to any other running instance belonging to the account.

Instances can be grouped in so-called security groups. These groups are a set of rules, defining firewall issues such as open ports and allowed protocols. This makes it possible to run different groups of instances with different firewall security levels. For example the access via a specific port to instances in one group can be allowed or denied. One instance can be run in several groups at once and is not restricted to one single group.

2.1.6 EC2 Tools and Utilities

EC2 instances and AMI images can be managed and controlled via commands typed at the console or via programming APIs. There are also graphical user interfaces like the plugin Elasticfox [13] for the Firefox webbrowser or the CloudStudio [14] software. With these applications EC2 instances can be managed in a comfortable way. The command line tools are implemented in Java and can be used via the Windows DOS prompt and via the Unix shell. The only requirement is an installed Java Runtime Environment.

Controlling Instances

The ec2-api-tools are used to control EC2 instances. They provide commands to launch, reboot or shutdown instances and to manage security groups, elastic IP addresses and SSH keypairs. The path to the ec2-api-tools should be stored in an environment variable called $EC2_HOME and in the $PATH variable. A hidden directory called .ec2 in the operating system user's home directory contains the AWS private key and the AWS certificate. It is accessed by the different ec2-api-commands when needed.

Controlling Images

The ec2-ami-tools are meant to be installed in the AMI image or on the local machine when AMI images are created from scratch. They are needed in the bundling process of an image. This process consists of commands to upload

and download bundles, to delete bundles and to unbundle an image. For the rebundling of a running AMI instance the command ec2-bundle-vol has to be used. The command ec2-bundle-image is used to bundle an image that is created from scratch.

2.1.7 Programming Interfaces

The EC2 webservice can be accessed via the SOAP protocol or the Query API, which are both described here. Further information can be retrieved from Amazon's EC2 Developer Guide [9] and the Amazon Webservices Programming Guide [8].

The SOAP Interface

The SOAP programming interface is described in an XML document written in the Web Services Description Language (WSDL). This document strictly defines which data types may appear in SOAP requests or responses. SOAP requests and responses within EC2 follow current standards. This means every programming language with appropriate library support, such as C++, C#, Java, Perl, Python and Ruby, can be used to interact with EC2. The SOAP interface authenticates request messages through an X.509 certificate, described in section 2.1.6, instead of the procedure through access and secret keys. The SOAP interface, or tools based on this interface requires the public and private X.509 certificate files in addition to the AWS Access Key Identifiers.

The Query Interface

The Query API uses standard HTTP requests like GET or POST and a query parameter called Action or Operation. While the Action parameter is recommended by Amazon's EC2 Developer Guide [9], the Operation parameter can be used for backward compatibility with other AWS Query APIs. To handle the authentication and selection of an action each Query request must include common parameters. Operations that take lists of parameters use the `param.n` notation, where values of n are integers starting from 1. The Query API authenticates every request to EC2 through a request signature. This signature is created by constructing a string and using the secret AWS access key to calculate an RFC 2104-compliant HMAC-SHA1 hash.

Libraries

There are several code examples for Ruby, Java, Python, Perl, PHP and other languages published by Amazon and others. However this are just code samples and there are no official libraries.

2.2 Virtual Private Network Aspects

2.2.1 VPN Topology Types

There are different types of VPN topologies according to their intended purpose and the environment where they are used. For the design of a VPN environment it is important to have a look at those types. The most important ones for the most typical usages will be described on the following pages.

End-to-End Connections

The simplest VPN topology is the end-to-end VPN. It is used to establish a temporary connection between two computers over the Internet or another insecure network. In this case there are no defined server or client roles. The most common case where this topology is used, is to secure the connection of two machines via the Internet. Basically this is not a genuine VPN, but the same technologies are used to establish connections in the other VPN topologies.

Figure 2.3 End-to-end VPN

End-to-Site Networks

The end-to-site VPN topology is often used by companies where the employees need to connect to the company's firewalled network from outside. Usually this is from the home office, during business travel or while doing customer service. This scenario is also called Roadwarrior VPN or Remote Access VPN. Another case where this topology is used are secured wireless networks. Here the VPN server provides the connection to the Internet. Therefore end-to-site VPN means that single computers are connecting from outside to a firewalled private network.

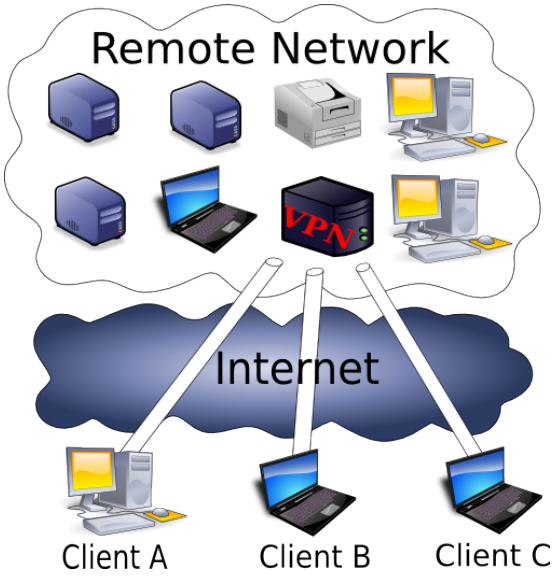

Figure 2.4 An end-to-site VPN with three clients connected

Site-to-Site Networks

A site-to-site VPN is often used to establish one permanent virtual network between two or more private networks [6]. This is useful for different facilities of one company in separate locations which are not connected through a private data cable. Here the single private networks appear like one big network. If two or more of the participating private networks are using the same network classes and subnets, it is necessary to translate their IP addresses with a NAT server [15]. For example, if there are two private networks which both use the IP addresses within the network 192.168.1.0/24, the addresses have to be translated into another IP address range or subnet on both sides.

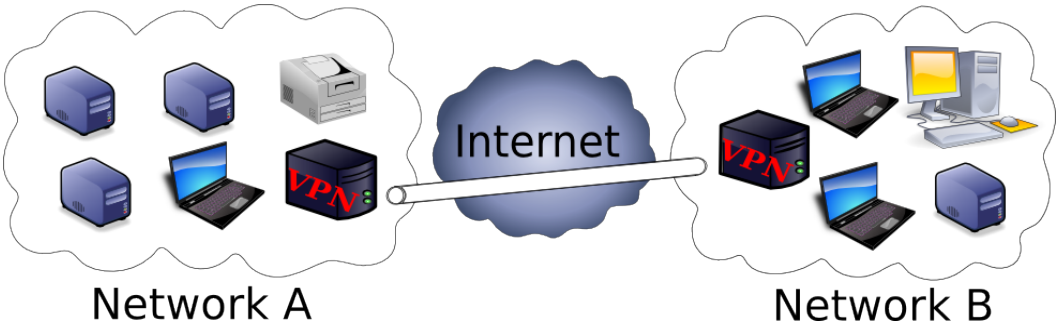

Figure 2.5 Two networks forming a site-to-site VPN

Multi-Singlepoint Networks

A Multi-Singlepoint VPN creates a new virtual network containing the participating machines without the local networks behind them. This new virtual network has to be a in subnet where none of the participating machines is connected to via their real network devices. Every single machine needs to connect as client to the VPN server and receives an IP address for the virtual device. This kind of VPN is often used for online games or other cases where different computers from different private networks need to communicate as if they were in one network. Machines participating in Multi-Singlepoint VPNs are separated from the real local networks behind them.

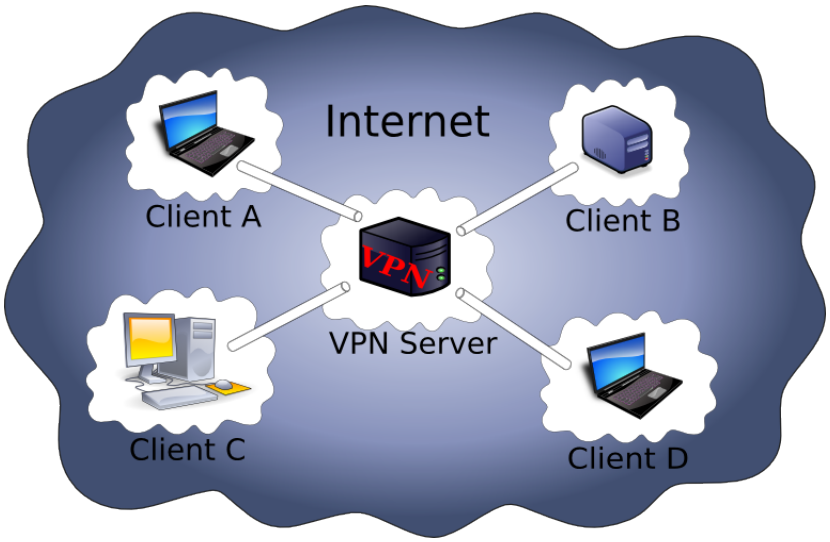

Figure 2.6 4 clients forming a Multi-Singlepoint VPN.

