Web Server Security

SAFA GALLAH (RT4)

CYRINE CHAYEB (RT4)

DHOUHA BEN SAID (GL4)

Table des matières

1.Présentation de l'atelier :

Avec Internet, les réseaux sont dorénavant ouverts et par conséquent, beaucoup plus exposés aux attaques virales ou autres actes de piratage. Les premières attaques réseau exploitaient des vulnérabilités liées à l'implémentation des protocoles de la suite TCP/IP. Avec la correction progressive de ces vulnérabilités les attaques se sont décalées vers les couches applicatives et en particulier le web, dans la mesure où la plupart des entreprises ouvrent leur système pare-feu pour le trafic destiné au web. Il est donc primordial de savoir faire face à ces différents risques pour protéger les données et garantir l'intégrité et le bon fonctionnement du système d'information ; et tout cela doit commencer par une bonne sécurisation du serveur web utilisé.

2.Introduction au serveur web:

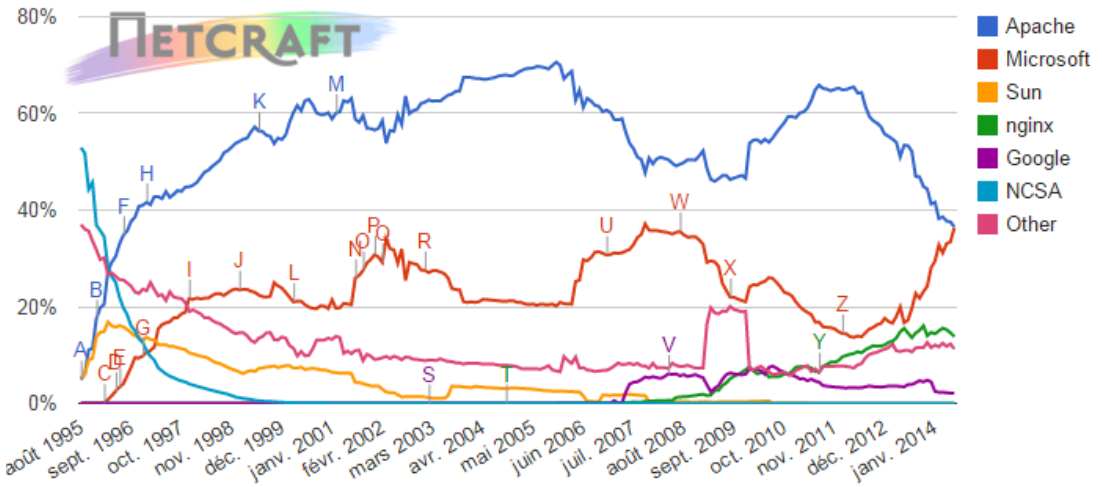
a) Définition d'un serveur web :

Un serveur Web est un serveur informatique utilisé pour publier des sites web sur Internet ou un intranet. Les serveurs Web publics sont reliés à Internet et hébergent des ressources (pages Web, images, vidéos, etc.) du Web. Certains serveurs sont seulement accessibles sur des réseaux privés (intranet) et hébergent des sites utilisateurs, des documents, ou des logiciels, internes à une entreprise, une administration, etc. La fonction principale d'un serveur Web est de stocker et délivrer des pages Web qui sont généralement écrites en HTML. Le protocole de communicationHyperText Transfer Protocol (HTTP) permet de dialoguer avec le logiciel client, généralement un navigateur Web. Vu les progrès achevés sur les serveurs web, il devrait ainsi être conçu pour prendre en charge de nombreux modules lui donnant des fonctionnalités supplémentaires : interprétation du langage Perl, PHP,Python et Ruby, serveur proxy, Common Gateway Interface, Server Side Includes, réécriture d'URL, négociation de contenu, protocoles de communication additionnels, etc.

b) Les serveurs web les plus utilisés :

