DESARROLLO DE FUNCIONALIDADES PARA LOS PRODUCTOS CLOUD DEFENDER Y CLOUD INSIGHT DE LA EMPRESA ALERT LOGIC INC COLOMBIA

Modalidad Pasantía Informe Final

FAIR ANDRÉS TARAPUÉS HIDALGO
CÓDIGO: 1356194
fair.tarapues@correounivalle.edu.co

Director en la Empresa
JOSÉ JULIÁN REINA MATERÓN *ING
Magister en Administración
jmateron@alertlogic.com

Directora EISC Universidad del Valle sede Tuluá
YURI MERCEDES BERMUDEZ MAZUERA *ING
Magister en Administración
yuri.bermudez@correounivalle.edu.co

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas

Tuluá, Junio de 2017

Nota de aceptación:

_____

_____

_____

_____

_____

Firma del Coordinador

_____

Firma del Jurado 1

_____

Firma del Jurado 2

Tuluá, Mayo de 2017

Resumen

El presente trabajo tiene como objetivo dar un informe completo sobre las actividades realizadas durante un año en la empresa Alert Logic Inc. Colombia, en la modalidad de pasantía.

Gracias a la iniciativa de esta compañía en conjunto con la Universidad del Valle nace el programa *Intership* el cual tiene como objetivo la capacitación de estudiantes del programa de Ingeniería de sistemas en las tecnologías usadas por la empresa.

Palabras Clave: Pasantía, Pruebas Unitarias, Defectos, AngularJS, Dunkirk.

Abstract

The present work aims to give a complete report on the activities carried out during a year in the company Alert Logic Inc. Colombia, in the form of internship.

Thanks to the initiative of this company in conjunction with the Universidad del Valle, the Intership program is born, which aims to train students in the systems engineering program in the technologies used by the company.

Key Words: Internship, Unit Tests, Defects, AngularJS, Dunkirk.

Agradecimientos

A la profesora Yuri Mercedes Bermúdez por su entrega, apoyo, dirección y acompañamiento en este proceso el cual me ha ayudado a crecer como persona y como profesional.

A mis Managers Jose Reina, Pablo Vélez y Diego Ruiz que me guiaron durante este proceso y me brindaron la oportunidad de desarrollarme como profesional y como persona en la empresa Alert Logic Inc Colombia, Gracias por enseñarme el valor del trabajo en equipo, la comunicación y el cumplimiento en cada una de las actividades desarrolladas.

A mis compañeros de trabajo Maryit Sánchez, Andrés Artunduaga, Carlos Orozco, Cristhian Fuertes, Jiovanny Ibarguen, Cristian Rendón, Julián Mondragón, Juan Pablo León, Juan Kremer, Rowinson Gallego, Gisler Garcés, Julián Galvis y Jorge Mario Valencia los cuales me brindaron su guía, su conocimiento y su acompañamiento en cada una de las actividades desarrolladas durante la pasantía.

A la Universidad del Valle junto con sus destacados docentes por brindarme la oportunidad de formarme como profesional.

CONTENIDO

ÍNDICE DE FIGURAS

:

Capítulo 1

Introducción

1.1. Descripción General

El presente trabajo de grado en modalidad pasantía tiene como objetivo presentar un informe detallado de las actividades realizadas dentro de la empresa Alert Logic Inc. Esta, es una empresa de desarrollo de software enfocada a prestar servicios de seguridad en la nube por medo del modelo Security as a Service, la cual, tuvo la iniciativa de crear un programa de pasantía en acuerdo con la Universidad del Valle para adquisición de nuevos talentos dentro de su personal dando así origen al programa de pasantía descrito en el presente documento. Este, consiste en la capacitación de estudiantes del programa de Ingeniería en Sistemas de la Universidad del Valle en las tecnologías, lenguajes de programación, frameworks y forma de trabajo de la empresa durante un periodo de un año. Es importante mencionar que el desarrollo de la pasantía se realizó en conjunto con los estudiantes Carlos Orozco y Andrés Artunduaga los cuales conforman el "equipo de desarrollo" mencionado a lo largo del documento.

Al inicio de la pasantía, el "equipo de desarrollo" y los jefes encargados planearon una serie de actividades las cuales consistieron en la contextualización de la empresa respecto a sus políticas, modelo de trabajo, herramientas de desarrollo usadas, lenguajes de programación y reglamento interno, la implementación de pruebas unitarias sobre los productos de la empresa, la solución a defectos reportados por los clientes de la empresa y la creación de nuevos componentes de software. Cabe mencionar que por fuera de estas actividades, el "equipo de desarrollo" brindó apoyo en la automatización de tareas requeridas por la empresa tales como la generación automática de documentación de historias de usuario y la creación de scripts para la ejecución de comandos en consola.

Todas las actividades anteriormente mencionadas únicamente se limitaron al desarrollo de la interfaz gráfica de usuario de los productos de la empresa y fueron supervisadas por el equipo de *User Interface* de la empresa Alert Logic a cargo de los *Managers* José Reina Materón y Pablo Andrés Vélez. Quienes se encargaron de verificar el cumplimiento de cada una de las actividades y de los objetivos propuestos.

1.2. Descripción de la empresa

Alert Logic es un proveedor de soluciones de seguridad en la nube y pionero en la implementación del modelo Security as a Service dentro de sus productos. Esta compañía fue fundada en 2002 por Misha Govshteyn y Matthew Harkrider y tiene su sede principal en Houston Texas con oficinas en Seattle, Cardiff, Cali y Belfast Londres.

Esta compañía tiene más de una década de experiencia, perfeccionando y redefiniendo soluciones seguras, flexibles, diseñadas para trabajar con proveedores de servicios en modalidad de hosting o en la nube ya que cuenta con la capacidad de ofrecer soluciones de detección de intrusiones, análisis de vulnerabilidad y gestión de registros que se complementan con el servicio de monitoreo y vigilancia 24 x 7 ofreciendo así, soluciones de seguridad y cumplimiento normativo para la nube, también, ofrece soluciones de Seguridad como Servicio (SaaS, Security as a Service) especialmente indicadas para ser implementadas en infraestructuras con instalaciones dedicadas o datacenters, en la nube, o en infraestructuras híbridas, entregando a los clientes una visión profunda de seguridad y protección continua.

En la actualidad la empresa cuenta con dos productos de software llamados, Cloud Defender y Cloud Insight, los cuales se encuentran construidos con distintas tecnologías. En el caso de Cloud Defender, fue construido sobre Symfony (Framework de PHP) y AngularJS para el caso de Cloud Insight. [3] Dada la diferencia en tecnologías, Alert Logic está llevando paulatinamente a Cloud Defender a que adopte la tecnología AngularJS. Este proceso de migración implica un gran esfuerzo dentro del equipo de User Interface convirtiéndose en una de las motivaciones para iniciar el programa de pasantía debido a que es necesario que las personas implicadas en este

Proceso conozcan el funcionamiento de los productos y puedan realizar los cambios requeridos.

1.3. Objetivos en la pasantía

1.3.1. Objetivo General

Desarrollar nuevas funcionalidades utilizando las metodologías y tecnologías de desarrollo web, de acuerdo a los requerimientos definidos para la convergencia de Cloud Defender y Cloud Insight.

1.3.2. Objetivos Específicos

- ✓ Apropiar metodologías y tecnologías de desarrollo web utilizadas en los productos en Alert Logic.
- ✓ Implementar pruebas unitarias específicas para incrementar la cobertura de código en los productos Cloud Defender y Cloud Insight.
- ✓ Implementar soluciones a defectos en los productos Cloud Defender y Cloud Insight reportados por los clientes de Alert Logic.
- ✓ Implementar funcionalidades específicas en los productos Cloud Defender y Cloud Insight utilizando la tecnología AngularJS.

Capítulo 2

Marco Referencial

A continuación se definen los conceptos que hacen parte de las tecnologías involucradas en el desarrollo de la pasantía las cuales definen del alcance y la aplicabilidad dentro del trabajo realizado. En primera instancia, se explicará detalladamente el funcionamiento de los productos principales de la empresa en este caso, Cloud Defender y Cloud Insight. Y a continuación se profundizará en los conceptos propios del desarrollo de la pasantía como: el modelo de negocio de la empresa, herramientas de desarrollo web y metodología de desarrollo utilizada.

2.1 Marco Teórico

2.1.1. Productos de la Empresa

2.1.1.1. Cloud Defender

Alert Logic Cloud Defender es un servicio de seguridad informática que provee monitoreo 24x7 por expertos en seguridad junto con detección de amenazas en red, gestión de registros (logs), evaluación de vulnerabilidades y protección de aplicaciones web. Gracias a la automatización y análisis integrado se obtiene un monitoreo continuo para proteger la información del cliente ofreciendo así un producto que cumple con todas las exigencias de seguridad requeridas. Las siguientes son características que provee este producto. [4]

- Gestión de seguridad centralizada: Protege los datos y aplicaciones localmente, en la nube y en ambientes híbridos a través de múltiples capas, redes, sistemas y aplicaciones web.
- Administración de detección de amenazas y respuesta: Detección proactiva de amenazas 24x7, monitoreo y rápida notificación por expertos en seguridad.

- Despliegue rápido: Expertos se encargan del despliegue, configuración, calibración y capacitación por lo que la solución es activada rápidamente4
- Cumple con los estándares: Es conforme con las regulaciones de ley: PCI DSS (Payment Card Industry Data Security Standard), HIPAA(Health Insurance Portability and Accountability Act), and Sarbanes-Oxley.
- Visibilidad dentro de la infraestructura TI: Evalúa el estado de la seguridad para entender a qué riesgos se enfrenta.

2.1.1.2. Cloud Insight.

Alert Logic Cloud Insight es una solución para la configuración y administración de vulnerabilidades de los servicios web de Amazon (AWS) nativos. Está diseñada para reducir la complejidad de proteger las cargas de trabajo realizadas por estos. Posee automatización integrada y monitoreo continuo, por tanto provee visibilidad a través del entorno AWS identificando vulnerabilidades a medida que surgen. Entre las características de este producto se encuentran las siguientes.

- Nativo de AWS: Cloud Insight usa la API de AWS y datos de CloudTrail para mantener un inventario correcto y actualizado de los activos y topologías de red. Gracias a su detección automática integrada y su capacidad de auto-despliegue, puede detectar nuevos activos que son desplegados, todo esto sin necesidad de intervención manual.
- Protección continúa en un entorno cambiante: Cloud Insight monitorea constantemente el entorno en búsqueda de vulnerabilidades desde la cuenta hasta el host, descubriendo automáticamente los puntos débiles en el entorno sin ningún esfuerzo por parte del cliente. Cuando el entorno cambia, Cloud Insight lo detecta y escanea los activos nuevos, cambiados o modificados para buscar vulnerabilidades.

- Foco en la solución y no en la vulnerabilidad: En vez de construir una larga lista de vulnerabilidades a investigar, priorizar y determinar cómo resolver, Cloud Insight da al cliente la información que necesita para eliminarlas rápidamente y disminuir el riesgo. Esto permite que el cliente enfoque sus esfuerzos en resolver las brechas de seguridad que tienen más impacto.