2.2.2 VPN Architectures

To describe different network technologies, protocols and applications, there are two common models which divide the network in several abstraction layers: the OSI Reference Model [16] and the TCP/IP model [17]. Tables 2.1 and 2.2 represent the two models. They show the different VPN technologies and protocols that will be discussed on the next pages and could be used to interconnect with Amazon's EC2 network. The descriptions will refer to the different layers of the network models.

OSI Layer	Class	Examples	VPN Types	Units
Application	application oriented	HTTP SMTP	-	data
Presentation		HTML	-	
Session		RPC NetBIOS	SSH	
Transport	transport oriented	TCP UDP	SSH/TLS GRE	segments
Network		IP ICMP	IPsec IP Routing	packets
Datalink		ARP RARP	Eth. Bridging	frames
Physical		V.28 X.21	-	bits

Table 2.1 OSI layers, example protocols, VPN technologies

TCP/IP Layer	Class	Examples	VPN Types
Application	application oriented	HTTP SMTP RPC HTML NetBIOS	SSH
Transport	transport oriented	TCP UDP	SSH/TLS GRE
Internet		IP ICMP	IPsec IP Routing
Subnet		ARP RARP	Ethernet Bridging

Table 2.2 TCP/IP layers, example protocols, VPN technologies

While different protocols and techniques can be used to build VPN tunnels on different hardware platforms running different operating systems, there are two major ways to establish a VPN connection on GNU/Linux-based machines, IP routing and ethernet bridging, which take place in different network layers. This is important for the usage within the EC2 network because of technical issues and security measures, taken by Amazon, described later on. The universal TUN/TAP driver [18], developed by Maxim Krasnyansky, is widely used on machines based on Unix-like operating systems.

IP Routing

In the IP routing mode clients and servers communicate on the OSI model's [19] network layer or on the TCP/IP model's [17] Internet layer. Therefore the routing mode only works with the IP protocol. The connection is usually established via the virtual network interface `tun` or if using IPsec the virtual `ipsec` device. The connecting client receives, most commonly via DHCP, an IP address from the VPN server. This IP address must be in in a different subnet than the part taking subnets like shown in figure 2.7. This connected machine can then be regarded as the virtual extension of the local network. Because the IP suite is standard for the Internet and has risen as quasi standard for local area networks, the IP routing mode is very flexible and can be applied in many network environments.

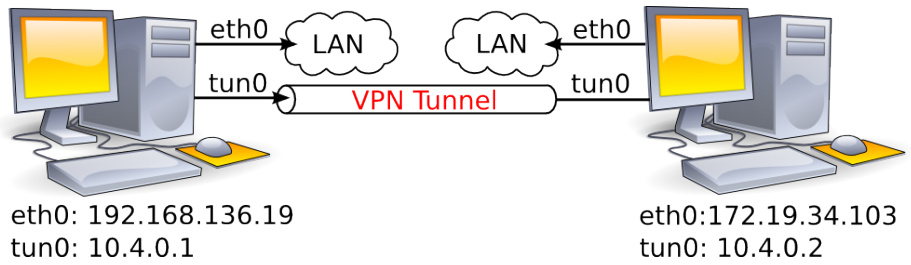

Figure 2.7 VPN endpoints need to connect with IP addresses of a different subnet than their local network addresses.

Ethernet Bridging

In the ethernet bridging mode the clients and servers communicate on the OSI model's datalink layer, which is equivalent to the TCP/IP model's subnet layer. Most commonly ethernet bridges are established via the virtual network interface `tap`, but in special cases it is possible to build ethernet bridges with IPsec as well. The two different networks behave similarly to subnets linked with hubs or switches. As hubs and switches operate in the datalink layer of the OSI model, bridging works not only between IP networks but also with IPX and other network types. Here it is necessary to build a bridge between the physical network interface card (NIC) and the `tap` device. The bridged network interface is switched into promiscuous mode, that means all incoming packets, even those which aren't addressed to the physical interface, are forwarded to the operating system's `tap` device. These packets are sent via an encrypted network tunnel to the other network, where they are accepted by another `tap` device. Here the `tap` device is bridged with a physical network interface card which routes the packets to the other local area network. A bridged network interface loses its IP address which usually is assigned to the `tap` device. Further information on ethernet bridging can be found at the websites of Vtun [18] or OpenVPN [20].

2.2.3 Protocols for Encryption and Transport

This section gives a small overview on different protocols used to establish the connections and encryption of VPNs.

The Point to Point Protocol (PPP)

Primarily the Point to Point Protocol (PPP) was intended to connect computers via phone lines to other hosts or networks and is described in RFC 1661 [21]. The Point to Point Protocol over ethernet (PPPoe) is still used today for connections via DSL. PPP is not restricted to physical connections.

It can be used to establish connections between any hosts that have network connectivity.

Transport Layer Security (TLS) and Secure Socket Layer (SSL)

The cryptographic protocols Transport Layer Security (TLS) and its predecessor Secure Socket Layer (SSL) provide data integrity and security for IP networks. They are located in the transport layer of the OSI model. TLS is described in RFC 5246 [22]. TLS version 1.0 is equivalent to SSL version 3.1. It consists of four protocols which are divided in two layers:

Handshake Protocol	Change Cipher Specification Protocol	Alert Protocol	Application Data Protocol
Record Protocol			

The TLS record protocol forms the lower layer and is used to secure the connection. The TLS handshake protocol is based on the record protocol. It performs the authentication and negotiates the keys and cryptographic algorithms. The change cipher specification protocol validates that a session changes to the negotiated cipher. The TLS alert protocol returns warnings or errors that may occur during the session. Finally the application data protocol is responsible to transport the data.

Internet Protocol Security (IPsec)

Internet Protocol Security (IPsec) is a standard set of rules and protocols. It is used to secure IP communications by authenticating and encrypting every single IP packet of a data stream. IPsec is an end-to-end security solution and operates at the OSI model's network layer. It is described in RFC 4301 [23]. IPsec also supports establishing mutual authentication between agents at the beginning of the session and negotiation of cryptographic keys to be used during the session. Bruce Schneier and Niels Fergusion, who made an evaluation of IPsec [24], were very dissappointed of IPsec. They criticize IPsec as too complex to be run be securely.

Generic Routing Encapsulation Tunnel (GRE)

The tunneling protocol Generic Routing Encapsulation (GRE) was developed by Cisco and is used to encapsulate a variety of network layer protocol packet types inside IP tunnels via a virtual point-to-point link. It is used by Cisco routers but is also part of the Linux kernel.

2.2.4 VPN Implementations

There are some ready-made software implementations which facilitate the usage of encryption and connection protocols and concepts. The most interesting ones for the connection concept with the EC2 network are explained here.

Point-to-Point Protocol over Secure Shell

This concept is described in the book "Building Linux Virtual Private Networks" [4]. Because PPP connections are not encrypted this concept wraps PPP in a Secure Shell (SSH). Thus the whole authentication and encryption is handled by SSH and the transport is done by a passwordless PPP connection. SSH supports RSA and DSA asymmetric cryptography algorithms. This concept uses SSH identities, which means keypairs are used instead of password authentication.

FreeS/WAN and Openswan

Openswan [25] is an implementation of IPsec for GNU/Linux and is the continuation of a project called FreeS/WAN. Openswan can, similar to OpenVPN, be managed via configuration files, where additional parameters like routing, authentication types, connection settings and other issues can be managed.

VTun

Vtun [18] stands for Virtual Tunnel and can set up encrypted tunnels over TCP/IP networks. It supports the Point-to-Point Protocol and the IP protocol via the "universal TUN/TAP driver". The encryption can be established through SSL or a simple XOR cipher, which is easy to decode. VTun uses private shared keys to negotiate the handshake procedure.

OpenVPN

OpenVPN [20] is an open-source VPN implementation which uses the SSL encryption protocol. It is possible to establish a VPN in IP routing mode or in ethernet bridging mode. If VPN technologies are used within the EC2 network the choice is very often OpenVPN, for example as secure Internet gateway [8] which is described in section 2.3.1. Like Vtun, OpenVPN uses the "universal TUN/TAP driver" for its virtual network interfaces and allows both, the use of X.509 certificates and pre-shared keys. The comfortable and multifunctional management of routing, authentication, connection settings and other issues via configuration files appears as an advantage. With

a properly set configuration file, a VPN can be initiated with one single command.