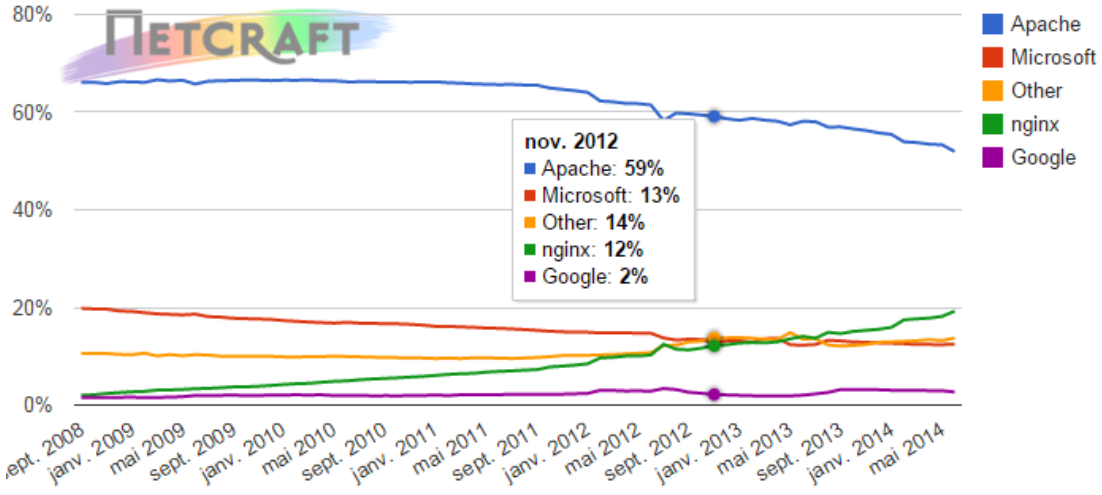
Le «Battle » entre les serveurs web est de plus en plus féroce. Cependant les leaders, qui restent au top 3 des serveurs web les plus utilisés, sont Apache HTTP server suivi de Microsoft IIS et enfin Nginx avec, en juin 2014, des parts du marchérespectivement51.92%, 19.21%, 12.45%.

Web server developers: Market share of all sites



Developer	May 2014	Percent	June 2014	Percent	Change
Apache	366,262,346	37.56%	353,672,431	36.50%	-1.05
Microsoft	325,854,054	33.41%	352,208,487	36.35%	2.94
nginx	142,426,538	14.60%	133,763,494	13.81%	-0.80
Google	20,685,165	2.12%	20,192,595	2.08%	-0.04

Web server developers: Market share of the top million busiest sites



Developer	May 2014	Percent	June 2014	Percent	Change
Apache	533,232	53.32%	519,227	51.92%	-1.40
nginx	181,612	18.16%	192,142	19.21%	1.05
Microsoft	123,823	12.38%	124,487	12.45%	0.07
Google	29,238	2.92%	26,833	2.68%	-0.24

c) Les failles les plus répandues dans les applications et les serveurs web :

D'après the Open Web application Security Project (OWASP) a fournit en 2013 comme chaque année, le top 10 des risques de sécurité applicatifs web les plus critiques qui sont :

1. Injection : telle l'injection SQL, OS et LDAP
2. Violation de Gestion d'Authentification et de Session : (compromettre les mots de passe, clés, jetons de session, ou exploiter d'autres failles d'implémentation pour s'approprier les identités d'autres utilisateurs)
3. Cross-Site Scripting(XSS)
4. Références directes non sécurisées à un objet :
5. Mauvaise configuration Sécurité
6. Exposition de données sensibles
7. Manque de contrôle d'accès au niveau fonctionnel
8. Falsification de requête intersites (CSRF)
9. Utilisation de composants avec des vulnérabilités connues
10. Redirections et renvois non validés

3. Introduction au serveur Hiawatha :

a) Présentation :

Hiawatha est un serveur web open source dont le développement a démarré en janvier 2002. Il était conçu à l'origine comme un petit serveur web par un étudiant en informatique Néerlandais ; ensuite, développé de façon expérimentale, il en est résulté un serveur HTTP très orienté sécurité. Sa dernière version 9.9 est disponible gratuitement sur son site officielwww.hiawatha-webserver.org .

