

Henri Kajova

# Datacenter Application Delivery Virtual Laboratory

Bachelor's Thesis  
Information Technology

November 2016



**KYAMK**  
University of Applied Sciences

<b>Tekijä/Tekijät</b>	<b>Tutkinto</b>	<b>Aika</b>
Henri Kajova	Insinööri (AMK)	Marraskuu 2016
<b>Opinnäytetyön nimi</b>		
Datacenter Application Delivery Virtual Laboratory		52 sivua 8 liitesivua
<b>Toimeksiantaja</b>		
Kymenlaakson ammattikorkeakoulu		
<b>Ohjaaja</b>		
Lehtori Vesa Kankare		
<b>Tiivistelmä</b>		
<p>Kyseisen opinnäytetyön tavoitteena oli luoda Virtual Laboratory -ympäristölle skenaario, jota voitaisiin käyttää application delivery controller -teknologioiden opetuksessa datakeskuskursseilla. Virtual Laboratory mahdollistaa tietoverkkoharjoitusten tekemisen virtuaalisesti ilman fyysisiä laitteita. Oli tärkeää tehdä harjoitus, jonka avulla käyttäjä tutustuisi skenaarioon ja ADC-laitteisiin.</p>		
<p>Työssä keskityttiin F5 BIG-IP Virtual Edition ADC-laitteeseen. Myös palvelimia ja muita verkkolaitteita jouduttiin asentamaan ja käyttöönottamaan, jotta toimiva verkko saataisiin rakennettua. Iso osa työtä oli ongelmien ratkaiseminen, koska kyseisiä laitteita ei ole aiemmin käytetty kyseisessä ympäristössä. Kaikkia ongelmia ei pystytty ratkaisemaan, ja joitain toimintoja ei voitu ottaa käyttöön, mutta toimiva skenaario saatiin valmiiksi.</p>		
<p>Harjoitus tehtiin niin, että sen avulla käyttäjä saa skenaarion otettua käyttöön. Verkon asetukset asetetaan ja ADC-laite asennetaan. Kyseiseen laitteeseen asennetaan virtuaalinen palvelin HTTP-palvelua varten, jonka avulla suoritetaan kuormanjakoa. Palvelua varten ADC-laitteeseen asennetaan kyberhyökkäyksiä heikentäviä ominaisuuksia.</p>		
<p>Työ oli onnistunut, sillä toimiva skenaario saatiin luotua, vaikka ongelmia sen toimintaan saamisessa oli. Joitakin ominaisuuksia jouduttiin jättämään pois, sillä kaikkia ongelmia ei pystytty ratkaisemaan. Todennäköisesti jotkin näistä ominaisuuksista voidaan saada toimintaan. ADC-laitteissa on vielä paljon ominaisuuksia testattavaksi, mutta skenaarion pitäisi toimia hyvänä alustana datakeskusopetukseen.</p>		
<b>Asiasanat</b>		
application delivery controller, virtualisointi, datakeskus, kuormanjako, kyberturvallisuus		

<b>Author (authors)</b>	<b>Degree</b>	<b>Time</b>
Henri Kajova	Bachelor of Engineering	November 2016
<b>Thesis Title</b>		
Datacenter Application Delivery Virtual Laboratory		52 pages 8 pages of appendices
<b>Commissioned by</b>		
Kymenlaakso University of Applied Sciences		
<b>Supervisor</b>		
Vesa Kankare, Senior Lecturer		
<b>Abstract</b>		
<p>The goal of this Bachelor's thesis was to provide a scenario for the Virtual Laboratory environment to implement application delivery controllers into a data center. Virtual Laboratory enables the possibility for training networking in a virtual environment. The scenario will be used as a tool for teaching on data center courses, which means that a basic case study needed to be provided. The case study introduces the user to the scenario, the basic setup and functions of an application delivery controller.</p> <p>The focus of the thesis was the F5 BIG-IP Virtual Edition application delivery controller. Implementation of a working network and creation of server and client machines were also done within the project. There was a lot of troubleshooting involved to get everything working, and not everything could be solved within this project. However, a working scenario was built with the most important features functional.</p> <p>A basic case study was created so that the user can get the scenario up and running. The case study begins with the configuration of the core network and then introduces the application delivery controller. A virtual server for an HTTP service is created where load balancing, SSL offloading and some denial of service attack mitigation are configured.</p> <p>The project was successful as a working scenario was created. Some functions had to be left out, because every problem could not be solved. There are more troubleshooting and advanced configuration options left to be tested. The scenario should serve as a great base to start working on data center courses.</p>		
<b>Keywords</b>		
application delivery controller, virtualization, data center, load balancing, cybersecurity		

## CONTENTS

1	INTRODUCTION .....	6
2	VIRTUAL LABORATORY .....	7
3	CYBER ATTACKS.....	11
4	APPLICATION DELIVERY CONTROLLERS .....	12
4.1	F5 Networks.....	14
4.1.1	F5 BIG-IP .....	14
4.1.2	F5 Virtual Editions & BIG-IQ.....	15
5	TINY CORE LINUX.....	16
6	IMPLEMENTATION.....	17
6.1	Licensing the software .....	18
6.2	Creating a working scenario .....	21
6.2.1	Networking .....	21
6.2.2	Hypervisors and servers.....	27
6.3	Creating a case study .....	30
6.3.1	Basic network configuration .....	31
6.3.2	ADC management and licensing.....	31
6.3.3	Basic ADC configuration .....	33
6.3.4	Load Balancing .....	36
6.3.5	SSL offloading.....	39
6.3.6	Mitigating slowloris attack.....	41
6.3.7	Mitigating SSL renegotiation flood attack .....	44
7	IDEAS FOR FURTHER PROJECTS .....	47
8	CONCLUSIONS .....	48
	REFERENCES .....	49
	APPENDICES	
	Appendix 1. Datacenter Application Delivery Case Study	

## ABBREVIATIONS

ADC	Application Delivery Controller
ADN	Application Delivery Network
AOM	Always On Management
CDN	Content Delivery Network
DNS	Domain Name System
DoS	Denial-of-Service
EUD	End User Diagnostics
HMS	Host Management Subsystem
HSRP	Hot Standby Routing Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
KVM	Kernel-based Virtual Machine
LACP	Link Aggregation Control Protocol
MAC	Media Access Control
MOS	Maintenance Operating System
NTP	Network Time Protocol
OSI	Open Systems Interconnections
RAM	Random-access memory
SQL	Structured Query Language
SSH	Secure Shell
SSL	Secure Sockets Layer
TMM	Traffic Management Microkernel
TMOS	Traffic Management Operating System
USB	Universal Serial Bus
VNC	Virtual Network Computing
VLAN	Virtual Local Area Network

## 1 INTRODUCTION

Datacenter hardware is expensive, and buying multiple sets of hardware for students to practice on would not be cost-effective. The hardware takes up a lot of space and electricity, and a powerful cooling system is necessary to keep the devices from overheating. Hardware also gets old and new better hardware needs to be bought to keep with the times, and to provide teaching with the latest technologies.

The goal of this project was to provide a scenario for a virtual environment called Virtual Laboratory. The scenario would be used to teach the latest application delivery controller technologies used in modern data centers. A basic case study, which would introduce the user to the scenario and devices in it, would also be necessary. By creating a basic case study, the scenario would also be tested to be working. This would allow to find any possible problems that would arise when a user starts working with it.

The implementation of Cyberlab, a cyber security laboratory, to the Kymenlaakso University of Applied Sciences motivated to start researching running data center appliances, one of them being application delivery controllers, in the Virtual Laboratory environment. The Virtual Laboratory was created as a bachelor's thesis work by Jaakko Nurmi for KyUAS in the spring of 2016. It provides an environment where the users can practice configuring networking devices and servers. (Nurmi 2016.)

The creation of Cyberlab began in the spring of 2014 and most of the groundwork was done during the following summer (Kettunen 2016). Cyberlab has generated a lot of thesis works for the students of KyUAS studying information technology. One of the thesis works related to application delivery controllers was the implementation of such devices in Cyberlab. The work was done by Antti Peltonen in the spring 2016. The students studying information technology in KyUAS were involved in the stress and attack testing of the devices. (Peltonen 2016.) This generated interest in looking into application delivery controllers and was a big motivation for this project.

## 2 VIRTUAL LABORATORY

The virtual laboratory is a virtual environment in which the user can train and experiment networking related case studies. It provides a set of networking devices for each student to configure themselves, instead of working in groups with a smaller set of devices. Because of the progression of virtualization over the years, it has become more cost efficient to use virtual devices to teach than to buy a set of real hardware.

Virtual laboratory uses a virtual machine called NestCore as its base, which runs a QEMU/KVM hypervisor. The machine has been designed to run on VMware hypervisors although it is possible to run it on other platforms that support the same format. The laboratory has a web interface where you can see the current topology and edit the device settings. In the interface, the user can access devices using Guacamole client-less remote desktop gateway, which supports VNC, SSH and telnet. Guacamole uses HTML5 to operate, that allows no need for any browser plugins. (Nurmi 2016.)

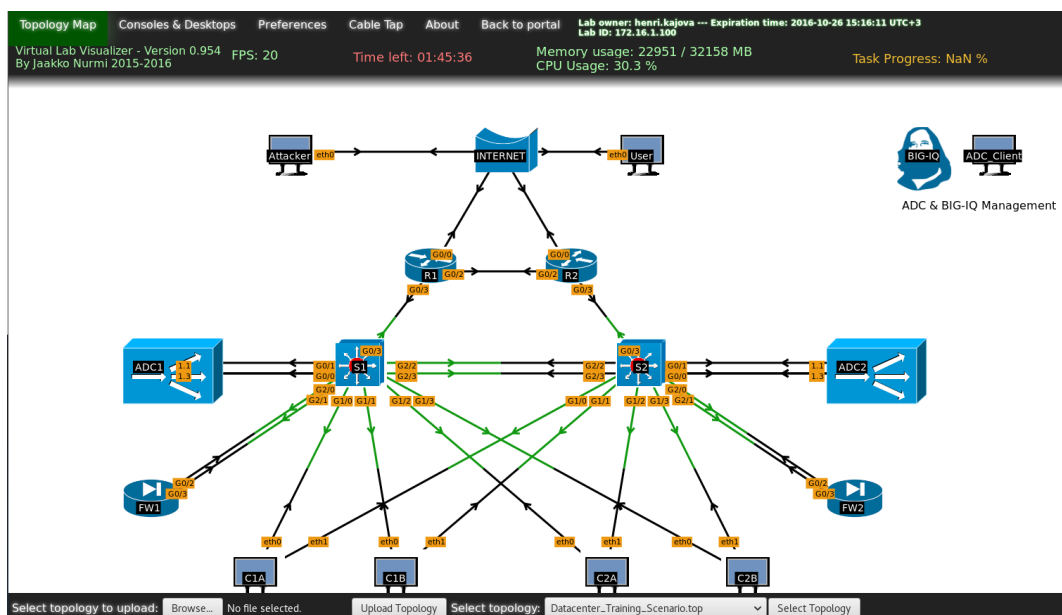


Figure 1. Virtual laboratory web interface showing the datacenter scenario topology

The web interface, shown in figure 1, is accessed using a web browser and navigating to the login page of the virtual laboratory system. User then logs in to the system using credentials assigned to them by the network administrators, where the system is run. The system uses RADIUS authentication, which makes it easy to manage users. Users with more permissions, who are usually the teachers and other administrators, have

administrative functions in their laboratory. For example, they can access other laboratories which they do not own, to assist and check the progress. The user reserves a laboratory to work on for a time they specify in the reservation process, and select the laboratory that they want to work with. The progress made in the laboratory can, however, be saved before the time runs out and it can be continued later. After loading the laboratory, a topology needs to be loaded from a list on the footer bar of the page, which can be seen at the bottom of figure 1. A custom made topology file can also be uploaded. The creation of topology files will be specified later. By right clicking the devices, it is possible to insert a CD or DVD from a dedicated laboratory filesystem server. It is also possible to mount a personal USB memory and some other USB devices.

Name	Base Image	Memory	Cpu	Interface count	Interface driver	MAC-address	base Access mode	GFX Driver	Snapshot mode	Device State
ADC1	REMOTE/BIGIP.qcow2	4096	2	8	e1000	00:79:C3:57	Xvnc	vmware-svga	0	Powered ON
ADC2	REMOTE/BIGIP.qcow2	4096	2	8	e1000	00:81:80:20	Xvnc	vmware-svga	0	Powered ON
Attacker	REMOTE/tinycore-dclab-attacker.vmdk	1024	1	2	e1000	00:58:88:00	Xvnc	vmware-svga	0	Powered ON
BIG-IQ	REMOTE/BIG-IQ-BASE.img	3072	1	8	e1000	00:09:EE:53	Xvnc	vmware-svga	0	Powered ON
C1A	REMOTE/DCLAB-C1A.img	512	1	2	e1000	00:E1:F4:6B	Xvnc	vmware-svga	0	Powered ON
C1B	REMOTE/DCLAB-C1B.img	512	1	2	e1000	00:C5:3F:C8	Xvnc	vmware-svga	0	Powered ON
C2A	REMOTE/DCLAB-C2A.img	512	1	2	e1000	00:31:5A:0D	Xvnc	vmware-svga	0	Powered ON
C2B	REMOTE/DCLAB-C2B.img	512	1	2	e1000	00:CC:45:C1	Xvnc	vmware-svga	0	Powered ON
Client	REMOTE/KaliLight_Tuned.vmdk	1024	1	2	e1000	00:9E:4B:A3	Xvnc	qxl-vga	0	Powered ON
FW1	REMOTE/asav.qcow2	2048	1	4	e1000	00:82:49:EC	telnet	vmware-svga	0	Powered ON
FW2	REMOTE/asav.qcow2	2048	1	4	e1000	00:59:66:BC	telnet	vmware-svga	0	Powered ON
Monitor	REMOTE/monitorv2i3.vmdk	1024	1	2	e1000	00:A2:F9:FC	Xvnc	vmware-svga	0	Powered ON
R1	REMOTE/iosv.qcow2	512	1	4	e1000	00:1D:65:56	telnet	vmware-svga	0	Powered ON
R2	REMOTE/iosv.qcow2	512	1	4	e1000	00:C0:E3:1C	telnet	vmware-svga	0	Powered ON
S1	REMOTE/iosl2v.qcow2	512	1	16	e1000	00:02:E4:1D	telnet	vmware-svga	0	Powered ON
S2	REMOTE/iosl2v.qcow2	512	1	16	e1000	00:87:14:87	telnet	vmware-svga	0	Powered ON

Figure 2. Virtual device manager under the preferences tab

There is a preferences tab, which can be seen in figure 2, where it is possible to manage the devices used in the laboratory. It is possible to add new devices or delete existing ones. The devices can be shut down, powered off or reset. Shut down is a clean way to power off the devices, which means that the software is shut down in a controlled manner. Powering off a device immediately removes the power and shuts the device down, which can lead to data loss. It should only be used when the device is frozen and cannot be shutdown in a clean fashion. Reset is used when the user wants to reset the device back to its original state. It creates the device again using the base image removing all existing changes. The base image can be specified from a list of devices, which are hosted on the laboratory filesystem server. The resources reserved for the devices can be edited, but they do not give the



overall laboratory more resources, which needs to be taken into consideration when adding more resources. A MAC address base, which is used to generate the addresses according to the interface number, can be specified. This helps in cases where the devices need to have a static MAC address. Network interface drivers and graphics drivers can be specified from a list of supported drivers. The access mode can be changed to specify how the device is accessed, with the choices of telnet, VNC or XVNC.

