ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

MÉTODO DE DESPLIEGUE Y MEJORAMIENTO DE LA SEGURIDAD WEB EN LAS PEQUEÑAS Y MEDIANAS EMPRESAS (PYME) UTILIZANDO LAS CARACTERÍSTICAS DE SEGURIDAD DE HTTP 2.0

AUTORA: ISABEL DEL ROCÍO LEÓN PAILIACHO

Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo, presentado ante el Instituto de Postgrado y Educación Continua de la ESPOCH, como requisito parcial para la obtención del grado de:

MAGÍSTER EN SEGURIDAD TELEMÁTICA

RIOBAMBA – ECUADOR

FEBRERO 2017

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

CERTIFICACIÓN

EL TRIBUNAL DE TRABAJO DE TITULACIÓN CERTIFICA QUE:

El Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo, titulado "***MÉTODO DE DESPLIEGUE Y MEJORAMIENTO DE LA SEGURIDAD WEB EN LAS PEQUEÑAS Y MEDIANAS EMPRESAS (PYME) UTILIZANDO LAS CARACTERÍSTICAS DE SEGURIDAD DE HTTP 2.0***", de responsabilidad de la Ing. Isabel del Rocío León Pailiacho, ha sido prolijamente revisado y se autoriza su presentación.

Tribunal:

Ph.D. Fredy Proaño Ortiz **PRESIDENTE**	_____ FIRMA
Ing. Paúl Fernando Bernal Barzallo, Msc **DIRECTOR**	_____ FIRMA
Ing. Diego Fernando Ávila Pesántez, Msc **MIEMBRO**	_____ FIRMA
Ing. Gloria de Lourdes Arcos Medina, Msc **MIEMBRO**	_____ FIRMA

Riobamba, febrero 2017

DERECHOS INTELECTUALES

Yo, Isabel del Rocío León Pailiacho, declaro que soy responsable de las ideas, doctrinas y resultados expuestos en el **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, “Método de despliegue y mejoramiento de la seguridad web en las pequeñas y medianas empresas (Pyme) utilizando las características de seguridad de HTTP 2.0”, y el patrimonio intelectual generado por la misma pertenece exclusivamente a la Escuela Superior Politécnica de Chimborazo.

_____

Ing. Isabel del Rocío León Pailiacho

No. Cédula 0603974437

DECLARACIÓN DE AUTENTICIDAD

Yo, Isabel del Rocío León Pailiacho, declaro que el presente **Trabajo de Titulación modalidad Proyectos de Investigación y Desarrollo**, es de mi autoría y que los resultados del mismo son auténticos y originales. Los textos constantes en el documento que provienen de otras fuentes están debidamente citados y referenciados.

Como autora, asumo la responsabilidad legal y académica de los contenidos de este Trabajo de Titulación de Maestría.

_____

Ing. Isabel del Rocío León Pailiacho

No. Cédula 0603974437

DEDICATORIA

Dedico este trabajo a mi madre, mis hermanos, mis amigos y tutor quién me ha ayudado en la consecución del proyecto haciéndome ver mis errores para corregirlos y alentándome a seguir mejorando para lograr grandes metas en mi vida profesional.

Isabel León Pailiacho

AGRADECIMIENTO

Agradezco a Dios y a la Virgen Santísima por guiar mi camino; a mi madre, por ser el pilar fundamental en mi vida quién me ha inculcado buenas valores y amor; a mis hermanos y a mi familia por estar conmigo en todos los momentos y brindarme un apoyo incondicional y verdadero.

Isabel León Pailiacho

CONTENIDO

LISTA DE TABLAS

LISTA DE GRÁFICOS

LISTA DE ANEXOS

RESUMEN

El trabajo de investigación método de despliegue y mejoramiento de la seguridad web en las pequeñas y medianas empresas (Pyme) utilizando las características de seguridad de HTTP 2.0, realizado en la ciudad de Riobamba beneficia a los administradores de sistemas para tener rendimiento y seguridad en la web de las Pyme. Se utilizó el método científico por tener una serie de etapas que van desde el planteamiento del problema, el análisis e interpretación de resultados, y se usó el método propuesto de despliegue y mejoramiento de la seguridad web cuyas 7 fases son: reconocimiento, exploración, obtención de acceso, explotación de vulnerabilidades, evaluación de vulnerabilidades de HTTP 1.1, mitigación de vulnerabilidades con HTTP 2.0 y valoración de las vulnerabilidades corregidas. El software empleado fue Fedora 23 en los servidores por poseer las últimas versiones de Apache para HTTP 2.0, Kali Linux para buscar, y explotar vulnerabilidades, VMware Workstation para virtualizaciones, Nessus para escanear vulnerabilidades y, Slowloris para ataques de denegación de servicios. Se implementó un escenario práctico en el cual frente a las vulnerabilidades encontradas de HTTP 1.1 se aplicó las características de HTTP 2.0, y se elaboró la guía práctica para desplegar HTTP 2.0 en servidores. Para la comprobación de la hipótesis se aplicó el método propuesto, se compararon los indicadores, se aplicó estadística descriptiva y se usó SPSS. El análisis de resultados concluye que HTTP 2.0 brinda mayor seguridad, rendimiento al manejar información por la web, y optimización de recursos frente al HTTP 1.1, obtuvo una vulnerabilidad activa del 14.58% y corregida del 85.42% frente a un 72.92% y 27.08% respectivamente con HTTP 1.1, motivos suficientes para recomendar a los Administradores de Sistemas usar la guía y mantener una continua actualización en HTTP 2.0 que aún es nuevo y falta explotar.

Palabras Clave: <SEGURIDAD WEB>, <PROTOCOLO DE TRANSFERENCIA DE HIPERTEXTO VERSIÓN 2.0 (HTTP 2.0)>, <SISTEMA OPERATIVO (FEDORA 23)>, <VULNERABILIDADES>, <EMPRESAS PEQUEÑAS Y MEDIANAS (PYME)>, <RIOBAMBA (CANTÓN)>.

ABSTRACT

The research work method of deployment and improvement of web security in small and medium-sized enterprises (SMES) using the security features of HTTP 2.0, carried out in the city of Riobamba benefits to system administrators to have performance and security in the web of SMES. The scientific method was used for having a series of stages that go from the exposition of the problem, the analysis and interpretation of results, and there was used the proposed method of deployment and improvement of the web safety which 7 phases are: recognition, exploration, obtaining access, exploitation of vulnerabilities, vulnerability assessment of HTTP 1.1, mitigating vulnerabilities with HTTP 2.0 and assessment of corrected vulnerabilities. The software used was Fedora 23 on servers by having the latest versions of Apache for HTTP 2.0, Kali Linux to search for and exploit vulnerabilities, VMware Workstation for virtualization, Nessus for scan vulnerabilities and, Slowloris for denial-of-service attacks. Implemented a practical stage in which compared to the vulnerabilities found HTTP 1.1 it is applied the characteristics of HTTP 2.0, and developed the practical guide to deploy HTTP 2.0 in servers. For the verification of the hypothesis the proposed method was applied, the indicators were compared, it applied descriptive statistics and SPSS was used. Results analysis concludes that HTTP 2.0 provides greater security, performance to manage information on the web and optimization of resources opposite to the HTTP 1.1, obtained an active vulnerability 14.58% and corrected of the 85.42% compared to 72.92% and 27.08% respectively with HTTP 1.1, sufficient grounds to recommend Systems Administrators to use the guide and maintain a continuous update in HTTP 2.0 which is still new and lack exploit.

KEY WORDS: <WEB SECURITY>, <HYPERTEXT TRANSFER PROTOCOL 2.0 (HTTP 2.0)>, <OPERATING SYSTEM (FEDORA 23)>, <VULNERABILITY>. <SMALL AND MEDIUM-SIZED ENTERPRISES (SMES)>, <RIOBAMBA (CANTON)>.

CAPÍTULO I

INTRODUCCIÓN

Con la creciente demanda de la Word Wide Web, es decir del sistema de distribución de documentos de hipertexto interconectado y accesible por internet; los sitios web compuestos de páginas web contienen no solo texto, imágenes, sino vídeos u otros contenidos multimedia, los cuales son accedidos por HTTP.

El protocolo de comunicación HTTP permite la transferencia de información en la Word Wide Web, la versión aún más usada es 1.1 pero frente a los problemas de rendimiento, mayor latencia al cargar las páginas web, varias conexiones, información redundante, y sobre todo la inseguridad.

Surge el nuevo protocolo HTTP 2.0 basado en el proyecto de SPDY, el cual es compatible con HTTP 1.1 no cambia la semántica, métodos, códigos de estado, e introduce mejoras como el uso de una sola conexión, compresión de cabeceras, prioridad de flujos, server push, uso de encriptación TLS, entre otras.

La seguridad de la información consiste en la preservación de los datos en confidencialidad, integridad y disponibilidad; pero aún la seguridad es el eslabón más débil de un sistema informático ya que siempre existirán pequeños agujeros siendo una batalla constante entre el atacante y el administrador de los sistemas.

Las empresas sean pequeñas, medianas o grandes son víctimas de los atacantes quienes aprovechan las vulnerabilidades para penetrar al sistema, donde los incidentes más comunes son accesos no autorizados, denegación de servicios, virus, mal uso de protocolos, etc.

Dado a conocer la situación de los problemas de velocidad de carga en las páginas web, latencia e inseguridad con el protocolo HTTP 1.1, se plantea la investigación titulada “Método de despliegue y mejoramiento de la seguridad web en las pequeñas y medianas empresas (PYME) utilizando las características de seguridad de HTTP 2.0”.

En el presente capítulo, se da a conocer el enfoque u orientación general, identificación del problema, justificación, objetivos e hipótesis que se desea comprobar con el desarrollo de la investigación.

1.1 Problema de Investigación

1.1.1. Planteamiento del Problema

HTTP "Hypertext Transfer Protocol", es el protocolo de transferencia de hipertexto, usado en cada transacción de la World Wide Web, es un protocolo orientado a la transacción basada en el esquema petición-respuesta entre el cliente y servidor.

El cliente que efectúa la petición se le llama agente del usuario, la información transmitida se llama recurso y se identifica mediante URL (localizador uniforme de recursos); y el resultado puede ser la ejecución de un programa, una consulta a una base de datos, etc. («Hypertext Transfer Protocol», 2015)

SPDY "Speedy", protocolo de nivel de aplicación del modelo OSI y complementario al HTTP que funciona sobre TCP/IP. Su propósito es reducir el tiempo de carga de las páginas web, usa una conexión TCP para manejar varias peticiones HTTP, también usa SSL (Secure Sockets Layer, capa de conexión segura, protocolo criptográfico que brinda una comunicación segura por la red) como capa subyacente para incrementar la seguridad. («SPDY», 2014)

HTTP 2.0, protocolo que permite un mayor intercambio de mensajes entre el computador y los servidores, lo que admite visitar más sitios al mismo tiempo y aumentar las velocidades de carga de los mismos. Por lo cual toda la navegación será mucho más rápida incluida las búsquedas web y no se gastará tantos recursos para lograr este cometido.

HTTP 2.0 se basa en SPDY que maneja gran cantidad de tráfico y no permite que éstos colapsen. Por ejemplo, Google recibe cerca de 40.000 solicitudes de búsqueda por segundo, lo cual haría caer a cualquier otro sitio web, pero con HTTP 2.0 se procesa dichos requerimientos sin que el servicio principal se caiga. («Por qué HTTP/2 es el protocolo que revolucionará Internet», 2015)

En el año 1999, las comunicaciones y tecnologías web fueron totalmente diferentes como el tráfico multimedia anecdótico, conexiones móviles inexistentes, por lo que fue increíble pensar todo el tráfico que la WWW mueve actualmente, y es indudable el hecho de que Internet y la informática cambian a un paso muy acelerado, avances que a finales de la década de 1990 no hubiera imaginado nadie, ni siquiera Tim Berners-Lee, padre de la web. («¿Qué es el HTTP 2.0 y qué cambios traerá?», 2014)

HTTP utiliza una conexión TCP independiente para cada archivo solicitado lo cual implica sobrecarga significativa e innecesaria, gran latencia, amplio uso de los recursos de red y ancho de banda, tiempo de respuesta de carga excesivo de los sitios web más visitados, poco rendimiento y eficiencia en las comunicaciones web lo cual es considerado también un aspecto de seguridad pues mientras más eficiente es un servicio, más fuerte y estable es para soportar cierto tipo de ataques, como los de tipo DoS capaz de atender muchas más peticiones por unidad de tiempo.

HTTP no puede regular el contenido de los datos que se transfiere, tampoco no tiene un método a priori que determine la prioridad del envío de la información; por consiguiente, las solicitudes deben suministrar mucho control sobre dicha información. Además, el revelar la versión de software específica del servidor podría permitir que el servidor sea más vulnerable a los ataques contra software conocidos como agujeros de seguridad. El uso de la criptografía protege de ataques de seguridad y privacidad, y HTTP 1.1 no usa criptografía.

HTTP 1.1 ya no es suficiente para manejar los volúmenes de transferencias en la web. Por eso, desde el 2009 con SPDY se pretendió mejorar la seguridad y eficiencia alcanzando una reducción del tiempo de carga de páginas de al menos un 50%; («¿Qué es el HTTP 2.0 y qué cambios traerá?», 2014), en el 2010 al duplicarse el tamaño total de transferencia en la web, HTTPbis Working Group de la IETF decidió usar SPDY como base para la nueva versión HTTP 2.0.

HTTP 2.0 fue aprobado el miércoles 18 de febrero del 2015 por parte del IESG “Grupo de Ingeniería de Internet Directivo, responsable de la gestión técnica de las actividades del IETF y el proceso de los estándares de Internet”, es considerado la gran actualización del sistema que utiliza la red informática mundial WWW en 15 años, totalmente compatible con HTTP 1.1 y que funciona con la red existente.

HTTP 2.0 soluciona algunos de los inconvenientes más comunes de HTTP 1.1, como el mejorar el rendimiento, uso más eficiente de los recursos de la red, reducción de latencia a través de la compresión de campos de cabecera y multiplexado, por lo tanto, permitirá múltiples solicitudes al servidor con una sola conexión de los navegadores y a la vez el servidor responderá a las distintas peticiones simultaneas.

Los desarrolladores manifiestan que la red será más rápida y los sistemas de encriptación mejorarán, pues HTTP 2.0 conlleva encriptación. Tanto Chrome y Firefox tienen soporte para HTTP 2.0.

Por último, los usuarios no visualizarán cambios visibles más que la velocidad del servicio, en la barra de direcciones se verá http:// para las direcciones que aún la muestran, y el navegador de forma automática hará el cambio entre HTTP 1.1 y 2.0.

Al ser HTTP 2.0 un protocolo que involucra un tema nuevo, el tiempo no ha sido suficiente para que se hayan publicado estudios científicos del tema, y de hecho este trabajo pretende ser justamente eso, uno de los primeros análisis científicos sobre HTTP 2.0.

Actualmente todos los sitios web usan HTTP 1.1, el cual tiene problemas como: según los autores ***E. Casilari, F. J. González, y F. Sandoval, en su paper Modeling of HTTP traffic.*** HTTP 1.0 o 1.1 abre una nueva conexión TCP para cada objeto incrustado en la página Web esto aumenta la latencia, la sobrecarga de protocolo, y el establecimiento de una conexión TCP requiere un handshake de 3 vías. Para hacer frente a este problema, versión 1.1 propuso las llamadas conexiones persistentes que se mantienen abiertas (hasta que se agote el tiempo de espera) una vez que un objeto es entregado al navegador. Si se emite una nueva petición, las conexiones abiertas son re-utilizados, evitando la necesidad de abrir nuevas. Tanto HTTP 1.0 y 1.1 en los tráficos no presentan diferencias notables, sin embargo, se demostró que la existencia de un tiempo de espera tanto para la técnica de mantenimiento de conexión y conexión persistente establece una fuerte correlación entre el comportamiento de los usuarios y el tráfico generado a nivel TCP.

Según ***L. Shuai, G. Xie, y J. Yang, en Characterization of HTTP behavior on access networks in Web 2.0.*** La mediana duración de HTTP es 240 bytes y la media de 320

bytes, esto significa que la longitud de petición HTTP viene aumentado casi dos veces más que antes. Las razones por las que la longitud de petición aumenta son:

1) La longitud de URL: en la solicitud HTTP se hace más grande que antes, y los datos experimentales muestran que el campo URL por lo general toma porcentaje de liderazgo en longitud total.
2) Browser: un hecho notable es que para la misma URL porque contiene más información como el campo de la cabecera. Así la actualización Web del navegador es también una razón por lo que la longitud de petición tiene aumento.
3) Carga: el resultado indica que sobre el 6% de peticiones son con POST y tienen un tamaño medio de alrededor de 640 a 700 bytes, y las solicitudes con POST toman 2% y son 1,600 bytes en promedio. (Shuai, Xie, & Yang, 2008).

HTTP 2.0 promete solucionar los inconvenientes de HTTP 1.1 según lo dicho en el estándar Hypertext Transfer Protocol version 2 -- draft-ietf-httpbis-http2-16. Por lo tanto, se va a realizar el estudio del protocolo HTTP 2.0 para probarlo, éste trabajo investigativo será de los primeros que se realicen sobre el tema, por lo que contribuirá en el inicio de la construcción del estado del arte.

El presente tema “Método de despliegue y mejoramiento de la seguridad web en las pequeñas y medianas empresas (Pyme) utilizando las características de seguridad de HTTP 2.0”, es aplicable al ámbito del despliegue de soluciones de aseguramiento y optimización de servidores web tanto en las instituciones públicas y privadas de nuestro país y la región.

1.1.2. Formulación del Problema

¿Cuál es el nivel de mejora con las características de seguridad que tiene el protocolo HTTP 2.0 para el despliegue y mejoramiento de la seguridad web en las Pyme?

1.1.3. Sistematización del Problema

¿Qué es el protocolo HTTP 2.0?
¿Cuáles son las nuevas características del protocolo HTTP 2.0 frente a HTTP 1.1?
¿Cuáles son las ventajas y desventajas del protocolo HTTP 2.0?

¿Qué características de seguridad brinda HTTP 2.0 en la navegación?
¿Cómo aporta HTTP 2.0 al rendimiento como primera frontera de protección contra ataques de inundación?
¿Cuál es la mejora de la seguridad web en las Pyme con el protocolo HTTP 2.0?

1.2 Justificación de la Investigación

1.2.1. Justificación Teórica

El lanzamiento del protocolo HTTP 2.0 tiene muchos beneficios por las nuevas características que incorpora como son: ser un protocolo binario y no textual como HTTP 1.1 lo que permite eficiencia y menor susceptibilidad a errores; la multiplexación logra que varias solicitudes y mensajes de respuesta puedan darse al mismo tiempo sin bloquear ninguna de ellas con lo cual elimina el Head of Line Blocking "bloqueo de cabeza de línea"[1].

La incorporación de la capa de framing binario mejora el rendimiento por que los mensajes entre el cliente y servidor son encapsulados de una forma más compacta, fácil, eficiente y segura similar al funcionamiento del HTTPS; también minimiza los gastos de protocolo y sobrecarga a través de la compresión eficiente de los campos de cabecera HTTP; agrega un soporte para la solicitud de priorización y servidor de inserción; y reduce la latencia innecesaria con un uso más efectivo de los recursos de la red con tiempos de carga más bajos en las páginas web.

En cuanto a la seguridad HTTP 2.0 obliga el uso del protocolo SSL por lo cual las aplicaciones serán más rápidas, sencillas y con recuperación de pérdidas. Mejorar globalmente la seguridad web es forzar a usar url con HTTPS, es decir direcciones de internet con HTTP sobre SSL, lo cual no significa que las páginas web no seguras desaparezcan al contrario seguirán funcionando con HTTP 1.1. Se usará HTTP Strict Transport Security "Seguridad de transporte HTTP estricta", que es un mecanismo de

[1] "El servidor envía sus respuestas en el mismo orden en que recibe las solicitudes tomando en cuenta que solo es una conexión a la vez en HTTP 1.1 y por usar TCP los paquetes son entregados en orden y retransmitidos si se pierde uno de ellos". (networking - SPDY Head of Line blocking - Stack Overflow. Disponible en: http://stackoverflow.com/questions/25221954/spdy-head-of-line-blocking. 2014

política de seguridad web donde un servidor interactúa con los navegadores solo mediante conexiones HTTP seguras.

Por tales razones el presente tema de investigación “Método de despliegue y mejoramiento de la seguridad web en las pequeñas y medianas empresas (Pyme) utilizando las características de seguridad de HTTP 2.0”, pretende realizar un estudio de éste nuevo protocolo HTTP 2.0, analizar sus características, ventajas y desventajas y sobretodo enfocarse en las mejoras de seguridad y rendimiento web que brinda frente al HTTP 1.1; y a la vez desarrollar una guía práctica de despliegue de servidores con HTTP 2.0 orientada a las Pyme.

1.2.2. Justificación Práctica

El presente tema de investigación se implementará en un ambiente de laboratorio con dos servidores de prueba correspondientes a los protocolos HTTP 2.0 y HTTP 1.1, sobre dichos equipos se realizarán y determinarán diferentes pruebas para medir comparativamente los valores de rendimiento y seguridad de ambos protocolos HTTP.

El resultado obtenido será no solo la comparativa y sus valores que de por sí ya resultan muy valiosos en un área sobre la cual todavía se sabe muy poco, sino también un método de implementación del protocolo HTTP 2.0, que será de enorme valía para todos los administradores de sistemas que deseen desplegar la nueva versión del protocolo en sus servidores, especialmente en las “Pyme” que son las que promueve el gobierno su desarrollo como células económicas que impulsan la economía del país, sin embargo, podría ser usado por todo tipo de empresa, pero en las grandes la infraestructura sería muy compleja como para abarcar en la investigación.

1.3 Objetivos de la Investigación

1.3.1. Objetivo General

Proponer un método de despliegue y mejoramiento de la seguridad web en las pequeñas y medianas empresas (Pyme) utilizando las características de seguridad de HTTP 2.0.

1.3.2. Objetivos Específicos

- Estudiar el protocolo HTTP 2.0 y determinar sus principales características respecto del HTTP 1.1.
- Identificar los mecanismos de seguridad que brinda HTTP 2.0 para la web.
- Realizar la comparabilidad de características de seguridad y rendimiento entre los protocolos estudiados.
- Diseñar o implementar un entorno de pruebas prácticas con los servidores HTTP 1.1 y HTTP 2.0.
- Realizar una guía práctica de despliegue para servidores con HTTP 2.0 orientado a las Pyme.

1.4 Hipótesis

La aplicación del método de despliegue de HTTP 2.0 mejorará la seguridad web en las Pyme.

CAPÍTULO II

MARCO DE REFERENCIA

En éste capítulo se profundiza el estado del arte y se da a conocer las bases teóricas del objeto de estudio de la presente investigación, empezando por el protocolo SPDY por ser el catalizador de HTTP 2.0, HTTP y sus versiones, y HTTP 2.0 como es definición, características, seguridad, etc.

2.1 Estado del Arte

HTTP 2.0 es un protocolo aún nuevo, por lo que no hay publicaciones de estudios científicos, y éste trabajo investigativo será de los primeros que se realicen sobre el tema, por lo que contribuirá en el inicio de la construcción del estado del arte. Entre las investigaciones se mencionan:

Investigación titulada "*Modeling of HTTP traffic"* (Casilari, Gonzalez, & Sandoval, 2001)

La motivación de los autores para la realización del presente artículo científico es mostrar que el comportamiento de los usuarios de internet afecta significativamente en la evolución de las conexiones TCP, y el tiempo entre las descargas de páginas es fundamental para determinar la reutilización de las conexiones.

Analizan que la versión HTTP 1.0 abre una conexión TCP por cada objeto de la página web, aumenta la latencia y sobrecarga, y para solventar este inconveniente HTTP 1.1 tiene conexiones persistentes las cuales se mantienen abiertas una vez que el objeto es entregado al navegador, si existe más solicitudes la conexión abierta es reutilizada.

Demostrando así que existe fuerte correlación entre el comportamiento de los usuarios y el tráfico generado a nivel TCP. Los autores para el flujo de HTTP proponen un modelo con 4 niveles como:

Nivel de sesión: describe el comportamiento de los usuarios en términos de sesión, como el número de web por día y la distribución de sesiones.

Nivel de página: determinar el número de páginas web visitadas por sesión y la distribución del tiempo entre páginas.

Nivel de conexión: el número de conexiones para cada página, el tamaño de las conexiones, y el tiempo entre dos conexiones consecutivas.

Nivel de paquete: la conexión es dividida en paquete TCP/IP, y se analiza los tamaños y tiempos en que llegan los paquetes.

Indican que HTTP impone una correlación entre estos niveles, y el tiempo de espera de las conexiones persistentes establece una dependencia entre las páginas y los niveles de conexión, para lo cual los autores proponen dos escenarios.

El primer escenario tiene más de 200 páginas web heterogenias situadas en diferentes contextos nacionales, y no existe ninguna diferencia entre tener HTTP 1.0 y 1.1 pues el número de conexiones por página permanece prácticamente sin cambios y en el gráfico 1-2 el tamaño de la conexión es modelada por una distribución de Pareto.

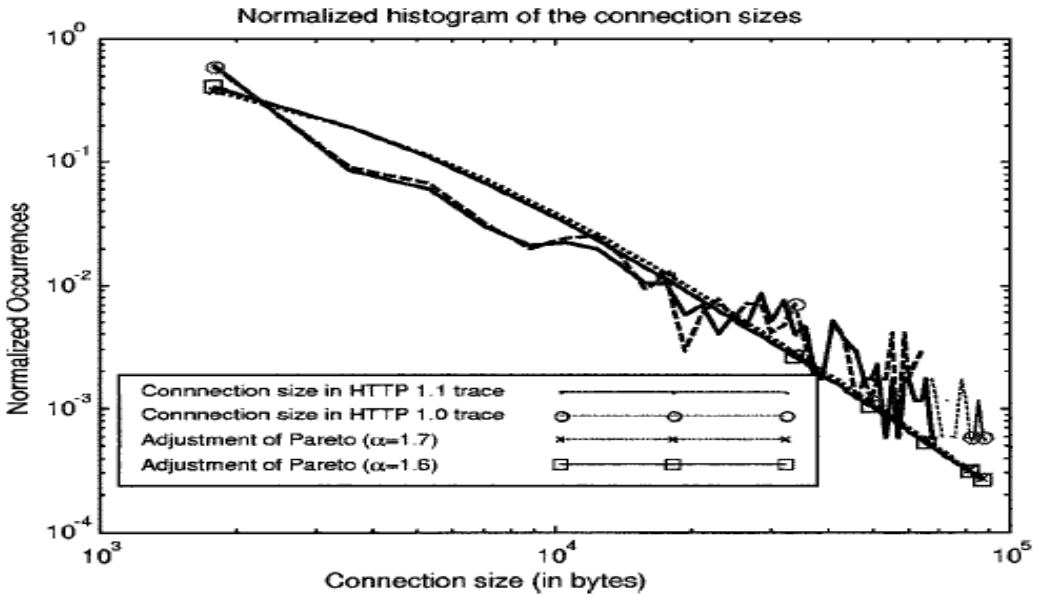


Gráfico 1-2 Histograma normalizado de los tamaños de conexión

Fuente: Casilari, Gonzalez, & Sandoval, 2001

En el segundo escenario la navegación es realizada a través de un único sitio web, donde se tiene que más de 150 páginas son visitadas por navegador, y en el gráfico 2-2 se muestra una fuerte influencia del tiempo entre páginas en el nivel de conexión con tres zonas definidas.

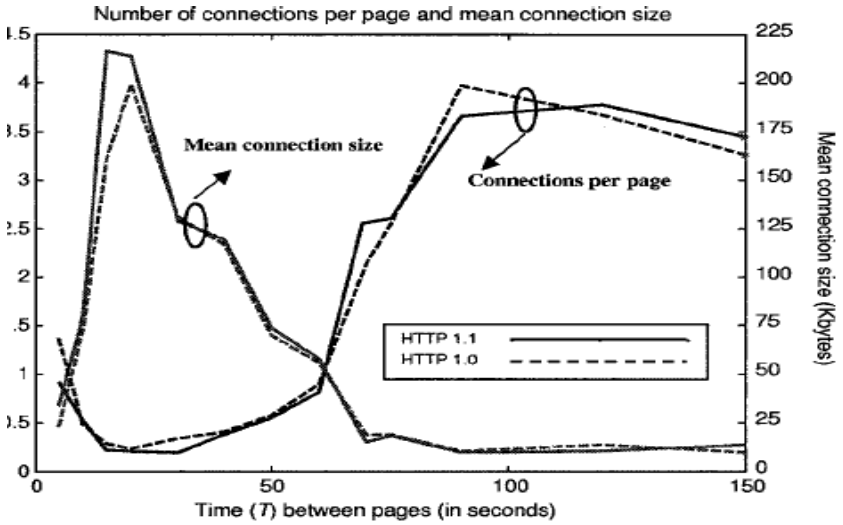


Gráfico 2-2 Media del número de conexiones por página y tamaño de la conexión como un medio en función del intervalo T entre páginas
Fuente: Casilari, Gonzalez, & Sandoval, 2001

1. Zona inicial con tiempos bajos entre las páginas con menos de 15s, considerando que es cuando el usuario no espera a que la pagina este cargado por completo y ya da clic en un hipervínculo. Por lo tanto, la conexión persistente es pobre y el número de conexiones por página aumenta.
2. Segunda zona con tiempos medios entre páginas, de 15 a 60s, implica visitas rápidas, debido a que las transmisiones han terminado, cargar una nueva página involucra reutilizar la conexión inactiva y el tráfico HTTP consiste en menos y más grandes conexiones TCP.
3. Zona con tiempos altos entre páginas, más de 60s, en este caso el tiempo de espera de las conexiones obliga a cerrarla antes de solicitar una nueva página y no son reutilizadas; por lo tanto, el número de conexiones incrementa notablemente, mientras que la carga media de cada conexión disminuye.

De acuerdo a la investigación los autores demuestran que el tiempo entre las páginas web es fundamental para determinar si se utiliza las conexiones TCP existentes o si se tiene que abrir otras nuevas, también que es mejor usar HTTP 1.1 frente a 1.0 por reducir

la latencia, introducir el concepto de conexión persistentes, y soportar mecanismos de negociación de la compresión de datos.

Sin embargo, la investigación realizada ve como ventaja la versión 1.1 frente a la 1.0, y considera que los tamaños de las conexiones TCP se verán afectados por HTTP 1.1. Pero no existen ni indicios de todos los problemas que se tiene actualmente con HTTP 1.1 y la actualización a ésta versión que es HTTP 2.0.

Investigación titulada "*Characterization of HTTP behavior on access networks in Web 2.0"* (Shuai, Xie, & Yang, 2008).

Los autores analizan el comportamiento de HTTP en tres niveles como son: el nivel del mensaje de HTTP, nivel de conexión de TCP y el nivel de flujo web dividido en dos categorías: la concurrencia de los flujos iniciados por el mismo host y los flujos iniciados por diferentes hosts.

Nivel de mensaje de HTTP

En este nivel los autores analizan la longitud y duración del mensaje, indicando que la mediana duración de HTTP es 240 bytes y la media es 320 bytes, significando que la longitud de HTTP es casi dos veces más que antes por el aumento en la longitud de URL, el browser que solicita campos de cabecera, y la carga por enviar peticiones con el método POST.

En el gráfico 3-2 se muestra la distribución de la longitud de petición, donde las solicitudes están a 2000 bytes; las curvas de la función de distribución acumulativa (CDF) son similares en los segmentos A, B y C, y en D es pronunciada teniendo una distribución centralizada en alrededor de 200 bytes.

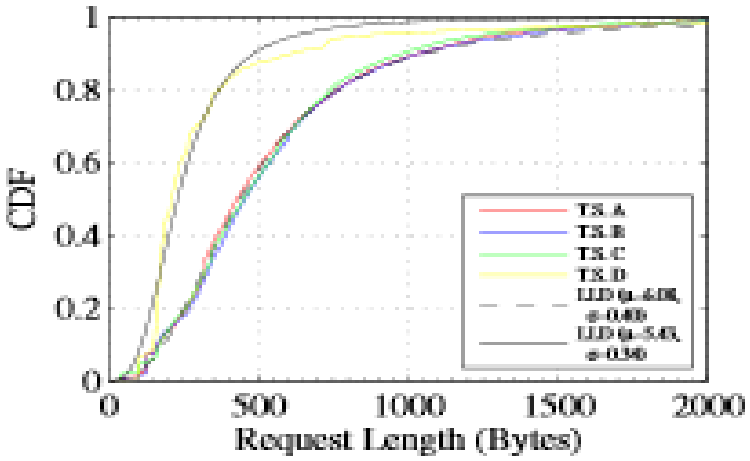


Gráfico 3-2 CDF de longitud de petición

Fuente: Shuai, Xie, & Yang, 2008

En el gráfico 4-2 se muestra el CDF de longitud de respuesta HTTP, donde la distribución de longitud de respuesta en A, B y C está cerca de cada otra mientras que en D tiene una pequeña diferencia.

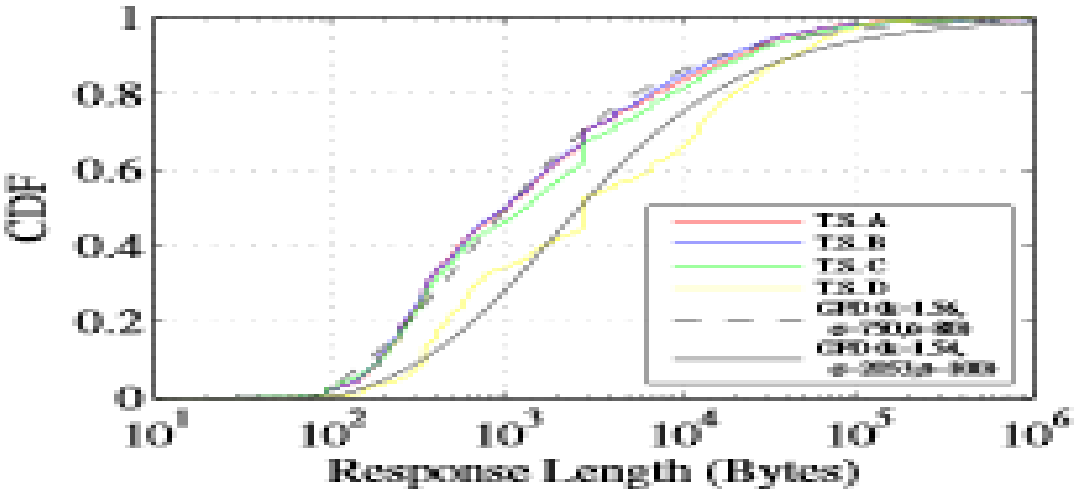


Gráfico 4-2 CDF de longitud de respuesta

Fuente: Shuai, Xie, & Yang, 2008

En la mayoría de las respuestas HTTP el campo de cabecera sólo toma una pequeña porción en todo el mensaje, y la longitud de la respuesta decide principalmente por la longitud del archivo transferido.

En concreto, en A, B, y C el 36% de respuestas son archivos HTML y el 40% son de imagen, mientras que en D alrededor del 76% las respuestas son archivos HTML y el

12% imagen. Casi todas las respuestas son HTML e imágenes y la mediana de duración de archivo HTML es 2.0-2.4 KB.

Nivel de conexión de TCP

El número de mensajes transportados y los intervalos entre los mensajes se caracterizan, el tener conexiones persistentes implica una solicitud y un par de respuestas porque la página web tiene varios objetos y los navegadores inicializan varias conexiones simultáneas para la página web con el fin de reducir el tiempo de transporte y mejorar la navegación del usuario. El tiempo de los mensajes puede ser caracterizado por las secuencias de duración de la petición y de la respuesta, el tiempo de espera, y tiempo de inactividad.

Nivel de flujo web

El flujo web se define como un grupo de conexiones TCP a la misma dirección IP de origen, donde el intervalo de conexiones TCP se ve afectada por dos factores como son el navegador web ya que luego de tener la página web inicializada se hará conexiones posteriores para obtener recursos como imágenes, archivos css, etc, esto implica 1000 milisegundos; y el comportamiento del usuario pues éste navega por varios hipervínculos lo que se implica tener varios segundos e incluso horas.

Sin embargo, la investigación realizada basa el comportamiento de HTTP solo en los tres niveles, pero no existe ningún indicio del nuevo protocolo HTTP 2.0 peor en relación a la seguridad, tampoco tiene el aporte de los autores para investigaciones futuras.

2.2 Spdy

Historia de SPDY

SPDY es un protocolo experimental, desarrollado por Google y anunciado a mediados del año 2009, para reducir la latencia de carga de páginas web que tiene HTTP 1.1. Tiene soporte en los navegadores Google Chrome, Internet Explorer (versión 11), Opera (versión 12.10 en adelante) y Mozilla Firefox (versión 11 hasta la 28), así como también muchos destinos web de gran tamaño como Google, Twitter, y Facebook.

Entre los objetivos están los siguientes:

- Reducir el 50% en el tiempo de carga de página.
- Minimizar la complejidad de implementación y evitar cambios en la infraestructura de red.
- Desarrollar éste nuevo protocolo de colaboración con la comunidad de código abierto.

Para lograr la mejora del tiempo de carga de página al 50%, SPDY dirige un uso más eficiente de la conexión TCP mediante la introducción de la capa framing binario que mejora el rendimiento porque los mensajes entre el cliente y servidor son encapsulados de forma más compacta, fácil, eficiente y segura; tiene multiplexación ya que logra que varias solicitudes y mensajes de respuesta puedan darse al mismo tiempo sin bloquear ninguna de ellas; así como el establecimiento de prioridades, y el hecho de que minimiza la latencia de red innecesaria. (SPDY, 2014)

El camino hacia HTTP 2.0

SPDY fue el catalizador de HTTP 2.0 y a principios del 2012 el Grupo de Trabajo HTTP (HTTP-WG) inicia con las propuestas desde entonces, muchos cambios y mejoras han sido y seguirán siendo hechos a la norma oficial HTTP 2.0. Es menester hablar de la línea de tiempo de HTTP 2.0, donde los hitos oficiales según el HTTP-WG son los siguientes:

- Marzo 2012: convocatoria de propuestas para HTTP 2.0
- Septiembre de 2012: primer borrador de HTTP 2.0
- Julio 2013: primer borrador de implementación de HTTP 2.0
- Abril 2014: grupo de trabajo de última convocatoria de HTTP 2.0
- Noviembre 2014: pone HTTP 2.0 a IESG como un Proyecto de Norma

2.3 Http

El protocolo HTTP es usado desde hace 16 años por ser el estándar que permite navegar por Internet, desarrollado por el “WWW Consortium y la Internet Engineering Task Force (Grupo de Trabajo de Ingeniería de Internet, que desarrolla y promueve estándares voluntarios de Internet, en particular las normas que conforman el conjunto

de protocolos de Internet TCP/ IP)"; define la sintaxis y semántica que utiliza el cliente, servidor, y proxy para comunicarse.

Es un protocolo sin estado por no guardar información de conexiones anteriores, el desarrollo de aplicaciones web necesita mantener un estado para lo cual usa cookies, esto permite a las aplicaciones web instituir la noción de "sesión", y también rastrear usuarios ya que las cookies se almacenan en el cliente por tiempo indeterminado. («Hypertext Transfer Protocol», 2015)

2.3.1. Versiones del protocolo Http

HTTP posee algunas versiones las cuales son compatibles con las anteriores. Cabe indicar que el cliente indica al servidor al principio de la petición la versión que usa, y el servidor usará la misma o una anterior en su respuesta.

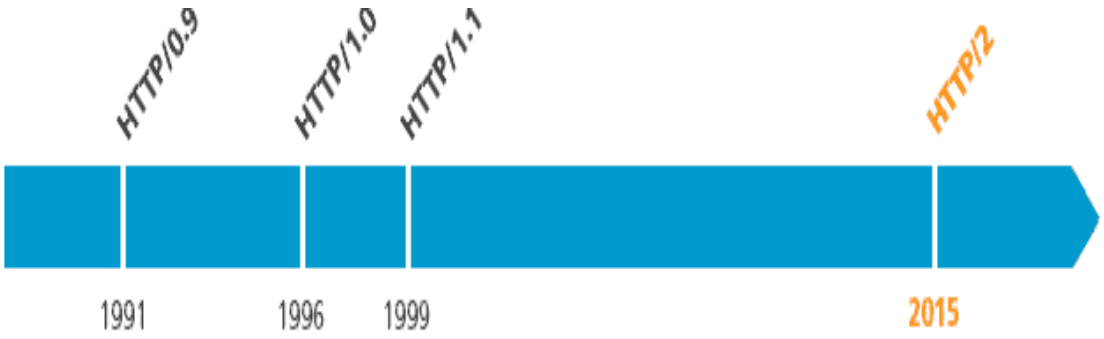


Gráfico 5-2 Historia de HTTP

Fuente: Aegis, http://www.aegis.pe/2015/03/aprobado-http2-gran-cambio-de-internet.html,2015

HTTP 0.9 en 1991

Es una versión obsoleta, soporto sólo un comando GET (pide una representación del recurso especificado como un archivo, por seguridad no debería ser usado ya que transmite información a través de la URI agregando parámetros a la URL), no especifica la versión y tampoco soporta cabeceras. Como no tiene POST el cliente no puede enviarle mucha información al servidor.

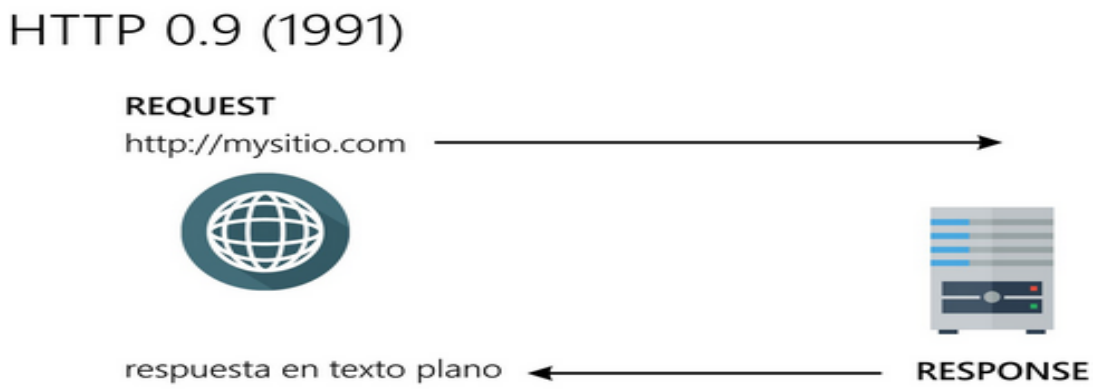


Gráfico 6-2 HTTP 0.9
Fuente: Angulo Jesús, https://somostechies.com/que-es-http2/#.V8cFdaJUUyM, 2016

HTTP 1.0 en mayo de 1996

Especifica su versión en las comunicaciones, y todavía se usa sobre todo en servidores proxy. Tiene 3 comandos: el GET que se usa para recibir información del servidor, HEAD solicita información sobre un objeto o fichero como su tamaño, tipo o fecha de modificación; y el POST que sirve para enviar información al servidor como los datos contenidos en el formulario.

HTTP 1.0 tiene un esquema se autenticación básica y el entity-body es transmitido en texto sin cifrar, se transfiere información confidencial en tres campos de cabecera como: el **server** donde revela la versión haciendo más vulnerable a los ataques, **referer** puede revelar información del usuario por cuanto dichos datos no están separados de la información, y el **from** por dar a conocer la política de seguridad del sitio. («Hypertext Transfer Protocol», 2015)

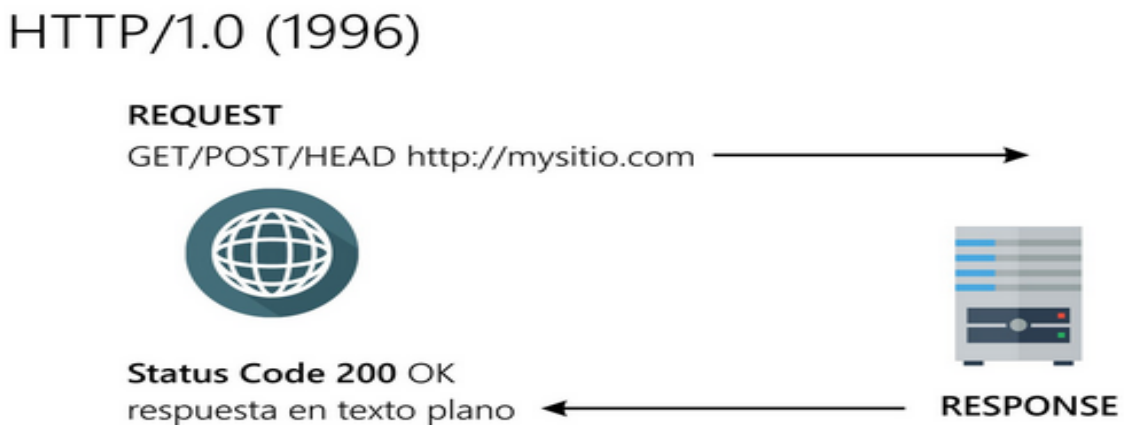


Gráfico 7-2 HTTP 1.0
Fuente: Angulo Jesús, https://somostechies.com/que-es-http2/#.V8cFdaJUUyM, 2016

HTTP 1.1 en junio de 1999

Es la versión actualmente usada, las conexiones persistentes están activadas por defecto y funcionan con los proxies, es decir el cliente envía múltiples peticiones a la vez por la misma conexión TCP y el servidor mantiene abierta dicha conexión para que las siguientes peticiones y respuestas se transmitan por esa conexión.