An Implementations Overview

Table 2.3 shows the technologies discussed, their according network layers and encryption technologies:

Implementation	OSI Layers	TCP/IP Layers	Encryption
PPP over SSH	Datalink	Subnet	SSH
Openswan	Network	Internet	IPsec
VTun	Datalink Network	Subnet Internet	TLS/SSL
OpenVPN	Datalink Network	Subnet Internet	TLS/SSL

Table 2.3 An overview of the different implementations.

2.3 Cloud Computing and Local Networks

There are already a few VPN applications for cloud computing environments which will be described here.

2.3.1 On-Demand VPN Server as Internet Gateway

A setup description of an end-to-end VPN connection (see section 2.2.1) for the use of an EC2 instance as a tunneled Internet gateway within an insecure network environment, like for example a wireless LAN, can be found in the Amazon Webservices Programming guide [8]. This connection is realized with OpenVPN. The complete Internet traffic is lead through the OpenVPN tunnel to EC2's Internet gateway. Thus eavesdropping by others within the insecure network is not possible.

2.3.2 Cohesive Flexible Technologies Corporation

A company named Cohesive Flexible Technologies Corp [26] offers the software "VPN-Cubed". This company runs an open source project called "VcubeV" as well [27]. "VcubeV" uses the same underlying VPN design like "VPN-Cubed". This design is intended for different datacenters and is not specifically optimized for EC2. It uses two or more different datacenters with one VPN server per datacenter. The different network environments with VPN servers are intended act as redundancy failover. This model focuses on high availability, which results in high running costs. Nevertheless the Harvard University's Center for Research on Computation and Society

describes EC2 services as *"fast, responsive, and very reliable"*. During approximately one year of working with EC2 in their project, only one unscheduled reboot and one instance freeze [12] occurred. Although there is a Service Level Agreement [28] for EC2, for small companies it is common to work with a standard Internet connection, which causes a short period of network disconnection when the dynamic IP address is remapped by the Internet service provider. Thus such highly available VPN solutions like VPN-Cubed or VcubeV are not appropriate in every case. The concept described in this thesis aims to be less expensive and adequate for useage in more common circumstances.

Chapter 3

The Concept

While the last chapter gave an overview of the different technologies and their options, this chapter describes the evaluation of the VPN concept design referring the EC2 network specifics. The first part contains the evaluation of the different technologies and concepts that are applied. The latter part describes the resulting design.

3.1 Determining the VPN Interconnection

3.1.1 The Topology

Internet connections of many local networks (e.g. networks of small companies) are based on dynamic IP addresses, allocated by an Internet Service Provider. Hence these external IP addresses, which hide the local networks, are changing in lease time intervals between a few hours and a day. It would be an extra effort to determine the current IP address and connect to a local VPN server from EC2's network. Therefore the end-to-site topology concept, with a VPN server in the local network and the EC2 instances as VPN clients, is not suitable. It would also be an additional effort to install VPN client software on every participating machine in the local network, which makes the multi-singlepoint concept unusable as well. Due to these facts the site-to-site topology appears as the most suitable solution. Here the local VPN server would connect to the VPN server in the EC2 network, which has statically allocated an IP address out of Amazon's IP address pool.

EC2 related Packet Loss

In common networks setting the correct routes is sufficient to make a host reachable in a site-to-site VPN. Here the EC2 network shows a distinctive problem. As soon as a packet containing an 192.168.0.0/24 IP address as

destination or source leaves an EC2 instance's eth0 network interface, it disappears in EC2's network as shown in the following figure. This can be approved by tcpdumps made on both ends.

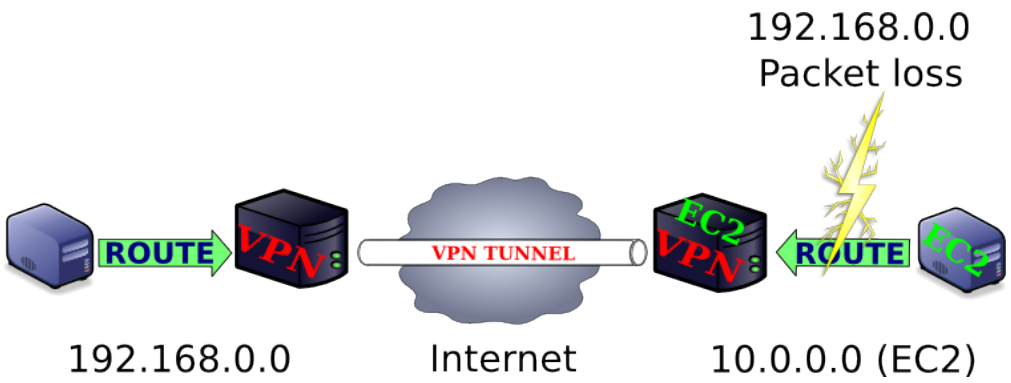

Figure 3.1 Packet loss in the EC2 network

A related behavior appears after setting up tunneling devices reaching from one instance to another. For example an IP over IP tunneling device [29] with an IP address 192.168.3.0 connecting to another instance can be easily set up on an instance and returns no error messages. However after setting up a tunnel device, pointing to another EC2 instance (e.g. the VPN server), EC2 blocks all packets sent to other EC2 instances, even those that are destined to or derived from the primarily IP addresses allocated by EC2. Only tunnels pointing outside EC2's 10.0.0.0/8 network are working. This is due to a measure taken by the EC2 administrators to prevent packet sniffing by EC2 customers [7]. This issue is related to another problem discussed in the next section 3.1.2 as well. EC2 strictly prohibits the sending of packets within its network that don't belong to its official assigned IP addresses. Thus a tunnel to the the VPN server has to be established via its external IP address. IP packets sent within an EC2 security group are declared as safe by Amazon [7]. Sending IP packets via EC2's Internet gateway could make eavesdropping possible. Therefore simple unencrypted tunnels like the "IP over IP" variant are not suitable and the tunnel connection needs to be encrypted. This requires authentication issues, executed by additional installed software. This might be the same software that is used to establish the tunnel between the local network and the EC2 network.

Alternative Solution

These facts make a plain site-to-site VPN impractical and the EC2 instances need adaption and configuration to gain access to the local area network and – conversely – be accessible from the local machines. This can be done by

configuring the clients as VPN clients, which are connected to the cloud-sided server via encrypted tunnels. Here the EC2 instances have to utilize the EC2 gateway to connect the cloud-sided VPN server via its external IP address.

This solution can be regarded as a mix of the multi-singlepoint and the site-to-site topology. Basically the resulting design is more a multi-singlepoint with the remarkable property that one connecting client acts as a gateway to its network behind. This client is located in the company's local network where the physical machines are located. In the following pages this machine will be referred as the "local VPN server".

3.1.2 The Architecture

If an EC2 instance's network interface card emits packets below the OSI model's network layer, which is true for `tap` devices used for ethernet bridging, EC2 does not handle these network packets. In this case this is true for the packets from the connected remote ethernet. The result of a test tunnel between an EC2 instance and a local server `tap` devices showed that it is generally possible to set up an ethernet bridge. Due to the system logs the tunnel was established successfully, although it was not possible to send or receive any IP traffic on the EC2 side. Therefore ethernet bridging is not possible and it is necessary to operate on the IP layer using IP routing via `tun` devices or IPsec.

3.1.3 The Implementation

Since packets wrapped by the GRE tunnel are not forwarded within the EC2 network the GRE technology is not suitable. The IPsec protocol requires a kernel patch [30] to do NAT traversal. Amazon does not allow to build own kernels for AMIs, and there is no adequate Amazon Kernel Image (AKI) available. So IPsec also turns out as inadequate. Thus there are three ways left to establish a VPN tunnel: OpenVPN and Vtun that both use SSL for encryption and the "PPP over SSH" method. According to the problem of EC2 internal connections in section 3.1.1 it is recommended to use an implementation where several clients are able to connect and retrieve routing rules. This is neither true for the "PPP over SSH method" nor for the Vtun software. Hence OpenVPN appears to be the best solution.

Access to the Cloud-Sided Server

The server needs to be accessible at a fixed address for the connecting OpenVPN clients and the local VPN server. Because of dynamic IP address allocation and therewith changing AWS-related DNS entries, this appears to

be a problem. A consistent way to connect to the cloud-sided server could be achieved through three ways:

- The allocation of an elastic IP address. An elastic IP address can be easily allocated and attached to an instance. The address needs to be attached to the instance every time when it is launched, this is indeed an additional effort, but could be automated as well. However the allocation of an elastic IP address also adds additional costs.
- Fetch the dynamic DNS entry as variable from EC2/S3. It would also be possible to write the actual DNS entry of the server instance as a variable in a file, located in a fixed S3 bucket. The clients could then fetch this file and use the variable as `remote` directive in their OpenVPN configuration files.
- Set up a dynamic DNS service on the cloud-sided server. The most elegant solution seems to be a dynamic DNS account running on the cloud-sided server. Every time the cloud-sided server is launched it connects to the dynamic DNS service which maps the new IP to the fixed DNS entry. The disadvantage of this solution is that free DNS accounts may expire when the host that needs to be mapped isn't launched for several weeks. Nonetheless dynamic DNS services that don't expire can be purchased as well. Thus the use of a dynamic DNS service appears as the best solution for this problem.