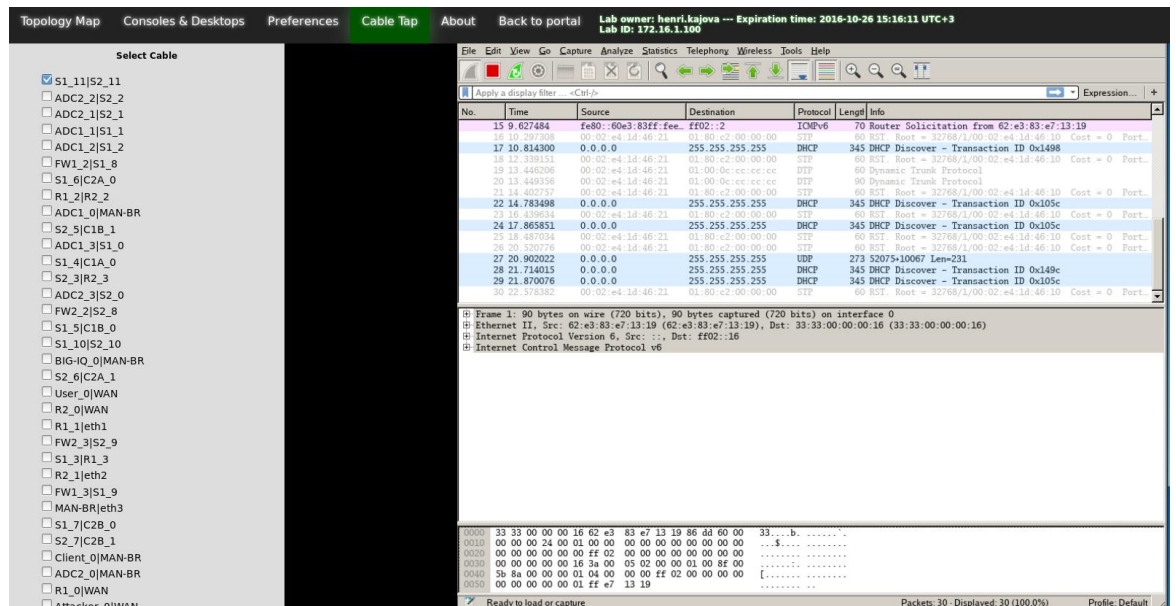


Figure 3. Cable tap tool in action performing packet capture

Cable tap is a tool that directs a copy of the traffic from a specific cable to an interface on a virtual machine running Wireshark packet analyzer. The traffic flow can be analyzed to troubleshoot and study the network performance. The cables are chosen from a list that automatically contains all the connected cables of the scenario, as seen on figure 3. Monitoring more than one cable at a time can cause loops.

The devices are installed using an installation file that is created on the laboratory filesystem server. The installation files use the following syntax:

TOPIC [Topic of the scenario]

DEVICE [Name] [REMOTE/filename] [Memory] [CPUs] [Number of interfaces] [Interface driver] [MAC-address base] [Access mode] [Graphics driver] [Snapshot mode]

TOPIC Datacenter Training Scenario

```
DEVICE ADC1 REMOTE/BIGIP.qcow2 4096 2 8 e1000
00:34:B2:23:12 Xvnc 5902 vmware-svga 0
```

The topologies are created using a topology file. The devices are positioned with the topology file using simple syntax:

```
[Type of the device] [Name of the device] [X axis] [Y axis]

ROUTER R1 600 270

ADC ADC1 200 400 hidden
```

Basically, the type field tells the laboratory to use a specific icon when rendering the topology. The name field is important, because it assigns the icon to a device with the same name. The X and Y axis fields are used to position the icon on the screen. The last example above shows that devices can be hidden by adding 'hidden' at the end of the line.

The cables between the devices are also configured and attached using the topology file. The syntax is similar with the positioning of the devices:

```
CABLE [Source device] [int #] [Destination device] [int #]

CABLE ADC1 1 S1 1
```

The source and destination device fields should contain the exact name of the device. The cable is attached between these devices by connecting it to available interfaces. The interfaces are inputted to the interface field using a number. The interfaces usually start from the number 0, however there may be some exceptions. An example of a functional topology can be seen on figure 1. The cables are rendered based on the connections, but are hidden if either of the devices is hidden.

Text can also be added to the topology file to instruct users. Text uses the following syntax:

```
TEXT [X axis] [Y axis] [Font size] [Background color] [Foreground
color] [Text]

TEXT 1300 200 16px #ffffff #000000 ADC & BIG-IQ Management
```

The font size is defined in pixels and the colors are defined using hexadecimals.

### 3 CYBER ATTACKS

Denial-of-service attacks are an effective way to render a web service unavailable. These attacks are usually done by flooding a service with legitimate traffic, but in such a volume that the system does not have enough resources to handle all the arriving data. This will leave the service unstable for normal users. Commands such as ping, which are available on most systems, can be used to attack against servers. When executed using a big group of clients, it can generate a lot of traffic to exhaust the bandwidth of the server. There are also attacks that exploit vulnerabilities in the systems such as configuration errors or other exploitable features or bugs. An example of an attack that is not trying to exhaust the bandwidth of the server, but is instead trying to exhaust the computational resources of the server, is SSL renegotiation attack. (Shakarian, Shakarian, Ruef 2013, 12-13.)

SSL renegotiation attack exploits vulnerability where by default the server allows the client to initiate an infinite number of renegotiations. The attacker initiates a SSL secure connection and renegotiates the handshake until it can exhaust the resources of the server. SSL handshake takes 10 times more resources from the server than from the client, which means a client with enough processing power could take a small, unprotected server down easily. More powerful servers need to be attacked using a botnet to exhaust the computational power of the server. Since only a tenth of the resources are needed in comparison with the server, the botnet does not necessarily need to be that big. (Holmes 2011.)

Slowloris attack is another application layer attack. It is a type of slow HTTP attack, that works by sending HTTP requests to the web server in a slow fashion. A HTTP header is send to the server just before timeout to keep the connection alive, but the request is never completed. This type of attack creates very little traffic and should not be detectable by looking at the bandwidth usage. In the end, the server cannot accept more connections and the service goes unavailable. (Holmes 2013.)

## 4 APPLICATION DELIVERY CONTROLLERS

Content delivery networks are used on the Internet to support content delivery in a scalable fashion. They are used to cache and optimize traffic flow using sophisticated technologies. The traffic is cached near the users to reduce traffic between links. This provides faster download speeds, lower latency and high availability. CDNs are often combined with application delivery controllers to form application delivery networks. ADCs are used to optimize application traffic in and out of data centers. This is done using application acceleration technologies and load balancing using layer 4-7 switching capabilities.

(Boucadair & Jacquenet 2015, 71.) OSI model layers 4-7 consist of transport, session, presentation and application layers, while 1-3 are physical, data link and network layers (Jönsson & Iveson 2014, 33). Unlike CDNs, which work using multiple sites, ADNs are usually operated in a single data center (Boucadair & Jacquenet 2015, 71).

The need for ADCs has arisen due to the rise of new type of networks. The need for dynamic workspaces and workers bringing their own devices to work, has brought the need for better optimization and security in networks. Workplaces need to ensure that the network has high availability. Networks need to function 24 hours a day, so that the workers can work at any time of the day. ADCs can help to offer this, by optimizing traffic, load balancing between servers and offering protection against cyber-attacks. (Citrix 2016.)

Load balancing is the key part of ADCs as they grew from load balancers to serve more advanced functions (Salchow 2012). When using load balancing, traffic is balanced between different servers to manage the load on the servers. If one of the servers goes down, the service will still be available, which offers for high availability. Load balancing can also be done to servers that are in a different physical location. (Citrix 2016.) This allows for protection against physical threats such as fires and natural disasters.

Round robin is a method of load balancing, that works with a simple technique. When a client contacts the load balancer, it directs the traffic to the first server on the list of servers. When another client contacts the load balancer, it is directed to the second server. When every server has been used, the load balancer goes back to the first server on the list. This makes round robin one of the easiest load balancing methods to implement, but it is not necessarily

the best. Unless application level health checking is implemented, the servers that are unavailable will still receive connection requests, which could affect the quality of service. (KEMP 2016a.)

Source IP hash load balancing, or other type of session persistence, needs to be implemented to ensure that clients stay connected to the server while their session is on. When using source IP hash load balancing, a unique hash is generated for every client. This is done using the source and destination addresses. The hash ties the client to a specific server until the session has ended, which ensures that if the client disconnects and needs to reconnect, the session will still be available to them. However, when a new session is initiated, new hash is generated. (KEMP 2016b.)

Another important part of ADCs is optimizing application performance. A great example is SQL database load balancing which works in a similar way to load balancing TCP traffic, but the device uses its knowledge to work with the application on the database level. It is also possible to offload tasks that are resource intensive for the servers. For example, SSL offloading does this by handling certificates, encrypting traffic leaving the servers, and decrypting traffic that is arriving at the servers. (Citrix 2016.)

Websites are often designed for connections offering high-speed and bandwidth, but this can deteriorate the user experience for mobile users (Citrix 2016). Mobile networks are not as resilient, which can make loading heavy pages slow for mobile clients. Application delivery controllers can convert images into formats that are more efficient, and they can optimize scripts and cascading style sheet files by removing unnecessary characters and white spaces. This helps to compress the data into a more efficient size. (Citrix 2016.) Compression techniques of this kind might add latency for users with a high-speed connection, because the compression takes time. This means that the need for compression should be considered carefully. (Salchow 2012.)

Protection against cyber threats is a key part of modern networks. Application delivery controllers can offer protection against threats coming in many different forms. They offer ways to protect against application layer attacks and can be used to mask sensitive data. The devices can even offer ways to identify new attacks. (F5 2016d.) As the devices are used as the entry points to the network, they can be used to authenticate and validate users, which protects

the network from intruders. It is also possible to weaken the effects of distributed denial of service attacks, since the devices can for example limit the inbound requests. (Citrix 2016.)

## 4.1 F5 Networks

F5 Networks, Inc. is the leading manufacturer of application delivery controller hardware and software, with 52 percent revenue share in the market in 2014 (PRNewswire 2014). They released their first web server load balancer in 1997 (Jönsson & Iveson 2014, 18). They aim at providing their customers software to enhance the delivery of application services in modern networks, but also the necessary hardware to process the tasks (F5 2016f).

### 4.1.1 F5 BIG-IP

F5 BIG-IP refers to a line of products that use F5's Traffic Management Operating System (F5 2016a). The products are available as either F5 branded hardware or virtual editions (F5 2016b). Traffic Management Operating System is not a standalone operating system, but instead a collection of software, which consists of all of the software used in BIG-IP platforms. This means that the name TMOS is used for the software suite instead of the platform. There is also a separate platform called BIG-IP Virtual Edition, that is run on a virtual hypervisor. (Jönsson & Iveson 2014, 23.) Some of the supported virtual hypervisors are VMware ESXi, Linux KVM and Citrix XenServer (F5 2016b).

The main elements of TMOS are Traffic Management Microkernel, Host Management Subsystem, Always On Management, Maintenance Operating System and End User Diagnostics. TMM is used to handle all the network activities and communication to the switch hardware. HMS is running a modified CentOS GNU/Linux system that provides tools and interfaces for system management. AOM is present on almost all BIG-IP hardware products. It allows for remote power management and access to the HMS through a serial console port or SSH through a network management port. MOS is run on RAM and used for disk and file system maintenance. It can be

accessed through the serial port by interrupting the normal boot sequence and entering the TMOs maintenance on GRUB. It can also be accessed with a bootable USB drive. EUD is capable of doing hardware tests, and can also be accessed through GRUB or USB drive. (Jönsson & Iveson 2014, 23-24.)

BIG-IP has a lot of different modules which can be activated to enable more functions. For example, there is Local Traffic Manager, which provides the basic functions to use load balancing, offloading and to achieve high availability. It also provides some protection against denial-of-service attacks. (F5 2016c.) Another example is the Application Security Manager, which provides even more in-depth protection against denial-of-service attacks and many other types of cyber threats. It protects against session hijacking, SQL injection and can even protect sensitive data by masking it. (F5 2016d.) There are many more modules and the modules have a great number of different functions. The choice of modules depends on what is needed, what the hardware can run, and what the license allows to be activated.

BIG-IP products have their own scripting language which can be used to create scripts called iRules. It comes with the local traffic management system and can be used to manage network traffic. It makes it possible to customize the way the traffic is handled, which makes it a powerful tool in traffic management. It uses the industry-standard Tools Command Language syntax as its base. (F5 2016e.)

The configuration and controlling of the devices can be done by using either the web interface or the traffic management shell. There is also a ZebOS Command Line Interface to configure more advanced routing options (IP Infusion Inc 2013).

#### 4.1.2 F5 Virtual Editions & BIG-IQ

In addition to selling hardware, F5 offers virtual editions on its BIG-IP and BIG-IQ systems. They are virtual versions of almost the same devices that you would get if you bought the hardware. The devices can be run locally on a hypervisor such as VMware ESXi or on a cloud service such as Amazon Web Services. (F5 2016i.)

BIG-IP Virtual Edition appliances are designed to support the growing cloud and software-defined market. The key selling points are that they are easy to deploy and migrate, which is great for cloud and software-defined environments, since they allow for adaptability and automation. This also means that they are easily deployed for existing servers. The virtual editions also offer more licensing choices, compared with the hardware, to suite the needs of a certain environment. The appliances are usually managed, deployed and licensed using BIG-IQ. (F5 2016i.)

BIG-IQ is a centralized management platform to manage all deployed BIG-IP devices. It is often used to license BIG-IP devices and can handle the licensing of up to 5000 devices. (F5 2016g.) It is capable of monitoring BIG-IP device resource usage, the status of high availability and SSL certificates. Deploying updates on all devices can be also done through BIG-IQ. It is possible to backup device configurations, which makes fallback easy in case of a misconfiguration. (F5 2016h.)

## 5 TINY CORE LINUX

Tiny Core is a Linux distribution aiming to be as small as possible. It is highly modular and has all the basic functions of a typical Linux distribution. However, it does not come with any unnecessary software and the users install what they need. (Tiny Core Team 2007.)

By default, Tiny Core is not designed to be installed onto a hard drive, instead it is loaded from a storage device to RAM. This makes the system run fast and it protects the files from changing, which ensures that the system is running properly every time you run it. Installing Tiny Core to a hard drive is still possible, but it is not an original goal of the project. Applications are also installed and loaded on RAM by default, and thus wiped after a reboot. (Kasanen 2013.)