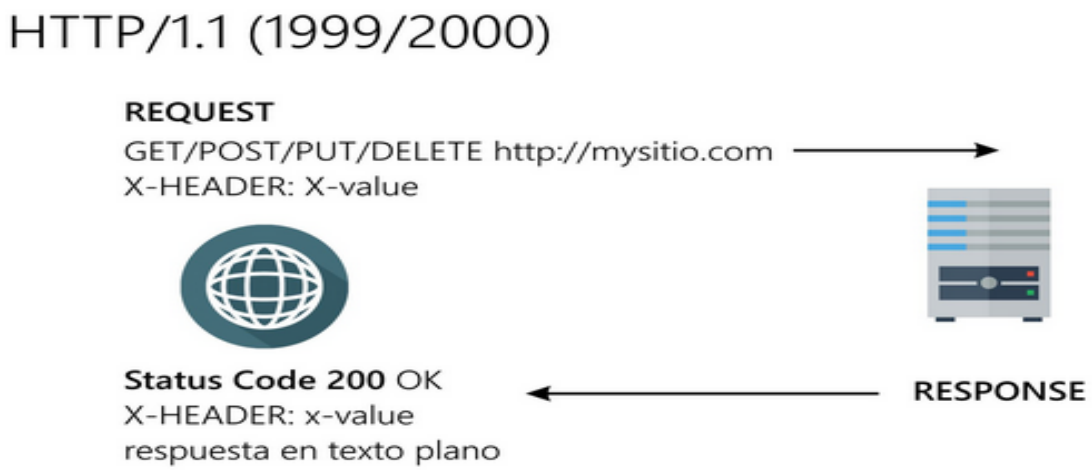


Gráfico 8-2 HTTP 1.1

Fuente: Angulo Jesús, https://somostechies.com/que-es-http2/#.V8cFdaJUUyM, 2016

Características de HTTP 1.1

Entre las características del protocolo HTTP 1.1 están las siguientes:

- HTTP 1.1 es un protocolo textual por lo cual tiene cierta sobrecarga adicional y necesita ser refinado.
- Utiliza una misma conexión para descargar secuencialmente los recursos, cada conexión utiliza recursos de red, y requiere un tiempo adicional de establecimiento que se añade al tiempo total de carga de la página.
- Tiene 8 comandos o métodos o verbos que son: Get, Head, Post, Options indica los métodos HTTP que el servidor soporta para un URL especifico, Put para subir, cargar o realizar un upload de un archivo especifico al servidor, Delete que borra el recurso especifico, Trace se usa con fines de comprobación y diagnóstico, es decir solicita al servidor que envíe toda la data del mensaje de solicitud; y Connect usado para saber si el proxy da acceso a un host bajo condiciones especiales. («Hypertext Transfer Protocol», 2015)

- Transfiere información confidencial en 4 campos de cabecera como: el server, referer, from y via. Los campos de cabecera del cliente y servidor son usados para determinar un agujero de seguridad, en especial la cabecera accept-language revela información privada del usuario.
- No tiene server push, es decir la página HTML se envía al navegador para que analice y decida cuáles son los medios que podría necesitar y pedir al servidor.
- En HTTP 1.1 cada solicitud enviada tiene un pequeño trozo de datos adicionales, conectados a los encabezados HTTP, que describen cómo se comporta un navegador o un servidor.

  En promedio, los navegadores abren entre 4 y 8 conexiones por origen, debido a que muchos sitios utilizan múltiples orígenes esto podría significar que una simple página cargada abre más de 30 conexiones; lo que deja una gran cantidad de datos que recuperar, algo que lleva consumo de tiempo y ancho de banda. (Prieto Francisco, 2011)
- Son propensos a ataques de seguridad basados en DNS "Servidor de Nombre de Dominio", retiene indefinidamente información de autenticación almacenada en cache; el proxy HTTP permite el ataque de hombre en el medio, tiene ataques de denegación de servicio en Proxies, y como no posee encriptación se tiene varios ataques de seguridad y privacidad.
- Tiene un problema llamado "bloqueo de cabeza de línea", donde sólo una solicitud puede ser excepcional en una conexión a la vez.

En la siguiente tabla 1-2 se da a conocer la evolución de las versiones de HTTP antes de empezar con la 2.0:

Tabla 1-2 Versiones de HTTP

HTTP 0.9	HTTP 1.0	HTTP 1.1
En 1991	En mayo de 1996	En junio de 1999
No especifica su versión en las comunicaciones	Si especifica su versión	Si especifica su versión
Comando Get	Comandos Get, Head y Post	Comandos Get, Head, Post, Options, Put, Delete, Trace y Connect
Transmite información a través de la URI	Esquema de autenticación básica y el entity-body transmitido en texto sin cifrar	Protocolo textual, propensos a ataques de denegación de servicios, de DNS, etc.
No soporta cabeceras	Cabeceras: Server, Referer y From que transfieren información confidencial	Cabeceras: Server, Referer, From y Via que transfieren información confidencial

Realizado por: León Isabel, 2016

2.4 Http 2.0

HTTP 2.0 es el mayor cambio en Internet desde 1999, aprobado por el comité IESG de IETF, realmente se trata de una mejora sustancial que se notará cuando se navegue por Internet. (Valero Claudio, 2015)

Éste estándar aporta mejoras a la Web como la carga más rápida de páginas, conexiones activas durante más tiempo o mejor manejo en casos de muchos accesos simultáneos a una página, utiliza los mismos parámetros que la versión actual 1.1 con la que están familiarizados los desarrolladores, pero añade más funciones.

La que más se destaca es que una página web no se bloquea al recibir muchas conexiones al mismo tiempo, supone una carga menor para los servidores y la seguridad es superior. (AEGIS, «Aprobado HTTP/2», 2014)

HTTP 2.0 permite un uso más eficiente de los recursos de la red y una percepción reducida de latencia mediante la introducción de compresión del campo de cabecera y permitiendo múltiples intercambios simultáneos en la misma conexión.

2.4.1. Características de Http 2.0

En el documento **draft-ietf-httpbis-http2-16** de (Belshe, Thomson, & Peon, 2014), se manifiesta que HTTP 2.0 ofrece un transporte optimizado para la semántica HTTP, que apoya todas las características básicas de 1.1, y pretende ser más eficiente de varias maneras.

Los métodos HTTP, códigos de estado, semántica, esquemas de URI "http" y "https" usados por HTTP 1.1 son los mismos de igual forma se mantiene el puerto por defecto 80 para http y 443 para https.

La unidad básica en HTTP 2.0 es un frame el cual tiene un propósito diferente, por ejemplo, los frames de datos y cabeceras forman la base de las peticiones y respuestas HTTP; y otros tipos de frames como: settings, window_update y push_promise son utilizados en apoyo a otras características de HTTP 2.0.

Según el líder del proyecto HTTP 2.0, Mark Nottingham: "HTTP 2.0 utiliza la multiplexación para permitir que muchos mensajes se intercalen entre sí en una

conexión al mismo tiempo, por lo que una respuesta grande (o una que toma mucho tiempo al servidor que piense) no bloquea a otros".

Multiplexación de solicitudes se consigue haciendo que cada intercambio de HTTP solicitud-respuesta se asocie con su propio stream (flujo bidireccional de frames a través de un canal virtual dentro de la conexión HTTP 2.0). Los Streams son en gran parte independientes entre sí, por lo que una petición o respuesta bloqueada no impide el progreso de otros streams.

El control de flujo garantiza que sólo los datos a ser utilizados por un receptor sean transmitidos; mientras que la priorización asegura que los recursos limitados se pueden dirigir primero a las stream más importantes.

Añade un nuevo modo de interacción, esto es un servidor push que envía datos anticipándose a lo que el cliente necesitará, el servidor hace esto mediante la síntesis de una solicitud para lo cual usa un frame push_promise, y es capaz de enviar una respuesta en un stream separado. (Belshe, Thomson, & Peon, 2014),

Debido a que los campos de cabecera HTTP utilizados en una conexión pueden contener grandes cantidades de datos redundantes, los frames se comprimen. Esto es totalmente ventajoso porque permite que muchas peticiones sean comprimidas en un paquete TCP. («¿Qué es el HTTP 2.0 y qué cambios traerá?», 2014)

HTTP 2.0 se puede probar en navegadores como Mozilla Firefox y Google Chrome utilizando el identificador "h2-14" según indican las FAQ oficiales. Google adopta éste protocolo en lugar del suyo propio SPDY, y las próximas versiones añadirán soporte nativo para HTTP 2.0. Hay varios proveedores y sitios que han comenzado su implementación como: Akamai, Google y Twitter. («Aprobado HTTP/2», 2014)

A continuación, en la tabla 2-2 se da a conocer la comparativa de las principales características entre HTTP 1.1 y HTTP 2.0:

Tabla 2-2 Características entre HTTP 1.1 y HTTP 2.0

Indicador	HTPP 1.1	HTTP 2.0
Protocolo	Textual, complicado encontrar el comienzo y final de cada frame	Binario, facilidad para encontrar el frame, y menos propensos a errores
Conexión	Múltiples TCP simultáneas para descargar los elementos de la web	Única que ofrece múltiples solicitudes y respuestas en paralelo
Información Redundante	Envío de datos repetidos en la misma conexión por lo tanto consume más recursos y tiene mayor latencia	La elimina en la misma conexión
Semántica	Métodos HTTP, códigos de estado, y semántica.	Semántica igual que HTTP 1.1
Multiplexación	El navegador envía una petición y espera la respuesta del servidor para poder enviar la siguiente solicitud.	La multiplexación permite enviar y recibir varios mensajes al mismo tiempo optimizando la comunicación.
Compresión de cabeceras	No tiene	Si posee
Server Push	No soporta	Si soporta siendo capaz de enviar información al usuario antes de que éste la solicite para que la información esté disponible de forma inmediata.
Prioridad	No hay prioridad de flujos	Tiene prioridad en las tramas
Encriptación	No soporta encriptación	El uso de la encriptación TLS es opcional

Realizado por: León Isabel, 2016

2.4.2. Ventajas de Http 2.0

Según HTTPbis Working Group, la sección de IETF encargada de diseñar la especificación de HTTP 2.0 da a conocer los cambios que aporta y que sirven para mejorar la eficiencia de las comunicaciones web de las cuales se puede mencionar las siguientes:

- Utiliza una sola conexión para ofrecer múltiples solicitudes y respuestas en paralelo.
- Compresión de las cabeceras HTTP reduce de manera sensible el volumen de datos transmitidos y es más rápido, pues todas las cabeceras se empaquetan en

un bloque comprimido para ser enviado como una unidad y cuando termina la transmisión el bloque de cabecera se decodifica.

- El envío de cookies sólo tiene lugar en caso de que su valor haya cambiado con respecto a su último envío.
- Multiplexado de peticiones HTTP sobre una misma conexión TCP.
- Soporte para 'server push', un servidor pueda forzar el envío de datos al cliente sin que éste los haya solicitado, ahorrado así una conexión y mejora el rendimiento.
- El ser binario y no textual, implica ser más compacto.
- Intercala múltiples peticiones y respuestas en paralelo sin bloquear ninguna de ellas.
- Entrega tiempos de carga más bajos mediante la eliminación de la latencia innecesaria.
- La capa framing binario resuelve "head-of-line blocking" bloqueo de cabeza de línea[2] problema de HTTP 1.1, y elimina la necesidad de múltiples conexiones para permitir el procesamiento y la entrega de solicitudes/respuestas en paralelo.
- La capacidad de descomponer un mensaje HTTP en frames independientes, intercalar, y luego volver a unirlos en el otro extremo.

Por lo tanto, con la reducción del número de conexiones TCP necesarios, HTTP 2.0 también reduce los costes de CPU y memoria tanto para el cliente y servidor, teniendo aplicaciones más baratas de implementar, rápidas y sencillas.

Todo esto se traduce en menor consumo de ancho de banda, menor sobrecarga de la información, mayor rapidez y rendimiento de carga del contenido web. («¿Qué es el HTTP 2.0 y qué cambios traerá?», 2014)

HTTP 2.0 propone el cifrado por defecto del tráfico de la "Internet abierta", con tres posibles soluciones:

A. Cifrado oportunista por http:// URI sin autenticación del servidor: conocido como "TLS relajado", según el documento draft-nottingham-http2-encryption-01.

[2] "El servidor envía sus respuestas en el mismo orden en que recibe las solicitudes tomando en cuenta que solo es una conexión a la vez en HTTP 1.1 y por usar TCP los paquetes son entregados en orden y retransmitidos si se pierde uno de ellos". (networking - SPDY Head of Line blocking - Stack Overflow. Disponible en: http://stackoverflow.com/questions/25221954/spdy-head-of-line-blocking. 2014

B. Cifrado oportunista por http:// URI con autenticación del servidor: el mismo mecanismo que el anterior pero no “relajado” y con algún tipo de protección adicional.
C. Utilizar HTTP 2.0 sólo con https:// URI en Internet “abierta”: no requerirá cifrado en redes privadas o locales. http:// URI seguiría utilizando HTTP 1.x usando cifrado como hasta ahora.

Según comenta Mark Nottingham lider del proyecto HTTP 2.0, la solución (C) es preferible a la (B), por ser más sencillo, no necesita implementar un mecanismo nuevo y HSTS se utiliza como protección adicional.

HSTS “Seguridad de transporte HTTP estricta”, es un protocolo de normas de IETF y se especifica en el RFC 6797. La política HSTS es comunicada por el servidor al cliente a través de un campo de la cabecera HTTP de respuesta denominado "Strict-Transport-Security"; y especifica un período de tiempo durante el cual el cliente deberá acceder al servidor sólo en forma segura. («HTTP Strict Transport Security», 2015).

La opción (A) es la más sencilla de aplicar para los sitios web, ya que no requiere certificado digital reconocido por los principales navegadores, certificado que tiene un coste; el inconveniente de esta opción es que se puede suplantar la identidad de un sitio con un certificado falso.

La opción (B) es la más segura ya que cifra el tráfico por defecto con autenticación, es más cara de implementar siendo el coste del certificado un obstáculo que puede impedir la adopción masiva de esta solución.

La opción (C) está en un término medio entre las dos anteriores, la cual implementa el protocolo HTTP 2.0 para conexiones seguras. (C) tiene ventaja sobre (A) ya que proporciona una mayor protección contra ataques activos.

En cuanto a la seguridad el protocolo HTTP 2.0 tiene algunas consideraciones que son:

- Obliga el uso de SSL con cifrado SSL/TLS, es decir se usará la criptografía proporcionando autenticación y privacidad.
- Usa el protocolo TLS 1.2 o superior, y utilizará la renegociación para proteger la confidencialidad de las credenciales del cliente en un handshake.

- Emplea conexiones persistentes que implica tener una conexión TCP donde el cliente puede enviar múltiples peticiones a la vez, estas conexiones se mantienen abiertas las cuales son reutilizadas si existe una nueva petición evitando múltiples conexiones con la misma configuración.
- La Reutilización de la conexión para recursos http si el host tiene la misma dirección IP y en recursos https depende si el host tiene un certificado valido en la URI.
- Se basa en la autoridad del servidor de HTTP 1.1 siendo autenticado para https.
- Protege de ataques a través de protocolo al usar un handshake TLS con un ALPN "protocolo de negociación de capa de aplicación".
- No son almacenadas en cache las respuestas push del servidor que no están autorizados.
- Para no tener una denegación de servicios se debe utilizar bien la compresión de cabecera, control de flujo, el frame settings que pueden causar gasto innecesario de recursos.
- Brinda compresión para la cabecera y el entity bodies.
- Usa el padding para ocultar el tamaño exacto del frame y mitigar ataques dentro de HTTP.

2.4.3. Desventajas de Http 2.0

Las desventajas que tiene el protocolo HTTP 2.0 están las siguientes:

- ✓ El uso del padding de forma redundante es contraproducente.
- ✓ HTTP 2.0 permite un total de 2^{31} intercambios de stream por conexión los mismos que deben limitarse porque puede darse una denegación de servicios.
- ✓ Vulnerabilidad por el uso de server push, donde el atacante aprovecharía las múltiples respuestas que realiza el servidor a una única solicitud, para acceder a contenidos web que fueron de uso público y ahora son de uso privado.
- ✓ A pesar de eliminar el head-of-line blocking de HTTP, aún está el head-of-line blocking a nivel de TCP.
- ✓ Los efectos del ancho de banda - retardo pueden limitar el rendimiento de la conexión si la ventana TCP está deshabilitada.
- ✓ Cuando se produce la pérdida de paquetes, el tamaño de la ventana de TCP se reduce, lo que implica que disminuye también el rendimiento máximo de toda la conexión.

Estas desventajas pueden afectar negativamente en el rendimiento de latencia de una conexión HTTP 2.0. Sin embargo, a pesar de estas limitaciones el uso de una única conexión TCP sigue siendo la mejor estrategia de despliegue de HTTP 2.0.

2.4.4. Identificadores de Http 2.0

El protocolo definido tiene dos identificadores:

- La cadena "h2" identifica que HTTP 2.0 utiliza TLS “Transport Layer Security”. Este identificador es usado en la capa de aplicación TLS extensión de la negociación del protocolo ALPN, y donde HTTP 2.0 sobre TLS se identifica. La cadena "h2" se serializa en un identificador de protocolo ALPN como la secuencia de dos octetos: 0x68 y 0x32.
- La cadena "h2c" identifica que HTTP 2.0 se ejecuta a través de TCP sin cifrar. Este identificador se utiliza en el campo de cabecera actualizado de HTTP 1.1, y donde HTTP 2.0 a través de TCP se identifica.

La negociación "h2" o "h2c" implica el uso del transporte, la seguridad, framing y semántica del mensaje. (Belshe, Thomson, & Peon, 2014)

2.4.5. Soporte de los navegadores para Http 2.0

Entre los navegadores que soportan HTTP2.0 están los siguientes:

- ❖ Chrome Canary, debe permitir SPDY 4 flag, escribiendo "chrome: // flags" en la barra de direcciones y la búsqueda de "HTTP 2.0" o "SPDY 4"
- ❖ Firefox
- ❖ Internet Explorer (en desarrollo)
- ❖ Safari (en desarrollo)

2.4.6. Http 2.0 para los URL "http"

Un cliente HTTP 2.0 debe saber si el servidor y los intermediarios soportan HTTP 2.0 cuando se inicia una nueva conexión. Hay tres casos a considerar que son:

1. Inicio de una nueva conexión HTTPS mediante TLS y ALPN
2. Inicio de una nueva conexión HTTP con el conocimiento previo
3. Inicio de una nueva conexión HTTP sin conocimiento previo

El establecer una conexión HTTP 2.0 a través de un canal normal no encriptado requerirá un poco más de trabajo. Tanto HTTP 1.1 y 2.0 se ejecutan en el puerto 80, por falta de cualquier información sobre el soporte de servidor de HTTP 2.0, el cliente tendrá que utilizar el mecanismo de HTTP para negociar el protocolo adecuado, para ello incluye el campo Upgrade como se puede visualizar:

```
GET /page HTTP/1.1
Host: server.example.com
Connection: Upgrade, HTTP2-Settings
Upgrade: HTTP/2.0 o h2c  //HTTP 1.1 inicia petición con cabecera
                              actualizada HTTP 2.0
HTTP2-Settings:(SETTINGS payload) //Base64 URL codificación
                          de HTTP 2.0 (SETTINGS carga útil)
```

Las solicitudes que contengan un cuerpo de entidad deben ser enviadas en su totalidad antes de que el cliente envíe frames HTTP 2.0, esto significa que una solicitud grande puede bloquear el uso de la conexión hasta que sea enviada por completo.

Una solicitud de OPTIONS se puede utilizar para realizar la actualización a HTTP 2.0, a costa de una ida y vuelta adicional.

Un ***servidor que no admite HTTP 2.0*** puede responder a la solicitud, aunque el campo de cabecera Upgrade esté ausente como se puede apreciar a continuación:

```
HTTP/1.1 200 OK  // servidor declina actualización,
                  devuelve la respuesta a través de HTTP 1.1
Content-length: 243
Content-type: text/html

(... HTTP 1.1 response ...)
```

Un ***servidor que soporta HTTP 2.0*** acepta la actualización con una respuesta 101 “protocolo de conmutación”. Después de la línea en blanco que termina la respuesta 101, el servidor puede comenzar a enviar frames HTTP 2.0, los cuales deben incluir una respuesta a la solicitud que inicia la actualización. Por ejemplo:

```
HTTP/1.1 101 Switching Protocols //El servidor acepta la
                                   actualización HTTP 2.0
Connection: Upgrade
Upgrade: HTTP/2.0 o h2c
(... HTTP 2.0 response ...)
```

El primer frame HTTP 2.0 enviado por el servidor deberá ser un frame settings como su prefacio de conexión. Al recibir la respuesta 101, el cliente deberá enviar un prefacio de conexión que incluye un frame settings.

A la petición HTTP 1.1 enviada antes de la actualización se le asigna un stream con identificador 1 y valor de prioridad por defecto. Dicha petición es completada como una solicitud HTTP 1.1 para luego comenzar con la conexión HTTP 2.0 y el stream 1 se utiliza para responder.

Si el servidor no admite HTTP 2.0 entonces se puede responder con la respuesta HTTP 1.1. Se confirma la actualización a 2.0 mediante la devolución del 101 Switching Protocols en respuesta a HTTP 1.1, luego cambiar a HTTP 2.0 y devolver la respuesta usando éste protocolo; en cualquier caso, no se incurre en ida y vuelta extra.

Finalmente, el cliente puede obtener información sobre el soporte HTTP 2.0 a través de otros medios por ejemplo registro DNS, configuración manual, y así sucesivamente, en lugar de tener que depender de la actualización de workflow. Se puede enviar frames HTTP 2.0 sobre un canal sin cifrar, donde si la conexión falla el cliente deberá actualizar el workflow o cambiar a un túnel TLS con la negociación ALPN.

2.4.7. Http 2.0 para los URL "https"

Un cliente que realiza una solicitud a un URI "https" utiliza TLS con la negociación del protocolo de capa de aplicación ALPN.

HTTP 2.0 sobre TLS utiliza el identificador de protocolo "h2", donde el "h2c" no debe ser enviado por un cliente o seleccionado por un servidor porque no usa TLS.

Una vez que la negociación TLS es completa, el cliente y el servidor deben enviar un prefacio de conexión.

2.4.8. Http 2.0 con conocimiento previo

Un cliente puede saber si un determinado servidor soporta HTTP 2.0, con el “ALT-SVC” que describe un mecanismo para determinar esta capacidad.

Un cliente deberá enviar primero el prefacio de conexión y luego enviar frames HTTP 2.0 a un servidor, mientras que los servidores pueden identificar éstas conexiones por la presencia del prefacio de conexión.

Esto sólo afecta al establecimiento de la conexión de HTTP 2.0 sobre TCP cleartext, así las implementaciones que soportan HTTP 2.0 sobre TLS deben utilizar el protocolo negociación de TLS. Del mismo modo, el servidor deberá enviar un prefacio de conexión.

2.4.9. Servicio alterno "ALT-SVC"

Un servicio alternativo no reemplaza o cambia el origen de cualquier recurso, y se identifica por la siguiente combinación:

- Protocolo de negociación de capa de aplicación (ALPN)
- Un host
- Un puerto

Un servicio alternativo se puede utilizar para interactuar con los recursos en un servidor de origen usando una configuración de protocolo diferente. En HTTP 2.0 el frame ALTSVC anuncia la disponibilidad de un servicio alternativo de un cliente.

Los servicios alternativos son información de enrutamiento alternativa que se puede utilizar para llegar al origen de la misma manera que el DNS CNAME o registros SRV que definen la información de enrutamiento en el nivel de resolución de nombres.

Los clientes que usan un servicio alternativo pueden cambiar el host, el puerto y el protocolo que están utilizando para buscar recursos, pero estos cambios no deberán ser propagados a la aplicación que está utilizando HTTP; desde ese punto de vista, el URI está accediendo y toda la información derivada de ella (esquema, host, puerto) son las mismos que antes.

Cuando TLS está en uso el servicio alternativo necesitará presentar un certificado para un nombre de host del origen, no el de la alternativa. Del mismo modo, el campo de cabecera Host todavía se deriva del origen no del servicio alternativo.

Autenticación del Host

Los clientes no deben utilizar servicios alternativos con un host diferente del original sin autenticación del servidor; una forma de lograr esto es usando TLS con un certificado.

Por ejemplo, si el host de origen es "www.example.com" y una alternativa es "other.example.com" con el protocolo "h2", el certificado es válido para "www.example.com", y el cliente puede utilizar la alternativa.

Sin embargo, si "other.example.com" es ofrecido con el protocolo "h2c", el cliente no puede utilizarlo porque no existe un mecanismo en ese protocolo para establecer autenticación del servidor.

Exigir Name Server Indication

Un cliente sólo debe utilizar un servicio alternativo basado en TLS, si el cliente es compatible con TLS Server Name Indication “SNI” soportará la conservación de direcciones IP en el servicio alternativo host.

Hay que tener en cuenta que la información SNI proporcionada en TLS por el cliente será el de origen no la alternativa. (McManus, Mnot, & Reschke, 2015)

2.4.10. Prefacio de conexión

En HTTP 2.0 el cliente y el servidor enviarán un prefacio de conexión diferente. El prefacio de conexión del cliente comienza con una secuencia de 24 octetos, que en notación hexadecimal son:

```
0x505249202a20485454502f322e300d0a0d0a534d0d0a0d0a
```

La secuencia (La cadena "`PRI * HTTP / 2.0 \ r \ n \ r \ NSM \ r \ n \ r \ n`") debe ser seguida por un frame Settings que puede estar vacía. El cliente envía el prefacio de conexión después de recibir la respuesta 101 “Protocolo de conmutación que indica una actualización correcta”, o con los primeros octetos de datos de aplicación de una conexión TLS.

El prefacio de conexión del servidor consiste de un frame Settings vacío que es el primer frame del servidor enviado en la conexión HTTP 2.0. Los frame Settings recibidos de un compañero como parte de la conexión de prefacio deben ser reconocidos después del envío de dicho prefacio de conexión.

Para evitar latencia innecesaria, los clientes están autorizados a enviar frames adicionales al servidor después de enviar un prefacio de conexión, sin esperar a recibir el prefacio de conexión del servidor.

Los clientes y servidores tratarán un prefacio de conexión no válido como error de conexión de tipo PROTOCOL_ERROR. Un frame goAWAY puede omitirse en este caso, ya que un prefacio inválido indica que el interlocutor no está utilizando HTTP 2.0. (Belshe, Thomson, & Peon, 2014)

2.5 Capa de framing binario en Http 2.0

Las mejoras de rendimiento de HTTP 2.0 es la nueva capa de framing binario de longitud-prefijo, que dicta cómo se encapsulan y se transfieren los mensajes HTTP entre el cliente y el servidor ofreciendo una representación más compacta, más fácil y más eficiente para procesar en código.

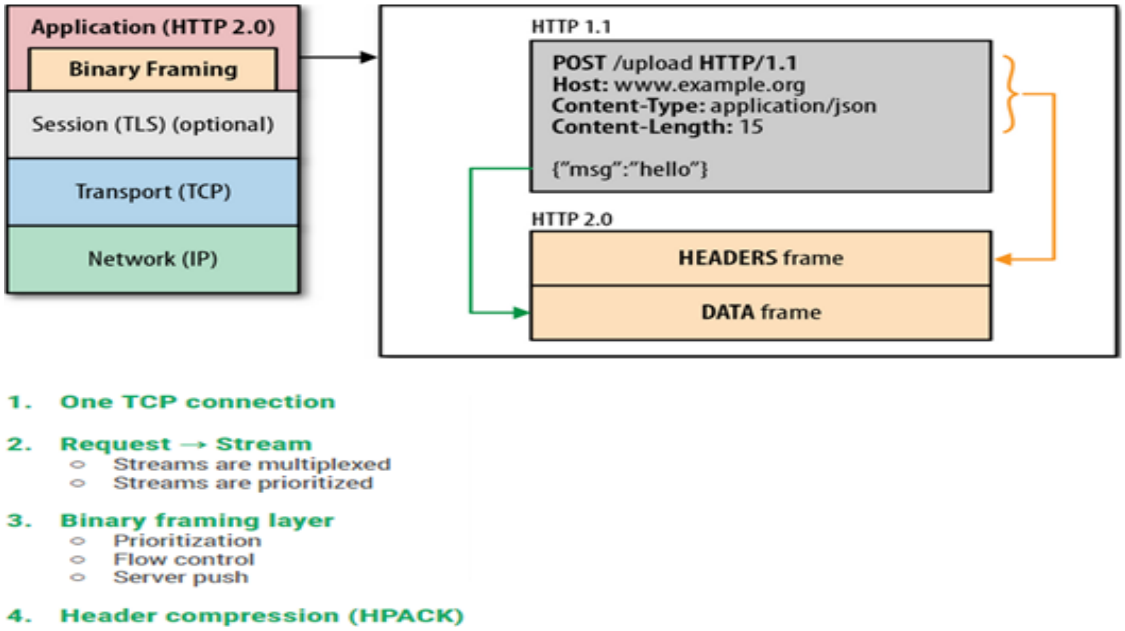


Gráfico 9-2 Capa de framing binario HTTP 2.0

Fuente: Brian Jackson, https://www.keycdn.com/blog/keycdn-http2-support/

La capa se refiere a una opción de diseño que introduce un nuevo mecanismo entre la interfaz socket y la API HTTP expuesto a las aplicaciones, donde la semántica HTTP, los verbs, los métodos y los encabezados no se ven afectados, pero la forma en que se codifican mientras transitan es lo que es diferente.

La comunicación HTTP 2.0 se divide en mensajes pequeños y frames donde cada uno está codificado en formato binario. Tanto el cliente como el servidor deben utilizar HTTP 2.0 para entenderse.

HTTPS es un ejemplo de framing binario, donde todos los mensajes HTTP son transparentemente codificados y decodificados en "Protocolo de Registro TLS", lo cual permite la comunicación segura entre el cliente y el servidor, sin necesidad de ninguna modificación de las aplicaciones. HTTP 2.0 funciona de una manera similar.

2.5.1. Streams, messages, y frames

Se da a conocer una nueva terminología HTTP 2.0 la misma que se describe a continuación:

- stream: es una secuencia independiente y bidireccional de frames intercambiados entre el cliente y el servidor en una conexión HTTP 2.0.
- message: una secuencia completa de frame que se correlacionan con un mensaje lógico.
- frame: la unidad más pequeña de la comunicación en HTTP 2.0.

Las comunicaciones HTTP 2.0 van dentro de una conexión que puede llevar cualquier número de streams bidireccionales. A su vez cada stream se comunica en los mensajes los cuales puede ser intercalados y se juntan a través del identificador de stream incrustado en la cabecera de cada frame individual. Todos los frames HTTP 2.0 utilizan codificación binaria, y los datos de cabecera se comprimen.

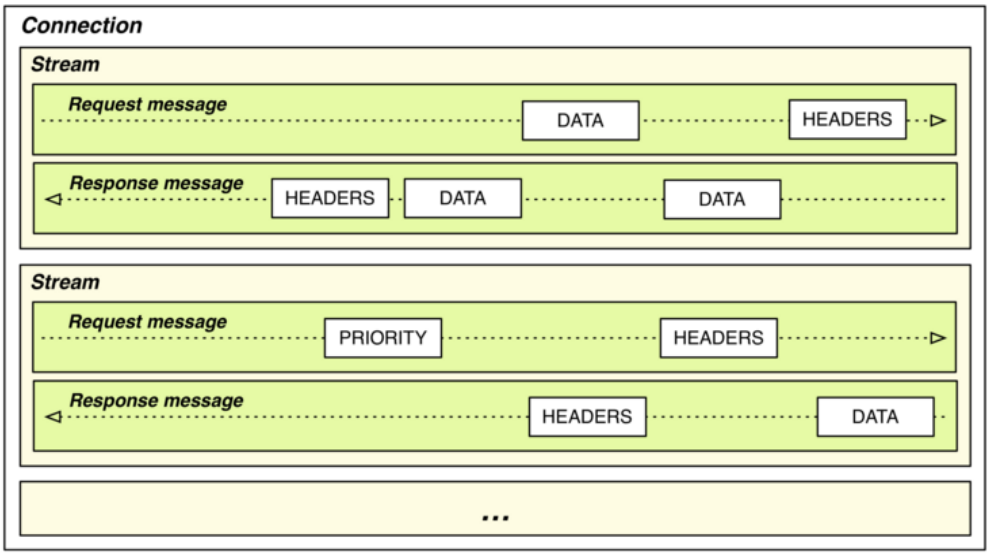


Gráfico 10-2 HTTP 2.0 streams, messages, and frames

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

Cabe indicar que las características que tiene stream son las siguientes:

- Una conexión HTTP 2.0 puede contener múltiples streams abiertos.
- Stream pueden establecerse y utilizarse de manera unilateral o compartida por el cliente o servidor.
- Los endpoint pueden cerrar los streams.
- El orden en que se debe enviar frames en un stream es significativo, ya que los destinatarios procesan frames en el orden en que se reciben.
- Los streams se identifican con un entero de 31 bits, si son iniciados por un cliente se usa identificadores impares y el servidor usa pares.
- Un stream (0x0) se utiliza para los mensajes de control de conexión y no se puede utilizar para establecer un nuevo stream.
- Las peticiones HTTP 1.1 que se actualizan a 2.0 son respondidas con (0x1).
- Los identificadores de stream no pueden ser reutilizados.

2.5.2. Formato del frame

Una vez que se establece una conexión HTTP 2.0 el cliente y el servidor se comunican por el intercambio de frames; los cuales tienen una cabecera de 9 bytes, la longitud del frame, el tipo, un campo de bits de flags, y un identificador de flujo de 31-bit.

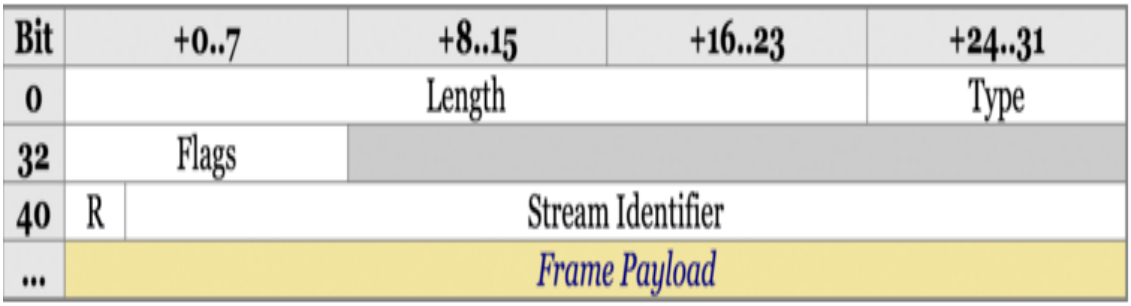


Gráfico 11-2 Frame de cabecera común 9 bytes

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

A continuación, una breve explicación del gráfico 11-2:

Length: la longitud del frame payload se expresa como un entero sin signo de 24 bits. Los valores superiores a 2^{14} (16.384) no deberán ser enviados al receptor a menos que se fije un valor mayor para SETTINGS_MAX_FRAME_SIZE. Los 9 octetos de la cabecera del frame no se incluyen en este valor.

Type: es de 8 bits, determina el formato y la semántica del frame. Las implementaciones deben ignorar y descartar cualquier frame que tiene un tipo desconocido.

Flags: un campo de 8 bits reservado para flags (banderas) booleanas específicas.

- Flags asignan una semántica específica para un tipo de frame indicado.
- Si las flags no tienen semántica definida para un tipo de frame particular, deben ser ignoradas y se debe dejar unset (0) al enviar.

R: campo de 1 bit reservado, la semántica de este bit no está definido y el bit debe permanecer unset (0) al enviar y deben ser ignorado cuando se recibe.

Stream Identifier: un identificador de stream de 31 bits. El valor 0 está reservado para frames que están asociados con la conexión como un todo en contraposición para un stream individual.

La estructura y el contenido del frame payload son totalmente dependientes del tipo de frame.

Tabla 3-2 Tipos de HTTP 2.0

CAMPO TIPO	DESCRIPCIÓN
DATA	Utiliza para transportar los cuerpos de mensajes de HTTP.
HEADERS	Se utiliza para comunicar campos de cabecera adicionales por un stream.
PRIORITY	Para asignar o reasignar prioridades de recurso referenciado.
RST_STREAM	Para transmitir una señal anormal de un stream.
SETTINGS	Utilizado por el cliente y servidor al inicio de una conexión y también puede ser enviado en cualquier momento durante el tiempo de vida de la conexión.
PUSH_PROMISE	Utilizado para transmitir la creación de un stream y servir un recurso referenciado.
PING	Para medir el tiempo de ida y vuelta, y para realizar comprobaciones de la vivacidad.
GOAWAY	Informa que se detenga la creación del stream de la conexión actual.
WINDOW_UPDATE	Para implementar el control de flujo de un per-stream o per-connection.
CONTINUATION	Usado para continuar con una secuencia de fragmentos de bloques de cabecera.

Realizado por: León Isabel, 2016

Con HTTP 2.0 no se tiene que preocupar por la semántica de control de flujo, control de errores, finalización de la conexión, y muchos otros detalles.

2.5.3. Tamaño del frame

El tamaño máximo del frame payload (carga útil) está en setting SETTINGS_MAX_FRAME_SIZE, éste valor esta entre 2^{14} (16.384) y 2^{24}-1(16.777,215) octetos.

Todas las implementaciones deben ser capaces de recibir y procesar frames superiores a 2^{14} octetos de longitud, y 9 bits de la cabecera de frame.

Un endpoint debe enviar un error FRAME_SIZE_ERROR si el frame excede el tamaño y si no existe ningún límite definido para el tipo de frame, por ello es tratado como error de conexión.

Los endpoints no están obligados a utilizar todo el espacio disponible en un frame, la capacidad de respuesta se mejora mediante el uso de frames pequeños. El envío de grandes frames da lugar a retrasos de frames time-sensitive (como RST_STREAM, WINDOW_UPDATE, o PRIORITY) que afectan el rendimiento.

2.5.4. Inicio de un nuevo stream

Antes de que los datos de aplicación se puedan enviar, un nuevo stream es creado y la solicitud de metadatos deberá enviar: stream opcionales, flags opcionales, y las cabeceras de petición HTTP codificados con HPACK. El cliente inicia este proceso mediante el envío de frame HEADERS con todo lo anterior.

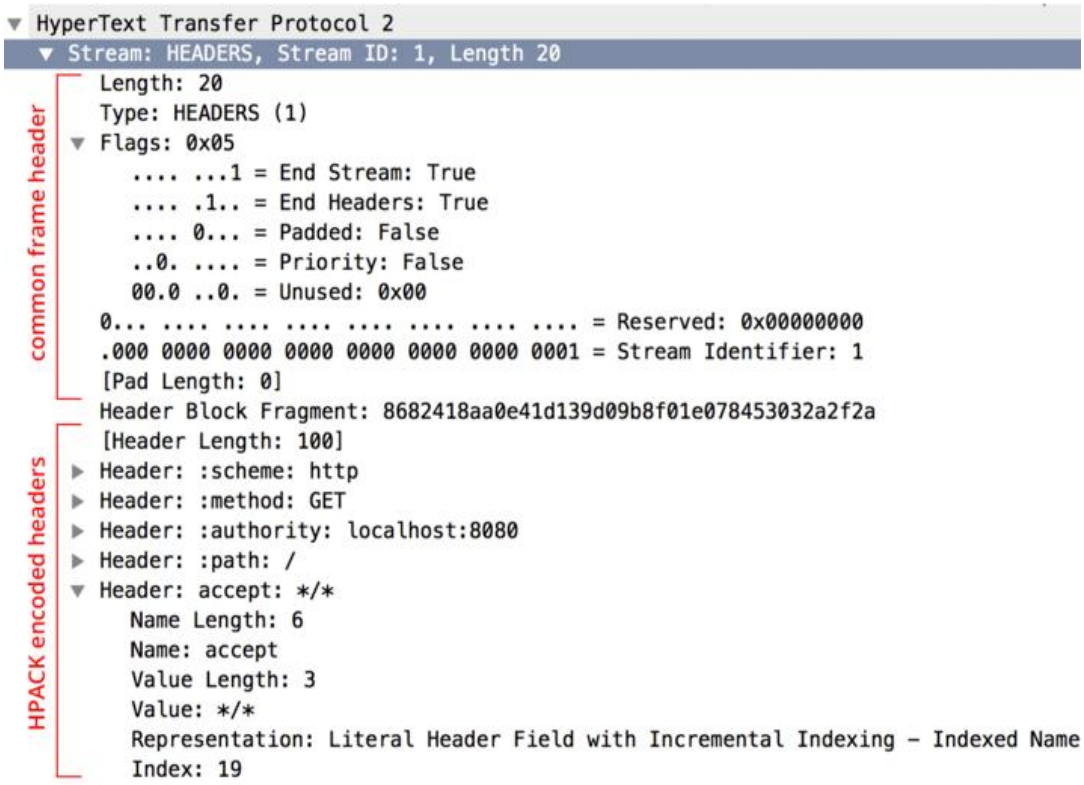


Gráfico 12-2 Decodificación del Frame Headers en Wireshark

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

El frame HEADERS se utiliza para declarar y comunicar metadatos de la nueva solicitud. Como la entrega de metadatos es independiente de los datos de aplicación, tanto el cliente como el servidor administran con diferentes prioridades, por ejemplo: el tráfico de control se puede entregar con mayor prioridad y el flujo de control se aplica sólo a los frames data.

Server inicia streams via PUSH_PROMISE

HTTP 2.0 permite tanto al cliente y servidor iniciar nuevos streams. En el caso de un stream iniciado por el servidor, un frame PUSH_PROMISE se utiliza para declarar y comunicar las cabeceras de respuesta codificadas con HPACK.

El formato del frame es similar a las cabeceras, excepto que se omite la dependencia y peso de stream opcionales, ya que el servidor tiene pleno control de cómo se entrega los datos prometidos.

Para eliminar las colisiones de stream ID los stream iniciados por el cliente tienen stream ID impares, y los stream iniciados por el servidor tienen stream ID pares. Por lo tanto, como el stream ID es 1 del gráfico 13-2 es stream iniciado por el cliente.

En síntesis, del inicio de un nuevo stream se puede decir que:

1. El cliente inicia una nueva solicitud mediante el envío de frame HEADERS que incluye la cabecera común con un nuevo ID de stream, un valor opcional de prioridad de 31 bits, y un conjunto de cabeceras HTTP de pares de key-value dentro de su carga útil.
2. El servidor inicia un push stream mediante un frame PUSH_PROMISE, que es idéntica a un frame HEADERS, excepto que lleva un "promised stream ID" extra, en lugar de un valor de prioridad.

2.5.5. Envío de datos de aplicación

Una vez que se crea el nuevo stream y las cabeceras de HTTP se envían, los DATA frames son usados para enviar la carga útil de la aplicación la misma que puede dividirse entre múltiples DATA frames para permitir la multiplexación eficiente, con el último frame se indica el final del mensaje conmutando la flag END_STREAM en la cabecera del frame.

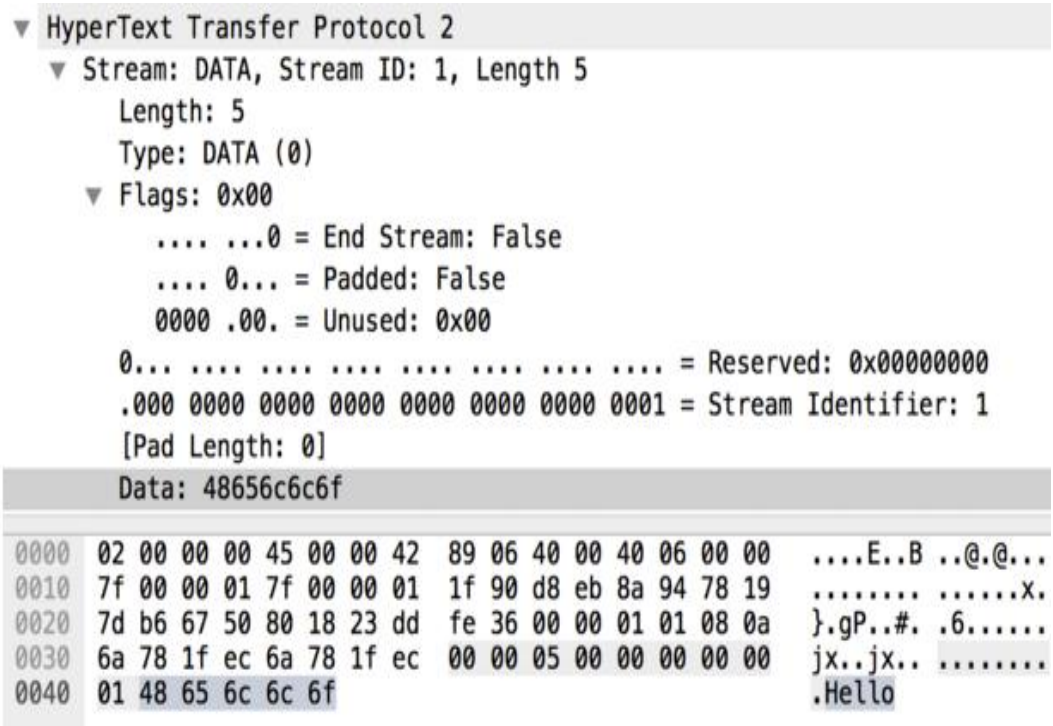


Gráfico 13-2 Data frame

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

La flag "End Stream" está establecido en "false" en el gráfico de arriba, lo que indica que el cliente no ha terminado de transmitir la carga útil de la solicitud; y más DATA frame están llegando.

2.5.6. Solicitud y respuesta multiplexing

Con el protocolo HTTP 1.x, si el cliente quiere hacer múltiples solicitudes paralelas para mejorar el rendimiento debe utilizar múltiples conexiones TCP, teniendo en cuenta que sólo una respuesta se puede entregar a la vez por conexión, existiendo por ende el bloqueo de cabeza de línea y el uso ineficiente de la conexión TCP subyacente.

La capa framing binario en HTTP 2.0 elimina estas limitaciones y permite full solicitudes y multiplexación de respuestas, al permitir que el cliente y el servidor descompongan un mensaje HTTP en frames independientes, intercalen, y luego vuelvan a montar en el otro extremo.

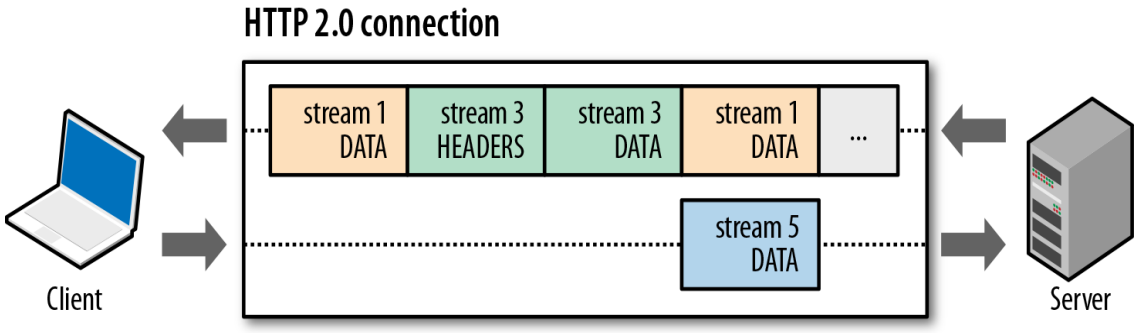


Gráfico 14-2 HTTP 2.0 solicitud y respuesta multiplexing

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

Como se puede visualizar en el gráfico de arriba se captura múltiples stream dentro de la misma conexión, el cliente está transmitiendo un frame de datos (stream 5) al servidor, mientras que el servidor está transmitiendo una secuencia de frames intercalados para el cliente como son los stream 1 y 3. Como resultado, hay tres intercambios de petición-respuesta paralelas que son los tres stream activos 1, 3, y 5.

- Los tres identificadores de stream son impares inicializados por el cliente.
- El servidor está enviando múltiples data frames para el stream 1, que son la respuesta de la solicitud anterior del cliente. Esto también indica la respuesta frame headers que fue transferida antes.
- El servidor ha intercalado un headers y data frame para stream 3 entre data frame para la respuesta del stream 1, multiplexación en acción.
- El cliente transfiere un frame data para stream 5, lo que indica que un header frame fue transferido antes.