2.1.2. Modelos de Negocio

A Continuación se mencionan los conceptos más relevantes relacionados con la forma en que la empresa ofrece sus productos y la arquitectura utilizada en la implementación de los mismos.

2.1.2.1. Cloud Computing.

Cloud computing se refiere a la entrega bajo demanda de recursos informáticos y aplicaciones a través de internet con un sistema de precios basado en el consumo realizado. [5] La importancia de usar los servicios “Cloud” por parte de las empresas radica en que estos proporcionan acceso rápido y flexible a recursos informáticos de bajo costo. Gracias a Cloud Computing, las compañías no necesitan realizar grandes inversiones en la adquisición de equipos ni en la administración de los mismos. En lugar de eso, podrá aprovisionar exactamente el tipo y el tamaño de recursos informáticos que necesite para hacer realizad el propósito de la empresa.[5]

Cloud Computing ofrece un método sencillo de obtener acceso a servidores, almacenamiento, bases de datos y una amplia gama de servicios de aplicaciones a través de internet. [5] Los proveedores de Cloud computing, son propietarios y responsables del mantenimiento de los equipos para el funcionamiento de las aplicaciones, mientras el cliente se dedica a usar los recursos por medio de una aplicación web.

2.1.2.2. Software as a Service.

Software as a Service, es una arquitectura de desarrollo en la que se pueden distinguir dos categorías:

- Servicios orientados al negocio: estos, se ofrecen en las empresas de todos los tamaños y generalmente son soluciones destinadas a procesos propios del negocio, tales como, finanzas, gestión de cadenas de suministros y relaciones con los clientes.
- Servicios orientados al consumidor: estos, se ofrecen al público en general y son servicios orientados al consumidor como por ejemplo, correos electrónicos. Generalmente son gratuitos y soportados por la publicidad.

El SaaS basa su funcionamiento en tres pilares fundamentales que son: el modelo de negocio, arquitectura de las aplicaciones y la estructura operacional. Tal como lo muestra la figura 2.1.

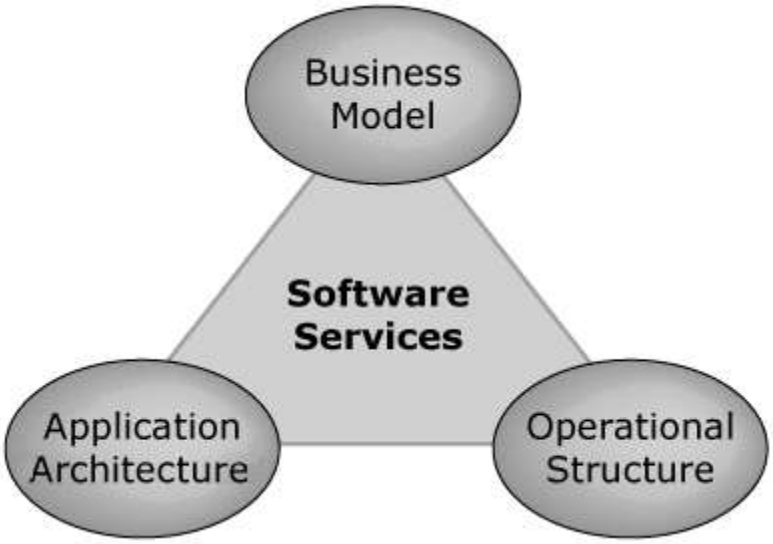


Figura 2.1: Áreas de funcionamiento del SaaS [2]

1. Modelo de negocio de SaaS

Hoy en día, la mayoría de productos de software se sigue vendiendo mediante la modalidad de "software as a product". Esto consiste en que el cliente compra una licencia para su uso y lo instala en su hardware. Convirtiéndose así en "dueño" del producto.

Con el modelo SaaS, la idea de ser "dueño" del producto se torna un tanto extraña para el cliente ya que en lugar de poseer el producto, se ofrece acceso a este mediante una suscripción. Esto, implica que la aplicación se ejecuta en un servidor externo a la empresa reduciendo así los costos de mantenimiento y soporte.

2. Arquitectura de la aplicación

Dentro de la arquitectura SaaS hay tres factores claves para asegurar el éxito del producto: La escalabilidad, la eficiencia para atender a múltiples clientes y la configuración.

- Escalabilidad: Significa, maximizar la concurrencia y uso de recursos de la aplicación de manera eficiente. Por ejemplo, la distribución de recursos tales como hilos, conexiones de red, almacenamiento en caché y particiónamiento de bases de datos.

- Multitendencia: Puede ser el cambio más significativo en comparación con la arquitectura de desarrollo tradicional, ya que los usuarios comparten en cierta manera la misma aplicación pero esto no debe afectar el funcionamiento de sus servicios. Para hacer posible esto, se requiere de una arquitectura capaz de maximizar el intercambio de recursos a través de los usuarios pero diferenciando los datos de cada uno. Por tanto no se puede simplemente escribir código personalizado para un solo cliente ya que cualquier cosa que se modifique afecta directamente a los demás clientes.

- Configuración: En lugar de la personalización, el modelo SaaS plantea el uso de metadatos para configurar la forma en que aparece la aplicación y se comporta para los usuarios. El reto para implementar esta arquitectura es asegurar que la configuración por parte del cliente sea simple y que no tenga que incurrir en costes de desarrollo adicional.

3. Estructura Operativa.

   Otro aspecto importante a tener encuentra dentro del modelo SaaS, es la estructura operativa de la aplicación, es decir, lo que se necesita para mantener disponible la aplicación.
   Proporcionar software como servicio añade una capa adicional en función de ejecutar el plan de negocios de la compañía, esto consisten en:

   - Seguimiento a los clientes y uso de recursos.
   - Restringir el acceso en ciertos momentos del día.
   - Vigilar el acceso al sitio.
   - Realizar otras funciones con el fin de garantizar una experiencia perfecta para los clientes.

2.1.3. Herramientas de Desarrollo web

2.1.3.1. AngularJS

AngujarJS es un Framework de JavaScript soportado por Google para el desarrollo de aplicaciones web dinámicas bajo el modelo MVW (Model View Whatever Works for you). [7] Esta herramienta, contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web del lado del cliente que permiten el intercambio de datos con el servidor de forma que este se libera de cargar complemente la página y solo se limita al envío de datos.

2.1.3.2. PHP

El lenguaje de programación PHP (*Personal Home Page* históricamente, oficialmente acrónimo recursivo de PHP: *Hypertex Preprocessor*). Fue diseñado en 1994 por Rasmus Lerdorf. [8] bajo la licencia GNU de software libre.

Actualmente, PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podría incorporar directamente en el documento HTML en lugar de llamar a un archivo externo. El código PHP es interpretado por un servidor web con un módulo de procesador propio del lenguaje, el cual genera la página web resultante.

PHP se considera uno de los lenguajes más flexibles, potentes y de alto rendimiento conocido hasta el día de hoy para el desarrollo de scripts [9] del lado del servidor, de hecho es usado por empresas como Facebook para saciar la demanda de tráfico en el servidor.

2.1.3.3. PHPUnit

PHPUnit es un entorno de desarrollo creado para realizar pruebas unitarias en el lenguaje de programación PHP, este es un Framework de la familia xUnit y fue creado por Sebastian Bergmann. PHPUnit se creó con la idea de crear una herramienta que detecte los errores de código de forma rápida y sencilla, permitiendo así que se integre con el flujo de trabajo del equipo de desarrollo. Como todos los Frameworks de pruebas unitarias, PHPUnit utiliza assertions para verificar que el comportamiento de una unidad de código es el esperado [12]. Permitiendo así detectar el comportamiento de la función y los datos que esta arroja.

2.1.3.4. Rally Software

Rally Software Development Corp es un proveedor de soluciones de software y servicios de clase empresarial que facilita el desarrollo ágil de productos de software. Esto, con el objetivo de acelerar el ritmo de la innovación, mejorar el rendimiento, y responder eficazmente a la evolución de los mercados competitivos y necesidades del cliente por parte de las compañías que desarrollan de software.

Rally ofrece una plataforma SaaS que transforma la manera en que las organizaciones gestionan el ciclo de vida de desarrollo de software mediante la alineación de los objetivos estratégicos de negocio, colaboración entre los equipos, y transparencia en el desarrollo [13].

2.1.3.5. Git

Git es un servicio de control de versiones distribuido diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales. Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 para el desarrollo del kernel de Linux. La licencia de GIT es libre y existen distribuciones oficiales para los sistemas operativos Mac OS X, Windows, Linux y Solaris. [10] La siguiente figura muestra los principales comandos de GIT (véase figura 2.2).

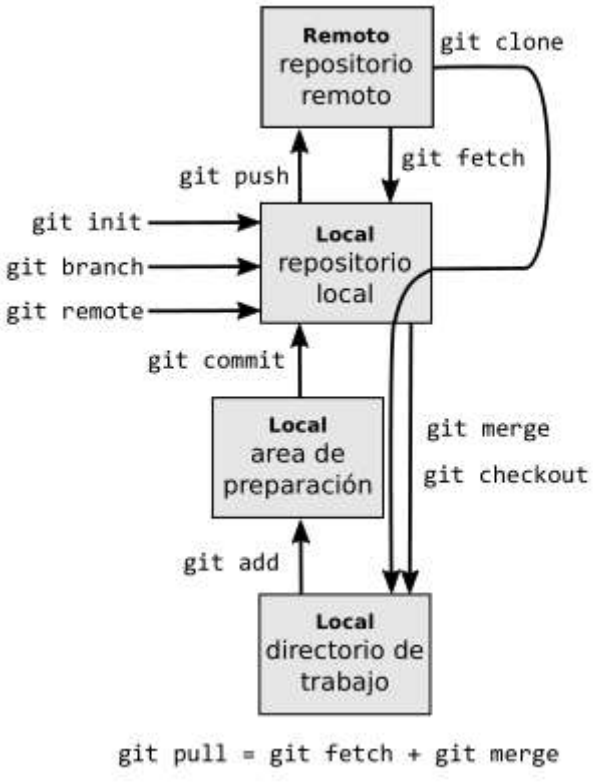


Figura 2.2: principales comandos de GIT [10]

La experiencia de Linus Torvalds en la gestión de la integración de diferentes aportes en un proyecto distribuido de la magnitud del kernel de Linux determino las siguientes decisiones de implementación. [10]

- Versiones no incrementales. Git almacena cada cambio como una instantánea de todos los archivos del proyecto. Para ser eficiente, si el archivo no ha sido modificado, solo se almacena un enlace al archivo idéntico previamente almacenado.
- Autenticación criptográfica de la historia. El identificador de cambio se computa utilizando un algoritmo criptográfico que utiliza como entrada el cambio y la historia completa de cambios. Esto permite que cualquier cambio sea detectado por Git
- Trabajo fuera de línea. Por ser un distribuido, cada repositorio Git es un repositorio completo capaz de funcionar sin acceso a la red o al resto de repositorios.

