PONTIFICIA UNIVERSIDAD CATÓLICA DEL ECUADOR



FACULTAD DE INGENIERÍA

MAESTRÍA EN REDES DE COMUNICACIÓN

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO DE:

MASTER EN REDES DE COMUNICACIÓN

TEMA:

“ESTUDIO DE HERRAMIENTAS DE SOFTWARE LIBRE PARA LA CREACIÓN DE SISTEMAS DE DETECCIÓN AUTOMATIZADA DE DEFACEMENTS DE SITIOS WEB”

PAUL FERNANDO BERNAL BARZALLO

Quito – 2014

Contenido

- Contenido
- GLOSARIO
- Capítulo 1 :: INTRODUCCIÓN
  - 1.1. Introducción
  - 1.2. Antecedentes
  - 1.3. Justificación
  - 1.4. Objetivos
    - 1.4.1. Objetivo General
    - 1.4.2. Objetivos Específicos
- Capítulo 2 :: ESTADO DEL ARTE
  - 2.1. La seguridad Web
    - 2.1.1. La seguridad en los servidores Web
    - 2.1.2. La seguridad en las aplicaciones Web
  - 2.2. Ataques a sitios web
  - 2.3. Defacements o desfiguraciones
  - 2.4. Detección de cambios en un sitio web
    - 2.4.1. El sitio web visto como código HTML
    - 2.4.2. El sitio web visto como una imagen
  - 2.5. El Software Libre como plataforma de trabajo
    - 2.5.1. Linux
    - 2.5.2. Bash y LAMP
- Capítulo 3 :: ANÁLISIS COMPARATIVO DE ALTERNATIVAS
  - 3.1. Captura y análisis de código HTML
    - 3.1.1. Métricas
    - 3.1.2. Captura de código HTML
    - 3.1.3. Análisis de cambios en código HTML
      - 3.1.3.1. Diff
      - 3.1.3.2. Pretty-Diff
      - 3.1.3.3. Cmp
      - 3.1.3.4. Comm
      - 3.1.3.5. JSdiff
  - 3.2. Captura y análisis de imágenes
    - 3.2.1. Métricas
    - 3.2.2. Captura de imágenes
    - 3.2.3. Análisis de cambios en imágenes
      - 3.2.3.1. Compare.py
      - 3.2.3.2. Perceptual Diff
      - 3.2.3.3. ImageMagick
      - 3.2.3.4. GraphicsMagick

3.3. Automatización de procesos
3.4. Recopilación de información de análisis
3.4.1. Resultados de análisis de cambios HTML
3.4.2. Resultados de análisis de cambios en imágenes
Capítulo 4 :: CONSTRUCCIÓN SISTEMA PROTOTIPO
4.1. Construcción del sistema CLI de análisis de diferencias en código HTML
4.2. Construcción del sistema CLI de análisis de diferencias en imágen
4.3. Construcción del sistema CLI de emisión de alertas programadas
4.4. Construcción del sistema web de configuración y administración
4.5. Complementación del sistema web con un área informativa
4.6. Pruebas y ajustes
4.7. Análisis de resultados
Capítulo 5 :: CONCLUSIONES Y RECOMENDACIONES
BIBLIOGRAFÍA
ANEXOS

GLOSARIO

- **Botnet.** Hace referencia a un *robot* informático o *bot*, que infecta un equipo y se ejecuta de manera autónoma y automática para algún propósito específico como el de recabar información del equipo ó usuario.
- **CEDIA.** Acrónimo de Consorcio Ecuatoriano para el Desarrollo del Internet Avanzado.
- **CERT.** Acrónimo en inglés de Equipo de Respuesta ante Emergencias Informáticas (Computer Emergency Response Team).
- **CLI.** Acrónimo en inglés de Interfaz de Línea de Comandos (Command Line Interface).
- **CMS.** Acrónimo en Inglés de Sistema de Gestión de Contenidos, es un programa informático que permite gestionar la creación y administración de contenidos, principalmente en páginas web.
- **Cron.** Es un demonio usado en los sistemas Unix y derivados, como Linux, para realizar de ejecución de ciertos procesos o tareas en segundo plano y de acuerdo a una programación temporal o periodicidad determinada. La frecuencia mínima de ejecución es cada un minuto.
- **CRUD.** Es el acrónimo de Crear, Leer, Actualizar y Borrar (del original en inglés: Create, Read, Update and Delete) y es usado comúnmente en desarrollo de sistemas para identificar todo el proceso y producto necesario para realizar dichas tareas dentro de un elemento o de una aplicación completa.
- **CSIRT.** Acrónimo en inglés de Equipo de Respuesta a Incidentes de Seguridad (Computer Security Incident Response Team).
- **Defacement.** Cuando se encuentra código que indica que los atacantes se burlan del administrador del sistema por no mantener la seguridad del servidor y/o para mostrar las habilidades en sistemas de un atacante, a través del cambios que generan variantes visuales de un sitio[9].
- **Forking.** Término usado cuando un desarrollador o grupo de ellos toman una copia del código fuente de un proyecto e inician con él un desarrollo independiente, creando una pieza de software distinta y separada.
- **LMS**. Acrónimo en Inglés de Sistema de Gestión de Aprendizaje, es un software instalado en un servidor web que se emplea para administrar, distribuir y controlar las actividades de formación o educación no presencial.
- **Open DNS resolver.** Es un servidor de DNS que permite que cualquier equipo, desde cualquier ubicación, use sus servicios. Típicamente es un servidor de DNS de caché que fue pobremente configurado/asegurado.
- **Path.** Es un conjunto de rutas donde los S.O. buscan los binarios o ejecutables en general, que han sido invocados de alguna forma.
- **Python.** Es un lenguaje de programación interpretado y muy ampliamente usado en sistemas de todo tipo en sistemas Linux y multiplataforma.

- **Raspberry PI.** Es un computador de placa reducida o (placa única) (SBC) de bajo costo, desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas. Está siendo ampliamente usado para el desarrollo de soluciones especializadas y de costo reducido.
- **Renderización.** Es el proceso de generar una imagen o vídeo mediante el cálculos a partir de un modelo. Normalmente se usa el término para el proceso de digitalización de imágenes de objetos 3D, pero se ha generalizado para la generación de otro tipo de imágenes digitales en base a cierto tipo de cálculos.
- **Sinkhole.** Es un servidor/servicio que captura todo el tráfico de cierto tipo y emite información falsa como respuesta para dar la sensación de legitimidad al servicio que ofrece. Se puede usar constructivamente, para analizar tráfico por ejemplo, o con efectos destructivos como saturación de conexiones por ejemplo.
- **TIC.** Acrónimo de Tecnologías de la Información y Comunicación.
- **URL.** Se trata de un localizador de recursos uniforme por sus siglas en inglés, que identifica un recurso en una red como Internet. Están formados por una secuencia de caracteres, de acuerdo a un formato estandarizado.
- **Wrapper.** Se usa comúnmente para denominar una función, comando, clase, etc., que sirve como envoltorio o sustituto para otra. Por ejemplo cuando se pueden presentar conflictos de nombre, se puede usar un envoltorio para disfrazar el nombre original y exponer el envoltorio en su lugar.

Capítulo 1 :: INTRODUCCIÓN

1.1. Introducción

A través de este trabajo, cuyo objetivo es la creación de un sistema de detección automatizada de *defacements* o desfiguración de sitios web, se pretende proveer a la comunidad del área de las *TIC*, de una valiosa herramienta de seguridad que ayude a mantener la imagen institucional de los sitios web. Este sistema, si bien se considera el objetivo final del proyecto, no es efectivamente la razón científica de ser del mismo, sino el trabajo de investigación mediante el método descriptivo, de las diferentes herramientas de Software Libre (SL) que puedan usarse para lograr la captura y análisis de la información de un sitio web y que permita la detección de cambios en el mismo.

Se plantea que el sistema tenga dos subsistemas de detección o sensores, que trabajarán de forma paralela y simultánea, para resultados más confiables y precisos. Estos subsistemas tomarían dos enfoques diferentes: el análisis del código HTML del sitio pues fundamentalmente un sitio web es eso; y el análisis imagenográfico de una captura de la página principal del sitio, que finalmente es lo que se ve en el navegador como resultado de ese código HTML.

Si bien, existen actualmente sistemas y servicios de detección de ataques, éstos trabajan en función de un procedimiento no automatizado y no desatendido, que típicamente implica intervención/evaluación humana basada en un reporte previo de ataque, normalmente emitido por el propio atacante. Uno de los más conocidos es Zone-H[1].

El entorno donde nace esta propuesta, es el del *CSIRT* del *CEDIA*[2], donde desde hace un tiempo se ha venido colaborando como especialista en sistemas de Software Libre. Como parte de este trabajo se emprendió en el desarrollo de un sistema de recepción, procesamiento y reporte de alertas de seguridad emitidas por Shadow Server[3], que a través de sus sensores detectan y reportan por email de problemas relacionados típicamente con pobres políticas de seguridad implementadas, pero limitado básicamente a *botnets*, *sinkholes* y *open dns resolvers*. Este sistema ya está listo y en pleno funcionamiento, pero al momento no detecta ni reporta web defacements. Para complementar esta limitación se empezó a usar los reportes de Zone-H, pero éstos no son gratuitos ni automatizados, pues dependen de que el propio atacante reporte su hazaña y ésta sea verificada por un equipo de técnicos que lo validan y entonces es oficial, se reporta a los suscriptores y se publica. Por ello, en este sentido se evidenció a la necesidad de implementar un sistema de detección automatizada de desfiguración de sitios web, que permita, no solo minimizar los tiempos de alerta a los administradores de los sitios atacados, sino también relacionar y verificar los datos obtenidos de Zone-H y similares.

En esta tesis se describe justamente el entorno y los elementos que enmarcan el proyecto, así como aquellos que sustentan y justifican su realización con un claro planteamiento de objetivos y planificación temporal para el desarrollo de la investigación e implementación. Todo aquello apoyado por una previa investigación bibliográfica adecuadamente referenciada.

1.2. Antecedentes

Uno de los ataques más comunes hacia los sitios web en la actualidad son los *defacements* que consisten en la desfiguración del sitio web a través de alguna técnica que permita el reemplazo del contenido web original por uno diferente, esto es mayormente, cambios al código HTML de la página de inicio del sitio. Estos cambios pueden ser desde pequeños y visualmente sutiles, hasta grandes y evidentes. Las razones de los ataques son también variadas, que van desde el simple ataque de principiantes hasta la sistemática y masiva desfiguración de sitios que comparten una vulnerabilidad común.

Como ya se dijo, en la actualidad, la única forma de detectarlos no es automática y requiere necesariamente de la intervención de un ser humano que visualmente detecte y reporte el incidente. También se comentó otra forma que es a través de la publicación de los ataques en sitios especializados, pero nuevamente requieren de la intervención de un ser humano y es más, en prácticamente todos ellos, los humanos que los reportan son justamente los atacantes, a manera de divulgación de sus logros.

Se podría pensar que una alternativa a lo propuesto es simplemente aplicar técnicas de aseguramiento para endurecer el servidor web. Pero en realidad no lo es, puesto que el objetivo de este proyecto no es el de ser una alternativa de aseguramiento, sino una forma de detección de problemas en los servidores que por cualquier razón no han sido apropiadamente asegurados -que desgraciadamente son la mayoría- y prueba de ello son justamente los reportes en sitios especializados como por ejemplo el de zone-h, a través del cual se puede evidenciar que el problema del defacement de sitios es totalmente actual y recursivo a nivel mundial y si se filtra para sitios claramente ecuatorianos (.ec)[7], la realidad respecto a los sitios gubernamentales (.gob.ec) y educacionales (.edu.ec) -por mencionar dos de las categorías más recurrentes- no es diferente sino quizá incluso más cruda, pues son muy frecuentes (por temporadas incluso diarios) los reportes de ataques a dichos sitios. Adicionalmente, aún con servidores o sitios endurecidos, que es la medida deseada, no se resuelve los temas de: ingeniería social, obsolescencia del código y/o aplicaciones, falta de mantenimiento, grupos en los que no se puede o no se atiende la seguridad adecuadamente.

Uno de los factores potenciadores de esta realidad, es que por un lado, las instituciones siguen sin darle la importancia necesaria al área de las TIC y la siguen viendo como cualquier cosa menos como lo que es: un área estratégica. El tiempo de tener un sitio web por mera vanidad pasó hace mucho y hoy por hoy el sitio web es mucho más que la imagen y presencia de una empresa en el ciberespacio. En este sentido, las empresas pueden gastar mucho en promoción por ejemplo, pero suelen invertir muy poco en las capacidades/capacitación de los administradores de su sitio web; sin contemplar las graves consecuencias que tiene el que sus clientes actuales o potenciales se topen con un sitio atacado o desfigurado que pudiera hacer por ejemplo publicidad en contra.

Por otro lado -y esta pudiera ser también una consecuencia de lo dicho- está el hecho de que la cultura de seguridad en nuestro medio es aún bastante pobre y conduce típicamente a que se actúe con la técnica del avestruz, es decir a evadir, esconder, minimizar, desestimar o encubrir el ataque; quizá por un falso sentido de vergüenza, quizá por sentirse impotente ante el caso, quizá por ver amenazada su condición laboral. Esto sin contar con el hecho de que lo más común es que para esto hayan ya pasado días, semanas, e incluso en contados casos, meses, de haberse producido el incidente; es decir, es poco frecuente que los administradores

tengan una alerta temprana del problema. Un sistema que detecte y notifique automatizadamente de posibles problemas de desfiguración del sitio web institucional, por supuesto le dará al administrador del mismo más tiempo de reaccionar apropiadamente y obtener mejores resultados.

Han sido tristemente célebres los defacements, claramente con tinte político, alrededor de la fecha patria del 10 de agosto u otras fechas significativas o mediáticamente relevantes[4]; el de tinte deportivo del sitio de una prestigiosa empresa deportiva exhibiendo el "vestido oficial" de un equipo de fútbol[5]; el del sitio de su contraparte azul[6] y muchos más que igualmente constatan lo dicho. En todos esos casos, lo más probable es que los administradores hayan realizado su esfuerzo de aseguramiento del servidor y/o sitio, pero en ninguno de ellos, tuvieron forma de ser alertados del ataque antes de que los visitantes del sitio pudieran hacerlo en base a haber visto los efectos del mismo.

1.3. Justificación

Por todo lo expuesto en el punto anterior, y a pesar de ello, son pocos los sistemas de detección de desfiguraciones y peor aún cuando se trata de sistemas de detección y reporte automáticos. Adicionalmente, estos sistemas/servicios no son gratuitos y ninguno de ellos está orientado a nuestro medio que, como ya se dijo, tiende a desatender el tema de seguridad en los sitios. El sistema de detección propuesto será totalmente desarrollado y liberado bajo la filosofía del Software Libre, de tal forma que por una lado impulse su uso extendido sin restricciones legales ni económicas y por otro lado promueva la colaboración de la comunidad para perfeccionarlo, mejorarlo y extenderlo. Se ha contemplado también la potencialidad de ser parte de un sistema más grande que detecte y reporte múltiples tipos de ataques a sus usuarios, cosa que también sería perfectamente factible.

Es importante también aclarar el hecho de que no será un sistema que deba ser instalado en un servidor específico para ser sensado, sino que pretende ser un servicio en línea gratuito, que cualquier persona o institución pueda usar para monitorear y recibir alertas de ataques de desfiguración en su/s sitio/s. Es un sistema de detección, no de protección y no está orientado al uso particular sino comunitario.

El estudio paralelo planteado de dos formas -diferentes pero complementarias- de detectar posibles cambios en un sitio web, también refuerza la necesidad científica del trabajo, sin dejar de lado la propia captura de información en cada caso, es decir cómo obtener sistemática, automatizada y confiablemente, por un lado el código y por otro la imagen, respectivamente, de cada sitio. Para esta parte, los muchos años de experiencia profesional en la administración de sistemas Linux y de desarrollo de aplicaciones web con SL, definitivamente fueron la base técnica que sustentó la búsqueda, análisis y selección de las herramientas más idóneas para el trabajo de obtención y detección. Una de las partes más atractivas del proyecto, fue la del subsistema de captura y procesamiento de imágenes, pues, aunque pudiera parecer sencillo capturar una vista de un sitio web, hacerlo desde *CLI* y de forma automatizada, fue reto, al igual que el procedimiento mismo que se debió realizar para compararlas y detectar cambios entre ellas. No con esto se implica que la captura y análisis de código fuente HTML sea simple, pero definitivamente no resulta tan atractiva aunque sí totalmente funcional. Se realizó un trabajo de investigación mediante el método descriptivo, para la búsqueda, análisis y selección de las herramientas de SL más apropiadas para el proyecto. El sistema prototipo planteado, es el resultado tangible

de aplicar lo encontrado en el trabajo de investigación. Evidentemente se pueden visualizar además las posibilidades estadísticas de la información que se recopila, que permiten la emisión de reportes detallados y periódicos que pueden ayudar no solo a visualizar tendencias sino también a la toma de decisiones.

De esta forma, se despliega el trabajo de implementar un sistema que, de forma automatizada, detecta cambios en una página web y le reporta el incidente a un contacto previamente definido para que, acciones adecuadas puedan ser tomadas oportunamente. Dado que elementalmente lo que se desea es minimizar la intervención humana, el sistema no es capaz de establecer juicios, sino porcentajes o un grado de discordancia entre una línea base y el estado actual de un sitio web. El análisis y las acciones a tomarse en base a ello seguirán totalmente en manos del administrador/responsable del sitio, teniendo sin embargo la enorme ventaja de, con este reporte y la información incluida en él, tomar la decisión más adecuada según sea el caso. Podría también, con pocas modificaciones, alimentar el sistema con respuestas de estados de confirmación falso positivo según fuese el caso.

El entorno de aplicación del sistema propuesto es global, pero se planea inicialmente implementarlo a nivel de las universidades, los CSIRT, los CERT y eventualmente instituciones públicas ecuatorianas, que pudieran ser los más frecuentemente atacados.

1.4. Objetivos

1.4.1. Objetivo General

Crear un sistema de detección automatizada de defacements de sitios web para la emisión de alertas tempranas a los administradores de los mismos..

1.4.2. Objetivos Específicos

1. Analizar las herramientas y métodos en SL para la captura automatizada de código fuente HTML de un sitio web.
2. Seleccionar la herramienta y método o conjunto de ellos, más adecuados para la captura automatizada de código fuente HTML de un sitio web.
3. Analizar las herramientas y métodos en SL para la captura automatizada de la imagen de una cierta página de un sitio web.
4. Seleccionar la herramienta y método o conjunto de ellos, más adecuado para la captura automatizada de la imagen de una cierta página de un sitio web.
5. Crear un sistema en *CLI* que use las herramientas y métodos seleccionados para la captura programada del código e imagen de un sitio web y el respectivo almacenamiento de los mismos.
6. Crear un sistema web de administración de parámetros del sistema y presentación de información y reportes.