Hiawatha a testé et est parfaitement compatible avec Linux, BSD, MacOS X et Windows. Il met en œuvre ce qui est généralement attendu d'un serveur web moderne mais il présente aussi de nombreuses fonctions axées sur la sécurité que peu de ses concurrents mettent en œuvre ; ainsi Hiawatha embarque des protections contre :

- les injections SQL
- les failles Cross-Site Scripting (XSS) et Cross-Site Request Forgery (CSRF)
- les attaques par déni de service (DoS)
- les liens externes vers des images

Hiawatha permet aussi la configuration qu'une interdiction paramétrable contre d'éventuels programmeurs malintentionnés ainsi qu'une limitation de l'exécution de script CGI (éviter l'emballement qui provoque généralement une perte de contrôle du serveur).

b) Installation et Configuration :

On se placera pour l'installation de Hiawatha 9.9 dans un environnement basé linux : ubuntu server 14.04. (on se placera en mode root pour tout se qui suit).

Remarque : la distribution de ubuntu server ne présente pas une interface graphique pour lui en apporté il suffit de taper la commande suivante : sudo apt-get install ubuntu-desktop

Pré-installation :

$Sudo su

#apt-get update
#apt-get dist-upgrade
#apt-get autoclean
#apt-get --purge autoremove

Step 1 - Installation de PHP5

#apt-get install php5-cgi php5 php5-cli php5-mysql php5-curl php5-gd php5-intl php-pear php5-imagick php5-imap php5-mcrypt php5-memcache php5-ming php5-ps php5-pspell php5-recode php5-snmp php5-sqlite php5-tidy php5-xmlrpc php5-xsl php5-xcache apache2-utils php5-fpm mysql-server mysql-client

Step 2 - Installation de Hiawatha :

#apt-get install libc6-dev libssl-dev dpkg-dev debhelper fakeroot libxml2-dev libxslt1-dev

Télécharger la dernière version de Cmake :

#wget http://www.cmake.org/files/v3.0/cmake-3.0.2.tar.gz
#tar -xvzf cmake-3.0.2.tar.gz
#cd cmake-3.0.2
#./configure
#make
#make install

Télécharger la dernière version de hiawatha :

#wget http://www.hiawatha-webserver.org/files/hiawatha-9.9.tar.gz
#tar -xzvf hiawatha-9.9.tar.gz
#cd hiawatha-9.9/extra
#./make_debian_package
#cd ..
#dpkg -i hiawatha_9.9_i386.deb

Step 3 - Configuration PHP5 (sécurisation)

sudo gedit /etc/php5/fpm/php.ini
Effectuer les changements suivant :.
zlib.output_compression = On
zlib.output_compression_level = 6

Step 3a - Configure PHP5 (Optionnnel for securitypurpose)

allow_url_fopen = Off
expose_php = Off
enable_dl = Off
session.cookie_httponly = 1
disable_functions =
system,show_source,symlink,exec,dl,shell_exec,passthru,phpinfo,escapeshellarg,esapeshellcmd,

cgi.fix_pathinfo = 0

Step 4 – Configuration de php-fpm

ajouter les lignes suivantes au fichier de configuration php-fpm.conf :

sudogedit /etc/php5/fpm/php-fpm.conf

[www]
user = www-data
group = www-data
listen.mode = 0666
listen = /var/run/php5-fpm.sock
pm = static
pm.max_children = 100
chdir = /

Step 5 - Configure Hiawatha (Part 1)