Por lo tanto, la conexión del gráfico es de multiplexación de tres streams en paralelo, cada uno en las diversas etapas del ciclo de procesamiento, el servidor determina el orden de los frames, y no tiene que preocuparse por el tipo o contenido de cada stream. Stream 1 podría ser una gran transferencia de datos o una secuencia de vídeo, pero no bloquea los otros streams dentro de la conexión compartida.

2.5.7. Solicitud de priorización

Una vez que un mensaje HTTP se divide en muchos frames individuales, el orden exacto en el que los frames se intercalan y se entregan se puede optimizar para mejorar aún

más el rendimiento de las aplicaciones. Para facilitar esto, a cada stream se puede asignar un valor de prioridad de 31 bits donde:

- 0: representa el stream de máxima prioridad.
- 2^{31}-1: representa el stream de más baja prioridad.

Con las prioridades asignadas tanto el cliente y el servidor pueden aplicar diferentes estrategias para procesar stream individuales, mensajes y frames en un orden óptimo; el servidor puede priorizar el procesamiento del stream mediante el control de la asignación de recursos como son CPU, memoria, ancho de banda, y una vez que los datos de respuesta estén disponibles prioriza la entrega de frames de alta prioridad para el cliente.

El protocolo HTTP 2.0 no especifica ningún algoritmo para las prioridades, sin embargo, el cliente proporciona datos de prioridad, y el servidor adapta su procesamiento y entrega en base a las prioridades los stream indicados.

Si el servidor no tiene en cuenta toda la información de prioridad, puede retrasar involuntariamente la aplicación, por ejemplo, un navegador bloqueado puede estar a la espera de CSS, JavaScript, envío de imágenes, etc. Sin embargo, con un estricto orden de prioridad se puede generar escenarios óptimos.

Los frames de varios niveles de prioridad pueden ser intercalados por el servidor y los stream de alta prioridad deben tener dicha prioridad durante la fase de transformación con respecto a la asignación de ancho de banda entre el cliente y el servidor. Para el mejor uso de la conexión se requiere una combinación de niveles de prioridad.

2.5.8. Solicitud de priorización del navegador y Http 2.0

No todos los recursos tienen la misma prioridad al renderizar una página en el navegador, el documento HTML es fundamental para construir el DOM[3] y el CSSOM[4],

---

3 El ***Document Object Model*** o Modelo de Objetos del Documento es una interfaz de programación de aplicaciones (API) para acceder, añadir y cambiar dinámicamente contenido estructurado en documentos, y proporciona un conjunto estándar de objetos para representar documentos HTML y XML. Disponible en https://es.wikipedia.org/wiki/Document_Object_Model, Abril -2015.

4 Modelo de objetos CSS orientado a proporcionar capacidades básicas a los scripts definidos por el autor para permitir el acceso y la manipulación de la información y de los procesos relacionados con el estado de estilo.Disponible en https://drafts.csswg.org/cssom/ Enero - 2015

los cuales pueden ser bloqueados en los recursos de JavaScript; mientras que el resto de recursos como las imágenes se buscan con menor prioridad.

Para acelerar el tiempo de carga de la página, todos los navegadores modernos dan prioridad a las solicitudes en función del tipo de activo, su ubicación en la página, y la prioridad incluso aprendida de visitas anteriores.

Con el HTTP 1.x, el navegador tiene capacidad limitada para los datos de prioridad, y no hay manera de comunicar la prioridad al servidor. Usa conexiones en paralelo con hasta 6 solicitudes por origen, dichas solicitudes se ponen en cola en el cliente hasta que la conexión esté disponible lo que añade latencia.

HTTP 2.0 resuelve todas estas ineficiencias, el navegador envía cada solicitud al momento en que el recurso se conoce, especifica la prioridad de cada stream, y el servidor determina la entrega de respuesta óptima. Así elimina la latencia de solicitud de colas innecesarias y permite hacer el uso más eficiente de cada conexión.

2.5.9. Una conexión por origen

Al tener menos conexiones desde el cliente, se reduce la latencia y el número total de paquetes enviados por HTTP 2.0 puede ser el 40% menos que con HTTP. El manejo de un gran número de conexiones simultáneas en el servidor se convierte en un problema de escalabilidad, pero con HTTP 2.0 se reduce esta carga.

Una conexión por origen reduce significativamente la sobrecarga asociada con el menor número de comunicaciones para administrar la conexión, tiene menor consumo de memoria, reduce costos operativos, y un mejor rendimiento de la conexión. Además, de muchos otros beneficios en todas las capas como:

- Priorización consistente entre todos los streams.
- Mejor compresión a través del uso de un único contexto de compresión.
- Mejora el impacto en la congestión de red debido a un menor número de conexiones TCP.
- Menos tiempo y recuperación de pérdidas.

La mayoría de las transferencias HTTP son cortas mientras que TCP está optimizado para las transferencias de datos de larga duración. («HTTP Strict Transport Security», 18 de enero del 2015)

2.6 Control de flujo

La multiplexación de múltiples stream sobre la misma conexión TCP introduce la limitación de recursos de ancho de banda compartido, las prioridades del stream pueden ayudar a determinar el orden de entrega, pero por sí solas son insuficientes para controlar la asignación de recursos entre los streams.

HTTP 2.0 proporciona un mecanismo para el control de stream y el flujo de conexión con las siguientes características de control de flujo:

a. El control de flujo es específico de la conexión, es decir hop-by-hop no de extremo a extremo.
b. El control de flujo se basa en frames windows update, donde el receptor anuncia la cantidad de bytes que está dispuesto a recibir en un stream y para toda la conexión.
c. El tamaño de la ventana de control de flujo se actualiza por un frame window_update, que especifica el ID de stream y el valor de incremento del tamaño de la ventana.
d. El control de flujo es direccional, el receptor puede optar por configurar cualquier tamaño de la ventana que desea para cada stream y para la conexión. Un remitente debe respetar los límites de control de flujo impuestos por un receptor.
e. El control de flujo puede ser desactivado por un receptor, para un stream individual o para toda la conexión.
f. El valor inicial de la ventana de control de flujo es 65.535 octetos tanto para los nuevos stream como para la conexión global.
g. El frame Data está sujeta al control de flujo, los demás tipos de frame no consumen espacio en la ventana de control de flujo.

En la conexión HTTP 2.0 el cliente y servidor intercambian frames settings, que establecen los tamaños de la ventana de control de flujo en ambas direcciones. Cada lado también puede desactivar el control de flujo en un stream individual o de la conexión.

Como el control de flujo TCP no puede diferenciar todos los streams dentro de la única conexión, HTTP 2.0 cuenta con su propio control de flujo.

El control de flujo puede regular la cantidad de recursos consumidos por cada stream, y el receptor puede anunciar un tamaño inferior de la ventana en un stream específico para limitar la velocidad a la cual los datos se entregan.

2.7 Server push

El servidor tiene la capacidad de enviar varias respuestas en una sola solicitud del cliente. Es decir, a más de la respuesta de la solicitud original, el servidor puede push (empujar) recursos adicionales para el cliente, sin que él tenga que solicitar explícitamente cada uno.

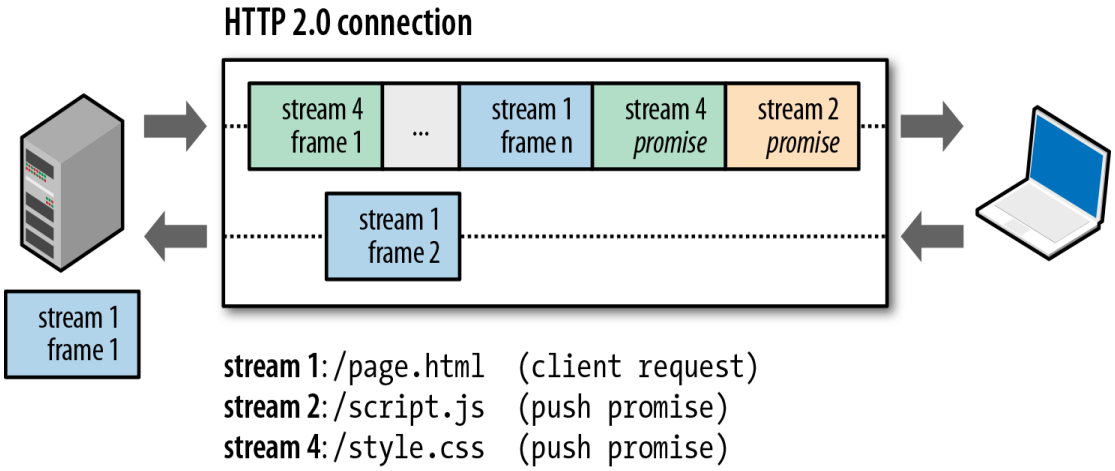


Gráfico 15-2 Server inicia nuevo stream (promises) para recursos push

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

Cuando se establece la conexión HTTP 2.0, el cliente y servidor intercambian frames settings, lo que limita el número máximo de streams simultáneos en ambas direcciones. De esta forma el cliente puede limitar el número de stream o desactivar el servidor push estableciendo este valor a cero.

¿Por qué necesitamos un mecanismo de ese tipo?

El servidor push ya sabe que recursos el cliente necesita sin tener que esperar a que él le solicite. Los beneficios importantes son los siguientes:

a) Recursos pushed pueden ser almacenados en caché por el cliente.
b) Recursos pushed pueden ser rechazados por el cliente.
c) Recursos pushed pueden ser reutilizados en diferentes páginas.
d) Recursos pushed pueden ser priorizadas por el servidor.

Todos los recursos pushed están sujetos a la política del mismo origen, y el servidor no puede arbitrariamente hacer push el contenido del cliente a terceros, y debe ser autorizado para proporcionar el contenido.

2.7.1. La implementación del servidor push en Http 2.0

El protocolo HTTP 2.0 no especifica ningún algoritmo y la decisión está en manos de los desarrolladores. Como resultado, hay muchas estrategias posibles cada una de las cuales se pueden adaptar al contexto de la aplicación o al servidor en uso:

- La aplicación puede iniciar el server push de forma explícita dentro de su código, esto requiere estrecha conexión con el servidor HTTP 2.0 teniendo el desarrollador un control total.
- La aplicación puede indicar al servidor los recursos asociados que desea pushed a través de una cabecera HTTP adicional, esto desacopla la aplicación desde el API servidor HTTP 2.0, por ejemplo, Apache's mod_spdy busca X-Associated-Content header, que enumera los recursos para ser pushed.
- El servidor puede aprender automáticamente los recursos asociados sin depender de la aplicación, también analiza el documento y deduce los recursos a ser pushed, o puede analizar el tráfico entrante y tomar las decisiones apropiadas; por ejemplo, el servidor puede recoger los datos de dependencia basados en la cabecera Referrer, y luego automáticamente hacer push los recursos críticos al cliente.

Es importante señalar que los recursos pushed van a la caché del cliente, como si él iniciará la solicitud, no hay API del cliente o devoluciones de llamada de JavaScript que sirven de notificaciones de que un recurso push ha llegado.

2.8 Compresión de cabecera

Cada transferencia HTTP lleva un conjunto de cabeceras que describen recursos y sus propiedades. En HTTP 1.1 los metadatos se envían como texto sin formato, añade 500 a 800 bytes de sobrecarga por solicitud, y kilobytes si se requieren cookies HTTP.

Para reducir esta sobrecarga y mejorar el rendimiento, HTTP 2.0 comprime cabeceras metadatos utilizando el formato de compresión HPACK de la siguiente manera:

- Permite que la transmisión de los campos de cabecera sea codificada por un código estático de Huffman, lo que reduce su tamaño de transferencia individual.
- Se requiere que tanto el cliente y el servidor mantengan y actualicen una lista indexada de campos de cabecera vistos anteriormente (es decir, establecer un contexto de compresión compartida), que luego se utiliza como referencia para codificar de manera eficiente los valores transmitidos anteriormente.

La codificación de Huffman permite que los valores individuales sean comprimidos cuando se transfieren, y la lista indexada de los valores transferidos con anterioridad permite codificar valores duplicados mediante la transferencia de los valores índice que se pueden utilizar para buscar de manera eficiente, reconstruir las claves y valores de cabecera completa.

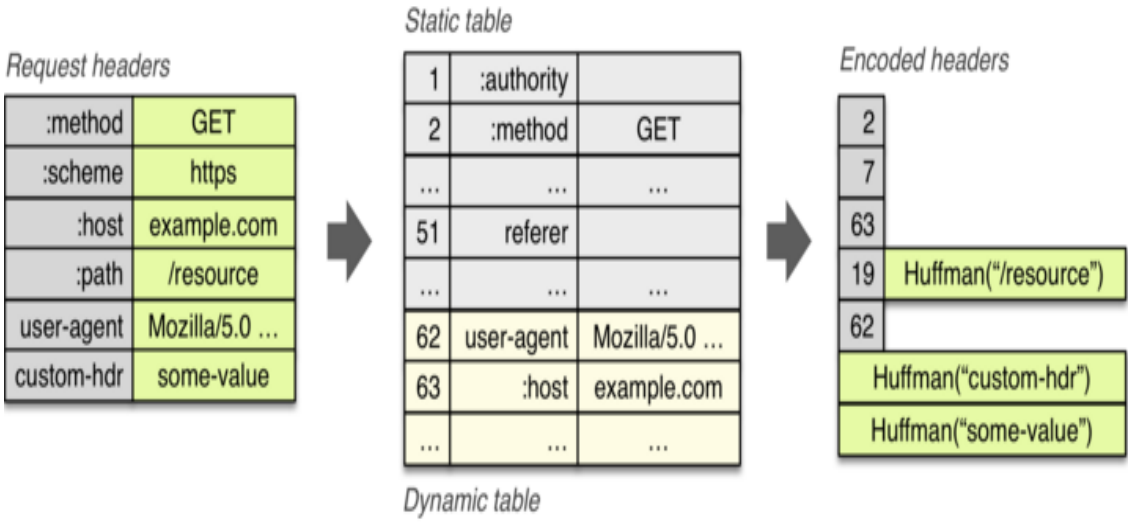


Gráfico 16-2 Compresión de cabecera HTTP 2.0 HPACK

Fuente: http://chimera.labs.oreilly.com/books/1230000000545/ch12.html#_analyzing_http_2_frame_data_flow

La compresión HPACK consta de tablas estáticas y dinámicas, donde la tabla estática define la especificación y proporciona una lista de campos comunes de cabecera HTTP (por ejemplo, nombres de encabezado válido); mientras que la tabla dinámica está inicialmente vacía y es actualizada basándose en los valores intercambiados dentro de una conexión particular.

Como resultado, el tamaño de cada solicitud se reduce con el uso de la codificación de Huffman para los valores no conocidos, y sustitución de índices para los valores que ya están presentes en las tablas estáticas o dinámicas en cada lado. HTTP 2.0 evita la transmisión de datos de cabeceras redundantes, lo que reduce significativamente la sobrecarga de cada solicitud.

Los pares key-value raras vez cambian a lo largo de la vida útil de una conexión, por lo que es necesario transmitir una sola vez. De hecho, si no cambian las cabeceras entre las solicitudes (por ejemplo, una solicitud que solicita el mismo recurso), entonces la sobrecarga de cabecera es de cero bytes, así todas las cabeceras se heredan automáticamente de la solicitud anterior.

La lista de cabecera es un conjunto de cero o más campos de cabecera, cuando se transmite a través de una conexión la lista de cabecera se serializa en un bloque de cabecera usando la compresión de cabecera de HTTP. El bloque de cabecera serializado se divide en una o más secuencias de octetos llamado fragmentos de bloques de cabecera, y se transmite dentro del payload de cabecera, push_promise, o frame continuation.

Un contexto de compresión y descompresión se utiliza para toda la conexión. El receptor descomprime el bloque cabecera para reconstruir la lista de cabecera. Un bloque de cabecera consiste de:

- Un frame push_promise con la flag END_HEADERS set, o de
- Un frame push_promise con la flag END_HEADERS cleared y uno o más frames Continuation cuyo último frame de éste tiene el flag END_HEADERS set.

Cada bloque de cabecera se procesa como unidad discreta, son transmitidos como una secuencia contigua de frames, sin frames intercalados de cualquier otro tipo o cualquier otro stream. El último frame de una secuencia de cabeceras o frames Continuation debe tener el flag End_Headers set; mientras que el último frame de una secuencia de

Push_Promise debe tener el flag End_Headers set, esto permite que un bloque de cabecera sea lógicamente equivalente a un solo frame.

Los fragmentos de bloques de cabecera sólo se envían como payload de cabeceras, Push_Promise o frames Continuation, porque transportan datos que modifican el contexto de compresión mantenido por un receptor. Un endpoint que recibe Push_Promise o frames Continuation debe realizar la descompresión incluso si los frames son descartados. Un receptor deberá terminar la conexión con un error de conexión de tipo Compression_Error si no descomprime un bloque de cabecera. (Belshe, Thomson, & Peon, noviembre 2014)

2.9 Manejo de errores

HTTP 2.0 permite dos clases de errores:

- Una condición de error que haga a la conexión inservible es un error de conexión.
- Un error en un stream individual es un error de stream.

2.9.1. Manejo de error de conexión

Un error de conexión es cualquier error que impide procesar la capa framing, o corrompe cualquier estado de la conexión.

Un endpoint que encuentra un error de conexión debe enviar un frame Goaway con el identificador de último stream recibido con éxito, y debe cerrar la conexión TCP.

2.9.2. Códigos de error

Son campos de 32 bits que utilizan RST_STREAM y Goaway para transmitir las razones del error de conexión.

Los códigos de error comparten un espacio de código común, se aplican a cualquier stream o toda la conexión y no tienen semántica definida en otro contexto. A continuación, se define los siguientes códigos de error:

Tabla 4-2 Códigos de error

No_Error (0x0)	Es resultado de un error. Por ejemplo, un goaway para indicar un shutdown de una conexión.
Protocol_Error (0x1)	El endpoint detecta un error de protocolo no especificado, es decir cuando no está disponible.
Internal_Error (0x2)	El endpoint encontró un inesperado error interno
Flow_Control_Error (0x3)	El endpoint detecta que fue violado el protocolo de control de flujo.
Settings_Timeout (0x4)	El endpoint envía un frame settings, pero no recibió una respuesta en el momento oportuno.
Stream_Closed (0x5)	El endpoint recibe un frame después de que un stream estaba medio cerrado.
Frame_Size_Error (0x6)	El endpoint recibe un frame con un tamaño inválido.
Refused_Stream (0x7)	El endpoint niega el stream antes de realizar cualquier procesamiento de la aplicación.
Cancelar (0x8)	El endpoint indica que ya no es necesario el stream.
Compression_Error (0x9)	El endpoint es incapaz de mantener el contexto de compresión de cabecera para la conexión.
Connect_error (0xa)	La conexión establecida en respuesta a una solicitud de conexión fue reset o anormalmente cerrado.
Enhance_Your_Calm (0xb)	El endpoint detecta un comportamiento que podría estar generando una carga excesiva.
Inadequate_Security (0xc)	El transporte subyacente tiene propiedades que no cumplen con los requisitos mínimos de seguridad.
HTTP_1_1_Required (0xd)	El endpoint requiere se utilice HTTP 2.0.

Realizado por: León Isabel, 2016

Códigos de error desconocido o no compatible, no deberán producir cualquier comportamiento especial. Estos pueden ser tratados por una aplicación como un equivalente a un Internal_Error.

2.10 Seguridad de Http 2.0

Se describe a continuación algunos de los atributos del protocolo HTTP que mejoran la interoperabilidad, y reducen la exposición a vulnerabilidades de seguridad.

2.10.1. Server name indication “SNI”

Las implementaciones TLS deben soportar la extensión Server Name Indication para los protocolos de nivel superior incluyendo HTTPS. Sin embargo, a diferencia de la implementación, el uso del SNI en determinadas circunstancias es una cuestión de política local.

TLS no proporciona un mecanismo para que el cliente diga el nombre del servidor, esto puede ser conveniente para facilitar conexiones seguras a los servidores que alojan varios servidores "virtuales" en una única dirección de red subyacente.

SNI admite la implementación de múltiples servidores virtuales protegidos con TLS en una sola dirección, y brinda una mayor seguridad permitiendo que cada uno tenga su propio certificado.

2.10.2. Gestión de la conexión

Para un mejor rendimiento se espera que las conexiones del cliente no se cierren hasta cuando él cierre la conexión. Los clientes no deben abrir más de una conexión HTTP 2.0 a un determinado host y par de puertos, un servicio alternativo “ALT-SVC”, o un proxy configurado.

Un cliente puede abrir varias conexiones con la misma dirección IP y puerto TCP utilizando diferentes valores de Server Name Indication o proporcionando diferentes certificados de cliente TLS, pero debe evitar la creación de múltiples conexiones con la misma configuración.

Los servidores mantienen las conexiones abiertas el mayor tiempo posible, y pueden terminar las conexiones inactivas. Cuando se cierra la conexión TCP, el punto de terminación debe primero enviar un goAWAY para que ambos extremos puedan determinar si frames enviados han sido procesados y si están completados o por terminar tareas restantes. (Belshe, Thomson, & Peon, noviembre 2014)

2.10.3. Reutilización de la conexión

En HTTP 2.0, el método Connect se utiliza para establecer un túnel sobre un único stream a un host remoto, y con proxies HTTP para establecer una sesión TLS con un servidor de origen para propósitos de interacción con recursos "https".

Una conexión puede ser reutilizada siempre y cuando el servidor de origen está autorizado, así para los recursos de "http" depende si el host tiene la misma dirección IP y para recursos "https" tener un certificado válido para el host en el URI.

Un servidor de origen ofrece un certificado con múltiples atributos "subjectAltName" o nombres con comodines, uno de los cuales es válido para la autoridad en la URI. Por ejemplo, un certificado con “subjectAltName" de "*.example.com" podría permitir el uso de la misma conexión para las peticiones al URI que comiencen con "https://**a**.example.com/" y "https: //**b**.example.com/ ".

En algunas implementaciones, la reutilización de una conexión para varios orígenes puede dar lugar a solicitudes dirigidas a un mal servidor de origen. Un servidor que no desee clientes que reutilicen las conexiones usará el código 421.

Un cliente que está configurado para utilizar un proxy a través de HTTP 2.0 dirige las solicitudes a ese proxy a través de una sola conexión, es decir se reutiliza la conexión con el proxy.

2.10.4. El código de estado 421 (solicitid misdirected)

Los clientes que reciben un 421 de respuesta de un servidor pueden volver a intentar la solicitud, esto es posible si una conexión se reutiliza o si se selecciona un servicio alternativo “ALT-SVC”.

Este código de estado no debe ser generado por proxies. Una respuesta 421 es cacheable por defecto, es decir a menos que se indique lo contrario por la definición del método o los controles de caché explícitas.

2.10.5. TLS

TLS proporciona privacidad e integridad de datos entre dos aplicaciones que se comunican y da seguridad de cifrado. El protocolo se compone de dos capas: el Protocolo de Registro TLS y el Protocolo de Enlace TLS.
El protocolo de registro TLS se utiliza para la encapsulación de varios protocolos de alto nivel, mientras que el protocolo de enlace TLS permite que el servidor y el cliente se autentiquen entre sí y negocien un algoritmo de encriptación y claves criptográficas antes de que el protocolo de aplicación transmita o reciba su primer byte de datos.

El protocolo de registro TLS proporciona seguridad de conexión y tiene dos propiedades básicas que son las siguientes:

a) ***La conexión es privada:*** la criptografía simétrica se utiliza para el cifrado de datos, por ejemplo: AES, RC4, etc., donde las claves para éste cifrado se generan de forma única para cada conexión y se basan en un secreto negociado por otro protocolo, tal como el Protocolo TLS Handshake. El Protocolo de registro también se puede utilizar sin cifrado.
b) ***La conexión es fiable:*** el transporte de mensajes incluye una comprobación de la integridad del mensaje utilizando un MAC con llave. El Protocolo de Registro puede funcionar sin un MAC por lo general sólo se utiliza en este modo, mientras que otro protocolo puede usarlo como medio de transporte para la negociación de los parámetros de seguridad.

El protocolo de enlace TLS tiene tres propiedades básicas:

1. La identidad del grupo puede ser autenticado con la criptografía asimétrica o clave pública, por ejemplo, la criptografía RSA, DSA, etc. Esta autenticación puede ser opcional, pero generalmente se requiere para al menos uno del grupo.
2. La negociación de un secreto compartido es segura: el secreto negociado no está disponible para los fisgones y para cualquier conexión autenticada, no se puede obtener el secreto ni siquiera por un atacante que puede colocarse en el medio de la conexión.
3. La negociación es fiable: ningún atacante puede modificar la comunicación de la negociación sin ser detectado por las partes en la comunicación. (< Tim Dierks>, 2008)

2.10.6. Servicios de seguridad de TLS

TLS requiere tres servicios de seguridad que son:

- **Confidencialidad**: todas las comunicaciones de capa de aplicación se cifran excepto el receptor, recomendable usar algoritmos y opciones de configuración que aplican el secreto del tránsito de datos.
- **Integridad de los datos:** los cambios realizados en la comunicación son detectables por el receptor.

- **Autenticación:** un endpoint de la comunicación TLS se autentica como la entidad que pretende comunicarse.

TLS permite la autenticación de uno o ambos endpoints en la comunicación. Aunque algunos escenarios de uso de TLS no requieren autenticación, pero la justificación para esta decisión se proporciona en “No autenticado TLS y cifrado oportunista”.

No autenticado TLS y cifrado oportunista

Si el cliente TLS no verifica el certificado del servidor, se conoce como "no autenticado TLS". En algunos escenarios el uso de TLS es opcional, donde el cliente decide dinámicamente ("oportunista") utilizar TLS con un servidor en particular o conectar en clear, esto se denomina "cifrado oportunista".

Los servidores pueden ser utilizados simultáneamente para TLS autenticado y no autenticado (de hecho, un servidor no puede saber si un cliente usará TLS autenticada o no autenticada).

2.10.7. Implementación de Http 2.0 con TLS y ALPN

HTTP 2.0 no requiere el uso de TLS, pero para obtener los mejores resultados, cualquier implementación HTTP 2.0 debe comenzar y soportar TLS con la negociación ALPN, además del flujo regular de trabajo de HTTP de actualización.

Las implementaciones de HTTP 2.0 deben utilizar TLS versión 1.2 o superior, y si usan 1.2 deben soportar TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 [TLS-ECDHE] con la curva elíptica P256.

La implementación TLS debe soportar Server Name Indication, donde los despliegues de HTTP 2.0 que negocian TLS 1.3 o superior necesitan soportar y utilizar SNI; mientras que si usan TLS 1.2 están sujetos a otros requisitos establecidos. («A 2x Faster Web», 2014)

Un endpoint puede terminar una conexión HTTP 2.0 cuando no cumpla con los requisitos TLS dando un error de conexión de tipo INADEQUATE_SECURITY.

Un despliegue de HTTP 2.0 sobre TLS 1.2 debe desactivar la compresión cuando da lugar a la exposición de información que no se reveló; y desactivar la renegociación con un error de conexión de tipo PROTOCOL_ERROR.

Un despliegue de HTTP 2.0 sobre TLS 1.2 no debe utilizar cualquier conjunto de cifrado y los endpoints generan un error de conexión de tipo Inadequate_Secuity.

Las implementaciones deben soportar el intercambio de tamaños de claves de por lo menos 2048 bits para conjuntos de cifrado que utilizan campo finito de Diffie-Hellman (DHE) y 224 bits para conjuntos de cifrado que utilizan curva elíptica Diffie-Hellman (ECDHE). Los clientes deben aceptar tamaños DHE de hasta 4096 bits. (Belshe, Thomson, & Peon, 2014)

2.10.8. TLS - protocolo de negociación de capa de aplicación

Cuando varios protocolos de aplicación son compatibles con un solo número de puerto del lado del servidor, tal como el puerto 443, el cliente y el servidor tienen que negociar un protocolo de aplicación a usar con cada conexión. Llevar a cabo esta negociación es conveniente ya que no añade viajes de ida y vuelta de red entre el cliente y el servidor y permite la selección de certificados según el protocolo de aplicación negociada.

La extensión TLS permite a la capa de aplicación negociar la selección de protocolo dentro del TLS handshake, esto fue solicitado por el HTTPbis WG para abordar la negociación de HTTP 2.0 sobre TLS; sin embargo, ALPN facilita la negociación de los protocolos de la capa de aplicación arbitraria.

Con ALPN, el cliente envía la lista de protocolos de aplicaciones compatibles como parte del mensaje TLS ClientHello; mientras que el servidor elige un protocolo y lo envía como parte del mensaje TLS ServerHello.

Extensión del Protocolo de Negociación de Capa de Aplicación

Un nuevo tipo de extensión "application_layer_protocol_negotiation (16)" está definida y es incluido por el cliente en su mensaje "ClientHello".

```
enum {
      application_layer_protocol_negotiation(16), (65535)
     } ExtensionType;
```

El campo "extension_data" de la extensión application_layer_protocol_negotiation(16) deberá contener un valor "ProtocolNameList".

```
opaque ProtocolName<1..2^8-1>;
```

```
struct {
    ProtocolName protocol_name_list<2..2^16-1>
} ProtocolNameList;
```

"ProtocolNameList" contiene la lista de protocolos anunciados por el cliente, en orden descendente de preferencia. Los protocolos son nombrados por IANA-registered, opaque, non-empty byte strings, las cadenas vacías no deben ser incluidas y las cadenas de bytes no deberán ser truncadas.

Los servidores que reciben un ClientHello que contiene la extensión "application_layer_protocol_negotiation" responden al cliente con la selección del protocolo e ignorarán cualquier otro que no reconozca.

La extensión ("application_layer_protocol_negotiation (16)") es devuelto al cliente dentro del mensaje ServerHello, y el "ProtocolNameList" deberá contener exactamente un "ProtocolName".

Por lo tanto, un handshake completo con la extensión "application_layer_protocol_negotiation" en los mensajes ClientHello y ServerHello tiene el siguiente flujo:

```
ClientHello                    -------->       ServerHello
  (ALPN extension &                                (ALPN extension &
   list of protocols)                               selected protocol)
                                                 Certificate*
                                                 ServerKeyExchange*
                                                 CertificateRequest*
                               <--------         ServerHelloDone
Certificate*
ClientKeyExchange
CertificateVerify*
[ChangeCipherSpec]
Finished                       -------->
                                                 [ChangeCipherSpec]
                               <--------         Finished
Application Data               <------->         Application Data
```

Gráfico 17-2 Handshake completo con la extensión ALPN
Fuente: Friedl, S., Langley, A., E. Stephan & Popov, A., https://tools.ietf.org/html/rfc7301#section-3

Dónde * indica los mensajes que pueden ser opcionales. Un handshake abreviado con la extensión "application_layer_protocol_negotiation" tiene el siguiente flujo:

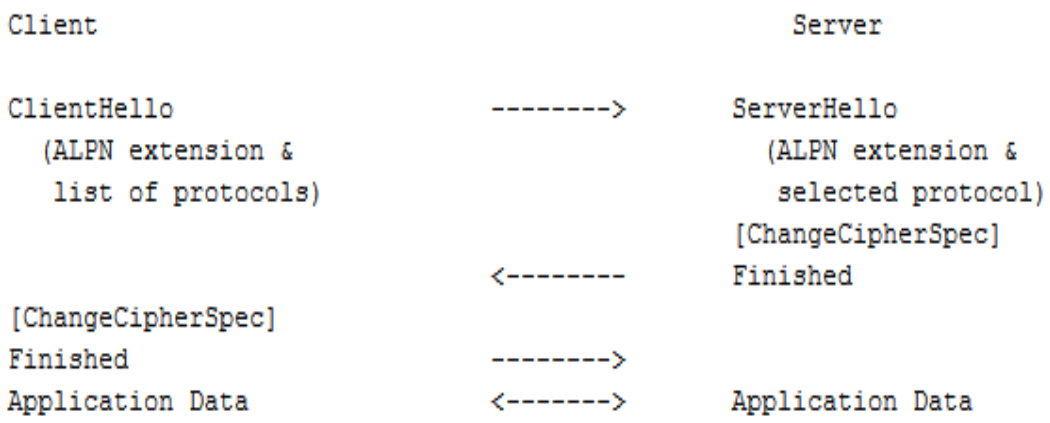


Gráfico 18-2 Handshake abreviado con la extensión ALPN
Fuente: Friedl, S., Langley, A., E. Stephan & Popov, A., https://tools.ietf.org/html/rfc7301#section-3

A diferencia de muchas otras extensiones TLS, esta extensión no establece las propiedades de la sesión, sólo el de la conexión. Cuando se utilizan reanudación de sesión, los contenidos anteriores de esta extensión son irrelevantes, y sólo los valores en los nuevos mensajes handshake se consideran.

Selección del Protocolo

El servidor sólo seleccionará un protocolo y en caso de no admitir ninguno el servidor responderá con una alerta fatal "no_application_protocol".

```
enum {
        no_application_protocol(120),
        (255)
      } AlertDescription;
```

El protocolo identificado en la extensión "application_layer_protocol_negotiation" en el ServerHello será definitivo para la conexión hasta su renegociación, es decir el servidor no debe responder con un protocolo seleccionado y posteriormente utilizar un protocolo diferente para el intercambio de datos de la aplicación.

2.10.9. Seguridad oportunista para Http

La seguridad oportunista no ofrece las mismas garantías que el uso de TLS con "https", es vulnerable a ataques activos, y no cambia el contexto de seguridad de la conexión.

Entre los objetivos están los siguientes:

- Hacer que el uso de HTTP sea más robusto.
- Limitar el potencial de ataques activos, no tiene la intención de ofrecer el mismo nivel de protección que otorga "https" URI, sino de aumentar la probabilidad de que un ataque activo pueda ser detectado.
- Proporcionar facilidad para la implementación, despliegue y funcionamiento. Se espera que este mecanismo tenga un impacto mínimo en el rendimiento, y un esfuerzo administrativo trivial al configurar.

Usando URI HTTP sobre TLS

Un servidor que apoya URI "http" proporciona un anuncio de servicio alternativo para un identificador de protocolo que utiliza TLS, como "h2".

Un cliente que da importancia a la protección contra ataques pasivos sobre el rendimiento podría optar por retener solicitudes hasta una conexión cifrada disponible. Sin embargo, si tal conexión no puede establecerse con éxito, el cliente puede reanudar el uso de la conexión sin cifrar.

Autenticación del servidor

El URI "http" no requiere autenticación criptográfica del servidor que sí está implícita en "https". Por lo tanto, la autenticación del servidor no es obligatoria para los URI "http".

Cuando se conecta a un servicio alternativo por un "http", los clientes no requieren autenticar al servidor. El servicio alternativo puede proporcionar cualquier certificado o incluso seleccionar cifrado TLS que no incluye la autenticación.

Interacción con URI "https"

Al utilizar servicios alternativos, las solicitudes de recursos identificados por ambos URI "http" y "https" pueden utilizar la misma conexión, ya que HTTP 2.0 permite las solicitudes de múltiples orígenes en la misma conexión.

Las URI "https" confían en la autenticación del servidor, una conexión que se crea inicialmente con "http" sin la autenticación del servidor no puede ser utilizado para "https" hasta que el certificado de servidor se autentique correctamente.

Exigir uso de TLS

Un cliente que no realiza la autenticación es una víctima fácil de la suplantación de servidores, a través de ataques man-in-the-middle.

Consideraciones de seguridad oportunista para HTTP

Entre las consideraciones de seguridad están las siguientes:

a. Indicadores de Seguridad

Agentes de usuario no deben proporcionar ningún indicio de seguridad cuando un recurso de "http" se adquiere mediante TLS. Los indicadores que pueden tener el mismo nivel de seguridad de https no pueden ser usados, por ejemplo, el uso de un bloqueo de dispositivo.

b. Ataques Downgrade

Un ataque downgrade contra la negociación de TLS es posible. Con el campo de cabecera “HTTP-TLS", esto se limita a ocasiones en las que los clientes no tienen información previa o cuando el valor PERSISTED de "HTTP-TLS" ha expirado.

Debido a que el campo de cabecera "Alt-SVC" probablemente aparezca en un canal autenticado y sin cifrar, está sujeto a downgrade por los atacantes de la red.

En su forma más simple, un atacante que quiere que la conexión permanezca en clear necesita únicamente el campo de cabecera "Alt-SVC" de respuestas.

Ataques Downgrade pueden ser mitigados parcialmente utilizando el campo de cabecera "HTTP-TLS", ya que el cliente puede evitar el uso de texto plano. Sin embargo, esto sólo funciona cuando una conexión anterior se ha establecido sin un atacante activo; un continuo atacante activo evita que el cliente use TLS u ofrezca su propio certificado.

c. Consideraciones de privacidad

Servicios alternativos en caché se pueden utilizar para realizar un seguimiento de los clientes a través del tiempo; por ejemplo, usando un nombre de host específico del usuario.

El borrado de la caché reduce la capacidad de los servidores para realizar un seguimiento a los clientes; por lo tanto, es recomendable que los clientes limpien la caché de servicio alternativo.

d. Confusión respecto al esquema de solicitud

Muchas implementaciones HTTP 1.1 usan la presencia o ausencia de TLS para determinar si las solicitudes son para "http" o "https". Esto es necesario en muchos casos por la forma de una solicitud HTTP 1.1 que no lleva una indicación explícita del esquema URI.

HTTP 1.1 no debe utilizarse para las solicitudes de forma oportunista asegurada. (Nottingham and Thomson, junio 2015)

2.10.10. Consideraciones de seguridad de Http 2.0

Entre las consideraciones de seguridad del protocolo HTTP 2.0 están:

1) Autoridad del servidor

HTTP 2.0 se basa en la definición de autoridad de HTTP 1.1 para determinar si un servidor está autorizado para proporcionar una respuesta. Basándose en la resolución de nombre local para el URI "http" y la identidad del servidor autenticado para el “https".

2) Ataques a través de protocolo

Un atacante incita a un cliente iniciar una transacción en un protocolo con un servidor que entiende otro protocolo, dicho atacante es capaz de hacer que la transacción pareciera como válida en el otro protocolo. Esto puede ser usado en servidores mal protegidos por lo general en redes privadas.

Un handshake TLS completado con un ALPN para HTTP 2.0 puede considerarse como suficiente protección contra los ataques a través de protocolo.

El cifrado de TLS hace difícil a los atacantes controlar los datos que podrían ser utilizados en un ataque a través de protocolo sobre un protocolo cleartext. La versión de cleartext de HTTP 2.0 tiene una protección mínima contra éstos ataques.

El prefacio de conexión contiene una cadena que está diseñado para confundir a los servidores HTTP 1.1, pero ninguna protección especial para otros protocolos. Un servidor que pasa por alto el campo de cabecera Upgrade de la petición HTTP 1.1, hace que el prefacio de conexión del cliente este expuesto al ataque a través de protocolo.

3) Ataques de encapsulación de intermediación

Las solicitudes o respuestas que contienen nombres no válidos de los campos de cabecera deben ser tratadas como malformación. Un intermediario, no puede traducir una petición o respuesta HTTP 2.0 que contiene un nombre de campo no válido en un mensaje HTTP 1.1.

Los valores que son codificados no alterarán el análisis del campo de cabecera, donde: carriage return “retorno de carro” (CR, ASCII 0xd), line feed “salto de línea” (LF, ASCII 0xa), y zero character “carácter cero” (NUL, ASCII 0x0) pueden ser aprovechados por un atacante si son traducidos textualmente. Hay que tener en cuenta que los caracteres válidos son definidos por el "field-content" regla ABNF.

4) Cacheabilidad de respuestas pushed

Las respuestas de caché que son pushed están basadas en el campo de la cabecera Cache-Control proporcionada por el servidor. Sin embargo, esto causar problemas si un solo servidor aloja más de un inquilino, porque ofrece a múltiples usuarios una pequeña parte de su espacio URI.

Cuando múltiples inquilinos comparten espacio en el mismo servidor, ese servidor debe asegurarse de que los inquilinos no sean capaces de representar push de los recursos a los que no tienen autoridad. El incumplimiento de esto permitiría a un inquilino proporcionar una representación que se sirve de caché, anulando la representación real que la autoridad del inquilino provee.

Por lo tanto, las respuestas push que no están autorizadas no deben almacenarse en caché.

5) Consideraciones de denegación de servicio

Una conexión HTTP 2.0 exige más recursos para operar que la conexión HTTP 1.1, el uso de compresión de cabecera y de control de flujo depende de más recursos para almacenar una mayor cantidad de estados.

El número de frames push_promise no está limitado, y un cliente que acepte el servidor push debe limitar el número de streams para estar en un estado reserved (remote). Un excesivo número de stream del servidor push puede ser tratado como un error de stream de tipo enhance_your_calm.

El frame settings causa un gasto adicional del tiempo de procesamiento, debido a cambios innecesarios en los parámetros, múltiples parámetros no definidos, o varios cambios de configuración en el mismo frame.

A la vez un gran número de frames pequeños o vacíos también causar un gasto en el tiempo de procesamiento de los frames de cabecera. Sin embargo, algunos usos son válidos como el envío de un Data vacío o frame de Continuation al final de un stream.

También Window_Update o Priority pueden causar un gasto innecesario de recursos, así con la compresión de cabecera puede desperdiciar recursos de procesamiento.

Todas estas características por ejemplo los cambios settings, frames pequeños, compresión de cabecera, tienen usos legítimos donde estas características se convierten en una carga cuando se utilizan innecesariamente o en exceso.

Un endpoint si no toma en cuenta estas consideraciones se expone a un riesgo de ataque de denegación de servicio, las implementaciones deben usar estas características y establecer límites en su uso. Así un endpoint puede tratar la actividad que es sospechosa como un error de conexión de tipo enhance_your_calm.

Límites en el tamaño del bloque de cabecera

Un bloque grande de cabecera tiene una implementación con una gran cantidad de estados, donde los campos de la cabecera críticos para el enrutamiento pueden aparecer al final del bloque evitando la transmisión a su destino final.

Un endpoint utiliza SETTINGS_MAX_HEADER_LIST_SIZE para indicar el límite del tamaño del bloque de cabecera, muchos endpoints exceden este límite arriesgándose a que la petición o respuesta se trate como malformación.

Un servidor que recibe un bloque grande de cabecera puede enviar un HTTP 431 (solicitud del campo de cabecera demasiado grande), un cliente descarta respuestas que no puede procesar, y el bloque de cabecera debe ser procesado para garantizar un estado de conexión constante a menos que la conexión se cierre.

6) El uso de compresión

HTTP 2.0 permite un mayor uso de la compresión para los campos de cabecera y entity bodies. La compresión permite al atacante recuperar datos secretos cuando se comprime en el mismo contexto que los datos controlados por el atacante.

Hay ataques en compresión que explotan las características de la web como el BREACH “INCUMPLIMIENTO”. El atacante induce múltiples peticiones que contienen variables de texto claro, y la longitud de texto más corta le ha hace suponer que ha dado con datos secretos.

Las implementaciones que se comunican en un canal seguro, no deberán comprimir el contenido que incluye los datos confidenciales a menos que los diccionarios de compresión sean separados para cada fuente de datos.

La compresión no debe utilizarse si la fuente de datos no puede ser determinada de forma fiable, así también la compresión de stream genérico como el previsto por TLS no debe utilizarse con HTTP 2.0.

7) El uso de padding

El Padding redundante puede ser contraproducente y una correcta aplicación depende de un conocimiento específico de los datos que se está siendo padding.

Para mitigar los ataques que se basan en la compresión, limitar la compresión podría ser preferible al padding como una contramedida.

El padding se puede utilizar para ocultar el tamaño exacto del contenido del frame, y para mitigar los ataques específicos dentro de HTTP, por ejemplo, ataques donde el contenido comprimido incluye texto plano controlado por el atacante y datos secretos.

El uso de padding puede resultar de menos protección, pues sólo hace que sea más difícil a un atacante inferir información de longitud al aumentar el número de frames que un atacante tiene que observar.

Esquemas de padding aplicados incorrectamente pueden ser fácilmente derrotados, y un padding al azar con una distribución predecible ofrece muy poca protección, del mismo modo las cargas útiles de padding a un tamaño fijo exponen información como tamaños de carga útil que cruzan el límite de tamaño fijo.

Los intermediarios deben conservar el padding para los frames data, pero pueden drop padding para los frames de cabecera y el push_promise. Una razón válida para que un intermediario cambie la cantidad de padding del frame es para mejorar la protección que proporciona el padding.

8) Consideraciones de Privacidad

Varias características de HTTP 2.0 proporcionan la oportunidad de correlacionar las acciones de un cliente o servidor con el tiempo, esto incluye el valor de settings, la forma en que el control de flujo de la ventana se gestiona, la forma en que las prioridades se asignan a los streams, tiempo de reacción a los estímulos, y la manipulación de las características opcionales.

HTTP 2.0 usa una única conexión TCP que permite la correlación de la actividad de un usuario en un sitio. Si las conexiones se vuelven a utilizar para diferentes orígenes, esto permite el seguimiento a través de esos orígenes.

Debido a que los frames de ping y settings solicitan respuestas inmediatas pueden ser usadas por un endpoint para medir la latencia a sus compañeros, lo cual podría tener implicaciones de privacidad en ciertos escenarios. (Belshe, Thomson, & Peon, noviembre 2014)

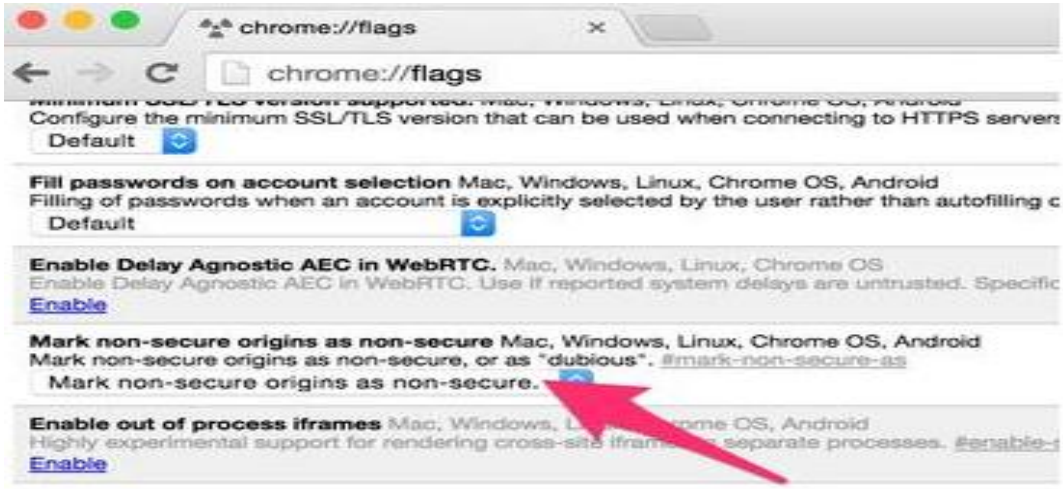
2.10.11. Google chrome se alista en el impulso a la web cifrada

La versión Canary bleeding-edge de Chrome no es estable ni probada lo suficiente para los usuarios normales, ofrece un ajuste manual que permite la advertencia acerca de las páginas cifradas. Una persona que visita una página sin cifrar verá en la barra de direcciones de Chrome un candado con una X roja sobre ella, como se muestra en el gráfico de abajo.