In the default mode, cloud/internet, Tiny Core loads on RAM, and Apps tool is used to download the applications the user wants to use. This is done after every reboot since the applications are not stored in a hard drive. This makes the system and applications run fast, because everything is loaded on RAM. Mount mode is the recommended mode for using Tiny Core. In this mode,



persistent storage is used to store the applications the user wants to use. A supported partition is needed on a disk to use this mode. Supported partitions types are ext2, ext3, ext4 and vfat. The applications are stored in a directory called *tce* on the partition, and can be configured to be mounted when booting the system. There is also copy mode, where the applications are not mounted, but instead copied from the partition to RAM. This mode takes longer to boot, because of the applications need to be copied, which takes more time than mounting. However, this way the applications run faster. (Kasanen 2013.)

A persistent install is possible using the Tiny Core installation tool, which comes with the default release image. There is a graphical installer and a command-line installer available. After installing, the system applications can be installed using Apps tool, which has an option for persistent installation. If a graphical user interface is not used, Tiny Core Extension: Application Browser, can be used to browse and install applications. The tool is accessed with the command *tce-ab*. A command called *tce-load* can be also used to install applications. This is a non-interactive version of the application installer, and the user needs to know the name of the package they want to install. In order to keep persistent files in the system, they need to be listed in */opt/filetool.lst* file. By default, the file has the */home/tc* directory listed. There is also a file */opt.xfiletool.lst* to exclude files from the filetool utility. Backups are written in a file called *mydata.tgz*. The */home/tc/.xsession*, */home/tc/.profile* and */opt/bootlocal.sh* files can be used to store and execute configurations. Additional scripts can be created to the */opt/* directory. (Kasanen 2013.)

## 6 IMPLEMENTATION

The implementation covers the process of creating a working data center scenario for the virtual laboratory environment and creating a basic case study for that environment. The focus of the implementation was the application delivery controllers, but some server and client machines were created to serve the needs of the scenario. The configuration of the other network devices for the case study is also covered within the implementation.

## 6.1 Licensing the software

The work on the laboratory scenario began with activating the F5 BIG-IP Virtual Edition license. The Virtual Edition image was run as standalone on a VMware Workstation hypervisor. The reseller provided two licenses, but it was not known what they were for. At first, the licenses were inputted to the initial configuration wizard at the web interface, but there was an error message telling that the license was not a valid F5 license. After contacting the reseller, it became clear that one of the licenses was a pool license, which allowed for 25 instances of BIG-IP VE to be activated. The pool license needed to be activated on a BIG-IQ Virtual Edition, which would then provision the licenses to the BIG-IP devices. The other license provided would activate the BIG-IQ VE system.

Work continued after downloading the image for BIG-IQ VE. It was also run on a VMware Workstation hypervisor and was activated with the license provided. The licenses were accepted by the device and then it was the matter of activating the BIG-IP image. An activated BIG-IP base image would be ideal for the virtual laboratory, since then the users do not activate the images themselves. The activation with BIG-IQ would not work on the VMware Workstation, and it was thought that it might be an issue with the routing in the hypervisor. The BIG-IQ image was moved to the virtual laboratory, running on a VMware ESXi hypervisor, along with the BIG-IP image. The devices were connected to a bridge that allows all traffic to pass through, so there should be no problems with routing. However, there was a problem with the BIG-IQ license since it had gone inoperative.

Following logs were from the actual devices TMOS command line, when the problem was troubleshooted:

```
admin@bigiq ModuleNotLicensed:LICENSE INOPERATIVE  
/Common tmos# Apr 26 02:06:08 bigiq emerg mcpd[4976]:
```

```
01070608:0: License is not operational expired or digital signature  
does not match contents.
```

```
admin@bigiq ModuleNotLicensed:LICENSE INOPERATIVE  
/Common tmos# show /sys license
```

```
Can't load license, may not be operational
```

```
admin@bigiq ModuleNotLicensed:LICENSE INOPERATIVE  
/Common tmos# install /sys license registration-key *license om-  
mitted*
```

License server has returned an exception.

Fault code: 51092

Fault text: Error 51092, This license has already been activated on a different unit. Please contact technical support for assistance

A show command used to show the license activated on the device returned an error telling that the license cannot be loaded. To fix the problem, reactivating the license was tested with a command used to install licenses, but it returned an error that the license was already activated elsewhere, and that help from technical support would be needed to reactivate the license. The reseller was contacted about the license and they contacted F5 support to get the license deactivated. A new BIG-IQ VE image for KVM hypervisors was downloaded to compliment the virtual laboratory running KVM. After working with the reseller, the BIG-IQ VE was finally activated on the virtual laboratory, but the pool license needed to be reset, since it was activated on the old image. After deactivating the license and activating in on the new BIG-IQ VE, an activated BIG-IQ VE base image was ready to be used to activate the BIG-IP devices.

Apparently, the licenses are linked to a certain MAC address of the network adapter. When you move the image from one hypervisor to another, the MAC address changes, since there is only a virtual MAC address on a virtual network adapter. It had to be made sure that the MAC address would not change and this could be achieved by using the virtual laboratory device files. The device files tell the KVM hypervisor a specific command to run each device and these files can be used to specify the MAC address of the network interfaces. These files with the correct addresses were preserved, because at this point it was the only way to make sure that the addresses stay the same. When you made changes that needed to be applied on the virtual laboratory web interfaces device manager, it would always regenerate new MAC addresses. This meant that you would then need to overwrite or fix the device file for the BIG-IQ by hand. In a later version of the virtual laboratory, this issue was fixed and you can now specify the MAC address in the web interface.

The BIG-IP VE image still needed to be activated to make a working base image. The BIG-IP was added to the devices managed by the BIG-IQ, but when the activation the device was tested, it was not successful. After looking at the */var/log/restjavad.0.log* file, which contains many logs for troubleshooting, it was clear that something was wrong with the system. Both devices, BIG-IQ and BIG-IP, had a message saying that the license was an invalid F5 license. After providing logs and qkview files for the reseller that was in contact with F5 support, it was found out that this was due to a bug in the BIG-IP VE. The bug, numbered with the code 565137, affected mainly hosts running on a KVM hypervisor. When the file */var/log/lrm* is accessed, there will be an error “Dossier error 16”. This is due to the output of *dmidecode -s system-uuid* differing from the value provided in the */sys/class/dmi/id/product\_uuid*. In this case, the value was nonexistent in both cases. The bug was fixed in the version 12.1.0 HF1, but had yet not been released. The problem was escalated by the reseller to a F5 escalation engineer, whose job would be to provide a patch to fix the problem. The patch never arrived, which meant that the project was on hold until the official update was released.

After the new version was released, a whole new image of BIG-IP Virtual Edition, version 12.1.0.0.0.1434, was downloaded and installed on the virtual laboratory. The licensing was now successful on the first try. After two BIG-IP devices were activated, the BIG-IQ was removed from the laboratory to save resources. This turned out to be a problem since the licenses would fail after a while. It meant that the BIG-IP needs to contact the BIG-IQ after a while, so it was reintroduced to the laboratory.

## 6.2 Creating a working scenario

### 6.2.1 Networking

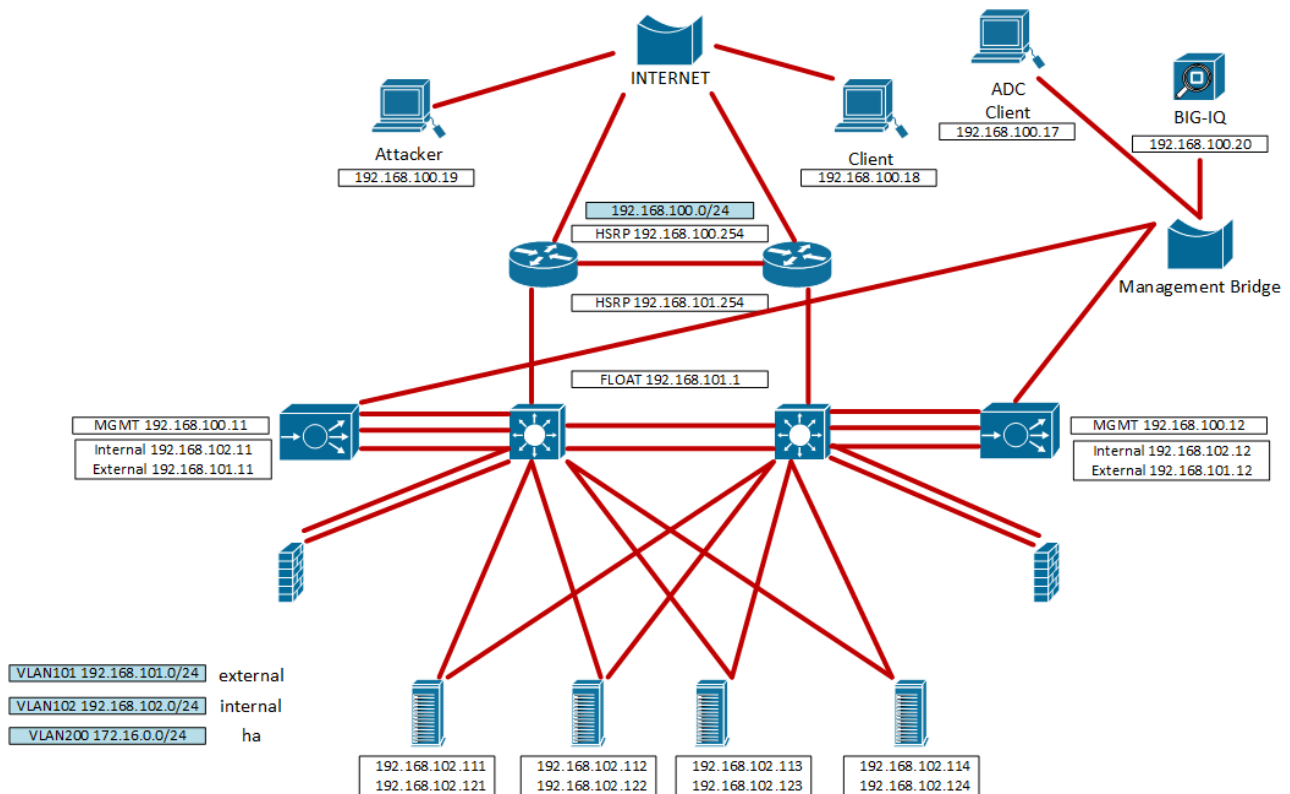


Figure 4. The full scenario topology and addressing

After licensing issues were sorted out, it was the matter of creating a working network. A virtual laboratory topology, illustrated on figure 4, was created during the licensing phase. The network contains two Cisco IOSv routers on the edge of the network which would connect the network to a wide area network. Two routers were used to ensure redundancy. A wide area network would be simulated using an ethernet bridge that would pass all the traffic through. The traffic would travel from the routers to one of the two switches also running Cisco IOSv software. The links between the routers and switches were configured as access ports using the external VLAN 101.

The switches were attached with each other through two interfaces with a port-channel, using the link aggregation control protocol. This would make the interfaces work as one logical interface, sharing the configurations made on the link. This would also allow load balancing between the interfaces. However, there were a couple of problems when using this technique between the link. First problem was that the protocol would sometimes fail during the negotiation phase and the link would never go up. Tweaking the delay, for the links,

on the virtual laboratory settings helped a bit, but it was not as consistent as you would hope it to be. Another problem arose when packet storms started to form between the link. It is not known why this started to happen, since there were no clear loops. The spanning tree protocol, which job is to prevent loops, seemed to be working as it should be. The MAC addresses of the interfaces were checked to ensure there were no duplicates, but none were found. Updating to a newer version of IOSv did not help either. After this it was decided that the port-channel should be abandoned and a single trunk link between the devices was to be used.

The ADCs were to be attached to the switches using link aggregation control protocol and trunking, to ensure high availability and load balancing. This was soon abandoned since the BIG-IP VE does not support link aggregation (F5 2016). The link used was configured as a trunk to allow several VLANs. Four servers were attached to the switches using access ports on the internal VLAN 102. The servers have two interfaces with one connected to each switch and each of the interfaces have their own IP address. This would allow better availability and load balancing. An HTTP virtual server with basic load balancing was configured on the ADC. The configuration of the ADC will be looked at in more detail at a later stage, but basically each of the server interface addresses are listed on the BIG-IP virtual server and it should perform load balancing between these interfaces. The server can be accessed by using a separate address assigned to the virtual server.

After the configuration, there was a problem because the ADC was not able to communicate with the servers in the internal VLAN. The ADC monitors the availability of the servers using configurable monitors. These monitors are chosen or created by the needs of the type of server used. They need to be accurate so that they confirm that the service on the server is working properly, if the availability cannot be confirmed, the service will be marked as unavailable. Using an ICMP monitor, the status was unavailable. Looking at the ARP table of the ADC, it seemed that the addresses were not resolving. By monitoring the traffic using the cable tap feature of the virtual laboratory, it became clear that the ARP request packets were leaving the ADC without dot1q tags. This meant that the packets would not be directed to the access ports using the internal VLAN. At first it seemed this might be a misconfiguration on the interface VLAN settings, but this was not the case. After trying

some different configurations on the interface, it was an idea that maybe changing the virtual network adapter would fix the problem. It seems that the E1000 network adapter that was used for the devices would not work with dot1q tagging. This was discovered to be true, after changing to VMXNET3 adapter. With the adapter, the ARP requests would leave as tagged and resolve without any problems. Now the ICMP monitor would show the server as available, however the HTTP monitor confirmed that the HTTP service was unavailable. Virtio network adapter was also tested on the ADC, but the interfaces would just flap up and down. BIG-IP Virtual Edition should support the virtio adapter on KVM hypervisor (F5 2016j).

The HTTP monitor sends a simple HTTP GET request, which the server should reply with an ACK packet. Cable tap was used to monitor the traffic and it was noticed that the ACK package was never received from the server. Tcpdump was used to monitor the traffic arriving at the network adapter of the server and the traffic arrived without problems. However, the adapter would not accept the packages. Further inspections revealed that the frames were considered too short. Analyzing the ethernet frame on Wireshark seemed that it was the right length and nothing out of ordinary could be seen. The server adapter was also changed to VMXNET3, but this did not change anything. Changing the adapter back to E1000 on both ends fixed the problem. After setting the interfaces to send the traffic as untagged, a handshake could be seen using cable tap, but this meant that VLAN tagging could not be used on the ADC. The VMXNET3 adapter would seem to be best supported by F5, as it is the one used on ESXi hypervisors, which makes the problems unusual.

Later the virtio adapter was further tested by including it in the installation file of BIG-IP and BIG-IQ. New scenario was installed and licensing the BIG-IP devices was tested. However, when trying to connect to the BIG-IQ web interface using the ADC client machine, the site was never reached. When analyzing the traffic using cable tap, the machines seemed to be talking with each other. A lot of TCP retransmissions were seen between the devices. When looking at the interface on the ADC client using ethtool, a climbing number of packages of wrong length could be seen. The problem seemed similar to the one with VMXNET3, but a clear reason for it could not be seen.