sudogedit/etc/hiawatha/hiawatha.conf

Cette étape concernera la configuration du service http et non du https:

```
# Hiawatha main configuration file

# GENERAL SETTINGS

#

ServerId = www-data

ConnectionsTotal = 1000

ConnectionsPerIP = 35

SystemLogfile = /var/log/hiawatha/system.log
```

```
GarbageLogfile = /var/log/hiawatha/garbage.log

ExploitLogfile = /var/log/hiawatha/exploit.log

LogFormat = extended

ServerString = SimpleHTTPserver

CGIwrapper = /usr/sbin/cgi-wrapper

# BINDING SETTINGS

# A binding is where a client can connect to.

#

Binding {

        Port = 80

#       Interface = 127.0.0.1

        MaxKeepAlive = 50

        TimeForRequest = 12,50

}

BanOnGarbage = 300

BanOnMaxPerIP = 300

BanOnMaxReqSize = 300

BanOnTimeout = 300

KickOnBan = yes

RebanDuringBan = yes
```

```
BanOnDeniedBody = 300

BanOnSQLi = 300

BanOnFlooding = 90/1:300

BanlistMask = deny 127.0.0.1

BanOnInvalidURL = 300

BanOnWrongPassword = 3:300

# COMMON GATEWAY INTERFACE (CGI) SETTINGS

# These settings can be used to run CGI applications.

#

CGIhandler = /usr/bin/perl:pl

#CGIhandler = /usr/bin/php5-cgi:php

CGIhandler = /usr/sbin/php5-fpm:php

CGIhandler = /usr/bin/python:py

CGIhandler = /usr/bin/ruby:rb

CGIhandler = /usr/bin/ssi-cgi:shtml

CGIextension = cgi

#

FastCGIserver {

        FastCGIid = PHP5

#       ConnectTo = 127.0.0.1:9000

        ConnectTo = /var/run/php5-fpm.sock
```

```
        Extension = php

        SessionTimeout = 600

}




# URL TOOLKIT

# This URL toolkit rule was made for the Banshee PHP framework,
which

# can be downloaded from http://www.hiawatha-webserver.org/banshee

#

UrlToolkit {

        ToolkitID = banshee

        Call scannerblocker

        Call vulnerabilityblocker

        RequestURI isfile Return

        Match ^/(css|files|images|js|slimstat)($|/) Return

        Match ^/(favicon.ico|robots.txt|sitemap.xml)$ Return

        Match ^/(crawler)($|/) Return

        Match .*\?(.*) Rewrite /index.php?$1

        Match .* Rewrite /index.php

}
```

```
UrlToolkit {

        ToolkitID = vulnerabilityblocker

        Header * \(\)\s*\{ DenyAccess   # Shellshock

        MatchCI ^/(crawler|pma|myadmin|phpmyadmin|cgi-bin)($|/) Ban
900 # phpmyadmin & cgi-bin

        MatchCI ^/(xmlrpc.php|phpinfo.php)$ Ban 900 # wordpress,
drupal & phpinfo

}



UrlToolkit {

        ToolkitID = scannerblocker

        Header User-Agent ^w3af.sourceforge.net DenyAccess

        Header User-Agent ^dirbuster DenyAccess

        Header User-Agent ^nikto DenyAccess

        Header User-Agent ^sqlmap DenyAccess

        Header User-Agent ^fimap DenyAccess

        Header User-Agent ^nessus DenyAccess

        Header User-Agent ^Nessus DenyAccess

        Header User-Agent ^whatweb DenyAccess

        Header User-Agent ^Openvas DenyAccess

        Header User-Agent ^jbrofuzz DenyAccess

        Header User-Agent ^libwhisker DenyAccess
```

```
Header User-Agent ^webshag DenyAccess

Header User-Agent ^Morfeus DenyAccess

Header User-Agent ^Fucking DenyAccess

Header User-Agent ^Scanner DenyAccess

Header User-Agent ^Aboundex DenyAccess

Header User-Agent ^AlphaServer DenyAccess

Header User-Agent ^Indy DenyAccess

Header User-Agent ^ZmEu DenyAccess

Header User-Agent ^social DenyAccess

Header User-Agent ^Zollard DenyAccess

Header User-Agent ^CLR DenyAccess

Header User-Agent ^Camino DenyAccess

Header User-Agent ^Nmap DenyAccess

Header * ^WVS DenyAccess

Header User-Agent ^Python-httplib DenyAccess

Header User-Agent ^Python-requests DenyAccess

Header User-Agent ^masscan DenyAccess

Header User-Agent ^Java DenyAccess

Header User-Agent ^Nutch DenyAccess

Header User-Agent ^Who.is DenyAccess

Header User-Agent ^immoral DenyAccess
```

```
        Header User-Agent ^crawler DenyAccess

        Header User-Agent ^NetShelter DenyAccess

        Header User-Agent ^Application DenyAccess

        Header User-Agent ^Validator.nu/LV DenyAccess

        Header * ^ssdp DenyAccess

        Header User-Agent ^Arachni DenyAccess

        Header User-Agent ^Spider-Pig DenyAccess

        Header User-Agent ^tinfoilsecurity DenyAccess

        Header User-Agent ^@ DenyAccess

        Header User-Agent ^shellshock-scan DenyAccess

}

# DEFAULT WEBSITE

# It is wise to use your IP address as the hostname of the default
website

# and give it a blank webpage. By doing so, automated webscanners
won't find

# your possible vulnerable website.

#

Hostname = 98.139.183.24

WebsiteRoot = /var/www/hiawatha

StartFile = index.html

AccessLogfile = /var/log/hiawatha/access.log
```

```
ErrorLogfile = /var/log/hiawatha/error.log

#ErrorHandler = 404:/error.cgi

ReverseProxy ^/.* http://www.example.com:80/



Include /etc/hiawatha/enable-sites/



# VIRTUAL HOSTS

# Use a VirtualHost section to declare the websites you want to
host.

#

#VirtualHost {

#       Hostname = www.my-domain.com

#       WebsiteRoot = /var/www/my-domain/public

#       StartFile = index.php

#       AccessLogfile = /var/www/my-domain/log/access.log

#       ErrorLogfile = /var/www/my-domain/log/error.log

#       TimeForCGI = 5

#       UseFastCGI = PHP5

#       UseToolkit = banshee

#}
```

```
   # DIRECTORY SETTINGS

   # You can specify some settings per directory.

   #

   #Directory {

   #       Path = /home/baduser

   #       ExecuteCGI = no

   #       UploadSpeed = 10,2

#}
```