Capítulo 2 :: ESTADO DEL ARTE

El tema de seguridad como universo para este estudio, es un área fundamental dentro de las ciencias informáticas y sin embargo es una de las que suelen recibir menor atención a nivel organizacional, casi siempre por lo complejo que resulta traducir la inversión en seguridad en beneficios tangibles para la organización. Es decir, los directivos suelen perder de vista la importancia e impacto de la seguridad informática en su organización, hasta que se produce el primer incidente y se presentan como consecuencia las primeras pérdidas o costos asociados.

2.1. La seguridad Web

En el gran universo de la seguridad informática, la seguridad en los sitios web es una de las áreas de más actualidad dada la gran profusión de sitios web de todo tipo, lo cual se da en gran medida gracias a lo cada vez más accesibles que se han venido convirtiendo estos servicios.

Pero justamente esta popularización, esta expansión en el número de sitios, esta facilidad y rapidez con que hoy se pueden poner en línea, hacen que en la mayoría de casos se descuide la seguridad. Las razones de esto son variadas, pero en gran medida se deben a que quienes desarrollan, construyen o preparan los sitios web, en muchos de los casos no son siquiera profesionales informáticos y por ende no tienen en cuenta estos aspectos de seguridad al momento de ponerlos en línea. También pasa mucho que las empresas que proveen servicios de hospedaje para estos sitios, tampoco se preocupan demasiado del tema pues es muy complejo estar pendiente de decenas, cientos o hasta miles de sitios que un proveedor puede alojar.

Todos estos factores provocan que hoy por hoy los sitios web sean uno de los nichos más apetecidos por los atacantes. Es que de alguna manera son un medio sencillo de lograr su cometido con relativamente poco esfuerzo y además con la posibilidad de masificar los resultados. Esto último porque de detectarse una vulnerabilidad en un tipo de sitio web -Joomla o Wordpress por mencionar dos de los más populares- potencialmente se puede aplicar la misma técnica para otros sitios que usen el mismo sistema base para su web.

Dicho esto, de todos los tipos de ataque que puede recibir un sitio web, quizá la desfiguración es uno de los que se ha vuelto más célebre, por el objetivo mismo que persigue. Es decir, si se pone por un lado el ataque que busca robar información, tiene mínimos efectos en la percepción que los usuarios del sitio pueden tener. En cambio una desfiguración, provoca un resultado instantáneo y generalizado en este sentido pues la gente claramente detecta que el sitio ha sido alterado. Claro que los efectos del robo podrían tener más impacto interno en la organización afectada, pero al menos no transmite tanto ni tan ampliamente como un cambio a la imagen del sitio. Es así que este tipo de ataque se ha utilizado con muchos fines, incluidos los políticos o ideológicos. Por ejemplo, si un sitio se destaca por ofrecer productos blancos, atacarlo para sugerir que el negro es mejor puede tener profundo impacto en la imagen y credibilidad de una organización.

Las razones por las que estos ataques se vuelven viables, también son múltiples, pasando desde el propio servidor web, hasta la propia aplicación. Y esta es un área tan amplia que, escapa totalmente al ámbito de éste

y muchos otros trabajos sobre el tema y por ello se aborda sólo ligeramente en este estudio. También se ha pensado en alternativas que no impliquen necesariamente buscar y corregir vulnerabilidades, sino en explorar medios que permitan una detección temprana de los ataques y alertar oportunamente a los administradores para que apliquen lo más pronto los correctivos necesarios.

2.1.1. La seguridad en los servidores Web

Como se había ya indicado, una de las áreas que suele ser fuente de vulnerabilidades para los entornos web, es el servidor web, entendiéndose por éste, no al equipo servidor, sino al software que una vez instalado y en ejecución, ofrece los servicios web para sitios, portales, aplicaciones web.

Por ser un software, inherentemente este contendrá siempre porciones de código que resulten en una vulnerabilidad, que no se explota, sino hasta que se detecta y normalmente una vez detectada se corrige y vuelve el ciclo hasta la próxima detección. Este período entre que se detecta y se corrige es el que suele permitir que atacantes con el conocimiento adecuado, las aprovechen para sus fines. Pero esta es sólo una de las realidades de los servidores web. Otras consideraciones están dadas también en cuanto al software mismo y su concepción. Es decir, no todos los servidores web fueron diseñados con los mismos objetivos ni con las mismas consideraciones, por ello hay unos que inherentemente pueden resultar más seguros que otros, como hay unos que resultan más rápidos que otros y más fáciles o difíciles de poner en marcha[7]. Todas estas variables -y otras más claro- deben ser tomadas en cuenta a la hora de poner en producción un/a sitio/portal/aplicación web pues claramente no todos los servidores serán igualmente apropiados para todas las aplicaciones. Esta falta de conocimiento o análisis previos hacen que comúnmente se generen fallas de seguridad por esta discordancia.

También existen sistemas de software complementarios -que pueden ser vistos como módulos- que han sido concebidos para agregar funcionalidades a un determinado servidor o incluso a una variedad de ellos. Entre estas funcionalidades está por supuesto la de la seguridad y por ende tienden a mejorar la robustez del servidor en sí. El problema está en que no es muy común que los administradores de los servidores web los conozcan o los usen.

También, por ser una pieza de software ejecutándose sobre un Sistema Operativo (SO), existe cierto trabajo de aseguramiento que se puede hacer a éste nivel en función de la seguridad. Se tiene desde técnicas hasta sistemas completos que pueden manejarse a nivel del SO y que pueden mejorar la seguridad del servidor web y sus contenidos. El problema en este caso es el mismo que el anterior, falta de conocimiento y/o aplicación[7].

No se puede decir entonces que no es posible asegurar adecuadamente un servidor web, pero la realidad es de que en la práctica es poco frecuente que se haga. El resultado es que el común denominador de los servidores web no están adecuadamente asegurados y por ende ofrecen un grado de propensión a los ataques.

Finalmente hay que aclarar que aún cuando un servidor web esté adecuadamente asegurado, no se garantiza que sea inmune a los ataques, principalmente porque la gran diversidad de sistemas web que suelen funcionar sobre ellos, lo impide. Es decir, es clara la complejidad de asegurar una variedad de sistemas, construidos por un -no menos diverso- grupo de personas/empresas, tan solo a nivel del SO y el servidor web.

2.1.2. La seguridad en las aplicaciones Web

Primero que todo, se condera a las aplicaciones web, como una definición genérica que puede englobar a páginas, portales, CMS, LMS y demás tipos de sistemas web que suelen ser publicados en Internet a través de un servidor web. De esta forma y por lo expuesto en el punto anterior, es claro que éste es un componente fundamental en el ámbito de la seguridad web.

Comenzando con lo más obvio, no es difícil inferir las implicaciones de seguridad que tendría lo bien o mal programada que pueda ser una aplicación web, pues para los desarrolladores experimentados, es conocido que en virtud de otros factores -principalmente el tiempo- suele ser descuidado el tema de la seguridad en la programación. Se obtienen entonces aplicaciones, funcionales, atractivas e incluso eficientes, pero poco seguras. Es común también que los desarrolladores deleguen o confíen la seguridad al SO y/o el servidor web.

Las aplicaciones web que son de uso generalizado; como por ejemplo Joomla, WordPress, Drupal, Moodle, entre otras; gozan de popularidad, pero esto mismo hace que sean más atractivas a un ataque. Lo paradójico es que éstas suelen ser también las más atendidas en función de la seguridad por parte de los propios equipos de desarrollo e incluso la comunidad mundial. El porqué de esta paradoja es simple: la sencilla falta de atención o mantenimiento a la aplicación web luego de haberse puesto en producción. Dicho de otra forma, es bastante común que las aplicaciones no sean actualizadas a las nuevas versiones a medida que salen. Suelen ser publicadas y elementalmente olvidadas. Claro, la gente las usa a diario, le publican novedades, noticias, le cambian incluso la imagen, pero no suelen preocuparse de actualizarlas. Las implicaciones son también simples de inferir: una aplicación desactualizada tiene y acumula vulnerabilidades que han ido siendo detectadas y explotadas en el tiempo, pero que no se han ido corrigiendo a través de la aplicación de las nuevas versiones que las solucionan.

Por otro lado, las aplicaciones web poco conocidas o realizadas *in-house* pueden tener la ventaja de que sólo son conocidas por el equipo de desarrollo y al no tenerse disponible el código fuente libremente, los atacantes pueden ver dificultada su labor en la detección de vulnerabilidades.

Pero esto es verdad sólo hasta cierto punto. De hecho este razonamiento es ampliamente utilizado en el debate del software propietario contra el software libre, aduciendo que el simple hecho de que el código fuente de los sistemas esté disponible para todos, hace que de forma más sencilla se detecten y exploten vulnerabilidades, lo cual no es así, al menos en la última parte de explotarlas. La disponibilidad para el estudio del código hace efectivamente que se pueda analizar en busca de problemas, pero el fin más común de esta práctica es la de solucionarlos, no el de explotarlos. Claro, están quienes desean explotarlos, pero el trabajo de estudio y análisis de un código comúnmente complejo, suele resultar mucho trabajo para alguien cuyo fin es únicamente el de atacar. Además dado que hay más gente buscando soluciones y no ataques, es más probable que, para cuando un mal intencionado encuentre la falla, un bien intencionado ya lo haya hecho e incluso corregido. Suele ser mucho más productivo para un atacante usar otras técnicas o sistemas para detectar las vulnerabilidades que no impliquen tanto conocimiento y labor como el estudio del código fuente[8]. Tampoco quiero decir que no pase y es por cierto una opción plausible.

Sin embargo, éstas aplicaciones web suelen carecer de lo que a las más populares y conocidas les sobra: gente. Las aplicaciones comunitarias crecen muchísimo más rápido que las particulares y al ser realizadas por una comunidad, tienen más calidad, después de todo si es un desarrollo particular, nadie más verá el código ni lo podrá criticar. Esto también hace que se descuiden muchos aspectos que pueden derivar en problemas de seguridad. Por supuesto existen sistemas realizados particularmente que son de excelente calidad y es importante mencionarlo.

Sea cual sea el caso, la realidad en cuanto a la seguridad de las aplicaciones web, es que sin importar su tipo o procedencia son propensas a ataques y dependiendo de varios factores, algunos de ellos prosperarán.

Por supuesto, existen técnicas y sistemas que permiten asegurar las aplicaciones y estos suelen depender del tipo de aplicación. Es decir, se pueden aplicar un conjunto de procedimientos conocidos para mejorar la seguridad de un sitio web con Joomla, uno parecido o diferente para asegurar un Drupal y están también los que pueden asegurar un conjunto de ellas. Pero al igual que con el caso de los servidores web, no las vuelve mágicamente invulnerables.

2.2. Ataques a sitios web

Existen trabajos que hablan más profundamente de los aspectos tratados, pero dado el objetivo de este trabajo, es importante centrarse en los ataques a los sitios web, más que en las formas de asegurarlos pues este trabajo se basa justamente en la premisa de lo que se puede hacer cuando -como es común- un sitio ha sido vulnerado.

Cuando se trata de atacar un sitio web, se pueden usar una variedad de técnicas, con una variedad de objetivos y por lo mismo, obtener una variedad de resultados, entonces existe diversidad de tipos de ataque a sitios web. CSI (Computer Security Institute por sus siglas en inglés) es una organización que desde hace casi 40 años se ha dedicado a analizar y estudiar los aspectos de la seguridad desde el punto de vista del administrador hasta el punto de vista del usuario final. Aunque realizan un evento/exhibición anual, es más conocida por sus entregas eventuales de resultados obtenidos conjuntamente con las oficinas del FBI de San Francisco denominadas *CSI/FBI Computer Crime and Security Survey*. [10]

En estas entregas se evalúan regularmente los principales factores y medios de ataque usados para vulnerar los sistemas informáticos y desde el 2004 se empezó a analizar y reportar resultados sobre ataques de desfiguración de sitios web. Ya en la edición del 2005 se reportó que los ataques por desfiguración tienen un costo bajo para la organización afectada, sin embargo ya en ese tiempo se empezó a detectar como uno de las áreas de más rápido crecimiento, quizá por el mismo hecho de considerarse “económicos”. Es decir, dado que se le considera un ataque de bajo impacto económico para la organización afectada, es razonable que se le preste poca atención a su prevención y contención[10].

Sin embargo hay que tener muy claro el hecho de que la desfiguración de un sitio web, más que un ataque es más propiamente el resultado de un ataque, y los medios usados para lograrlo son lo que realmente se debe considerar el ataque. Por otro lado, el objetivo de un ataque para la desfiguración de un sitio, aunque bien puede ser únicamente la propia desfiguración, puede también con cierta frecuencia, en realidad ser una

distracción para actividades más siniestras como el robo/eliminación de información y la propia infección del sitio con alguna clase de malware.

Como ya se indicó, los ataques propiamente dichos se pueden realizar por varios medios/objetivos, algunos de los más frecuentes han sido reportados por el CSI/FBI en su informe del 2010, junto con porcentajes de encuestados en de donde se extrajo la figura 1.

Types of Attacks Experienced
By Percent of Respondents

Type of Attack	2005	2006	2007	2008	2009	2010
Malware infection	74%	65%	52%	50%	64%	67%
Bots / zombies within the organization	added in 2007		21%	20%	23%	29%
Being fraudulently represented as sender of phishing messages	added in 2007		26%	31%	34%	39%
Password sniffing	added in 2007		10%	9%	17%	12%
Financial fraud	7%	9%	12%	12%	20%	9%
Denial of service	32%	25%	25%	21%	29%	17%
Extortion or blackmail associated with threat of attack or release of stolen data	option added in 2009				3%	1%
Web site defacement	5%	6%	10%	6%	14%	7%
Other exploit of public-facing Web site	option altered in 2009				6%	7%
Exploit of wireless network	16%	14%	17%	14%	8%	7%
Exploit of DNS server	added in 2007		6%	8%	7%	2%
Exploit of client Web browser	option added in 2009				11%	10%
Exploit of user's social network profile	option added in 2009				7%	5%
Instant messaging abuse	added in 2007		25%	21%	8%	5%
Insider abuse of Internet access or e-mail (i.e. pornography, pirated software, etc.)	48%	42%	59%	44%	30%	25%
Unauthorized access or privilege escalation by insider	option altered in 2009				15%	13%
System penetration by outsider	option altered in 2009				14%	11%
Laptop or mobile hardware theft or loss	48%	47%	50%	42%	42%	34%
Theft of or unauthorized access to PII or PHI due to mobile device theft/loss	option added in 2008			8%	6%	5%
Theft of or unauthorized access to intellectual property due to mobile device theft/loss	option added in 2008			4%	6%	5%
Theft of or unauthorized access to PII or PHI due to all other causes	option added in 2008			8%	10%	11%
Theft of or unauthorized access to intellectual property due to all other causes	option added in 2008			5%	8%	5%
2010 CSI Computer Crime and Security Survey					2010: 149 Respondents	

Fig 1. Tipos de Ataque Experimentados por porcentaje de encuestados.

Fuente: 2010/2011 Computer CrIme and SeCurIty Survey[11].

Se puede ver cómo los ataques de desfiguración de sitios entre los años 2005 y 2010 promedian un 8%, un valor relativamente alto, si se considera que algunos de los tipos de ataque indicados no necesariamente están orientados a la web, aunque sí pueden servir perfectamente como medio para la obtención de privilegios para una posible desfiguración.

Otro análisis interesante, de la misma fuente pero de años anteriores[10], es el monto de pérdidas provocadas y clasificadas por tipo de ataque. En aquel momento (2005) los Virus eran el número uno con cerca de 43 millones de dólares y los ataques de desfiguración estaban en el último lugar con 115 mil dólares, es lo que se considera el costo directo. Sobre estas cifras hay que considerar dos puntos fundamentales: por un lado, a diferencia de casi todos los demás, al momento sólo se había estudiado las desfiguraciones sólo por el lapso del último año y por otro lado y más importante aún está el hecho de que independientemente del costo económico, este tipo de ataques son los que suelen -en cambio- producir el mayor impacto a nivel de imagen a una organización y/o a el/los administrador/es del sistema. La percepción global de confianza que un usuario/cliente de una organización pueda tener, se ve gravemente comprometida con una desfiguración de su sitio web. Es a lo que se le considera el costo indirecto. Por esto es que mi criterio sobre los costos relacionados a este tipo de ataques es menos conservador que los análisis referidos que sólo se centran en los gastos operacionales de detección, contención y corrección del ataque. Este razonamiento cobra mayor fuerza cuando se trata de organizaciones financieras pues el usuario/cliente puede simplemente decidir llevarse su inversión a otra parte porque percibe -a través de ver el sitio de su proveedor desfigurado- que su dinero no está lo suficientemente seguro con esa institución.

2.3. Defacements o desfiguraciones

Si bien el trabajo de prevención y aseguramiento a nivel del servidor web son siempre bienvenidos, esto no siempre es suficiente para evitar una desfiguración y en éste sentido, una pronta detección tiende a minimizar cualquier impacto negativo que el ataque pudiera tener. Los sitios de contenido cada vez más dinámico tampoco ayudan a facilitar un adecuado monitoreo de cambios y el incremento en la complejidad del Internet y sus servicios no ha provocado nada más que un equivalente incremento en la complejidad de los ataques y los atacantes y por cierto también a la hora de defenderse de ellos.

Esto abre la necesidad de pensar en sistemas que ayuden a mejorar los tiempos de respuesta, pues independientemente de la gestión de prevención que se haga, hay que considerar que siempre es cierta la posibilidad de un ataque exitoso.

De quienes se dedican al estudio, análisis y reporte de sitios web desfigurados, Zone-H es aquel que podría considerarse el de mayor presencia y visibilidad a nivel global, pues aparte de ser muy conocido y usado por los especialistas del medio, lo he visto nombrado en la mayoría de artículos y documentos usados como estudio bibliográfico previos al presente trabajo, algunos de los cuales son parte de la bibliografía de esta tesis.

Según Zone-H, a nivel mundial, hubo un promedio de cerca de 1.5 millones de ataques de desfiguración por año entre el 2010 y 2013.

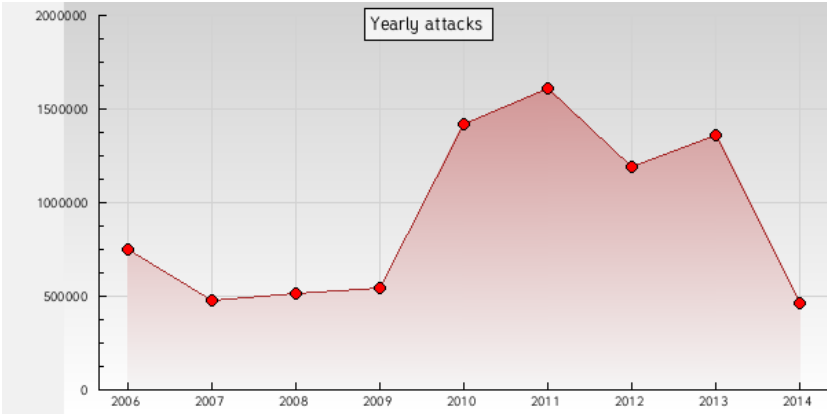


Fig 2. Ataques por año entre el 2006 y 2014 según Zone-H.