3.2 The Result

After determining different topologies, technologies and models the result is described here. For better clearness a common network design is assumed where an extra subnet that belongs to the local network infrastructure is added. Thereafter the components are described in detail. Finally the concept is completed by technical considerations and specifications.

3.2.1 The Concept Topology

The next page's figure 3.2 shows the resulting VPN design. The upper side shows the EC2 network with instances in the VPN security group and next to them EC2 instances in another security group, which are not connected via VPN. The lower side shows the local network consisting of the internal network (192.168.1.0/14) and the DMZ (192.168.2.0/24) besides the two local subnets it shows the new virtual subnet containing the EC2 instances (192.168.3.0/24).

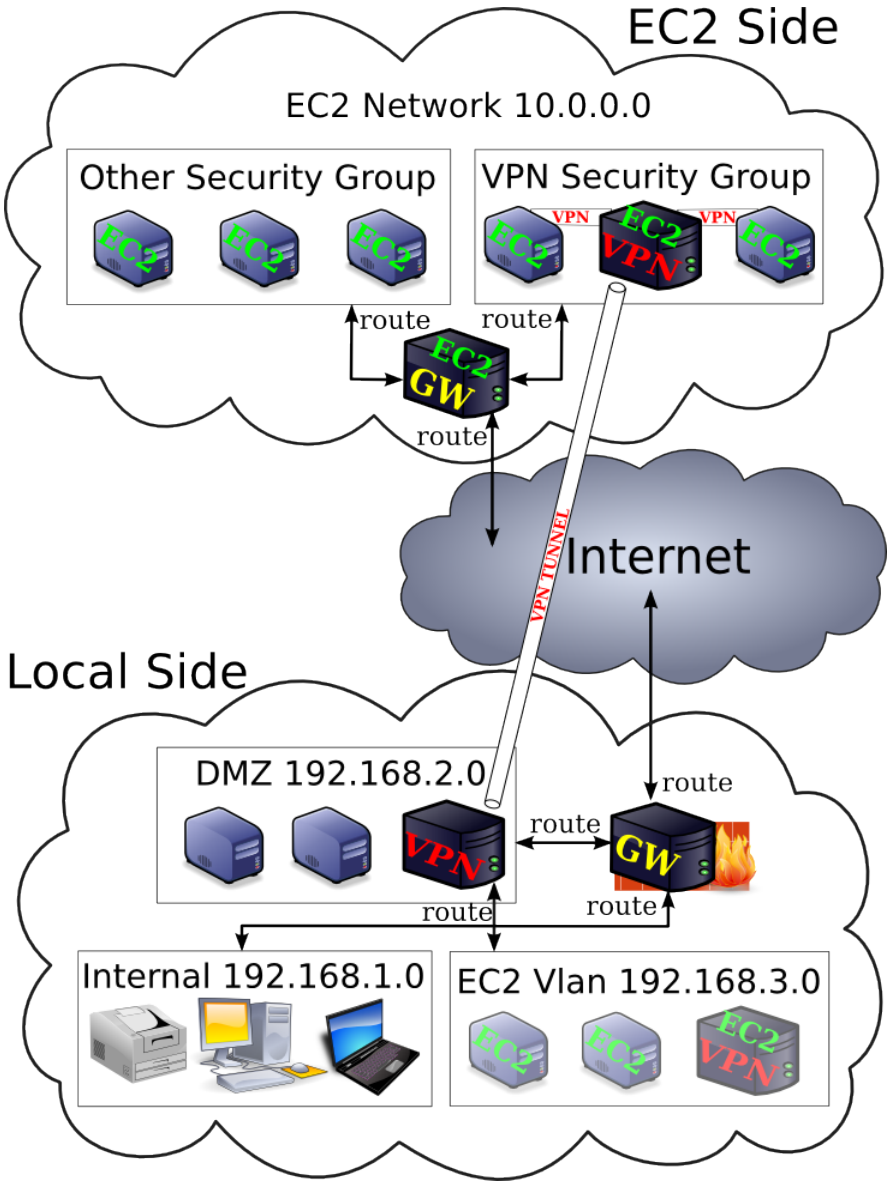

Figure 3.2 The resulting design with the two VPN servers and routes. The local network topology is assumed as the classic arrangement consisting of two subnets; one "internal" subnet containing workstations, printers and machines that doesn't or shouldn't be accessible from the Internet and one DMZ subnet where servers are partially accessible via the Internet. The concept is working with more or fewer subnets as well.

3.2.2 Details

The figure on the previous page shows an example scenario that will be described more detailed here.

The Cloud-Sided VPN Server

The VPN server located in the EC2 network forms the heart of the new virtual subnet. The tunnel to the local network, as well as the connections to all the instances converge at this point. In this example the `tun` device has the IP address 192.168.3.1 and holds the IP pool for the entire subnet 192.168.3.0/24. Furthermore this machine is running the dynamic DNS service that maps its IP address to a DNS entry that is listed in the client sided configuration files. The local VPN server and the clients automatically connect to the machine labeled by this entry.

The Local VPN Server

The local VPN server is in fact a usual VPN client which is configured as gateway between the local network (192.168.2.0/24) and the new virtual subnet (192.168.3.0/24). But since it is a physical machine that is responsible to establish the VPN connection and operate as gateway to the EC2 instances, it is called server. This machine opens a connection to the static IP address of the running EC2 sided VPN server. After a remap of the local dynamic IP address, or other network inconsistencies, the tunnel breaks down and gets re-established after a few seconds.

The Gateway in the Local Network

The gateway machine in the local network (192.168.1.0/24) contains the firewall and provides the routing. Additional to the routes pointing at the different subnets and to the Internet, a further route that leads to the local VPN server as gateway for the new subnet (192.168.3.0/24) needs to be added. If this is not possible each local machine intended to communicate with the EC2 instances needs the route added manually.

The Gateway in the EC2 Network

The gateway in the EC2 network routes the Internet traffic of the instances. This gives more performance because it disburdens the tunnel by lowering the encrypted throughput. This is common to other site-to-site VPN designs. Further informations to this concept can be found in section 3.2.3.

The EC2 Instances

EC2 instances are connected to the EC2 sided VPN server via OpenVPN. Thus they need VPN client software and appropriate certificates installed. They further require the route to the local network (192.168.1.0/24) with the local VPN server as gateway. This can be managed with OpenVPN as well.

The EC2 Instance's Local Appearance

Each connected EC2 instance gets an additional network tunneling interface `tun0`. The EC2 sided VPN server assigns a local IP address (192.168.3.0/24) to this device. Hence it can be reached in this subnet via the local VPN server and appears as a local machine.

Local Machines

The machines from the DMZ (192.168.2.0/24) are intended to interact with the EC2 instances. They need the route to the new virtual subnet (192.186.3.0/24). This route needs to be set whether on the gateway or on the machine itself (see section 4.1.2). The machines from the internal subnet (192.168.1.0/24) are then able to interact with the EC2 instances in the new subnet (192.168.3.0/24) and vice versa. This is however, depending on the setting of the local network's firewall located on the gateway machine described above.

3.2.3 Technical Specifications

OpenVPN

The latest version of OpenVPN is 2.1. It is available through the package management systems for most GNU/Linux distributions today. Version 2.1 features several improvements such as the `topology subnet` directive that allows more flexible client IP address assignments.

Operating Systems

OpenVPN runs on GNU/Linux, Windows and several Unix-like operating systems. Since most AMI images are based on GNU/Linux, the local VPN server is chosen to run GNU/Linux as well. This makes the concept a bit more consistent. Other operating systems would work as well if they are capable of running OpenVPN in version 2.1.

Amazon Machine Images

The design is designated to run instances of any AMI that are capable of running OpenVPN 2.1. Once OpenVPN is installed and configured properly on an instance, it can be rebundled and used as template for other AMIs destined for more concrete issues. Thus the design works not only with GNU/Linux based AMIs, but with Windows and Solaris AMIs as well.