Since VLANs could no longer be used, the ADCs needed to be attached to the switches using three interfaces, one for each VLAN. Each of the interfaces was configured as access VLANs on the switch. The HTTP monitor status was still offline, so telnet was used to test the connection.

```
Trying 192.168.102.112...
Connected to 192.168.102.112.
Escape character is '^]'.

HTTP/1.0 400 Bad Request
Content-Type: text/html
Content-Length: 349
Connection: close
Date: Tue, 06 Sep 2016 17:26:59 GMT
Server: lighttpd/1.4.26

<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>400 - Bad Request</title>
  </head>
  <body>
    <h1>400 - Bad Request</h1>
  </body>
</html>
Connection closed by foreign host.
[root@ADC1:Active:Standalone] config # telnet 192.168.102.112 80
```

Figure 5. Attempting a telnet connection to the HTTP port of the server

On figure 5, telnet connection to port 80 of the server returned a message telling it was a bad request. The HTTP connection was now working, but the monitor was somewhat wrong. The server did not answer HTTP 1.0 GET requests, which meant that a custom monitor needed to be done, to support HTTP 1.1. The custom monitor had a couple of more parameters, as opposed to the default one.

Default monitor send string:

```
GET /
```

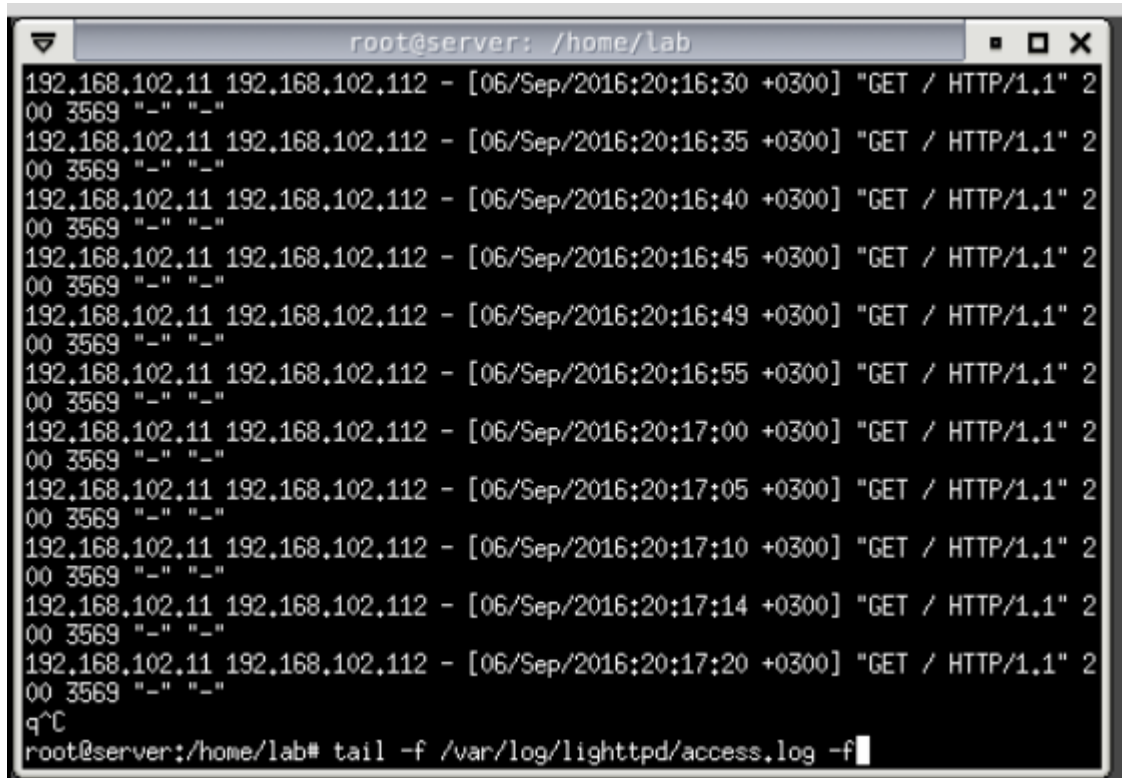
Custom HTTP 1.1 monitor string:

```
GET / HTTP/1.1\r\nHost: 192.168.102.112\r\n
```

The difference is that the custom send string tells the server that it is using HTTP 1.1 and it has a line telling the host it is meant for. A real-life scenario would use the domain of the server, rather than the IP address for the host.



After creating and taking the monitor in use, the HTTP server access log was checked.

A terminal window titled 'root@server: /home/lab' displays the output of the 'tail -f /var/log/lighttpd/access.log -f' command. The log shows a series of successful HTTP GET requests from the client IP 192.168.102.11 to the server IP 192.168.102.112. Each entry includes a timestamp, the request method and path, and a status code of 200. The requests occur at regular intervals from 16:30 to 17:20 on 06/Sep/2016. The terminal prompt is visible at the bottom, indicating the command is still running.

```
root@server: /home/lab
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:16:30 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:16:35 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:16:40 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:16:45 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:16:49 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:16:55 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:17:00 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:17:05 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:17:10 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:17:14 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
192.168.102.11 192.168.102.112 - [06/Sep/2016:20:17:20 +0300] "GET / HTTP/1.1" 200 3569 "-" "-"
q^C
root@server:/home/lab# tail -f /var/log/lighttpd/access.log -f
```

Figure 6. Successful HTTP GET requests seen on the access log file

On figure 6, successfully received GET requests can be seen. They arrive from the address of the ADC 192.168.102.11 to the server located in 192.168.102.112. The status message '200' implies that the request has been successful (Fielding 1999). The connection from a client to the servers was still not working. The traffic between the devices was monitored using cable tap.

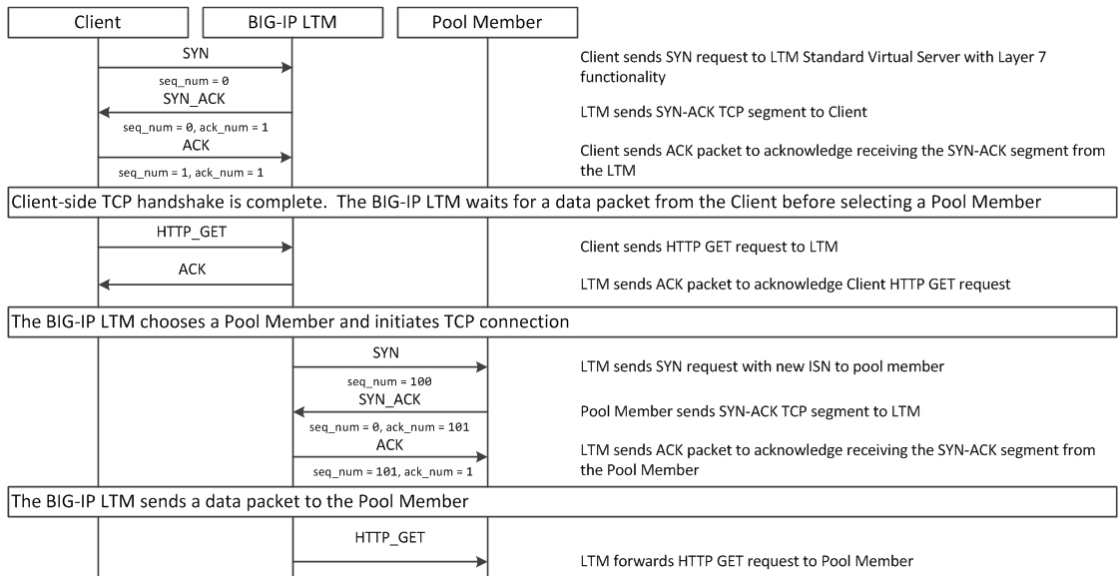


Figure 7. TCP handshake of a standard virtual server with layer 7 functionalities (F5 2007)

9732	2110.102940	192.168.100.16	192.168.101.111	TCP	74	58294 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSe
9733	2110.104217	192.168.101.111	192.168.100.16	TCP	74	http > 58294 [SYN, ACK] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 TSval=
9734	2110.106910	192.168.100.16	192.168.101.111	TCP	66	58294 > http [ACK] Seq=1 Ack=1 Win=29200 Len=0 TSval=108499216 TSe
9735	2110.107295	192.168.100.16	192.168.101.111	HTTP	368	GET / HTTP/1.1
9736	2110.108580	192.168.101.111	192.168.100.16	TCP	66	http > 58294 [ACK] Seq=1 Ack=303 Win=4682 Len=0 TSval=75751338 TSe
9744	2120.122109	192.168.100.16	192.168.101.111	TCP	66	[TCP Keep-Alive] 58294 > http [ACK] Seq=302 Ack=1 Win=29200 Len=0 TS
9745	2120.123392	192.168.101.111	192.168.100.16	TCP	66	[TCP Keep-Alive ACK] http > 58294 [ACK] Seq=1 Ack=303 Win=4682 Len=
9747	2122.122101	192.168.101.111	192.168.100.16	TCP	60	http > 58294 [RST, ACK] Seq=1 Ack=303 Win=0 Len=0

Figure 8. Captured TCP handshake.

In figure 8, the connection between the ADC and the client seemed to be working, as the 3-way TCP handshake was successful. HTTP GET request was received and acknowledged by the ADC. Figure 7 tells that after receiving the ACK package, the ADC should choose a pool member to establish a TCP connection with. However, the ADC would not try to establish a connection to a pool member and the connection was reset. After some investigation, this was due to secure network address translation, or SNAT, not being enabled on the virtual server. It had been turned off at an earlier stage when troubleshooting connection problems. Auto-map setting for SNAT was enabled and the connection worked without problems.

## 6.2.2 Hypervisors and servers

At the beginning of the project there was an idea of running a hypervisor inside the virtual laboratory. The servers would run inside the hypervisor thus simulating a real data centers which usually use hypervisors to run the servers. VMware ESXi would not boot up inside the virtual laboratory, or at least not by default. After this an open-source hypervisor, XenServer, was tested. The hypervisor booted up inside the virtual laboratory, and the management interface worked. However, the boot times for the machines inside the hypervisor were too long. A machine, which normally booted in a minute, took over an hour to boot up. Since the main focus of the project is application delivery controllers, the idea of running a hypervisor inside the virtual laboratory in the confines of this project was abandoned, since it would have most likely been a project of its own.

A CentOS 7 server was created to use as a base image for the servers. An Apache HTTP server was installed to serve a webpage, which was a default Wordpress page. MariaDB SQL database software and PHP were installed, because they were needed for the Wordpress to run correctly. After the server base image was deployed to the laboratory, the Wordpress site did not load correctly. Wordpress has some issues if the IP address of the server changes. Modifications to the Wordpress configuration need to be done to fix this, so Wordpress was removed. This would not serve the purpose of easily deployable web servers in an environment where the IP addresses might often change. A basic webpage with some text was created to serve something from the HTTP server. An Ubuntu server running Lighttpd HTTP server was also used when troubleshooting issues with F5 HTTP monitors. Neither of these servers worked with the default HTTP monitor of the BIG-IP, and needed to use a custom HTTP 1.1 monitor.

To save resources, it was decided that Tiny Core would be used to create some basic servers. When creating the image on VMWare Workstation hypervisor, it was important to use IDE as the hard disk type, since Tiny Core only supports IDE (Buys 2012). The installation was done using the official graphical installer, which is can be accessed from the desktop after booting up the graphical user interface. A disk named *sda* was chosen on the first menu and other settings were right by default. Ext4 was used as the file system type and

*nodhcp* boot option code was added to disable the use of DHCP. The server was installed as *Core Only*, which means installing the system without a graphical interface, since there was no need for a graphical interface.

After installation, before rebooting, the Apps tool was launched. An Apache HTTP server was installed by installing the `httpd` package. PHP and MariaDB were installed in case they would be needed in the future. A TCPdump tool, which can be used to analyze data arriving in a network interface, was installed for troubleshooting purposes. In the Apps tool, it was important to select the *On-Boot* option to ensure that the packages were saved on the disk and automatically loaded on the boot process. After installing the needed packages, the image was ready to be deployed into the virtual laboratory. The servers needed to be set up to serve a webpage using HTTP.

All the text file modifications had to be done using the `vi` editor, which usually comes with almost every GNU/Linux distribution. What is important to know about the editor is that the `i` key is used access the insert mode, which allows to insert and manipulate the text. After the changes are made, the `escape` key is pressed to leave the insert mode. To write the changes and quit, `:wq!` is typed. If the user wants to discard the changes they made, they can quit by typing `:q!`. To run some setup commands on boot up, `/opt/bootlocal.sh` script was modified. The following lines were added to the script file:

```
sudo ifconfig eth0 192.168.102.111 netmask 255.255.255.0
sudo ifconfig eth1 192.168.102.121 netmask 255.255.255.0
sudo apachectl start
```

The first two lines setup the IP addresses to two network interfaces and the third line starts the Apache HTTP server. The network interfaces on Tiny Core start with `eth0` and they increase in the increments of one. It was important to use the `sudo` command to run these commands as a super user, since otherwise the commands would not work.

A basic HTML site was created in the `/usr/local/apache2/htdocs/index.html` file, after removing the default `index.html` file. The following design was used for the site:

```
<h1>CYBER CORPORATION</h1>  
<br>  
Welcome to DCLAB server C1A!
```

Very little content was added, but the most important thing is that every server had a different line telling which server they were. This would help to check and illustrate working load balancing. In order to have the Apache configuration and HTML files stored persistently after a reboot, the `/opt/filetool.lst` file needed to be edited. Following lines were added to the end of the file, to ensure that the changes would be stored:

```
/usr/local/apache2/htdocs  
/usr/local/apache2/conf
```

First line would store the contents of the directory holding the site content files, and the second line would ensure that the Apache configuration files would be stored. However, the Apache configuration was not modified, but the action was to ensure that if some modifications would be done in the future, they would be stored persistently. The following actions had to be done on all four servers manually, but after doing so working servers were ready. It would not be sensible for the users to create these configurations themselves, which meant that the servers needed to be made into base images. Since the images are split into a base image and a linked clone, they were to be combined. This was done using a QEMU feature called convert. QEMU has a disk image utility that can be used to manage images, and it has a convert feature which can be used to combine the images into a new base image. The servers needed to be shut down before attempting the operation, since changes happening while they are running could corrupt the images. An example of the command that was used:

```
qemu-img convert -c -f qcow2 -O qcow2 C1A.img DCLAB-  
C1A.img
```

The `-c` option is used for the compression of the image, `-f` option is for the original format of the image, `-O` is the format of the output file, then the name of the file that is to be converted is inputted, and last the name of the output file.