2.1.3.6. GitHub

GitHub es un servicio comercial de alojamiento de repositorios Git remotos creado en el año 2008. GitHub proporciona una interfaz Web que permite al usuario registrado crear repositorios vacíos o por clonación de otro repositorio hospedado en GitHub, enviar solicitudes de cambio entre repositorios y gestionar dichas solicitudes. Además, los repositorios en GitHub pueden actuar como repositorios remotos de repositorios locales. Los repositorios que se crean en GitHub son por defecto de acceso público. [11] Sin embargo por medio de una cuenta de pago se pueden alojar repositorios privados. Cada repositorio es propiedad de una cuenta de usuario o de organización. Las modificaciones a un repositorio solo se pueden realizar por usuarios que ha iniciado sesión y que estén autorizados a modificar su contenido.

Además del alojamiento, GitHub proporciona a cada repositorio una wiki, un gestor de tareas, un completo sistemas de gestión de comentarios, un cuadro de control con grafos sociales e, incluso una página web propia. Finalmente, todo este contenido puede ser accedido y manipulado mediante un API web.

2.1.4. Metodología de Desarrollo.

2.1.4.1. Scrum.

Scrum es una metodología de desarrollo ágil dentro de la cual las personas pueden abordar problemas adaptativos y complejos ofreciendo productos del más alto valor posible. Esta metodología de desarrollo se ha venido usando como una herramienta útil dentro de los equipos de desarrollo desde los 90's. Aunque es necesario aclarar que Scrum no es un proceso o una técnica para construir productos. Sino más bien, es una metodología que se puede aplicar a varios procesos y técnicas.

La metodología Scrum consiste en equipos, roles, actividades, artefactos y reglas las cuales tienen un propósito específico dentro del éxito del Scrum y que trabajan de forma conjunta en pro de lograr el objetivo del proyecto.

2.1.4.1.1. Equipos del Scrum.

Los equipos Scrum se componen de el *Product Owner*, el equipo de desarrollo y el Scrum master. Estos son auto-organizados e inter-funcionales lo cual permite que cada equipo escoja la mejor forma de trabajar en sus actividades sin afectar el trabajo de los demás equipos, este concepto es una característica fundamental del Scrum ya que permite mejorar la productividad y la retroalimentación del equipo.

1. Product Owner.

El product owner es el responsable de maximizar el valor del producto y el trabajo del equipo de desarrollo. Este, se compone de una sola persona y es el encargado de gestionar el product backlog y otras tareas tales como:

- Expresar claramente los elementos del backlog.
- Ordenar los elementos del backlog buscando alcanzar los objetivos propuestos.
- Optimizar el valor del trabajo del equipo de desarrollo.

- Asegurarse de que el product backlog es claro, trasparente y mostrar los ítems que serán trabajados por el equipo.
- Asegurarse de que el equipo entiende los ítems y el nivel de necesidad de cada uno de estos.

2. Equipo de Desarrollo.

El equipo de desarrollo consiste en profesionales que trabajan sobre los incrementos del producto en cada uno de los Sprints. Y son los encargados de entregar el producto "hecho" al product owner. El equipo de desarrollo cuenta con las siguientes características:

- Es auto-organizado.
- Es inter-funcional.
- Los miembros del equipo de desarrollo individual pueden tener habilidades especializadas y áreas de enfoque, pero la rendición de cuentas pertenece al Equipo de Desarrollo en su conjunto.

3. Scrum Master.

El Scrum Master es la persona encargada de asegurar el aprendizaje y adaptación del Scrum dentro del equipo. Enseñando la teoría respecto a la metodología, las practicas que lo componen y las reglas. Se puede describir al Scrum Master como un líder que sirve al equipo potencializando las capacidades de los miembros en cada una de las iteraciones y asegurando la calidad en cada uno de los incrementos. Las características principales del Scrum master son:

- Dirigir al equipo de desarrollo.
- Ayudar al equipo a incrementar la calidad del producto final.
- Trabajar con otros scrum masters en mejorar la implementación de la metodología dentro de la compañía.

2.1.4.1.2. Actividades del Scrum.

Las actividades del Scrum son usadas dentro de la metodología como una herramienta para la organización del equipo. Todas estas, tienen un tiempo límite dentro del cual se tienen que realizar y cada una cumple un propósito específico dentro de la metodología, entre las que se destacan:

1. Sprint.

El Sprint es una de las actividades más importante dentro del Scrum, esta, tiene una duración no mayor a un mes y su importancia radica en que durante el tiempo planeado

Se debe entregar un producto "hecho" al product owner. Dentro del Sprint se destacan otras actividades tales como el S*print planning, el Daily Scrum y el Sprint Retrospective.* [11].

2. Sprint Planning.

Esta actividad se planea junto con el equipo de desarrollo y consiste en planificar todas las actividades que se van a realizar durante el Sprint.
Esta actividad del Sprint tiene un tiempo máximo de 8 horas por mes de trabajo y es el Scrum Master quien se asegura de que la reunión se realice y que se entienda el propósito de la misma. El Sprint Planning responde básicamente a estas dos preguntas.

- ¿Qué se puede entregar en el Incremento resultante del próximo Sprint?
- ¿Cómo se llevará a cabo el trabajo necesario para lograr el Incremento?

3. Daily Scrum.

El Daily Scrum es una reunión que toma cerca de 15 minutos en la cual el equipo de desarrollo sincroniza sus actividades y crea un plan para las siguientes 24 horas. Esto con el objetivo de inspeccionar el trabajo desde el ultimo Daily y planear el trabajo que será realizado hasta la siguiente reunión.

Durante la sesión el equipo de desarrollo responde a las siguientes preguntas.

- ¿Qué actividades realicé ayer en pro del objetivo del proyecto?
- ¿Qué actividades realizaré el día de hoy?
- ¿he tenido impedimentos o problemas en la realización de las actividades?

4. Sprint Retrospective.

El Sprint Retrospective es la oportunidad que tiene el equipo de Scrum (Scrum master y Equipo de desarrollo) para revisarse y crear un plan de mejoramiento según lo hecho en el anterior Sprint y pensando en el siguiente.
Esta actividad del Scrum toma aproximadamente 1 hora y es el Scrum Master quien se Asegura que la reunión sea realizada y entendida por los miembros del equipo de Desarrollo. Cabe aclarar que la labor del Scrum Master en la reunión es imparcial y siempre velando por el logro de los objetivos. Durante el Sprint Retrospective se tratan los siguientes ítems.

- Revisar el anterior Sprint en términos de: Personas, Relaciones, Procesos y Herramientas.
- Crear un plan de mejoramiento teniendo en cuenta los aspectos malos y buenos del anterior Sprint por medio de compromisos adquiridos por el equipo.

2.1.4.1.3. Artefactos de Scrum

1. Product Backlog

Es el inventario en el que se almacenan todas las funcionalidades o requisitos del cliente. Estos, serán los que tendrá el producto al final del ciclo de desarrollo. Los requisitos serán ordenados en una lista de prioridad por el cliente y el Scrum Master, el cual indicará el coste estimado de los requisitos.
Es necesario que antes de empezar el primer “Sprint” se definan cuáles van a ser los objetivos del producto y tener la lista de requisitos ya definida, no es necesario que sea muy detallada pero si lo suficiente para que el equipo pueda trabajar.

Finalmente, el Product Backlog irá evolucionando mientras el producto exista en el mercado. [11]

2. Sprint Backlog

Es la lista de tareas que elabora el equipo durante la planificación de un sprint. Se asigna las tareas a cada persona y el tiempo que queda para terminarlas. De esta manera, el proyecto se descompone en unidades más pequeñas y se puede determinar o ver en que tareas no se está avanzando.
El sprint backlog, se muestra como una lista ordenada por prioridades para el cliente y puede existir independencia entre una tarea u otra.

3. Incrementos

Representa los requisitos que sean completado en una iteración y que son perfectamente operativos. Segundo los resultados obtenidos, el cliente puede realizar cambios cuando estos sean necesarios. [11]

Capítulo 3

Desarrollo del proyecto

Para el desarrollo de la pasantía fue de suma importancia la implementación de la metodología de desarrollo ágil Scrum. La cual permitió al equipo coordinar las actividades durante cada sprint y fortalecer el trabajo grupal para el logro de los objetivos propuestos. Muchas actividades y/o elementos de scrum fueron implementadas en pro del cumplimiento de los objetivos planteados anteriormente, entre los que se encuentran:

- **Scrum Daily Meeting**: tal como se menciona anteriormente, esta es una actividad del scrum que se realiza diariamente junto con el "equipo de desarrollo" y en ella se habla acerca de las tareas que se están trabajando. Esta actividad fue importante durante el desarrollo de la pasantía ya que brindó un espacio para resolver dudas con el Scrum Master y solicitar ayuda si era necesario a otros miembros del equipo de *User Interface*. Esto, con el objetivo de mantener la comunicación con el equipo y resolver bloqueos o impedimentos que podrían retrasar el cumplimiento de los objetivos propuestos durante el sprint.

- **Sprint Planning**: esta actividad consiste en realizar una planeación sobre las tareas que se van a ejecutar durante el sprint y permite al equipo estimar el tiempo de duración de cada tarea. Durante el desarrollo de la pasantía esta actividad permitió al equipo saber exactamente en cuales tareas se iba a trabajar. Esto, con el objetivo de realizar una planeación acertada de los objetivos del sprint teniendo en cuenta los tiempos establecidos en el cronograma de trabajo y el ritmo de trabajo del equipo.

- **Demo Meeting**: esta actividad aunque no hace parte del Scrum es de suma importancia para la compañía ya que permite realizar una serie de presentaciones de todas las tareas realizadas durante cada sprint por medio de diapositivas y pruebas en vivo a los Managers y Scrum Masters. Esto, con el objetivo de mostrar el trabajo realizado y dar constancia de que el equipo esta dando cumplimiento a los objetivos planteados Sprint a Sprint. Cabe resaltar que esta actividad se realiza el ultimo día de cada sprint.

- **Retrospective Meeting**: esta actividad consiste en evaluar el sprint inmediatamente anterior destacando aspectos a mejorar de todas las tareas realizadas por el "equipo de desarrollo", esta actividad ayudó a encontrar falencias al momento de ejecutar el Sprint y comprometer al equipo a superar aspectos que dificultaron el logro de los objetivos propuestos.

Tal como se mencionó en los anteriores ítems, la aplicación de Scrum fue indispensable en el éxito de la pasantía y en el cumplimiento de los objetivos propuestos, cabe destacar el trabajo de los Managers José Reina y Pablo Vélez y los Scrum Masters Diego Ruiz, Rowinson Gallego y Gisler Garcés los cuales brindaron apoyo al "equipo de desarrollo" en cada etapa de la pasantía y facilitaron la comunicación entre las partes. También, aspectos como el cumplimiento, puntualidad, y profesionalismo de todas las partes involucradas facilitaron el logro de los objetivos propuestos.
Como resultado de las actividades desarrolladas durante la pasantía en la empresa Alert Logic Inc. Se presenta a continuación evidencias del cumplimiento de cada uno de los objetivos específicos propuestos.