Gráfico 19-2 Navegador Chrome

Fuente: Stephen Shankland, http://www.cnet.com/news/chrome-becoming-tool-in-googles-push-for-encrypted-web/

Para activar la función, se tiene que instalar Chrome Canary y en la interfaz de éste con la siguiente sintaxis: Chrome://flags activar los "mark non-secure origins as non-secure".



Google's Chrome Canary, an early test version of the browser, can be configured to show the warning about unencrypted HTTP connections.

Gráfico 20-2 Google´s Chrome Canary

Fuente: Stephen Shankland, http://www.cnet.com/news/chrome-becoming-tool-in-googles-push-for-encrypted-web/,

Google sugiere una transición gradual a las advertencias, pero en la actualidad hay iconos de bloqueo verdes que denotan páginas seguras mientras que las páginas cifradas son claras.

El Electronic Frontier Foundation “EFF” y asociados, Firefox desarrollador de Mozilla, fabricante de equipo de redes Cisco Systems, y distribuidor de contenidos de Akamai Technologies lanzaron un proyecto a finales del año 2014 llamado Let's Encrypt para hacer más fácil a los operadores de sitios Web moverse a HTTPS. En concreto, Let's Encrypt ofrece certificados libres, y credenciales electrónicas requeridas para cifrar una conexión con el sitio Web.

La industria de Internet no está lista para entregar conexiones HTTPS en la escala que se ofrece conexiones HTTP, dijo Anton Karpov, jefe de seguridad de la información de Yandex**.** Los operadores de sitios Web tienen que preocuparse de que las conexiones HTTPS a veces se bloquean en áreas como aeropuertos y que contrariamente a la posición de Google, HTTPS no requiere hardware más robusto para manejar los cálculos de cifrado. La red de distribución de contenidos “CDN” puede ofrecer conexiones HTTPS, pero a menudo cobra una prima. (Stephen Shankland, «Be warned: Google enlist Chrome in push for encrypted Web», Enero 2015)

2.10.12. Características de seguridad de las Pyme

La pequeña y mediana empresa con su acrónimo PYME, tiene características distintivas, y dimensiones con ciertos límites ocupacionales y financieros prefijados por los Estados o regiones. (Diccionario de la lengua española, 2001)

Las Pyme son entidades independientes, con alta predominancia en el mercado de comercio, quedando excluidas del industrial por las grandes inversiones necesarias y por las limitaciones que impone la legislación en cuanto al volumen de negocio y de personal.

La adquisición de tecnología no es de alto costo, por no tener un excesivo capital y más bien es de acorde a su dimensión, aunque muchas empresas no invierten, razones por lo que las Pyme se han convertido en un blanco atractivo por los atacantes que aprovechan la poca seguridad para adquirir secretos industriales, bases de datos, etc. (Borghello Cristian, 2016)

Tomar en cuenta que la ingeniería social es por la cual logran extraer información de correo electrónico y redes sociales como Twitter o Facebook que abren la puerta a nuevas amenazas para los datos corporativos.

Entre las medidas de seguridad de las Pyme están las siguientes que son de costo mínimo:

- Es fundamental saber qué recursos de la empresa necesitan protección para controlar el acceso al sistema y los derechos de los usuarios del sistema de información.
- Establecer una política de seguridad de las Pyme.
- Restringir el acceso a programas y archivos.
- Mantener actualizado el software utilizado.
- Instale un antivirus en los servidores y estaciones de trabajo. Varios antivirus son gratuitos y se recomienda actualizar diariamente.
- Instale un Firewall de software o hardware. Los firewalls por software son gratuitos y las actualizaciones de sus reglas siempre son libres. En el caso del Firewall por hardware, el costo es muy bajo, el mantenimiento casi nulo y la protección que brindan es superior a la del software.

- Configure los permisos necesarios en cada recurso de su red, donde el usuario no accederá al recurso que no tenga permiso.
- Utilice software legal y obténgalo de sitios confiables. Existe mucho software libre y gratuito que ayuda a tareas habituales.
- Utilice passwords seguras mínimo de 8 caracteres y la misma debe ser modificado cada semana.
- No responda mails de desconocidos, ni tampoco haga click en adjuntos que recibe por mail.
- No instale aplicaciones por defecto.
- La tecnología no lo es todo, por eso tenga en cuenta los lineamientos y políticas existentes.
- Realice copias de seguridad (backup) de forma automática o manual, pero realícelas.

CAPÍTULO III

METODOLOGÍA DE INVESTIGACIÓN

En este capítulo, se detalla el diseño y tipo de la investigación, los métodos, técnicas e instrumentos utilizadas con su respectiva validación, adicionalmente se propone un método que ayude a evaluar para mejorar la seguridad web en las Pyme mediante criterios que se vieron en el capítulo anterior y principios de comprobación de errores sobre la web, que permitan en lo posible encontrar y mitigar las vulnerabilidades cuando no se aplica ciertas características a los servidores HTTP.

Se aclara que no existe una herramienta que garantice el 100% de seguridad en un sitio web ya que siempre va existir día a día que los hackers busquen nuevas vulnerabilidades impidiendo así adquirir el 100% de seguridad.

3.1 Diseño de la investigación

El presente tema de investigación es de tipo experimental donde se implementó un ambiente de laboratorio con dos servidores Linux instalados con la distribución de Fedora 23 donde el un servidor tuvo la configuración HTTP 1.1 y otro con el nuevo protocolo 2.0.

3.2 Tipo de investigación

El grado de profundidad con que se abordó la presente investigación es con una **investigación descriptiva** por la descripción de las características de los dos protocolos HTTP 1.1 y 2.0, el método propuesto, así como la guía de despliegue de HTTP 2.0, y, también se utilizó una **investigación experimental** que se basa en las pruebas realizadas en el escenario de laboratorio donde se observó los elementos más relevantes del objeto de estudio que se investiga.

3.3 Métodos, técnicas e instrumentos

3.3.1. Métodos

La presente investigación se desarrolló bajo el método científico por cuanto utiliza sus fases: planteamiento del problema, formulación de la hipótesis, levantamiento de la información, análisis e interpretación de resultados, comprobación de la hipótesis para luego de ésta secuencia ordenada de acciones establecer las conclusiones y difundir los resultados sobre el estudio realizado.

También se empleó el método propuesto de despliegue y mejoramiento de la seguridad web con sus 7 fases, con el fin de identificar las vulnerabilidades que posee el protocolo HTTP 1.1 y mitigarlas con las características de HTTP 2.0. Así también se elaboró la guía con los pasos para desplegar HTTP 2.0 en servidores web.

3.3.2. Técnicas

Entre las técnicas empleadas están las siguientes:

- **Revisión de documentación:** del objeto de estudio para su desarrollo, para lo cual se usó fuentes secundarias.
- **Pruebas:** realizadas en ambiente de laboratorio que para mayor objetividad el ambiente es real, los servidores se encuentran alojados en un hosting que de antemano fue comunicado sobre el trabajo de investigación que se realiza para evitar causar problemas a otros usuarios, por los ataques realizados.
- **Observación:** permite determinar los resultados de las pruebas realizadas en laboratorio.
- **Análisis comparativo:** en base a las pruebas ejecutadas a los dos protocolos se realizó el análisis comparativo y determinó los resultados de la investigación.

3.3.3. Fuentes

Para el desarrollo del presente tema las fuentes de información que se utilizaron son:

- Páginas de internet con información confiable

- Páginas web académicas
- Artículos científicos en base de datos de bibliotecas virtuales
- Conferencias académicas, seminarios
- Revistas científicas

3.3.4. Instrumentos

Para obtener el método de mejoramiento de seguridad web en las pequeñas y medianas empresas (Pyme) utilizando las características de seguridad de HTTP 2.0 se requiere de los siguientes instrumentos que evaluaron las vulnerabilidades y debilidades que tiene el protocolo HTTP 1.1.

Tabla 1-3 Requerimientos de la Investigación

INSTRUMENTO	CARACTERÍSTICAS
Fedora	23
Kali Linux	386i versión 2.3 – 2015
VMware Workstation	Versión 11
Nessus	Versión 5.2.1
Slowloris	Anónima
Botnets	Anónima

Realizado por: León Isabel, 2016

3.3.5. Validación de instrumentos

En la investigación se debe validar instrumentos que permitan la detección de vulnerabilidades en los protocolos HTTP 1.1 para lo cual se utiliza:

- **Fedora**

Fedora es una distribución de Linux estable, basada en Red Hat Package Manager, mantenida por la comunidad internacional de ingenieros, diseñadores gráficos y usuarios que informan fallos y prueban nuevas tecnologías. Fedora 23 trabaja con el kernel Linux 4.2, GNOME 3.18 y tiene 3 ediciones Workstation, Cloud, y Server.

En esta investigación se usará la versión 23 instalada en los dos servidores HTTP 1.1 y 2.0; dicha versión fue seleccionada por tener las últimas versiones de Apache que

soportan el protocolo HTTP 2.0, la versión Apache HTTP Server 2.4.17 tiene la negociación para ALPN, Upgrade, Direct y soporta el protocolo h2 y h2c.

- **Kali Linux**

Es la nueva generación de Linux Backtrack esta distribución tiene 600 herramientas para realizar pruebas de penetración y seguridad informática, en esta investigación se utilizará por ser gratuito, y para realizar pruebas de penetración y auditoría, es decir se buscará, controlará, y explotará las vulnerabilidades de un sistema informático con HTTP 1.1 y 2.0.

- **VMware Workstation**

Es un programa que permite virtualización, es decir permite correr varios sistemas operativos en una sola máquina física, dentro del trabajo se utilizará para tener el Kali Linux.

- **Nessus**

Es un programa que permite escanear vulnerabilidades de cualquier sistema operativo, tiene un nessusd que es un nessus demonio que realiza escaneos al objetivo, y un nessus basado en consola o gráfico que muestra el reporte de escaneos. Realiza un escaneo de puerto abiertos sea con nmap o con su propio escaneador para inmediatamente intentar atacar. Dentro de la investigación se usa para escanear las vulnerabilidades encontradas en los dos protocolos como son HTTP 1.1 y 2.0.

- **Slowloris**

Es un cliente de HTTP que su objetivo es realizar ataques como denegación de servicios a un servidor web. Éste intenta abrir muchas conexiones al servidor y mantenerlas abiertas por una infinidad de tiempo, para que no se cierre estas conexiones pone headers a la petición del HTTP. Se usará el script de slowloris para hacer ataques en ambos servidores HTTP 1.1 Y 2.0.

- **Botnets**

Son robots informáticos que son enviados por el internet ejecutado de manera automática contaminando el sistema informático a través de gusanos que son el transporte para este ataque masivo.

3.4 Método de despliegue y mejoramiento de la seguridad web en las Pyme utilizando las características de seguridad de Http 2.0

En el entorno del medio actual la infraestructura de internet ha crecido rápidamente y con ello conllevando que crezca el cibercrimen de manera que descontrola la seguridad de algunas infraestructuras.

El problema que se encuentra al intentar reaccionar a los agujeros de seguridad que fueron encontrados son la velocidad con que evoluciona la web 2 tanto en hardware como en software, esto hace que día a día surtan vulnerabilidades de 0 - day, que los usuarios son incapaces de percibir este tipo de ataque.

El mayor problema que existe en la seguridad es que no hay concienciación por parte de los usuarios responsables ante la seguridad en los protocolos de red como lo es HTTP 2.0, el ataque más común en el mundo de la web son los websites, ebanking que ofrecen servicios financieros que ponen en peligro los datos personales financieros.

Es así que en esta investigación se pretende utilizar un método de despliegue de la seguridad web con características de seguridad HTTP 2.0 para que no esté expuesto información financiera o de cualquier otra índole para mejorar la seguridad de la información en las Pyme.

En nuestro medio la brecha digital es grande especialmente en el sector empresarial ya que en su mayoría utilizan un computador para su trabajo cotidiano tanto en empresas grandes como las Pyme, el propósito de esta investigación es proponer líneas generales de investigación sobre la adopción de las características del protocolo HTTP 2.0, específicamente hacia el e-commerce y el e-business.

3.4.1. Método propuesto de despliegue y mejoramiento de la seguridad web

Se presenta la propuesta del método mediante la cual se pretende identificar los factores más comunes que limitan o favorecen su adopción como elemento clave en la estrategia

competitiva de las Pyme en nuestro medio. El método que se propone sigue las siguientes 7 fases:

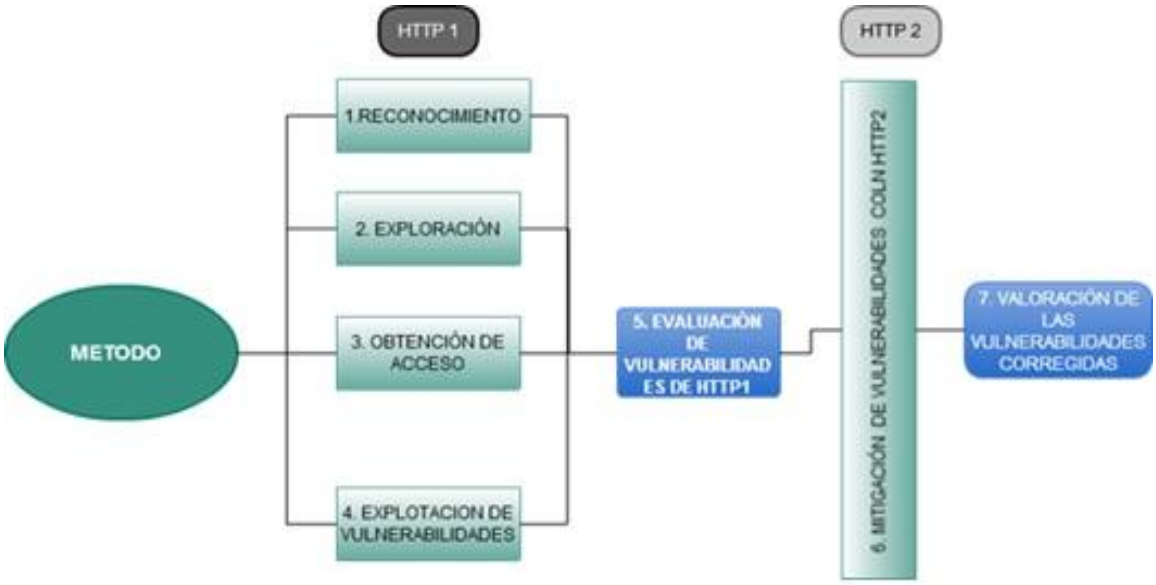


Gráfico 1-3 Método para mejorar la seguridad
Realizado por: León Isabel, 2016

Para el mejor aseguramiento en la web de las Pyme se utilizó el método expuesto en la Gráfico 1-3 que muestra las etapas que un atacante pueda aprovechar en el momento que logra penetrar al ambiente informático de las Pyme, como es conocido todo ataque siempre es producido desde el internet, en la mayor parte se lanzan desde equipos infectados sin que los usuarios de un equipo de cómputo sepan lo que está ocurriendo. A continuación, se describen las fases del método propuesto:

1.- RECONOCIMIENTO

La primera fase del método a utilizar para esta investigación es el reconocimiento porque los atacantes para introducirse en sitios web de las Pyme comienzan a crear estrategias, preparar información sobre la empresa mediana o pequeña que esta de blanco fácil de atacar obteniendo información a través de la ingeniería social el cual el objetivo es utilizar la seducción (Facebook, Twitter, correos electrónicos) para que los mismos empleados o personas en particular otorguen información.

Otra manera de obtener información también es el buscar en la basura o llamado también Dumpster diving, averiguar el tipo de SO del servidor donde están almacenados, aplicaciones de software en donde están ubicados los routers, que host

son los más accesibles, buscar bases de datos de internet (Whois) que da información como direcciones, nombres de dominios, información de contacto, servidores mails y DNS. Esta fase es la más larga debido a que los atacantes deben analizar todos los datos que logran obtener.

Gráfico 2-3 Fase de reconocimiento
Realizado por: León Isabel, 2016

2.- EXPLORACIÓN

La segunda fase Exploración o también denominada Escaneo se utiliza antes de lanzar un ataque a la red, es conocido que las Pyme manejan frecuentemente el internet para realizar transacciones o para realizar actividades que necesitan de la red, con la aplicación de esta metodología se da a conocer a propietarios y gerentes de empresas medianas y pequeñas las vulnerabilidades que tienen un sistema informático y las soluciones que se pretende corregir para mejorar la seguridad, eficacia y la eficiencia.

Ésta fase tiene como finalidad utilizar toda la información obtenida en la fase 1 para conocer las vulnerabilidades, como ejemplo podemos decir que si el atacante descubrió algún puerto abierto el tratara de buscar todas las vulnerabilidades y atacara por ese punto vulnerable.

Gráfico 3-3 Fase de exploración
Realizado por: León Isabel, 2016

3.- OBTENCIÓN DE ACCESO

La fase tres es la Obtención de Acceso, aquí se va a realizar la penetración, es decir, una vez que se realizó un reconocimiento en las Pyme, luego se escaneo, se realizará la penetración a los sistemas informáticos de las Pyme en las que un hacker va a explotar todas las vulnerabilidades encontradas.

Las penetraciones se los puede realizar de varias maneras offline (sin conexión a las LAN) o sobre el internet en esta incluye técnicas como:

- **Buffer overflows:** es el desbordamiento de la memoria temporal
- **Denial-of-servicie:** es el nivel de petición y denegación de servicio
- **Session hijacking:** es el secuestro o robo de sesión
- **Password cracking:** es adivinar claves por diversos métodos como diccionarios, fuerza bruta

En ésta tercera fase se demostrará como los intrusos pueden llegar a tener éxito en la penetración a un sistema informático con la siguiente información:

- Conociendo la arquitectura del sistema informático con la que cuenta las Pyme y su configuración de red, seguridad deficiente significa mayor acceso al sistema.
- Conocer el nivel de conocimientos de los profesionales que están a cargo de los sistemas informáticos de las Pyme.
- Dependerá de las destrezas del hacker que pretende incursionar en el sistema, conocimientos avanzados de redes.

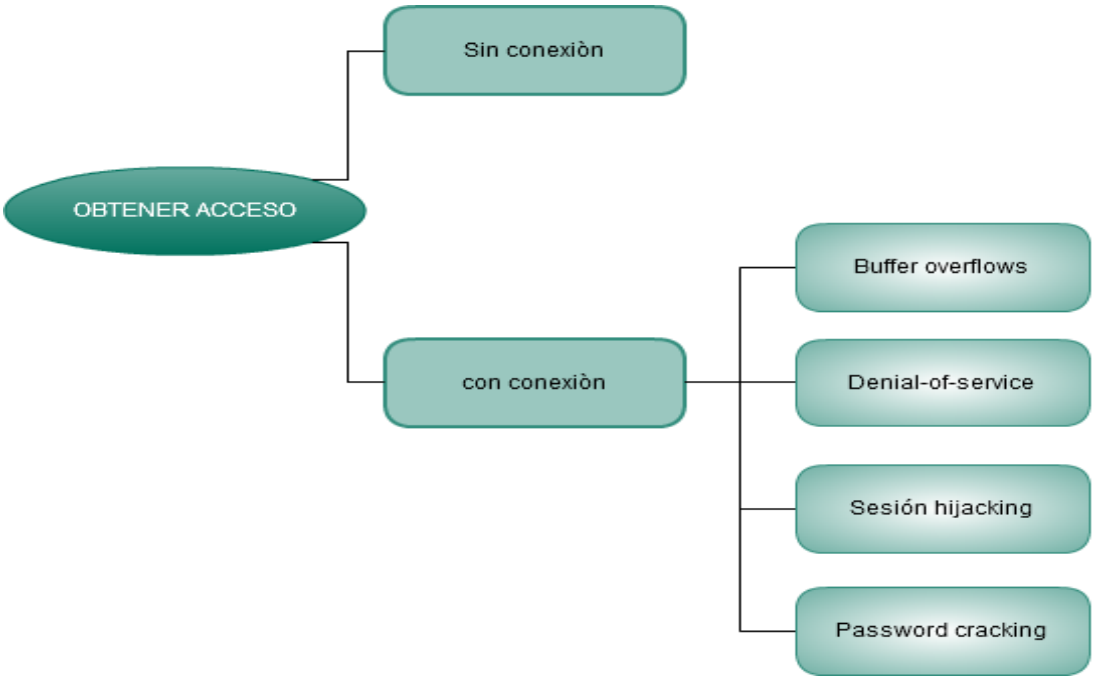


Gráfico 4-3 Fase de obtener acceso
Realizado por: León Isabel, 2016

4.- EXPLOTACIÓN DE VULNERABILIDADES

En la cuarta fase se explotan las vulnerabilidades después de obtener acceso a los sistemas informáticos, esto a través de ataques directos al sistema, mecanismos que son aprovechados por los hacking para entrar por alguna puerta que este abierta y robar información, credenciales, contraseñas, etc.

La explotación se ha dividido en dos partes payload y el código, unos de éstos métodos son capaces de explotar las vulnerabilidades de las Pyme.

En la investigación se explota las debilidades por estos dos métodos para que observen las personas encargadas de la seguridad de los datos que tan fácil son de vulnerar si no se aplica seguridad a la red, en forma especial al protocolo HTTP 1.1 que tiene ciertas características que ponen en desventaja la seguridad web de las Pyme tales como:

- La velocidad
- Múltiples conexiones
- Información redundante
- Envío de peticiones y esperar la respuesta para enviar la siguiente
- Protocolo de texto
- Comprensión de cabeceras tardías

- No se dispone de forma inmediata Server Push
- Retardo de flujo de información
- Uso de encriptación TLS (Transport Layer Security)

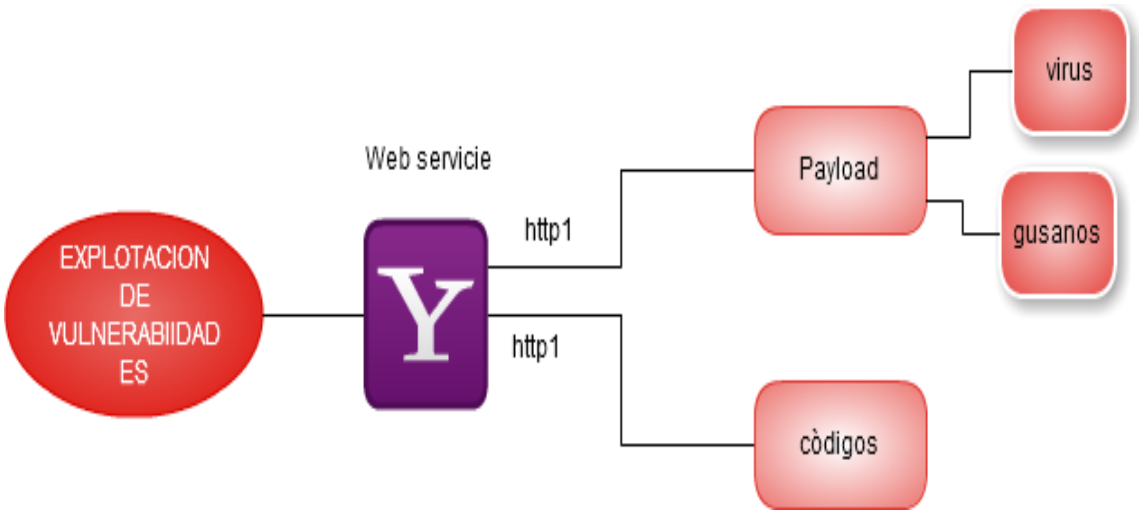


Gráfico 5-3 Fase de explotación de vulnerabilidades de la web de las Pyme
Realizado por: León Isabel, 2016

5.- EVALUACIÓN DE VULNERABILIDADES

Para realizar una evaluación de seguridad en las aplicaciones web de las Pyme, se llevara a cabo una prueba de penetración al sitio web con servidor HTTP 1.1 utilizando software linux y se aplicara criterios de comprobación de errores basándose en indicadores de incidentes de seguridad, esta etapa es la más importante del método ya que detectando las vulnerabilidades del sitio web de las Pyme ya que de acuerdo al estatus de seguridad se puede buscar las mejores características de la evolución de HTTP 1.1 a HTTP 2.0.

6.- SOLUCIÓN DE VULNERABILIDADES

En base a los resultados que se obtienen en la evaluación de vulnerabilidades que se encontraron en la web de las Pyme a través del protocolo HTTP 1.1, ya que es un protocolo que al momento utilizan en su mayoría las empresas medianas y pequeñas por la falta de conocimiento por parte del personal que manejan los sistemas informáticos, es así que con la investigación que se realizó se da a conocer brechas y debilidades que conlleva en si el HTTP 1.1, y llegan a ser un flanco fácil para los hackers que hoy en día roban manipulan información de empresas organizaciones no con el fin de curiosidad sino con el fin de la delincuencia.

Para solucionar los agujeros de la web de las Pyme se aplica las características de seguridad del protocolo HTTP 2.0.

- Como es el mecanismo Padding que oculta el tamaño de la trama, para mantener segura la información sensible.
- Otra característica que se le impone en esta investigación es la limitación del intercambio de stream por conexión ya que son vulnerables a una denegación de servicios porque permite un total de 2^{31} intercambios de stream por conexión.
- También hay otro mecanismo dentro de las características de HTTP 2.0 como el server push, donde el servidor envía información inmediatamente al navegador web incluso antes que sea solicitada como recursos adicionales, es decir el servidor envía varias respuestas en una única solicitud.

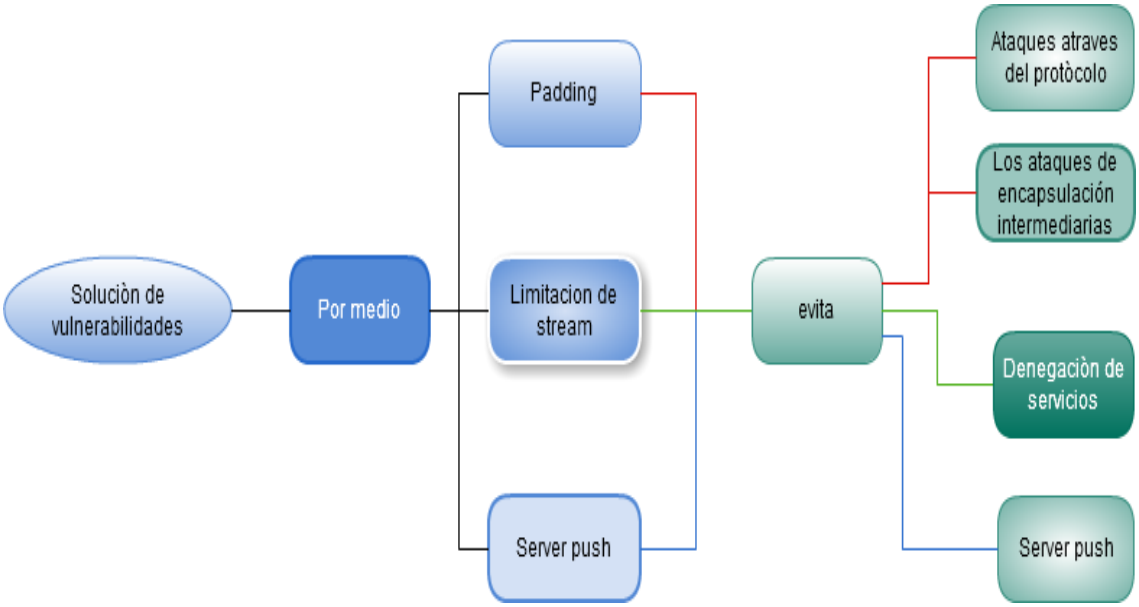


Gráfico 6-3 Fase de mitigación de vulnerabilidades
Realizado por: León Isabel, 2016

7.- VALORACIÓN DE VULNERABILIDADES

En la séptima fase trata de la valoración de las vulnerabilidades que fueron corregidas, se dará un valor cualitativo si la vulnerabilidad persiste se pondrá un valor de activa y si es lo contrario se pondrá un valor de corregida.

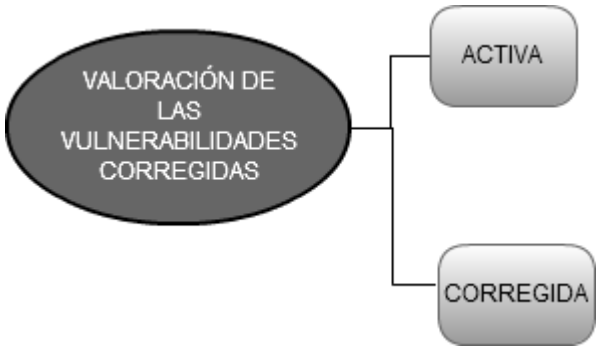


Gráfico 7-3 Fase de valoración de vulnerabilidades corregidas
Realizado por: León Isabel, 2016

La fase 5 “evaluación de vulnerabilidades” en los sitios web utilizando el servidor con HTTP 1.1, es la parte más importante dentro del método propuesto ya que para obtener información de vulnerabilidades por medio de una prueba de penetración y con el fin de no comprometer los sitios web, se presenta la penetración a un sitio web creado para la investigación como complemento al procedimiento propuesto con un seguimiento de la fase 5 en el método propuesto.

Es preciso aclarar que se creó un ambiente de prueba de laboratorio como se muestra en la sección 3.5. Éste ambiente está diseñado intencionalmente con los dos servidores que contienen la configuración de HTTP 1.1 para determinar los errores de seguridad que presenta éste protocolo, y el servidor HTTP 2.0 para mitigar al máximo los errores que se muestre en el precedente.

3.5 Ambiente de laboratorio

Para desarrollar esta mejora de seguridad en los sitios web se ha procedido a montar un escenario práctico como el gráfico 8-3, en la cual por medio de la utilización del método se pretende llegar al objetivo.

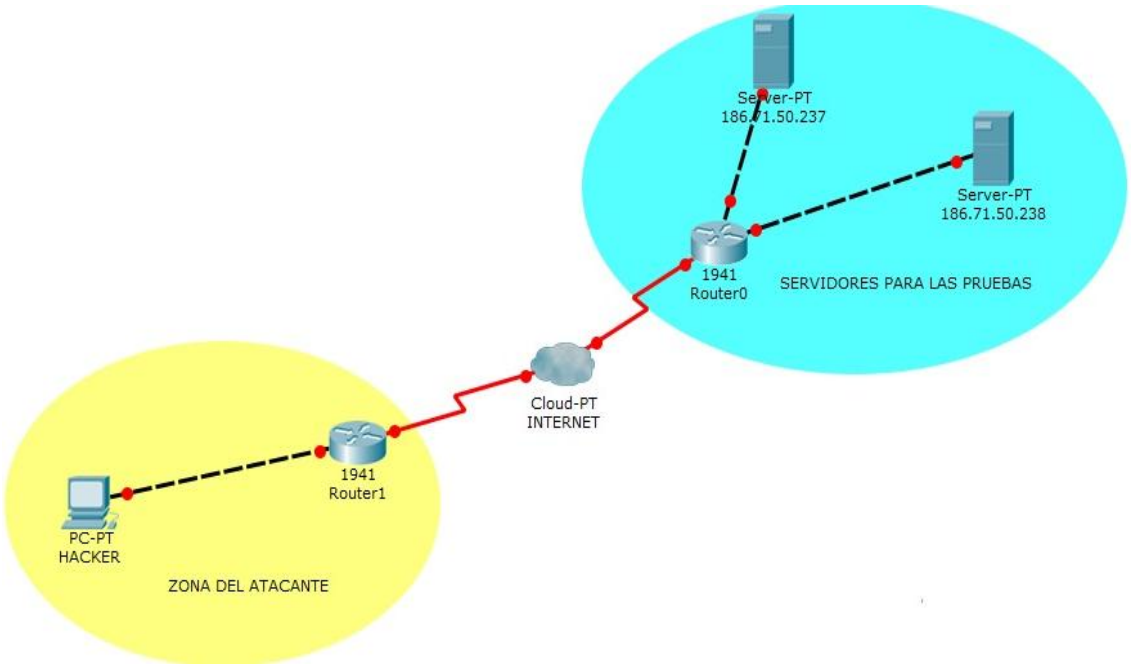


Gráfico 8-3 Escenario de la investigación
Realizado por: León Isabel, 2016

El escenario consta de dos servidores con direcciones IP públicas la mismas que están configurados para el ambiente de laboratorio, el servidor con dirección 186.71.50.237 esta específicamente configurado con el protocolo HTTP 2.0 y el servidor con dirección 186.71.50.238 tiene HTTP 1.1 que se denominarán servidores para las pruebas, la zona del atacante y el internet la interconexión, como refirió el escenario instalado servirá para mostrar las características que tiene el HTTP 2.0 frente al antecesor para mantener la integridad, confidencialidad y privacidad en las web de las pequeñas y medianas empresas y no ser flanco fácil de los atacantes externos o internos.

El proceso en que los dos servidores HTTP 1.1 y HTTP 2.0, comienza con la configuración de cada uno de los servidores, la elaboración de una página web para evidenciar la seguridad y el rendimiento de cada uno de los protocolos utilizados en las Pyme ya que están son las que recurren a la web para realizar transacciones financieras, intercambio de información importantes que necesitan de su integridad.

Para comenzar a identificar las vulnerabilidades que se encuentran presentes en el protocolo HTTP 1.1 se asegura que disponga de lo siguiente:

- ✓ Un equipo que cuente con las herramientas instaladas que permita realizar ataques y configurar la seguridad de los protocolos.
- ✓ Página web para la ejecución de las pruebas.

3.6 Guía de despliegue de los protocolos Http 1.1 y Http 2.0

Se tendrá dos guías de despliegue esto es con el protocolo HTTP 1.1 y con el HTTP 2.0, de la siguiente manera:

3.6.1. Guía de despliegue del protocolo Http 1.1

A continuación, se indica los pasos a seguir para tener en el servidor el protocolo HTTP 1.1:

1. Instalación del sistema operativo en el servidor

Se contrató a la empresa NodoVip el servidor virtual con Linux el mismo que tiene instalado la versión de Fedora 23.

2. Configuración de la dirección IP del servidor

Al servidor se le denominará f2 (f2.local), cuya dirección IP será 186.71.50.238, y funcionará con la configuración por defecto por ende con la versión 1.1 del protocolo.

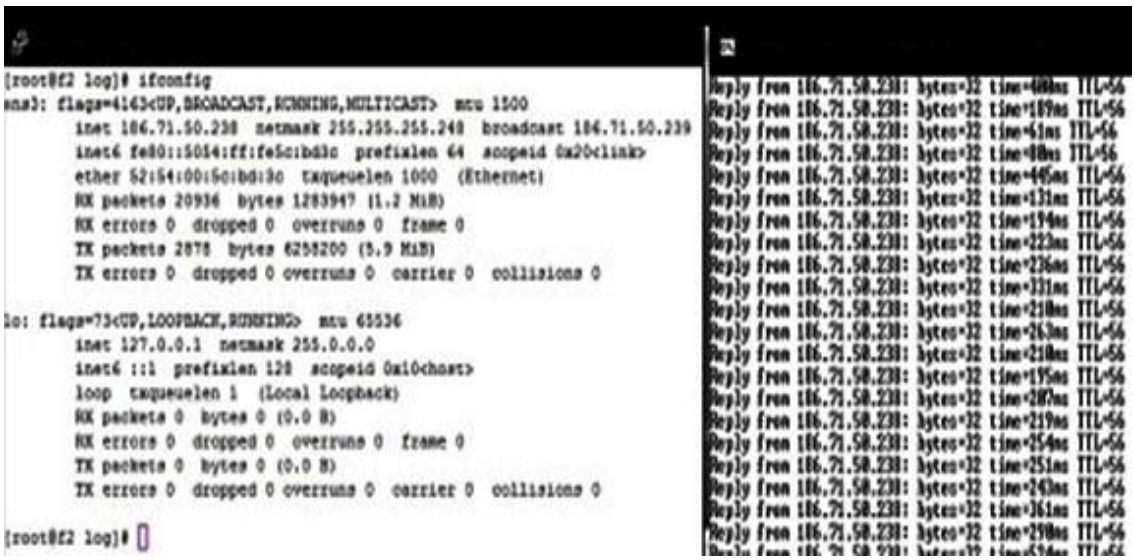


Gráfico 9-3 Configuración de interfaz del servidor HTTP 1.1
Realizado por: León Isabel, 2016

3. Actualización del sistema operativo

Es recomendable actualizar el sistema antes de empezar a utilizarlo ejecutando el comando:

```
[root@f2 ~]# dnf -y update
```

4. Instalación del servidor HTTP Apache

Se procede a instalar Apache con el comando:

```
root@f2 ~]# dnf -y install httpd
```

5. Inicialización del servidor HTTP Apache

Luego de instalar el Apache se debe arrancar el servicio para tenerlo disponible, dicha tarea se realiza con:

```
root@f2 ~]# systemctl start httpd
```

6. Verificación del servidor HTTP Apache

Para verificar el estado del servicio apache recién instalado se necesita digitar:

```
root@f2 ~]# systemctl status httpd
```

Similar al gráfico 14-3 Estado del Servicio Apache en el servidor f1, podemos observar que el servicio está activo con la línea *Active: active (running).*

7. Verificación del funcionamiento del servidor HTTP Apache en el navegador

Para probar el funcionamiento del servidor en el navegador se coloca la IP http://186.71.50.238, y se observará una ventana similar al gráfico 15-3 Página de Prueba del servidor Apache en f1.

8. Verificación del funcionamiento del servidor HTTP 1.1 mediante la CLI

Otra forma de probar el funcionamiento del servidor es mediante la CLI (línea de comandos) desde una consola ejecutando el siguiente comando:

```
[root@rpi2 ~]# curl -v --http2 http://186.71.50.238/
*   Trying 186.71.50.238...
* Connected to 186.71.50.238 (186.71.50.238) port 80 (#0)
> GET / HTTP/1.1
> Host: 186.71.50.238
> User-Agent: curl/7.43.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Mon, 02 May 2016 17:15:50 GMT
< Server: Apache/2.4.18 (Fedora) OpenSSL/1.0.2g-fips
< Last-Modified: Mon, 02 May 2016 17:14:01 GMT
< ETag: "177-531e0d1e35e34"
< Accept-Ranges: bytes
< Content-Length: 375
< Content-Type: text/html; charset=UTF-8
<
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
      <head>
            <title>Página de Prueba</title>
            <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
      </head>
      <body>
            <div>Contenido de prueba</div>
      </body>
</html>
* Connection #0 to host 186.71.50.238 left intact
```

En la salida del comando curl[5], podemos ver con la opción -v los encabezados de la conexión donde se puede verificar que se está usando la versión 1.1 de HTTP. Hasta aquí se tiene el servidor f1 que corresponde a HTTP 1.1 configurado y funcional.

9. Verificación del servidor HTTP 1.1 mediante el sitio web tools.keycdn.com

Usando tools.keycdn.com para el servidor con dirección IP 186.71.50.238, se verifica que no soporta el protocolo HTTP 2.0 sino HTTP 1.1 para lo cual está configurado. En los dos gráficos siguientes se verifica HTTP 1.1 usando http y https, es decir https://186.71.50.238 y http://186.71.50.238.

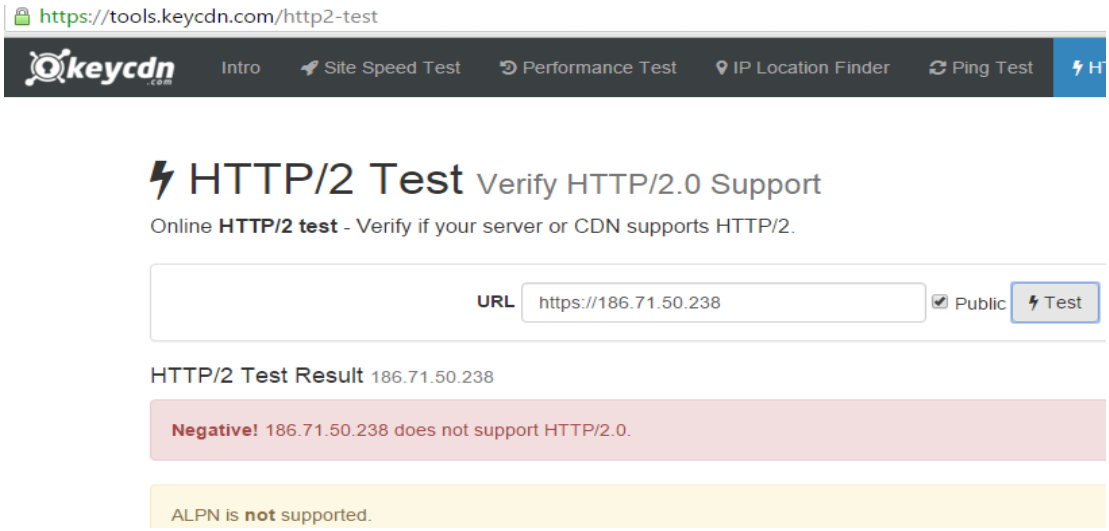


Gráfico 10-3 Verificación de HTTP 1.1 con https en tools.keycdn.com
Realizado por: Isabel León, 2016

[5] Curl es un comando de código abierto que sirve para probar HTTP 2.0 a través de la línea de comandos Disponible en https://curl.haxx.se/

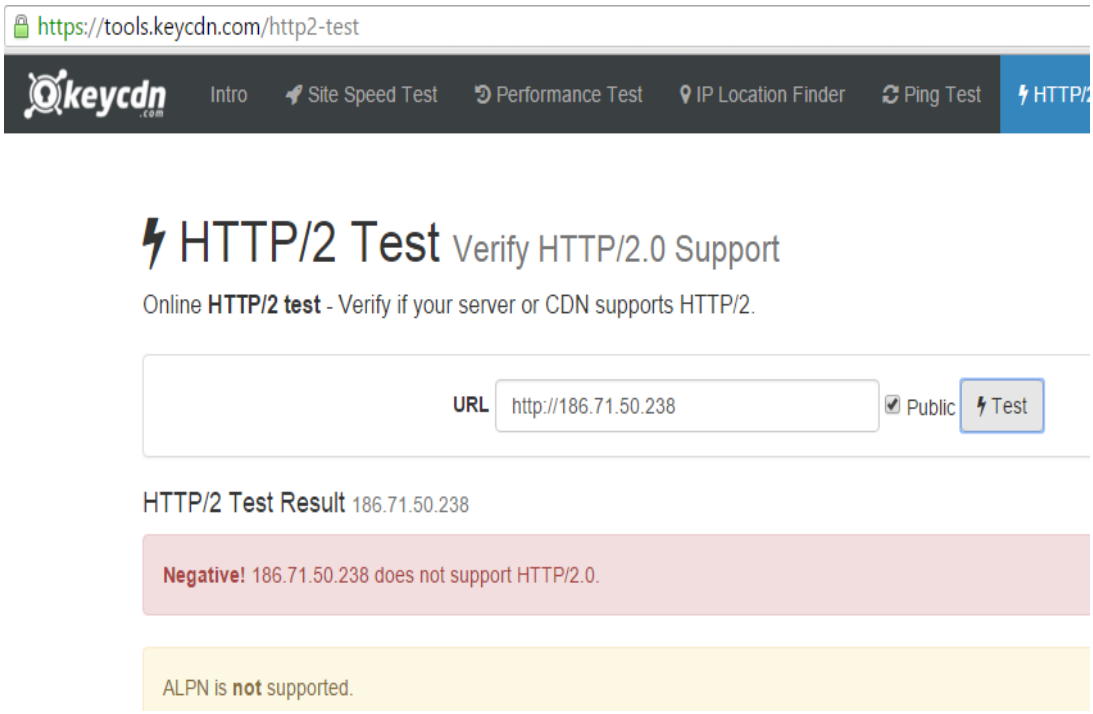


Gráfico 11-3 Verificación de HTTP 1.1 con http en tools.keycdn.com
Realizado por: León Isabel, 2016

3.6.2. Guía de despliegue del protocolo Http 2.0

Los administradores de sistemas de las empresas medianas y pequeñas deberán seguir los pasos de ésta guía para tener en el servidor el despliegue del protocolo HTTP 2.0.

1) Selección e instalación del sistema operativo

Varias empresas seleccionan el software Linux para tenerlo especialmente en servidores que, en computadoras de escritorio, teléfonos, etc.; por ser software libre, código abierto, gratuito y estable, es así que miles de Pymes ya ofrecen productos y servicios basados en ésta tecnología.

En esta investigación no será la excepción y se tendrá un servidor Linux contratado a la empresa Nodovip, dicho servidor tiene instalado Fedora 23 por ser la distribución dispone de las últimas versiones de Apache que tienen soporte para el protocolo HTTP 2.0.

2) Configuración de la dirección IP del servidor

El servidor se denomina f1 (f1.local), con dirección IP 186.71.50.237, y contendrá toda la configuración para soportar HTTP 2.0.

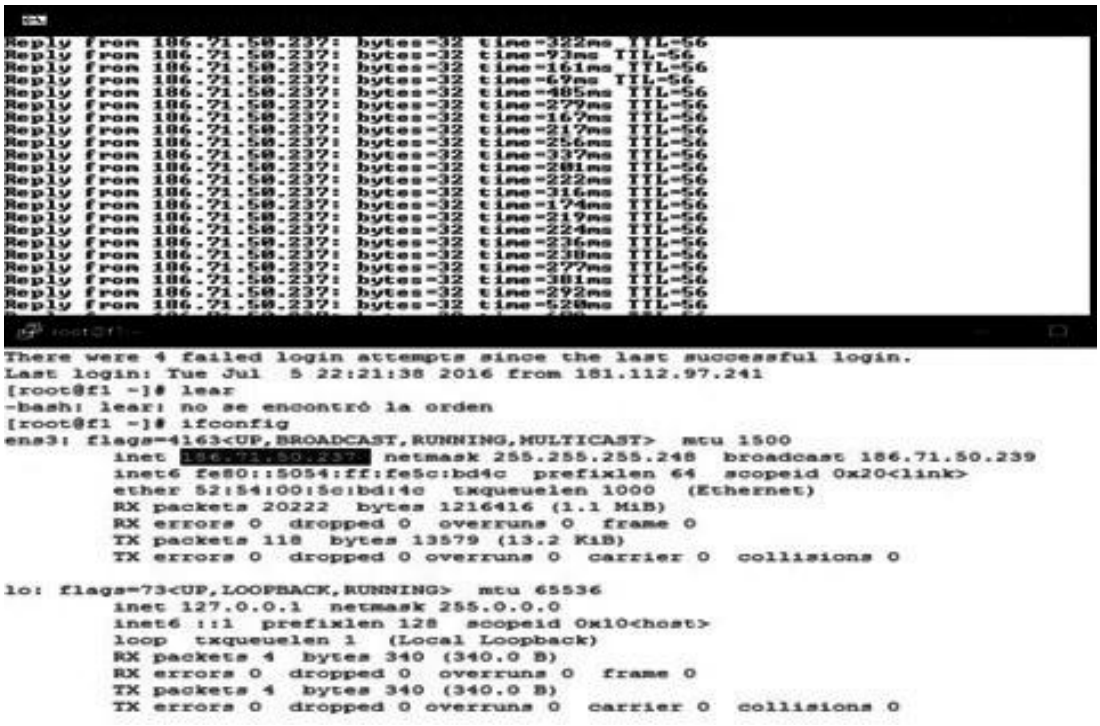


Gráfico 12-3 Configuración de interfaz del servidor HTTP 2.0
Realizado por: León Isabel, 2016

3) Actualización del sistema operativo

Se recomienda luego de tener instalado el sistema operativo actualizarlo con el comando:

```
[root@f1 ~]# dnf -y update
```

4) Instalación del servidor HTTP Apache

El servidor HTTP Apache es un servidor web HTTP de código abierto que implementa el protocolo HTTP 1.1 para plataformas Unix/ Linux y Windows. Usado principalmente para enviar páginas web estáticas y dinámicas en la WWW. El comando para instalar Apache es:

```
[root@f1 ~]# dnf -y install httpd
```

En el gráfico 13-3 se puede observar el proceso de Instalación de Apache en el servidor f1.