Step 5a :

→ajouter la ligne suivante à VIRTUAL HOSTS : Include /etc/hiawatha/enable-sites/

sudomkdir /etc/hiawatha/enable-sites
sudomkdir /etc/hiawatha/disable-sites

Step 6 – Configuration de Hiawatha (Part 2)

→Ayant un nom de domaine mysite.com, on crée un fichier nommé mysite.com et on le place sous /etc/hiawatha/enable-sites/mysite.com.

```
   VirtualHost {

      Hostname = www.mysite.com, mysite.com

      WebsiteRoot = /var/www/mysite

      StartFile = index.php

      AccessLogfile = /var/log/hiawatha/access.log



      ErrorLogfile = /var/log/hiawatha/error.log

      TimeForCGI = 1000

      UseFastCGI = PHP5
```

```
    UseToolkit = banshee

    # if ownCloud or alike is installed, otherwise, it should be "no"

#    WebDAVapp = yes

    # <script .. </script>

    # e.g. <script>alert("xss");</script>

    DenyBody = ^.*%3Cscript.*%3C%2Fscript%3E.*$

    DenyBody = ^.*%3CsCrIpT.*%3C%2FScRiPt%3E.*$

    DenyBody = ^.*%3CScRiPt.*%3C%2FsCrIpT%3E.*$

    DenyBody = ^.*%3CSCRIPT.*%3C%2FSCRIPT%3E.*$

    DenyBody = ^.*%3CSCRIPT.*%3C%2Fscript%3E.*$

    DenyBody = ^.*%3Cscript.*%3C%2FSCRIPT%3E.*$

    # <meta .. />

    # e.g. <meta http-equiv="refresh" content='0;
URL=http://some.domain"/>

    DenyBody = ^.*%3Cmeta.*%2F%3E.*$

    DenyBody = ^.*%3CMETA.*%2F%3E.*$

    DenyBody = ^.*%3CMeTa.*%2F%3E.*$

    DenyBody = ^.*%3CmEtA.*%2F%3E.*$

    # <iframe .. />

    DenyBody = ^.*%3Ciframe.*%2F%3E.*$

    DenyBody = ^.*%3CIFRAME.*%2F%3E.*$

    # Null Byte
```

```
    DenyBody = ^.*%00.*$

#   ExecuteCGI = yes

    PreventCSRF = yes

    PreventSQLi = yes

    PreventXSS = yes

    WrapCGI = jail_mysite

}
```

Step 7 - Configure Hiawatha (Part 3) (Optional for securitypurpose)

sudogedit /etc/hiawatha/cgi-wrapper.conf

CGIhandler = /usr/bin/perl
#CGIhandler = /usr/bin/php5-cgi
CGIhandler = /usr/sbin/php5-fpm
CGIhandler = /usr/bin/python
CGIhandler = /usr/bin/ruby
CGIhandler = /usr/bin/ssi-cgi