3.1.1. Desarrollo Objetivo específico I.

Para el desarrollo del objetivo específico I *(Apropiar metodologías y tecnologías de desarrollo web utilizadas en los productos en Alert Logic.)* la empresa brindó una serie de capacitaciones en: los productos Cloud Defender, Cloud Insight, Seguridad Informática, Cultura de Trabajo, Normas se seguridad aplicadas al desarrollo de software, flujo de trabajo con Git y GitHub, Pruebas unitarias con PHPUnit y Pruebas

Unitarias en Jasmine las cuales fueron importantes no solo en el cumplimiento del objetivo como tal sino en la adaptación al ambiente de trabajo y filosofía de la empresa. Estas actividades se llevaron a cabo entre las fechas del 6 Julio 2016 al 29 de Julio de 2016 de la siguiente manera:

3.1.1.1 Entrenamiento en los productos Cloud Defender y Cloud Insight.

Entre las fechas 6 al 15 de Julio de 2016, el equipo de *User Interface* a través de sus desarrolladores senior Diego Ruiz, Rowinson Gallego y Gisler Garcés ofreció una capacitación en el funcionamiento, arquitectura y lenguajes de programación de los productos Cloud Defender y Cloud Insight facilitando así la adquisición de conocimiento por parte del equipo en el funcionamiento de estos productos.

Esta capacitación fue fundamental en el desarrollo de la pasantía ya que facilito al "equipo de desarrollo" conocer que características tienen los productos sobre los que se iban a trabajar, por ejemplo: para el desarrollo sobre Cloud Defender, los programadores acceden por medio de una IP a una máquina virtual personalizada en la nube donde se encuentra almacenado el código de la aplicación para de esta manera mantener aislado el trabajo de cada desarrollador. Por el contrario, para Cloud Insight todo el código se encuentra almacenado localmente.

3.1.1.2. Entrenamiento en Git y GitHub.

Entre las fechas 18 al 22 de Julio de 2016 la empresa ofreció una capacitación en la herramienta de control de versiones Git con el objetivo de instruir al "equipo de desarrollo" en comprender el flujo de trabajo de la compañía y las reglas al momento de desarrollar nuevas funcionalidades dentro de los productos de la empresa.
Dentro de la capacitación se trataron temas de suma importancia para la compañía tales como:

- **Modelo de ramas de Git**

En la figura 3.1 se muestra el flujo de trabajo de GIT usado dentro de la compañía, este consta de una rama principal llamada *"Master"* en la cual se encuentra la versión final del producto, es decir, el código que se está ejecutando del lado del cliente. Este también es conocido como ambiente de producción. En el siguiente nivel se encuentra la rama de *"Integration"* la cual funciona como soporte para *"Master"*. En esta se realizan todas las pruebas necesarias antes de realizar cambios a master. Gracias a esto el ambiente de producción no se ve afectado por los cambios realizados por los desarrolladores. Uno de los aspectos más interesantes dentro del flujo de trabajo de Git son los *"Hot Fixes"*. Estos se definen como reparaciones "en caliente" sobre el código de producción y solamente suceden cuando el ambiente de producción no trabaja de la forma correcta afectando directamente a los clientes de la compañía.

A continuación se encuentra la rama *"Develop"* la cual se nombra de la siguiente manera, ejemplo: 2017-Q1-Sprint6-S, este formato está compuesto por el año actual, el cuarto de mes actual, el sprint actual y el uso de la rama, es decir; si la rama está hecha para nuevas funcionalidades termina en "S" (Service Pack) por el contrario cuando la rama se usa para realizar pruebas unitarias termina en "U" (Unit Test). Durante el desarrollo del sprint, cada desarrollador crea sus ramas "Feature" basándose en la rama "Develop", esto con el fin de mezclar los cambios de cada desarrollador en una misma rama, gracias a esto, el equipo evita conflictos entre los archivos o afectar el ambiente de producción de la compañía.

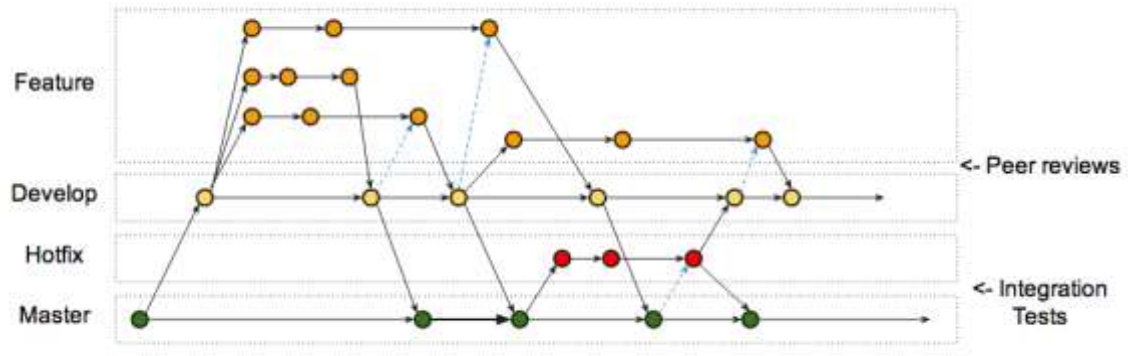


Figura 3.1: flujo de trabajo con Git

- **Code Reviews**

Otro aspecto relevante tratado durante la capacitación en Git fue la implementación de *Code Reviews* durante el proceso de desarrollo del equipo, esta, es una tarea que consiste en la revisión del código de otro desarrollador antes de ser mezclado en la rama "Develop". Gracias a esta tarea el equipo mantiene un control constante de todos los cambios realizados en el código y facilita la comunicación entre los miembros del equipo.

En conclusión, esta capacitación facilito el conocimiento del flujo de trabajo del equipo durante los sprints así como también, los compromisos que adquiere cada desarrollador dentro de la compañía al momento de desarrollar nuevas funcionalidades o atender alguna solicitud de los clientes.

3.1.1.3 Entrenamiento en PHPUnit y Jasmine para la implementación de pruebas unitarias en Cloud Defender y Cloud Insight

Entre las fechas 25 al 29 de Julio de 2016 la empresa ofreció una capacitación en PHPUnit y Jasmine para implementar pruebas unitarias en los productos de la empresa por medio de talleres prácticos y ejemplos previamente preparados por los ingenieros Jiovanny Ibarguen y Gisler Garcés.

Esta capacitación facilitó el conocimiento del código de los productos de la empresa y la manera en que están implementados todos los componentes de estos. Gracias a esta capacitación se logró comprender la dimensión del trabajo que se llevaría a cabo durante la pasantía ya que en el caso de Cloud Defender, la implementación de pruebas unitarias permitirá al todo el equipo de *User Interface* hacer modificaciones al código sin tener que consultar con las altas directivas de la empresa debido a que el reporte de cobertura de pruebas unitarias serviría como sustentación para dichas modificaciones.

Para dar constancia del trabajo realizado, la empresa entregó un certificado el cual se encuentra en el anexo (Anexos_Tesis/Anexo_Objetivo1/CertificadoTraining)

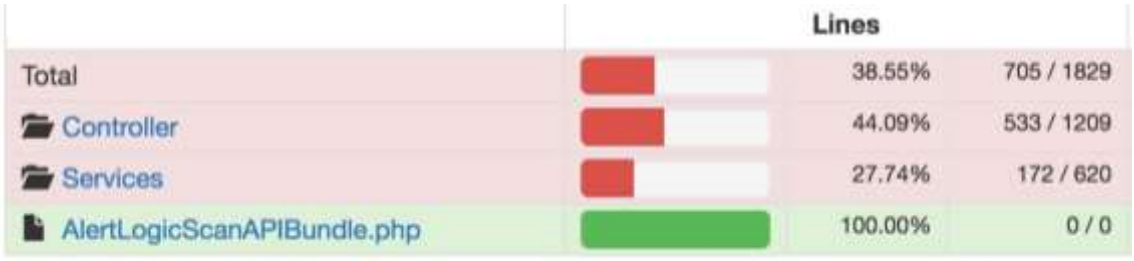
3.1.2. Desarrollo Objetivo específico II.

El desarrollo de este objetivo *(Implementar pruebas unitarias específicas para incrementar la cobertura de código en los productos Cloud Defender y Cloud Insight).* Fue de gran importancia dentro de la pasantía debido a que permitió adquirir conocimiento en el código del producto y ayudar al equipo de *User Interface* a aumentar la cobertura de pruebas unitarias principalmente en Cloud Defender, cabe aclarar que Cloud Insight cuenta con una cobertura de pruebas cercana al 90%, razón por la cual el equipo tomó la decisión de no implementar pruebas en dicho producto.

Para el caso de Cloud Defender, es importante mencionar que este producto esta implementado en Symfony PHP y está divido en módulos independientes llamados *Bundles*. Cada uno de estos módulos tiene asignados procesos independientes dentro del funcionamiento del producto y permitió al "equipo de desarrollo" trabajar de forma aislada en cada componente. El objetivo principal de la implementación de pruebas fue lograr una cobertura superior al 60% en cada Bundle ya que de esta manera el equipo puede realizar cambios en el código sin contar con la aprobación del comité de cambios de la compañía, cabe resaltar que este comité se encarga de revisar todas las actualizaciones de software hechas sobre el ambiente de producción. Teniendo en cuenta esto, la implementación de pruebas unitarias sobre Cloud Defender es una tarea significativa para la compañía.

Durante el desarrollo de la pasantía se trabajaron los Bundles *ScanAPIBundle* y *CoreBundle* respectivamente. El primero se encarga de manejar el proceso de disputa de vulnerabilidades entre el cliente y la compañía, este proceso se define como la aceptación por parte del cliente de las vulnerabilidades encontradas por Cloud Defender, por otra parte *CoreBundle* es el Bundle principal de la aplicación el cual se encarga de almacenar componentes usados por todos los demás Bundles tales como API's, servicios de Autenticación de usuarios, navegación, formularios, notificaciones, re direccionamiento, zona horaria y traducción

En las figuras 3.2 y 3.3 se muestra el estado inicial de *ScanAPIBundle* y *CoreBundle* antes de empezar a implementar las pruebas Unitarias. Cabe mencionar que este reporte es generado por PHPUnit por medio de comandos en consola.



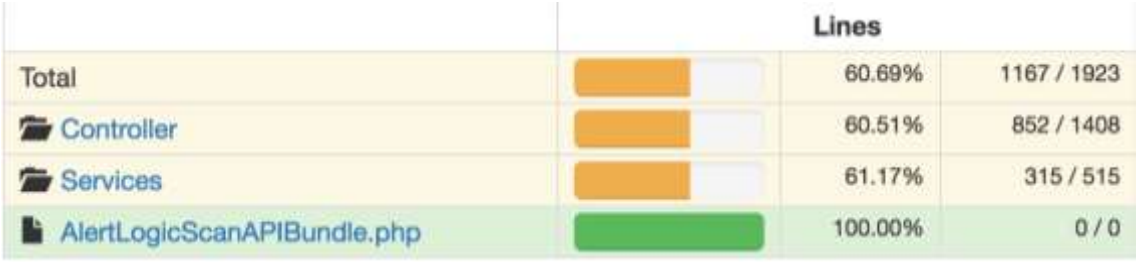
	Lines	
Total	38.55%	705 / 1829
Controller	44.09%	533 / 1209
Services	27.74%	172 / 620
AlertLogicScanAPIBundle.php	100.00%	0 / 0

Figura 3.2: Reporte de cobertura al inicio de la pasantía de ScanAPIBundle.
Fuente: elaboración propia.