Internet Connection of the Instances

A site-to-site VPN is intended as two independent networks with an encrypted connection that lets them appear like one big network. While it is common practice for multi-singlepoint and end-to-site VPNs to route the Internet traffic through the VPN tunnel, this concept allows EC2 instances to send their Internet traffic directly via Amazon's gateway in the EC2 network. The routing of Internet traffic through the encrypted tunnel is a security measure. When different clients connect from any environment to a secured network, they represent in fact security holes in the firewall. If it would be possible to access the connected client within an insecure environment, it would be possible to access the entire network where it is connected via VPN as well. Therefore other network connections of "roadwarriors" connecting via VPN, are usually closed by automated scripts. The Internet is then accessed via the VPN tunnel. EC2 instances though are running in a secured network themself. Their connectivity is restricted by the rules set in the VPN security group; and the network they are connecting to is a DMZ, which is also not underlying the highest possible security level. Because the VPN tunnel is the bottleneck of a site-to-site VPN and it is not necessary to route the EC2 instances Internet through that bottleneck, they are allowed to access the Internet directly.

Subnet Addressing

The scenario shown in 3.2 arranges the subnet 192.168.3.0/24 as new virtual subnet for the EC2 instances. While Amazon EC2 is allocating IP addresses in the 10.0.0.0/8 network, any other private network described in RFC 1918 [6] can be used for the EC2 instances. However, if a subnet in the 10.0.0.0/8 network is desired, some subnets might be available there too. The on demand Internet gateway described in section 2.3.1 allocates IP addresses in the 10.4.0.0/16 network. There might be other subnets available in the 10.0.0.0/8 subnet. A clear statement about Amazon's use of internal subnets is found neither in the "Amazon EC2 Developer Guide [9]" nor in other descriptions or manuals. In this case a question to the AWS team via the EC2 user forum [11] might be helpful.

Chapter 4

The Implementation

This chapter describes the implementation of the concept specified in the previous chapter. The first part shows the set up of the network environment, the servers and the cloud-sided instances. The second part describes how the VPN infrastructure can be established.

4.1 Preparing the Infrastructure

4.1.1 Limitations

Due to the failure of ethernet bridging described in point 3.1.2, it is necessary to use `tun` devices. This is the cause of some limitations between the VPN subnet containing the EC2 instances and the local network. The main aspect is that ARP requests between the two subnets don't work. Thus IP broadcasting is not possible and routes must be set manually.

4.1.2 Prerequisites in the Local Network

Generally it is better not to touch a running firewall of a local network, but there is at least one change required to make this VPN concept working. The port 1194 needs to be opened for incoming and outgoing traffic. 1194 is the OpenVPN standard port but other ports can be used as well. This port needs to be defined in the OpenVPN configuration files described in 4.2.5 and 4.3.5. It needs to be mapped to the internal IP address of the VPN server in the LAN. The encrypted tunnel will use this port to communicate with the instances in the EC2 Network. Therefore it is necessary to open this port for the EC2 network as well, which is described in the next section.

It is recommended to set a default route to the new subnet on the Internet gateway if possible. Usually this is the same machine that runs the firewall. This example uses 192.168.3.0/24 as new subnet.

If this is not possible, every participating local machine needs the route to the EC2 network set manually by the following command:

```
# route add -net 192.168.3.0 netmask 255.255.255.0 \
  gw 192.168.1.201 dev eth0
```

In this example the network where the EC2 instances are located is 192.168.3.0, the local VPN server has the IP address 192.168.2.201 and the network interface is eth0. The backslash (\) can be omitted if the command is typed in a single line.

4.1.3 Defining a Security Group

Like the machines in the local network, the EC2 instances need an open port to communicate with the remote network. For this case we can use the security groups mentioned in section 2.1.5. Another reason for the use of security groups is that not every running EC2 instance is wanted to run in the local network which can be achieved by leaving them out of the VPN-related security group. Thus it is required to generate a security group and open the VPN port with the following commands:

```
$ ec2-add-group VPNgroup -d ``EC2 VPN group''
$ ec2-authorize VPNgroup -P udp -p 1194 -s 0.0.0.0/0
$ ec2-authorize VPNgroup -P tcp -p 22 -s 0.0.0.0/0
```

Here a security group "VPNgroup" with the description "EC2 VPN group" is created with an open udp port 1194 accessible from any IP address (0.0.0.0/0). Further the port 22 is opened for access via SSH.

4.2 Preparing the Local VPN Server

After the necessary settings in the participating networks the VPN servers can be set up. At first the VPN server in the local network will be installed and configured, which is shown here. For the installation of the local VPN server it takes several steps described below:

- Set up a GNU/Linux server.
- Install OpenVPN.
- Generate certificates and keys.
- Enable packet forwarding.
- Set up and and start OpenVPN.

4.2.1 Set up a GNU/Linux Server

As mentioned before for the local VPN server any operating system capable of running OpenVPN in version 2.1 could be used. In this example, the GNU/Linux distribution Debian[1] is chosen. It is enough to start with the installed base system. Additional packages that may be used are fetched automatically by the Debian package management tool in the next step. Package management systems of other GNU/Linux distributions should do this alike.

4.2.2 Install OpenVPN

OpenVPN can be installed on a Debian-based operating system by the following command:

```
# apt-get install openvpn
```

The `apt-get` command belongs to Debian's package management tool APT.

4.2.3 Create the Public Key Infrastructure

OpenVPN provides a collection of scripts called EasyRSA. With these scripts certificates and keys, used for authentication and encryption, can be created in a comfortable way.
These scripts are templates and need to be customized. The `easy-rsa` directory is located in `/usr/share/doc/openvpn/examples/` and should be copied to `/etc/openvpn/`, where the scripts can be edited. The setup of the Public Key Infrastructure (PKI) is done with the following steps:

- Set up a certification authority (CA).
- Create server keypair.
- Create client keypairs.

To set up of the certificate authority, the `easy-rsa` directory contains a file named `vars` with several variables that need to be defined via a text editor. The most important ones are `KEY_SIZE`, `KEY_COUNTRY`, `KEY_EMAIL` and similar ones that are used to specify characteristics of the certificate's owner. After adapting the variables in the `vars` file, the variables must be imported to the shell context to initialize the certification authority. This is done by the following commands:

```
$ . ./vars
$ ./clean-all
```

[1] www.debian.org

Now the keypair for the certificate authority can be created. It is used later on to generate the client and server key pairs. The `ca.crt` is copied to the server and the clients to validate the opponent's key pair. It is highly recommended to keep the `ca.key` at a save place. The generation is issued by the following command:

```
$ ./build-ca
$ ./build-dh
```

These two commands return the `CA-Key` and `CA-Cert`. The certificate authority is based on this keypair. It is used to generate the keypairs for the server and the client machines which will be generated in the next step:

```
$ ./build-key-server server
$ ./build-key localclient
$ ./build-key ec2client
```

Thereafter the directory `/etc/openvpn/keys` should contain the following files that need to be copied to the server or the clients: `dh1024.pem, ca.crt, ca.key, server.crt, server.key, localclient.crt, localclient.key, ec2client.crt, ec2client.key`.
The table 4.1 shows the file's purposes and where they are destined to.

Filename	Location	Description	secret
ca.crt	server and clients	root CA certificate	no
ca.key	somewhere safe	root CA key	yes
dh1024.pem	server	Diffie Hellman	no
server.crt	server	server certificate	no
server.key	server	server key	yes
vpnclient.crt	client	client certificate	no
vpnclient.key	client	client key	yes

Table 4.1 An overview of different keys and certificates.

The keys and certificates can now be moved with the `scp` command to their destinations.

4.2.4 Enable Packet Forwarding

Because the machine is intended as a gateway, it must be able to forward IP packets. For that purpose the parameter `net.ipv4.ip_forward` in the file `/etc/sysctl.conf` needs to be set to `1`. The enabling of packet forwarding on other operating systems may differ. Admittedly it is a common busines to set up routers that should be well documented.

4.2.5 Set up and Start OpenVPN

OpenVPN can be controlled via command line with the necessary informations as parameters or via configuration files. A complete list of possible directives and options can be found on the manual page, which is available online as well [20]. The next listing shows the configuration file with the directives necessary to connect to the cloud-sided server:

```
1  client
2  remote <EC2-server-dynamicDNS-name>
3  port 1194
4  proto udp
5  dev tun
6  ca ca.crt
7  cert client01.crt
8  key client01.key
9  route 10.0.0.0 255.0.0.0 192.168.3.1
10 ns-cert-type server
11 verb 3
```

Line one and two declare the machine as VPN client that connects to the specified `remote` address.

The lines three to five define the `tun` device and the use of the UDP protocol. The defined network `port` is 1194.

Line 6 to 8 specify the paths to the different keys and certificates. In this example they are located directly in `/etc/openvpn/`.