Alrededor de 100 mil ataques mensuales en lo que va de 2014, lo cual confirma la tendencia para este año:

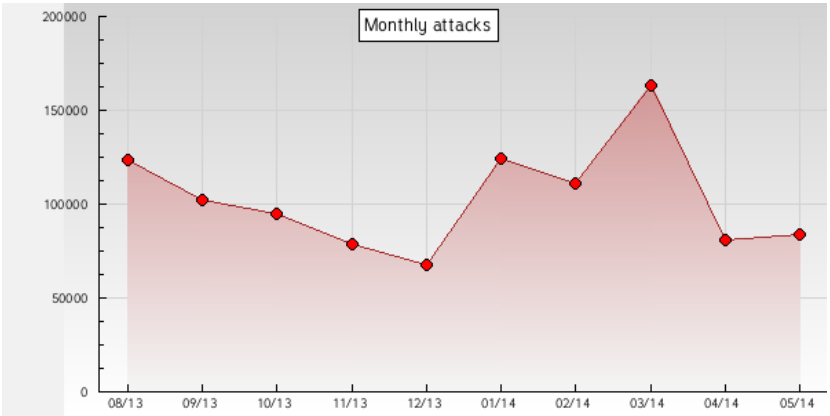


Fig 3. Ataques mensuales entre agosto de 2013 y mayo de 2014 según Zone-H.

A la fecha (mayo de 2014) el promedio diario durante el último mes alcanza unos 2500 ataques diarios:

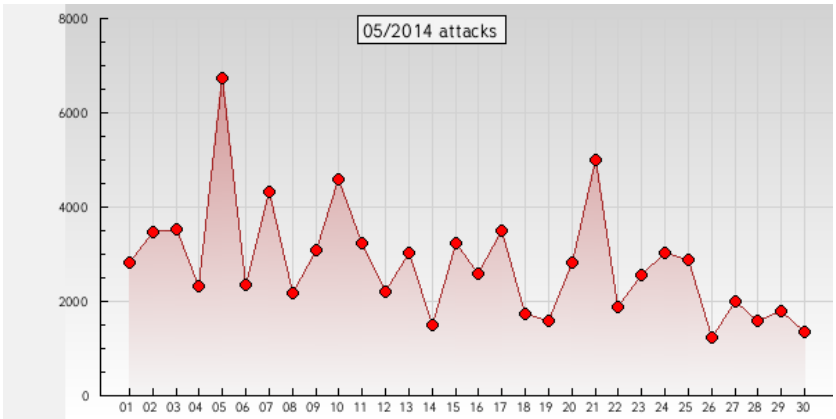


Fig 4. Ataques diarios del mes de mayo de 2014 según Zone-H.

Es claro que los ataques por desfiguración de sitios web se han más que duplicado, pasando de poco más de medio millón en 2009, a superar ampliamente el millón de ataques a partir del 2010, incluso sobrepasando el millón y medio en 2011, y no se ve una tendencia a disminuir.

Otro aspecto muy importante y que tiene también directa relación con el objetivo de este trabajo, es el tema de la permanencia o duración del ataque. Debido justamente a los factores ya expuestos, especialmente la falta de un sistema automatizado de detección y alerta temprana de desfiguraciones, en casi la mitad de los sitios atacados, los efectos de estos ataques suelen permanecer por un lapso considerable de tiempo, con un 43% de los ataques durando al menos una semana y más del 37% permaneciendo luego de la segunda semana[12], y no es necesariamente debido al tiempo de reacción o el tiempo que le toma al equipo técnico solucionar el inconveniente, sino más bien tiene que ver con el tiempo que pasa antes de que el equipo se percate siquiera del problema o pueda ser alertado del mismo por terceras partes.

2.4. Detección de cambios en un sitio web

Detectar cambios en un sitio web, por parte de un ser humano, es bastante sencillo y más aún si ésta persona está familiarizada con el sitio web en cuestión. Un simple vistazo puede ser suficiente para detectar un cambio anómalo en el sitio. Pero el escenario es poco práctico pues sería muy costoso tener a alguien tan constantemente pendiente de un sitio. Adicionalmente, la sencillez de éste proceso de detección es tal, gracias a la inteligencia y poder de análisis del ser humano, algo que en sistemas de cómputo es bastante complejo de lograr.

Entonces, cuando se trata de un sistema autónomo de detección de cambios de un sitio web, se requiere realizar tareas que puedan acercarse un poco a ese poder analítico deseado, lo cual no se podría lograr si no se toman al menos un par de enfoques diferentes pero complementarios para el análisis de cambios a un sitio web, o más bien dicho, a una página web -típicamente la portada del sitio-, pues no es raro que cambios sustanciales a nivel de código fuente, no provoquen mayores diferencias visualmente, por ejemplo cuando se cambia la filosofía de diseño de un sitio de tablas (<table>) a capas ( ).

2.4.1. El sitio web visto como código HTML

Cuando un usuario realiza la petición de visitar un sitio web, a través del navegador digita/selecciona la URL y sin importar la tecnología usada del lado del servidor para construir y desplegar el sitio web, lo que recibe sin excepción, es código HTML que es interpretado y presentado gráficamente por el propio navegador, tal cual se grafica a continuación:

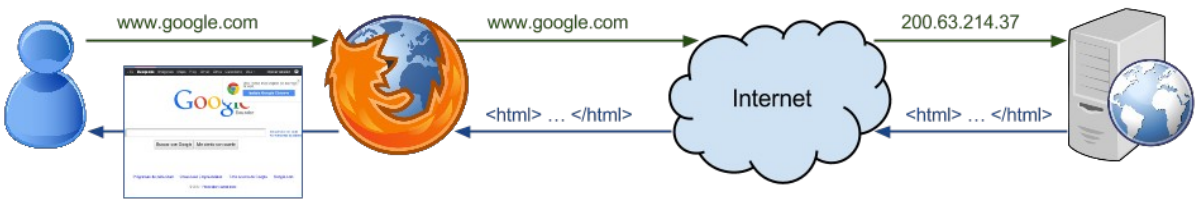


Fig 5. Proceso de solicitud, respuesta e interpretación de una petición web.

Entonces, es bastante lógico que una aproximación común, para detectar cambios de un momento a otro en el tiempo, sea la de buscar los cambios producidos en el código HTML recibido del mismo sitio en dichos momentos separados.

Este análisis de código fuente puede realizarse con diversos métodos y criterios, que van desde los muy simplistas hasta los muy complejos.

Quizá la aproximación más sencilla sea la de un simple conteo de caracteres, palabras y/o líneas, pero como es de esperarse, estos resultados no arrojan información muy valiosa pues puede darse el caso de que un sitio se vea totalmente diferente con un número muy similar o igual de caracteres, palabras y/o líneas.

Aproximaciones muy complejas tampoco son lo ideal dado que siempre, la velocidad y los recursos empleados para el trabajo, son un factor importante a tomarse en cuenta. Analizadores sintácticos avanzados pueden realizar un trabajo muy profundo sobre el código, pero al ser el HTML más bien orientado a la simplicidad, existe digamos, un desperdicio de recursos.

Como en muchas cosas, el reto está en la búsqueda del equilibrio que logre los mejores resultados con la mayor simplicidad, e idealmente con el menor costo en cuanto a tiempo y recursos.

El compromiso es analizar y determinar la herramienta de SL que se ajuste a ésta premisa y que permita obtener resultados que, de una forma automatizada, determinen con un cierto grado de certeza la probabilidad de cambios no intencionados en el código de un sitio. Por supuesto, los cambios naturales en el sitio pudieran ser constantes y detectar un cambio anormal es en definitiva el reto. Para esto -y por adelantado, también para el siguiente punto- deberá manejarse un período de ajuste donde eventualmente se puedan determinar las tendencias normales de cambio del sitio y en base a ello, el análisis de cambios podrá arrojar apreciaciones más certeras.

2.4.2. El sitio web visto como una imagen

Como ya se anticipó en el punto anterior, si bien técnicamente lo que viaja de un sitio por la red, es código HTML, lo que la gente espera a la larga no es un grupo de caracteres sino la representación gráfica de ello, dígase la visualización de la página web. Y es donde realmente se hacen evidentes los cambios, intencionales o no, del código.

Podría pensarse que ya con el analizador de código sería suficiente para detectar un cambio, pero no es así. Tener un método adicional y complementario de análisis, contribuirá a mejorar los índices de confiabilidad y de certeza sobre un cambio. Es más, sería deseable crear más métodos alternativos de análisis de cambios, como idea simple, mientras más mejor, pero sólo hasta justo antes de volverlo demasiado complejo. Pero como una primera aproximación, tener dos métodos basados en las dos formas principales del resultado final de un sitio web, conducirá a resultados que puedan contrastarse y analizarse, e incluso ayuden en la selección de potenciales métodos adicionales.

Sobre el análisis imagenográfico de un sitio web, se puede inferir un grado de complejidad mayor al del código, porque existen dos retos separados en sí: está la captura automatizada de una imagen del sitio, tal cual se vería

en un navegador, y por otro lado la ciencia detrás de cómo comparar dos imágenes, algo que, podría parecer muy simple pues el cerebro humano lo hace constantemente, pero para una máquina o sistema puede no serlo.

Se deberá entonces buscar las herramientas que permitan tanto capturar visualmente el sitio y luego analizar dichas capturas en busca de cambios. Habrá que finalmente determinar una forma de obtener un valor cuantitativo que sea resultado del cambio, un valor que se pueda medir, pues un simple sí son iguales o no, no será demasiado útil, habrá que tratar de determinar qué tan similares/diferentes son.

2.5. El Software Libre como plataforma de trabajo

Quizá una de las formas más viables de generar soluciones, sea la del uso de SL para buscar herramientas y usarlas/adaptarlas para construirlas. Erróneamente se piensa que se puede construir tecnología con herramientas privativas, pero lo que termina haciéndose en realidad, es usar soluciones preexistentes y acomodando muchas veces las necesidades reales al esquema pre planteado por ellas. Eventualmente se combinan dos o más para mejorar el espectro de la solución, pero nuevamente es usar tecnología, tiene poco que ver con crearla.

La mayoría de la personas piensa que la principal ventaja del SL es su costo reducido o inexistente en contraste con el cuasi omnipresente costo en las opciones privativas, pero esta es sólo una de las diferencias y no necesariamente la mejor. Para quienes vienen usando y explotando el SL por alrededor de una década y media, es claro que la libertad de estudio, uso y modificación del SL, es en realidad la que abre todo un mundo de posibilidades y específicamente, abre la posibilidad de crear, no sólo de usar. El simple hecho de adaptar una herramienta preexistente a una necesidad específica, ya significa una enorme diferencia respecto a tan solo usarla[13].

Hacer uso de las herramientas que, luego del respectivo análisis, se determine como las más viables para generar una solución de detección automatizada de cambios a sitios web, será justamente un ejemplo de aquello, pues sobre la base de dichas herramientas se construirá un prototipo de sistema que las use y con los resultados obtenidos entregue resultados/alertas al administrador del sitio.

El uso y la creación de un proyecto de SL, con una licencia acorde a dicha filosofía, posibilita además la colaboración y extensibilidad del producto, que de hecho es uno de los objetivos a mediano plazo del mismo.

2.5.1. Linux

El SL no es nuevo, de hecho, el software nació siendo libre, menospreciado quizá, como un mero anexo del hardware, sin costo y para muchos transparente y sin valor apreciable. Pero no pasó mucho para que las potencialidades comerciales fueran visualizadas y empezaran a ser explotadas, alcanzando, en algunos casos, escenarios extremadamente complejos de licenciamiento que buscan más que cualquier cosa el sentido comercial. Esto hizo que, prácticamente desde el inicio, hubiesen quienes abogaban por la re-liberación del software. De ellos, el movimiento que por mucho tuvo más éxito, aceptación y promoción fue el de la FSF[14] fundada por Richard Stallman y que ayuda a desarrollar el sistema GNU, que se uniera con el kernel de Linus Torvalds[15] para crear -en mi concepto- la punta de lanza del SL.: GNU/Linux, muy a pesar de Stallman, más comúnmente conocido simplemente como Linux.

El éxito de Linux fue casi instantáneo y su divulgación y uso se siguen extendiendo, siendo al momento y por mucho, el software más ampliamente desplegado en el mundo; más aún desde hace unos años, con la masiva distribución y uso de, la que por mucho es, la más popular de las ramas de Linux: Android, que en 2013 alcanzó 1.3 millones de activaciones diarias. Además de las 700 mil TV vendidas a diario con Linux, el 92% de las computadoras de alto rendimiento en el mundo, corren Linux, el 85% de los mercados de intecambio de valores a nivel mundial usan linux -sólo el de Chicago procesa 1000 billones de transacciones diariamente- y finalmente: Google, Facebook, Amazon son ejemplos de renombre de empresas que usan y/o usaron Linux desde sus inicios y otras como IBM que le están dando gran impulso al SL [16].

La oportunidad de usar un mismo S.O., en virtualmente cualquier plataforma de hardware, es sin duda una característica muy atractiva para el proyecto, pues permite que las herramientas que se analicen, usen, desarrollen, sean también portables y abran el espectro de plataformas de uso, desde equipos pequeños como un Raspeberry PI -del tamaño de una tarjeta de crédito y un costo de unos 50usd-, hasta supercomputadoras de miles de dólares. Simplemente se estaría creando una solución que minimice la necesidad de adquirir un hardware específico, así como la posibilidad de modificarlo y extenderlo para soluciones más o menos específicas.

2.5.2. Bash y LAMP

Bash[17] es uno de los lenguajes que están específicamente pensados y desarrollados con el fin de manejar archivos/procesos y es tradicionalmente el entorno natural de ejecución de las herramientas disponibles a ser analizadas, por ende es el ambiente lógico para el trabajo, dado que en definitiva es con archivos -de código y de imagen- que trabajaremos. Será el entorno para el trabajo medular del sistema. Las interfaces de usuario deberán realizarse en un entorno de más alto nivel para proveer de una imagen y funcionalidad amigables. La forma más adecuada de ofrecer esta interface es a través de un sistema web que facilite su uso desde cualquier parte y entorno, desde dispositivos móviles hasta computadores de escritorio, donde quiera que se disponga de una conexión a Internet. En ésta línea y manteniéndose en el SL, Linux+Apache+MySQL+PHP (LAMP)[18] es la combinación más apropiada, principalmente porque PHP es un lenguaje muy cercano a Bash y permite una integración digamos, natural, con el uso e interacción con la CLI de Bash. Apache y MySQL suelen simplemente ser los compañeros habituales de PHP, pero son meramente referenciales porque igualmente se podría usar otro servidor web -como lighttpd o nginx- y un motor de base de datos más simple como SQLite o más complejo como PostgreSQL, será simplemente extender el sistema con los módulos/plugins apropiados.

Capítulo 3 :: ANÁLISIS COMPARATIVO DE ALTERNATIVAS

Si bien a este punto, la idea del proyecto es bastante clara y se visualiza perfectamente el ámbito de las herramientas que van a buscarse y analizarse, se considera importante detallar los criterios que se han definido para la preselección de las herramientas de análisis de código e imagen:

- La herramienta debe distribuirse libremente bajo una licencia de SL.
- La herramienta puede o no tener costo, pero se preferirá una gratuita si comparte características similares con otra.
- La herramienta debe ser un proyecto activo y mantenido de forma razonablemente periódica.
- Tendrán preferencia aquellas que se encuentren en un estado maduro/estable de desarrollo.
- La herramienta deberá correr al menos sobre Bash o al menos ofrecer una interface/API de programación que la vuelva accesible desde Bash.
- Las herramientas que pertenezcan, sean parte o estén disponibles en repositorios para una o varias distribuciones de Linux, cumplirán por lo mismo la mayoría de las premisas previas y por ende serán más opcionadas.
- Finalmente, la herramienta debe poder ser usada por CLI (Bash), dado que se requiere un grado de automatización en su uso, una interfaz gráfica será inútil.

Con estos criterios se ha realizado una búsqueda previa, de la cual se desprendió una lista de herramientas que, sin ser analizadas técnicamente, cumplieron con los criterios y pasaron a ser estudiadas y analizadas a continuación.

3.1. Captura y análisis de código HTML

En ésta categoría se encontraron decenas de herramientas, pensadas específicamente en la búsqueda de diferencias entre archivos. La mayoría de ellas son gráficas y están orientadas al código fuente con el fin de generar parches y ayudar en el mantenimiento de versiones del código, pero hacen poco en cuanto a determinar qué tanto han cambiado o qué tan relevantes éstos cambios pueden ser. Quizá era mucho pedir, con un deseo interno de encontrar una solución más completa a la necesidad específica del proyecto.

3.1.1. Métricas

Para las pruebas y mediciones, se preparó un equipo virtual con una instalación mínima de CentOS 6. El equipo dispone de un procesador QEMU Virtual CPU version 1.5.3 a 2.7MHz, 1GB de RAM + 1GB de Swap y un disco de 20GB. El mismo equipo ha sido usado para todas las fases de prueba y para las mediciones de rendimiento se ha cuidado que nada más que lo esencial se encuentre corriendo en el mismo, por lo cual se puede partir de una carga promedio de 0.0 en todos los casos.

Dicho esto, igualmente se han definido criterios de comparación, que permitan una calificación técnica y lo más imparcial posible y por supuesto se contemplan sólo aquellas que ya han cumplido con los criterios generales antes indicados.