	Lines	
Total	1.75%	98 / 5589
Controller	8.34%	93 / 1115
EventListener	0.00%	0 / 65
Form	0.00%	0 / 146
Security	0.00%	0 / 436
Services	0.13%	5 / 3743
Twig	0.00%	0 / 57
Utilities	0.00%	0 / 20
AlertLogicCoreBundle.php	0.00%	0 / 7

Figura 3.3: Reporte de cobertura al inicio de la pasantía de CoreBundle.
Fuente: Elaboración propia.

Al finalizar el presente objetivo, los resultados obtenidos fueron los siguientes: para el caso de ScanAPIBundle la cobertura alcanzó un 60.69% mientras que CoreBundle logró una cobertura de 63.02%. Las figuras 3.4 y 3.5 muestran el reporte de cobertura de cada Bundle.



	Lines		
Total		60.69%	1167 / 1923
Controller		60.51%	852 / 1408
Services		61.17%	315 / 515
AlertLogicScanAPIBundle.php		100.00%	0 / 0

Figura 3.4: reporte de cobertura al final de la pasantía de ScanAPIBundle.
Fuente: Elaboración propia.

/var/alertlogic/ui/src/AlertLogic / CoreBundle / (Dashboard)

	Lines		
Total		63.02%	2901 / 4603

Figura 3.5: reporte de cobertura al final de la pasantía de CoreBundle.
Fuente: Elaboración propia.

Tal como se muestra en las imágenes, el objetivo inicial fue alcanzado logrando un gran impacto en el equipo de *User Interface* debido a que cualquier cambio en el código de alguno de los Bundles trabajados implicaba mantener la integridad de las pruebas unitarias es decir, si se añadía código al Bundle debía ser probado para evitar que la cobertura descendiera o que las pruebas fallaran.

Para más información sobre el cumplimiento del presente objetivo, ver el anexo (Anexos_Tesis/Anexos_Objetivo2) en el cual se encuentran documentadas las presentaciones hechas por el "equipo de desarrollo" al final de cada Sprint.

3.1.3 Desarrollo Objetivo específico III.

El desarrollo de este objetivo *(Implementar soluciones a defectos en los productos Cloud Defender y Cloud Insight reportados por los clientes de Alert Logic.)* consintió en trabajar sobre los defectos reportados por los clientes de la empresa. Es decir, participar activamente en el proceso de resolución de casos para dar solución a las fallas presentadas por el producto, estas fallas consisten en "bugs" que los clientes encuentran sobre la interfaz de la aplicación tales como: fallas en la generación de reportes, indexación de tablas, problemas en los filtros de búsqueda y demás. Para la solución de defectos, la empresa sigue un proceso el cual se explica a continuación:

- Customer service: como primera instancia, los clientes cuentan con canales de comunicación que les permite reportar fallas del producto o recibir instrucciones sobre el mismo, estos pueden ser por vía telefónica, email o por chat. Todas estas solicitudes son atendidas por un equipo llamado *Technical Operations Center (TOC)* El cual se encarga de recibir las solicitudes de los clientes y dar solución al problema, dicha solución es llamada *Work Around* debido a que está al alcance del cliente y no requiere modificar el código de la aplicación. Cuando el equipo de *TOC* determina que el problema del cliente debe ser resuelto desde el código, el caso debe ser escalado a otro nivel en la jerarquía gestión de casos.
- Virtual Triage Team (VTT): una vez se determina que el caso no puede ser resuelto desde la interfaz de usuario, este pasa al equipo de VTT el cual se compone de todos los representantes del área de ingeniería entre los que se encuentran: recolección y acceso de datos, servicios de incidentes, interfaz de usuario y experiencia de usuario, infraestructura de automatización y arquitectura.
- User Interface: una vez el caso es estudiado por el equipo que componen el área de ingeniería y es asignado al equipo de *User Interface* este pasa a convertirse formalmente en un defecto a resolver, cabe resaltar que toda la información relacionada con el caso se documenta en una historia de usuario en Rally y pasa a ser adicionada al *Sprint Backlog*.

- Coding: en este paso, el desarrollador se encarga de leer la documentación correspondiente del caso y pasa a reproducir el problema en el ambiente de desarrollo del equipo. A continuación, el programador soluciona el problema y mezcla su código en el repositorio de la empresa a espera de la instalación final.
- Release: al final del *Sprint* el equipo de *User Interface* se encarga de instalar los cambios realizados por todos los miembros en los servidores donde se encuentra la aplicación, los cuales se encuentran en United Kingdom, Denver y Ashburn.

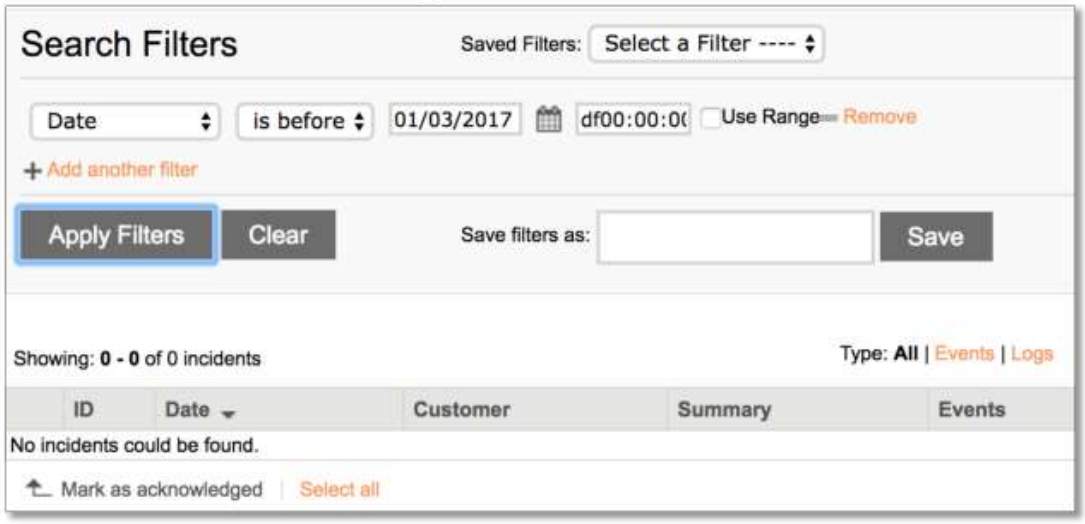
A continuación en la figura 3.6 se muestra la documentación de uno de los defectos trabajados durante la pasantía, este consiste en una descripción del defecto (la cual se toma desde Rally), la solución aplicada al defecto y las capturas de pantalla mostrando el antes-después.

DE47651: As Altair Advisers I want the date filters to work as expected with single digits for days.

Description:
Customer is experiencing unreliable results when filtering by date and single digits are entered for the day value. A validation or conversion must be developed for this filter.

Solution:
A validation function was added to "webroot/js/monitor.js" to check the date and time frame for the Date filter.

Behavior Before Fix:
There is not a validation for wrong dates.



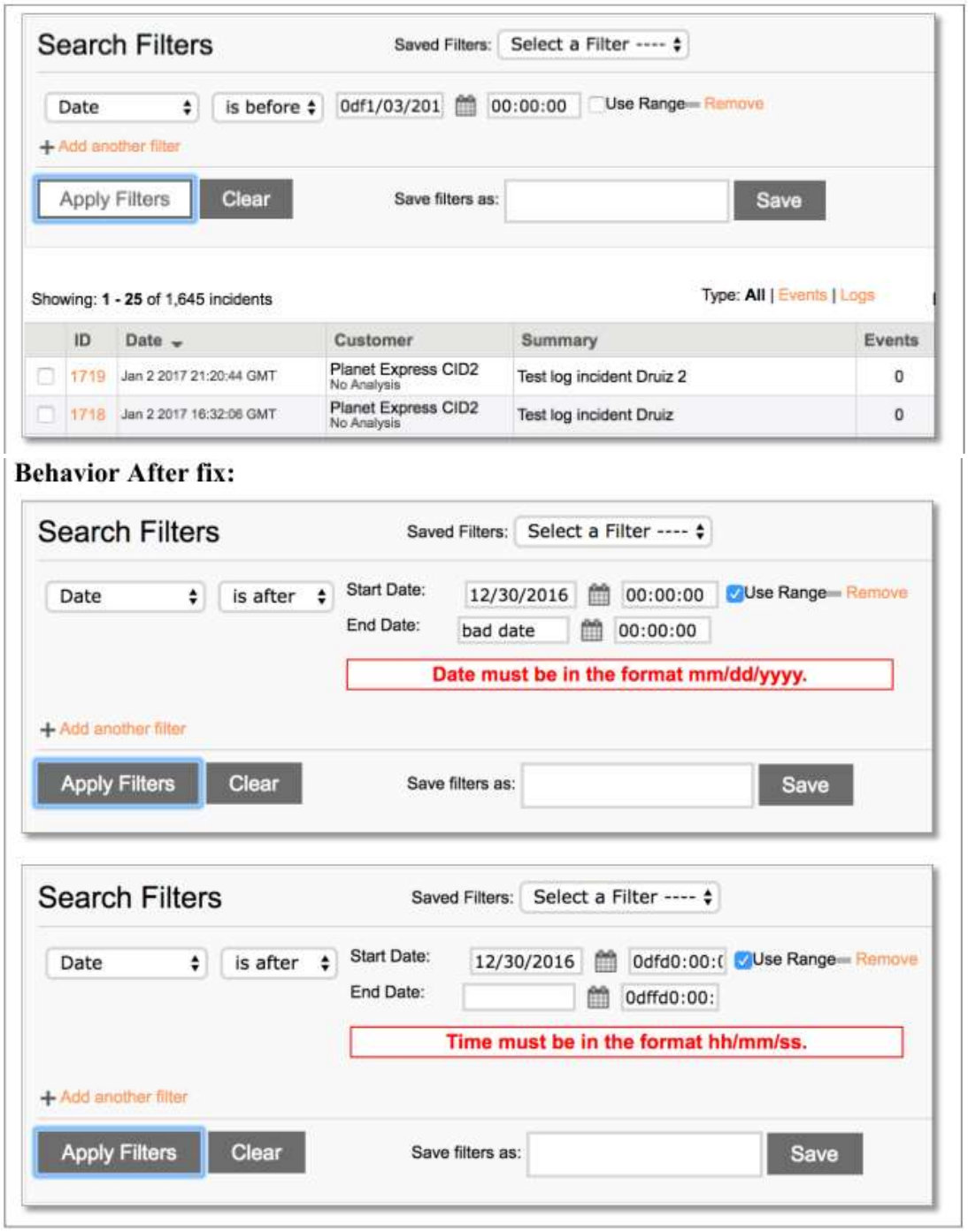


Figura 3.6: Defecto DE47651 Función de validación de fechas en Cloud Defender

Fuente: Elaboración propia.