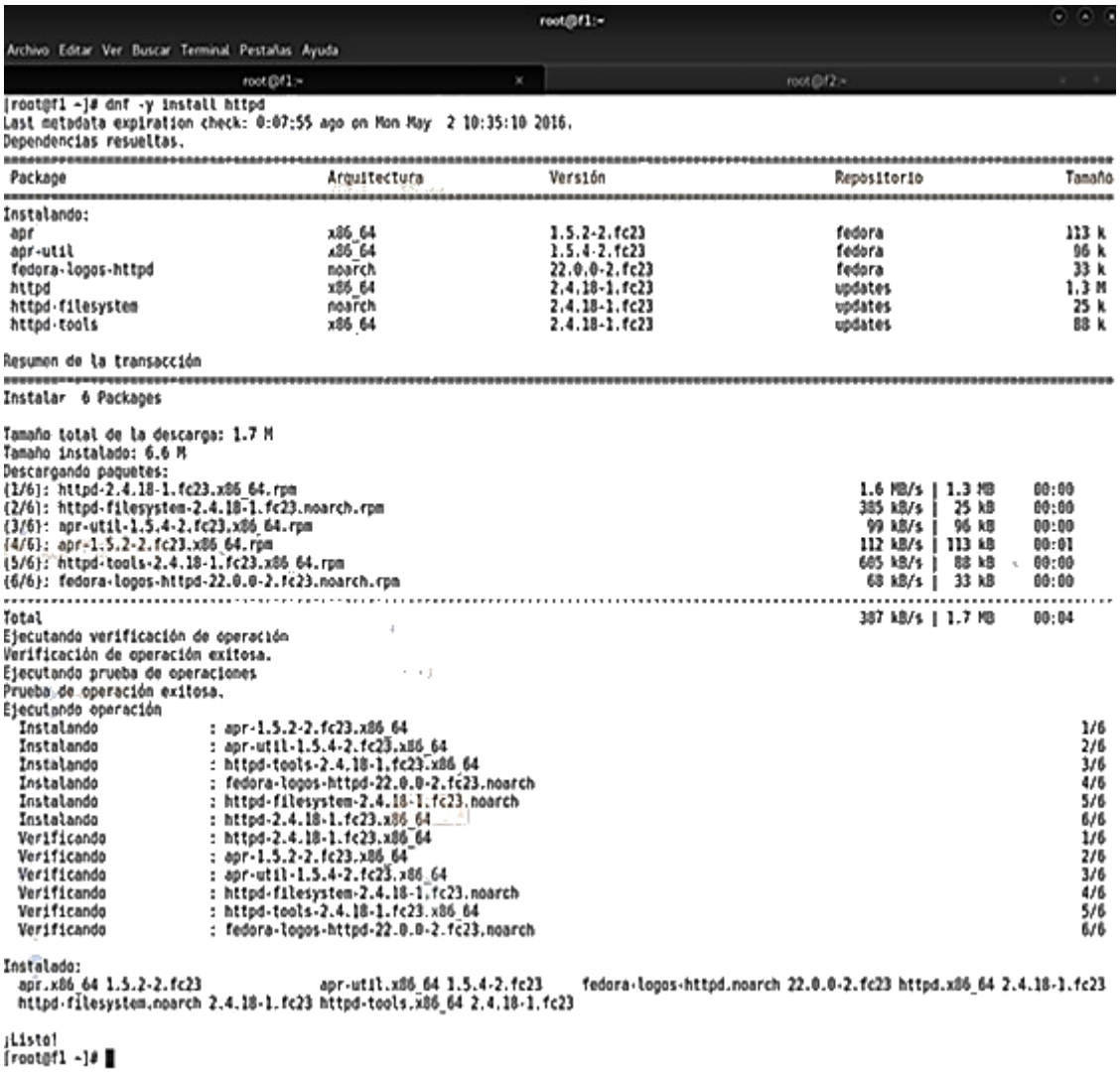


Gráfico 13-3 Instalación de Apache en el servidor f1

Realizado por: León Isabel, 2016

5) Inicialización del servidor HTTP Apache

Una vez instalado Apache se debe arrancar el servicio para tenerlo disponible, ejecutando:

```
[root@f1 ~]# systemctl start httpd
```

6) Comprobación del funcionamiento del servidor Apache

Para comprobar el estado del servicio apache se digita:

```
[root@f1 ~]# systemctl status httpd
```

En el Gráfico 14-3 Estado del Servicio Apache en el servidor f1, se puede ver la salida de la ejecución de los 2 comandos del paso 5 y 6, es decir iniciar el servicio y ver su estado el cual está activo mediante la línea *Active: active (running).*

Gráfico 14-3 Estado del Servicio Apache en el servidor f1
Realizado por: León Isabel, 2016

7) Verificación del funcionamiento del servidor HTTP Apache en el navegador

Se puede probar el correcto funcionamiento del servidor, abriendo la dirección IP del equipo en un navegador (en este caso http://186.71.50.237), y tendremos la página funcional como se puede observar en el Gráfico 15-3 Página de Prueba del servidor Apache en f1.

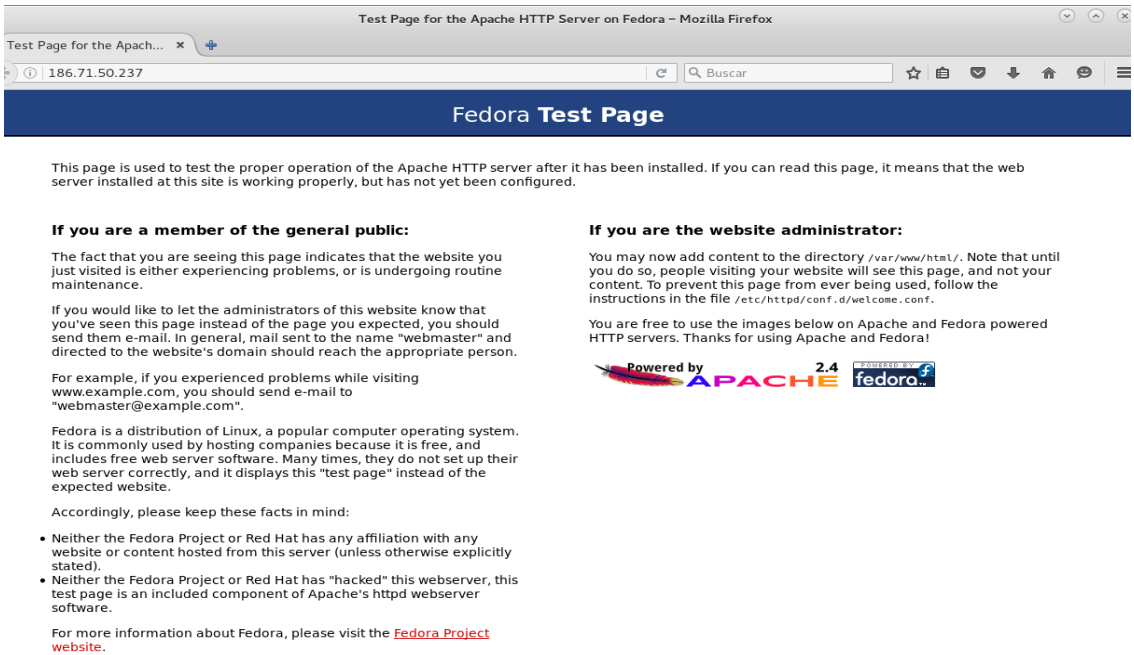


Gráfico 15-3 Página de Prueba del servidor Apache en f1
Realizado por: León Isabel, 2016

Como se puede observar, no hace falta ningún paso adicional a la instalación y arranque del servidor Apache para poder ver la página de prueba y validar su funcionamiento. Es importante notar que por ahora el servidor está configurado de forma estándar, es decir sin SSL ni HTTP 2.0 sino con HTTP 1.1.

8) Verificación del funcionamiento del servidor HTTP 1.1 mediante la CLI

También se puede probar el funcionamiento del servidor mediante la CLI desde una consola ejecutando el comando:

```
[root@rpi2 ~]# curl -v --http2 http://186.71.50.237/
*   Trying 186.71.50.237...
* Connected to 186.71.50.237 (186.71.50.237) port 80 (#0)
> GET / HTTP/1.1
> Host: 186.71.50.237
> User-Agent: curl/7.43.0
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAAQAAP__
>
< HTTP/1.1 200 OK
< Date: Mon, 02 May 2016 17:15:50 GMT
< Server: Apache/2.4.18 (Fedora) OpenSSL/1.0.2g-fips
< Last-Modified: Mon, 02 May 2016 17:14:01 GMT
< ETag: "1d0-531df1e6e768e"
< Accept-Ranges: bytes
< Content-Length: 464
< Content-Type: text/html; charset=UTF-8
<
<!DOCTYPE    html    PUBLIC    "-//W3C//DTD    XHTML    1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
      <head>
            <title>Página de Prueba</title>
            <meta  http-equiv="Content-Type"  content="text/html;
charset=UTF-8" />
      </head>
      <body>
            <div>Contenido de prueba</div>
      </body>
</html>
* Connection #0 to host 186.71.50.237 left intact
```

Se puede comprobar que se está usando la versión 1.1 de HTTP observando la línea subrayada “HTTP/1.1 200 OK”, ya que aún no se implementa HTTP 2.0.

9) Habilitación de HTTP 2.0

La primera versión de Apache que incluye el módulo de HTTP 2.0 es la 2.4.17. El módulo se denomina mod_http2 y anteriormente se le conoció como mod_h2. En los dos servidores de prueba que están instalados con la distribución Fedora 23 se dispone de la versión 2.4.18 que viene directamente incluida. Para ver la versión instalada se digita el comando:

```
[root@f1 ~]# rpm -q httpd
httpd-2.4.18-1.fc23.x86_64
```

Para versiones anteriores de Apache, se puede usar una implementación beta del protocolo SPDY que se puede encontrar en:

- //Debian/Ubuntu (32-bit) https://dl-ssl.google.com/dl/linux/direct/mod-spdy-beta_current_i386.deb
- //Debian/Ubuntu (64-bit) https://dl-ssl.google.com/dl/linux/direct/mod-spdy-beta_current_amd64.deb
- //CentOS/Fedora (32-bit) https://dl-ssl.google.com/dl/linux/direct/mod-spdy-beta_current_i386.rpm
- //CentOS/Fedora (64-bit) https://dl-ssl.google.com/dl/linux/direct/mod-spdy-beta_current_x86_64.rpm

10) Configurar HTTP 2.0 en Apache sin soporte SSL

Con la implementación oficial de HTTP 2.0 en Apache, la habilitación del protocolo se resume a agregar/modificar la siguiente línea en el archivo de configuración:

```
Protocols h2c http/1.1
```

Para esto, editamos el archivo de configuración principal de Apache con:

```
root@f1 ~]# vi /etc/httpd/conf/httpd.conf
```

Y agregamos la línea detallada al final del archivo.

11) Reiniciar el servidor Apache

Con el cambio ejecutado en el paso 10, se requiere reiniciar el servicio para que los cambios surtan efecto:

```
[root@f1 ~]# systemctl restart httpd
```

12) Verificación del funcionamiento del servidor HTTP mediante la CLI luego de configurar el protocolo HTTP 2.0

Una vez configurado en el servidor el protocolo HTTP 2.0 se procede con la prueba de verificación por medio del comando curl:

```
[root@rpi2 ~]# curl -v --http2 http://186.71.50.237/
*   Trying 186.71.50.237...
* Connected to 186.71.50.237 (186.71.50.237) port 80 (#0)
> GET / HTTP/1.1
> Host: 186.71.50.237
> User-Agent: curl/7.43.0
> Accept: */*
> Connection: Upgrade, HTTP2-Settings
> Upgrade: h2c
> HTTP2-Settings: AAMAAABkAAQAAP__
>
< HTTP/1.1 101 Switching Protocols
< Upgrade: h2c
< Connection: Upgrade
* Received 101
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer
after upgrade: len=28
* http2_recv: 16384 bytes buffer at 0x55ac8a34 (stream 1)
* http2_recv: 16384 bytes buffer at 0x55ac8a34 (stream 1)
* http2_recv: returns 593 for stream 1
< HTTP/2.0 200
< date:Mon, 02 May 2016 17:20:52 GMT
< server:Apache/2.4.18 (Fedora) OpenSSL/1.0.2g-fips
< last-modified:Mon, 02 May 2016 17:16:53 GMT
< etag:"153-531df28b0f296"
< accept-ranges:bytes
< content-length:339
< content-type:text/html; charset=UTF-8
<
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
      <head>
            <title>Página de Prueba</title>
            <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
```

```
        </head>
        <body>
              <div>Contendio de prueba</div>
        </body>
</html>
* Connection #0 to host 186.71.50.237 left intact
```

Como se puede ver en la sección resaltada en negrita y curvisa, el protocolo HTTP 2.0 en efecto está funcionando para las solicitudes realizadas al servidor. Sin embargo, para que los navegadores, que son las herramientas cotidianas de uso de la web, puedan abrir correctamente el sitio por medio de HTTP 2.0, se requiere cierto trabajo adicional.

13) Instalación de SSL en el servidor

Si bien el uso de SSL ***no es obligatorio*** para el despliegue de HTTP 2.0, muchos de los navegadores que actualmente soportan HTTP 2.0 sí lo requieren. Esto significa que si bien, por definición no se requiere habilitar SSL para poner HTTP 2.0 a disposición en un servidor; en la práctica se vuelve necesario por requerimiento de los navegadores.

Con el siguiente comando se instala el módulo para el manejo de SSL: `[root@f1 ~]# dnf -y install mod_ssl`

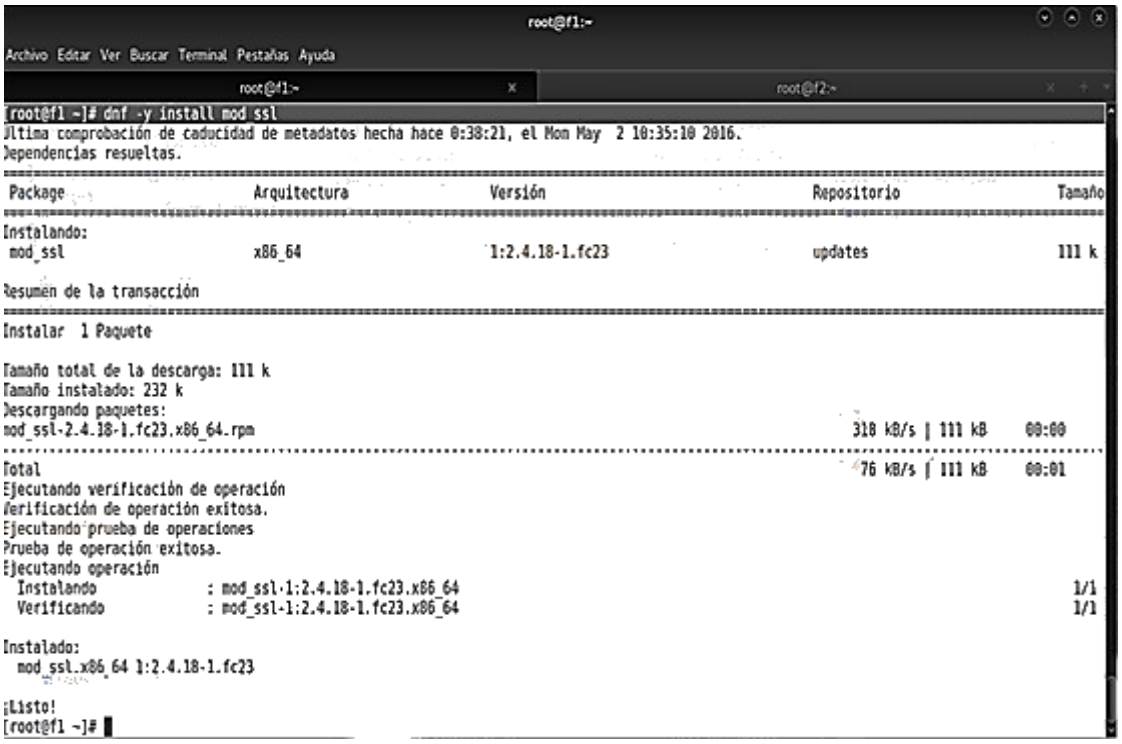


Gráfico 16-3 Instalación de SSL

Realizado por: León Isabel, 2016

14) Verificación del servidor HTTP con SSL

Con el módulo instalado del paso anterior, ya se puede probar que el servidor está atendiendo vía SSL:

```
[root@rpi2 ~]# curl -vk --http2 https://186.71.50.237/
*   Trying 186.71.50.237...
* Connected to 186.71.50.237 (186.71.50.237) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* ALPN, server accepted to use http/1.1
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
* Server certificate:
*     subject:
E=root@f1.local,CN=f1.local,OU=SomeOrganizationalUnit,O=SomeOrga
nization,L=SomeCity,ST=SomeState,C=--
*     start date: may 02 19:06:23 2016 GMT
*     expire date: may 02 19:06:23 2017 GMT
*     common name: f1.local
*     issuer:
E=root@f1.local,CN=f1.local,OU=SomeOrganizationalUnit,O=SomeOrga
nization,L=SomeCity,ST=SomeState,C=--
> GET / HTTP/1.1
> Host: 186.71.50.237
> User-Agent: curl/7.43.0
> Accept: */*
>
 < HTTP/2.0 200 OK
< Date: Mon, 02 May 2016 19:21:26 GMT
< Server: Apache/2.4.18 (Fedora) OpenSSL/1.0.2g-fips
< Last-Modified: Mon, 02 May 2016 19:15:47 GMT
< ETag: "168-531e0d13cf326"
< Accept-Ranges: bytes
< Content-Length: 360
< Content-Type: text/html; charset=UTF-8
<
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
      <head>
            <title>Página de Prueba</title>
            <meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
      </head>
      <body>
                <h1>Server F1</h1>
```

```
            <div>Contenido de prueba</div>
        </body>
</html>
* Connection #0 to host 186.71.50.237 left intact
```

15) Implementación de SSL en HTTP 2.0

Para implementar el protocolo HTTP 2.0 a través de SSL se requiere editar el siguiente archivo:

```
root@f1 ~]# vi /etc/httpd/conf.d/ssl.conf
```

En dicho archivo se busca la sección `<VirtualHost _default_:443>` y se debe poner justo debajo la línea `Protocols`, como se detalla a continuación:

```
<VirtualHost _default_:443>
    Protocols h2 http/1.1
... el resto del archivo ...
```

16) Reinicio del servidor Apache

Con este sencillo cambio realizado en el paso anterior, el servidor Apache ya está habilitado para entregar HTTP 2.0 sobre SSL, aunque se requiere reiniciar el servicio para que los cambios surtan efecto:

```
root@f1 ~]# systemctl restart httpd
```

17) Verificación del funcionamiento del servidor HTTP 2.0 con SSL mediante la CLI

Realizamos la prueba respectiva con curl:

```
[root@rpi2 ~]# curl -vk --http2 https://186.71.50.237/
*   Trying 186.71.50.237...
* Connected to 186.71.50.237 (186.71.50.237) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* ALPN, server accepted to use h2
* SSL connection using TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
* Server certificate:
*  subject:
E=root@f1.local,CN=f1.local,OU=SomeOrganizationalUnit,O=SomeOrganiz
ation,L=SomeCity,ST=SomeState,C=--
*  start date: may 02 16:13:36 2016 GMT
*  expire date: may 02 16:13:36 2017 GMT
*  common name: f1.local
*  issuer:
E=root@f1.local,CN=f1.local,OU=SomeOrganizationalUnit,O=SomeOrganiz
```

```
ation,L=SomeCity,ST=SomeState,C=--
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after
upgrade: len=0
* Using Stream ID: 1 (easy handle 0x563854a0)
> GET / HTTP/1.1
> Host: 186.71.50.237
> User-Agent: curl/7.43.0
> Accept: */*
>
 * http2_recv: 16384 bytes buffer at 0x56385a34 (stream 1)
* HTTP/2 stream 1 was not closed cleanly: error_code = 8
* Closing connection 0
curl: (16) HTTP/2 stream 1 was not closed cleanly: error_code = 8
```

Pero la prueba arroja un error en la secuencia HTTP 2.0, como se puede observar en el texto resaltado con negrita y cursiva.

18) Resolver el problema que arroja HTTP 2.0

Dicho error observado en el paso anterior se resuelve agregando/modificando las siguientes configuraciones en el mismo archivo `/etc/httpd/conf.d/ssl.conf`:

```
SSLCipherSuite      ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES128-GCM-
SHA256:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-AES256-GCM-SHA384:DHE-
RSA-AES128-GCM-SHA256:DHE-DSS-AES128-GCM-SHA256:kEDH+AESGCM:ECDHE-RSA-
AES128-SHA256:ECDHE-ECDSA-AES128-SHA256:ECDHE-RSA-AES128-SHA:ECDHE-
ECDSA-AES128-SHA:ECDHE-RSA-AES256-SHA384:ECDHE-ECDSA-AES256-
SHA384:ECDHE-RSA-AES256-SHA:ECDHE-ECDSA-AES256-SHA:DHE-RSA-AES128-
SHA256:DHE-RSA-AES128-SHA:DHE-DSS-AES128-SHA256:DHE-RSA-AES256-
SHA256:DHE-DSS-AES256-SHA:DHE-RSA-AES256-
SHA:!aNULL:!eNULL:!EXPORT:!DES:!RC4:!3DES:!MD5:!PSK
```

La anterior es una sola línea, que por ser tan larga aquí se presenta en varias líneas, pero debe agregarse como una sola en el archivo de configuración, junto con la siguiente: `SSLProtocol All -SSLv2 -SSLv3`

19) Reinicio del servidor Apache

Nuevamente se requiere reiniciar el servicio para que los cambios surtan efecto:

```
[root@f1 ~]# systemctl restart httpd
```

20) Verificación del funcionamiento del servidor HTTP 2.0 con SSL

Repetimos la prueba respectiva con curl y se notara que está solventado el error:

```
[root@rpi2 ~]# curl -vk --http2 https://186.71.50.237/
*   Trying 186.71.50.237...
* Connected to 186.71.50.237 (186.71.50.237) port 443 (#0)
* Initializing NSS with certpath: sql:/etc/pki/nssdb
* skipping SSL peer certificate verification
* ALPN, server accepted to use h2
* SSL connection using TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
* Server certificate:
*     subject:
E=root@f1.local,CN=f1.local,OU=SomeOrganizationalUnit,O=SomeOrga
nization,L=SomeCity,ST=SomeState,C=--
*     start date: may 02 16:13:36 2016 GMT
*     expire date: may 02 16:13:36 2017 GMT
*     common name: f1.local
*     issuer:
E=root@f1.local,CN=f1.local,OU=SomeOrganizationalUnit,O=SomeOrga
nization,L=SomeCity,ST=SomeState,C=--
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer
after upgrade: len=0
* Using Stream ID: 1 (easy handle 0x561e24a0)
> GET / HTTP/1.1
> Host: 186.71.50.237
> User-Agent: curl/7.43.0
> Accept: */*
>
 * http2_recv: 16384 bytes buffer at 0x561e2a34 (stream 1)
* http2_recv: 16384 bytes buffer at 0x561e2a34 (stream 1)
* http2_recv: 16384 bytes buffer at 0x561e2a34 (stream 1)
* http2_recv: returns 614 for stream 1
< HTTP/2.0 200
< date:Mon, 02 May 2016 19:34:08 GMT
< server:Apache/2.4.18 (Fedora) OpenSSL/1.0.2g-fips
< last-modified:Mon, 02 May 2016 19:15:36 GMT
< etag:"168-531e0d13cf326"
< accept-ranges:bytes
< content-length:360
< content-type:text/html; charset=UTF-8
<
 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
	<head>
		<title>Página de Prueba</title>
		<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />
	</head>
	<body>
		<h1>Server F1</h1>
		<div>Contenido de prueba</div>
	</body>
</html>
* Connection #0 to host 186.71.50.237 left intact
```

Ya se puede constatar que el servidor responde por el protocolo HTTP 2.0.

21) Validación de HTTP 2.0 mediante el navegador/sitios web con HTTP Y HTTPS

Existen sitios web disponibles en línea que permiten realizar la validación del soporte del protocolo HTTP 2.0 en un navegador web. Se realiza la prueba en https://tools.keycdn.com/http2-test con http con la dirección IP 186.71.50.237.

Gráfico 17-3 Verificación de HTTP 2.0 con http en tools.keycdn.com
Realizado por: León Isabel, 2016

Como se puede ver, la prueba se realizó para la configuración sin soporte para SSL, que en el caso de este sitio sí lo permite, pero también se puede validar la versión SSL, como se puede ver a continuación:

Gráfico 18-3 Verificación de HTTP 2.0 con https en tools.keycdn.com
Realizado por: León Isabel, 2016

22) Verificación de HTTP 2.0 mediante Google Chrome

Otra forma de validar el soporte HTTP 2.0 en el navegador es a través de una extensión para Google Chrome[6] que luego de instalarle, presenta un ícono de un rayo color azul, si es que el sitio lo soporta:

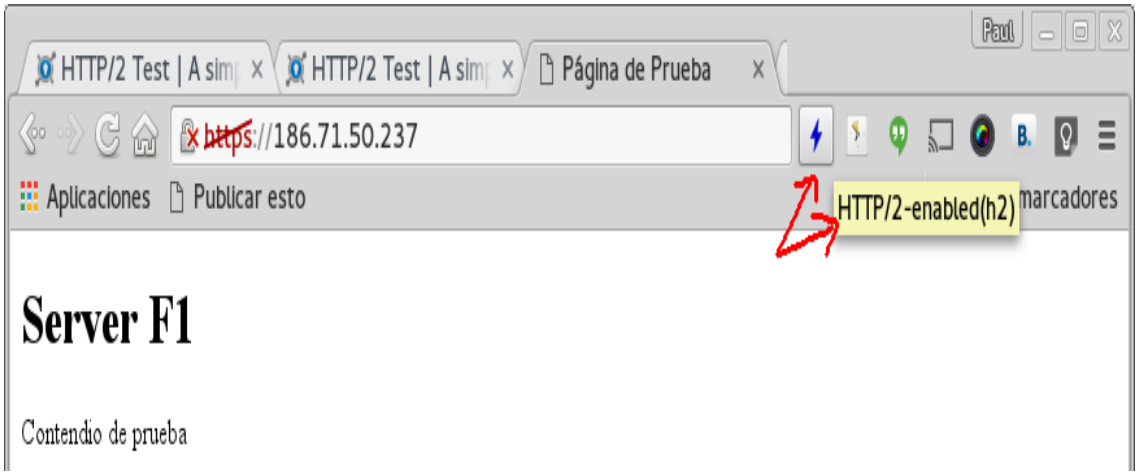
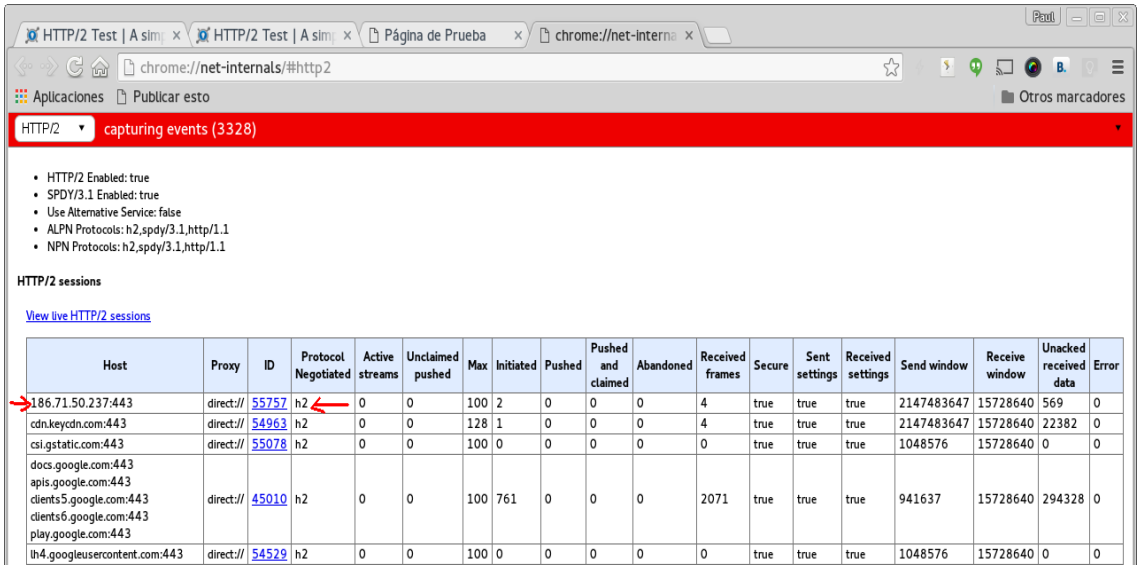


Gráfico 19-3 Verificación de HTTP 2.0 con Google Chrome
Realizado por: León Isabel, 2016

6 Disponible en: https://chrome.google.com/webstore/detail/http2-and-spdy-indicator/mpbpobfflnpcgagjijhmgnchggcjblin?hl=en

Dando clic en el ícono, se puede obtener un detalle de la conexión en cuestión:



Host	Proxy	ID	Protocol Negotiated	Active streams	Unclaimed pushed	Max	Initiated	Pushed	Pushed and claimed	Abandoned	Received frames	Secure	Sent settings	Received settings	Send window	Receive window	Unacked received data	Error
186.71.50.237:443	direct://	55757	h2	0	0	100	2	0	0	0	4	true	true	true	2147483647	15728640	569	0
cdn.keycdn.com:443	direct://	54963	h2	0	0	128	1	0	0	0	4	true	true	true	2147483647	15728640	22382	0
csi.gstatic.com:443	direct://	55078	h2	0	0	100	0	0	0	0	0	true	true	true	1048576	15728640	0	0
docs.google.com:443 apis.google.com:443 clients5.google.com:443 clients6.google.com:443 play.google.com:443	direct://	45010	h2	0	0	100	761	0	0	0	2071	true	true	true	941637	15728640	294328	0
lh4.googleusercontent.com:443	direct://	54529	h2	0	0	100	0	0	0	0	0	true	true	true	1048576	15728640	0	0

Gráfico 20-3 Detalle de conexión HTTP 2.0 con Google Chrome
Realizado por: León Isabel, 2016

Las mismas pruebas las realizamos con el servidor que tiene HTTP 1.1 con IP 186.71.50.238 y se podrá observar que no dispone de soporte para HTTP 2.0 ya que el ícono del rayo está en color gris:

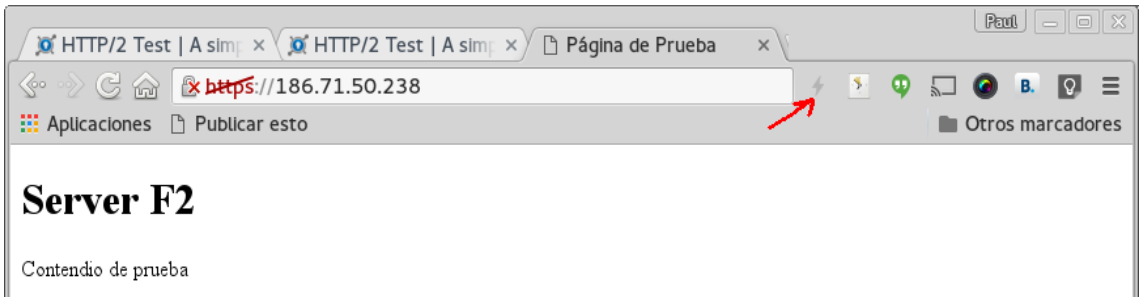


Gráfico 21-3 Sin soporte de HTTP 2.0 con Google Chrome
Realizado por: León Isabel, 2016

Todos estos son los pasos a seguir para tener implementado y funcional el protocolo HTTP 2.0 en el servidor.

3.7 Hipótesis

3.7.1. Determinación de variables

En el presente tema de investigación la hipótesis es:

"La aplicación del método de despliegue de HTTP 2.0 mejorará la seguridad web en las Pyme."

3.7.2. Operacionalización conceptual de variables

La tabla 2-3 muestra la operacionalización conceptual de las variables.

Tabla 2-3 Operacionalización Conceptual de Variables

VARIABLE	TIPO	CONCEPTO
Método de despliegue de HTTP 2.0	Variable Independiente	Son las características incorporadas del nuevo protocolo HTTP 2.0.
Seguridad web en las Pyme	Variable Dependiente	Las conexiones seguras en sitios web dan protección de activos en las Pyme.

Realizado por: León Isabel, 2016

3.7.3. Operacionalización metodológica de variables

La tabla 3-3 muestra la operacionalización metodológica de las variables.

Tabla 3-3 Operacionalización Metodológica de Variables

VARIABLE	INDICADOR	TÉCNICA	INSTRUMENTO/FUENTE
Método de despliegue de HTTP 2.0	▪ Estructura del Método ▪ Características de seguridad ▪ Infraestructura Tecnológica	• Revisión de documentación de HTTP 2.0 • Observación • Guía • Pruebas	Estándar de la IETF "Hypertext Transfer Protocol version 2 -- draft-ietf-httpbis-http2-16" Fedora 23
Seguridad web en las Pyme	Confidencialidad Integridad Disponibilidad Vulnerabilidad	• Pruebas • Análisis comparativo • Resultados	Resultados del análisis y pruebas realizadas a los servidores HTTP 1.1 Y HTTP 2.0 Wireshark 1.12.6, Ettercap 0.8.2, Nessus 5.2.1

Realizado por: León Isabel, 2016

A continuación, una breve descripción de:

Confidencialidad: garantiza que la información esté accesible únicamente a personal autorizado.

Integridad: que la información no pueda ser accedida ni modificada sin contar con la autorización necesaria.

Disponibilidad: aseguramiento de que los usuarios autorizados tengan acceso a la información en cualquier momento.

Vulnerabilidad: son puntos débiles del software que permiten que un atacante comprometa la integridad, disponibilidad o confidencialidad del mismo.

CAPÍTULO IV

RESULTADOS Y DISCUSIÓN

En éste capítulo, se desarrolla las pruebas en los escenarios establecidos, se analiza los resultados obtenidos y se comprueba la hipótesis planteada.

4.1 Aplicación del método de despliegue y mejoramiento de la seguridad web en las Pyme utilizando las características de seguridad de Http 2.0

Como se observó y se explicó el método utilizado para la mejora de seguridad web en las Pyme se procede a su aplicación, el escenario que se ensambló para realizar las pruebas de intrusión en el servidor de las Pyme con protocolo HTTP 1.1, se centra en la explotación de las vulnerabilidades de las aplicaciones y se simulo un ataque externo donde el atacante no conoce sobre ninguna aplicación que maneja las Pyme.

A partir de la información que se obtiene se analiza y se dicta un fallo a la calidad de servicio de seguridad que brinda el personal del área técnica de las Pyme, y por último se recomienda la mitigación de las vulnerabilidades del sitio web de las Pyme con 2.0.

Utilizo la distribución Kali Linux ya que éste sistema operativo permite ejecutar pruebas de penetración, cuenta con herramientas para buscar, controlar y explotar vulnerabilidades.

Gráfico 1-4 Kali Linux
Realizado por: León Isabel, 2016

Se ingresa a la web del servidor 186.71.50.238 donde se observa que tiene una página de prueba que evidencia las características que tienen HTTP 1.1 y sus debilidades en seguridad al realizar una DoS (Denegación de servicios).

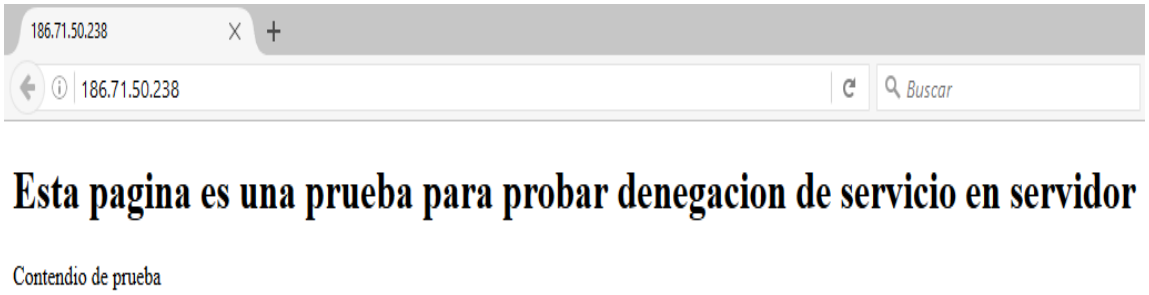


Gráfico 2-4 Web Pyme de prueba en el servidor HTTP 1.1
Realizado por: León Isabel, 2016

Por otra parte, se presenta el proceder de la realización del rastreo de vulnerabilidades en el sitio web de las Pyme que cuentan con el protocolo HTTP 1.1, de tal manera que se complementa con la fase 4, cabe recalcar que las fases de la 1 a la 3 del método de despliegue ya están atendidas puesto que ya han sido debidamente explicadas en el capítulo anterior, por otro lado para realizar la prueba de penetración en las web de las Pyme se han utilizado aplicaciones y herramientas deliberadamente inseguras con el propósito de enseñar las inseguridades de estos sitios. Así mismo en este ambiente creado se cuenta con todos los permisos necesarios para la ejecución de análisis de vulnerabilidades.

ANÁLISIS DE RESULTADOS

Se muestra los pasos por el cual se va a determinar las vulnerabilidades que tiene una Pyme cuando utiliza un servidor con la configuración HTTP 1.1. En la tabla 1-4 respectivamente se especifican los posibles precursores e indicadores que concuerdan con la explotación de vulnerabilidades de los sitios web de las Pyme con HTTP 1.1.

Tabla 1-4 Vulnerabilidades del protocolo HTTP 1.1

VULNERABILIDADES DE LA WEB (PYME) CON EL PROTOCOLO HTTP 1.1		
IP ADDRESS	**VULNERABILIDAD**	**DESCRIPCIÓN**
186.71.50.238	Fully Qualified Domain Name (FQDN)	Da el nombre del computador y del dominio.
186.71.50.238	Inconsistent Hostname and IP Address	El nombre de esta máquina, sea o no resuelve remitir a una dirección IP diferente.
186.71.50.238	Traceroute Information	Hace un traceroute a la máquina remota.
186.71.50.238	Nessus SYN scanner	Este plugin es un escáner de puertos SYN 'semi-abierta'. Será bastante rápido, incluso frente a un objetivo con cortafuegos.
186.71.50.238	SSH Algorithms and Languages Supported	Este script detecta qué algoritmos y lenguajes son soportados por el servicio remoto para el cifrado de comunicaciones.
186.71.50.238	DROWN (Decrypting RSA with Obsolete and Weakened Encryption)	Es una variante del ataque cross-platform, permite interceptar y descifrar conexiones TLS creando conexiones especialmente diseñadas a un servidor de SSLv2 que utiliza la misma clave privada.
186.71.50.238	Las cookies HTTP	Sitios que aplican cookies de autenticación sobre un inicio de sesión HTTPS (protocolo seguro), y luego transmiten las cookies sobre HTTP (protocolo no seguro).

Realizado por: León Isabel, 2016

4.2 Rastreo de vulnerabilidades en el servidor con el protocolo Http 1.1

El programa NESSUS se utilizó en esta investigación para realizar un escaneo de redes y detectar vulnerabilidades dentro de los servidores antes expuestos, se inicia el programa mostrando la pantalla en el gráfico 3-4

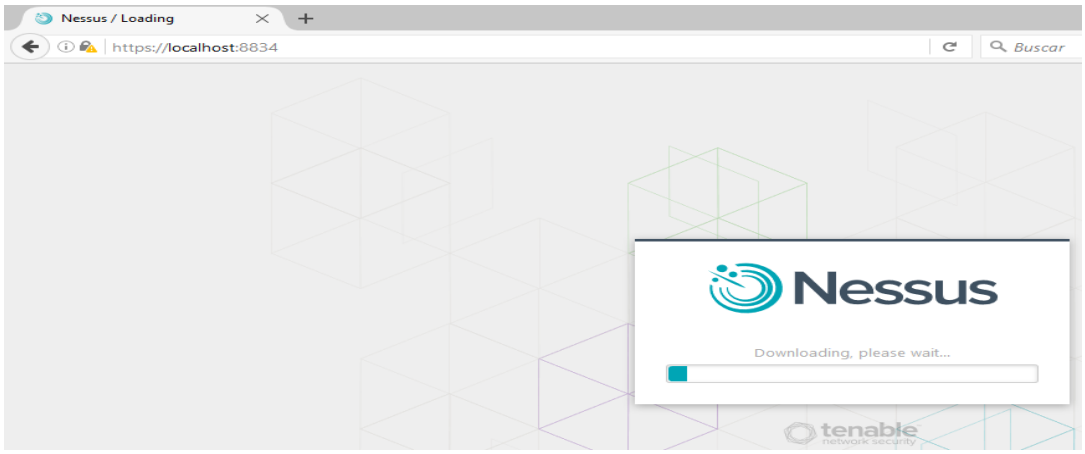


Gráfico 3-4 Nessus
Realizado por: León Isabel, 2016

En el grafico 4-4 como se puede observar se generó un archivo con todas las vulnerabilidades que presento la dirección del servidor HTTP 1.1 con el nombre Tesis Isabel, el archivo generado por el software se lo puede observar en el anexo 1.

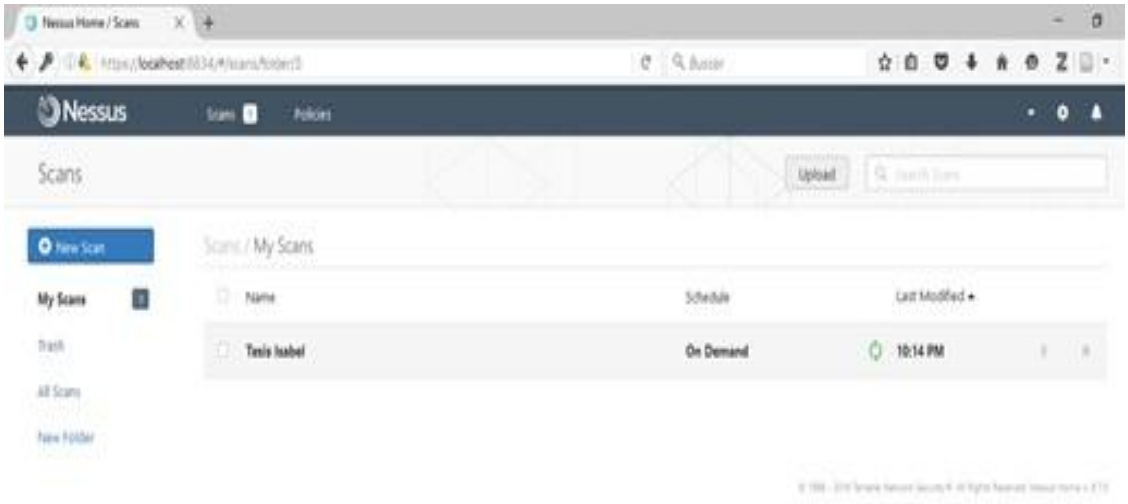


Gráfico 4-4 Ejecución Nessus
Realizado por: León Isabel, 2016

4.3 Análisis de explotación de vulnerabilidades en el protocolo Http 1.1

Las vulnerabilidades que se obtuvieron con el programa nessus en el protocolo HTTP son a nivel de capa 6 nivel de aplicación que como se observara en el anexo 1 presentan deficiencia de seguridad, esto se debe a que existe una gran cantidad de protocolos que existen en esta capa.

Existen una cantidad de deficiencias que se presentan en la capa de aplicación, pero esta investigación se centra a la deficiencia de seguridad que presentan las Pyme en el protocolo HTTP 1.1.

Analizando las vulnerabilidades que están presentes dentro del servidor 186.71.50.238 se muestran los ataques más comunes por parte de los hackers al que es expuesta la web de las Pyme en el servidor HTTP 1.1 para obtener información de las mismas.

HERRAMIENTAS ESPECIALIZADAS QUE DETECTARON LAS VULNERABILIDADES EN LAS WEB DE LAS PYME EN EL SERVIDOR HTTP 1.1

En el gráfico 5-4 indica que se está realizando un ataque de denegación de servicios para tumbar una página web Pyme para lo cual se ocupa un script llamado slowloris, éste trata de mantener varias conexiones con el servidor web de destino abierto y mantenerlas abiertas el mayor tiempo posible, esto da éxito mediante la apertura de conexiones con el servidor web de destino y el empleo de una solicitud parcial.

Periódicamente se envía siguientes cabeceras http las mismas que se van añadiendo, pero nunca se completará el pedido, el servidor afectado seguirá manteniendo las conexiones abiertas llenando a lo máximo de conexiones simultáneas, que con el tiempo ira negando el intento de conexiones adicionales de los clientes de las Pyme.

Gráfico 5-4 Script Slowloris
Realizado por: León Isabel, 2016

Como se está utilizando Kali Linux se ejecuta el script slowloris desde el terminal con la dirección de la página a la que se desea atacar y se desplegara una pantalla con la ejecución del slowloris como se observa en el gráfico 6-4, donde lo primero que hace es construir los paquetes para luego proceder al envío masivo y así poder saturar al servidor con múltiples solicitudes.

Gráfico 6-4 Ejecución del Slowloris
Realizado por: León Isabel, 2016

Después del ataque DoS realizado a la página web de las Pyme del servidor 186.71.50.238, se procede a capturar el tráfico para conocer el origen del incidente para poder tomar medidas y conseguir una correcta protección.

El analizador wireshark ayudara a identificar, analizar, y correlacionar el tráfico identificando las amenazas de red, se realizó una captura de 7100 paquetes como muestra el gráfico 7-4; se puede ver que tiene tres zonas donde se identifica la zona de listado de los paquetes capturados con información de número de frame, tiempo de la captura, origen, destino, protocolo y longitud.