Line 9 contains a routing directive that sets the route to the new virtual subnet. The IP address 192.168.3.1 represents the cloud-sided VPN server as gateway.

The `ns-cert-type` in Line 10 is a security measure to prevent man-in-the-middle attacks. It verifies that the server certificate's `nsCertType` field is set to "server".

The latter line sets the system log verbosity to 3. This is a common value for ready set up machines. If needed the details can be increased through higher values.

Start OpenVPN

OpenVPN can be started with the standard configuration file `client.conf`, located in `/etc/openvpn` as deamon or with a specified configuration as parameter. It is possible to start OpenVPN automatically at boot time which is however not recommended for the local VPN server.

4.3 Preparing the Server in the Cloud

After preparing the network environment for the new VPN, this part describes the set up of the VPN server inside the EC2 network. In this example the author's notebook was used to run the commands and connect to EC2. The notebook runs under GNU/Linux, thus the quoted commands are typed in the bash shell and the connection is established via SSH directly instead of Putty for which would be appropriate for Windows machines. The most EC2-related instructions could also be directed via graphical user interfaces like CloudStudio or Elasticfox. To set up the server it takes the following steps:

- Select an AMI and launch an instance.
- Set up a dynamic IP address service.
- Install OpenVPN.
- Enable packet forwarding.
- Copy the keypair.
- Set up and start OpenVPN.
- Test connection.
- Bundle and register to a new AMI.

4.3.1 Select an AMI and Launch an Instance

This command shows a list of the available AMIs:

```
$ ec2-describe-images -a | less
```

Any GNU/Linux based AMI is usable. In this case the "Ubuntu Intrepid Base Install" AMI is chosen. The AMI ID is `ami-1a5db973`. It is launched by the following command:

```
$ ec2-run-instances ami-1a5db973 -g VPNgroup -k gsg-keypair
```

This lets the instance appear in the security group "VPNgroup" which was created above. The `gsg-keypair` will be used to access the instance in the next step. After approximately one minute of waiting for the instance to boot up the `ec2-describe-instances` command returns the internal and external DNS entries (see2.1.5). With SSH it is now possible to connect with the keypair:

```
ssh -i .ec2/id-gsg-keypair \
root@ec2-72-44-53-18.compute-1.amazonaws.com
```

Now the instance's command prompt appears and the root access is accomplished.

```
root@domU-12-31-39-02-F1-C2:~#
```

The instance's hostname is similar to the internal DNS entry which contains the MAC address as well.

4.3.2 Set up the Dynamic DNS Service

The connecting clients need a fixed URL or IP address to connect the cloud-sided server automatically. As described in section 3.1.3 this is realized through a dynamic DNS service. There are several different dynamic DNS services that offer basic functionality for free. In Chapter 7 of the Amazon Webservices Programming Guide [8] a detailed tutorial for such a setup can be found.

4.3.3 Install OpenVPN

The installation of OpenVPN is done similar to the OpenVPN installation on the local VPN server described before in section 4.2.2. In this example the Debian-based Ubuntu Linux[2] distribution is used, hence this is done with the apt packaging system as well.

4.3.4 Enable Packet Forwarding

Like the local VPN server, this machine operates as a gateway. Thus the flag `net.ipv4.ip_forward` in the file `/etc/sysctl.conf` must be set to `1` likewise.

4.3.5 Set up and start OpenVPN

The next step is to edit the configuration file. This file controls the behavior of the OpenVPN software. Here the connection type, encryption issues, security aspects, routing set up and many other functions are defined. The configuration file is called `server.conf` and is usually located in `/etc/openvpn/`:

[2] www.ubuntu.com

```
1  server 192.168.3.0 255.255.255.0
2  topology subnet
3  port 1194
4  proto udp
5  dev tun
6  dh keys/dh1024.pem
7  ca keys/ca.crt
8  key keys/server.key
9  cert keys/server.crt
10 client-to-client
11 client-config-dir /etc/openvpn/ccd
12 keepalive 10 60
13 verb 3
14 route 192.168.1.0 255.255.255.0 192.168.3.2
15 push "route 10.0.0.0 255.0.0.0"
16 push "route 192.168.3.0 255.255.255.0"
```

The first line defines the machine as VPN server that provides the defined IP address pool. This means that this machine is running the OpenVPN service in server mode and other machines connect to it as clients. The `server` entry includes the `tls-server` and the `push` directive. `tls-server` enables TLS encryption and assumes the server role during TLS handshake. OpenVPN is designed as a peer-to-peer application. The designation of the client or server directive is only for negotiating the TLS control channel. The subnet 192.168.3.0 with the appropriate netmask 255.255.255.0 behind the `server` entry signifies that the server is providing IP addresses for the connection clients common to a DHCP server.

The entry `topology subnet` in line two assigns one IP address to every connecting machine. This entry is necessary because otherwise the client IP addresses would be organized in so called "mini subnets". This would consume four IP addresses per connection. The `topology subnet` directive is initially available since OpenVPN version 2.1. The former mini subnet arrangement was necessary for clients running Windows and is now obsolete.

The lines three to five define the connection. The tunnel is set up through a tunneling device, the network port is 1194 and the connection is established via the UDP protocol.

The directives in line 6 to 9 contain the paths to the certificates and keys. In this example they are all located in `/etc/openvpn/keys/`. The OpenVPN server mode handles multiple clients through the `tun` interface. Hence it acts effectively a router. In line 9 the `client-to-client` flag lets OpenVPN internally route client traffic instead of pushing all client-originating traffic to the `tun` interface.

The `client-config-dir` is used for specific operations executed when the local VPN server connects. This is described later on.

In line 12 the directive `keepalive 10 60` lets the server ping its connections every 10 seconds. If there is no response for more than 60 seconds, the connection will be closed.

For a moderate verbosity in the system log the `verb` flag in line 13 is set to `3`. Finally in lines 15 and 16 the routing directives are pushed to the connecting clients.

The `/etc/openvpn/ccd/` directory is used for specific adjustment of the local VPN server. It contains a file with the client's X509 common name in this example the common name is `localVPN`. This name was chosen during the generation of the local VPN servers' keypair, described in section 4.2.3.

```
iroute 192.168.1.0 255.255.255.0
push route "192.168.1.0 255.255.255.0"
```

When the local VPN server connects to the cloud sided server the TLS handshake takes place. The common name defined in local VPN servers' keypair is then read and processed by OpenVPN. The `ccd` directory contains a file with that common name and processes the commands listed there. This sets the routes to the local network.

Start OpenVPN

If OpenVPN is installed on the Ubuntu operating system, it is set to start automatically on boot time. However when using different GNU/Linux derivates or other operating systems it might be necessary to configure this manually. The best way to check if OpenVPN really starts correctly on boot time is a simply reboot.

4.3.6 Test the Connection

Before the instance is bundled to a new AMI it is recommended to test the connection through the tunnel. The cloudsided server's `tun` device is assigned to the IP address 192.168.3.1, thus the other end of the tunnel is represented by the local VPN server with 192.168.3.2. In this example the IP address of the local VPN server's eth0 interface card is 192.168.2.201.

```
$ ping 192.168.3.2
$ ping 192.168.2.201
```

This two commands assure the tunnel is up and the local network is reachable. On the local VPN server the IP address 192.168.3.1 must be pingable now as well.

4.3.7 Bundle the Instance to a new AMI

Now that the cloud-sided server is set up correctly and ready to use, the running instance can be bundled to a new AMI. This is done by the following command:

```
$ ec2-bundle-vol -k <private key> -u <user-id> -c \
<certificate> -r i386 -p vpnserver
$ ec2-upload-bundle -b <bucket-name> -m      \
  /tmp/vpnserver.manifest.xml  \
   -a <accesskey> -s <secret key>
$ ec2-register <bucket-name>/vpnserver.manifest.xml
```

The `ec2-bundle-vol` command generates bundles of a new AMI and stores them in the running instance's `/tmp` directory.
Afterwards `ec2-upload-bundle` copies the bundle to the S3 bucket.
Finally `ec2-register` makes the new AMI available. After the registration the new AMI ID is displayed:

```
IMAGE  ami-858463ec
```

The VPNserver AMI is now ready to use.

4.4 Prepare EC2 Client Template Images

The AMI images of EC2 instances intended to act as VPN clients need the following steps to be set up.

- Select an AMI and launch an instance.
- Install OpenVPN.
- Set up OpenVPN.
- Copy the keypair.
- Bundle and register the AMI.