Esta documentación se adjunta a Rally para dar constancia de que el defecto fue solucionado y revisado por el equipo de *User Interface* dando por terminado el caso. Durante la pasantía se logró trabajar en seis defectos los cuales se encuentran documentados en el anexo (Anexos_Tesis/Anexos_Objetivo3).

3.1.4 Desarrollo Objetivo específico IV.

Para el desarrollo del objetivo específico cuatro *(Implementar funcionalidades específicas en los productos Cloud Defender y Cloud Insight utilizando la tecnología AngularJS.)*, el "equipo de desarrollo" participó en el proyecto *Dunkirk*, este, es un proyecto del equipo de UI que consiste en migrar la interfaz gráfica de usuario de Cloud Defender de PHP Symfony a AngularJS, esto; con el fin de mejorar la experiencia de usuario y establecer convergencia con Cloud Insight el cual encuentra actualmente implementado en AngularJS (por esta razón no se trabajará sobre él durante el proyecto).

Cloud Defender se compone de sub-productos entre los que se destacan *Thread Manager*, *Incidents Monitor y Log Manager*. Este último fue seleccionado para la primera etapa del proyecto y sobre él se desarrolló el cuarto objetivo de la pasantía. En la Figura 4.1 se muestra la interfaz gráfica de Log Manager actualmente.

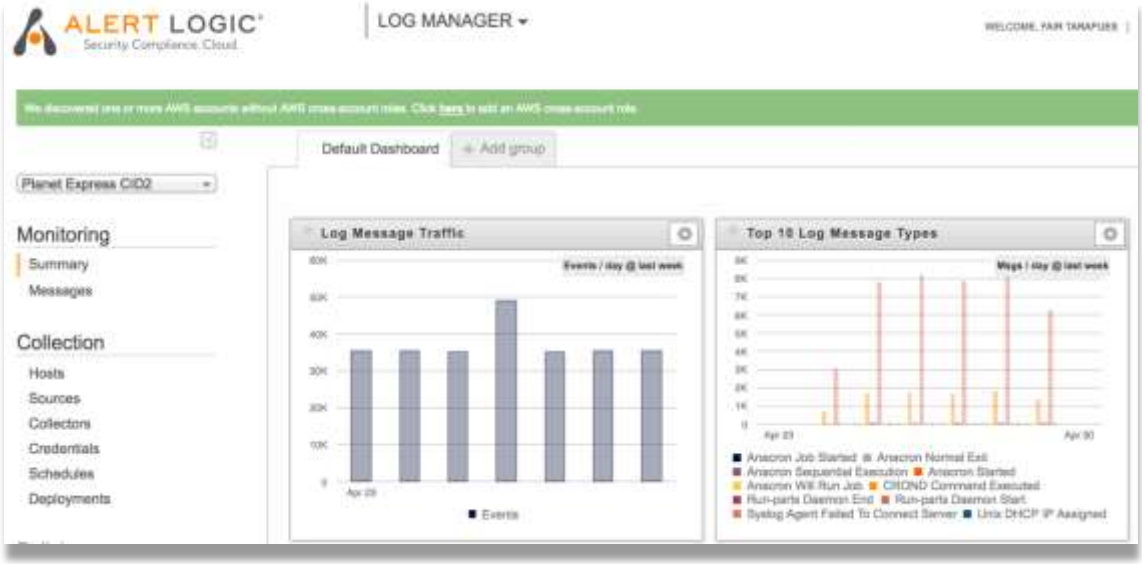


Figura 4.1: Interfaz gráfica de Log Manager actualmente en Cloud Defender
Fuente: Elaboración propia.

Para el desarrollo del proyecto Dunkirk fue necesario la incursión del equipo de *User Experience* el cual se encargó de crear los diseños de las vistas y verificar que la experiencia del usuario no fuera afectada por la implementación de la nueva interfaz, el diseño de *Log Manager* versión 2 se muestra en la Figura 4.2.

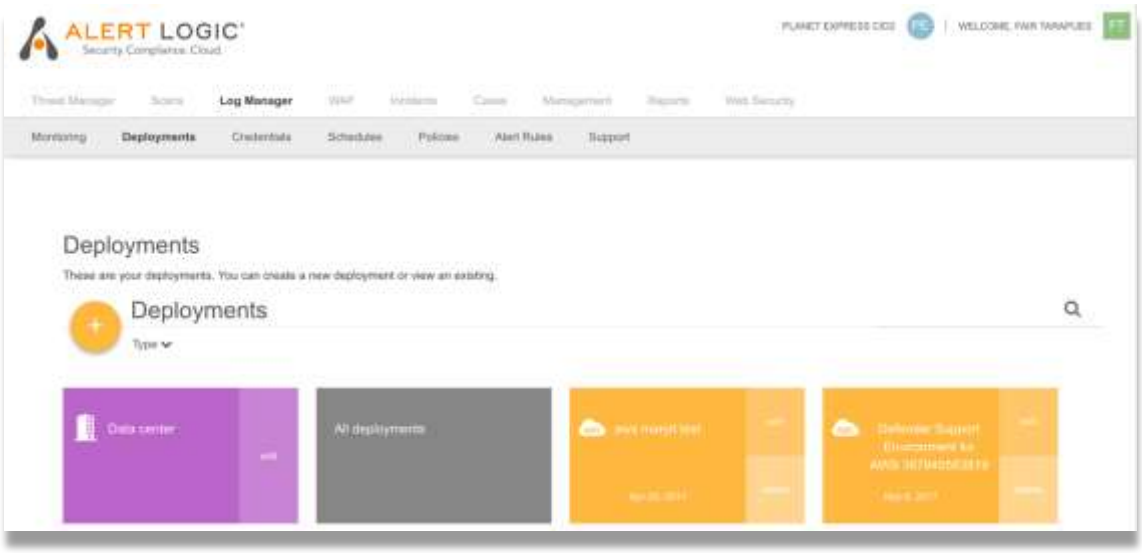


Figura 4.2.Diseño de la Interfaz Gráfica de Log Manager v2 implementada en AngularJS
Fuente: Elaboración propia.

Durante el desarrollo del presente objetivo se trabajaron tres tareas las cuales fueron:

1. Implementar los formularios web del módulo *Credentials* en Log Manager:

   Al inicio del proyecto Dunkirk los managers del equipo de *User Interface*, Jose Reina, Pablo Vélez y Diego Ruiz establecieron la distribución de las tareas entre los demás miembros del equipo, dando como resultado la asignación de la presente tarea. La figura 4.3 muestra el estado actual del formulario de *Credentials* en Log Manager.

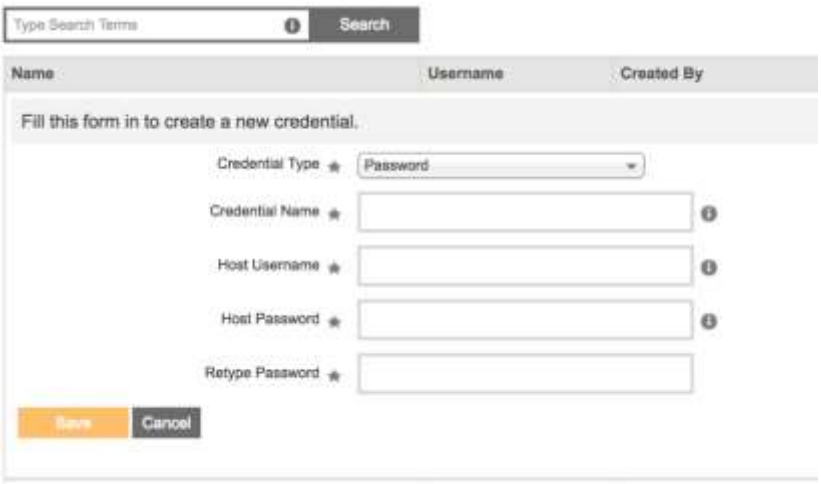


Figura 4.3. Formulario actual de Password type credentials en Log Manager
Fuente: Elaboración propia

El objetivo principal fue implementar el formulario de la figura 4.3 en AngularJS basándose en los diseños entregados por el equipo de *User Experience.* El resultado se muestra en la figura 4.4.

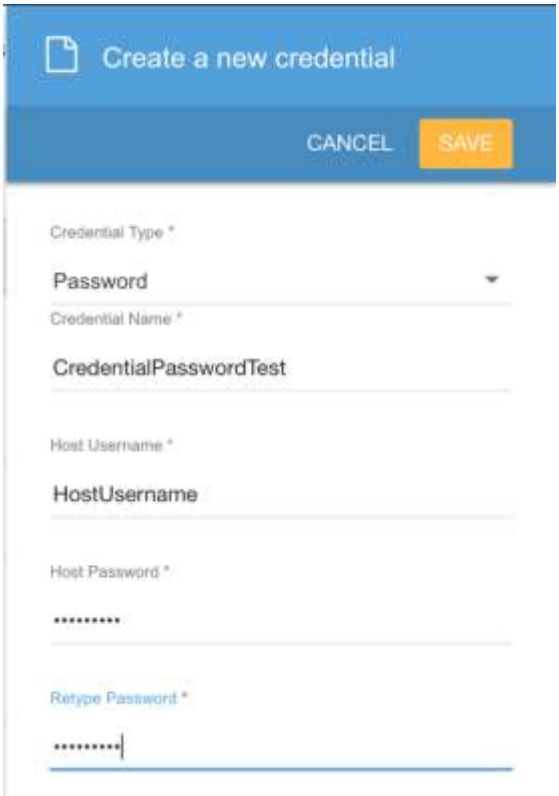


Figura 4.4 Formulario de Password type en Credentials en Dunkirk
Fuente: Elaboración propia.

2. Implementar una API desde el ambiente de desarrollo para el módulo *Schedules* en Log Manager:

Durante el desarrollo del proyecto Dunkirk, el equipo de *User Interface* se vio en la necesidad de implementar un Bundle de Symfony llamado *ConvergenceAPIBundle* en el cual se encargó del enrutamiento, las API's de *Log Manager* versión 2 y el intercambio entre *Log Manager* versión 1 y 2. Sobre este Bundle se desarrolló el presente ítem el cual tuvo como objetivo la implementación de un API para la programación de reportes. Los resultados se muestran en las figuras 4.5 y 4.6 en las cuales se puede observar que los datos del ambiente de desarrollo coinciden con los datos mostrados en la interfaz de *Schedules* de Log Manager V2.

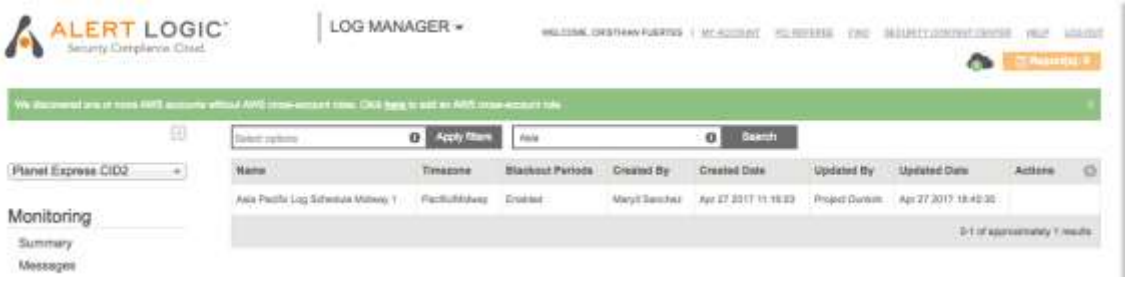


Figura 4.5 Datos del API de Schedules en el ambiente de producción
Fuente: Elaboración propia.