En la segunda zona muestra los datos del frame capturado donde se identifica todos los protocolos involucrados en la captura, y la última zona da a conocer las contraseñas, el sitio no es cifrado; en esta investigación en el momento que se utiliza un filtro donde se consiga el tráfico de http y se encuentren contraseñas será porque de acuerdo a las características identificadas el sitio HTTP 1.1 no es cifrado.

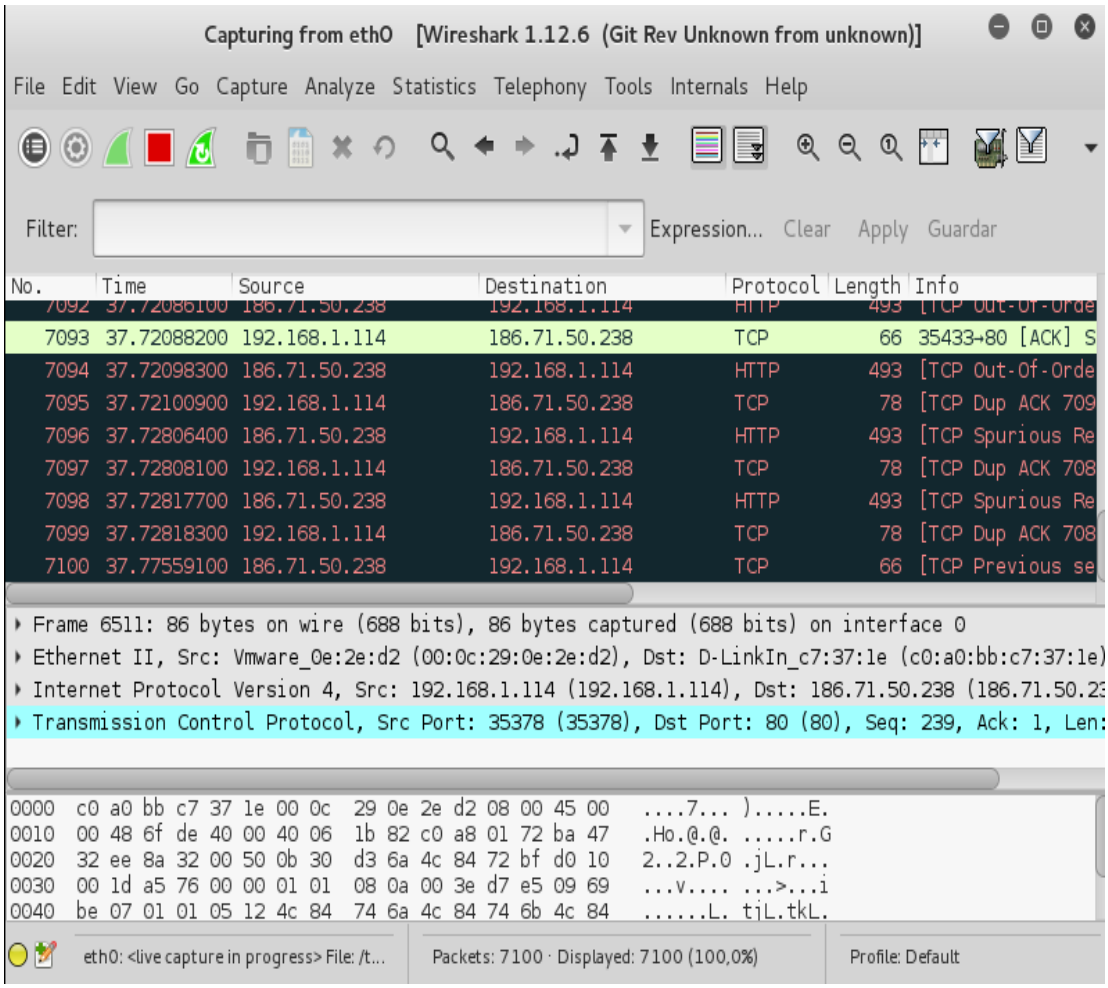


Gráfico 7-4 Análisis de tráfico de red HTTP 1.1
Realizado por: León Isabel, 2016

En el gráfico 8-4 indica el ataque efectuado dando a conocer los frame capturados, tiempo de captura del origen y del destino de los paquetes enviados, y el protocolo utilizado que es el HTTP 1.1.

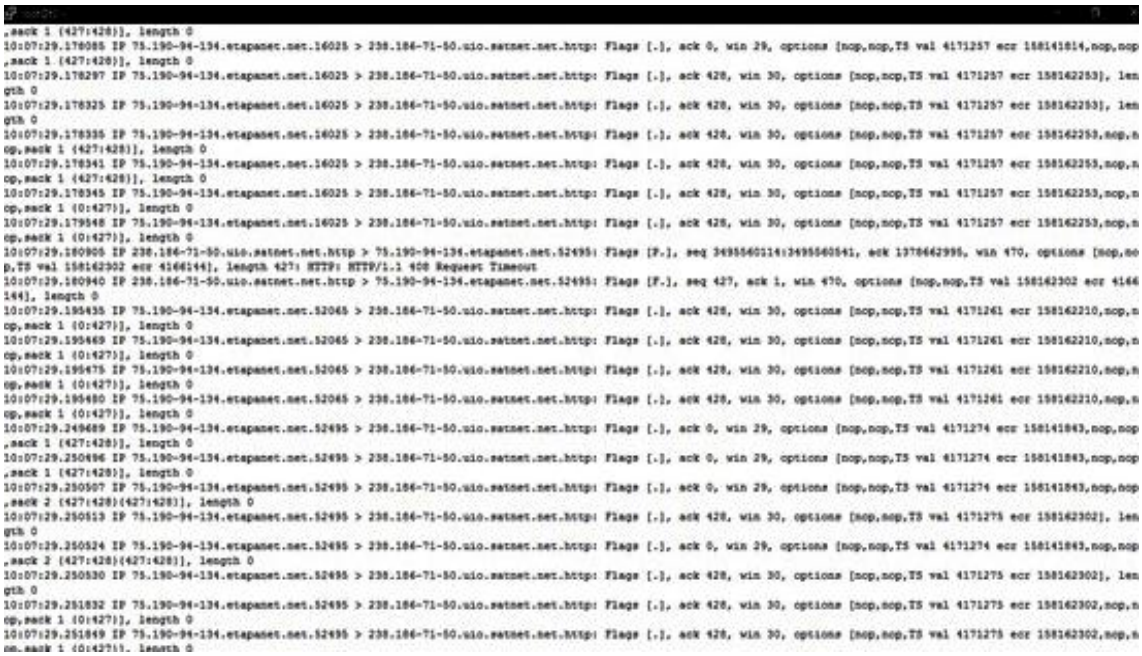


Gráfico 8-4 Escaneo
Realizado por: León Isabel, 2016

En el gráfico 09-4 se observa una inundación de paquetes tcp6, estas conexiones se acumulan haciendo que el proceso sea más lento hasta que deje de funcionar el servidor HTTP 1.1.

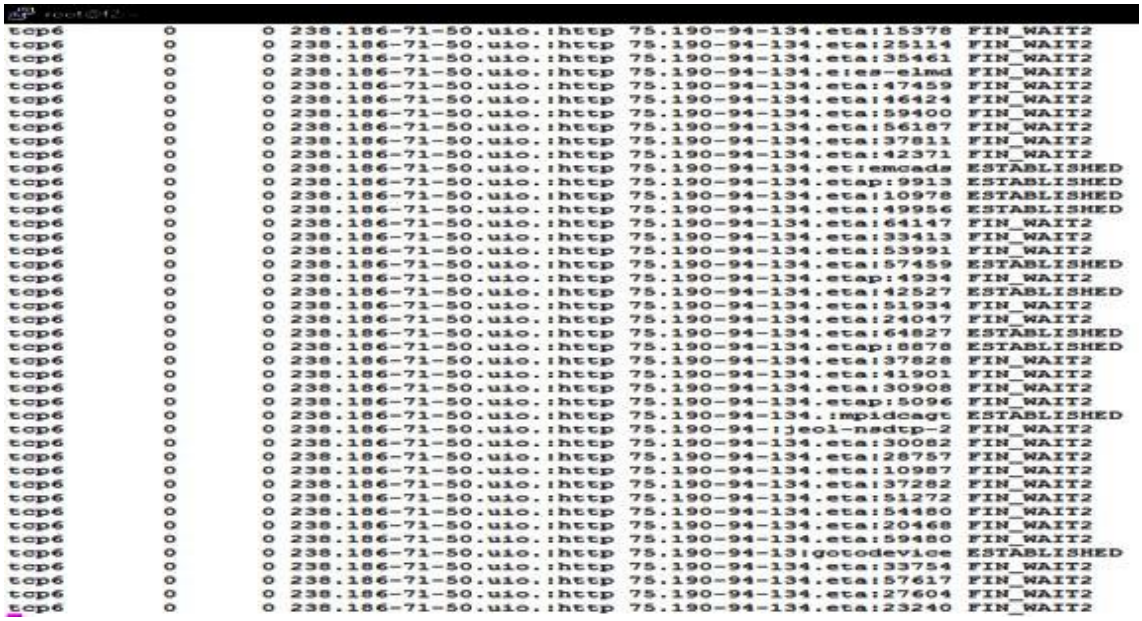


Gráfico 9-4 Paquetes TCP simultáneas
Realizado por: León Isabel, 2016

En el gráfico 10-4 se realiza un ping a la dirección del servidor de HTTP 1.1 para observar la accesibilidad a esta página después de enviar el ataque, como es conocido

la mayor latencia en el tiempo de respuesta se debe a que existe algún problema en la página, se puede visualizar que los tiempos de respuesta suben y es por eso que se identifica problemas en ella.

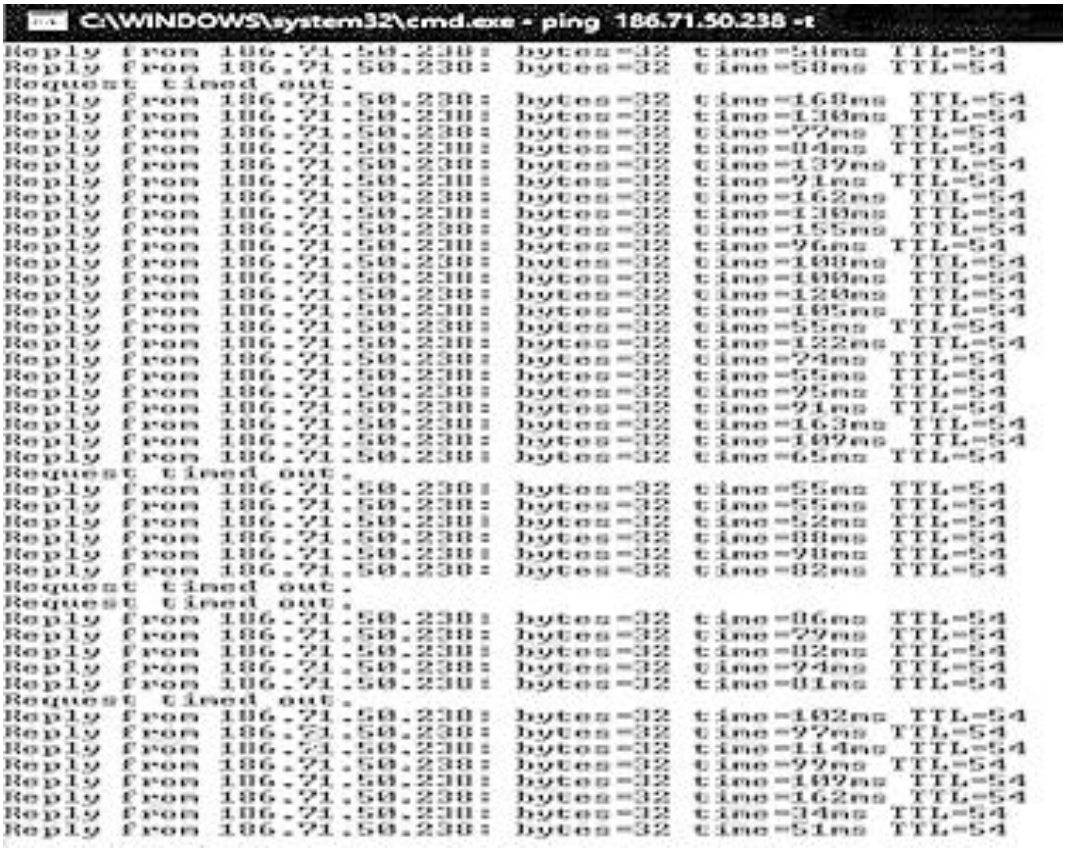


Gráfico 10-4 Saturación de tráfico
Realizado por: León Isabel, 2016

En el gráfico 11-4 se aplica la herramienta netstat -plan|grep :80 de la línea de comandos de Kali Linux que muestra un listado de direcciones ip y el número de conexiones activas por el puerto 80, como se conoce el puerto 80 es utilizado por HTTP de forma continua para realizar peticiones en la web.

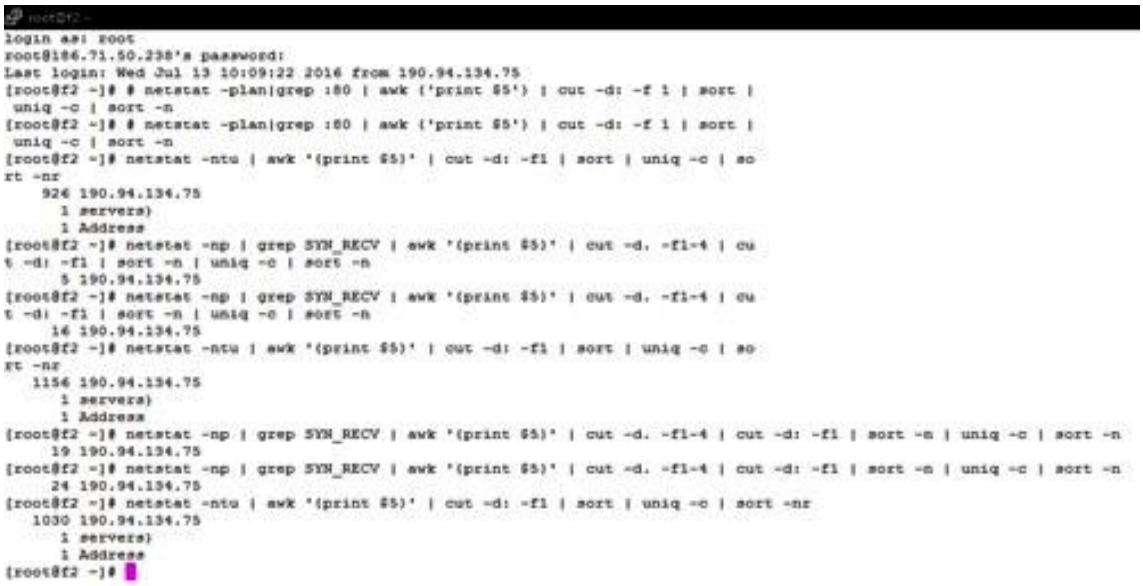


Gráfico 11-4 Conexiones activas en el puerto 80
Realizado por: León Isabel, 2016

Se vuelve aplicar un ping al servidor HTTP 1.1 y como es conocido el ataque que se envió aún sigue afectando a la web con tiempos largos de respuesta como se observa 2,53 minutos.

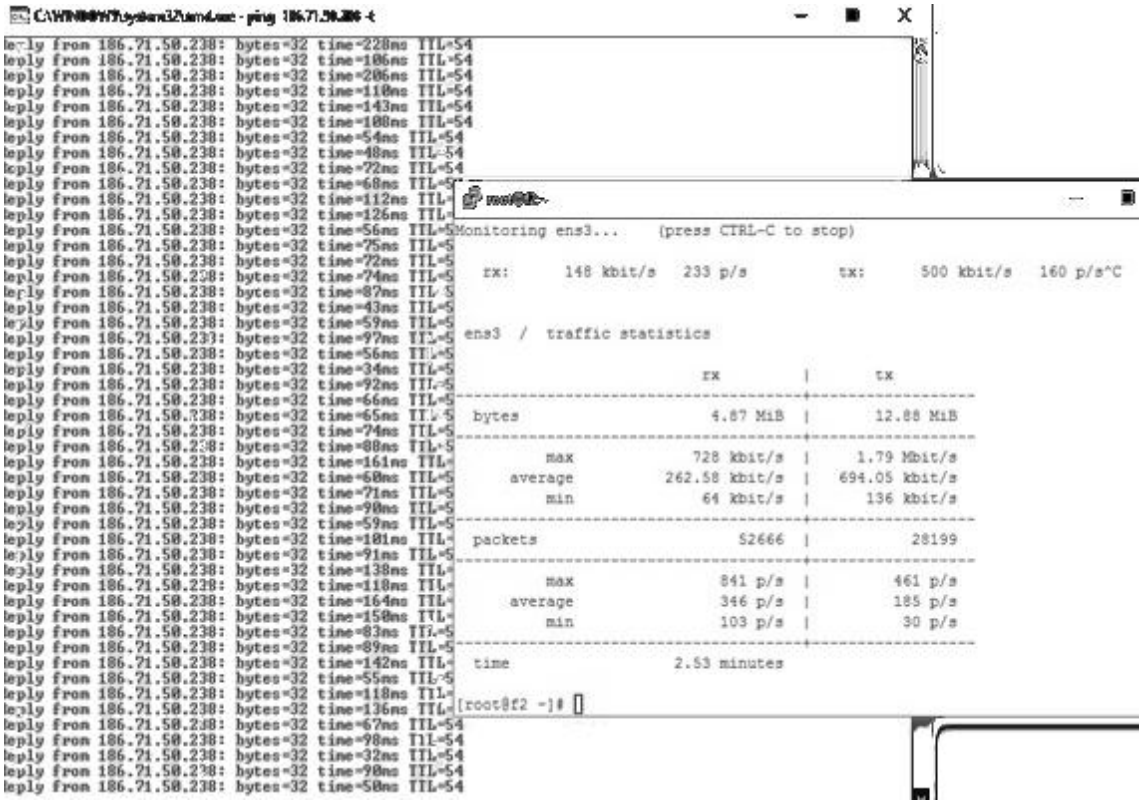


Gráfico 12-4 Acumulación de tráfico en la web Pyme con HTTP 1.1

Realizado por: León Isabel, 2016

El protocolo HTTP se responsabiliza de todos los servicios Word Wide Web, la vulnerabilidad procede cuando existe intercambio de información entre los usuarios de las Pyme con la dirección de la víctima que ha sido atacada, se observa los cambios que se realizan con cada inundación de paquetes que se envía.

En los gráficos 13-4, y 14-4 se observa que llegaron varios paquetes de la inundación que fue víctima éste servidor, obteniendo los resultados que se muestran en la pantalla como son los paquetes máximos, mínimos, y promedios.

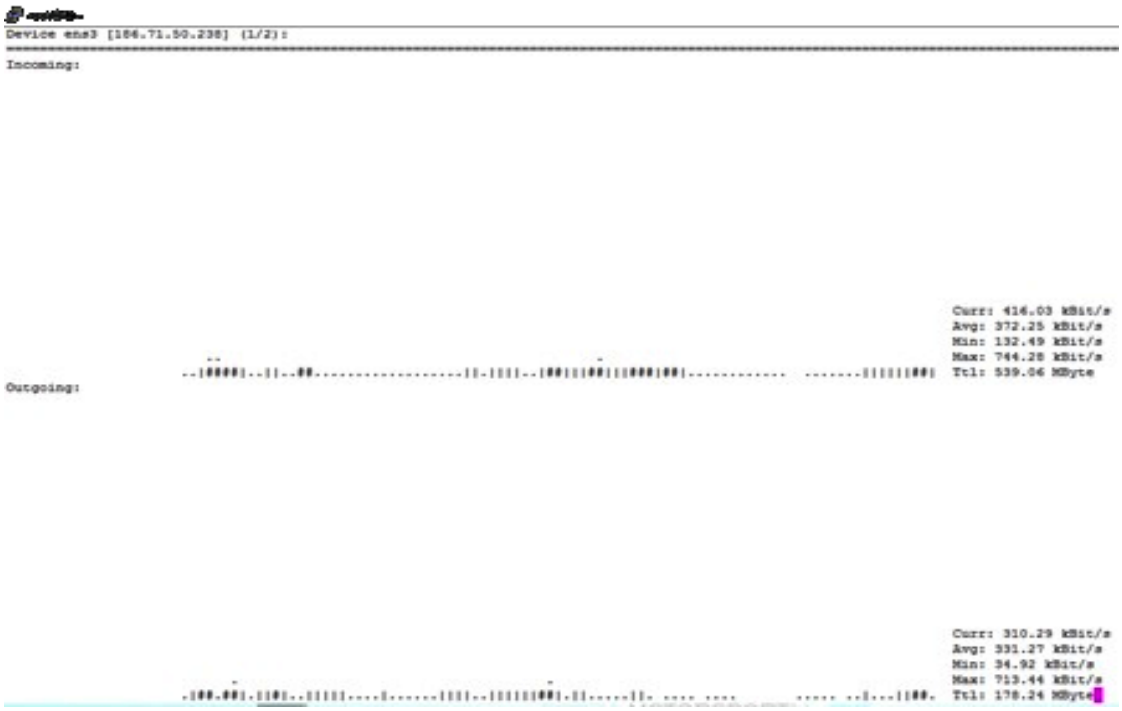


Gráfico 13-4 Ataque en la web Pyme con características HTTP 1.1
Realizado por: León Isabel, 2016

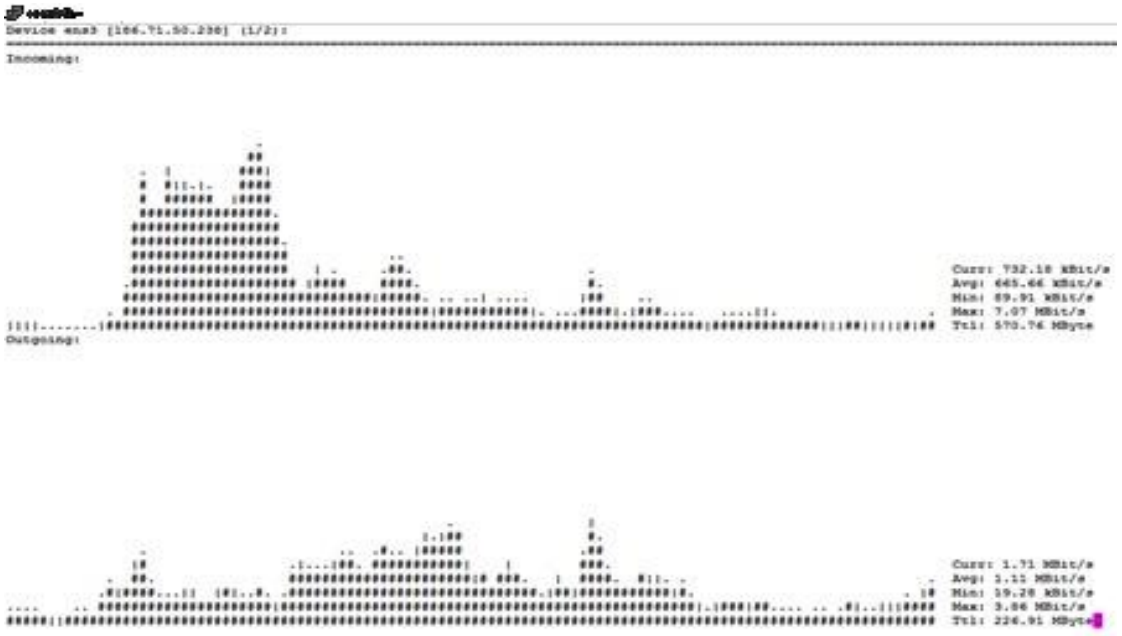


Gráfico 14-4 Análisis paquetes ilegibles en HTTP 1.1
Realizado por: León Isabel, 2016

Como se observa en los ataques realizados con una denegación de servicios se dio a notar el descubrimiento de mucha información acerca de puertos abiertos, información del server como librerías criptográficas, usuarios, idl, etc.

1	2	97	0	0	0	0	0	60k	78k	0	0	394	1797
2	2	95	0	0	1	0	0	78k	130k	0	0	517	2903
2	2	96	0	0	0	0	81928	51k	97k	0	0	436	2268
1	2	96	0	0	1	0	0	61k	132k	0	0	472	2776
usr	sys	idl	wai	hig	sig	read	writ	recv	send	in	out	int	csw
1	3	94	1	0	1	0	12k	64k	121k	0	0	451	2705
2	2	96	0	0	0	0	0	58k	129k	0	0	420	2582
3	3	93	0	0	1	0	0	92k	215k	0	0	571	4154
2	3	95	0	0	0	0	0	58k	142k	0	0	461	2918
1	3	96	0	0	0	0	0	55k	145k	0	0	451	3026
2	3	93	1	0	1	0	12k	70k	174k	0	0	553	3634
1	3	96	0	0	0	0	0	65k	144k	0	0	450	3035
2	3	95	0	0	0	0	0	73k	188k	0	0	563	3943
2	2	95	0	0	1	0	0	44k	105k	0	0	415	2515
1	3	96	0	0	0	0	0	64k	168k	0	0	474	3536
1	1	97	0	0	1	0	12k	36k	73k	0	0	340	1672
1	2	97	0	0	0	0	0	47k	96k	0	0	349	2222
1	2	96	0	0	1	0	0	64k	116k	0	0	420	2697
2	2	96	0	0	0	0	0	55k	98k	0	0	359	2132
1	2	97	0	0	0	0	0	60k	76k	0	0	345	1776
2	2	95	0	0	1	0	0	66k	77k	0	0	380	1777
1	2	96	1	0	0	0	12k	66k	79k	0	0	403	1999
1	1	98	0	0	0	0	0	46k	57k	0	0	344	1524
2	1	97	0	0	0	0	0	66k	95k	0	0	429	2245
1	2	96	0	0	1	0	0	69k	86k	0	0	441	2166
2	2	96	0	0	0	0	0	90k	139k	0	0	510	3087
1	3	95	0	0	1	0	0	73k	135k	0	0	459	2964
2	2	96	0	0	0	0	0	84k	147k	0	0	547	3289
2	3	93	1	0	1	0	12k	88k	155k	0	0	522	3325
3	3	94	0	0	0	0	0	103k	188k	0	0	566	3868
2	2	95	0	0	1	0	0	104k	183k	0	0	543	3578
2	3	95	0	0	0	0	0	82k	175k	0	0	551	3784
2	2	95	0	0	1	0	0	88k	196k	0	0	509	4144
3	3	92	1	0	1	0	112k	92k	187k	0	0	511	4139
2	3	94	0	0	1	0	0	85k	179k	0	0	546	3861
2	3	95	0	0	0	0	0	90k	230k	0	0	562	4676
2	3	94	0	0	1	0	0	80k	165k	0	0	490	3755
2	3	95	0	0	0	0	0	107k	243k	0	0	619	5580
2	2	93	2	0	1	0	20k	52k	103k	0	0	386	2139
1	1	98	0	0	0	0	0	42k	92k	0	0	364	2076
1	3	96	0	0	0	0	0	43k	97k	0	0	386	2327

Gráfico 15-4 Tabla de resultados del ataque de DoS
Realizado por: León Isabel, 2016

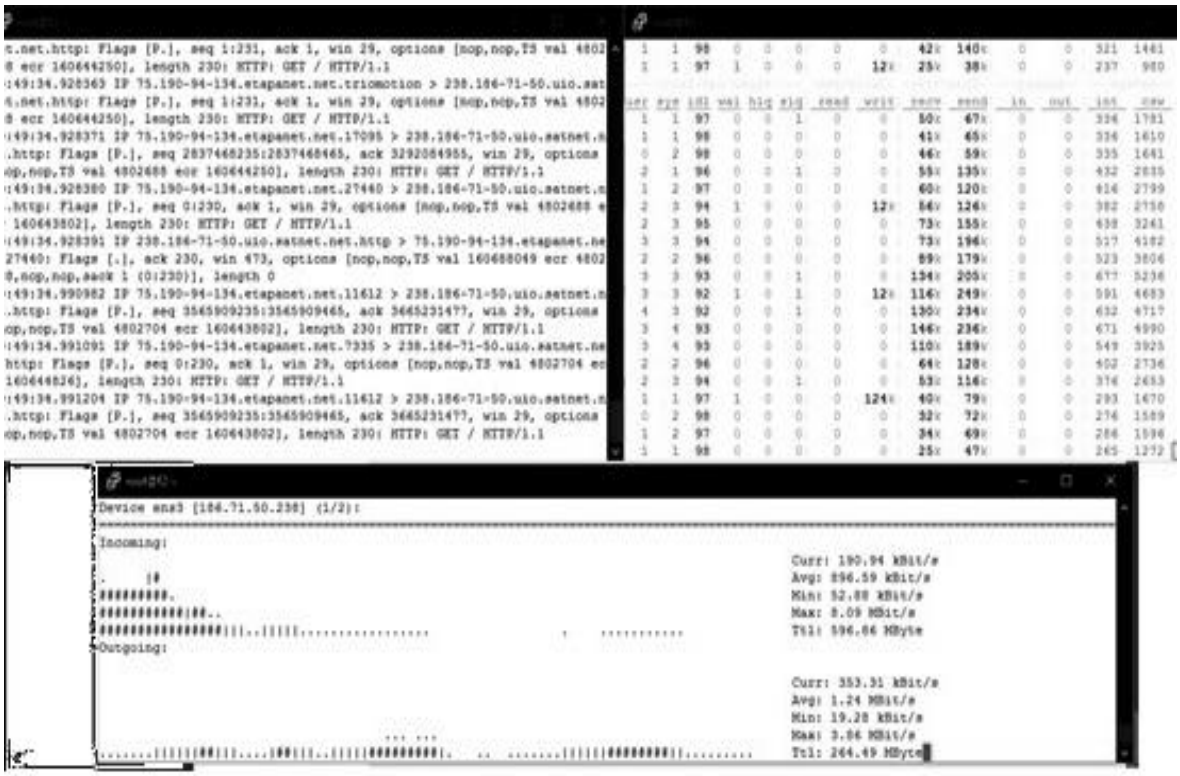


Gráfico 16-4 Explotación a las Pyme y resultados del ataque de DoS
Realizado por: León Isabel, 2016

Luego de explotar las vulnerabilidades de las Pyme del servidor HTTP 1.1 en Kali con el script slowloris, regresamos a la página web de la misma y al recargar observamos que durante unos minutos se va a demorar hasta que el servidor nos da un mensaje de que tiene un error que no puede cargar, dando una petición denegada ya que sigue siendo víctima.

El protocolo HTTP 1.1 envía y recibe información limpia, es decir tal como se envía la información es recibida sin ningún tipo de encriptación o codificación es por eso que en este protocolo no se evidencia el cifrado en la información.

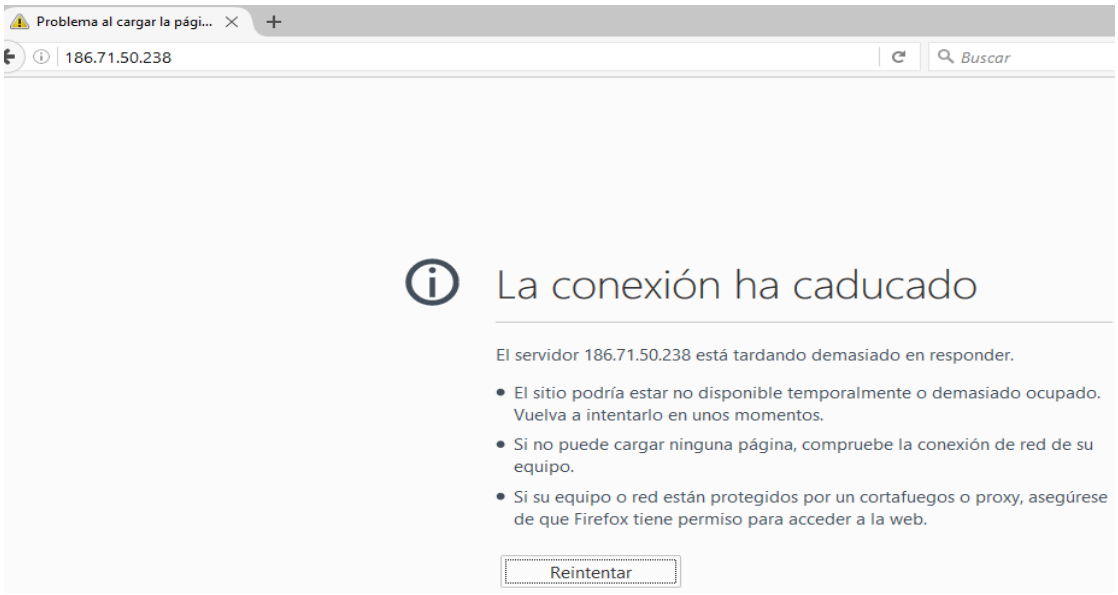


Gráfico 17-4 Ataque del servidor HTTP 1.1
Realizado por: León Isabel, 2016

Se detiene el ataque y se vuelve a intentar entrar a la página web de las Pyme en prueba, y como se observó en la gráfica 17-4 no se restaura del ataque que ha sido víctima, dando a conocer al atacante que esta página no cuenta con certificaciones.

Se realiza un ataque al servidor que cuenta con HTTP 2.0, después del ataque el servidor tiene que resolver las peticiones pendientes por lo que no tarda el servicio en dar una respuesta positiva y se observa que la web no cayó a la denegación de servicios que se realizó como se lo hizo en el servidor que tiene HTTP 1.1.

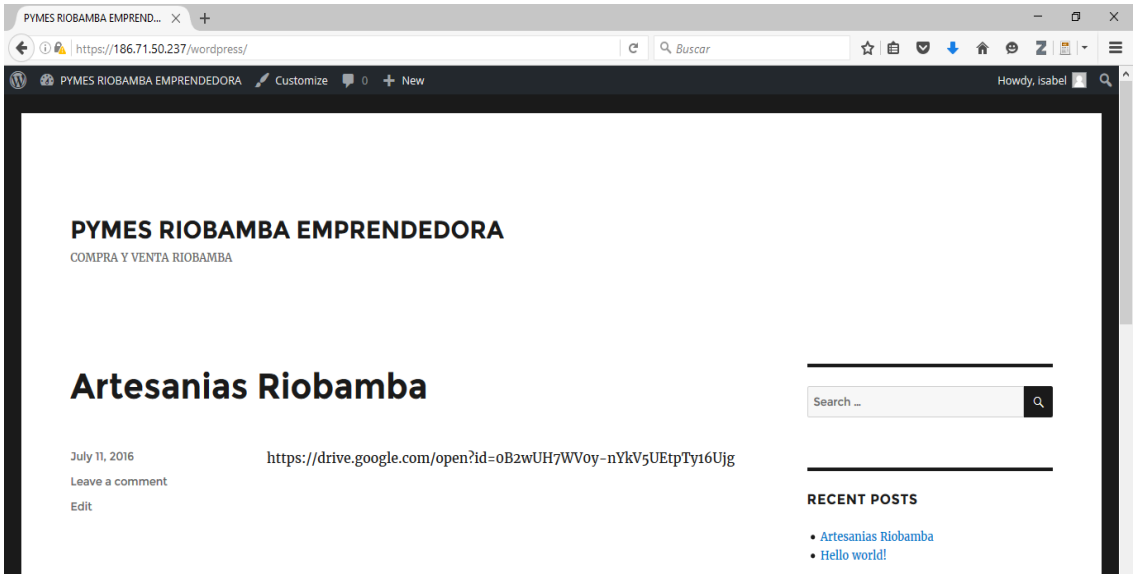


Gráfico 18-4 Web Pyme restablecida
Realizado por: León Isabel, 2016

4.4 Mejoramiento de seguridad web en las Pyme con las características Http 2.0

Una vez realizado el análisis de la inseguridad que tienen las Pyme, con la configuración del HTTP 1.1, a través de la explotación de las vulnerabilidades y conociendo los ataques más frecuentes que utilizan los hackers para el robo de información confidencial, la mejora que pueden hacer éstas empresas, son la utilización de las características de HTTP 2.0 que ayudan a mejorar el rendimiento de la web así como la seguridad bloqueando los agüeros que presenta HTTP 1.1. En la tabla 2-4 se presenta la solución de vulnerabilidades con las características HTTP 2.0.

Tabla 2-4 Solución de vulnerabilidades

SOLUCIÓN DE VULNERABILIDADES DE LA WEB EN EL SERVIDOR HTTP 1.1				
VULNERABILIDAD EN HTTP 1.1 186.71.50.238	**NIVEL DE GRAVEDAD**	**PUNTUACION**	**CONFORMIDAD**	**SOLUCIÓN CON HTTP 2.0 186.71.50.237**
Host Fully Qualified Domain Name (FQDN)	Low	0	Pass	Configurar encriptación, certificados (integridad, confidencialidad)
Inconsistent Hostname and IP Address	Low	0	Pass	Fijar el archivo DNS o host inversa
Traceroute Information	Low	0		n/a
Nessus SYN scanner	Low	0	Pass	Configurando SSL y TSL para proteger la tarjeta con filtrado IP
SSH Algorithms and Languages Supported	Low	0		Se cambia el archivo /etc/sshd_config
DROWN (Decrypting RSA with Obsolete and Weakened Encryption)	Low	0	Pass	Las claves privadas no utilizar en cualquier otro servidor que soporta conexiones SSLv2
Las cookies HTTP	Low	0	Pass	Configurando las cookies Set Cookies: cookie=data; path=/; domain=.aaa.it; secure la información será enviada solo por el canal seguro https

Realizado por: León Isabel, 2016

HERRAMIENTAS ESPECIALIZADAS QUE CORRIGIERON LAS VULNERABILIDADES EN LAS WEB DE LAS PYME CON LAS CARACTERÍSTICAS HTTP 2.0

A continuación, se presentan los siguientes gráficos donde se utilizan las herramientas para detectar las vulnerabilidades de seguridad y de la misma manera se muestra la mejora que se obtienen al utilizar las características de HTTP 2.0.

En las pruebas del servidor 186.71.50.237 que tiene activo HTTP 2.0 se tiene una característica importante la cual acepta una conexión única por cada IP que se conecte, así se lo puede demostrar en el gráfico de abajo.

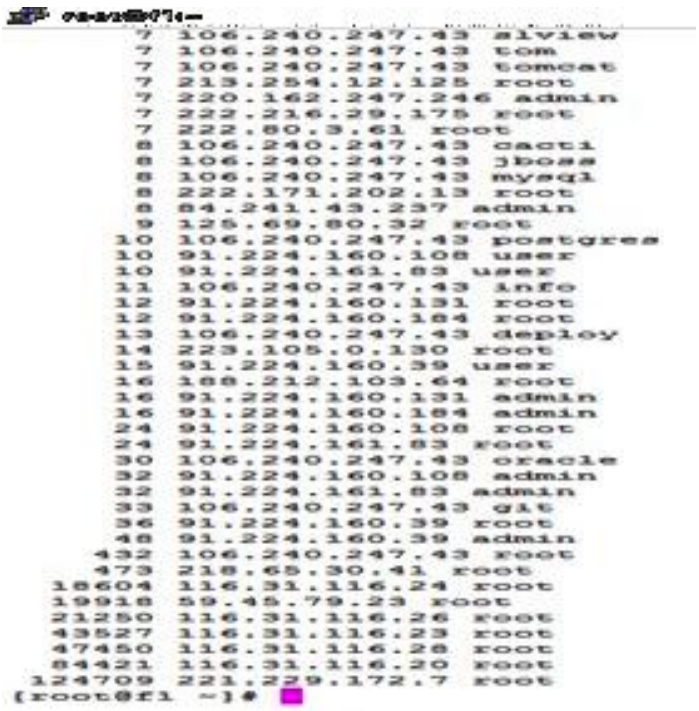
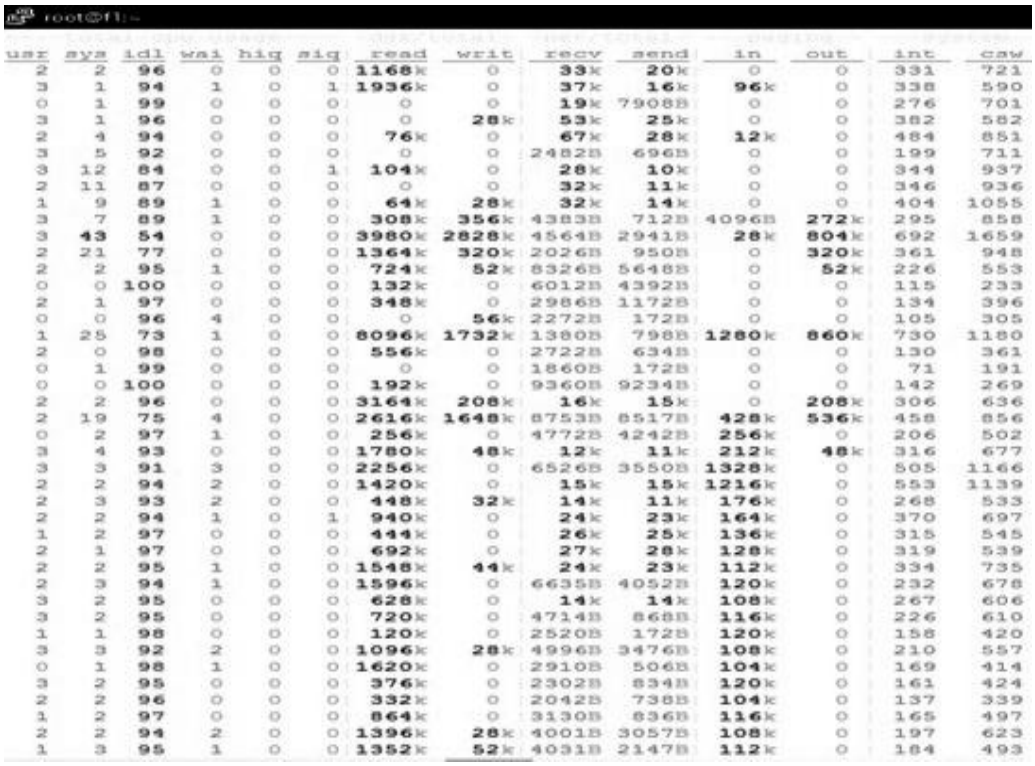


Gráfico 19-4 Una única conexión TCP/IP por página web
Realizado por: León Isabel, 2016

En el gráfico 20-4 se observa que no existe saturación de tráfico a pesar de los ataques podemos concluir que maneja muy bien un ataque DoS.



usr	sys	idl	wai	hig	sig	read	writ	recv	send	in	out	int	csw
2	2	96	0	0	0	1168k	0	33k	20k	0	0	331	721
3	1	94	1	0	1	1936k	0	37k	16k	96k	0	338	590
0	1	99	0	0	0	0	0	19k	7908B	0	0	276	701
3	1	96	0	0	0	0	28k	53k	25k	0	0	382	582
2	4	94	0	0	0	76k	0	67k	28k	12k	0	484	851
3	5	92	0	0	0	0	0	24828	6968	0	0	199	711
3	12	84	0	0	1	104k	0	28k	10k	0	0	344	937
2	11	87	0	0	0	0	0	32k	11k	0	0	346	936
1	9	89	1	0	0	64k	28k	32k	14k	0	0	404	1055
3	7	89	1	0	0	308k	356k	4383B	712B	4096B	272k	295	858
3	43	54	0	0	0	3980k	2828k	4564B	2941B	28k	804k	692	1659
2	21	77	0	0	0	1364k	320k	2026B	950B	0	320k	361	948
2	2	95	1	0	0	724k	52k	8326B	5648B	0	52k	226	553
0	0	100	0	0	0	132k	0	6012B	4392B	0	0	115	233
2	1	97	0	0	0	348k	0	2986B	1172B	0	0	134	396
0	0	96	4	0	0	0	56k	2272B	172B	0	0	105	305
1	25	73	1	0	0	8096k	1732k	1380B	798B	1280k	860k	730	1180
2	0	98	0	0	0	556k	0	2722B	634B	0	0	130	361
0	1	99	0	0	0	0	0	1860B	172B	0	0	71	191
0	0	100	0	0	0	192k	0	9360B	9234B	0	0	142	269
2	2	96	0	0	0	3164k	208k	16k	15k	0	208k	306	636
2	19	75	4	0	0	2616k	1648k	8753B	8517B	428k	536k	458	856
0	2	97	1	0	0	256k	0	4772B	4242B	256k	0	206	502
3	4	93	0	0	0	1780k	48k	12k	11k	212k	48k	316	677
3	3	91	3	0	0	2256k	0	6526B	3550B	1328k	0	505	1166
2	2	94	2	0	0	1420k	0	15k	15k	1216k	0	553	1139
2	3	93	2	0	0	448k	32k	14k	11k	176k	0	268	533
2	2	94	1	0	1	940k	0	24k	23k	164k	0	370	697
1	2	97	0	0	0	444k	0	26k	25k	136k	0	315	545
2	1	97	0	0	0	692k	0	27k	28k	128k	0	319	539
2	2	95	1	0	0	1548k	44k	24k	23k	112k	0	334	735
2	3	94	1	0	0	1596k	0	6635B	4052B	120k	0	232	678
3	2	95	0	0	0	628k	0	14k	14k	108k	0	267	606
3	2	95	0	0	0	720k	0	4714B	868B	116k	0	226	610
1	1	98	0	0	0	120k	0	2520B	172B	120k	0	158	420
3	3	92	2	0	0	1096k	28k	4996B	3476B	108k	0	210	557
0	1	98	1	0	0	1620k	0	2910B	506B	104k	0	169	414
3	2	95	0	0	0	376k	0	2302B	834B	120k	0	161	424
2	2	96	0	0	0	332k	0	2042B	738B	104k	0	137	339
1	2	97	0	0	0	864k	0	3130B	836B	116k	0	165	497
2	2	94	2	0	0	1396k	28k	4001B	3057B	108k	0	197	623
1	3	95	1	0	0	1352k	52k	4031B	2147B	112k	0	184	493

Gráfico 20-4 Sin saturación de tráfico
Realizado por: León Isabel, 2016

Realizado el ataque DoS al servidor HTTP 2.0 muestra que no existe perdida de paquetes y los tiempos de respuesta de la página no son largos.