- **P1: Tiempo de ejecución.** Será medido principalmente en base a la ejecución de los comandos antecedidos de `time` lo cual permite obtener información bastante detallada de los tiempos consumidos. Se reportará el tiempo `real` usado. El tiempo menor obtendrá 5 y los 3 subsecuentes obtendrán 4, 3 y 2 puntos respectivamente, los demás 1 punto cada uno.
- **P2: Consumo de recursos.** Simultáneamente a la ejecución de la prueba, se monitoreará el consumo de recursos con `top`. Se medirán cambios en consumo de memoria y cpu. Se procederá con la calificación con la misma filosofía que el tiempo de ejecución, aunque en este caso, si los cambios no son perceptibles o mínimos se obtendrá 5.
- **P3: Arroja resultados numéricos usables.** Si de alguna forma la herramienta arroja estos valores, se obtendrá 3 puntos, si se el cálculo adicional es sencillo-medio tendrá entre 2 y 1 y si no un 0. Esto en razón de que de no entregarlos, el cálculo complejo deberá realizarse como parte del prototipo y los resultados pueden no necesariamente ser consistentes con los valores.
- **P4: Nivel de complejidad de aprendizaje e implementación.** Este valor es de tipo más bien cualitativo y de carácter subjetivo. Se calificará con un valor entre 1 y 3 de acuerdo al grado de complejidad percibido durante el proceso de análisis y pruebas.
- **P5: Disponibilidad de paquete instalable nativo o complejidad de creación de uno.** Al igual que el ítem sobre valores numéricos, se calificará con 2 puntos a la que tenga los medios o esté disponible en paquetería pre compilada y con 0 las que no. Esto en función de que si bien es posible armar un paquete instalador de las herramientas que no lo tengan, no siempre es igual de sencillo.
- **P6: Relevancia/Precisión de los datos que emite.** Este ítem también es de carácter cualitativo y se medirá en función de qué tan útiles se considera que serán los datos emitidos o que se pueden conjeturar de cada herramienta. Es decir, no es igual que se pueda medir el número de cambios en función de líneas, que de caracteres, de bytes o número absoluto de cambios. Se calificará la relevancia/precisión con un peso de entre 1 y 5, siendo 5 aquella que disponga de la mayor precisión/relevancia y 1 la de menor.

Para las pruebas respectivas, se usaron dos archivos de código HTML de un sitio web que tuvo cambios naturales en un período de tiempo corto. Los archivos se llaman: sitioV1.html y sitioV2.html.

3.1.2. Captura de código HTML

La herramienta escogida para la captura de código HTML es `wget`, pues en base a los años de experiencia con Linux, es la que permite hacerlo con la mayor versatilidad, eficiencia y simplicidad. Si bien pudieran existir otras herramientas que hagan lo mismo, como el mismo cURL, `wget` es la de uso más común y extendido y es de hecho una de los miembros elementales del kit del administrador Linux. Y dado que el objetivo primario del estudio no es la obtención del código, sino las herramientas para su análisis, no hay necesidad de desplegar una búsqueda de alternativas luego del trabajo de investigación previo que pasó por el análisis de herramientas como:

- cURL,
- lftp,

- HTTrack,
- aria2,
- uGet y
- wget

De ellos el único que, entre varias opciones o modos de operación, dispone de la capacidad de realizar una descarga recursiva del sitio completo de una forma tan simplificada es `wget`. En éste caso se usa éste sencillo operador `-p` que ofrece la muy útil capacidad de descargar el sitio completo (no sólo el HTML del archivo inicial):

```
# wget -p http://www.elsitioweb.com
```

Esto genera un directorio llamado `www.elsitioweb.com` conteniendo el código HTML de la página inicial, así como una copia de todos los recursos relacionados con la misma, esto es, archivos JavaScript, hojas de estilo, etc. Pero `wget` también dispone de un operador `-O` que permite que todo este resultado se almacene consolidado en un solo archivo monolítico:

```
# wget -p http://www.elsitioweb.com -O elsitio.out
```

Esto puede permitir que se analicen cambios no sólo a nivel de código HTML puramente, sino también a nivel de sus elementos enlazados, pues todo está contenido dentro del archivo `elsitio.out` (la extensión es meramente referencial pues tanto el nombre como la extensión pueden ser cualquiera).

Lo siguiente será, una vez disponiendo de una "copia" diaria del código fuente de un sitio, poder comparar cada una con su predecesora y determinar los cambios producidos.

3.1.3. Análisis de cambios en código HTML

3.1.3.1. Diff

Esta es una herramienta de comparación de archivos basada en líneas y no en caracteres. Si se compara un archivo con otro, diff trata de encontrar el menor conjunto de inserciones y eliminaciones para crear un archivo a partir de otro. El resultado de la ejecución se muestra en formato estándar de tal forma que tanto humanos como máquinas puedan visualizar e interpretar los resultados y eventualmente aplicarlos. Asumiendo que se tiene uno de los archivos y los cambios, el otro archivo puede ser generado. Frecuentemente se usa para la generación de los denominados parches, que permiten mantener diferentes versiones de un mismo archivo, por lo cual es la herramienta más ampliamente usada por los administradores de sistemas para estos fines y más aún por los programadores de sistemas.

P1: El tiempo de ejecución alcanzó únicamente 1 centésima de segundo:

```
[pbernal@tesis ~]$ time diff sitioV1.html sitioV2.html
265a266
```

```
> <a href="#" rel="slide-1-2" class="tab carousel-jumper ">02</a>
270a272,283
…
… [DETALLE DE CAMBIOS OMITIDO]
…
> <img src="http://l.yimg.com/a/i/us/we/52/30.gif" width="31" height="31" alt="Parcialmente
nublado" title="Parcialmente nublado" />

real    0m0.010s
user    0m0.007s
sys     0m0.003s
```

P2: Una ejecución tan acelerada no permite percibir ningún cambio a nivel de rendimiento. Por lo cual, para tener una percepción un poco más real, se realizó una ejecución del 10 mil repeticiones. En este caso el tiempo fue de 27.949 segundos, que da un promedio de algo menos de 3 milisegundos por corrida:

```
[pbernal@tesis ~]$ time for i in {1..10000};do diff sitioV1.html sitioV2.html> /dev/null;
done

real    1m39.835s
user    0m59.825s
sys     0m40.509s
```

El cambio en el rendimiento general fue pequeño durante el casi medio minuto que tomó la corrida de las 10 mil comparaciones subió de 0.00 a 0.02:

```
top - 14:50:35 up 2 days, 18:44,  2 users,  load average: 0.00, 0.03, 0.09
Tasks:  89 total,   1 running,  88 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020232k total,   267004k used,   753228k free,   103548k buffers
Swap:  1048568k total,        0k used,  1048568k free,    62708k cached
```

```
top - 14:52:02 up 2 days, 18:46,  2 users,  load average: 0.02, 0.03, 0.08
Tasks:  90 total,   2 running,  88 sleeping,   0 stopped,   0 zombie
Cpu(s): 30.1%us, 21.7%sy,  0.0%ni, 48.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.2%st
Mem:   1020232k total,   267424k used,   752808k free,   103556k buffers
Swap:  1048568k total,        0k used,  1048568k free,    62708k cached
```

Fig 6. Carga promedio y consumo de recursos antes y después de las 10 mil ejecuciones de diff.

P3: No arroja resultados numéricos directamente usables, pero puede aplicarse un algoritmo de cálculo que estime el porcentaje de cambio en base al tamaño total del archivo y el número cambios detectados.

P4: El grado de complejidad en el aprendizaje y uso de la herramienta es bastante bajo. Realmente cualquiera con nociones básicas de CLI puede usarla en pocos minutos. Sin embargo hay que considerar una penalización porque se requiere implementar el cálculo de un valor numérico de cambios.

P5: Diff está disponible en prácticamente todas las instalaciones Linux por defecto, si no lo estuviera, el paquete `diffutils` la provee con tan solo instalarlo.

P6: Si bien la precisión de los resultados es muy alta, lo que se puede obtener de diff es el número de líneas cambiadas, y se podría relacionar éste con el número total de líneas del archivo original, lo cual le penaliza en cuanto a relevancia.

3.1.3.2. Pretty-Diff

Pretty-Diff[19] es una herramienta desarrollada en JavaScript (JS) porque está orientada mayormente para funcionar en entornos web. Dispone de varias características muy interesantes como la del "embellecimiento" y la minimización de código, así como la capacidad de realizar comparación de archivos de varios tipos y emitir un reporte en formato HTML de las diferencias detectadas.

Si bien todas estas características son muy útiles para el desarrollador, sólo la de diferenciación es de interés para este proyecto y justamente la emisión por defecto de un muy completo reporte de las diferencias no es necesariamente la forma más ideal para los fines del proyecto, aunque sí podría ser un punto muy a favor para el sistema web pues el administrador podría revisar visualmente de los cambios.

Puede ejecutarse a través de varios medios, entre los cuales está el de NodeJS[20] que permite justamente correr scripts JS vía CLI, pero adiciona complejidad a la hora de implementar porque se requiere primero desplegar la plataforma de NodeJS.

P1: El tiempo de ejecución alcanzó algo más de los 2 segundos:

```
[pbernal@tesis ~]$ time node Tesis/3.1.2/Pretty-Diff-master/api/node-local.js
source:sitioV1.html readmethod:filescreen diff:sitioV2.html lang:html
…
… [DETALLE DE CAMBIOS DE COMANDO SUPRIMIDA]
…

real    0m2.154s
user    0m1.810s
sys     0m0.374s
```

P2: Nuevamente se envió una ejecución repetitiva, pero dado el tiempo de ejecución de cada una, esta vez se realizaron sólo 100 iteraciones, igualmente suprimiendo la salida. El tiempo total de ejecución alcanzó poco más de los 3 minutos y medio, confirmando los 2.1 segundos por corrida:

```
[pbernal@tesis ~]$ time for i in {1..100};do node Tesis/3.1.2/Pretty-Diff-master/api/node-
local.js source:/home/pbernal/Tesis/TT/sitioV1.html readmethod:filescreen
diff:/home/pbernal/Tesis/TT/sitioV2.html lang:html > /dev/null; done

real    3m34.121s
user    3m0.429s
sys     0m35.844s
```

El cambio en el rendimiento general fue apreciable pues la carga general del sistema subió de 0.00 a 0.96:

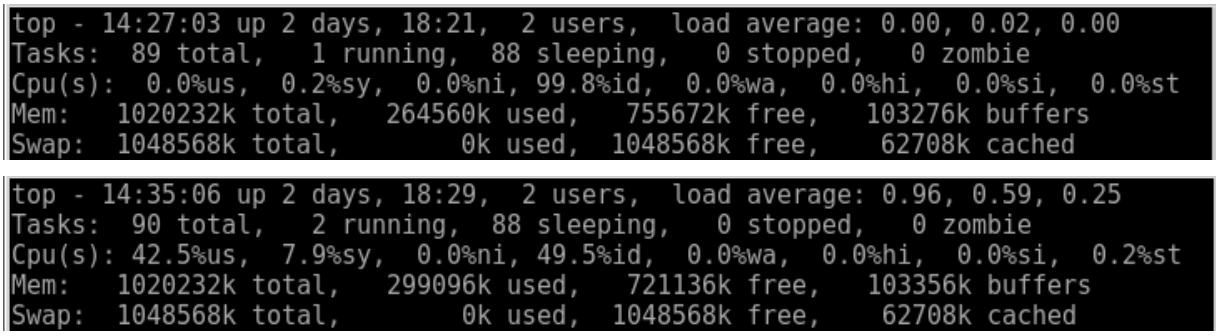


Fig 7. Carga promedio y consumo de recursos antes y después de las 100 ejecuciones de Pretty-Diff.

P3: Sí arroja resultados numéricos, pero no son directamente usables, es decir se emiten como parte del reporte detallado en HTML, pero puede modificarse programáticamente la API `node-local.js` para que omita la salida HTML y genere sólo los valores numéricos.

P4: El grado de complejidad en el aprendizaje y uso de la herramienta es medio porque se requieren conocimientos previos de programación y JavaScript, pero de ser así, la herramienta se puede aprender a usar e incluso adaptar en cuestión de horas. En este caso también hay que considerar una penalización porque se requiere realizar cambios en el código de la API para la emisión de valores numéricos de cambios.

P5: Pretty-Diff, al tratarse de una herramienta basada en JavaScript, no está disponible en formato RPM. La realización de un RPM si bien se estima de complejidad baja, no se percibe muy necesaria pues basta con descargar la librería y empezar a usarla con NodeJS, por lo cual se requiere de la instalación previa del paquete `nodejs`, que sí está disponible en formato RPM en el repositorio de EPEL[21].

P6: Si bien parece no ser la herramienta más eficiente en razón de tiempos y consumo de recursos, sí es la que se destaca por la razón de cambios otorgada casi directamente, es más, provee no sólo el número de líneas cambiadas, sino también el número de cambios dentro de ellas.

3.1.3.3. Cmp

Al igual que `diff`, `cmp` está disponible en todas las distribuciones Linux pues es una de esas herramientas heredadas de Unix. Compara dos archivos de cualquier tipo y emite el resultado en la salida estándar. Por defecto, si no hay diferencias, cmp no emite ningún resultado, en caso contrario se reporta el byte y número de línea en que se encontró la primera diferencia. Actualmente se usa la versión GNU de `cmp`.

Su principal particularidad es que realiza una comparación byte-a-byte, y reporta todos los bytes que difieren entre los archivos. Esto puede parecer una ventaja en un inicio, pero luego de unas pocas pruebas preliminares, con archivos muy simples, se vió que ésto puede ser un problema cuando se trabaja con archivos HTML donde hay caracteres no significativos como los espacios y líneas en blanco ó los saltos de línea. De escogerse ésta herramienta y para realizar una comparación más precisa debería realizarse primero una “limpieza” previa a los archivos, de todos estos caracteres no significativos.

P1: El tiempo de ejecución llegó casi a la décima y media de segundo:

```
[pbernal@tesis ~]$ time cmp -l sitioV1.html sitioV2.html
16193 163  42
16194 143  40
…
[DETALLE DE CAMBIOS DEL COMANDO OMITIDO]
…
70295 154 157
70296  76  46

real    0m0.144s
user    0m0.022s
sys     0m0.083s
```

P2: Al igual que con `diff`, por el tiempo pequeño de ejecución, para poder medir el consumo, se envió a ejecutar 10 mil veces el mismo comando,. El tiempo de ejecución llegó a los 41 segundos y medio, lo cual arroja un promedio de sólo 4 milésimas de segundo por corrida:

```
[pbernal@tesis ~]$ time for i in {1..10000};do cmp -l sitioV1.html sitioV2.html > /dev/null;
done

real    0m41.567s
user    0m13.591s
sys     0m26.481s
```

Ésta diferencia con lo medido anteriormente se puede explicar en el hecho de que para esta segunda corrida se omitió la salida del comando. Finalmente se obtuvo un cambio mínimo de 0.07 en la carga promedio del equipo:

```
top - 20:01:15 up 2 days, 23:55,  2 users,  load average: 0.00, 0.02, 0.00
Tasks:  89 total,   1 running,  88 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.2%sy,  0.0%ni, 99.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020232k total,   599040k used,   421192k free,   109424k buffers
Swap:  1048568k total,        0k used,  1048568k free,   376048k cached
```

```
top - 20:02:00 up 2 days, 23:56,  2 users,  load average: 0.07, 0.01, 0.00
Tasks:  89 total,   1 running,  88 sleeping,   0 stopped,   0 zombie
Cpu(s): 16.2%us, 37.9%sy,  0.0%ni, 45.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.7%st
Mem:   1020232k total,   601072k used,   419160k free,   109432k buffers
Swap:  1048568k total,        0k used,  1048568k free,   376048k cached
```

Fig 8. Carga promedio y consumo de recursos antes y después de las 10 mil ejecuciones de cmp.

P3: Si bien cmp no arroja valores numéricos usables directamente, sí permite realizar un cálculo rápido del número de bytes modificados con un simple conteo de las líneas que emite como resultado pues cada línea informa los detalles de cada byte encontrado diferente. En resúmen con tan solo “entubar” la salida de `cmp` hacia el comando `wc -l` se obtendrían los bytes cambiados, que podrían contrastarse con el tamaño del archivo original.

P4: El grado de complejidad en el aprendizaje y uso de la herramienta es mínimo y literalmente en menos de 30 minutos uno puede empezar a obtener resultados. Igualmente para ésto se requieren nociones básicas de administración de sistemas Linux.

P5: Al igual que `diff` éste es un comando directamente disponible con una instalación típica de cualquier distribución de Linux, de hecho viene en el mismo RPM que `diff`, el `diffutils`.

P6: Cmp entrega un reporte muy detallado de los bytes que han sido modificados de un archivo a otro, sin embargo como ya se anotó antes, estos cambios podrían ser no significativos para el caso de este estudio, por ejemplo el byte de una letra en minúscula es diferente al de la misma letra en mayúscula, pero para HTML simplemente no hay diferencia. Pero hay que considerar también que éstos cambios sutiles no son comunes en un ataque de desfiguración, es decir, ningún atacante se tomará la molestia de cambiar la capitalización del código pues ésto no genera un cambio visual en el sitio.

3.1.3.4. Comm

Es una herramienta de comparación de archivos ordenados, lo cual le agrega una característica interesante a este estudio, pues los cambios normales en un sitio seguirían un orden específico dentro del código HTML del mismo. Al mismo tiempo podría ser una desventaja para sitios que cambien mucho normalmente. Al igual que `diff`, Comm detecta y reporta las líneas que han sido modificadas entre un archivo y otro, en 3 modos diferentes: las líneas que son únicas en el primer archivo, las que son únicas en el segundo archivo y las que son comunes en los dos archivos; pudiéndose suprimir cualquiera de los valores reportados. En este caso, para las pruebas, se usó la opción combinada que suprime las líneas únicas en ambos archivos dado que así se sabrá las que se mantuvieron sin cambios.

P1: El tiempo de ejecución fue de sólo 8 milésimas de segundo:

```
[pbernal@tesis ~]$ time comm -12 sitioV1.html sitioV2.html
       <a href="#" rel="slide-1-2" class="tab carousel-jumper ">02</a>
…
[DETALLE DE CAMBIOS DEL COMANDO OMITIDO]
…
<img src="http://l.yimg.com/a/i/us/we/52/34.gif" width="31" height="31" alt="Parcialmente
nublado" title="Parcialmente nublado" />

real    0m0.008s
user    0m0.005s
sys     0m0.003s
```

P2: Como todas las demás, se realizó una corrida de 10 mil repeticiones suprimiendo la salida, el tiempo de ejecución alcanzó el minuto con casi nueve segundos, siendo el promedio de ejecución de casi 7 milésimas de segundo:

```
[pbernal@tesis ~]$ time for i in {1..10000};do comm -12 sitioV1.html sitioV2.html >
/dev/null; done
```

```
real    1m8.766s
user    0m26.044s
sys     0m41.738s
```

Un valor muy cercano a lo obtenido en las corridas sencillas. la carga promedio del sitio subió también muy poco, sólo 0.03, manteniendo la tendencia de las herramientas propias del S.O:

```
top - 16:40:46 up 3 days, 20:34,  2 users,  load average: 0.00, 0.00, 0.00
Tasks:  89 total,   1 running,  88 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020232k total,   617308k used,   402924k free,   121168k buffers
Swap:  1048568k total,        0k used,  1048568k free,   379196k cached
```

```
top - 16:42:01 up 3 days, 20:36,  2 users,  load average: 0.03, 0.01, 0.00
Tasks:  89 total,   1 running,  88 sleeping,   0 stopped,   0 zombie
Cpu(s): 16.6%us,  2.5%sy,  0.0%ni, 84.9%id,  0.0%wa,  0.0%hi,  0.0%si,  1.0%st
Mem:   1020232k total,   617596k used,   402636k free,   121168k buffers
Swap:  1048568k total,        0k used,  1048568k free,   379196k cached
```

Fig 9. Carga promedio y consumo de recursos antes y después de las 10 mil ejecuciones de comm.