After creating working server images, an attacker client image needed to be created. The attacker client would be used to attack the servers using tools to test for denial-of-service attack vulnerabilities. A new Tiny Core installation was created without a graphical interface. The attack tools needed to be compiled, since they were not available in the Tiny Core repository. This was done before rebooting the image after installation. Using Apps tool `wget`, `compicetc` and `openssl` packages were downloaded, and the first two were run only on RAM, since they would not be needed in the final image. `Openssl` package would, however, be needed for one of the attack tools. The `wget` tool is used to download the source code of the tools and `compiletc` has all the dependencies needed to compile the tools. The attack tools that were installed were `slowhttptest` and `thc-ssl-dos`. `Slowhttptest` can be used to test for slow HTTP attacks, while `thc-ssl-dos` is a proof of concept tool for SSL renegotiation flood attack.

The source for `slowhttptest` was downloaded using `wget` and extracted using `tar -xzvf <package filename>`. The directory was accessed and `./configure` command was used to configure before compiling. The command `make` was used to compile the source code into working binary. After this, the tool was installed using `sudo make install`. The `thc-ssl-dos` tool was installed in the same way, except `make all install` was used instead of `make` and `sudo make install`. It was then made sure that the files would be saved by `filetool`.

### 6.3 Creating a case study

The goal for the case study was to provide a practice that would get the user started working with the laboratory. It was important to introduce the user to the basic functions and tools of the laboratory and BIG-IP. This was done by creating a basic configuration practice, which would make the network functional, introduce the user to how the devices are managed and to where the configuration options are located. The goal was to configure the basic networking functions for the switches and routers, license the BIG-IP device using BIG-IQ, create a basic configuration for the BIG-IP, create load balancing between the servers and implement a couple of security features. The final case study can be found in appendix 1.

### 6.3.1 Basic network configuration

To enable basic networking, the routers and switches were configured. The routers were configured with Hot Standby Routing Protocol, which makes the gateways redundant by having the gateway of the other router in a standby, when it sees that the other gateway is down it switches to the active mode. Both router interfaces are configured with an address, but instead of using these addresses to access the gateway, a virtual gateway address is used between the routers to access the gateway. (Li, Cole, Morton & Li 1998.) Both, the inside and outside gateways, were configured redundantly using HSRP.

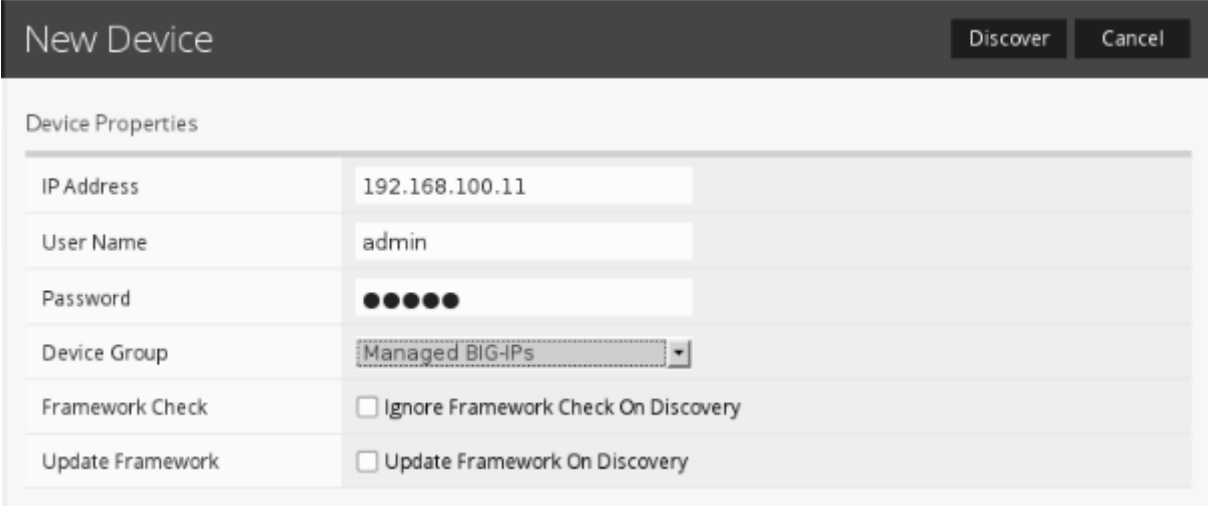
The switches needed to be configured with the appropriate VLANs, which were the external and internal VLANs. External VLAN, is the one that is next to the outside network, which means the interfaces facing the routers were configured as access ports using external VLAN. The interfaces facing servers were configured as access ports using internal VLAN, since they are in the internal network. Since the ADC device needs to have its own dedicated interfaces for each VLAN, because the dot1q tagging does not work with the E1000 network adapter, the interfaces facing ADC on the switch were configured as access ports using the appropriate VLANs. The link between switches was configured as a trunk port to support multiple VLANs.

### 6.3.2 ADC management and licensing

The ADC needs to be assigned a management address, before it can be licensed using BIG-IQ. This is done using the traffic management operating system shell on the command line, which can be accessed with the command *tms*. The user needs to login to the command line using *root* as username and *default* as password. These are the default values, which can be changed if needed in the initial setup of the device. The setup is done after licensing. First the management DHCP service is disabled using *modify sys db dhclient.mgmt value disable* command. Management address is added using *create /sys management-ip [ip address/prefixlen]* command. The changes are saved using *save /sys config partitions all* command. It is to be noted that when using Nordic keyboard, the forward slash needs to be inputted using the minus key next to the shift key. This is due to not being able to find a way to

change the keyboard layout to a Nordic layout. The regular methods of changing the keyboard layout on GNU/Linux systems seem to not work. The configuration can be checked by leaving the shell using *quit* command, and using *ifconfig eth0* command on the bash shell. The first interface is the management interface, which means it should always be the eth0 interface.

The BIG-IP devices need to be licensed before they can be used, which is done using a BIG-IQ device. Using the ADC client machine, the BIG-IQ web interface can be accessed. The licensing and addressing for the BIG-IQ was done in an earlier stage and they are automatically available on the base image. The BIG-IQ can be accessed from the IP address *192.168.100.20*, but if needed, the address can be changed in exactly the same way as in the BIG-IP devices. To log in to the BIG-IQ web interface, username *admin* and password *admin* are used.



New Device		Discover	Cancel
Device Properties			
IP Address	192.168.100.11		
User Name	admin		
Password	●●●●●		
Device Group	Managed BIG-IPs		
Framework Check	<input type="checkbox"/> Ignore Framework Check On Discovery		
Update Framework	<input type="checkbox"/> Update Framework On Discovery		

Figure 9. Adding a new device to BIG-IQ

The devices are licensed by navigating to *Device > Provisioning*. Next to *Devices* there will be a + button, which can be used to add a new BIG-IP device. Figure 9 shows an example of the filled *New device* form. The management address and credentials of the BIG-IP are filled. *Device Group* is chosen as *Managed BIG-IPs*. Everything else can be left to their default values. After discovering the device using the *Discover* button, the device is ready to be licensed. This is done by right clicking the device on the list, and choosing *License device*. Pool licensing and the appropriate pool is selected and the device is then licensed.



### 6.3.3 Basic ADC configuration

After the licensing is done, the BIG-IP device is ready to be configured. The web interface of the device is accessed using the ADC client machine by navigating to the management address of the device. However, unlike BIG-IQ, the BIG-IP does not automatically redirect the user to the SSL secured connection. Since there is no unsecured connection available, the user needs to add *https://* in front of the address. For example, if the management address of the device is *192.168.100.11*, the user needs to connect the address *https://192.168.100.11* to reach the web interface. The default credentials for the device are username *admin* and password *admin*. On log in, the user is first prompted to a setup utility to configure all the basic settings for the device.

The screenshot shows the 'Setup Utility >> Device Certificates' window. It features a table for 'SSL Certificate/Key Source' with the following fields:

Import Type	Certificate and Key
Certificate Name	server
Certificate Source	<input checked="" type="radio"/> Upload File <input type="radio"/> Paste Text <input type="button" value="Browse..."/> laboratory.cert.cert
Key Source	<input checked="" type="radio"/> Upload File <input type="radio"/> Paste Text <input type="button" value="Browse..."/> laboratory.cert.key
Free Space on Disk	229 MB

At the bottom of the window are 'Cancel' and 'Import' buttons.

Figure 10. Importing device certificates

Navigating through the setup, in *Resource Provisioning* more modules can be activated, but the default modules are enough for the case study. In *Device Certificates*, new device certificates should be imported. A self-signed certificate was created earlier on the ADC client machine and can be found on the home directory of the Root user. Figure 10 shows that *Import Type* of *Certificate and Key* should be chosen, and the correct files are selected using the *Browse* button. The certificate source file ends with *cert* and the source key ends with *key*. After importing, the certificate user is logged out of the device, because the certificate for the connection has changed. Since it is a self-signed certificate, the browser tells that the connection is untrusted. This does not matter in a laboratory environment, but in real world applications a signed

certificate should be acquired. After logging back in, the applied certificate should be seen, showing that it was successfully imported.

General Properties	
Management Port Configuration	<input type="radio"/> Automatic (DHCP) <input checked="" type="radio"/> Manual
Management Port	IP Address/prefix: <input type="text" value="192.168.100.11"/> Network Mask: <input type="text" value="255.255.255.0"/> <input type="text" value="255.255.255.0"/> <input type="button" value="v"/> Management Route: <input type="text"/>
Host Name	<input type="text" value="ADC1.laboratory"/>
Host IP Address	<input type="text" value="Use Management Port IP Address"/> <input type="button" value="v"/>
Time Zone	<input type="text" value="UTC"/> <input type="button" value="v"/>
User Administration	
Root Account	<input type="checkbox"/> Disable login Password: <input type="password" value="••••••"/> Confirm: <input type="password" value="••••••"/>
Admin Account	Password: <input type="password" value="•••••"/> Confirm: <input type="password" value="•••••"/>
SSH Access	<input checked="" type="checkbox"/> Enabled
SSH IP Allow	<input type="text" value="* All Addresses"/> <input type="button" value="v"/>
<input type="button" value="Back"/> <input type="button" value="Next..."/>	

Figure 11. Filled platform settings

Next platform settings are configured. As seen on figure 11, the management port settings should be correct, since they were configured using the traffic management shell. A management route could be added if needed, but since the traffic is directed through a bridge it is not necessary. The host name of the device needs to be a fully qualified domain name, which means that for example *adc1.dclab.fi* could be used. It is also desirable to change the time zone to *UTC*. The credentials can be changed or the default values can be inputted to keep using them. In real-world applications, the credentials should always be changed, but since the laboratory is not connected to any public network, it does not matter.

Setup Utility » VLANs	
<b>External Network Configuration</b>	
External VLAN	<input checked="" type="radio"/> Create VLAN external <input type="radio"/> Select existing VLAN
Self IP	Address: 192.168.101.11 Netmask: 255.255.255.0 Port Lockdown: Allow Default
Default Gateway	192.168.101.254
Floating IP	Address: 192.168.101.1 Port Lockdown: Allow Default
<b>External VLAN Configuration</b>	
VLAN Name	external
VLAN Tag ID	101
Interfaces	VLAN Interfaces: 1.2 Tagging: Untagged Add 1.1 (untagged) Edit Delete

Figure 12. Example configuration for external VLAN

Network settings are configured by creating internal, external and high availability VLANs. The addressing is done using the designed addressing plan. The floating IP address is the default gateway that the devices in the network use, which means the traffic is directed to the ADC. As seen on figure 12, the external VLAN needs its own default gateway, which is the HSRP floating IP address of the routers on the same network. The VLAN ID tag is the same tag that is used in the switches. However, the traffic is not tagged since VLAN tagging does not work. The VLAN interface wanted is chosen and tagging is set *untagged*. Only one VLAN per interface can be used. The interface should be connected to the switch port which has been configured as an access port for the same VLAN. NTP or DNS are not configured, but ConfigSync, Failover and Mirroring are set to use high availability VLAN. However, failover or configuration synchronization are not configured in this case study, because they do not seem to work reliably. Using configuration synchronization might have been a cause of some boot loops and other problems on the ADCs. Sometimes the synchronization works for a while, but after some time the link starts

working unreliably. The ADCs start telling that the high availability is offline, and because of this they do not synchronize. The connection might come back up, but this has caused some boot loops after a while. This is most likely due to corrupted configurations. Because high availability cannot be used, it means that after the settings are ready and the *Finished* button can be pressed to finish the setup.

### 6.3.4 Load Balancing

To start load balancing between the HTTP servers, a server pool and a virtual server need to be created. The load balancing happens between the configured pool members. By navigating to *Local Traffic > Pools > Pool List*, it is possible to create a new pool.

Configuration: Basic

Name: DCLAB-HTTP-Servers

Description:

Health Monitors:

Active: Common gateway\_icmp, http

Available: Common http\_head\_f5, https, https\_443, https\_head\_f5

Resources:

Load Balancing Method: Round Robin

Priority Group Activation: Disabled

New Members:

New Node  New FQDN Node

Node Name: C2B-S2 (Optional)

Address: 192.168.102.124

Service Port: 80 HTTP

Add

R:1 P:0 C:0 C1A-S1 192.168.102.111 :80  
R:1 P:0 C:0 C1A-S2 192.168.102.121 :80  
R:1 P:0 C:0 C1B-S1 192.168.102.112 :80  
R:1 P:0 C:0 C1B-S2 192.168.102.122 :80  
R:1 P:0 C:0 C2A-S1 192.168.102.113 :80

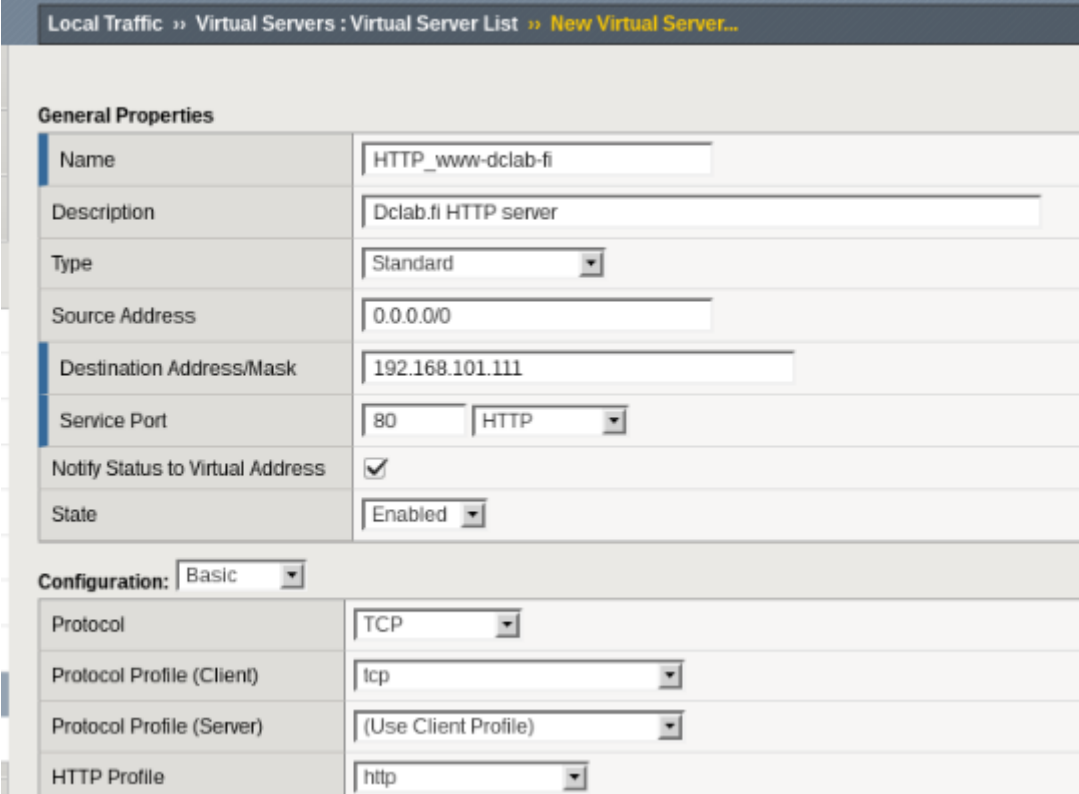
Edit Delete

Cancel Repeat Finished

Figure 13. Example server pool configuration

Figure 13 shows an example where all server interfaces have been added to the pool members. The information for each specific interface is filled and the *Add* button is used to add them to the pool. The load balancing method used

is left as *Round Robin*. The pool is named and health monitors are added. Since the servers are HTTP servers, the *http* monitor is used to monitor of availability of the HTTP service. Additionally, *gateway\_icmp* monitor is used to monitor the availability of the interface. After finishing the settings, the pool members should all be available, which is indicated by a green dot next to the member in the members list.