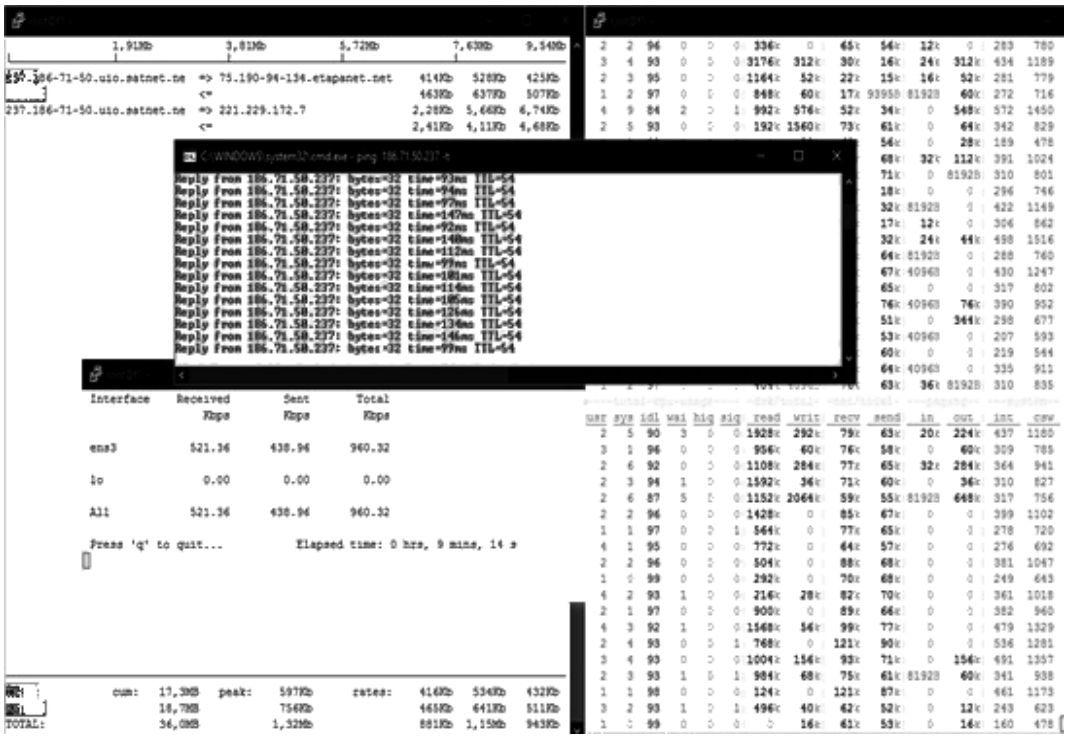


Gráfico 21-4 Sin saturación de tráfico y respuesta
Realizado por: León Isabel, 2016

Otra característica de seguridad importante que presenta HTTP 2.0 es la encriptación de la información, es decir que los datos que envía el cliente al servidor son encriptados de esa forma toda la trama que es enviada son datos ilegibles para otras personas que tratan de obtener esta información ya que el momento que es enviada la información no se lo hace de forma directa sino que tiene que pasar por otros servidores para llegar a su destino final, es ahí donde los hacker aprovechan para conseguir la información pero como esta va encriptada van a obtener datos que no van a entender.

En la pantalla del gráfico 22-4 enseña que los paquetes enviados están encriptados de modo SSL.

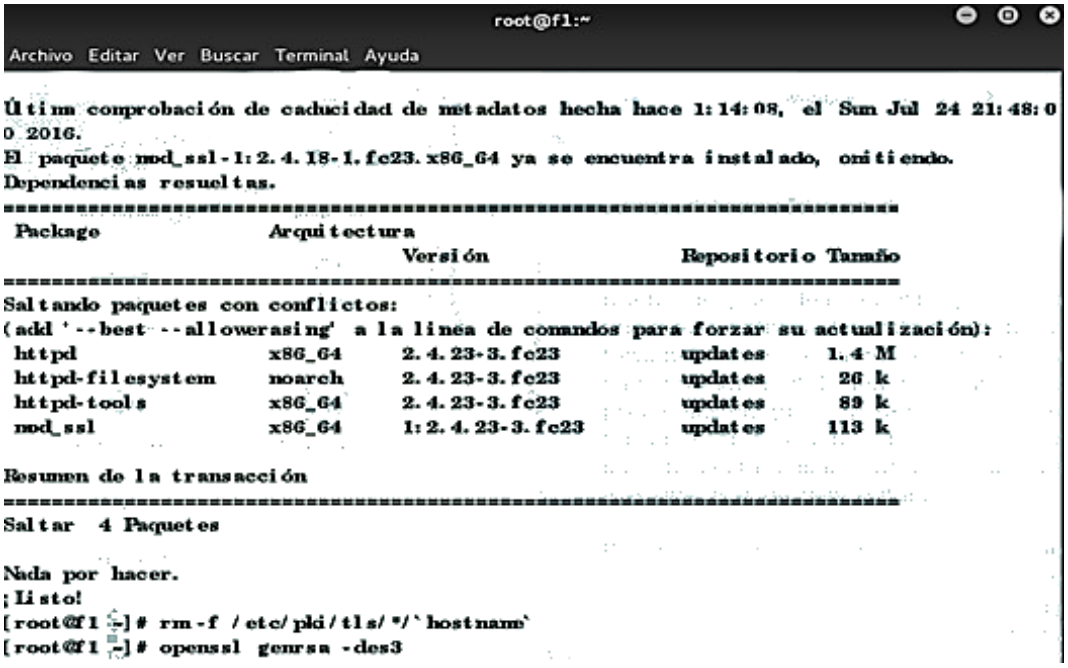


Gráfico 22-4 Comprobación de la instalación de SSL
Realizado por: León Isabel, 2016

Para encriptar la información del cliente se hace utilizando los protocolos SSL (Secure Sockets Layer) y TSL (Transport Layer Security). Lo que presenta el servidor cuando se hace el primer pedido es un certificado que es un mini documento que verifica la web a la que está enviando la información, la comprueban con un proveedor de certificados, y si es el que dice ser entonces se envía la información encriptada desde el cliente que va a generar una llave pública (Key public) y el servidor que recibe la información con otra llave privada que se denomina como (Key private).

Una vez configurado el certificado en el servidor HTTP 2.0 queda por verificar la URL desde el navegador, para esto se escribe https:// seguido del dominio con que se configuro los certificados.

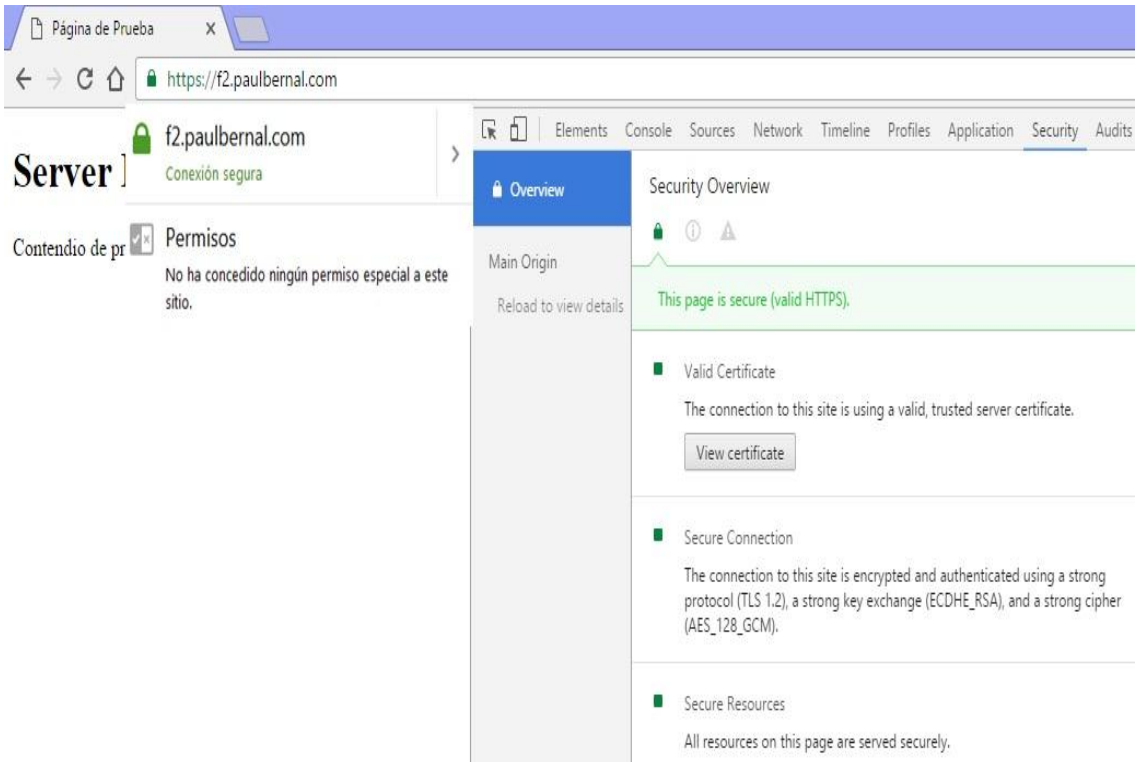


Gráfico 23-4 Verificación url con certificación
Realizado por: León Isabel, 2016

Como se puede ver en el gráfico 23-4, la conexión es segura por lo que se ve un candado en la barra de navegación, de igual forma la página es segura validada para HTTPS.

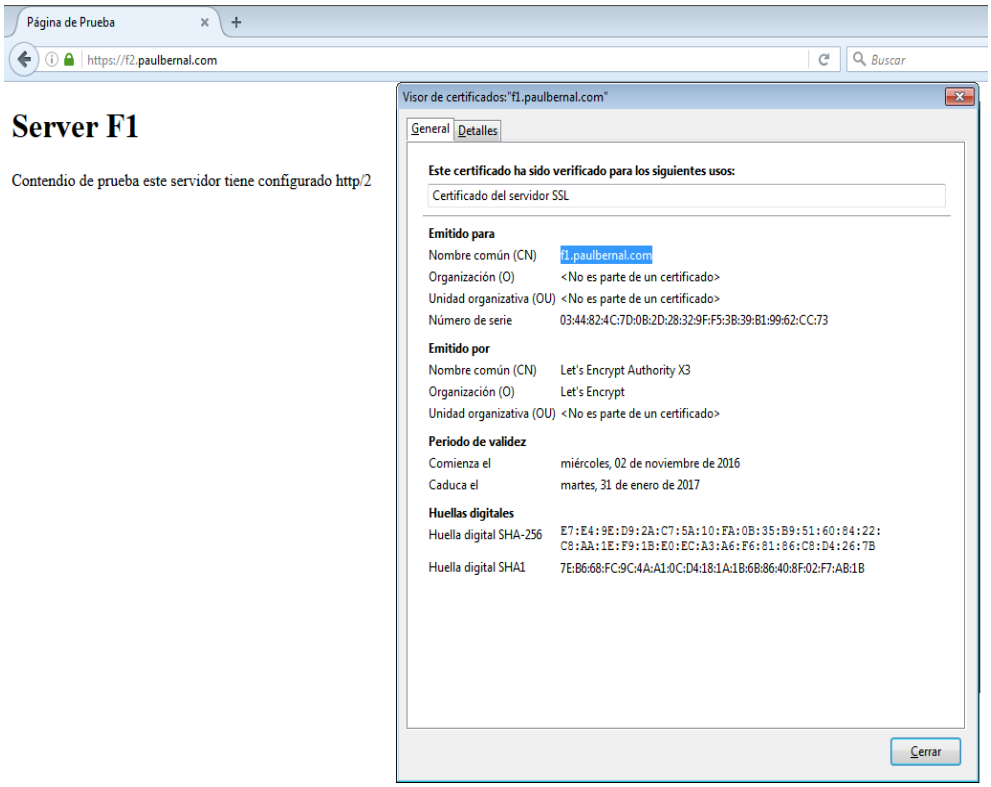


Gráfico 24-4 Dominio seguro del servidor HTTP 2.0
Realizado por: León Isabel, 2016

En el gráfico 24-4 se ve la información de la página del servidor del HTTP 2.0, la misma que está cifrada antes de ser transmitida.

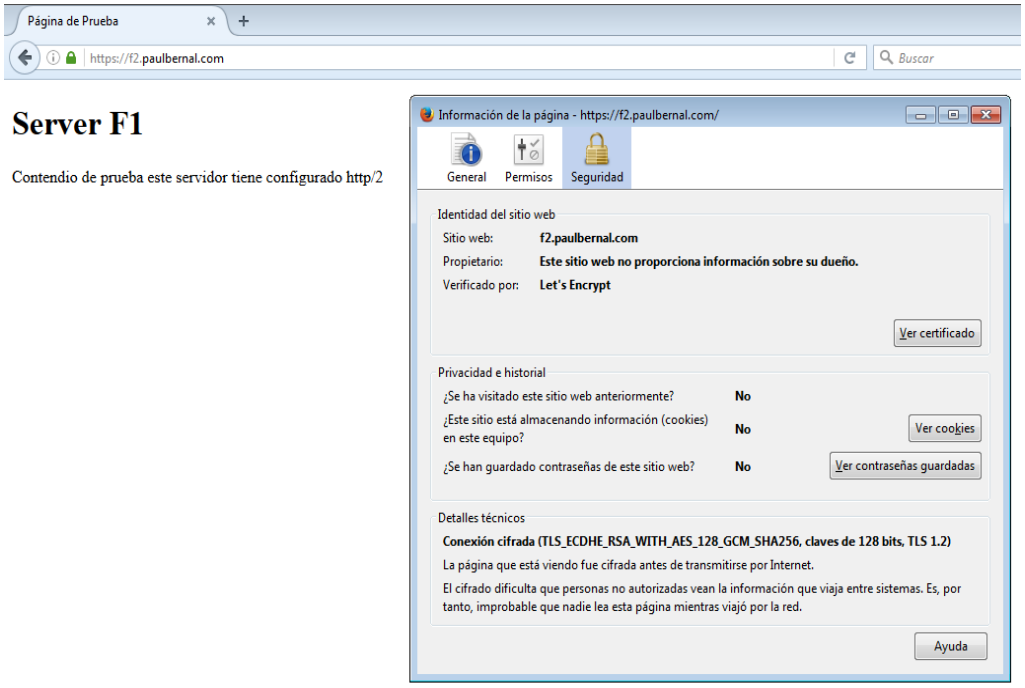


Gráfico 25-4 Conexión Encriptada
Realizado por: León Isabel, 2016

El gráfico 26-4 muestra el uso del padding que es otra caracteristica de seguridad del HTTP 2.0 que ayuda a ocultar el tamaño exacto del contenido del frame, y mitiga los ataques específicos dentro de HTTP, por ejemplo, ataques donde el contenido comprimido incluye texto plano controlado por el atacante y datos secretos.

Los esquemas del padding utilizados incorrectamente serán facilmente derrotados por parte de los atacantes, es por eso que se muestra la programacion que se necesita realizar para otener un padding confiable.

Gráfico 26-4 Uso del Padding
Realizado por: León Isabel, 2016

Ataque hombre en el medio

En éste ataque se re direcciona el tráfico de todos los protocolos a la ip del atacante, se tiene una computadora comprometida a la cual ha ingresado a la computadora que esta con el protocolo http://186.71.50.238/index.php, misma que está montado en php un login para inicio de sesión, el siguiente gráfico 27-4 muestra el respectivo ingreso.

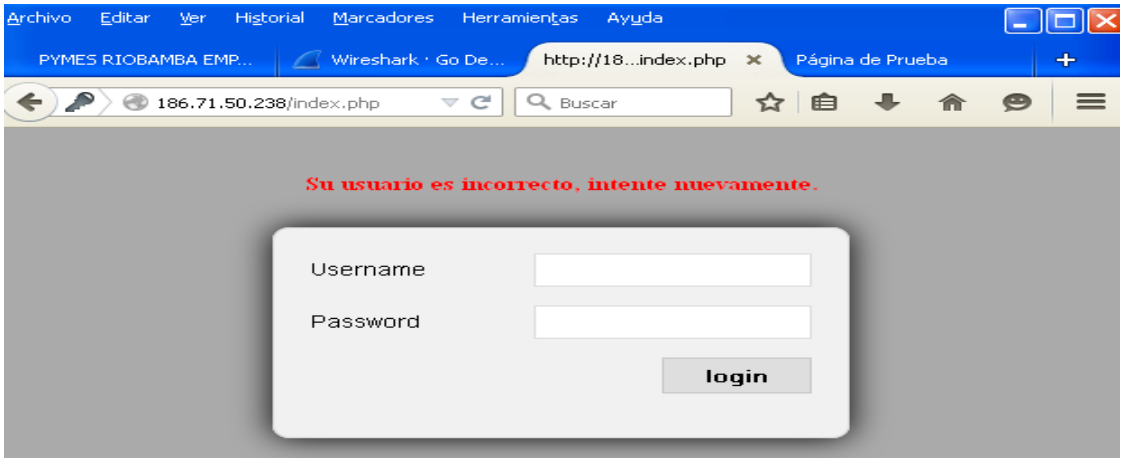


Gráfico 27-4 Ingreso al servidor HTTP 1.1 (ataque hombre en el medio)
Realizado por: León Isabel, 2016

En el gráfico 28-4 con el ataque realizado en el servidor HTTP 1.1 se observa que el usuario y contraseña se pueden obtener fácilmente ya que no está encriptado, porque son datos limpios es decir cómo se envía desde el cliente la información recibe el servidor, pero como se sabe que él envió no es directo si no que pasa por otros servidores y es ahí donde los hackers pueden obtener contraseñas confidenciales.

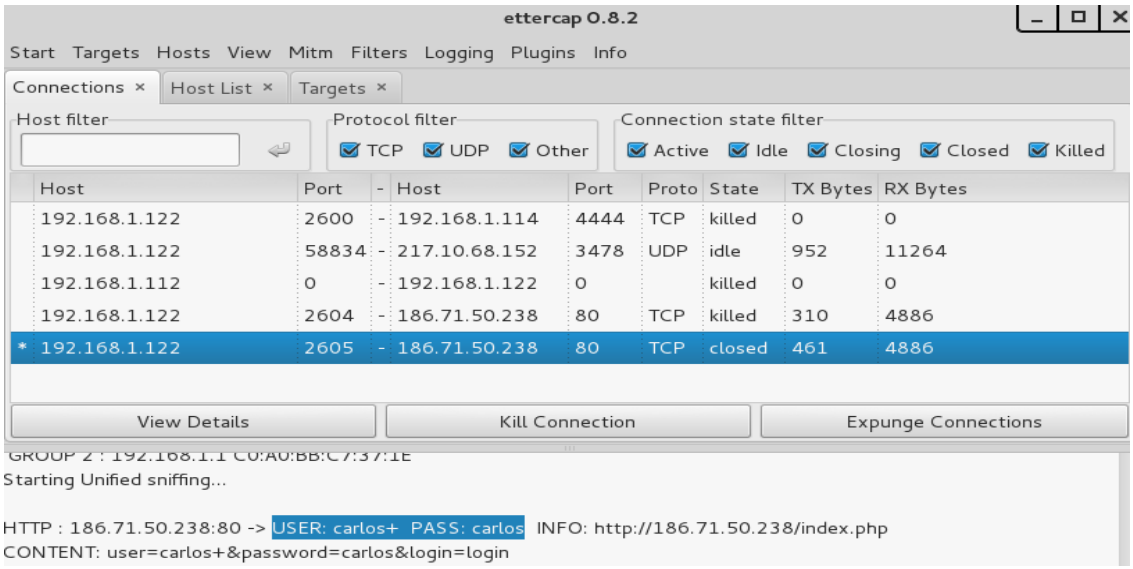


Gráfico 28-4 Servidor no encriptado HTTP 1.1 (ataque hombre en el medio)
Realizado por: León Isabel, 2016

En el gráfico 29-4 muestra el servidor con HTTP 2.0 encriptado y se envía el mismo ataque hombre en el medio para verificar que es más seguro y muestra en la barra de navegación el candadito que está configurado con la certificación.

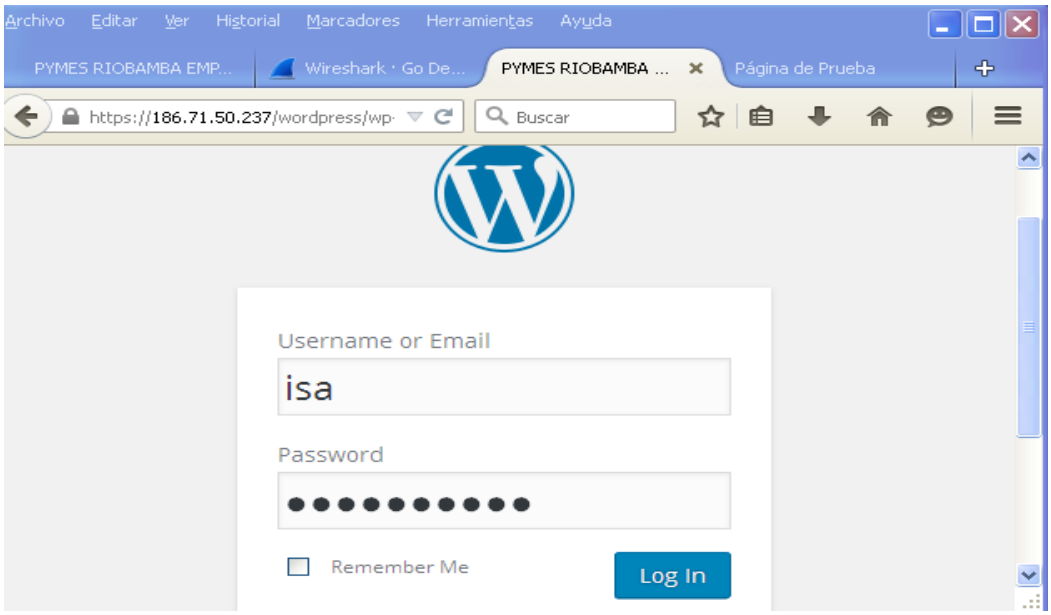


Gráfico 29-4 Ingreso al servidor HTTP 2.0 (ataque hombre en el medio)
Realizado por: León Isabel, 2016

En el gráfico 30-4 se observa que la conexión es segura y que presenta certificado, es decir que la información a simple vista es ilegible porque esta encriptada y solo los que tienen las llaves podrán saber la información.

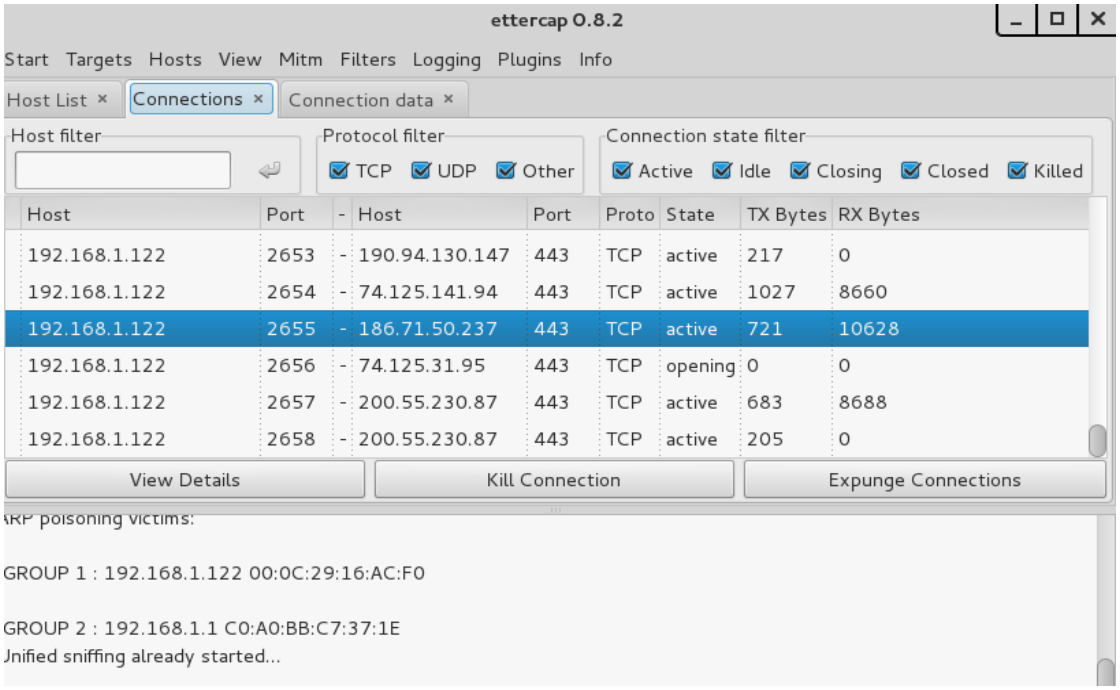


Gráfico 30-4 Servidor encriptado HTTP 2.0 (ataque hombre en el medio)
Realizado por: León Isabel, 2016

Como se puede observar ya no permite esnifar la conexión segura ya que todo el tráfico esta encriptado como se ve en el gráfico 31-4.

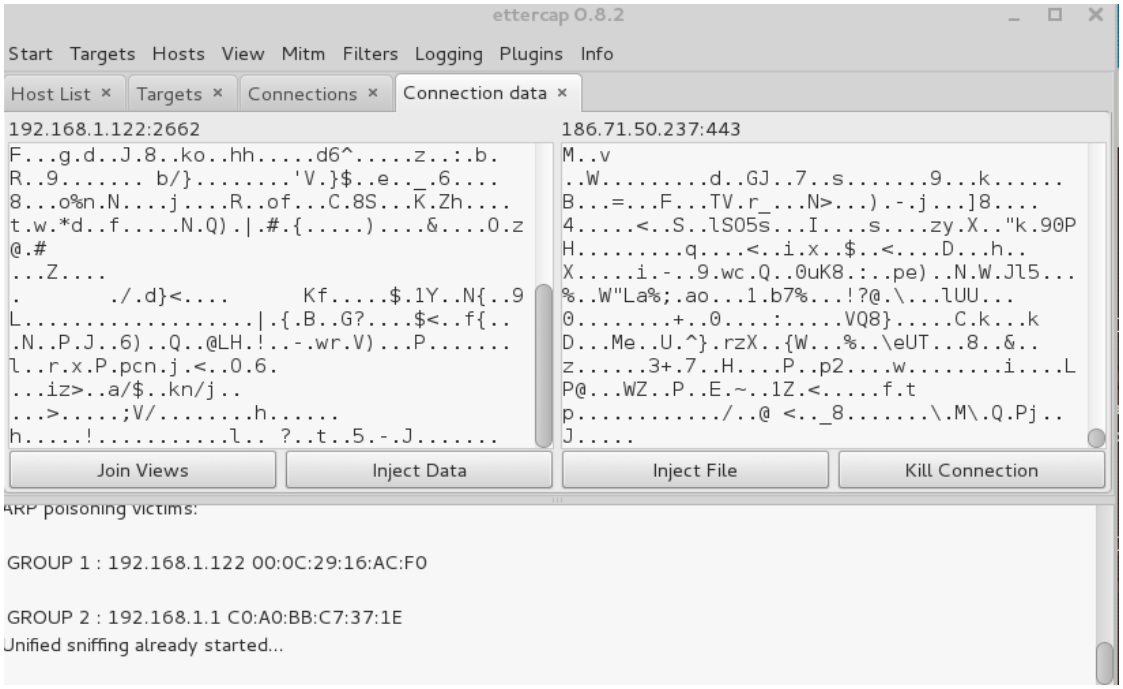


Gráfico 31-4 Sniffing de HTTP 2.0
Realizado por: León Isabel, 2016

4.5 Comprobación de la hipótesis

La hipótesis en la presente investigación es *"La aplicación del método de despliegue de HTTP 2.0 mejorará la seguridad web en las Pyme".*

Para la comprobación de la hipótesis fue necesario la aplicación del método con sus fases del gráfico 1-3, se usó la estadística descriptiva y el programa estadístico SPSS versión 22 con licencia; se aplicó a una web Pyme en donde se anotó el número de vulnerabilidades que tiene HTTP 1.1 ver anexo 1, y las soluciones que se les dio aplicando las características de HTTP 2.0 que dio como resultado la comprobación de la hipótesis con un índice de confianza del 95% por tanto un margen de error del 5%.

En la tabla 3-4 se muestra la primera vulnerabilidad encontrada cuando se aplicó nessus la cual es ***Fully Qualified Domain Name*** (FQDN) que presenta en las Pyme con HTTP 1.1 cuando se realizó un exploit.

Donde N es el número de veces que se ingresó a la web para verificar que la vulnerabilidad está activa en el protocolo ingresado, el mínimo representa el número imperceptible de veces que se eliminó la vulnerabilidad, y el máximo es el valor real que se eliminó la vulnerabilidad en este protocolo.

Tabla 3-4 Estadística descriptiva de HTTP 1.1

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 1.1 – FQND	8	1	2	1,25	,164	,463
N válido (por lista)	8					

Realizado por: León Isabel, 2016

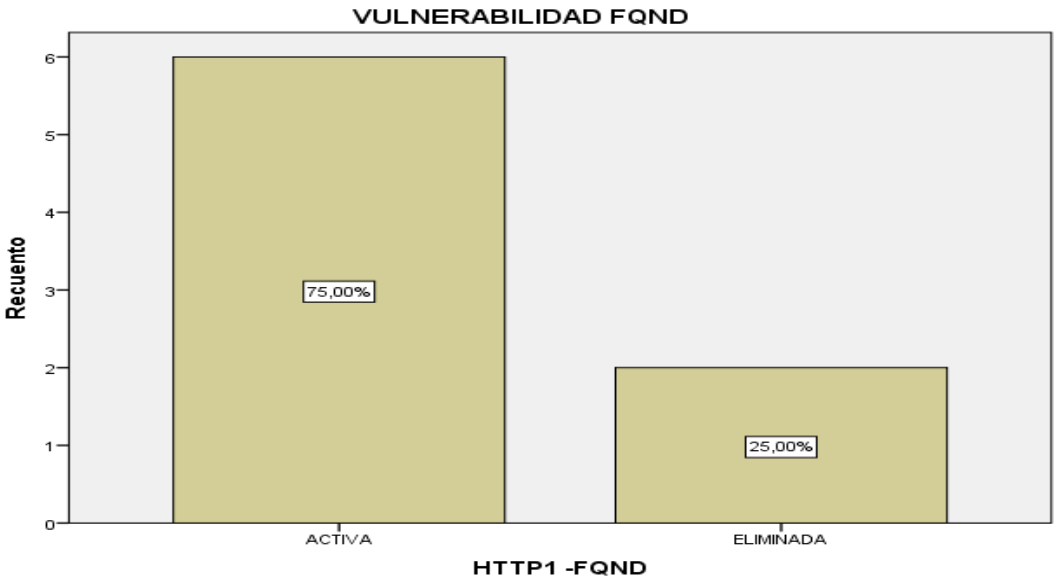


Gráfico 32-4 Estadística descriptiva de HTTP 1.1
Realizado por: León Isabel, 2016

Análisis

Se observa que al realizar una explotación de vulnerabilidades en el servidor HTTP 1.1 dio como resultado que la seguridad no es contrarrestada en un 75% en este protocolo ya que no cuenta con características específicas que ayuden a la seguridad de las Pyme, mientras que se aplica la misma vulnerabilidad al servidor con HTTP 2.0 y se observa que usa configuraciones de seguridad y mejora en un 87,50% indudablemente en la seguridad en la web se puede observar en la tabla 4-4 y el gráfico 33-4.

Tabla 4-4 Estadística descriptiva de HTTP 2.0

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 2.0 – FQND	8	1	2	1,88	,125	,354
N válido (por lista)	8					

Realizado por: León Isabel, 2016

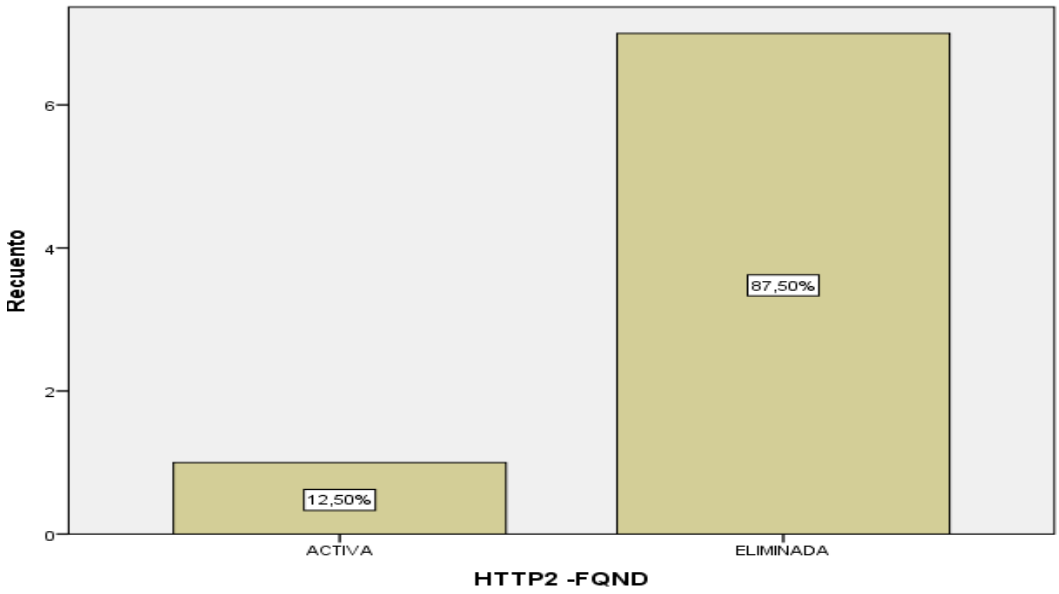


Gráfico 33-4 Estadística descriptiva de HTTP 2.0
Realizado por: León Isabel, 2016

Vulnerabilidad Inconsistent Hostname and IP Address

En la tabla 5-4 la estadística demuestra que se realizó los ataques en el servidor HTTP 1.1, 8 veces y comprobando que se detectó la vulnerabilidad especificada con una media de 1,38.

Tabla 5-4 Estadística descriptiva de host del HTTP 1.1

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 1.1 – HOST	8	1	2	1,38	,183	,518
N válido (por lista)	8					

Realizado por: León Isabel, 2016

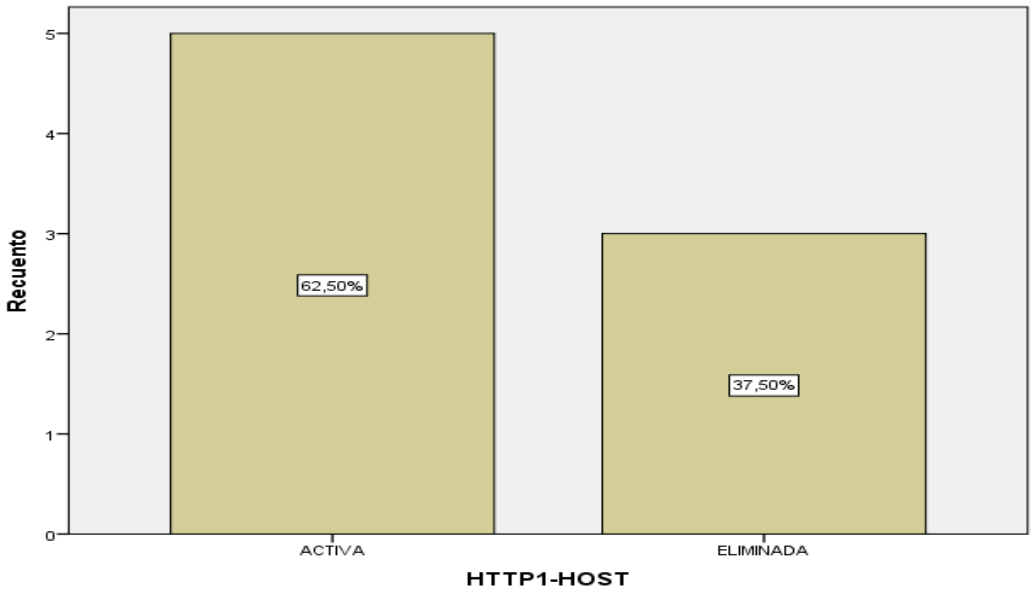


Gráfico 34-4 Estadística descriptiva de inconsistencia de host HTTP 1.1
Realizado por: León Isabel, 2016

Análisis

En un 62,50% la vulnerabilidad Inconsistent Hostname and IP Address sigue activa en el protocolo HTTP 1.1, mientras que la seguridad mejora en un 87,50% aplicando en el servidor HTTP 2.0 la misma vulnerabilidad, esta se contrarresta significativamente como se muestra en la tabla 6-4 y gráfico 35-4.

Tabla 6-4 Estadística descriptiva de host del HTTP 2.0

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 2.0 – HOST	8	1	2	1,88	,125	,354
N válido (por lista)	8					

Realizado por: León Isabel, 2016

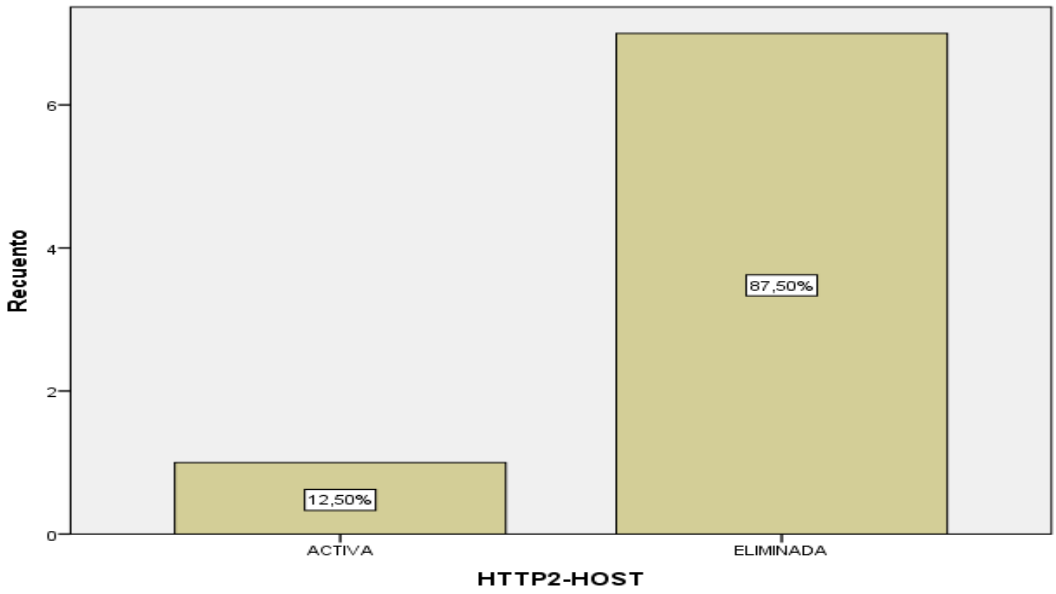


Gráfico 35-4 Estadística descriptiva de inconsistencia de host HTTP 2.0
Realizado por: León Isabel, 2016

Vulnerabilidad Nessus SYN scanner

Tabla 7-4 Estadística descriptiva de SYN del HTTP 1.1

	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 1.1 – SYN	8	1	2	1,25	,164	,463
N válido (por lista)	8					

Realizado por: León Isabel, 2016

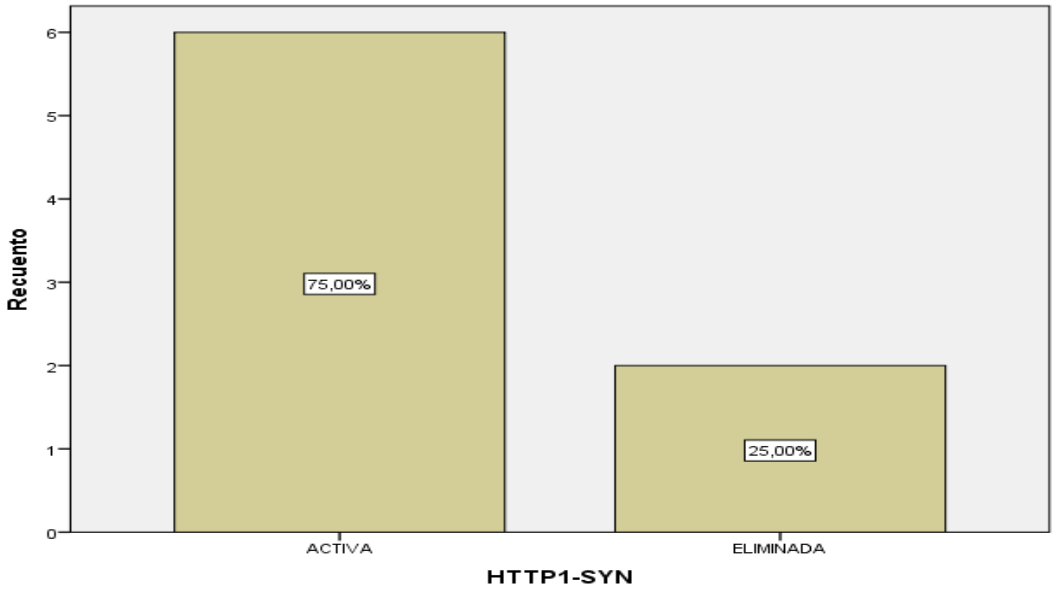


Gráfico 36-4 Estadística descriptiva de inconsistencia SYN HTTP 1.1
Realizado por: León Isabel, 2016

Análisis

Se continúa comprobando la seguridad que tienen los protocolos HTTP 1.1 gráfico 36-4 y HTTP 2.0 gráfico 37-4, con un 75% la vulnerabilidad esta activa en el HTTP 1.1 mientras que en un 75% se encuentra eliminada en HTTP 2.0.

Tabla 8-4 Estadística descriptiva de SYN del HTTP 2.0

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 2.0 – SYN	8	1	2	1,75	,164	,463
N válido (por lista)	8					

Realizado por: León Isabel, 2016

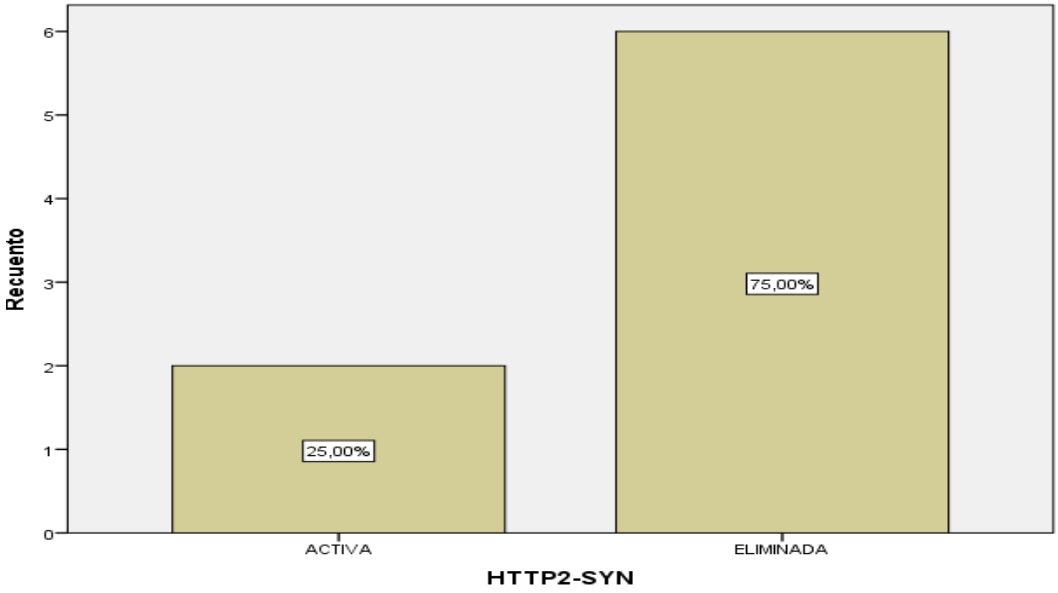


Gráfico 37-4 Estadística descriptiva de inconsistencia SYN HTTP 2.0
Realizado por: León Isabel, 2016

Vulnerabilidad SSH Algorithms and Languages Supported

En HTTP 1.1 está activa en un 100% no contrarresta la vulnerabilidad este protocolo.

Tabla 9-4 Estadística descriptiva de SSH del HTTP 1.1

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 1.1 – SSH	8	1	1	1,00	,000	,000
N válido (por lista)	8					

Realizado por: León Isabel, 2016

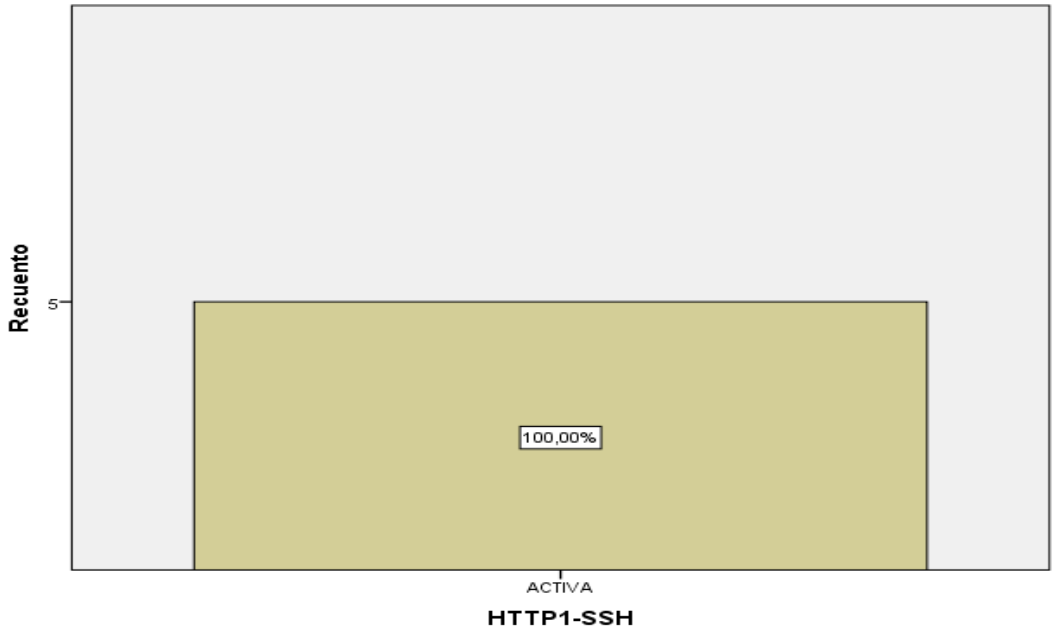


Gráfico 38-4 Estadística descriptiva de SSH HTTP 1.1
Realizado por: León Isabel, 2016

Análisis

Mientras que en el protocolo que cuenta con las características de seguridad actualizados en un 75% se eliminó esta vulnerabilidad y acredita la seguridad que cuenta las Pyme con HTTP 2.0, como se muestra en el gráfico 39-4.