P3: Como la mayoría de las herramientas estudiadas al momento, `comm` no arroja directamente valores significativos para el propósito del proyecto, sin embargo igualmente se puede establecer un valor numérico en base a una cálculo que relacione el número de líneas cambiadas/comunes con el total de líneas del archivo.

P4: En la misma línea de las demás herramientas de consola de la era Unix y/o GNU, comm es sencilla de aprender y usar para una persona con habilidades administrativas básicas, aunque como ya se dijo, requiere algo de penalización por el trabajo adicional que requiere para poder obtener resultados significativos para el estudio.

P5: De forma similar a `diff` y `cmp`, `comm` estará disponible normalmente en cualquier instalación convencional de Linux a través del paquete `coreutils`, que la implementa junto con un grupo generoso de otras herramientas.

P6: Por lo explicado ya anteriormente, `comm` no ofrece un resultado demasiado confiable porque depende de circunstancias que para el caso del trabajo con HTML pueden resultar en valores ligeramente desviados de la realidad. Si bien se entrega directamente el número de líneas modificadas, cosa que diff no hace, se requeriría un proceso previo de “normalización” de los archivos, que los ponga en un estado que minimice la detección de cambios que en HTML no son significativos.

3.1.3.5. JSdiff

Ésta también es una implementación de una librería en JS que puede usarse con NodeJS[20], específica para el trabajo con la comparación de cadenas de caracteres y dispone de métodos para detectar cambios a nivel de líneas, de palabras y de caracteres. Promete ser muy versátil pero requiere de un trabajo mucho más profundo de programación pues no se dispone de una API como sí la tiene Pretty-Diff, y ésto podría significar una inversión riesgosa pues no se pueden garantizar resultados confiables en poco tiempo. Para las pruebas se desarrolló un script JS llamado `args.js` que permite invocarse con dos argumentos que contienen el código

HTML que sería comparado. El script está basado en el ejemplo provisto por la propia documentación de JSdiff[22].

P1: Luego de ciertas pruebas preliminares con archivos más pequeños, y más allá de toda expectativa previa, el tiempo de ejecución alcanzó el valor extremo de 3 horas y 27 minutos, sólo para terminar en fallo de falta de memoria:

```
[pbernal@tesis NodeJS]$ time node args.js ~/sitioV1.html ~/sitioV2.html
FATAL ERROR: JS Allocation failed - process out of memory
Abortado

real    207m1.409s
user    89m55.326s
sys     0m52.827s
```

Es importante notar que el script args.js funcionó sin éstos problemas con los archivos más pequeños.

P2: El rendimiento tampoco fue un punto a favor, manteniéndose con la carga promedio por encima de 1 durante todo el tiempo. Se notaba también constantes consumos de 100% del procesador.

```
top - 21:19:53 up 4 days,  1:14,  2 users,  load average: 1.19, 1.27, 1.31
Tasks:  93 total,   2 running,  91 sleeping,   0 stopped,   0 zombie
Cpu(s): 38.4%us,  3.7%sy,  0.0%ni, 44.6%id, 12.3%wa,  0.0%hi,  0.2%si,  0.8%st
Mem:   1020232k total,   950436k used,    69796k free,      288k buffers
Swap:  1048568k total,   625948k used,   422620k free,     2940k cached
```

Fig 10. Carga promedio y consumo de recursos durante la ejecución con JSdiff.

Bajo éstas condiciones, definitivamente era impensable intentar una prueba iterativa.

P3: Por supuesto tampoco se pudo obtener ningún resultado ni abstracto ni numérico.

P4: El trabajo de adopción del conocimiento básico necesario para empezar a usar `JSdiff` es alto pues requiere de conocimientos más o menos sólidos en programación y especialmente JS, aunque con estas premisas, el desarrollo realmente se logró en menos de una hora.

P5: Al igual que Pretty-Diff, al tratarse de una herramienta basada en JavaScript, no está disponible en formato RPM. La realización de un RPM si bien se estima de complejidad baja, no se percibe muy necesaria pues basta con descargar la librería y empezar a usarla con NodeJS, por lo cual se requiere de la instalación previa del paquete `nodejs` y `npm` para manejar las librerías adicionales de NodeJS.

P6: Dado que no se pudieron obtener resultados en un tiempo razonable, los datos no pueden ser evaluados ni como relevantes ni como precisos.

3.2. Captura y análisis de imágenes

A diferencia del ítem 3.1, las herramientas para captura y análisis de imágenes son bastante escasas, aunque se coincide en que la mayoría de ellas son gráficas, es decir tienen una GUI desde la cual se usan y manipulan, lo

cual no es más que un estorbo a la hora de automatizar. De entre las que sí se pueden usar vía CLI para soportar automatización, son menos aún aquellas que cumplen con la condición de ser SL.

3.2.1. Métricas

Al igual que el ítem 3.1.1, para las pruebas y mediciones, el mismo equipo y condiciones has sido usadas para todas las fases de prueba y para las mediciones de rendimiento se ha cuidado que nada más que lo esencial se encuentre corriendo en el mismo, por lo cual se puede partir de una carga promedio de 0.0 en todos los casos.

Dicho esto, de la misma forma se han usado los mismos criterios de comparación del análisis de las herramientas de captura y análisis de código HTML (P1..P6), que igualmente permitan una calificación técnica y lo más imparcial posible y por supuesto se contemplan sólo aquellas que ya han cumplido con los criterios generales establecidos al inicio del capítulo.

3.2.2. Captura de imágenes

Dado que el objetivo primario del estudio no es realmente la captura en sí de una captura/vista de la página web en cuestión, sino más bien la detección de los cambios en un determinado período de tiempo, se realizó una selección simplificada de las herramientas de captura que no requieren de un servidor X (GUI) funcionando pues ésta dependencia provoca que la herramienta capture sólo el área visible o un área previamente determinada de la “pantalla”, lo cual no es lo más deseable para el estudio. El trabajo de investigación previo pasó por el análisis de múltiples herramientas:

- ImageMagick,
- wkhtmltoimage,
- CutyCapt,
- khtml2png.
- PyWebShot,
- python-webkit2png y
- PanthomJS.

De todos ellos el único que no requiere de un servidor X, no necesita que se le indique las proporciones/tamaño de la captura y no termina en fallos inexplicables con ciertos sitios fue PhantomJS[23] que además demostró ser un proyecto con mucha actividad y actualidad.

Su uso es bastante simple pues viene ya precompilado y listo para usarse. Basta con descompactar el paquete apropiado (32 ó 64 bits) y colocarlo en alguna de las rutas del *path*. También dispone de la documentación necesaria para compilar y construir un paquete RPM, aunque preferí modificarlas para generar un verdadero paquete RPM de fuente (.src.rpm) y en base a él compilar los RPM para las diferentes versiones y arquitecturas de CentOS, mismos que están disponibles en el repositorio Anku de EcuaLinux[24].

Una vez instalado es relativamente sencillo de usar, necesitando sólo un archivo JS que contenga el código que se requiere ejecutar. De la propia documentación oficial se extrajo el código:

```
var page = require('webpage').create();
```

```
page.open('http://github.com/', function() {
  page.render('github.png');
  phantom.exit();
});
```

Basta con reemplazar el nombre del sitio que se desea capturar y mandar a correr con `phantomjs`:

```
[pbernal@tesis ~]$ time phantomjs ~/Tesis/capture.js

real    0m4.540s
user    0m0.900s
sys     0m1.060s
```

El comando tomó unos 4 segundos y medio y generó un archivo con el nombre especificado en el script JS:

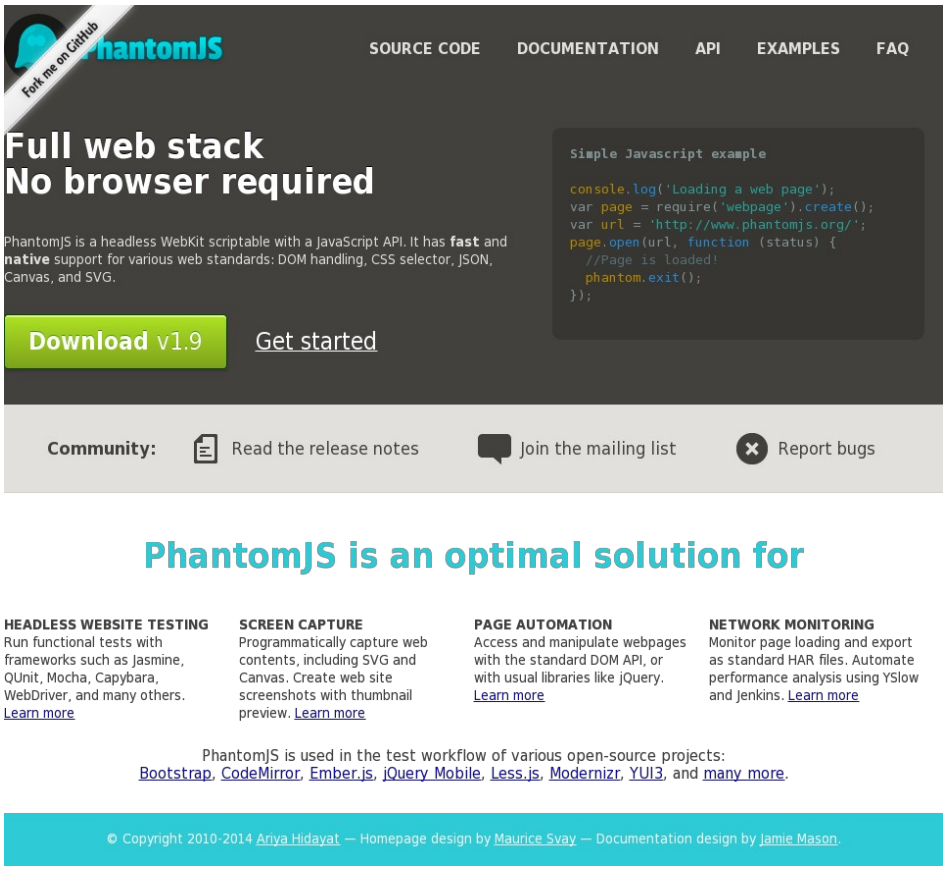


Fig 11. Imagen resultado de la captura del sitio de PhantomJS.

3.2.3. Análisis de cambios en imágenes

3.2.3.1. Compare.py

Esta es una solución bastante directa al problema de detectar cambios en una imagen. Se trata de un script[25] sorprendentemente sencillo, desarrollado en *Python* y que provee un método rápido y confiable de comparar

dos imágenes haciendo uso de las librerías que para estos efectos dispone Python. Primero las convierte a escala de grises para evitar lidiar con los colores y obtener valores más absolutos de los cambios.

Requiere obviamente de tener instalado python, algo que es bastante común y casi esencial en la mayoría de distribuciones Linux. Requiere también de los paquetes: `scipy` y `python-imaging` para funcionar, algo que hubo que determinar en base un proceso de prueba-error, pues la documentación se limita a las dos o tres líneas iniciales del propio script que más que nada sugieren la forma de uso, nada más que eso.

Otro problema fue un error de "ZeroDivisionError: float division":

```
[pbernal@tesis Tesis]$ ./compare.py TT/sitioV1.png TT/sitioV2.png
Traceback (most recent call last):
  File "./compare.py", line 47, in <module>
    main()
  File "./compare.py", line 19, in main
    n_m, n_0 = compare_images(img1, img2)
  File "./compare.py", line 30, in compare_images
    z_norm = norm(diff.ravel(), 0)  # Zero norm
  File "/usr/lib64/python2.6/site-packages/scipy/linalg/basic.py", line 441, in norm
    return numpy.sum(abs(x)**ord,axis=0)**(1.0/ord)
ZeroDivisionError: float division
```

Que solucioné revisando la documentación de la función `norm` de la librería `numpy`, de donde decidí retirar el segundo parámetro que era claramente el que daba el error. El archivo de parche compare.py.path fue generado como referencia. Una vez solventados todos estos inconvenientes menores, el funcionamiento es bastante simple.

P1. El tiempo de ejecución es bastante bueno, con 8 décimas de segundo:

```
[pbernal@tesis Tesis]$ time ./compare.py TT/sitioV1.png TT/sitioV2.png
Manhattan norm: 3626859.33333 / per pixel: 3.69702894267
Zero norm: 22873.5112642 / per pixel: 0.0233160498912

real    0m0.811s
user    0m0.572s
sys     0m0.238s
```

P2. Al correr iterativamente el mismo comando cien veces, suprimiendo la salida como en los casos anteriores de análisis de cambios al código, se obtuvo un tiempo total de ejecución de un minuto 21 segundos, lo cual arroja un promedio de 8 décimas, corroborando lo medido en el parámetro anterior:

```
[pbernal@tesis Tesis]$ time for i in {1..100};do ./compare.py TT/sitioV1.png TT/sitioV2.png
> /dev/null; done

real    1m20.999s
user    0m57.880s
sys     0m23.078s
```

Durante ese tiempo, progresivamente la carga promedio alcanzó un valor de 0.73, bastante razonable considerando que se trata de una comparativa entre imágenes:

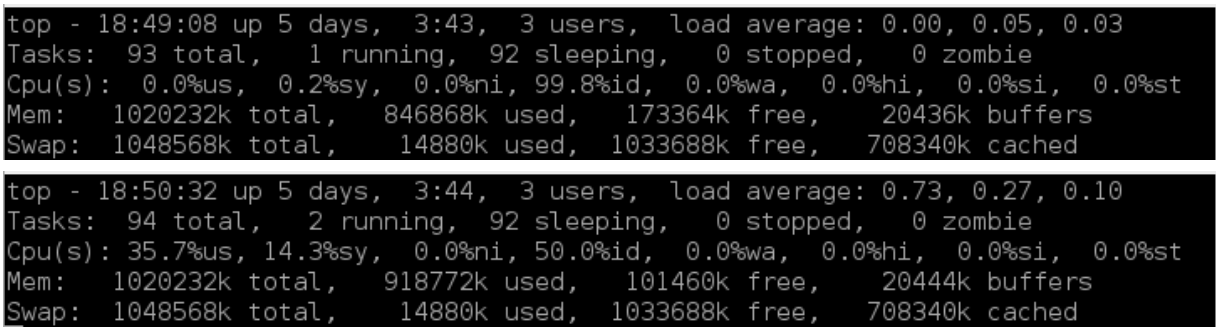


Fig 12. Carga promedio y consumo de recursos durante la ejecución con compare.py.

P3. Un punto a favor es que su resultado es justamente un valor numérico de cambio que se obtiene de relacionar el valor calculado de cambios sobre el total de la imagen (líneas 20 y 21 del script). Claro que en cambio no es capaz de producir un resultado gráfico de las diferencias, lo cual por supuesto lo penaliza pues requiere de una herramienta complementaria para ésto.

P4. Si bien el script viene listo y es muy sencillo de usar, sí requirió de una buena dosis de conocimiento para ponerlo en marcha y hacerlo funcionar. Sólo un administrador con conocimientos de programación puede solucionar los inconvenientes en un tiempo razonable.

P5. Si bien la herramienta es un simple script que no requiere instalarse, sí requiere de paquetes adicionales para correr, mismos que sí están disponibles directamente en las distribuciones más populares de Linux.

P6. En cuanto a la relevancia y precisión del resultado del cálculo, también esta herramienta se lleva un buen puntaje pues realiza una renderización matemática de las imágenes y las diferencia de ésta manera. Sin embargo sólo dispone de dos métodos de cálculo, en base a la normalización Manhathan[26] y Zero[27], de las cuales la segunda fue alterada por lo explicado anteriormente.

3.2.3.2. Perceptual Diff

Esta es una herramienta muy interesante para los propósitos del proyecto pues realiza una comparación de las imágenes con un enfoque en la visión humana, es decir crea un modelo de la imagen que busca simular la visión humana y desecha ciertas diferencias sutiles o poco significativas[28], cambios que pueden darse por pequeñas desviaciones en los cálculos computacionales de *renderización* o por las diferentes metodologías de cálculo de diferentes arquitecturas de hardware.

Su principal aplicación, según el sitio oficial[28], es la comparación sistemática de imágenes generadas en procesos industriales, así que promete ser muy eficiente y veloz, además de ofrecer la minimización significativa de falsos positivos en las comparaciones, basando en el hecho de desechar ciertas diferencias no significativas.

Se ofrecen descargas binarias sólo para sistemas Windows, aunque se ofrece también una descarga en formato comprimido que contiene el binario precompilado y que funciona perfectamente en diversas plataformas tipo

Unix sin mayores inconvenientes. Por esta misma característica se decidió compilar un RPM que está disponible en el repositorio Anku de EcuaLinux[24], de donde se puede instalar y usar fácilmente.

P1. El tiempo de ejecución es un poco superior al de `compare.py`, llegando a poco menos de 11 segundos, lo cual es comprensible dado que éste realiza una comparativa más compleja:

```
[pbernal@tesis Tesis]$ time perceptualdiff TT/sitioV1.png TT/sitioV2.png
FAIL: Images are visibly different
31059 pixels are different
real    0m10.721s
user    0m10.471s
sys     0m0.186s
```

A diferencia de `compare.py`, `perceptualdiff` es capaz de generar una imagen con las diferencias detectadas entre ambas imágenes:

```
[pbernal@tesis Tesis]$ time perceptualdiff TT/sitioV1.png TT/sitioV2.png -output
TTsitioDiffV1V2.png
FAIL: Images are visibly different
31059 pixels are different
real    0m10.764s
user    0m10.554s
sys     0m0.206s
```

La diferencia en el tiempo de ejecución es mínima, pues realiza no sólo el cálculo, sino la generación de la imagen con los cambios también, donde claramente se puede apreciar qué ha cambiado entre la V1 y la V2 de una captura de un mismo sitio:



Fig 13. De derecha a izquierda, imagen V1, imagen V2 y la imagen compuesta de las diferencias entre ambas.