General Properties	
Name	HTTP_www-dclab-fi
Description	Dclab.fi HTTP server
Type	Standard
Source Address	0.0.0.0/0
Destination Address/Mask	192.168.101.111
Service Port	80 HTTP
Notify Status to Virtual Address	<input checked="" type="checkbox"/>
State	Enabled

Configuration: Basic	
Protocol	TCP
Protocol Profile (Client)	tcp
Protocol Profile (Server)	(Use Client Profile)
HTTP Profile	http

Figure 14. Part of the virtual server settings

After the pool is created it needs to be associated with a virtual server. By navigating to *Local Traffic > Virtual Servers > Virtual Server List*, it is possible to create new virtual servers. Figure 14 shows a part of the settings needed for the virtual server. The server is named and standard type of server is used. The source address is *0.0.0.0/0*, which means that the inbound traffic can originate anywhere. The destination address is the address that is used to access the virtual server. This is the address the client connects to. Default HTTP service port is used, protocol used is TCP and HTTP profile is set as *http*. It is important to set the source address translation to *Auto-map* to ensure that address translation works correctly. The server pool created earlier is selected as the default pool. Now the server should be accessible by the client machine using the virtual server IP address.

Next a pool for gateway load balancing is created. The router interfaces on the external VLAN are added to a pool and used as a gateway pool. First HSRP must be removed from the interfaces, which happens with the *no standby <process number>* command on the router interface. As before, a new pool was created with the gateway addresses as members. This time only *gateway\_icmp* monitor is used to monitor the gateway status.

The screenshot shows the 'Properties' window for the 'external\_default\_gateway' route. The 'Resource' dropdown is set to 'Use Pool...' and the 'Pool' dropdown is set to 'Gateways'. The 'Destination' and 'Netmask' are both set to '0.0.0.0'. The 'MTU' is set to '0'. There are 'Update' and 'Delete' buttons at the bottom.

Properties	
Name	external_default_gateway
Partition / Path	Common
Description	<input type="text"/>
Destination	0.0.0.0
Netmask	0.0.0.0
Resource	Use Pool... ▾
Pool	Gateways ▾
MTU	0

Figure 15. Using pool as an external default gateway

By navigating to *Network > Routes*, the user can manage the *external\_default\_gateway*. Figure 15 shows that the resource has been changed to *Use pool*, and that the pool created for gateways has been selected as the pool. By updating the settings, the load balancing between pool members is activated. It is to be noted that the load balancing only happens between the gateways when leaving the ADC. Traffic coming from the outside network comes through the active HSRP interface and through the same device. It would most likely make more sense to use HSRP on both gateways instead of mixing up technologies, but this is something that was tested and included in the case study, since there are cases where using load balancing between gateways makes sense.

### 6.3.5 SSL offloading

To implement SSL offloading, the certificates need to be imported into the system. This means navigating to *System > File Management > SSL Certificate List* to import the needed files. The certificate and key need to be imported separately.

SSL Certificate/Key Source	
Import Type	Certificate
Certificate Name	<input checked="" type="radio"/> Create New <input type="radio"/> Overwrite Existing dclab-cert
Certificate Source	<input checked="" type="radio"/> Upload File <input type="radio"/> Paste Text <input type="button" value="Browse..."/> DCLAB.cert.cert
Free Space on Disk	228 MB

Figure 16. Importing the certificate file

As seen on figure 16, the import type *Certificate* is selected, the certificate is named and the certificate file is selected using the *Browse* button. In this case study, the same self-signed certificate that is used for the device certificate, is used also for the servers. The certificate is located on the home directory of the root user of the ADC client machine. The key is also imported using the same method, but import type *Key* is selected.

Add SSL Certificate to Key Chain	
Certificate	dclab-cert
Key	dclab-key
Chain	dclab-cert
Passphrase	<input type="text"/>
OCSP Stapling	None

Figure 17. Adding SSL certificate to the key chain of a client SSL profile

A new client SSL profile needs to be created, to use the certificate with a virtual server. The profile can be created by navigating to *Local Traffic > Profiles > SSL > Client*. First the profile is named and *clientssl* is used as the parent profile. The certificate needs to be added to a certificate key chain, which is done as seen in figure 17. By clicking the *Add* button the user is prompted with a window where the right files are selected. The Certificate, key and chain are selected and they are added to the key chain. After saving the changes, the profile is ready to be used.

General Properties					
Name	HTTPS_www-dclab-fi				
Partition / Path	Common				
Description	Dclab.fi HTTPS server				
Type	Standard				
Source Address	0.0.0.0/0				
Destination Address/Mask	192.168.101.111				
Service Port	443 HTTPS				
Notify Status to Virtual Address	<input checked="" type="checkbox"/>				
Availability	<span style="color: green;">●</span> Available (Enabled) - The virtual server is available				
Synccookie Status	Off				
State	Enabled				
Configuration: Basic					
Protocol	TCP				
Protocol Profile (Client)	tcp				
Protocol Profile (Server)	(Use Client Profile)				
HTTP Profile	http				
FTP Profile	None				
RTSP Profile	None				
SSH Proxy Profile	None				
SSL Profile (Client)	<table border="1"> <thead> <tr> <th>Selected</th> <th>Available</th> </tr> </thead> <tbody> <tr> <td>Common dclab-ssl</td> <td>Common clientssl clientssl-insecure-compatible clientssl-secure crypto-server-default-clientssl</td> </tr> </tbody> </table>	Selected	Available	Common dclab-ssl	Common clientssl clientssl-insecure-compatible clientssl-secure crypto-server-default-clientssl
Selected	Available				
Common dclab-ssl	Common clientssl clientssl-insecure-compatible clientssl-secure crypto-server-default-clientssl				

Figure 19. Creating an HTTPS virtual server

A new virtual server is created to SSL. Like shown on figure 19, this time the service port *443* for *HTTPS* is selected. The SSL client profile is selected to use SSL offloading using the certificate. All the other settings are the same as in the HTTP server and the same address can be used. After saving the settings, SSL offloading is functional. This can be confirmed by navigating to the



address of the server using the HTTPS protocol, for example  
*https://192.168.101.111.*

Redirecting the HTTP traffic to HTTPS is done by creating a policy. The policy can be created by navigating to *Local Traffic > Policies > Policy List* and using the *Create* button. The policy is named to describe the contents of the policy, for example *HTTPtoHTTPS*. The strategy setting is left at *Execute first matching rule*. After the policy is created a rule is added by using the *Create* button next to *Rules*. It is also named to describe the contents of the rule, which can in this case be the same name as the policy itself. The match conditions are left as *All traffic* to match all traffic. By using the + next to *Do the following when the traffic is matched*, a task is created to execute when the traffic matches the conditions. A redirect task is added from the drop-down menu and the following line is added as the location:

```
tcl:https://[getfield [HTTP::host] ":" 1][HTTP::uri]
```

The changes are then saved, and using the policy is saved and published the drop-down menu next to the *Save Draft* button in the policy overview. The policy is then activated on the HTTP virtual server to enable redirection. By navigating to the virtual server settings and accessing the resources tab, a policy list can be found. By using the *Manage* button, a list of available policies can be seen and the desired policy is moved to the enabled policies. After finishing the settings, the policy is active and should work right away. If the redirection fails, the most likely cause is a typing error in the location line and should be double checked. It was also noticed that it can sometimes take a moment to get active.

### 6.3.6 Mitigating slowloris attack

A slowloris denial of service attack was tested against the virtual server of the ADC. The attack was conducted using a tool called slowhttptest, which can be used to test slowloris attack and other derivatives. The test proved that the virtual server was easy to knock down using one client slowloris attack. Next it was the case of mitigating the attack. Mitigating the attack would be done using iRule provided by F5. New iRule can be created by navigating to *Local Traffic > iRules > iRule List*. The iRule is named and the rule itself is provided

in the *Definition* section. The following iRule is used to mitigate a slowloris attack:

```

when CLIENT_ACCEPTED {
    set rtimer 0
    after 1000 {
        if { not $rtimer} {
            drop
        }
    }
}

when HTTP_REQUEST {
    set rtimer 1
}

```

(F5 2009)

The way the iRule works is that, when an HTTP request is not completed in one second, the connection is dropped. This could also affect users with a slow connection if the request cannot be completed within a second. iRules are activated on a virtual server basis by navigating to virtual server settings and *Resources* tab. There is a list of iRules that can be managed. The iRule is then moved from the list of available iRules to the enabled iRules. After finishing the configuration, the iRule is active. Now the slowloris attack should be unsuccessful. However, when trying out the attack again it was still successful. Even though slowloris attack does not generate a lot of traffic, because of the limitations of running virtual switches and routers, it seemed that lines were getting excessive amounts of traffic. The attack needed to be toned down, so that the network before the virtual server could cope with the traffic, but it would still knockdown the virtual server. The following command with the settings toned down was found to be good for the purpose, and would be used in the case study to demonstrate a slowloris attack:

```
slowhttpstest -c 10000 -H -i 5 -r 50 -t GET -u https://www.dclab.fi
```

To break down the command used, the `-c` value is the number of connections, `-H` tells the program to run in slowloris mode, `-i` value is the interval until the

program follows up with the follow up data, *-r* is the number on connections per second, *-t* is the used verb and *-u* is the url of the site that is tested. The program can be interrupted with the regular keyboard combination used to cancel running commands *CTRL + C*.

```
slow HTTP test status on 50th second:
initializing: 0
pending: 755
connected: 1253
error: 0
closed: 62
service available: NO
```

Figure 20. Slowhttptest running slowloris attack without mitigation

```
slow HTTP test status on 50th second:
initializing: 0
pending: 5
connected: 234
error: 0
closed: 1869
service available: YES
```

Figure 21. Slowhttptest running slowloris attack with mitigation

The command was tested with the iRule disabled and it knocked down the virtual server, but when the iRule was again enabled the server stayed available. This was confirmed from the status monitor of the slowhttptest program and can be seen in figures 20 and 21. The service was also confirmed to be unavailable using the browser of the client machine. When the attack began, the site became quickly unavailable for the client, but when the iRule was enabled the site remained available for the client. In figure 21, A fast rising number of closed connections can be seen on the slowhttptest statistics, which indicates that the iRule works correctly. When using an HTTP profile on the server a real BIG-IP device should be able to block a single client slowloris attack, but in the virtual laboratory environment it could not be confirmed. The HTTP profile resets the connection when the maximum header size is exceeded by the attack. (F5, 2009.) In a real-life situation where an HTTP profile is not used, the HTTP servers should be correctly configured to render slow HTTP attacks

useless, but the iRule could be used to mitigate the attack if needed. For example, if the attack is distributed or when the servers are not in the care of the network administrator.

### 6.3.7 Mitigating SSL renegotiation flood attack

Vulnerability to SSL renegotiation flood attack was also tested against the virtual server. The principle is that if the server allows the client to renegotiate the SSL handshake indefinitely, the server is vulnerable to the attack. The following command was executed from the user client machine to test for the vulnerability:

```
openssl s_client -connect www.dclab.fi:443
```

An HTTP GET request was executed by typing *GET / HTTP/1.0* and pressing enter. After this, the connection was renegotiated five times by typing *R* and pressing *enter*. The server did not reset the connection, but instead the program lets the user to renegotiate five times before the session needs to be restarted. However, it was possible to do this indefinitely, which meant that the server was vulnerable. With an automated tool that repeats the renegotiation process, the attack would be successful. It needs to be noted that the slowloris iRule needed to be disabled to do this test, because the connection would otherwise reset before the commands could be typed. The iRule would not, however, protect from this attack, because the process would be faster with an automated attack tool. Like the slowloris attack, the attack could be mitigated using iRule endorsed by F5. The following iRule was created and activated the same way as before, but only on the HTTPS virtual server:

```
when RULE_INIT {  
    set static::maxquery 5  
    set static::mseconds 60000  
}  
when CLIENT_ACCEPTED {  
    set ssl_hs_reqs 0  
}  
when CLIENTSSL_HANDSHAKE {  
    incr ssl_hs_reqs
```

```

    after $static::mseconds { if { $ssl_hs_reqs > 0 } {incr
ssl_hs_reqs -1} }
    if { $ssl_hs_reqs > $static::maxquery } {
        after 5000
        log "Handshake attack detected, dropping [IP::client_addr]:[TCP::client_port]"
        drop
    }
}
}

```

(Holmes 2011b)

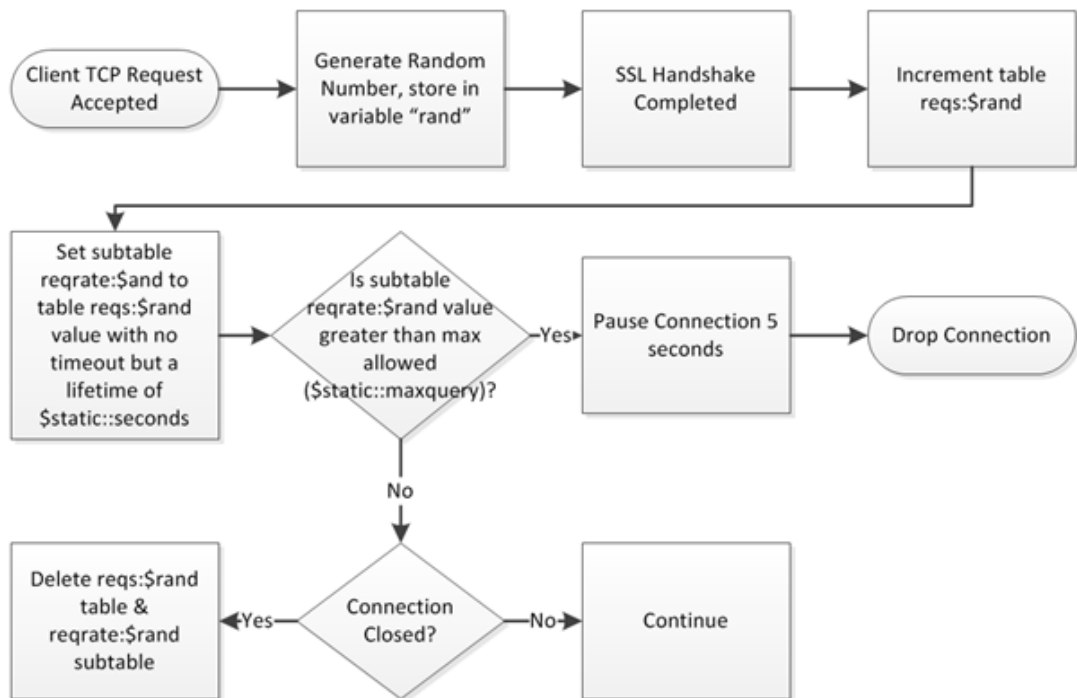


Figure 22. SSL renegotiation flood attack countermeasure iRule workflow (Holmes 2011a)

After implementing the iRule, the renegotiation could be performed five times and the connection was stalled and dropped after a while. Another connection was not possible within a minute after the attack was detected. The iRule sends a notice to the BIG-IP logs that an attack was detected and dropped. In figure 22, the workflow of the iRule can be seen. After a successful handshake, the number of requests is counted and when the maximum number is reached, the connection pauses for 5 seconds before dropping it. If the limit is not reached, the connection will continue to work normally. (Holmes 2011a.)