Tabla 10-4 Estadística descriptiva de SSH del HTTP 2.0

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 2.0 – SSH	8	1	2	1,75	,164	,463
N válido (por lista)	8					

Realizado por: León Isabel, 2016

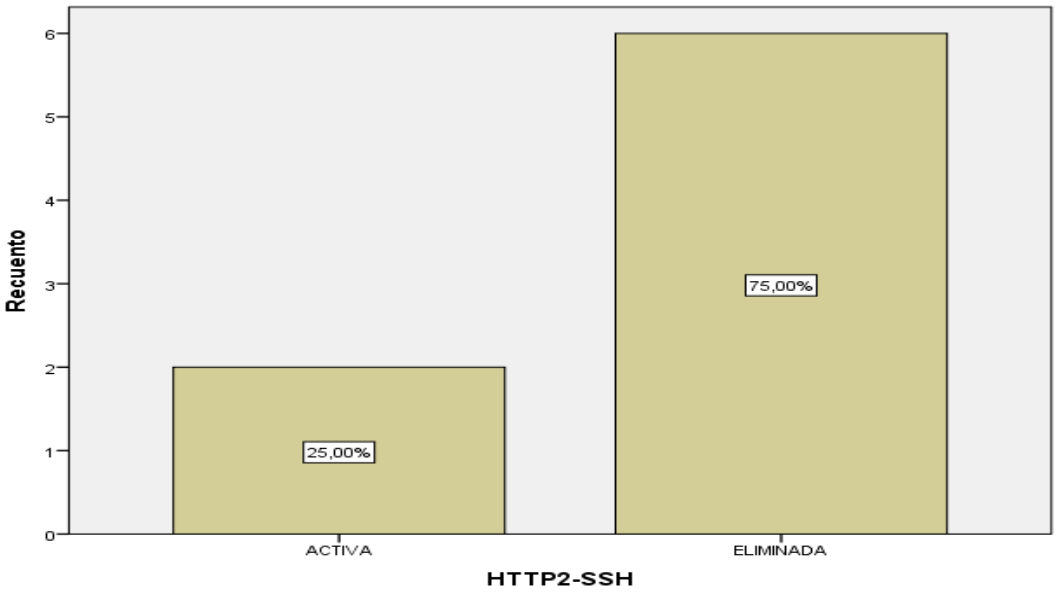


Gráfico 39-4 Estadística descriptiva de SSH HTTP 2.0
Realizado por: León Isabel, 2016

DROWN (Decrypting RSA with Obsolete and Weakened Encryption)

En el gráfico 40-4 muestra que en esta vulnerabilidad tiene un 50% activada y por otra parte el 50% esta eliminado de las Pyme.

Tabla 11-4 Estadística descriptiva de DROWN del HTTP 1.1

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 1.1 – DROWN	8	1	2	1,50	,189	,535
N válido (por lista)	8					

Realizado por: León Isabel, 2016

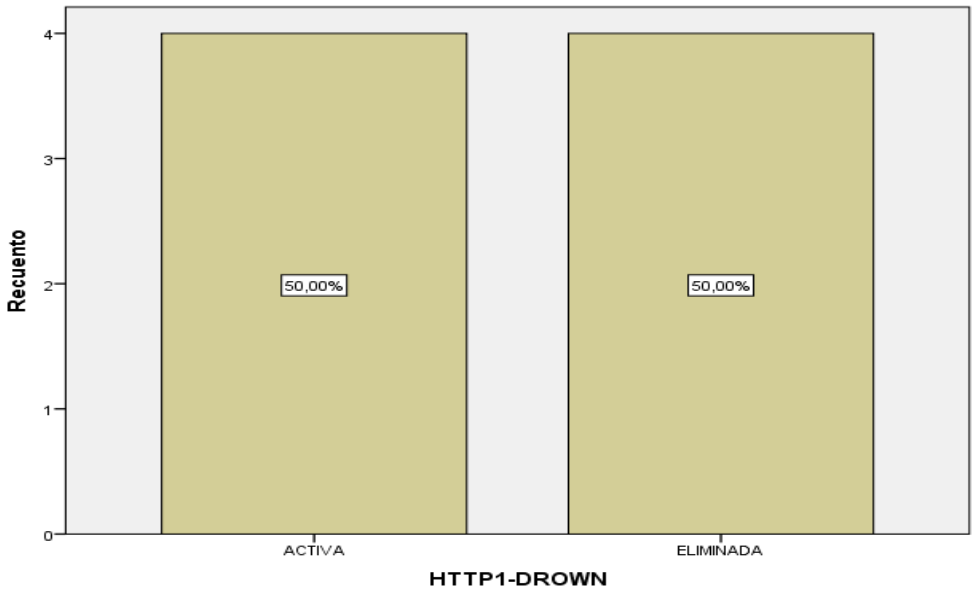


Gráfico 40-4 Estadística descriptiva de DROWN HTTP 1.1
Realizado por: León Isabel, 2016

En el gráfico 41-4 donde se utiliza el servidor HTTP 2.0 la vulnerabilidad drown es eliminada en un 87,50%.

Tabla 12-4 Estadística descriptiva de DROWN del HTTP 2.0

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 2.0 - DROWN	8	1	2	1,87	,125	,354
N válido (por lista)	8					

Realizado por: León Isabel, 2016

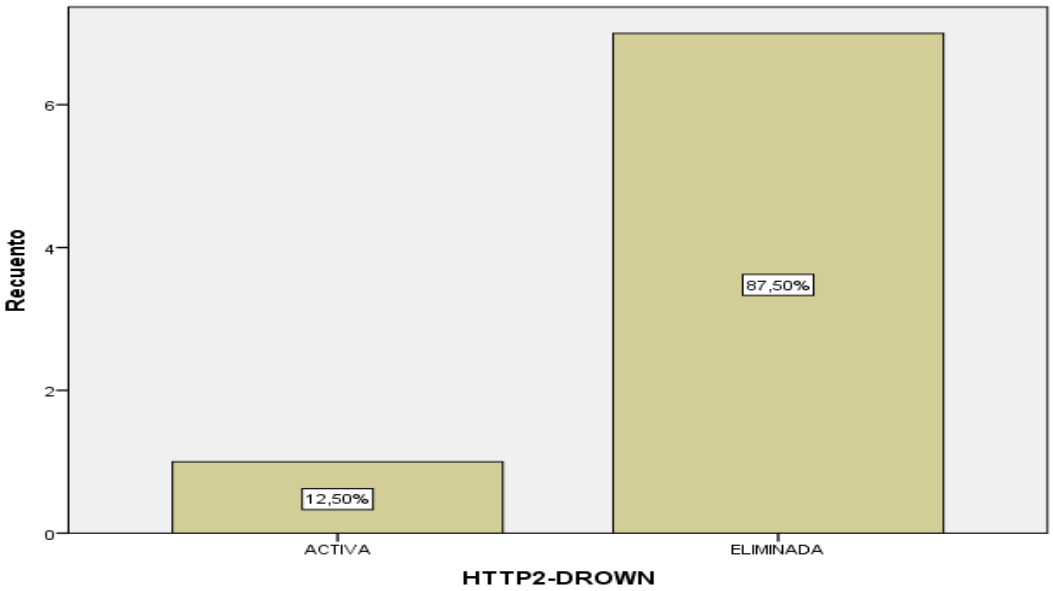


Gráfico 41-4 Estadística descriptiva de DROWN HTTP 2.0
Realizado por: León Isabel, 2016

Las cookies HTTP

Cuando transmiten las cookies sobre HTTP (protocolo no seguro), son particularmente vulnerables en un 75%, puesto que es más probable que los usuarios piensen que la seguridad de la página a la que están conectados se aplica a toda la sesión lo cual no lo es.

Tabla 13-4 Estadística descriptiva de cookies del HTTP 1.1

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 1.1 – COOKIES	8	1	2	1,25	,164	,463
N válido (por lista)	8					

Realizado por: León Isabel, 2016

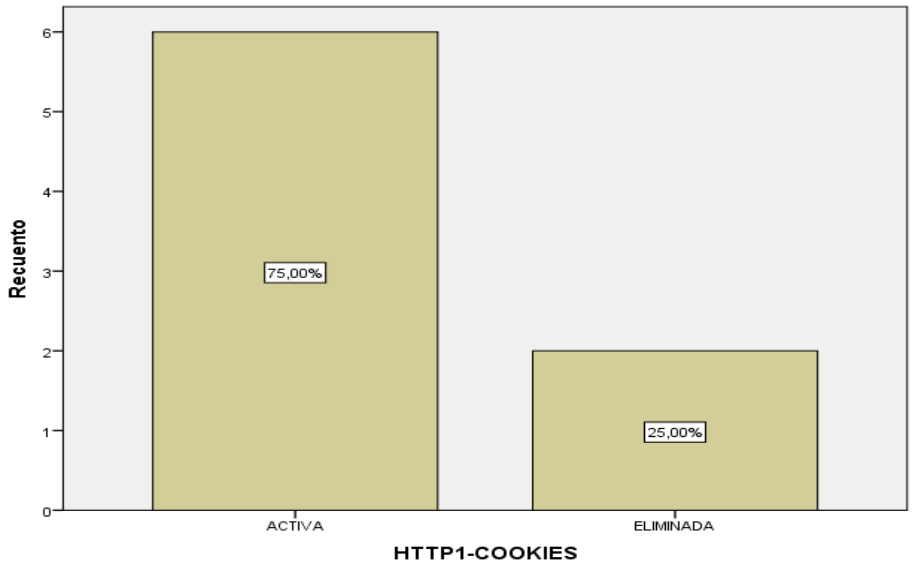


Gráfico 42-4 Estadística descriptiva de COOKIES HTTP 1.1
Realizado por: León Isabel, 2016

Cuando se aplica la misma vulnerabilidad tanto al servidor de HTTP 1.1 y 2.0 muestra una diferencia de seguridad entre los dos protocolos, ya que la versión 2.0 cuenta con mejores características y va a garantizar la seguridad de las Pyme.

Tabla 14-4 Estadística descriptiva de cookies del HTTP 2.0

ESTADÍSTICOS DESCRIPTIVOS						
	N	Mínimo	Máximo	Media	Error estándar	Desviación estándar
HTTP 2.0-COOKIES	8	2	2	2,000	,000	,000
N válido (por lista)	8					

Realizado por: León Isabel, 2016

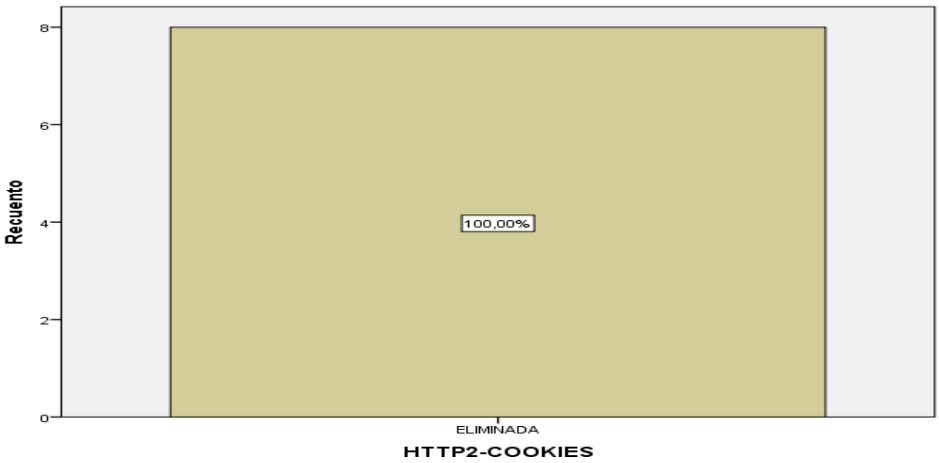


Gráfico 43-4 Estadística descriptiva de COOKIES HTTP 2.0
Realizado por: León Isabel, 2016

La tabla 15-4 muestra la comparativa de los protocolos HTTP 1.1 y 2.0 en función a los indicadores:

Tabla 15-4 Comparativa de resultados en función a indicadores

Indicador	HTTP 1.1	Valor eliminado	HTTP 2.0	Valor eliminado	Herramienta
Confidencialidad	A pesar de tener un inicio de login en la página web, accediendo solo el personal autorizado, el usuario y clave pueden ser fácilmente obtenidos mediante un snifer por no estar encriptado.	15%	Al tener encriptación la conexión es segura, tiene certificado y solo los que tengan la clave pueden acceder al sistema.	95%	Wireshark 1.12.6 y Ettercap 0.8.2
Integridad	Al ser los datos de clave y usuario fácilmente encontrado, implica que la información puede ser manipulada por terceros.	10%	Como el tráfico esta encriptado, será difícil acceder y por ende no habrá modificaciones.	93%	Ettercap 0.8.2
Disponibilidad	Luego del ataque, existen inundaciones de paquetes tcp6, haciendo lento y no funcional la página web después del ataque. Existe mayor latencia en el tiempo de respuesta	5%	Después del ataque el servidor resuelve las peticiones pendientes, no tarda el servicio en dar una respuesta positiva, se observa que la web sigue disponible. No hay saturación de tráfico.	88%	Ping y navegador
Vulnerabilidad	Colapsa frente al ataque	27.08%	Maneja de mejor manera los ataques como el de denegación de servicios.	85.42%	Nessus 5.2.1

Realizado por: León Isabel, 2016

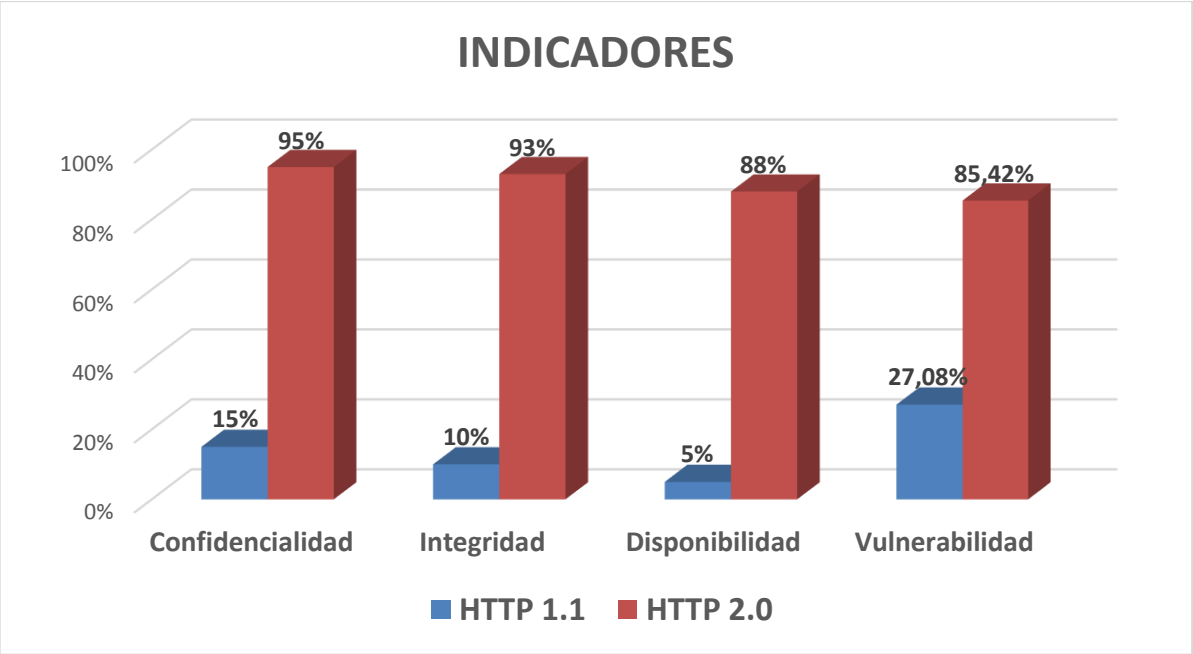


Gráfico 44-4 Comparativa de indicadores entre HTTP 1.1 y HTTP 2.0
Realizado por: León Isabel, 2016

A continuación, en la tabla 16-4 se resume los resultados arrojados al explotar las vulnerabilidades a ambos protocolos HTTP 1.1 y HTTP 2.0.

Tabla 16-4 Comparativa de resultados del indicador vulnerabilidad

VULNERABILIDAD	HTTP 1.1		HTTP 2.0	
	ACTIVA	CORRIGE	ACTIVA	CORRIGE
Fully Qualified Domain Name (FQDN)	75%	25%	12.50%	87.50%
Inconsistent Hostname and IP Address	62.50%	37.50%	12.50%	87.50%
Nessus SYN scanner	75%	25%	25%	75%
SSH Algorithms and Languages Supported	100%	0%	25%	75%
DROWN (Decrypting RSA with Obsolete and Weakened Encryption)	50%	50%	12.50%	87.50%
Cookies	75%	25%	0%	100%
TOTALES	**72.92%**	**27.08%**	**14.58%**	**85.42%**

Realizado por: León Isabel, 2016

Como se puede evidenciar la mejora es notable con el protocolo HTTP 2.0. En el gráfico de abajo se visualiza cuan activa están las vulnerabilidades tanto en el protocolo HTTP 1.1 como en el HTTP 2.0, es decir un 72.92% frente a un 14.58%.

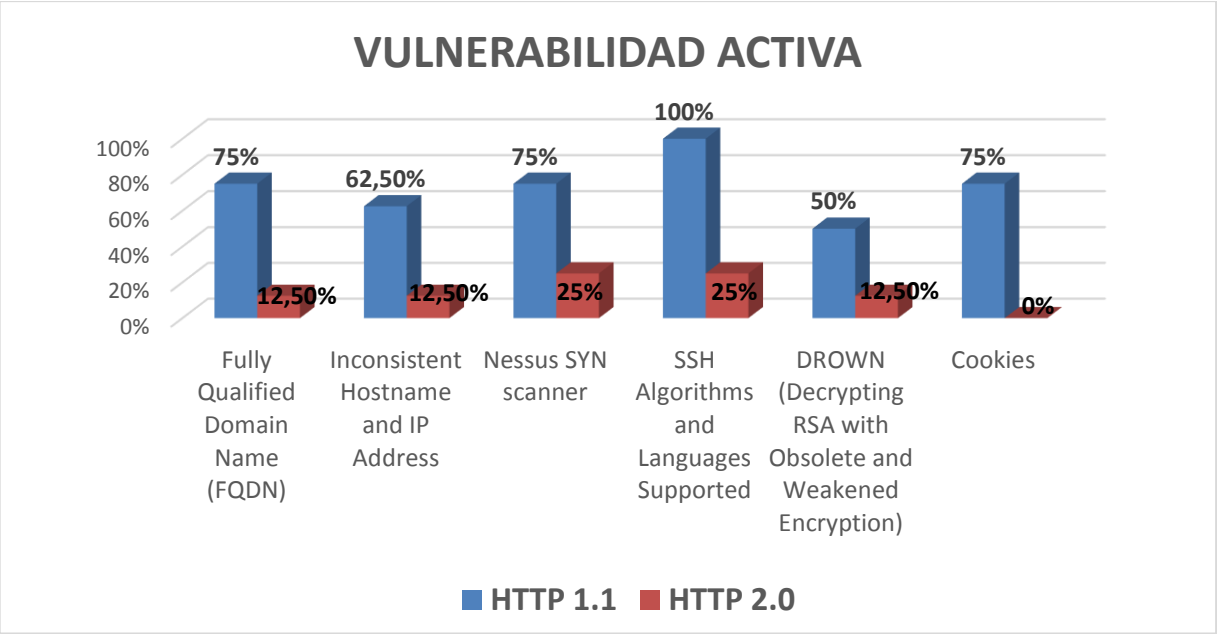


Gráfico 45-4 Vulnerabilidad Activa en HTTP 1.1 y HTTP 2.0
Realizado por: León Isabel, 2016

En el gráfico de abajo se visualiza cuan eliminada están las vulnerabilidades, el protocolo HTTP 1.1 tiene un 27.08% mientras que HTTP 2.0 un 85.42%.

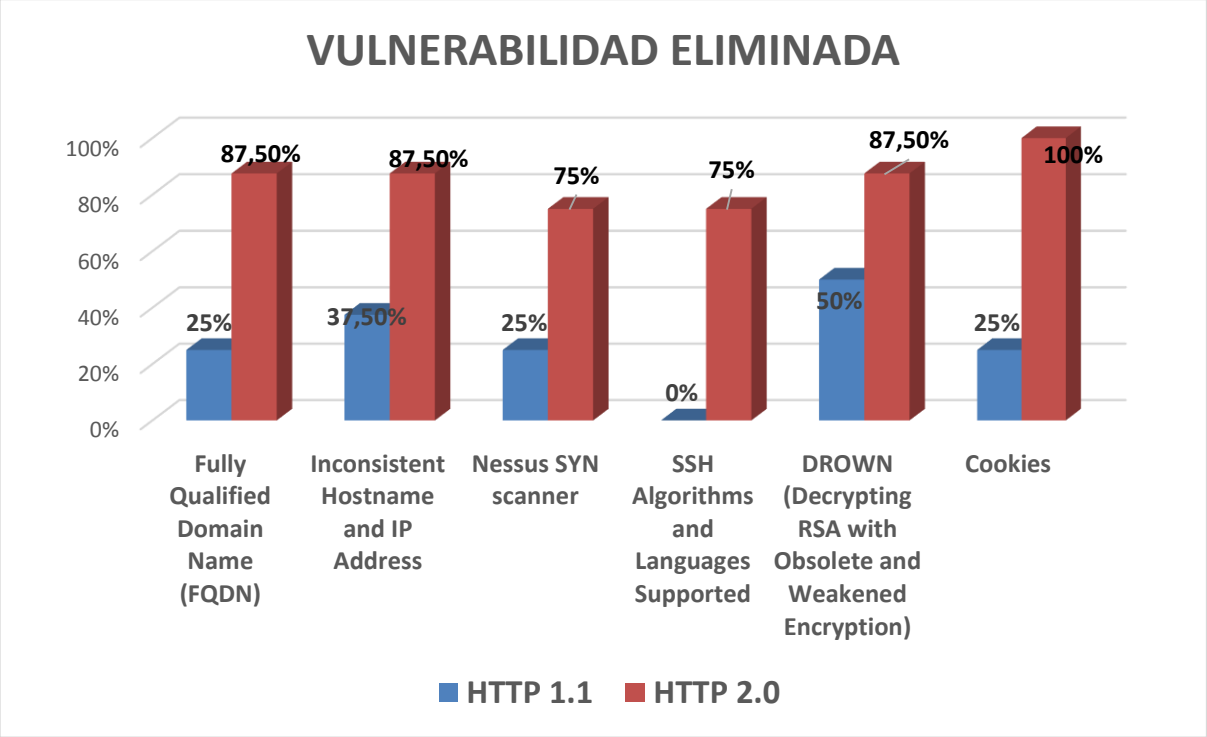


Gráfico 46-4 Vulnerabilidad Eliminada en HTTP 1.1 y HTTP 2.0
Realizado por: León Isabel, 2016

Como se ha mostrado con cada una de las vulnerabilidades que se han aplicado a los servidores que tienen los protocolos HTTP 1.1 y HTTP 2.0, se comprueba la hipótesis que la aplicación del método de despliegue de HTTP 2.0 mejora la seguridad web en las Pyme.

Obteniendo de ésta forma la comprobación de la hipótesis planteada en la investigación realizada.

CONCLUSIONES

- Con HTTP 2.0 el navegador envía la solicitud una vez que el recurso se conoce, especifica la prioridad de cada stream, y el servidor determina la entrega de respuesta óptima. Por lo tanto, elimina la latencia de solicitud de colas innecesarias, y tiene un uso más eficiente de cada conexión, así como de los recursos de la red.
- En este trabajo se ha realizado el análisis, explotación y solución de vulnerabilidades que tiene HTTP 1.1 desde el punto de vista de su capacidad de transmitir datos por la web, es decir a nivel de protocolo, y a partir de la configuración de HTTP 2.0 utilizando sus características se mejoró la seguridad de la web en las Pyme.
- Los problemas de seguridad en la web se encuentran presentes a nivel de la capa de aplicación por la gran cantidad de información que se manipula, y las falencias asociadas a la programación, permiten a terceras personas actuar de forma anónima, creando softwares maliciosos (Virus) para manipular ciertos servicios.
- HTTP 2.0 permite tener aplicaciones seguras como el protocolo SSH, que brinda una respuesta más fuerte y restringida a algunas de las vulnerabilidades de denegación, esto por su estructura de funcionamiento situada entre la capa de transporte y aplicación, haciendo hincapié en la integridad, cifrado y autenticación de la información que se manipula.
- A través de mediciones de laboratorio se pudo observar que HTTP 2.0 tiene menor latencia por tener un menor número de conexiones desde el cliente, la página web de las Pyme no dejo de ser funcional, no tuvo pérdida de paquetes y saturación de tráfico después del ataque de denegación de servicio; y brinda seguridad por tener encriptación de la información con SSL y TLS garantizando la confidencialidad.
- Luego de haber aplicado el método propuesto, HTTP 2.0 obtuvo una vulnerabilidad activa del 14.58% y corregida del 85.42% frente a un 72.92% y 27.08% respectivamente con HTTP 1.1.
- La configuración de las características de HTTP 2.0 que se realizó a las Pyme confirmó que se tiene una sola conexión con múltiples solicitudes y respuestas, eliminó la información repetida, evito el excesivo consumo de recursos, permitió enviar y recibir mucha información al mismo tiempo lo que optimiza la comunicación. Por lo tanto, brinda aplicaciones más rápidas, sencillas y baratas de implementar.
- Dado que implementar HTTP 2.0 al menos en Apache no es compleja y prácticamente no existen costos, por lo cual la implementación en las Pyme es viable, así como para las empresas grandes.

- Este trabajo exploró únicamente las ventajas de seguridad de HTTP 2.0 a nivel del protocolo, es decir todas las mejoras de seguridad que el protocolo implementa con tan solo desplegarlo en el servidor web. No cubre de ninguna forma vulnerabilidades en otros ámbitos que no son objeto de este trabajo como por ejemplo la capa de aplicación, esto es, el protocolo no protege de ninguna forma a la programación web con falencias de seguridad o diseño de la misma.

RECOMENDACIONES

- A los administradores de sistemas se recomienda utilizar la guía práctica de despliegue de HTTP 2.0 para servidores, por las ventajas que tiene sobre 1.1; pues se observó que la seguridad y eficiencia mejoró notablemente en las páginas web de las Pyme.
- Las personas que manejan la web de las empresas deben mejorar la seguridad a nivel de la capa de aplicación, para ello deben saber manipular los servicios de la web para no dejar agujeros abiertos para la intrusión de los hackers.
- Mantener al protocolo SSH por brindar una respuesta fuerte y restringida a algunas vulnerabilidades como la denegación de servicios.
- En HTTP 2.0 no es obligatorio usar SSL, pero se recomienda usarlo para obtener un mejor rendimiento y se está convirtiendo en un requerimiento de los navegadores web.
- Siendo el padding usado para ocultar el tamaño exacto del frame, y para mitigar los ataques específicos dentro de HTTP, debe aplicarse correctamente para no ser fácilmente derrotados por los atacantes, a la vez recordar que padding al azar con una distribución predecible ofrece poca protección.
- Se debe limitar el intercambio de stream por conexión para que un atacante que quisiera sobrecargar la red no pueda abrir múltiples conexiones en las que se intercambiasen numerosos stream.
- Las respuestas push del servidor que no están autorizadas no deben almacenarse en caché, se debe limpiar la cache para evitar los seguimientos de los atacantes.
- Realizar de forma periódica un análisis de vulnerabilidades a las Pyme y usar las actualizaciones de los protocolos para garantizar la seguridad de los datos.
- Utilizar el método propuesto de la investigación para detectar los errores que se están cometiendo en la web de las Pyme, darlos solución y así mantener íntegro los datos confidenciales y sobre todo confiables.
- Se recomienda revisar las mejores prácticas sobre temas de programación segura y endurecimiento de aplicaciones, para mejorar la seguridad a nivel del diseño y desarrollo de los sistemas web que se desplieguen sobre servidores HTTP 2.0.

GLOSARIO DE TERMINOS

Frame: es la unidad más pequeña de la comunicación dentro de una conexión HTTP 2.0, consta de una cabecera y una secuencia de longitud variable de octetos estructurado de acuerdo con el tipo de trama.

Framing binario: es la nueva capa de longitud-prefijo, que dicta cómo se encapsulan y se transfieren los mensajes HTTP entre el cliente y el servidor ofreciendo una representación más compacta, más fácil y más eficiente para procesar en código.

Head of line blocking: bloqueo de cabeza de línea, el servidor envía sus respuestas en el mismo orden en que recibe las solicitudes considerando que solo es una conexión a la vez en HTTP 1.1 y por usar TCP los paquetes son entregados en orden y retransmitidos si se pierde uno de ellos.

Stream: es una secuencia independiente y bidireccional de frames intercambiados entre el cliente y el servidor en una conexión HTTP 2.0.

Stream error: error de un stream individual HTTP 2.0.

Server push: capacidad del servidor para enviar varias respuestas en una sola solicitud del cliente. Es decir, a más de la respuesta de la solicitud original, el servidor puede push (empujar) recursos adicionales para el cliente, sin que él tenga que solicitar explícitamente cada uno.

HSTS: “Seguridad de transporte HTTP estricta”, es un protocolo de normas de IETF, especificado en RFC 6797. El servidor comunica la política HSTS al cliente a través de un campo de la cabecera de respuesta "Strict-Transport-Security"; y especifica un período de tiempo durante el cual el cliente accederá al servidor en forma segura.

SSL: ***Secure Sockets Layer*** (Capa de conexión segura) proporciona autenticación y privacidad de la información entre extremos sobre Internet mediante el uso de criptografía. Habitualmente, sólo el servidor es autenticado (garantiza su identidad) mientras que el cliente se mantiene sin autenticar.

TLS: ***Transport Layer Security*** (Seguridad de capa de transporte) al igual que ssl son protocolos criptográficos que proporcionan comunicaciones seguras por una red.

ALPN: protocolo de negociación de capa de aplicación, permite a la capa de aplicación negociar qué protocolo debe funcionar a través de una conexión segura de manera que evite ida y vuelta adicional y sea independiente de los protocolos de la capa de aplicación. Se utiliza para descubrir y negociar HTTP 2.0 como parte de la negociación HTTPS.

BIBLIOGRAFIA

AEGIS, P. (2014). *Aprobado HTTP/2: el mayor cambio de Internet desde 1999.*

Descargado a partir de http://www.aegis.pe/2015/03/aprobado-http2-gran-cambio-de-internet.html

ANGULO, J. (2016). *¿Qué es HTTP/2 y qué ventajas tiene sobre HTTP 1.1?*

Descargado mayo de 2016, a partir de https://somostechies.com/que-es-http2/#.V8cFdaJUUyM

BELSHE, M., & PEÓN, R. (2014). *A 2x Faster Web*. Descargado a partir de

http://blog.chromium.org/2009/11/2x-faster-web.html

BELSHE, M., THOMSON, M., & PEON, R. (2014). *Hypertext Transfer Protocol version*

2. Descargado 12 de mayo de 2015, a partir de https://tools.ietf.org/html/draft-ietf-httpbis-http2-16

BRIAN J. (2016). *KeyCDN – HTTP/2 Support For All Customers.* Descargado a partir de https://www.keycdn.com/blog/keycdn-http2-support

CASILARI, E., GONZBLEZ, F. J., & SANDOVAL, F. (2001). *Modeling of HTTP traffic.*

IEEE Communications Letters, 5(6), 272-274. Descargado a partir de http://doi.org/10.1109/4234.929610

CLEGG, P. (2015). *How To Setup Http/2 Support (Nginx, Apache, Plesk).* Descargado

abril de 2016, a partir de https://www.gatherdigital.co.uk/community/post/how-to-setup-http-2-support/41

COMUNIDAD WIKIPEDIA. (2015). *HTTP Strict Transport Security.* Descargado el 18

de enero de 2015, a partir de http://es.wikipedia.org/wiki/HTTP_Strict_Transport_Security

COMUNIDAD WIKIPEDIA. (2015). Hypertext Transfer Protocol. Descargado 07 de marzo de 2015, a partir de http://es.wikipedia.org/wiki/Hipertext_Transfer_Protocol

COMUNIDAD WIKIPEDIA. (2014). *SPDY*. Descargado el 19 de marzo de 2014, a partir de http://es.wikipedia.org/wiki/SPDY

COMUNIDAD WIKIPEDIA. (2016). *Fedora (distribución Linux.* Descargado el 23 de mayo de 2016, a partir de https://es.wikipedia.org/wiki/Fedora_%28distribuci%C3%B3n_Linux%29#Fedora_20

EISSING, S. (2015). *How to h2 in apache*. Descargado abril de 2016, a partir de http://icing.github.io/mod_h2/howto.html

FRIEDL, S., LANGLEY, A., & POPOV, A. (2014). *Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension.* Descargado 30 de agosto de 2015, a partir de https://tools.ietf.org/html/rfc7301#section-3

GITHUB. (2015). *HTTP/2 Draft Specifications.* Descargado 8 de marzo de 2015, a partir de https://github.com/http2/http2-spec

GRAHAM, C. (2015). *Tools for debugging, testing and using HTTP/2.* Descargado abril de 2016, a partir de https://blog.cloudflare.com/tools-for-debugging-testing-and-using-http-2/

HANWAY, T. (2015). *Apache HTTP/2 Web Server Setup on Ubuntu 14.04.* Descargado abril de 2016, a partir de http://pixelinc.co/ubuntu-14-04-3-apache-http-2-web-server-setup/

LAND, S. (2015). *mod_http2 Chrome*

ERR_SPDY_INADEQUATE_TRANSPORT_SECURITY. Descargado abril de 2016, a partir de https://www.apachelounge.com/viewtopic.php?p=31857

MANUEL, F. (2013). *HTTP 2.0 cifrado es la respuesta de IETF al espionaje de la NSA.* Descargado 9 de marzo de 2015, a partir de http://www.genbeta.com/web/http-2-0-cifrado-es-la-respuesta-de-ietf-al-espionaje-de-la-nsa

MCMANUS, P., MNOT, & RESCHKE, J. (2015). *HTTP Alternative Services.* Descargado 13 de septiembre de 2015, a partir de https://tools.ietf.org/html/draft-ietf-httpbis-alt-svc-07

MERINO, M. (2014). *¿Qué es el HTTP 2.0 y qué cambios traerá?* Descargado a partir de http://www.ticbeat.com/tecnologias/que-es-el-http-2-que-cambios-traera/

MIRANDA, L. (2015). *Google añadirá soporte a HTTP/2 en Chrome.* Descargado 8 de marzo de 2015, a partir de https://www.fayerwayer.com/2015/02/google-anadira-soporte-a-http2-en-chrome/

NOTTINGHAM, MARK, & THOMSON, M. (2015). *Opportunistic Security for HTTP.* Descargado 12 de septiembre de 2015, a partir de https://tools.ietf.org/html/draft-ietf-httpbis-http2-encryption-02

O'REILLY, M. (2013). *Chapter 12. HTTP/2.* Descargado 13 de julio de 2015, a partir de http://chimera.labs.oreilly.com/books/1230000000545/ch12.html

SHANKLAND, S., (2015). *Be warned: Google enlists Chrome in push for encrypted Web.* Descargado 10 de marzo de 2015, a partir de http://www.cnet.com/news/chrome-becoming-tool-in-googles-push-for-encrypted-web/

SHUAI, L., XIE, G., & YANG, J. (2008). *Characterization of HTTP behavior on access networks in Web 2.0.* En *International Conference on Telecommunications, 2008. ICT 2008* (pp. 1-6). Descargado a partir de http://doi.org/10.1109/ICTEL.2008.4652621

SILVA, M. (2015). *Por qué HTTP/2 es el protocolo que revolucionará Internet.* Descargado 8 de marzo de 2015, a partir de https://www.fayerwayer.com/2015/02/por-que-http2-es-el-protocolo-que-revolucionara-internet/

SILVA, M. (2015). *Firefox se actualiza con soporte para HTTP/2.* Descargado 8 de marzo de 2015, a partir de https://www.fayerwayer.com/2015/02/firefox-se-actualiza-con-soporte-para-http2/

THE CHROMIUM PROJECTS. (2014). *SPDY: An experimental protocol for a faster web.* Descargado 8 de marzo de 2015, a partir de http://www.chromium.org/spdy/spdy-whitepaper

TIM, D. (2008). *The Transport Layer Security (TLS) Protocol Version 1.2.* Descargado 06 de septiembre de 2015, a partir de https://tools.ietf.org/html/rfc5246#section-8.1.2

VALERO, C. (2015). *Llega HTTP/2, el mayor cambio de Internet desde 1999.* Descargado 30 de abril de 2015, a partir de http://www.adslzone.net/2015/02/18/http-2-nuevo/

ANEXOS

Anexo A Reporte Nessus

4

Scan Information
Start time: Sun Jul 10 22:14:18 2016
End time: Sun Jul 10 23:25:02 2016
Host Information
DNS Name: 237.186-71-50.uio.satnet.net
IP: 186.71.50.237
Results Summary
Critical High Medium Low Info Total
0 0 0 0 7 7
Results Details
0/tcp
12053 - Host
Synopsis
It was possible to resolve the name of the remote host.
Description
Nessus was able to resolve the FQDN of the remote host.
Solution
n/a
Risk Factor
None
Plugin Information:
Publication date: 2004/02/11, Modification date: 2012/09/28
Ports
tcp/0
186.71.50.237 resolves as 237.186-71-50.uio.satnet.net.

Synopsis
The remote host's hostname is not consistent with DNS information.
Description
The name of this machine either does not resolve or resolves to a different IP address.
This may come from a badly configured reverse DNS or from a host file in use on the Nessus scanning host.

As a result, URLs in plugin output may not be directly usable in a web browser and some web tests may be
incomplete.

Solution

Fix the reverse DNS or host file.

Risk Factor

None

Plugin Information:

Publication date: 2010/05/03, Modification date: 2016/06/13

Ports

tcp/0

The host name '237.186-71-50.uio.satnet.net' does not resolve to an IP address

10919 - Open Port Re-check

5

Synopsis

Previously open ports are now closed.

Description

One of several ports that were previously open are now closed or unresponsive.

There are several possible reasons for this :

- The scan may have caused a service to freeze or stop running.
- An administrator may have stopped a particular service during the scanning process.

This might be an availability problem related to the following :

- A network outage has been experienced during the scan, and the remote network cannot be reached anymore by
the scanner.
- This scanner may has been blacklisted by the system administrator or by an automatic intrusion detection / prevention system that detected the scan.
- The remote host is now down, either because a user turned it off during the scan or because a select denial of
service was effective.

In any case, the audit of the remote host might be incomplete and may need to be done again.

Solution

- Increase checks_read_timeout and/or reduce max_checks.
- Disable any IPS during the Nessus scan

Risk Factor

None

Plugin Information:

Publication date: 2002/03/19, Modification date: 2014/06/04

Ports

tcp/0

Port 443 was detected as being open but is now unresponsive
Port 80 was detected as being open but is now unresponsive

Synopsis

This plugin displays information about the Nessus scan.

Description

This plugin displays, for each tested host, information about the scan itself :

- The version of the plugin set.
- The type of scanner (Nessus or Nessus Home).
- The version of the Nessus Engine.

- The port scanner(s) used.
- The port range scanned.
- Whether credentialed or third-party patch management checks are possible.
- The date of the scan.
- The duration of the scan.
- The number of hosts scanned in parallel.
- The number of checks done in parallel.

Solution

n/a

Risk Factor

None

Plugin Information:

Publication date: 2005/08/26, Modification date: 2016/04/08

Ports

tcp/0

Information about this scan :
Nessus version : 6.7.0
Plugin feed version : 201607081830
Scanner edition used : Nessus
Scan type : Normal
Scan policy used : Advanced Scan
6
Scanner IP : 192.168.1.21
Port scanner(s) : nessus_syn_scanner
Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report verbosity : 1
Safe checks : yes
Optimize the test : yes
Credentialed checks : no
Patch management checks : None
CGI scanning : disabled
Web application tests : disabled
Max hosts : 5
Max checks : 5
Recv timeout : 5
Backports : None
Allow post-scan editing: Yes
Scan Start Date : 2016/7/10 22:14 SA Pacific Standard Time
Scan duration : 4244 sec

/udp

10287 - Traceroute Information

Synopsis

It was possible to obtain traceroute information.

Description

Makes a traceroute to the remote host.

Solution

n/a

Risk Factor

None

Plugin Information:

Publication date: 1999/11/27, Modification date: 2013/04/11

Ports

udp/0
For your information, here is the traceroute from 192.168.1.21 to 186.71.50.237 :
192.168.1.21
192.168.1.1
?
80/tcp
11219 -
Synopsis
It is possible to determine which TCP ports are open.
Description
This plugin is a SYN 'half-open' port scanner. It shall be reasonably quick even against a firewalled target.
Note that SYN scans are less intrusive than TCP (full connect) scans against broken services, but they might cause
problems for less robust firewalls and also leave unclosed connections on the remote target, if the network is loaded.
Solution
Protect your target with an IP filter.
Risk Factor
None
Plugin Information:
Publication date: 2009/02/04, Modification date: 2016/06/23
Ports
tcp/80
Port 80/tcp was found to be open
443/tcp
7
11219 - Nessus
Synopsis
It is possible to determine which TCP ports are open.
Description
This plugin is a SYN 'half-open' port scanner. It shall be reasonably quick even against a firewalled target.
Note that SYN scans are less intrusive than TCP (full connect) scans against broken services, but they might cause
problems for less robust firewalls and also leave unclosed connections on the remote target, if the network is loaded.
Solution
Protect your target with an IP filter.
Risk Factor
None
Plugin Information:
Publication date: 2009/02/04, Modification date: 2016/06/23
Ports
tcp/443
Port 443/tcp was found to be open
8
186.71
Scan Information
Start time: Sun Jul 10 22:14:18 2016
End time: Sun Jul 10 23:23:51 2016
Host Information
DNS Name: 238.186-71-50.uio.satnet.net
IP: 186.71.50.238
Results Summary

Critical High Medium Low Info Total
0 0 0 0 7 7
Results Details
0/tcp
12053 -
Synopsis
It was possible to resolve the name of the remote host.
Description
Nessus was able to resolve the FQDN of the remote host.
Solution
n/a
Risk Factor
None
Plugin Information:
Publication date: 2004/02/11, Modification date: 2012/09/28
Ports
tcp/0
186.71.50.238 resolves as 238.186-71-50.uio.satnet.net.
46215
Synopsis
The remote host's hostname is not consistent with DNS information.
Description
The name of this machine either does not resolve or resolves to a different IP address.
This may come from a badly configured reverse DNS or from a host file in use on the Nessus scanning host.
As a result, URLs in plugin output may not be directly usable in a web browser and some web tests may be
incomplete.
Solution
Fix the reverse DNS or host file.
Risk Factor
None
Plugin Information:
Publication date: 2010/05/03, Modification date: 2016/06/13
Ports
tcp/0
The host name '238.186-71-50.uio.satnet.net' does not resolve to an IP address
19506 - Nessus Scan Information
9
Synopsis
This plugin displays information about the Nessus scan.
Description
This plugin displays, for each tested host, information about the scan itself :
- The version of the plugin set.
- The type of scanner (Nessus or Nessus Home).
- The version of the Nessus Engine.
- The port scanner(s) used.
- The port range scanned.
- Whether credentialed or third-party patch management checks are possible.
- The date of the scan.
- The duration of the scan.
- The number of hosts scanned in parallel.

- The number of checks done in parallel.

Solution

n/a

Risk Factor

None

Plugin Information:

Publication date: 2005/08/26, Modification date: 2016/04/08

Ports

tcp/0

Information about this scan :
Nessus version : 6.7.0
Plugin feed version : 201607081830
Scanner edition used : Nessus
Scan type : Normal
Scan policy used : Advanced Scan
Scanner IP : 192.168.1.21
Port scanner(s) : nessus_syn_scanner
Port range : default
Thorough tests : no
Experimental tests : no
Paranoia level : 1
Report verbosity : 1
Safe checks : yes
Optimize the test : yes
Credentialed checks : no
Patch management checks : None
CGI scanning : disabled
Web application tests : disabled
Max hosts : 5
Max checks : 5
Recv timeout : 5
Backports : None
Allow post-scan editing: Yes
Scan Start Date : 2016/7/10 22:14 SA Pacific Standard Time
Scan duration : 4173 sec

0/udp

10287 - Traceroute Information

Synopsis

It was possible to obtain traceroute information.

Description

Makes a traceroute to the remote host.

Solution

n/a

Risk Factor

None

10

Plugin Information:

Publication date: 1999/11/27, Modification date: 2013/04/11

Ports

udp/0

For your information, here is the traceroute from 192.168.1.21 to 186.71.50.238 :
192.168.1.21
192.168.1.1
?

22/tcp

11219 - Nessus SYN scanner

Synopsis

It is possible to determine which TCP ports are open.

Description

This plugin is a SYN 'half-open' port scanner. It shall be reasonably quick even against a firewalled target.

Note that SYN scans are less intrusive than TCP (full connect) scans against broken services, but they might cause problems for less robust firewalls and also leave unclosed connections on the remote target, if the network is loaded.

Solution

Protect your target with an IP filter.

Risk Factor

None

Plugin Information:

Publication date: 2009/02/04, Modification date: 2016/06/23

Ports

tcp/22

Port 22/tcp was found to be open

70657 - SSH Algorithms and Languages Supported

Synopsis

An SSH server is listening on this port.

Description

This script detects which algorithms and languages are supported by the remote service for encrypting communications.

Solution

n/a

Risk Factor

None

Plugin Information:

Publication date: 2013/10/28, Modification date: 2014/04/04

Ports

tcp/22

Nessus negotiated the following encryption algorithm with the server : aes128-ctr

The server supports the following options for kex_algorithms :

curve25519-sha256@libssh.org
diffie-hellman-group-exchange-sha256
diffie-hellman-group14-sha1
ecdh-sha2-nistp256
ecdh-sha2-nistp384
ecdh-sha2-nistp521

The server supports the following options for server_host_key_algorithms :

ecdsa-sha2-nistp256
rsa-sha2-256
11
rsa-sha2-512
ssh-ed25519
ssh-rsa

The server supports the following options for encryption_algorithms_client_to_server :

aes128-ctr
aes128-gcm@openssh.com
aes192-ctr

aes256-ctr
aes256-gcm@openssh.com
chacha20-poly1305@openssh.com
The server supports the following options for encryption_algorithms_server_to_client :
aes128-ctr
aes128-gcm@openssh.com
aes192-ctr
aes256-ctr
aes256-gcm@openssh.com
chacha20-poly1305@openssh.com
The server supports the following options for mac_algorithms_client_to_server :
hmac-sha1
hmac-sha1-etm@openssh.com
hmac-sha2-256
hmac-sha2-256-etm@openssh.com
hmac-sha2-512
hmac-sha2-512-etm@openssh.com
umac-128-etm@openssh.com
umac-128@openssh.com
umac-64-etm@openssh.com
umac-64@openssh.com
The server supports the following options for mac_algorithms_server_to_client :
hmac-sha1
hmac-sha1-etm@openssh.com
hmac-sha2-256
hmac-sha2-256-etm@openssh.com
hmac-sha2-512
hmac-sha2-512-etm@openssh.com
umac-128-etm@openssh.com
umac-128@openssh.com
umac-64-etm@openssh.com
umac-64@openssh.com
The server supports the following options for compression_algorithms_client_to_server :
none
zlib@openssh.com
The server supports the following options for compression_algorithms_server_to_client :
none
zlib@openssh.com

443/tcp

11219 - Nessus SYN scanner

Synopsis

It is possible to determine which TCP ports are open.

Description

This plugin is a SYN 'half-open' port scanner. It shall be reasonably quick even against a firewalled target.

Note that SYN scans are less intrusive than TCP (full connect) scans against broken services, but they might cause problems for less robust firewalls and also leave unclosed connections on the remote target, if the network is loaded.

Solution

Protect your target with an IP filter.

Risk Factor
12
None
Plugin Information:
Publication date: 2009/02/04, Modification date: 2016/06/23
Ports
tcp/443

Port 443/tcp was found to be open