P2. Al ejecutar iterativamente el mismo comando, esta vez sólo 10 veces considerando el tiempo de una sola corrida, obtuvimos un tiempo total de 1 minuto con 49 segundos, para un promedio de casi 10.9 segundos por corrida:

```
[pbernal@tesis Tesis]$ time for i in {1..10};do perceptualdiff TT/sitioV1.png TT/sitioV2.png
>/dev/null; done
real    1m48.926s
user    1m46.866s
```

```
sys     0m2.013s
```

Durante este tiempo la carga promedio llegó a 0.9, nuevamente bastante razonable considerando el trabajo que se está realizando:

```
top - 13:58:47 up 5 days, 22:52,  3 users,  load average: 0.00, 0.00, 0.00
Tasks:  93 total,   1 running,  92 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.2%sy,  0.0%ni, 99.8%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020232k total,   713740k used,   306492k free,    33304k buffers
Swap:  1048568k total,    14836k used,  1033732k free,   562564k cached
```

```
top - 14:01:29 up 5 days, 22:55,  3 users,  load average: 0.90, 0.39, 0.14
Tasks:  94 total,   2 running,  92 sleeping,   0 stopped,   0 zombie
Cpu(s): 49.9%us,  0.2%sy,  0.0%ni, 49.9%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020232k total,   832852k used,   187380k free,    33416k buffers
Swap:  1048568k total,    14836k used,  1033732k free,   562564k cached
```

Fig 14. Carga promedio y consumo de recursos durante la ejecución con perceptualdiff.

P3. Si bien la herramienta genera como salida el número total de píxeles que han cambiado, se requiere de realizar un cálculo adicional sencillo, que lo relacione con el total de píxeles de la imagen, para así obtener un valor usable. Adicionalmente hay que resaltar nuevamente que el resultado gráfico que genera es más que elocuente, con un impacto casi despreciable en el rendimiento.

P4. A pesar del trabajo complejo que realiza, es una herramienta que, gracias a haberla puesto ya en RPM, se puede empezar a usar en minutos, aunque sí requerirá algo más de trabajo para realizar el cálculo de cálculo total.

P5. Aún cuando, como ya se dijo, se compiló un RPM para facilitar el proceso de estudio y pruebas, y el proceso fue relativamente sencillo, se le penalizó un poco por este trabajo extra que hubo que realizar y que hay que seguir haciendo para mantener el RPM actualizado.

P6. Los resultados son muy reales precisos, aunque hay que prepararlos para una interpretación humana de los mismos pues el número de píxeles cambiados no dice mucho, aunque la imagen obtenida habla más que mil datos, éstos son necesarios para la automatización.

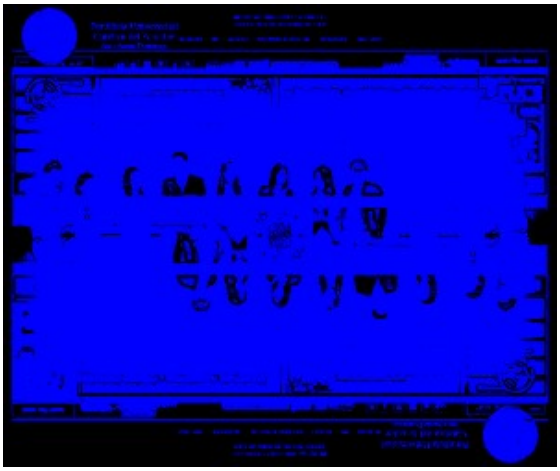


Fig 15. Imagen resultante de comparar dos imágenes intencionalmente diferentes (volteada H/V) con PDiff.

3.2.3.3. ImageMagick

Se trata de una herramienta ampliamente usada desde hace mucho tiempo en sistemas Linux y otras plataformas para todo tipo de tareas relacionadas a imágenes. Se trata de un proyecto[29] muy activamente mantenido, soportado y popular. Provee, por ejemplo, un modo muy sencillo de convertir varias imágenes en un solo archivo PDF o extraer las imágenes de un PDF, recortarlas, cambiarles la resolución, etc.

Pero también ofrece los medios para realizar comparativas entre imágenes y en este sentido dispone de todo un arsenal de opciones/parámetros que pueden ser usados según las necesidades específicas y para el estudio, por supuesto una cierta combinación de unos pocos de ellos[29] entregan toda la información necesaria.

Otra gran característica de ImageMagick (IM) es que, aún siendo una herramienta orientada a CLI, dispone de interfaces para lenguajes como C, C++, Java, .NET, Pascal, Perl, PHP, Python, Ruby, entre los más conocidos, que permite una integración aún más rápida en casi cualquier aplicativo.

También es bueno el hecho de que está disponible para todas las distribuciones más populares de Linux y se puede instalar y empezar a usar en pocos minutos.

P1. Se ha medido la ejecución de IM en 2 casos: Efectuando únicamente el cálculo numérico de la diferencia, con lo cual se obtuvo un impresionante tiempo de dos y media décimas de segundo:

```
[pbernal@tesis Tesis]$ time compare -metric AE TT/sitioV1.png TT/sitioV2.png null: 2>&1
35480

real    0m0.253s
user    0m0.316s
sys     0m0.060s
```

Si se reemplaza el `null: 2>&1` que suprime la salida, para obtener la imagen resultado de la comparativa, el tiempo se incrementa a las casi 5 décimas y media, lo cual no deja de ser bueno pues está realizando las dos tareas a la vez:

```
[pbernal@tesis Tesis]$ time compare -metric AE TT/sitioV1.png TT/sitioV2.png sitioV1V2IM.png
35480

real    0m0.542s
user    0m0.725s
sys     0m0.041s
```

En este caso, la imagen generada tiene también una ventaja visualmente y es que se genera con una “sombra” de la imagen original con la cual se contrasta mucho mejor los cambios resaltados en rojo, aunque en la documentación indica que es un comportamiento que se puede variar de ser requerido:



Fig 16. Imagen resultado de la comparación con ImageMagick.

P2. Se envió la corrida de 100 iteraciones, que tomaron un tiempo total de 54 segundos y medio, confirmando el tiempo de 5 décimas y media por corrida, calculado anteriormente :

```
[pbernal@tesis Tesis]$ time for i in {1..100};do compare -metric AE TT/sitioV1.png
TT/sitioV2.png sitioV1V2IM.png > /dev/null; done

real    0m54.653s
user    1m10.840s
sys     0m5.744s
```

Estas corridas se mandaron con la operación de generar la imagen resultante con el fin de revisar qué tanto impacto le causa al rendimiento del equipo en el escenario de más trabajo. La carga promedio alcanzó un valor de 0.77:

```
top - 16:52:54 up 6 days,  1:47,  3 users,  load average: 0.00, 0.00, 0.00
Tasks:  93 total,   1 running,  92 sleeping,   0 stopped,   0 zombie
Cpu(s):  0.0%us,  0.0%sy,  0.0%ni,100.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   1020232k total,   757432k used,   262800k free,    36584k buffers
Swap:  1048568k total,    14832k used,  1033736k free,   601376k cached

top - 16:53:51 up 6 days,  1:48,  3 users,  load average: 0.77, 0.21, 0.07
Tasks:  94 total,   2 running,  92 sleeping,   0 stopped,   0 zombie
Cpu(s): 62.7%us,  4.8%sy,  0.0%ni, 29.5%id,  0.0%wa,  0.0%hi,  0.0%si,  3.0%st
Mem:   1020232k total,   782060k used,   238172k free,    36664k buffers
Swap:  1048568k total,    14832k used,  1033736k free,   601228k cached
```

Fig 17. Carga promedio y consumo de recursos durante la ejecución con ImageMagick.

P3. Los resultados son elementalmente iguales a los de perceptualdiff, así que el análisis es muy similar y por ende la valoración.

P4. La instalación es muy sencilla, aunque empezar a obtener los resultados puede costar un poco de tiempo y esfuerzo en leer y probar la documentación, encontrar las opciones/parámetros apropiados para las necesidades del proyecto, otro tanto.

P5. Dado que es una herramienta disponible y activamente mantenida oficialmente por todas las distribuciones populares de Linux, obtiene el mejor puntaje.

P6. Al tratarse de una herramienta tan ampliamente usada en todo tipo de ámbitos, la confiabilidad de los resultados está acorde a las necesidades y por supuesto la combinación de un valor numérico y una imagen resultante, en un solo paso es más que relevante, especialmente si se comparan dos imágenes intencionalmente diferentes (una volteada horizontal y verticalmente respecto a la otra):



Fig 18. Imagen resultante de comparar 2 imágenes intencionalmente diferentes (volteada H/V) con IM.

3.2.3.4. GraphicsMagick

Esta herramienta es un *forking* de IM y que promete superarlo en cuanto a rendimiento y consumo de recursos. Dado que se trata de una derivación de la anterior, se aplican casi los mismos criterios, aunque ésta no goza de la aceptación y popularidad de la anterior, pero ofrece una instalación mucho más pequeña y es la usada en sitios tan populares como el propio Flickr[30].

Está diseñado para convivir con IM y cualquier otro software de procesamiento gráfico y se destaca por la posibilidad de trabajar con imágenes en el orden de los gigabytes e igualmente soporta interfaces para lenguajes como: C, C++, Perl, PHP, Python, .NET y Ruby.

Está disponible para ciertas distribuciones de Linux, especialmente Debian/Ubuntu y también para la línea de RedHat como CentOS, pero a través del repositorio de EPEL[21]. Dado que los comandos son los mismos que los de IM, GraphicsMagick (GM) se usa a través de un *wrapper* llamado `gm`. Se ha buscado ejecutar los mismos comandos que con IM con el fin de obtener una comparativa más real.

P1. La ejecución de GM para calcular la diferencia neta en píxeles tomó sólo poco más de una décima de segundo:

```
[pbernal@tesis Tesis]$ time gm compare -metric mae TT/sitioV1.png TT/sitioV2.png
Image Difference (MeanAbsoluteError):
           Normalized    Absolute
```

```
          ============  ==========
     Red: 0.0205773383        5.2
   Green: 0.0174790424        4.5
    Blue: 0.0104079028        2.7
 Opacity: 0.0000000000        0.0
   Total: 0.0121160709        3.1
real    0m0.117s
user    0m0.110s
sys     0m0.021s
```

Al realizar la prueba con la generación de imagen resultante, se obtuvo un tiempo de poco menos de cinco décimas de segundo confirmando con poco la oferta de mejorar en rendimiento a IM.

```
[pbernal@tesis Tesis]$ time gm compare -metric mae TT/sitioV1.png TT/sitioV2.png -file
sitioV1V2GM.png
Image Difference (MeanAbsoluteError):
           Normalized    Absolute
          ============  ==========
     Red: 0.0205773383        5.2
   Green: 0.0174790424        4.5
    Blue: 0.0104079028        2.7
 Opacity: 0.0000000000        0.0
   Total: 0.0121160709        3.1
real    0m0.455s
user    0m0.548s
sys     0m0.029s
```

A diferencia de IM, la imagen obtenida es la misma imagen original con las secciones diferentes en un color entre morado y marrón, visualmente menos elocuente que IM:



Fig 19. Imagen resultado de la comparación con GraphicsMagick.

P2. Al ejecutar las 100 corridas del mismo comando generando la imagen, obtuvimos un tiempo total de algo menos de 45 segundos y medio, con tiempo promedio por corrida de 4 décimas y media, confirmando las mediciones anteriores:

```
[pbernal@tesis Tesis]$ time for i in {1..100};do gm compare -metric mae TT/sitioV1.png
TT/sitioV2.png -file sitioV1V2GM.png > /dev/null; done

real    0m45.436s
user    0m53.540s
sys     0m2.683s
```

La carga promedio del equipo alcanzó 0.94 durante este tiempo, contradiciendo un poco el ofrecimiento de mejor rendimiento respecto a IM:

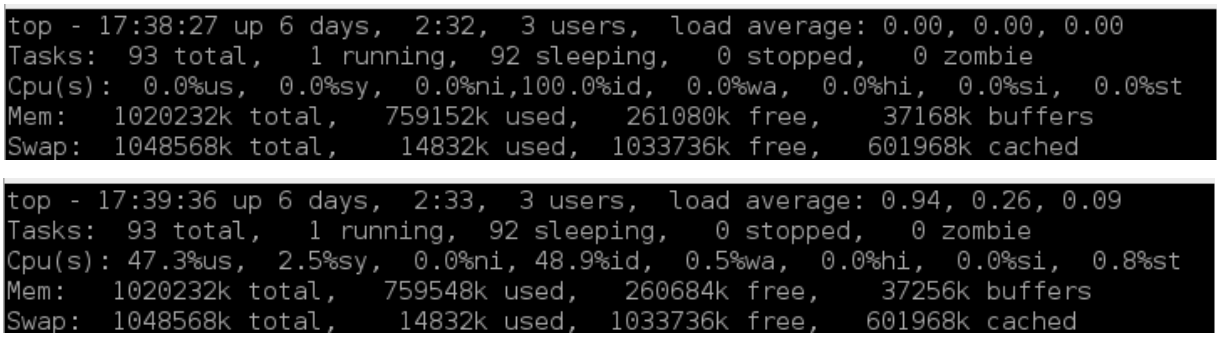


Fig 20. Carga promedio y consumo de recursos durante la ejecución con GraphicsMagick

P3. Al igual que IM entrega valores numéricos, pero de una manera menos versátil aunque más directa, es decir el valor de absoluto del total del cálculo de la desviación, representa ya un porcentaje de cambio, sin necesidad de cálculos adicionales.

P4. En éste caso también se le otorga la misma valoración que IM por ser 2 herramientas muy similares y con elementalmente las mismas condiciones de aprendizaje y adaptación para su uso.

P5. Dado que está disponible directamente en RPM como ya se indicó, obtiene el mejor puntaje en este rubro.

P6. En cuanto a la precisión de la información proporcionada, dado que no soporta la métrica AE[31] (Absolute Error) con la cual obtuvimos los resultados más precisos en IM, sólo se puede calcular la MAE[31] (Mean Absolute Error) o media del error absoluto, para imágenes que intencionalmente son muy diferentes (una volteada horizontal y verticalmente respecto a la original), el valor sólo se acercó al 47%. Visualmente también penaliza porque éste mismo resultado sólo muestra una especie de velo sobre la imagen original, algo que realmente requerirá de un ojo un poco más entrenado para ver que se trata de una imagen muy diferente:



Fig 21. Imagen resultante de comparar dos imágenes intencionalmente diferentes (volteada H/V) con GM.

3.3. Automatización de procesos

Con las herramientas seleccionadas para la captura de código HTML de los sitios, así como para captura imagenográfica de los mismos, es preciso crear scritps de automatización que permitan una captura diaria y sistemática para que puedan ser procesados posteriormente con el sistema a realizarse con las herramientas de análisis de cambios seleccionadas en el punto 3.4.

Por ser lo más complejo, lo primero fue crear un script que haga de *wrapper* para PhantomJS realice la captura de la imagen del sitio. Esto es necesario porque como ya se dijo, se requiere de pasarle un JS a phantomjs con el código que le indica qué sitio capturar y con qué nombre grabarlo. Este script se denomina captureSite.sh. y se colocó en la carpeta `bin/` del directorio principal del usuario para que quede directamente en el *Path* y pueda ser directamente usado:

```
[pbernal@tesis ~]$ captureSite.sh
Uso: /home/pbernal/bin/captureSite.sh <URL>
Ejemplo: /home/pbernal/bin/captureSite.sh www.ecualinux.com
```

El siguiente paso es más sencillo y sólo requiere de invocar a `wget` y `captureSite.sh` para cada sitio en una predefinida con las URL de los sitios que se desea capturar. El script se denomina getSites.sh e igualmente se colocó en `bin/` para que se pueda invocar desde cualquier lugar:

```
[pbernal@tesis ~]$ getSites.sh
Downloading... www.pucesd.edu.ec
WARNING: combining -O with -r or -p will mean that all downloaded content
will be placed in the single file you specified.

--2014-08-24 19:27:27--  http://www.pucesd.edu.ec/
Resolviendo www.pucesd.edu.ec... 190.15.137.130
Connecting to www.pucesd.edu.ec|190.15.137.130|:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: no especificado [text/html]
Saving to: `www.pucesd.edu.ec.out'
```

```
    [ <=>                                       ] 24.364      --.-K/s   in 0,1s

2014-08-24 19:27:28 (168 KB/s) - `www.pucesd.edu.ec.out' saved [24364]

Cargando robots.txt; por favor ignore los errores.
--2014-08-24 19:27:28--  http://www.pucesd.edu.ec/robots.txt
Connecting to www.pucesd.edu.ec|190.15.137.130|:80... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 849 [text/plain]
Saving to: `www.pucesd.edu.ec.out'

100%[==============================================>] 849         --.-K/s   in 0s
…
…
[ SALIDA DEL COMANDO SUPRIMIDA]
…
```

Finalmente ya con toda la infraestructura lista, sólo queda programar la tarea que invoque diariamente el script, lo cual se realiza a través de *cron*, con la siguiente configuración agregada usando `crontab -e`:

```
5 2 * * *       getSites.sh &> /dev/null
```

Según la cual se ejecutará el script `getSites.sh` diariamente a las 02H05, de esta forma no interferimos mayormente con otras tareas que pudieran estarse realizando y que pudieran además requerir del ancho de banda que el script usará.

3.4. Recopilación de información de análisis

Para finalizar el capítulo, y luego del estudio realizado, se han tabulado los datos para poder determinar de forma objetiva las herramientas más idóneas a ser usadas en el prototipo del sistema de detección automatizada de desfiguraciones de sitios.

3.4.1. Resultados de análisis de cambios HTML

Según el análisis realizado en el punto 3.1.3 las valoraciones de cada parámetro se detallan en la tabla 1:

Herramienta	P1(/5)	P2(/5)	P3(/3)	P4(/3)	P5(/2)	P6(/5)	Total
diff	5 (0.005s)	5 (0.02)	1	2	2	2	**17**
Pretty-Diff	2 (2.154s)	2 (0.96)	3	1	1	5	14
Cmp	3 (0.144s)	3 (0.07)	1	3	2	4	16
Comm	4 (0.008s)	4 (0.03)	2	2	2	3	**17**
NodeJS-diff	1 (207m1.409s)	1 (1.19)	--	1	1	--	4

Tabla 1. Comparativa de resultados de anális de cambios HTML.

De lo cual obtuvimos un empate en el valor absoluto total obtenido, pero considerando el parámetro 6 de evaluación, el prototipo se realizará con **`comm`** pues ofrece un valor mejor en la relevancia/exactitud de la información proporcionada.

3.4.2. Resultados de análisis de cambios en imágenes

Según el análisis realizado en el punto 3.2.3 las valoraciones de cada parámetro se detallan en la tabla 2:

Herramienta	P1(/5)	P2(/5)	P3(/3)	P4(/3)	P5(/2)	P6(/5)	Total
compare.py	3 (0.811s)	5 (0.73)	2	1	2	3	16
Perceptualdiff	2 (10.764s)	3 (0.90)	3	3	1	5	17
ImageMagick	4 (0.524s)	4 (0.77)	3	2	2	5	**20**
GraphicsMagick	5 (0.455s)	2 (0.94)	3	2	2	3	17

Tabla 2. Comparativa de resultados de anális de cambios en imágenes.

De lo cual obtuvimos un claro ganador que es ImageMagick para la tarea de comparar y obtener valores de diferenciación entre las imágenes capturadas diariamente con el proceso de automatización.