Wed Oct 26 09:54:14 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44520
Wed Oct 26 09:54:14 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44512
Wed Oct 26 09:54:13 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44534
Wed Oct 26 09:54:13 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44554
Wed Oct 26 09:54:13 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44530
Wed Oct 26 09:54:11 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44536
Wed Oct 26 09:54:11 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44556
Wed Oct 26 09:54:10 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44510
Wed Oct 26 09:54:10 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44548
Wed Oct 26 09:54:09 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44502
Wed Oct 26 09:54:09 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44500
Wed Oct 26 09:54:07 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44504
Wed Oct 26 09:54:06 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44494
Wed Oct 26 09:54:06 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44524
Wed Oct 26 09:54:06 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44496
Wed Oct 26 09:54:06 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44498
Wed Oct 26 09:54:05 UTC 2016	info	ADC1	tmm[15746]	01220002	Rule /Common/ssl-renegotiation-attack-block <CLIENTSSL_HANDSHAKE>: Handshake attack detected. dropping 192.168.100.40:44506

Figure 23. BIG-IP logs after using an attack tool for the iRule testing

A proof of concept SSL renegotiation flood attack tool called `thc-ssl-dos` was also tested against the unprotected virtual server. The tool was created by a French hacking group The Hacker's Choice. It can be used to perform an attack against any server that allows SSL renegotiation. (Holmes 2011a.)

When the attack was launched against the virtual server, the service became not as responsive, but it never went unavailable. There were many handshakes happening and the tool was working, but it seemed that the switches could not keep up with the amount of packets being transmitted by the tool. A growing number of input errors could be seen on the switch interface as runts. Number of connections was limited from the default 400, but the input errors were still happening. It did not seem that the renegotiation itself was the cause for the slow responses, but the connection to the virtual server instead. Hardware BIG-IP should be able to withstand an attack from a single client, because of its crypto-processors (Holmes 2011a). It was not known if there is some software that would also do the trick of blocking an attack from one client, or it is just that the attack was weakened by the connection between the attacker and the ADC. The server was protected using the iRule and the logs were showing blocked attacks. As seen on figure 23, the connections seemed to be dropping as they should be, but the page

was still not as responsive, which would indicate that the cause is the connection instead of the renegotiation.

## 7 IDEAS FOR FURTHER PROJECTS

There are a great number of things that can be done to develop this scenario further, since not all features work yet. The troubleshooting of the VLAN and synchronization issues of the ADCs, and looking in to the switch LACP issues, could be done as a project. The virtio network adapter should be functional, but the reason why it does not work is not known. It might be because all the links are not using the same adapter. It was not tested if the IOSv switches and routers support virtio adapter. One idea would be to replace the IOSv switches with OpenSwitch virtual appliances. This was something that would have been tested, if there were enough time within this project.

Creating more advanced case studies should also be a possibility since this project only scratches the surface of what the ADCs are capable of. An entire thesis work about iRules would most likely be a possibility and would provide a great resource for further development of the data center courses.

One of the more interesting projects would be trying to run a hypervisor inside the virtual laboratory. It could be used to simulate a more realistic data center server environment and would serve the purpose of this scenario perfectly. In theory, this should be possible but needs more advanced settings to be enabled to run properly.

Some ideas for the Cyberlab ADCs came to mind while doing this project. SSL offloading has been preliminary implemented into Cyberlab, but it has not been fully tested to be working with the servers needed. HTTP to HTTPS redirection could be implemented if needed. Advantages of different HTTP profiles could be studied. Server-side SSL traffic could offer more protection for the traffic. The hardening of the ADCs is one thing that could be considered.

## 8 CONCLUSIONS

Overall the project had a lot of variety. It was not just the implementation of application delivery controllers as one might have thought, but instead there were many things involving troubleshooting, servers and other network devices as well. Valuable troubleshooting experience was acquired from the project as there was much to troubleshoot, so much that everything could not be resolved within this project.

Working with the support to resolve the licensing issues was a great experience on what to look for when trying to resolve issues. A lot of information and logs needed to be provided, and it became clearer as to what kind of information is necessary. The licensing issues, however, were a bit of a disappointment since they took up a lot of time from the implementation. Because of that, many of ideas had to be left untested. It would have been great to get to know one of the more interesting features iRules better, because it seems like a powerful tool. Because of the lack of experience, it would most likely have taken some time to create your own scripts.

A lot of knowledge about application delivery and ADCs was learned while studying for and carrying out the project. At present, the knowledge about application oriented networking is valuable and will only grow to be more valuable in the future.

Even though there were several problems, the project can be considered successful. A working scenario and case study was created to serve the datacenter courses. Most of the basic functions so far have worked without problems, at least the ones that are necessary. Some functions such as VLAN tagging and synchronization between ADCs had to be left out since a fix for either one could not be found.



## REFERENCES

Boucadair, M. & Jacquenet, C. 2015. Handbook of Research on Redesigning the Future of Internet Architectures. 1st edition. Hershey: Information Science Reference.

Buys, J. 2012. Thinking Small With Tiny Core Linux. Available at: <http://ostatic.com/blog/thinking-small-with-tiny-core-linux> [Accessed: 5 October 2016].

Cisco Systems, Inc. 2014. Data Center Technology Design Guide. Available at: <https://www.cisco.com/c/dam/en/us/td/docs/solutions/CVD/Aug2014/CVD-DataCenterDesignGuide-AUG14.pdf> [Accessed: 5 October 2016].

Citrix. 2016. What is an Application Delivery Controller (ADC)?. Available at: <https://www.citrix.com/products/netscaler-adc/resources/what-is-an-adc.html> [Accessed: 18 September 2016].

F5. 2007. sol8082: Overview of TCP connection setup for BIG-IP LTM virtual server types. Available at: <https://support.f5.com/kb/en-us/solutions/public/8000/000/sol8082.html> [Accessed: 2 October 2016].

F5. 2009. sol10260: Mitigating Slowloris DoS attacks with the BIG-IP system. Available at: <https://support.f5.com/kb/en-us/solutions/public/10000/200/sol10260.html> [Accessed: 24 October 2016].

F5. 2016a. F5 Products. Available at: <https://f5.com/products/big-ip> [Accessed: 17 September 2016].

F5. 2016b. F5 Virtual Editions. Available at: <https://f5.com/products/platforms/virtual-editions> [Accessed: 17 September 2016].

F5. 2016c. F5 Local Traffic Manager. Available at: <https://f5.com/products/modules/local-traffic-manager> [Accessed: 17 September 2016].

F5. 2016d. F5 Application Security Manager. Available at: <https://f5.com/products/modules/application-security-manager> [Accessed: 17 September 2016].

F5. 2016e. Writing iRules. Available at: [https://support.f5.com/kb/en-us/products/big-ip\\_ltm/manuals/product/lm\\_configuration\\_guide\\_10\\_0\\_0/lm\\_rules.html](https://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/lm_configuration_guide_10_0_0/lm_rules.html) [Accessed: 17 September 2016].

F5. 2016f. Why Choose F5?. Available at: <https://f5.com/about-us/news/articles/why-choose-f5> [Accessed: 24 September 2016].

F5. 2016g. BIG-IQ and ADN Management. Available at: <https://f5.com/products/big-iq> [Accessed: 24 September 2016].

F5. 2016h. Centralized Management and Your Devices. Available at: <https://f5.com/products/big-iq/big-iq-device> [Accessed: 24 September 2016].

F5. 2016i. BIG-IP Virtual Editions Datasheet. Available at: <https://www.f5.com/pdf/products/big-ip-virtual-editions-datasheet.pdf> [Accessed: 24 September 2016].

F5. 2016j. Deploying BIG-IP Virtual Edition. Available at: [https://support.f5.com/kb/en-us/products/big-ip\\_ltm/manuals/product/bigip-ve-kvm-setup-11-3-0/2.html](https://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/bigip-ve-kvm-setup-11-3-0/2.html) [Accessed: 28 September 2016].

Holmes, D. 2011a. SSL Renegotiation DOS attack – an iRule Countermeasure. Available at: <https://devcentral.f5.com/articles/ssl-renegotiation-dos-attack-ndash-an-irule-countermeasure> [Accessed: 5 October 2016].

Holmes, D. 2011b. SSL Renegotiation DOS iRule - Updates. Available at: <https://devcentral.f5.com/articles/ssl-renegotiation-dos-irule-updates> [Accessed: 24 October 2016].

Holmes, D. 2013. Mitigating DDoS Attacks with F5 Technology. Available at: <http://www.igxglobal.com/wp-content/uploads/2013/03/mitigating-ddos-attacks-tech-brief.pdf> [Accessed: 25 October 2016].

IP Infusion Inc. 2013. ZebOS Network Platform IMI Command Reference. Available at: [https://support.f5.com/content/kb/en-us/products/big-ip\\_ltm/manuals/related/arm-imi-commandreference-7-8-4/\\_jcr\\_content/pdfAttach/download/file.res/arm-imi-commandreference-7-8-4.pdf](https://support.f5.com/content/kb/en-us/products/big-ip_ltm/manuals/related/arm-imi-commandreference-7-8-4/_jcr_content/pdfAttach/download/file.res/arm-imi-commandreference-7-8-4.pdf) [Accessed: 18 September 2016].

Jönsson, P. & Iveson, S. 2014. F5 Networks Application Delivery Fundamentals Study Guide. 1st edition. Self-Published.

Kasanen, L. 2013. Into the Core - A Look at Tiny Core Linux. Available at: <http://tinycorelinux.net/corebook.pdf> [Accessed: 5 October 2016].

KEMP. 2016a. Round Robin Load Balancing. Available at: <https://kemptech-nologies.com/load-balancing/round-robin-load-balancing/> [Accessed: 22 October 2016].

KEMP. 2016b. Source IP Hash load balancing. Available at: <https://kemptech-nologies.com/glossary/source-ip-hash-load-balancing/> [Accessed: 22 October 2016].

Kettunen, M. 2016. CyberLab-datakeskus. Available at: <http://www.ictlab.ky-amk.fi/index.php/fi/kyberturvallisuus/etusivu/oppimisymparisto/100-cyberlab-datakeskus> [Accessed: 25 October 2016].

Li, T., Cole, B., Morton, P. & Li, D. 1998. RFC 2281: Cisco Hot Standby Router Protocol (HSRP). Available at: <https://www.ietf.org/rfc/rfc2281.txt> [Accessed: 14 October 2016].

Nurmi, J. 2016. Implementation of Nested Virtual Laboratory System. Available at: [https://www.theseus.fi/bitstream/handle/10024/107061/Nurmi\\_Jaakko\\_Thesis.pdf?sequence=1](https://www.theseus.fi/bitstream/handle/10024/107061/Nurmi_Jaakko_Thesis.pdf?sequence=1) [Accessed: 17 September 2016].

Peltonen, A. 2016. Application Delivery Controller Implementation to Cyberlab Data Center. Available at: [https://www.theseus.fi/bitstream/handle/10024/110229/antti\\_peltonen.pdf?sequence=1](https://www.theseus.fi/bitstream/handle/10024/110229/antti_peltonen.pdf?sequence=1) [Accessed: 17 September 2016].

PRNewswire. 2014. Flat to Slight Growth for Application Delivery Controller (ADC) Market in Third Quarter 2014 as Virtual Appliance Revenues Surge, According to Dell'Oro Group. Available at: <http://www.prnewswire.com/news-releases/flat-to-slight-growth-for-application-delivery-controller-adc-market-in-third-quarter-2014-as-virtual-appliance-revenues-surge-according-to-delloro-group-300011171.html> [Accessed: 24 September 2016].

Salchow, K. 2012. Load Balancing 101: The Evolution to Application Delivery Controllers. Available at: <https://f5.com/resources/white-papers/load-balancing-101-the-evolution-to-application-de> [Accessed: 20 October 2016].