Basically most of these steps are similar to the ones for the cloud sided server, described in the last section. First an AMI has to be chosen and launched in the VPN security group. A dynamic DNS service is not needed here. Because the instance is intended to run as VPN client, after the installation of OpenVPN the configuration file different looks different:

```
client
nobind
port 1194
proto udp
dev tun
ca ca.crt
cert client02.crt
key client02.key
ns-cert-type server
route 192.168.2.0 255.255.255.0 192.168.3.2
verb 3
```

In essence the configuration file is the same as the one of the local VPN server. The only difference is the entry for the `route` directive. It is not pointing to the EC2 network (10.0.0.0/8) with the cloud-sided server (192.168.3.1) as gateway, but to the local network (192.168.2.0/24) using the local VPN server (192.168.3.2) as gateway. Furthermore the client AMI requires another keypair to connect.

4.5 Initiating the VPN Infrastructure

With a prepared infrastructure and ready set up servers and instances it is now possible to establish the new VPN manually. This is done in three steps described here:

- Starting the cloud sided server.
- Establishing the tunnel.
- Adding instances to the local network.

At last it is explained how the VPN can be shut down.

4.5.1 Starting the Cloud-Sided Server

The cloud-sided server is the heart of the new subnet and is therefore the first machine to be started. With a correct configuration described in the chapter above it is sufficient to launch an instance of the appropriate AMI in the VPN-related security group:

```
$ ec2-run-instances ami-858463ec\
  -g VPNgroup -k gsg-keypair
```

Alternatively this can be done with other tools like Elasticfox or CloudStudio as well. During the instance's boot process, OpenVPN is started as daemon and running in the background. Now the instance is running and listening on port 1194 for connecting clients.

4.5.2 Establishing the Tunnel

That followed the local VPN server can connect to the cloud-sided server and establish the tunnel. Here OpenVPN can be run as daemon as well. The OpenVPN service is started by the following command:

```
# /etc/init.d/openvpn start
```

4.5.3 Adding Cloud Instances to the Local Network

Now that the two servers are running the EC2 instances can be added to the new local subnet. The AMIs described in section 4.4 are defined to start OpenVPN on boot time and therewith to connect to the cloud-sided VPN server automatically. Hence it is sufficient to launch an instance of the AMI in the correct security group ether with a graphical suer interface or in the command line:

```
$ ec2-run-instances ami-e69c7b8f \
  -g VPNgroup -k gsg-keypair
```

The tunnel and routing is initiated automatically and the instance should be reachable via its assigned `tun0` IP address. In this example it is an IP address between 192.168.3.3 and 192.168.3.255. If a VPN instance should be removed from the VPN infrastructure it is sufficient to terminate it like every EC2 instance. There are no further steps to be done.

4.5.4 Shutting down

If the VPN infrastructure is no longer used, all machines running in the security group `VPNgroup` should be shut down and the OpenVPN process on the local VPN server should be stopped. This command returns a list of all running VPN instances:

```
$ ec2-describe-instances
```

The single instances can now be terminated:

```
$ ec2-terminate-instances <instance-ID_a> <instance-ID_b>
   ... <instance-ID_n>
```

To make sure that all instances are terminated it is recommended to run the `ec2-describe-instances` command once again.

On a Unix-like operating system the VPN infrastructure can be shut down with one single command:

```
$ ec2-terminate-instances $( ec2-describe-instances \
 | grep VPNgroup -A 2  \
 | grep -o -E '[^kmr]i\-([a-z]|[0-9]){8}' )
```

This filters the instance IDs from the output of the `ec2-describe-instances` command and forwards it to `ec2-terminate-instances`. This shuts down all instances running in the `VPNgroup`, including the cloud-sided VPN server.

Chapter 5

Analysis

After discussing the concept and its realization this chapter pays attention to the result's efficiency and benefit.

5.1 EC2 Instances and VPN

It was a clear aim to realize a site-to-site VPN, which turned out as not practicable. This is caused mainly by the lack of an accessible standard gateway for the EC2 instances. The first attempt for this VPN concept consisted of a VPN server in the local network and one in EC2's network, performing the VPN tunnel and managing the routing. It was planned to make the EC2 instances available in the local network by adding a route command to the cloud-sided server and another one that adds it as gateway to the local network. In a common site-to-site VPN even these steps would not be necessary. The routing through the VPN tunnel would then be done by the network's standard gateway, which is in this case managed by Amazon. Amazon's security measures to prevent packet sniffing might be very effective – in this case they are an obstacle.

5.1.1 The Network Address Translation Attempt

An alternative solution was searched in Network Address Translation. It succeeded to perform an IP masquerading on the cloud-sided server, what made the EC2 instances reachable from the local network. Although the instances were then still not capable to connect machines in the local network (e.g. 192.168.2.0/24). The IP masquerding on the cloud-sided server lets all forwarded packets appear like they were emitted by itself. Thus they are forwarded in the EC2 network and reach their destination. This was realized through iptables commands:

```
$ modprobe iptables_nat
$ iptables -t nat -A POSTROUTING -s 192.168.0.0/16 \
  --out-interface eth0 -j MASQUERADE
```

Attempts to manipulate packets in the backward direction through destination network translation (DNAT) were not pursued because this design respects the security measures taken by Amazon and does not try to work around them. It would have been a big effort build up an VPN infrastructure where every instance address needs to be translated three times: The IP masquerading from the local machines to the EC2 instances, the destination network address translation vice versa and a full nat on the cloud-sided server to make them appear in the local network.

5.1.2 General Experience with EC2

EC2 turned out to be easy and comfortable to handle. It is still labeled beta, which does not mean that it is not reliable, but that it is still evolving quickly. During four months of Thesis work EC2 announced service level agreements, changed the pattern of instance's internal DNS name, an EU availability zone was added and Windows-based AMIs were provided.

Chapter 6

Summary

6.1 Conclusion

Within EC2 there are restrictive network related security measures taken by Amazon. This causes extra effort to set up a VPN environment. Although, after considering these special circumstances, a feasible VPN interconnection can be established. Due to the performance loss for IP traffic, send through the tunnel, this constrains, like in other VPNs, the use of EC2 instances in the local area network. The failure of ethernet-bridging is a deficit that might not be resolved in near future. This might be due to the specific situation where packet sniffing needs to be prevented or because of issues implicated by the used virtualization technology.

6.2 Future Perspective

Cloud computing is a young technology that offers new opportunities. Especially when it is offered as a service to customers it is an inexpensive way for short-term computing solutions. The possibility to run cloud instances in the local network represents an another interesting aspect. With the combination of cloud computing and VPN in mind, it would be a benefit to have a cloud computing environment with a small subnet for each customer as an extra service. This might be an additional security aspect as well.

Bibliography

[1] Nicholas Carr. *The Big Switch: Rewiring the World, from Edison to Google*. W. W. Norton, January 2008. ISBN: 978-0-393-06228-1.

[2] Fran Berman, Anthony Hey, and Geoffrey Fox. *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley & Sons, April 2003. ISBN: 978-0-470-85319-1 (HB).

[3] Hewlett-Packard Labs Milojicic Dejan. Cloud computing: Interview with russ daniels and franco travostino. *IEEE Internet Computing*, 12(5):7, 2008. DOI: 10.1109/MIC.2008.97 ISSN: 10897801.

[4] Brian Hatch Oleg Kolesnikov. *Building Linux virtual private networks (VPNs)*.
New Riders Publishing, Indianapolis, IN, USA, 2002. ISBN: 1-57870-266-6.

[5] Paul Ferguson and Geoff Huston. What is a vpn. *Whitepaper*, Apr. 1998. DOI: 10.1.1.28.972.

[6] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address Allocation for Private Internets. RFC 1918 (Best Current Practice), February 1996.

[7] Amazon web services: Overview of security processes.
Security Whitepaper URL: http://aws.amazon.com, Nov. 2008.

[8] James Murty. *Programming S3, Ec2, Sqs, and Fps*. O'Reilly Media, Inc, Sebastopol, 2008. ISBN: 0596515812.

[9] Amazon Web Services LLC. *Amazon Elastic Compute Cloud Developer Guide*, 2008-05-05 edition, May 2008. Amazon EC2 (API Version 2008-05-05).

[10] *Amazon Simple Storage Developer Guide API version 2006-03-01*, 03 2006.

[11] Amazon web services home page. URL: http://aws.amazon.com, Jan. 2009.

[12] Simson L. Garfinkel. An evaluation of amazon's grid computing services: Ec2, s3 and sqs. Technical report, Center for Research on Computation an Society Harvard University. URL: ftp://ftp.deas.harvard.edu/techreports/tr-2007.html.

[13] *Website Elasticfox EC2 plugin for the Firefox Browser*, 2008. URL: http://sourceforge.net/projects/elasticfox/ last visited 2.1.2009.