Wrap = jail_mysite ; /var/www/mysite ; www-data:www-data

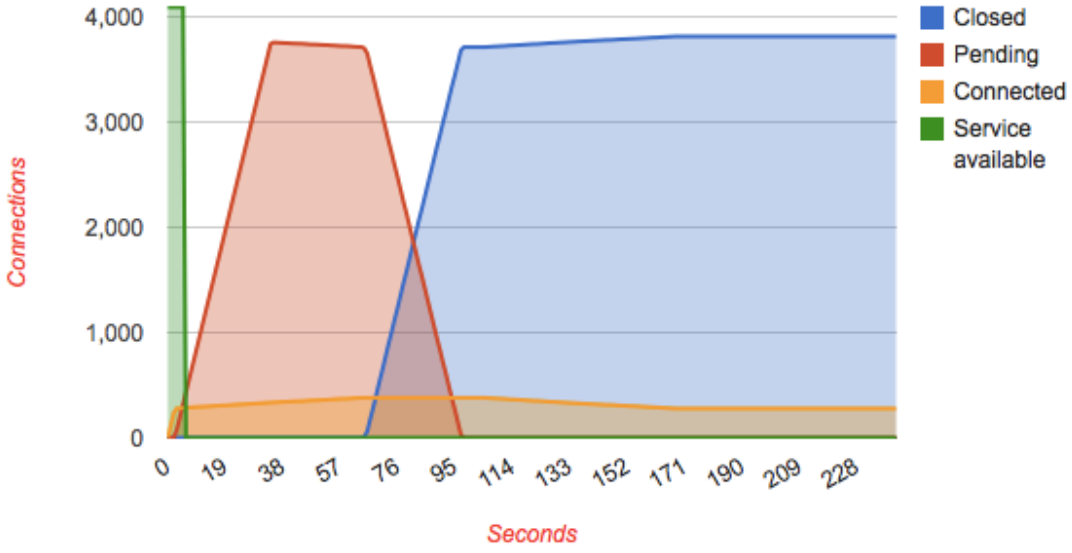
Remarque : beaucoup d'autres configurations peuvent être considérées afin d'assurer une meilleure robustesse du serveur web. Pour cela on pourraitconsulter le site officiel www.hiawatha-webserver.orgou le blog suivant secure-ubuntu-server.blogspot.com

4. Résultats de test d'attaque :

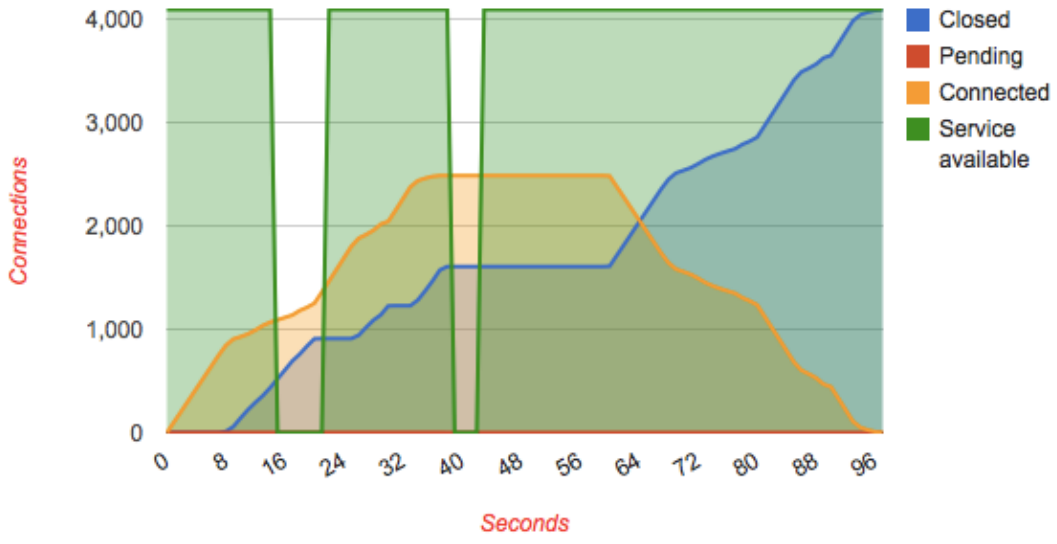
Afin d'évaluer laperformance de certains serveurs web par rapport à Hiawatha on effectuer le cas de test suivant et les résultats du test :