Capítulo 4 :: CONSTRUCCIÓN SISTEMA PROTOTIPO

Con la información recabada y las herramientas seleccionadas, se ha construido un prototipo de sistema de detección automatizada de desfiguraciones a sitios web. Es sistema hace uso de la información que se almacenó sistemáticamente cada día y que fue almacenada por el proceso de automatización descrito en 3.3.

Se comenzó con crear todos los sub sistemas de CLI, tanto para análisis de código HTML, como para análisis imagenográfico de las capturas de los sitios.

4.1. Construcción del sistema *CLI* de análisis de diferencias en código HTML

Este es quizá el sistema más simple de todos, pues se reduce a la preparación de un sistema que haga uso de los resultados provistos por `comm` y los transporte a un valor numérico que pueda ser evaluado pues sólo se obtienen las líneas que son comunes en ambos archivos.

El script usa `wc` para calcular el número total de líneas del archivo original y usa `comm` para calcular las líneas que son comunes en ambos archivos. Finalmente relaciona estos valores para obtener una valor porcentual de cambio. El resultado final será usado por el sistema principal para emitir alertas si se sobrepasa un cierto valor. El script terminado getCodeChanges.sh es bastante simple en realidad y su uso es igualmente muy sencillo, aún así incluye la opción de presentar información de uso si no se le proporciona ningún argumento:

```
[pbernal@tesis ~]$ getCodeChanges.sh
Uso: getCodeChanges.sh <file1.html> <file2.html> [-i]
Ejemplo: getCodeChanges.sh sitioV1.html sitioV2.html (Floating point results)
         getCodeChanges.sh sitioV1.html sitioV2.html -i (Integer results)
```

Al correrlo ya con los argumentos apropiados, ofrece el valor calculado de cambio de 24.949(%):

```
[pbernal@tesis ~]$ getCodeChanges.sh sitioV1.html sitioV2.html
24.9491
```

4.2. Construcción del sistema *CLI* de análisis de diferencias en imágen

Este sub sistema resultó ligeramente más complejo dadas las complicaciones naturales asociadas la trabajo con un comando un poco más complejo y el hecho de que se trata de imágenes y no de texto simple. La filosofía del script resultante getImageChanges.sh es elementalmente la misma que la del punto anterior, pero realiza unos pocos pasos adicionales/diferentes para calcular un nombre apropiado para el archivo de imagen resultante, dado que en este caso el resultado del mismo no es sólo el valor porcentual de cambio, sino también la posibilidad de generar la imagen que los muestra visualmente. Al igual que `getCodeChanges.sh` también se provee ayuda sobre el uso si se ejecuta sin parámetros:

```
[pbernal@tesis ~]$ getImageChanges.sh
```

```
Uso: getImageChanges.sh <file1.png> <file2.png> [-i]
Ejemplo: getImageChanges.sh sitioV1.png sitioV2.png (Just gets diff value)
         getImageChanges.sh sitioV1.png sitioV2.png -i (Generates diff image)
```

También se incluyó la lógica necesaria para recortar una de las dos imágenes de ser necesario. Esto debido a que si el sitio creció verticalmente (algo relativamente común), éste cambio no introduzca valores adicionales de diferenciación entre ellas. Se detecta cuál de los dos es el más grande y se ajusta al tamaño calculado del más pequeño.

Sin embargo su uso es muy sencillo y basta con ejecutarlo con los dos parámetros necesarios, para que emita directamente el valor calculado de cambio visual y de agregar el tercer parámetro -i, genera automáticamente la imagen resultante con el nombre compuesto de los nombres de los dos archivos comparados, `sitioV1-sitioV2.png` en este caso.

```
[pbernal@tesis ~]$ getImageChanges.sh sitioV1.png sitioV2.png -i
16.0389
[pbernal@tesis TT]$ ls -l *.png
-rw-rw-r--. 1 pbernal pbernal  1273346 ago 25 18:42 sitioV1-SitioV2.png
-rw-r--r--. 1 pbernal pbernal  1480519 ago 23 15:09 sitioV1.png
-rw-r--r--. 1 pbernal pbernal  1513298 ago 25 18:41 sitioV2.png
```

4.3. Construcción del sistema *CLI* de emisión de alertas programadas

Para la emisión de alertas programadas, hace falta primero establecer un mecanismo que con cierta frecuencia realice la revisión de cambios a través de los dos sub sistemas anteriores y los almacene en una base de datos (BD). Este almacenamiento en BD no es obligatorio pero sí muy útil, especialmente si se piensa en función de reportes y estadísticas.

Este proceso de revisión se puede realizar fácilmente con un script el Bash, pero no es tan simple la inserción de los datos en una BD, al menos no con Bash. Para esto, se usó PHP y el Yii Framework[32] (YF) que es muy popular y entre otras cosas, permite crear una aplicación web en poco tiempo con la opción de disponer de comandos que se pueden correr por CLI y que estén integrados con la aplicación.

Para iniciar la aplicación, lo primero es el modelo de datos, que para este caso es más bien simple, tratándose de un prototipo, y consta de 3 tablas:



Fig 22. Modelo de datos del sistema

El modelo dbModelRE.sql está diseñado para que se almacenan clientes que tienen uno o múltiples sitios y para cada sitio se almacenan uno o múltiples registros de chequeo. Cada registro de chequeo contiene elementalmente la fecha y los valores de cambio de código e imagen calculados en cada caso.

De esta forma, se realizarán barridos diarios a la tabla Check buscando valores que pudieran haber superado los establecidos en las preferencias del sitio correspondiente. De ser así se enviará una notificación al email registrado del cliente.

Con el modelo creado, el siguiente paso es descargar el paquete del YF (versión 1.1.15 es la versión estable más reciente a la fecha), descomprimirlo en `/var/www` y crear un enlace `yii` para usar un nombre más corto y sencillo:

```
[root@tesis ~]# cd /var/www
[root@tesis www]# wget https://github.com/yiisoft/yii/releases/download/1.1.15/yii-
1.1.15.022a51.tar.gz
[root@tesis www]# tar zxf yii-1.1.15.022a51.tar.gz
[root@tesis www]# ln -sf yii-1.1.15.022a51/framework yii
```

A partir de este momento simplemente se siguió las instrucciones para crear una aplicación yii de consola[33]. Un proceso bastante sencillo y simplificado. De esta forma se desarrolló el comando (como en yii se denominan a estos scripts) denominado CheckSitesCommand.php que es el que contiene toda la lógica para realizar el trabajo.

El script se conecta a la BD y obtiene la lista de sitios activos desde la BD. Para cada sitio en la lista realiza el chequeo de código e imagen y almacena los valores obtenidos en un nuevo registro en la BD. Adicionalmente actualiza el archivo `sites.lst` ya referido anteriormente y que es usado por el sub sistema de capturas.

Al finalizar, el script envía una notificación a todos los contactos de los sitios que hubieren alcanzado o superado los límites establecidos en el campo `preferences` de cada uno de ellos. Para correrlo cada día, basta con colocar la tarea programada en cron como ya se hizo anteriormente para la captura de sitios:

```
5 3 * * *      /var/www/html/protected/yiic CheckSites &> /dev/null
```

En este caso se programó a correr una hora y 5 minutos más tarde de la tarea de captura para darle tiempo a ella de terminar su trabajo. El comando también puede recibir 2 argumentos con fechas anterior-actual para ejecutar un chequeo entre dos fechas, pues por defecto revisa la versión actual y la coteja con la del día anterior, pero en este caso ya no sería una tarea de cron, sino un comando ejecutado bajo demanda:

```
[pbernal@tesis ~]$ /var/www/html/protected/yiic CheckSites 20140824 20140825
Insertado: siteID=1 - url=www.pucesd.edu.ec - codeChange=0 - imgChange=0.721779
Insertado: siteID=2 - url=www.epn.edu.ec - codeChange=4.2906 - imgChange=5.66998
Insertado: siteID=3 - url=www.educacionsuperior.gob.ec - codeChange=.2610 - imgChange=0
Insertado: siteID=4 - url=www.ecuavisa.com - codeChange=35.1039 - imgChange=0.141647
Insertado: siteID=5 - url=www.mercadolibre.com.ec - codeChange=39.6838 - imgChange=3.73796
```

De esta forma se realiza el análisis y almacenamiento respectivo de los registros de la comparativa entre las capturas de los días 24 y 25 de agosto de 2014. La imagen resultante de la comparación se almacena en la carpeta del día menor, del 24 en este caso.

4.4. Construcción del sistema web de configuración y administración

Una vez lista toda la parte de CLI del sistema, se debía construir la parte web que facilite la administración de clientes, sitios y la visualización de los resultados de los chequeos diarios. Se continuó con el mismo YF, pues de hecho es para lo que la enorme mayoría de desarrollos se orienta, la construcción de aplicaciones web. También en este sentido, provee de grandes facilidades y ayuda a minimizar los tiempos de desarrollo, permitiendo que se dedique el esfuerzo a las reglas de negocio y la programación de las mismas, más que a las típicas tareas de levantamiento del aplicativo como tal.

Al seguir el procedimiento simplificado de creación de una aplicación web[34] se puede obtener un prototipo de un sitio web en muy poco tiempo. En este caso, alguna parte del camino ya se recorrió en el punto 4.3 para el sistema de CLI. El procedimiento explica cómo activar el módulo Gii que permite que se genere las interfaces de administración para las tablas del modelo con tan solo llenar unos pocos campos y dar clic en “Generar”, con lo cual se obtiene todo el código necesario básico CRUD. Sólo queda realizar ciertas personalizaciones para lograr que el aplicativo se ajuste mejor a las necesidades particulares.

El sistema fue bautizado con el nombre de **Qhaway**, que en Qhechua significa: mirar, observar, escudriñar. Un término bastante apropiado para lo que el sistema realiza.

Lo primero fue ajustar el módulo de Clientes, en el cual se realizaron en realidad muy pocos cambios, que fueron más bien orientados a la traducción de ciertos elementos de la interface y el ajuste de los elementos del menú de administración y la lógica de navegación entre las diferentes interfaces CRUD del módulo.

Por cierto, el sistema base provisto por YF, además de las interfaces, contiene todo el manejo de usuarios, accesos, sesiones, acceso a BD, paginación, búsquedas, y validación dentro del aplicativo, así que ahorra una enorme cantidad de tiempo en estos aspectos. En el módulo, sólo debo escribir unas pocas líneas para indicar quién tiene o no acceso a cada parte del mismo.

El usuario administrador por defecto tiene las credeciales admin/admin, de dicha forma se puede entrar a administrar clientes:

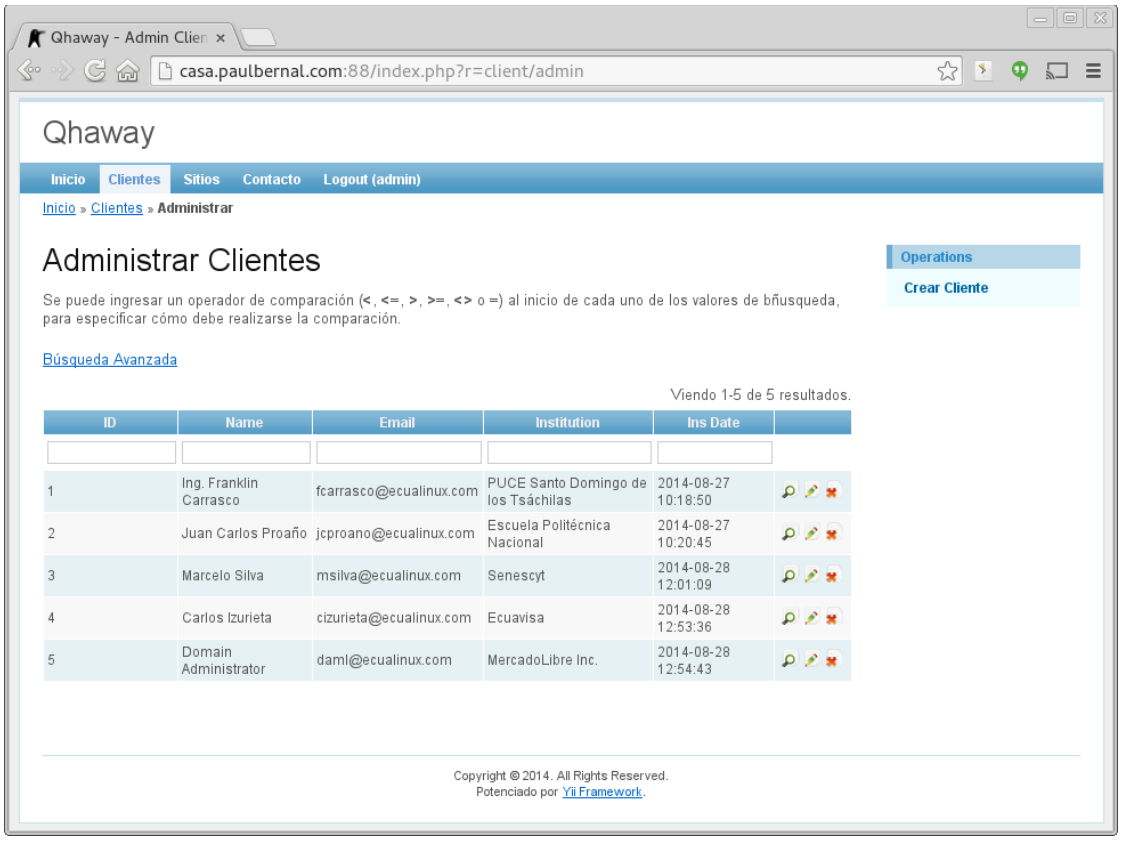


Fig 23. Interface de Administración de Clientes de Qhaway.

El link de la derecha “Crear Cliente” permite agregar nuevos registros al sistema:

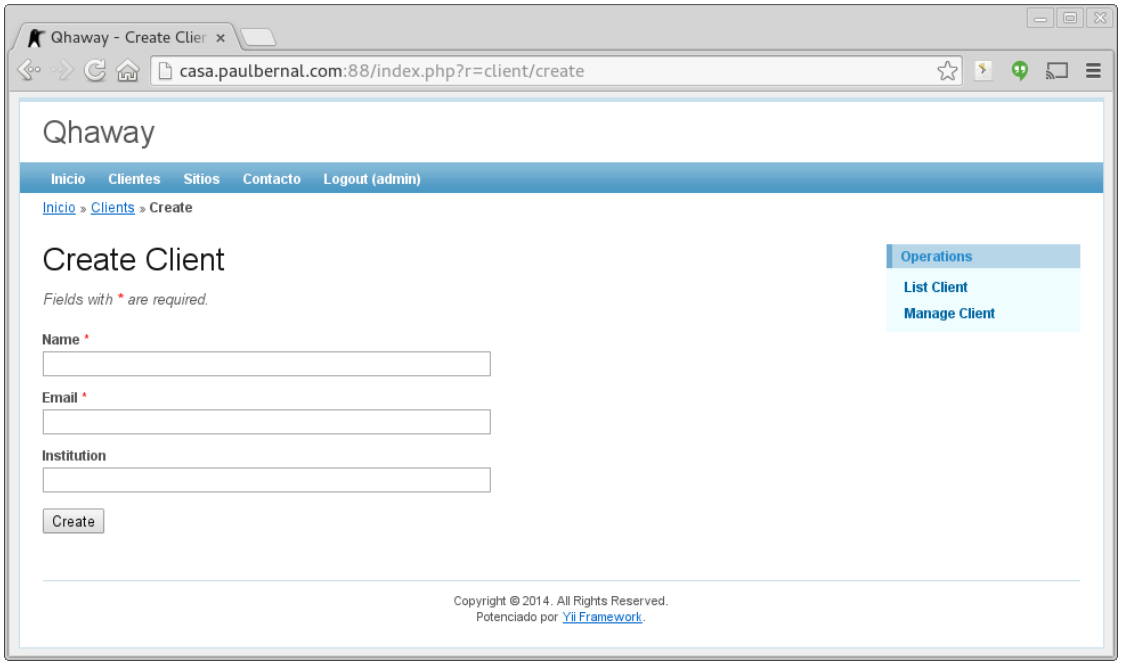


Fig 24. Interface de creación de nuevos clientes de Qhaway.

El funcionamiento básico del módulo de sitios es el mismo, sin embargo, en este caso sí se realizó más personalización porque era necesario visualizar la lista de registro resultado del chequeo diario, debajo del detalle de un sitio:

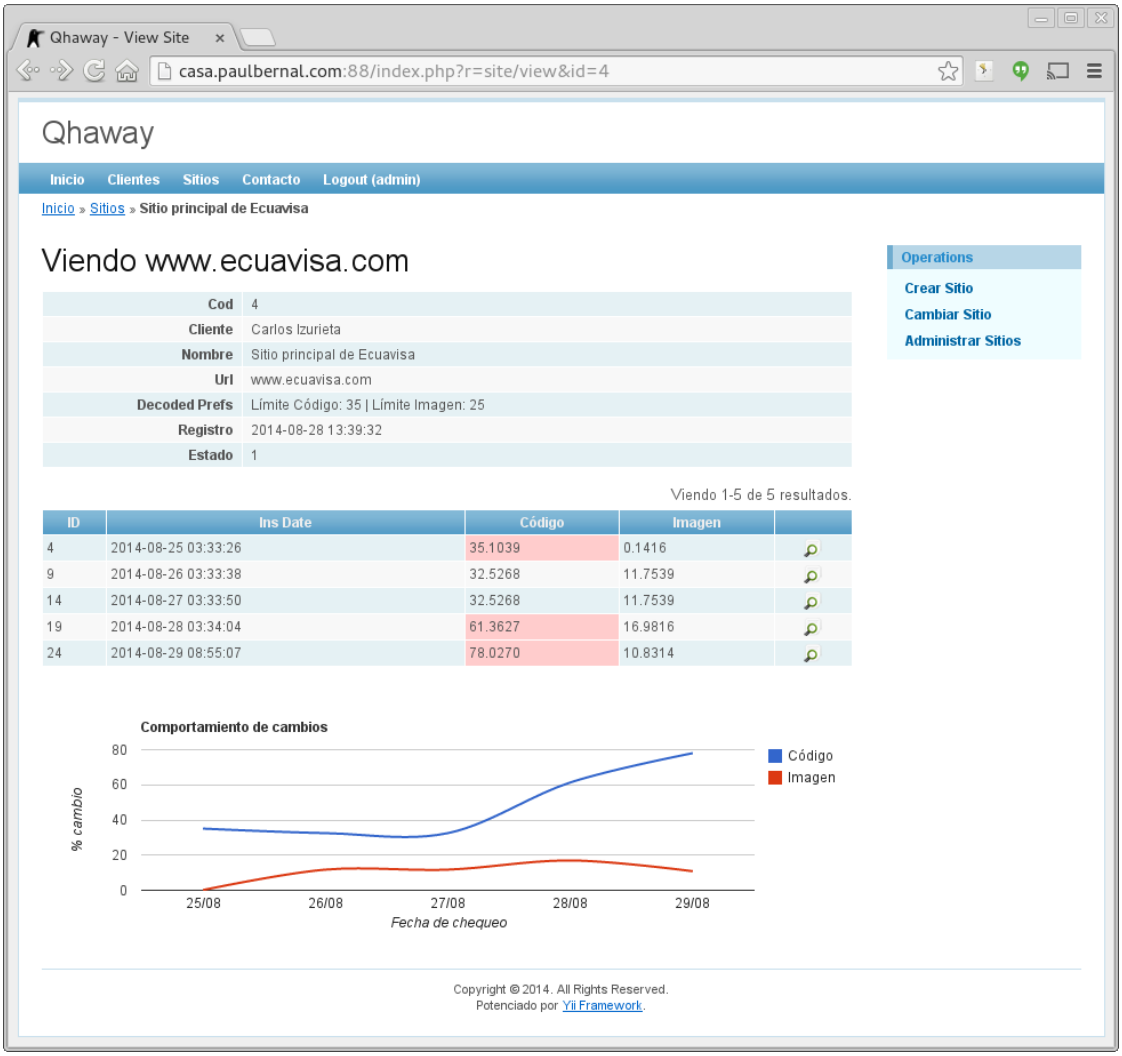


Fig 25. Interface de detalle del sitio y resultados de sus revisiones diarias.