[14] *Website CloudStudio Graphical User Interfache for EC2*, 2008. URL: http://www.service-cloud.com/ last visited 2.1.2009.

[15] P. Srisuresh and K. Egevang. Traditional IP Network Address Translator (Traditional NAT). RFC 3022 (Informational), January 2001.

[16] L. Andersson and T. Madsen. Provider Provisioned Virtual Private Network (VPN) Terminology. RFC 4026 (Informational), mar 2005.

[17] R. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122 (Standard), October 1989. Updated by RFCs 1349, 4379.

[18] *Website Vtun*, 2009. URL: http://vtun.sourceforge.net last visited 2.1.2009.

[19] H. Zimmermann. Osi reference model–the iso model of architecture for open systems interconnection. *IEEE Transactions on Communications*, 28(4):425–432, 1980. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1094702.

[20] *Website OpenVPN*, 2009. URL: http://openvpn.net last visited 2.1.2009.

[21] W. Simpson. The Point-to-Point Protocol (PPP). RFC 1661 (Standard), July 1994. Updated by RFC 2153.

[22] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), August 2008.

[23] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301 (Proposed Standard), December 2005.

[24] Niels Ferguson and Bruce Schneier. A cryptographic evaluation of ipsec. Technical report, 2000.

[25] *Website Openswan.*
URL: http://www.openswan.org/
last visited 2.1.2009.

[26] *Website Cohesive Flexible Technologies Corp*, 2008.
URL: http://www.cohesiveft.com/vpncubed/.

[27] *Website VcubeV Open Source Project*, 2008.
URL: http://www.cohesiveft.com/Developer/.

[28] *Service Level Agreement for EC2*, 10 2008.
URL: http://aws.amazon.com/ec2-sla/
last visited 2.1.2009.

[29] C. Perkins. IP Encapsulation within IP. RFC 2003 (Proposed Standard), October 1996.

[30] *Website Openswan Wiki*, 2009.
URL: http://wiki.openswan.org/index.php/Openswan/install
last visited 2.1.2009.

[31] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), November 1982. Updated by RFC 5227.

[32] P. Vixie, O. Gudmundsson, D. Eastlake 3rd, and B. Wellington. Secret Key Transaction Authentication for DNS (TSIG). RFC 2845 (Proposed Standard), May 2000. Updated by RFC 3645.

[33] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280 (Proposed Standard), May 2008.

[34] Neil Smyth. Xen virtualization essentials. Technical report, June 2008. http://www.virtuatopia.com.

Glossary

Amazon Kernel Images (AKI)

AMI images can not be build with a customized kernel and rely on predefined kernel images. During the launch of an instance it is possible to chose between several predefined Amazon Kernel Images.

APT Package Management System

The Advanced Packaging Tool (APT) is an open source tool that simplifies the process of managing software packages. It provides amongst others routines to install, uninstall, search or update software packages and is used by Debian and its derivates.

ARP

The Address Resolution Protocol (ARP) is used to translate a IP address to a host's link layer (hardware) address, which is also known as MAC address. ARP is defined in the RFC 826 [31]

Certification Authority

A certification authority (CA) is an entity that provides digital certificates for the use by other parties. A certification authority is also an example for the trusted third party concept.

DNS

The Domain Name Service (DNS) is a protocol that translates domain names to IP addresses. Domain names are easier to handle for humans than IP addresses.

Dynamic DNS

Dynamic DNS is a technique that allows a networked device to let a domain name server change the active DNS configuration. Mostly this is used to allocate dynamic IP addresses to a fixed DNS entry. Dynamic DNS is defined in RFC 2845 [32]

EC2 Compute Units

EC2 Compute Units define the processing power which can be expected from each instance. Due to the XEN Technology, the measure is merely split up in two categories: “moderate” and “high”.

Failover Technology

The failover technology switches from one server that became unavailable to a secondary server that takes over the role of the failed one. Usually the two servers are monitoring each other via a hartbeat.

HMAC algorithm

A Hash Message Authentication Code (HMAC), is used to authenticate messages. It is calculated using an algorithm involving a cryptographic hash function combined with a secret key. HMAC may be used to verify both, the data integrity and the authenticity of a message simultaneously.

Lease Time

In the dynamic mode DHCP provides the client a lease on an IP address for a period of time. This could range from hours to months, depending on the stability of the network. The DHCP client can request renewal of the lease on its current IP address any time before the lease expires.

Loopback Device

In Unix-like operating systems, a loop device is a pseudo-device that makes a file accessible as a block device. A loop device must be connected to an existing file in the filesystem to be usable. If the file already contains a file system, it may then be mounted like a disk device.

MAC Address

A Media Access Control Address (MAC Address), also known as Ethernet Hardware Address is a quasi-unique identifier assigned to network interface cards by the manufacturer for identification.

Man-in-the-Middle Attack

The man-in-the-middle attack is a form of active eavesdropping in cryptography. The attacker initiates independent connections to the victims by relaying messages between them and making them believe that they are talking to each other over a private connection. In fact the conversation is controlled by the attacker.

NAT Traversal

When IP packets are crossing gateways or firewalls that are translating private IP addresses into public IP addresses (NAT) this is called NAT traversal.

OpenSolaris Operating System

OpenSolaris is an Unix-based open source operating system, based on Solaris from Sun Microsystems.

Packet Forwarding

Describes the relaying of packets in a computer network from one network segment to another by nodes. In Unix-based operating systems packet forwarding is turned off by default and must be enabled when the host is intended to run as a router or gateway.

Packet Sniffing

Packet sniffing can be regarded as eavesdropping of network packets destined for other hosts. Here it is necessary to run the ethernet interface card in promiscuous mode. Usually the tools used for packet sniffing are common network analyzing programs.

Promiscuous Mode

A network interface running in promiscuous mode lets all received traffic pass to the CPU rather than just the packets that are destined to it. Setting up a tunnel device with a different IP address is not equivalent to promiscuous mode.

SLA

A Service Level Agreement (SLA) is an agreement between a customer and a service provider. In this case it means that a customer receives a Service Credit if the annual uptime percentage for drops under 99,95% (approximated 4.5 hours per year) [28].

Tcpdump

A Tcpdump is a common packet sniffer. It allows the user to intercept and display (mostly TCP/IP) packets being transmitted or received at the host's network device. In this thesis project it was used to ensure that network packets, sent from another host, are received. This is helpful to localize the point where ping-ICMP packets get lost.

WSDL

WSDL is a language, based on XML, that provides a model for describing Web services. It does not depend on a programming language or a specific protocol.

X.509 Certificates

The X.509 standard is defined by the International Telecommunication Union (ITU). It describes a public key infrastructure and is a common standard that is described in RFC 5280 [33].

Xen virtual machine monitor

Xen is a virtual machine monitor for several CPU architectures, initially created by the University of Cambridge Computer Laboratory and is now developed and maintained by a company called XenSource. With Xen it is possible to run different guest operating systems on the same computer hardware concurrently. The Xen hypervisor is used to manage the Xen system's structure as the most privileged layer [34].

XOR Cipher

The XOR cipher is a quite primitive cipher that operates after the following principles $A \oplus 0 = A$, $A \oplus A = 0$ $(B \oplus A) \oplus A = B \oplus 0 = B$. It is used in cases where no particular security is needed.

List of Abbreviations

ACL	Access Control List
AKI	Amazon Kernel Image
AMI	Amazon Machine Image
API	Application Programming Interface
CA	Certification Authority
DHCP	Dynamic Host Control Protocol
DSA	Digital Signature Algorithm
DSL	Digital Subscriber Line
EBS	Elastic Block Store
EC2	Elastic Compute Cloud
GRE	Generic Routing Encapsulation
HTML	Hypertext Markup Language
HTTP	Hyper Text Transfer Protocol
ICMP	Internet Control Message Protocol
IP	Internet Protocol
IPsec	Internet Protocol Security
NAT	Network Address Translation
NetBIOS	Network Basic Input Output System
OSI Model	Open Systems Interconnection Reference Model
PHP	PHP Hypertext Preprocessor
PPP	Point to Point Protocol
RARP	Reverse Address Resolution Protocol
REST	Representational State Transfer
RFC	Request for Comments
RPC	Remote Procedure Call
RSA	Rivest Shamir Adleman Algorithm
S3	Simple Storage Service
scp	secure copy
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SSH	Secure Shell
SSL	Secure Socket Layer
TCP	Transfer Control Protocol

TLS ..... Transport Layer Security
UDP ..... User Datagram Protocol
VPN ..... Virtual Private Network
XML ..... E**x**tensible Markup Language

Index

DECLARATION

I hereby declare that I have written the Bachelor's Thesis on my own and have used no other than the stated sources and aids.

Bernd Hietler _____

Date _____