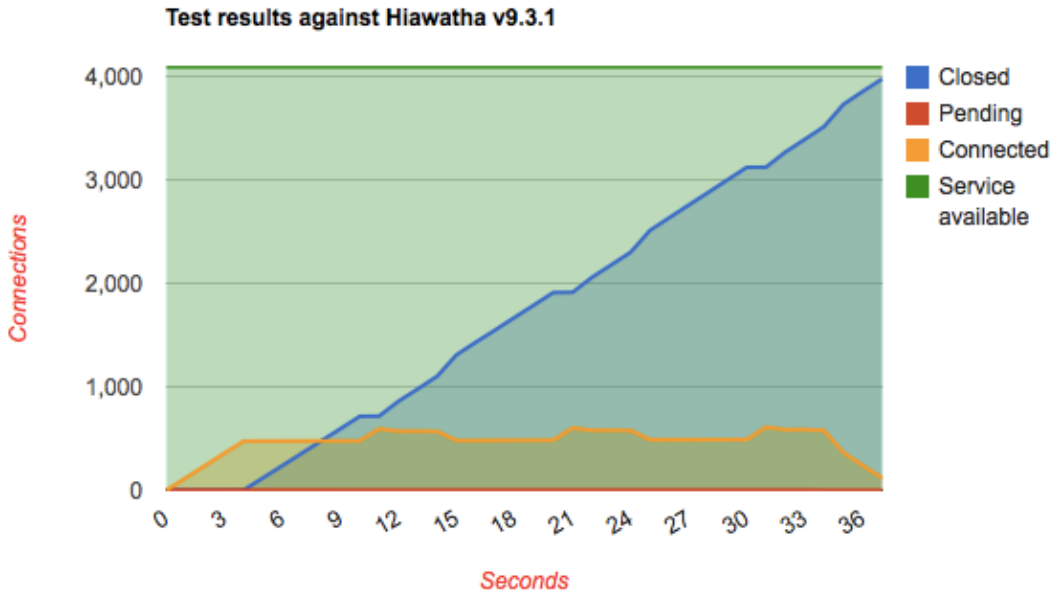
Test parameters	
Test type	SLOW HEADERS
Number of connections	4096
Verb	GET
Content-Length header value	4096
Extra data max length	52
Interval between follow up data	10 seconds
Connections per seconds	128
Timeout for probe connection	3
Target test duration	240 seconds
Using proxy	no proxy

Test results against Apache/2.4.7



Test results against nginx/1.4.5





	Cherokee	Hiawatha	Lighttpd	LiteSpeed	Nginx
Dernière version	1.2.101	8.1	1.4.30	4.2.1	1.0.13
Date de sortie	18 oct. 2011	25 févr. 2012	18 déc. 2011	2 août 2011	5 mars 2012
Licence	GPL	GPL 2	BSD	Free (Standard Edition) Commercial license (Enterprise Edition)	BSD
Site web	cherokee-project.com	hiawatha-webserver.org	lighttpd.net	litespeedtech.com	nginx.org
Langage de développement	C	C	C		C
Fonctionnalités					
Authentification	✓	✓	✓	✓	✓ Basic-Only
Support HTTPS	✓	✓	✓	✓	✓
Domaine virtuel	✓	✓	✓	✓	✓
Support CGI	✓	✓	✓	✓	✗
Support FastCGI	✓	✓	✓	✓	✓
ASP.NET	✗	✗	✗	✗	✗
Java Servlets	✗	✗	✗	✗	✗
Server Side Includes	✓	✓	✓	✗	✓
Console d'administration	✓	✓	✗	✓	✗ Plugin
Support IPv6	✓	✓	✓	✓	✓
Plateformes					
Windows	✓	✓	✓	✗	✓
Linux	✓	✓	✓	✓	✓
Mac OS	✓	✓	✓	✓	✓
Détails des plateformes supportés	Unix-like, Windows	Unix-like, Windows, Haiku os	Unix-like, Windows		Unix-like, Windows

Tableau comparatif entre les différents Serveurs web

5.Conclusion

Malgré sa performance, les statistiques sur l'usage de Hiawatha sont fréquemment sous-estimées.(il est utilisé par moins de 0.1% de tous les sites web actifs). En effet En raison de l'accent qu'il met sur la sécurité, le serveur HTTP Hiawatha peut rejeter

des requêtes de robots n'étant pas identifiés comme les robots d'indexation des moteurs de recherche, parmi lesquels peuvent se trouver les robots d'entités collectant simplement des statistiques