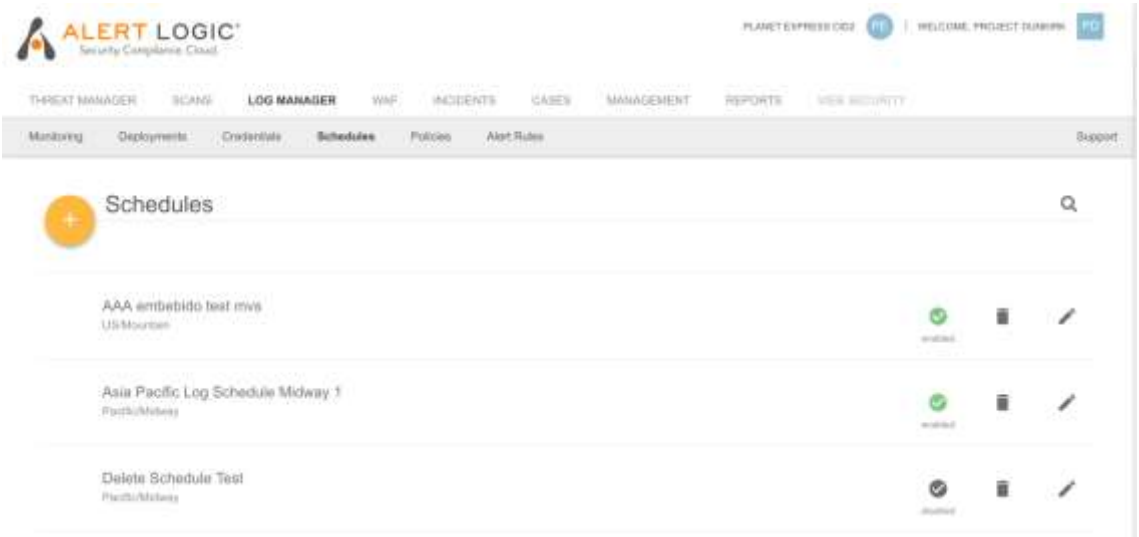


Figura 4.6 Datos del API de Schedules en la interfaz de usuario en Dunkirk
Fuente: Elaboración propia

3. Implementar el controlador y los formularios del sub-módulo *Updates* en el módulo *Policies* de Log Manager:

   Al final de la pasantía, la tarea asignada fue implementar el controlador y la vista del sub-módulo *Updates* en el módulo *Policies* de Log Manager. La imagen 4.7 muestra el estado actual del sub-modulo.

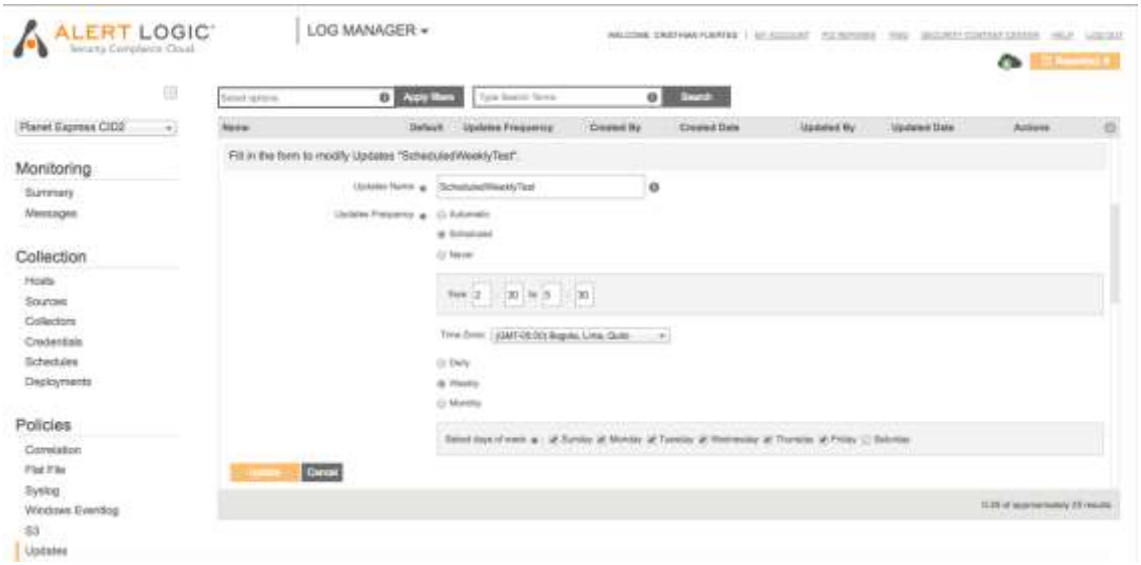


Figura 4.7 Interfaz gráfica del sub-módulo Updates en Log Manager versión 1
Fuente: Elaboración propia.

Al finalizar el presente ítem se logró migrar toda la interfaz de *Updates* a AngularJS tal como lo muestra la imagen 4.8. (Para más información ver Anexos_Tesis/Anexos_Objetivo4).

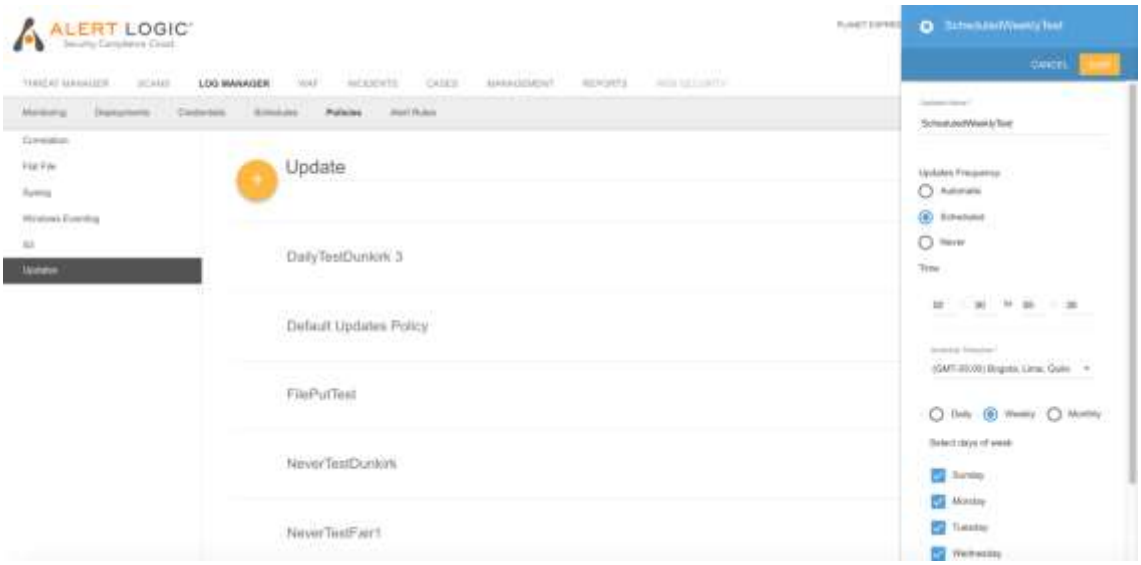


Figura 4.8 Interfaz gráfica del sub-módulo Updates en Log Manager versión 2
Fuente: Elaboración propia

Capítulo 4

Conclusiones

Como parte del cumplimento de los objetivos planteados en el presente trabajo es importante mencionar el papel que cumple la implementación de pruebas unitarias en el desarrollo de software a gran escala, ya que durante la pasantía, el “equipo de desarrollo” pasó dificultades en la implementación de algunas pruebas unitarias debido a que el código originalmente no fue programado pensando en las pruebas unitarias lo cual lleva a que la declaración de variables e instanciación de objetos quede a merced del programador generando problemas al diseñar pruebas unitarias.

Otro aspecto importante en la organización de flujo de trabajo de los equipos de desarrollo es la implementación de un control de versiones estable y seguro como Git, por medio de este, el trabajo de cada miembro del equipo se mantiene aislado y permite un mayor control del código que se mezcla en cada repositorio.

Por último, Cada una de las tareas realizadas durante la pasantía permitió al “equipo de desarrollo” conocer un ambiente trabajo real e involucrarse en cada uno de los procesos del equipo de *User Interface*, tales como reuniones, debates, trabajo en conjunto con otros miembros y demás procesos que ayudaron al crecimiento como profesionales del área de la informática a cada uno de los miembros del equipo.

Capítulo 5

Trabajos futuros

Durante el desarrollo de la pasantía se identificaron algunos aspectos los cuales pueden trabajarse en el futuro. En primer lugar, las pruebas unitarias no se completaron en todos los Bundles del producto debido a la cantidad de líneas de código que posee, de lograrse esto, el equipo podría solicitar al comité de cambios la pre-aprobación para realizar actualizaciones sobre el producto lo cual agilizaría el proceso de instalación y documentación durante los sprints. En segundo lugar, para el caso de los defectos; el equipo de *User interface* necesita personal para estudiar y reparar los casos que llegan al sprint backlog. Actualmente, la compañía cuenta con cerca de 4000 clientes lo cual hace que el proceso de resolver casos sea de suma importancia para el equipo.

6. REFERENCIAS

[1]. Alert Logic – About-us [Online]. Available: https://www.alertlogic.com/solutions/

[2]. Microsoft - Architecture Strategies for Catching the Long Tail. us [Online]. Available: https://msdn.microsoft.com/en-us/library/aa479069.aspx

[3]. J.J. Reina Materón (Private Comunication), 2016.

[4]. Alert Logic – Solutions [Online]. Available: https://www.alertlogic.com/assets/cloud-defender/AlertLogic_CloudDefender_HowItWorks.pdf

[5]. Amazon web services - ¿what is cloud computing? [Online]. Available: https://aws.amazon.com/es/what-is-cloud-computing/

[6]. Alert Logic – Solutions [Online]. Available: https://www.alertlogic.com/assets/cloud-insight/Alert-Logic-Cloud-Insight-How-it-Works-Datasheet.pdf

[7]. A. Basalo, M. Alvarez, P. Hurtado and X. Cerdá, "Tutorial de AngularJS"

[8]. Spona, H. *Programación de bases de datos con MYSQL y PHP*; Ed, Marcombo, 2010

[9]. Heurtel, O. *PHP 5.3: Desarrollar un sitio Web dinámico e interactivo*. Ed, ENI, 2011

[10]. Lopez-Pellicer, F. J., Béjar, R., Latre, M. A., Nogueras-Iso, J., & Zarazaga-Soria, F. J. (2015, July). GitHub como herramienta docente. In *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática* (pp. 66-73). Universitat Oberta La Salle.