Como se puede observar en la figura 25, además de generar el listado de revisiones, se modificó la lista para que, si el ítem respectivo de cambio en código ó imagen, igualara o superara el límite establecido en las preferencias respectivas, se marque la celda específica con un color de alerta, de tal forma que sea más sencillo visualizar directamente los potenciales problemas de cambios.

También se agregó una extensión para poder incluir un gráfico de Google Chart[35] que da una idea más clara de la tendencia del sitio durante los últimos días, lo cual le agrega un elemento bastante útil para obtener una idea más clara del comportamiento diario del sitio y posibles alteraciones abruptas asociadas..

4.5. Complementación del sistema web con un área informativa

Con toda la parte medular del sistema completada, era necesario agregar ciertos elementos informativos y de contacto al sitio, para darle una orientación más de portal web de servicios, que la de un sistema solamente. En

este sentido, nuevamente YF hizo casi todo el trabajo, dejando sólo el trabajo de personalización del mismo, que se reducen a agregar/modificar los textos de la página principal del sitio, así como agregarle un control Captcha al formulario de Contactos:



Fig 26. Vista de la interface del sistema Web Qhaway.

4.6. Pruebas y ajustes

El sistema fue sometido a constantes pruebas pues desde que se completó el sistema de CLI, está recolectando datos, analizándolos y almacenándolos en la BD. Inicialmente hubo problemas con las variables de entorno que resultó que eran diferentes para las tareas de `cron` que para los scripts desarrollados y que eran ejecutados manualmente desde la consola.

Dado que los scripts fueron colocados en el directorio `bin/` del usuario que los ejecuta, si se corren por consola el PATH contiene también a éste directorio `bin/`, pero cuando se corre a través de un cron, el PATH contiene sólamente `/usr/bin:/bin`, razón por la cual en un inicio no se corrían las capturas de forma automática. La solución (parche) fue la de crear un enlace simbólico en `/usr/bin`, hacia el script en `/home/pbernal/bin`:

```
[root@tesis ~]$ cd /usr/bin
[root@tesis ~]$ ln -s /home/pbernal/bin/getSites.sh
```

La solución definitiva es la de crear un paquete RPM que instale adecuadamente todos los elementos en su lugar, pero es una tarea que había que dejar al final para que pudiera incluir las posibles correcciones a otros problemas que pudieran aparecer antes de terminar todo el sistema.

Otro problema detectado tuvo que ver con la otra tarea programada, el comando de YF que se encarga de realizar el análisis de cambios y notificación a los responsables de ser el caso. El problema recae nuevamente en la misma causa, por lo cual se crearon igualmente los enlaces simbólicos necesarios bajo los mismos principios que en el caso anterior:

```
[root@tesis ~]$ cd /usr/bin
[root@tesis ~]$ ln -s /home/pbernal/bin/getCodeChanges.sh
[root@tesis ~]$ ln -s /home/pbernal/bin/getImageChanges.sh
```

Con esto pudimos obtener las capturas y realizar los chequeos de forma desatendida diariamente.

El sistema web sufrió de muchas dificultades técnicas, relacionadas mayormente con la curva de aprendizaje del YF, pero éstas iban siendo solventadas sobre la marcha y no viene al caso detallarlas pues están más bien directamente relacionadas con asuntos de programación. Afortunadamente, también se encontró una ventaja adicional al YF, que es la disponibilidad de documentación tanto técnica como práctica, pues hay muchos recursos de ayuda de todo tipo y gente usándolo que ya pasó por las mismas dificultades antes y decidió compartir sus soluciones. Por cierto, ésta es una característica común en el SL.

4.7. Análisis de resultados

Como se pudo observar en los ítems anteriores, se arrancó el sistema con un grupo finito de sitios, tratando de cubrir con un representante de cada uno de los sectores principales en el ámbito web nacional:

- Dos sitios educacionales (www.pucesd.edu.ec, www.epn.edu.ec), a donde mayormente está orientado este trabajo;
- Un sitio gubernamental (www.educacionsuperior.gob.ec), con relación directa a la educación;
- Un sitio de un medio de comunicación (www.ecuavisa.com), que se espera que cambie muy frecuentemente y
- Un sitio comercial (www.mercadolibre.com.ec), en representación de los `.com` globales.

Con la información recolectada de cada uno de estos sitios se puede observar provisionalmente (porque el período de tiempo analizado ha sido de 5 días) que:

- El sitio web de la PUCE, sede Santo Domingo, no ha cambiado ni a nivel de código ni a nivel de su imagen, al menos no durante esta semana. Esto puede deberse a la época que quizá no requiere de mayores actualizaciones, o quizá la política de administración del sitio es la de modificar los contenidos internos más que la página inicial:

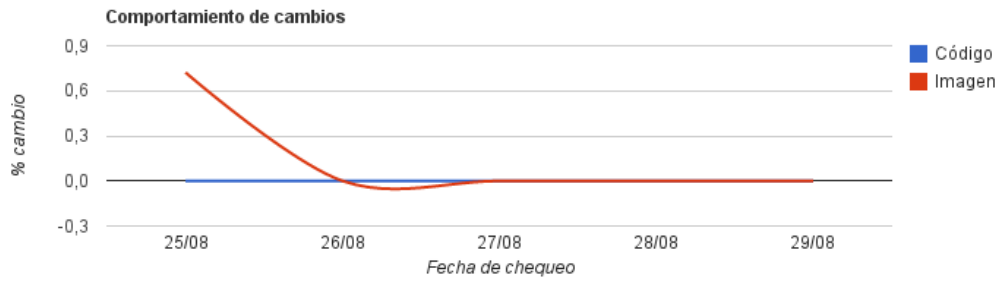


Fig 27. Tendencia de cambios de www.pucesd.edu.ec.

- El sitio web de la EPN, tuvo un cambio pequeño del 24 al 25 de agosto y el más significativo del 27 al 28 de agosto. Sin embargo todos los cambios se han mantenido por debajo del 10%, mismo que pueden considerarse totalmente normales luego de haber revisado los cambios a nivel de imagen.

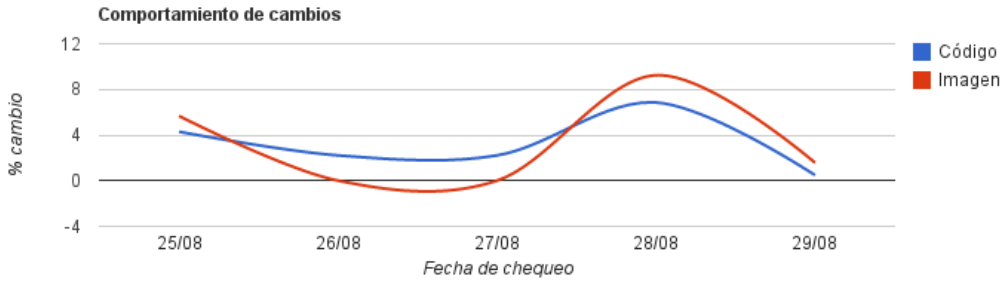


Fig 28. Tendencia de cambios de www.epn.edu.ec.

- El sitio web de la Senescyt no tuvo cambios durante los primeros 3 días y tuvo cambios, mismos que sí se presentaron de forma consistente en los últimos 2, por la tendencia, parecerían ser normales:

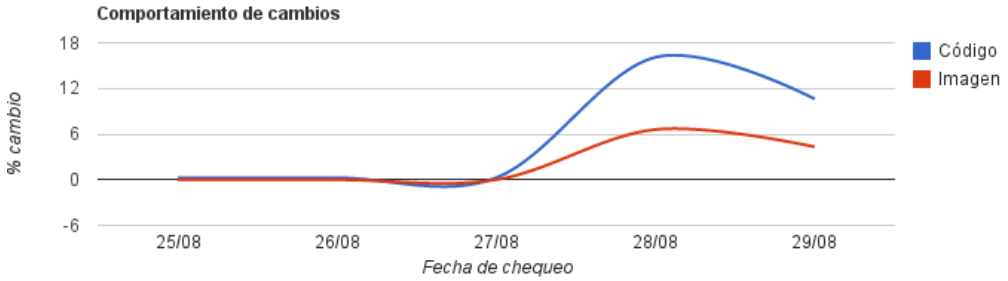


Fig 29. Tendencia de cambio de www.educacionsuperior.gob.ec.

- El sitio web de Ecuavisa, como era de esperarse, es el que se muestra mucho más activo en función de cambios. Sin embargo, los valores que cambian mayormente son a nivel del código, manteniéndose alrededor del límite establecido del 35%, por lo cual incluso se muestran 3 lecturas con alerta.. A nivel de imagen difícilmente alcanzan el 12%:

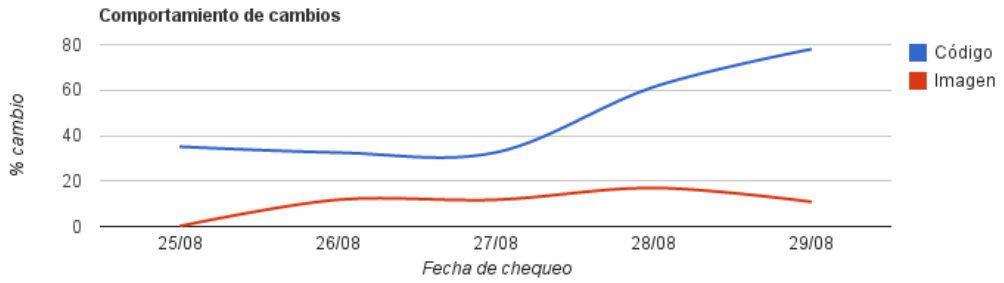


Fig 30. Tendencia de cambio de www.ecuavisa.com.

- Finalmente, MercadoLibre tiene una muy consistente tendencia de cambios a nivel de código, que igualmente se reflejan a nivel de imagen, pero quizá es el ejemplo más claro de que puede haber una normal y natural discrepancia entre el nivel de cambios en código y el nivel de cambios a nivel de lo que el usuario finalmente percibe visualmente. Los cambios de código son iguales todos los días con casi el 40% de cambios y muy pequeñas variantes por sobre el 3.5% a nivel visual:

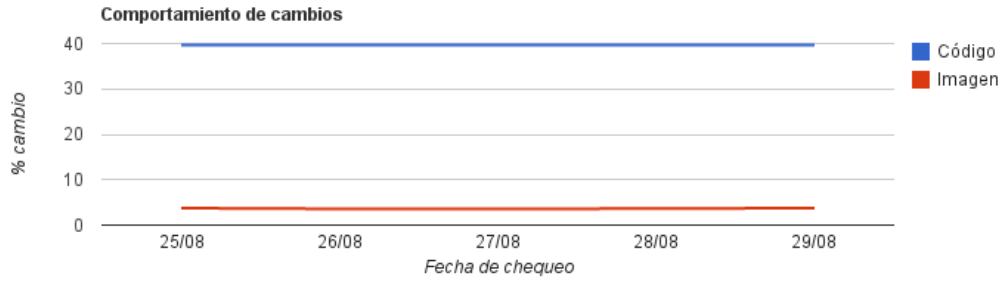


Fig 31. Tendencia de cambios de www.mercadolibre.com.ec.

Este último caso también se muestra como principal candidato a un ajuste en los niveles de alerta, ajuste que bien pudiera realizarse en los otros también, pero que no se ha realizado, sencillamente por tratarse de datos de prueba usados para el desarrollo de éste trabajo.

Tampoco se puede descartar un potencial problema de desviación en el sistema de detección de cambios en el código, que esté devolviendo resultados muy altos para ciertos casos que pudieran variar en código de cierta forma. Ésto puede estar relacionado directamente al hecho de que comm realiza una comparativa únicamente a nivel de líneas y si un solo caracter cambia en un línea, arroja el mismo valor que si se cambiara la línea entera, pero justamente previendo aquello se pensó en el sistema adicional y complementario del análisis imagenográfico de cambios.

Capítulo 5 :: CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Existen diversas herramientas de captura automatizada de código HTML en SL, con las cuales se puede obtener el código de una página o del sitio completo, en poco tiempo y esfuerzo.
- `wget` es la herramienta más idónea para la captura automatizada de código HTML de un sitio web, por su capacidad, directamente disponible, de descargar el sitio web completo en un archivo monolítico.
- Existen varias herramientas para captura visual automatizada de un sitio web en SL, con las cuales y con diferentes requisitos adicionales, se puede obtener una copia visual completa o parcial de un sitio web.
- PhantomJS es la herramienta escogida por su eficiencia y rapidez para la captura imagenográfica de un sitio web, porque no requiere de sistemas adicionales ni dimensiones de captura, para obtener una copia visual completa de un sitio dado.
- Es viable realizar un sistema CLI para el análisis de cambios en código fuente HTML, usando `comm` como herramienta de SL para detección de diferencias entre versiones del código del mismo sitio web.
- Es viable realizar un sistema CLI para el análisis de cambios imagenográficos, usando ImageMagick como herrameinta de SL para la detección de diferencias entre versiones de capturas visuales del mismo sitio web.
- Es necesario usar un procesamiento adicional para obtener resultados numéricos relevantes a partir de la salida original de la comparativa realizada de código con `comm`. y de imágenes con ImageMagick.
- Se puede usar Yii Framework, para desarrollar un sistema web de administración y presentación de información y resultados, minimizando los tiempos y tareas de programación.
- Es viable, económico y productivo realizar un sistema de detección de defacements de sitios web en SL.

5.2. Recomendaciones

- Mejorar el sistema de análisis de cambios de código HTML incluyendo la posibilidad de realizar una comparativa más específica que la de cambios a nivel de líneas de código.
- Realizar un sistema de detección complementario que relacione los cambios de código con los de imagen pues lo cambios de código fuente HTML no necesariamente representan cambios visuales notables.
- Elaborar un RPM que facilite la distribución, mantenimiento e instalación de Qhaway por parte de cualquier interesado/a en hacerlo.
- Publicar el proyecto en SourceForge, GitHub o similares, para ofrecer la posibilidad a la comunidad del SL de involucrarse e idealmente colaborar con el desarrollo y extensión del mismo.
- Adaptar el sistema para funcionar como módulo sensor en el proyecto del Sistema de detección, procesamiento y envío de alertas de seguridad del CSIRT-CEDIA[36].
- Usar a Qhaway como precedente que es posible y viable crear nuestras propias herramientas con SL.

BIBLIOGRAFÍA

[1] http://zone-h.org/archive

[2] http://csirt.cedia.org.ec/

[3] https://www.shadowserver.org/wiki/

[4] http://www.andes.info.ec/actualidad/9404.html

[5] http://www.ultimasnoticias.ec/noticias/6752-inocentada-para-bsc.html

[6] http://www.elcomercio.com.ec/deportes/Barcelonista-hackea-pagina-web-Emelec_0_818918297.html

[7] Trabajo de tesis: Estudio de las características de seguridad de servidores web en entornos de Software Libre aplicables a la protección de sitios dinámicos. Ernesto Pérez, PUCE. 2014.

[8] http://insecure.org/tools/tools-es.html

[9] http://www.c-sirt.org/defacement,

http://actualidad.rt.com/actualidad/view/116869-hackers-blancos--ciber-espacio-ruso

[10] http://www.cpppe.umd.edu/Bookstore/Documents/2005CSISurvey.pdf

[11] http://scadahacker.com/library/Documents/Insider_Threats/CSI - 2010-2011 Computer Crime and Security Survey.pdf

[12] The Reaction Time to Web Site Defacements. Internet Computing, IEEE (Volume:13 , Issue: 4)

[13] http://www.pcworld.com/article/2140210/why-use-open-source-security-and-quality-top-companies-lists.html

[14] https://www.fsf.org/es

[15] http://www.linuxfoundation.org/about/staff#torvalds

[16] What the Tech Industry Has Learned from Linus Torvalds: Jim Zemlin at TEDxConcordiaUPortland.

https://www.youtube.com/watch?v=7XTHdcmjenI [03:07-04:18]

[17] http://www.gnu.org/software/bash/

[18] http://es.wikipedia.org/wiki/LAMP

[19] https://github.com/austincheney/Pretty-Diff

[20] http://nodejs.org/

[21] https://fedoraproject.org/wiki/EPEL/es

[22] https://github.com/kpdecker/jsdiff

[23] http://phantomjs.org/

[24] http://anku.ecualinux.com/

[25] https://gist.github.com/astanin/626356

[26] http://cs.brown.edu/~aritz/files/SDMpresentation.pdf

[27] https://code.google.com/p/jmotif/wiki/ZNormalization

[28] http://pdiff.sourceforge.net/

[29] http://www.imagemagick.org/Usage/compare/#statistics

[30] http://www.graphicsmagick.org/

[31] http://www.imagemagick.org/script/command-line-options.php?#metric

[32] http://www.yiiframework.com/

[33] http://www.larryullman.com/2010/10/07/creating-a-yii-console-application/

[34] http://www.larryullman.com/series/learning-the-yii-framework/

[35] https://developers.google.com/chart/?hl=es

[36] http://csirt.cedia.org.ec/servicios/sistema-de-avisos/

Estudio de herramientas de SL para la creación de sistemas de detección automatizada de defacements de Sitios Web.

Paul Fernando Bernal Barzallo, Quito – Ecuador, paul.bernal@cedia.org.ec, pbernal@ecualinux.com, 09 8466 9053