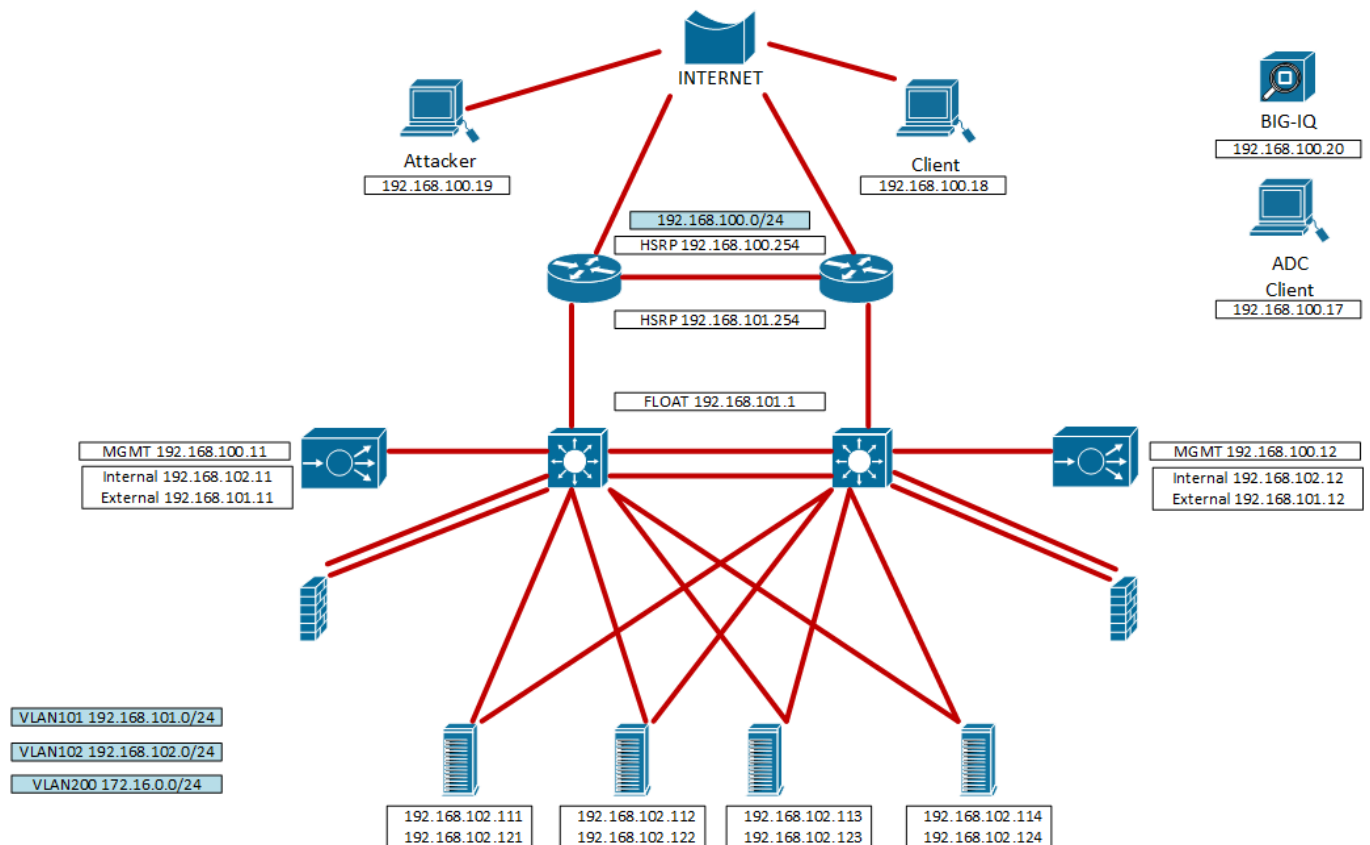
Shakarian, P., Shakarian, J. & Ruef, A. 2013. Introduction to Cyber-Warfare : A Multidisciplinary Approach. Syngress.

Tiny Core Team. 2007. Introduction to Core. Available at: <http://tinycorelinux.net/intro.html> [Accessed: 2 October 2016]

## DATACENTER APPLICATION DELIVERY

## CASE STUDY

20.11.2016



## Basic networking

1. Configure addressing and HSRP on the routers
2. Create the VLANs needed on the switches:
  - 101 external
  - 102 internal
  - 200 ha
3. Configure switch ports as appropriate access ports and trunk between switches
4. Ping from the user client to the HSRP gateways

## Management addresses and licensing

1. Access the ADC1 terminal with the credentials *root/default*
2. Access the traffic management operating system shell

```
tms
```

3. Disable DHCP on management port

```
modify sys db dhclient.mgmt value disable
```

4. Configure the management address

```
create /sys management-ip [ip address/prefixlen]
```

5. Save changes

```
save /sys config partitions all
```

6. Exit the shell

```
quit
```

7. Check the management address

```
ifconfig eth0
```

8. To license the ADC1, access the BIG-IQ device using the ADC client by navigating to <https://192.168.100.20> with the browser
9. Login with the credentials *admin/admin*
10. Navigate to Device > Provisioning
11. Add a new device using the + button

New Device	
Discover Cancel	
Device Properties	
IP Address	192.168.100.11
User Name	admin
Password	●●●●●
Device Group	Managed BIG-IPs
Framework Check	<input type="checkbox"/> Ignore Framework Check On Discovery
Update Framework	<input type="checkbox"/> Update Framework On Discovery

12. Right click the device and choose license device. License the device using the pool license.

### Basic configuration of the ADC

1. Access the management address of the ADC1 using the browser. Remember to use HTTPS!
2. Use the credentials *admin/admin*
3. Device certificate is imported using certificate and key located on the home directory of the ADC client

Setup Utility » **Device Certificates**

**SSL Certificate/Key Source**

Import Type	Certificate and Key ▾
Certificate Name	server
Certificate Source	<input checked="" type="radio"/> Upload File <input type="radio"/> Paste Text Browse... laboratory.cert.cert
Key Source	<input checked="" type="radio"/> Upload File <input type="radio"/> Paste Text Browse... laboratory.cert.key
Free Space on Disk	229 MB

Cancel Import

4. Navigate to platform settings
5. Name the device using a fully qualified domain name e.g. *adc1.dclab.fi*
6. Set the Time Zone as UTC
7. Next create the VLANs

Setup Utility » **VLANs**

**External Network Configuration**

External VLAN	<input checked="" type="radio"/> Create VLAN external <input type="radio"/> Select existing VLAN
Self IP	Address: 192.168.101.11 Netmask: 255.255.255.0 Port Lockdown: Allow Default ▾
Default Gateway	192.168.101.254
Floating IP	Address: 192.168.101.1 Port Lockdown: Allow Default ▾

**External VLAN Configuration**

VLAN Name	external
VLAN Tag ID	101
Interfaces	VLAN Interfaces: 1.2 ▾ Tagging: Untagged ▾ Add 1.1 (untagged) Edit Delete

Cancel Next...

8. DO NOT use VLAN tagging
9. Leave Port lockdown settings to default
10. For default gateway on external VLAN, use the HSRP gateway created earlier
11. After creating VLANs the Setup Utility can be exited. Failover is not configured.

### Load balancing

1. To start load balancing, first create a pool of servers start by navigating to Local Traffic > Pools > Pool List

The screenshot shows the configuration page for a new pool named "DCLAB-HTTP-Servers". The configuration is set to "Basic".

**Name:** DCLAB-HTTP-Servers

**Description:** (Empty)

**Health Monitors:**

- Active:** gateway\_icmp, http
- Available:** http\_head\_f5, https, https\_443, https\_head\_f5

**Resources:**

- Load Balancing Method:** Round Robin
- Priority Group Activation:** Disabled

**New Members:**

- Radio Buttons:**  New Node  New FQDN Node
- Node Name:** C2B-S2 (Optional)
- Address:** 192.168.102.124
- Service Port:** 80, HTTP
- Add Button:** Add
- Member List:**
  - R:1 P:0 C:0 C1A-S1 192.168.102.111 :80
  - R:1 P:0 C:0 C1A-S2 192.168.102.121 :80
  - R:1 P:0 C:0 C1B-S1 192.168.102.112 :80
  - R:1 P:0 C:0 C1B-S2 192.168.102.122 :80
  - R:1 P:0 C:0 C2A-S1 192.168.102.113 :80
- Edit/Delete Buttons:** Edit, Delete

2. Use 'gateway\_icmp' and 'http' health monitors for availability checking
3. Use Round Robin as load balancing method
4. Service port is 80 for HTTP
5. Add all the server interfaces, two on each server, eight in total
6. When finished, access the Pool list and select the pool
7. Check that all the servers are available, which is indicated by a green dot
8. Create a virtual server. Start by navigating to Local Traffic > Virtual Servers > Virtual Server List



Local Traffic » Virtual Servers : Virtual Server List » **New Virtual Server**

**General Properties**

Name	HTTP_www-dclab-fi	
Description	Dclab.fi HTTP server	
Type	Standard	
Source Address	0.0.0.0/0	
Destination Address/Mask	192.168.101.111	
Service Port	80	HTTP
Notify Status to Virtual Address	<input checked="" type="checkbox"/>	
State	Enabled	

**Configuration:** Basic

Protocol	TCP	
Protocol Profile (Client)	tcp	
Protocol Profile (Server)	(Use Client Profile)	
HTTP Profile	http	

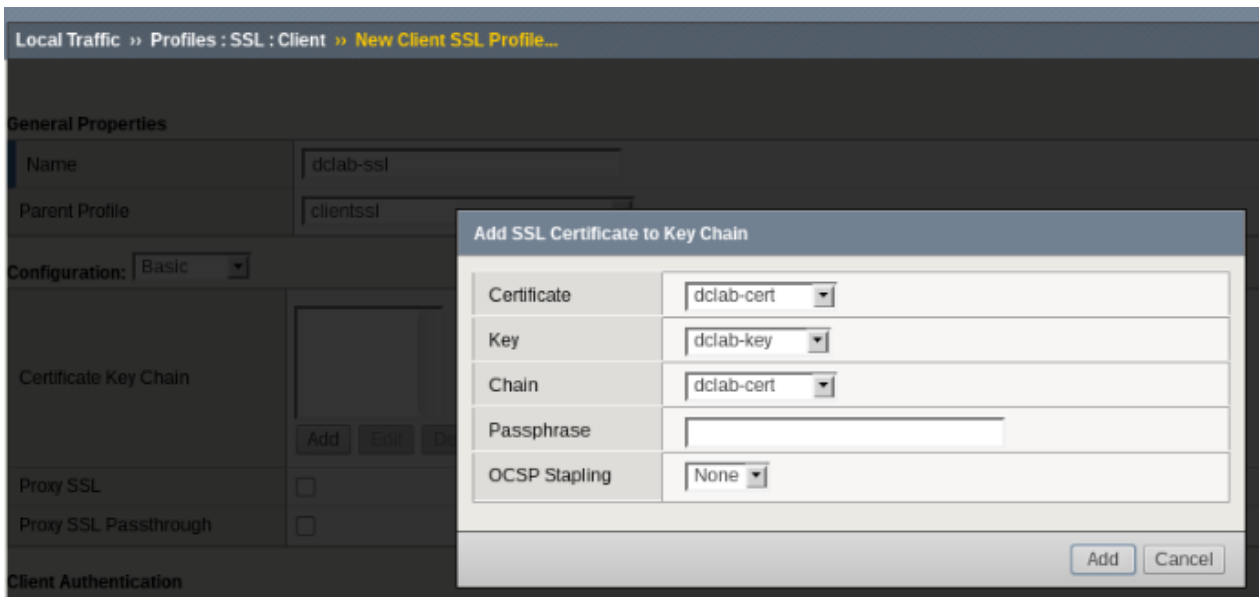
9. Use source address *0.0.0.0/0* to allow all source addresses
10. The destination address is an address for the virtual server that is used to access the service. e.g. *192.168.101.111*
11. Use the HTTP port and select *http* as HTTP Profile
12. Under Resources, access the drop-down menu of Default Pool and select the pool created earlier
13. After saving the settings load balancing should be active, and can be tested using the User machines browser to access the address of the virtual server

#### Load balancing between gateways

1. Remove HSRP from VLAN 101 (Not from the outside network)
2. Create a pool for the router interfaces as before, but only use *gateway\_icmp* for monitoring
3. Navigate to Network > Routes to manage the *external\_default\_gateway*
4. Change the resource to use pool and select the pool created
5. Now load balancing between the interfaces should be enabled. However, note that the load balancing does not happen between the outside network gateways and the traffic returns from the active HSRP interface. What is the problem with this?

#### SSL offloading

1. Navigate to System > File Management > SSL Certificate List and import the certificate and key used earlier
2. The key and certificate are imported separately. Choose import type depending on which you are importing



3. A Client SSL Profile needs to be created. Start by navigating to Local Traffic > Profiles > SSL > Client
4. Create a new profile and add the certificate to the Certificate Key Chain
5. Select the correct certificate, key and chain
6. Leave everything else to their default values and save the profile



7. Create a new HTTPS virtual server
8. This time select the HTTPS service port
9. Select the SSL Profile (Client) created earlier
10. Everything else is identical to the HTTP virtual server
11. SSL offloading should be active and can be tested by accessing the virtual server address using a secure connection e.g. `https://192.168.101.111`
12. Next the HTTP traffic needs to be redirected to HTTPS using a policy. Start by navigating to Local Traffic > Policies > Policy List
13. Create a policy that executes the first matching rule
14. After creating the policy, add a new rule to the policy
15. Match all traffic
16. Configure the task

Do the following when the traffic is matched:

*Redirect to location: `tcl:https://[getfield [HTTP::host] ":" 1][HTTP::uri]`*

17. Save and publish the policy
18. Navigate to HTTP virtual server Resources tab
19. Add the policy under Policies
20. Check that the policy redirects the HTTP traffic to HTTPS

Mitigating denial of service attacks

1. Use `slowhttptest` on the Attacker client to launch a slowloris denial of service attack on the virtual server

```
slowhttptest -c 10000 -H -i 5 -r 50 -t GET -u https://192.168.101.111
```

2. Test accessing the HTTP service using the User client. The service should be inaccessible soon after the attack begins
3. You can exit the attack tool using CTRL + C
4. Effects of a slowloris attack can be mitigated using an iRule
5. To create a new iRule, navigate to Local Traffic > iRules > iRule List
6. Add the following iRule provided by F5 to mitigate slowloris attacks

```
when CLIENT_ACCEPTED {
  set rtimer 0
  after 1000 {
    if { not $rtimer} {
      drop
    }
  }
}

when HTTP_REQUEST {
  set rtimer 1
}
```

7. After saving the iRule it needs to be activated on the virtual server on the Resources tab
8. Activate the iRule on both virtual servers
9. Use the attack earlier to again. Looking at the `slowhttptest` statistics, do you see a difference?
10. Try to access the service again, and it should be accessible.
11. Note that if you launch a bigger attack, the switches and routers do not have enough resources to cope with it. This should not happen in a real environment since the attack does not generate much traffic.
12. Now that SSL is enabled the virtual server is vulnerable to SSL renegotiation attack
13. Disable the slowloris iRule to be able to manually test for the vulnerability
14. Use the following command on the User machine

```
openssl s_client -connect www.dclab.fi:443
```

15. Type R and press enter to renegotiate the connection. You can do it five times before you need to reconnect. Please reconnect and try once more. You should be able to renegotiate as before, which means that it can be done indefinitely

16. Renegotiation takes up 10 times the resources from the server compared with the client. If you can renegotiate the handshake indefinitely, it is a serious vulnerability to denial-of-service attack. Using an automated attack tool, it is possible to take up a lot of resources
17. It possible to limit this by using the following iRule also provided by F5

```

when RULE_INIT {
    set static::maxquery 5
    set static::mseconds 60000
}
when CLIENT_ACCEPTED {
    set ssl_hs_reqs 0
}
when CLIENTSSL_HANDSHAKE {
    incr ssl_hs_reqs
    after $static::mseconds { if {$ssl_hs_reqs > 0} {incr ssl_hs_reqs -1} }
    if { $ssl_hs_reqs > $static::maxquery } {
        after 5000
        log "Handshake attack detected, drop-ping
[IP::client_addr]:[TCP::client_port]"
        drop
    }
}

```

18. Activate the iRule on the HTTPS server and try to renegotiate the connection again
19. It should be possible to renegotiate the connection only five times and after this the connection hangs
20. You should not be able to connect for a minute after this
21. Look at the logs on the ADC. There should be messages saying that an attack was detected