[11] Gallego, Manuel Trigas. "Metodología Scrum." *Gestión de Proyectos Informaticos,* 2012 [Online]. Available: *http://openaccess. uoc. edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria. Pdf*.

[12] BERGMANN, Sebastian. PHPUnit Manual. [Online]. Available: *http://www. phpunit. de/pocket_guide/3.2/en/index. html, своб*, 2005.

[13] Rally Software About us [Online]. Available: http://www.ca.com/us/company/newsroom/press-releases/2015/ca-technologies-agrees-to-acquire-rally-software.html.

ALERT LOGIC®

ALERT LOGIC INC. COLOMBIA S.A.S
NIT 900.666.093-8

CERTIFICA QUE

Fair Andres Tarapues, identificado con cédula de ciudadanía No. 1.113.655.833 de Palmira, estudiante del programa de Ingeniería de Sistemas de la Universidad del Valle, recibió capacitación en los siguientes temas:

- Entrenamiento en los productos Cloud Defender y Cloud Insight. Realizado por el Ingeniero Rowinson Gallego, del 6 al 15 de Julio de 2016
- Entrenamiento en Git y GitHub. Realizado por el Ingeniero Diego Ruiz, del 18 al 22 de Julio de 2016
- Entrenamiento en PHPUnit y Jasmine para la implementación de pruebas unitarias en Cloud Defender y Cloud Insight. Realizado por los Ingenieros Jiovanny Ibargüen y Gisler Garcés, del 25 al 29 de Julio de 2016

Este certificado se expide a petición de la Coordinación del programa de Ingeniería de Sistemas de la Universidad del Valle sede Tuluá, a los 4 días del mes de Mayo de 2017.

ALERTLOGIC
Security. Compliance. Cloud.

JOSE JULIAN REINA MATERON
Gerente de Ingeniería
Alert Logic Inc. Colombia.

Alert Logic, Inc.

Calle 13A No. 100-35, Oficina 1001. Cali, Valle del Cauca | (57)2 3735111 (Conmutador) | www.alertlogic.com

Unit test for CoreBundle - Demo

UI Interns Team

Cloud Defender

November 1, 2016

ALERT LOGIC

Fixing warnings and errors in LogManagerBundle

Before

ERRORS!

Tests: 207, Assertions: 296, Errors: 36, Failures: 8, Warnings: 50, Skipped: 1.

	Lines		
Total		9.50%	1602 / 16857

After

OK, but incomplete, skipped, or risky tests!

Tests: 206, Assertions: 582, Skipped: 9.

	Lines		
Total		17.16%	2889 / 16832

ALERT LOGIC

Comparison coverage CoreBundle

Before

/var/alertlogic/ui/src/AlertLogic / CoreBundle / (Dashboard)

	Lines		
Total		50.65%	2422 / 4782

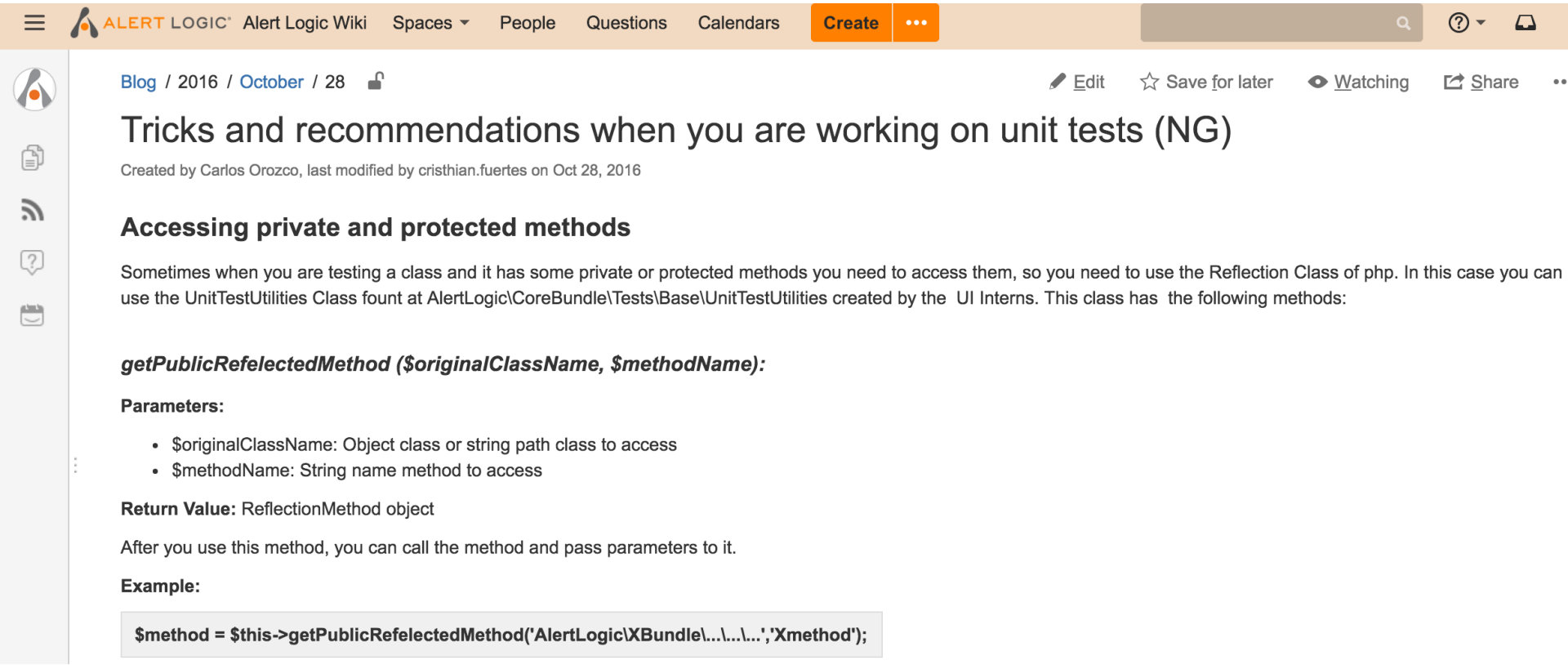
After

/var/alertlogic/ui/src/AlertLogic / CoreBundle / (Dashboard)

	Lines		
Total		63.02%	2901 / 4603

ALERT LOGIC

Guide Line about Unit Test - Wiki



https://alertlogic.atlassian.net/wiki/pages/viewpage.action?pageId=171508226

ALERT LOGIC

Skeleton template for unit test classes

- The unit-test-skelgen command created on robo allows generate a basic template to make unit test for a class in NG only.

```
robo-cloud-defender-ui (master) $ robo unit-test-skelgen
?  Repository's name (CoreBundle, ScanAPIBundle, etc.,): CoreBundle
?  Relative path to input file ../CoreBundle/: Controller/SignUpController.php
 [ExecStack] Executing phpunit-skelgen-1.2.0.phar --bootstrap /var/alertlogic/ui/app/autoload.php --test -- "AlertLogic\CoreBundle\Controller\SignUpController" /var/alertlogic/ui/src/AlertLogic/CoreBundle/Controller/SignUpController.php SignUpControllerTest /var/alertlogic/ui/src/AlertLogic/CoreBundle/Tests/Unit/Controller/SignUpControllerTest.php
PHPUnit Skeleton Generator 1.2.0 by Sebastian Bergmann.

Wrote skeleton for "SignUpControllerTest" to "/var/alertlogic/ui/src/AlertLogic/CoreBundle/Tests/Unit/Controller/SignUpControllerTest.php".
 [ExecStack] Done in 0.145s
➜
➜  Skeleton test class created
➜
```

SignUpControllerTest.php

```php
<?php
/**
 * Generated by PHPUnit_SkeletonGenerator 1.2.0 on 2016-10-31 at 15:01:17.
 */
class SignUpControllerTest extends PHPUnit_Framework_TestCase
{
    /**
     * @var SignUpController
     */
    protected $object;

    /**
     * Sets up the fixture, for example, opens a network connection.
     * This method is called before a test is executed.
     */
    protected function setUp()
    {
        $this->object = new SignUpController;
    }
```

ALERT LOGIC

Thank you.

SCAN API FIXED UNIT TEST DEMO

Cloud Defender UI Interns Team

Wednesday, Aug 24th 2016

ALERT LOGIC

Scan API Bundle Unit Tests deprecated warnings

The php-unit deprecated function getMock() was replaced by getMockBuilder()

```
There was 1 warning:

1) AlertLogic\ScanAPIBundle\Tests\Controller\BaseScanAPIControllerTest::testEvaluatePreflightAction
PHPUnit_Framework_TestCase::getMock() is deprecated, use PHPUnit_Framework_TestCase::createMock() or P
HPUnit_Framework_TestCase::getMockBuilder() instead

WARNINGS!
Tests: 1, Assertions: 1, Warnings: 1.
```

```
ui $ phpunit -c app --filter testEvaluatePreflightAction src/AlertLogic/ScanAPIBundle/Tests/Unit/Contr
oller/BaseScanAPIControllerTest.php
PHPUnit 5.4.8 by Sebastian Bergmann and contributors.

.                                                                   1 / 1 (100%)

Time: 259 ms, Memory: 15.50MB

OK (1 test, 1 assertion)
```

Stories solved

DE5290
DE5292
DE5293
DE5294
DE5296
DE5297
DE5299
DE5300
DE5302

ALERT LOGIC

Scan API Bundle Unit Tests Warnings

The route of the service 'Logger' was replaced by 'Monolog\Logger' because the mock didn't recognize the location of the service

Stories solved
DE5303
DE5304

Before the fix

```
There was 1 warning:

1) AlertLogic\ScanAPIBundle\Tests\Services\Permission\PermissionCheckerTest::testCanManagerCustomer
Trying to configure method "addError" which cannot be configured because it does not exist, has not been specified, is final
, or is static

WARNINGS!
Tests: 1, Assertions: 0, Warnings: 1.
```

After the fix

```
Time: 330 ms, Memory: 15.25MB

OK (1 test, 1 assertion)
```

ALERT LOGIC

Scan API Bundle Unit Tests Warnings

The getMock() method was replaced for getMockBuilder() of PHP Unit. Replaced the @covers and @var in the header, and added a new variable for the Date data from Response Object. **DE5364**

```
There were 3 warnings:
1) AlertLogic\ScanAPIBundle\Tests\Controller\ScanAPIControllerTest::testGetScanDisputeAction
PHPUnit_Framework_TestCase::getMock() is deprecated, use PHPUnit_Framework_TestCase::createMock() or
PHPUnit_Framework_TestCase::getMockBuilder() instead
2) AlertLogic\ScanAPIBundle\Tests\Controller\UtilityScanAPIControllerTest::testSessionInfoAction
Trying to @cover or @use not existing class or interface "AlertLogic\ScanAPIBundle\Controller\ScanAPIController".
3) AlertLogic\ScanAPIBundle\Tests\Controller\UtilityScanAPIControllerTest::testUserLookupAction
Trying to @cover or @use not existing class or interface "AlertLogic\ScanAPIBundle\Controller\ScanAPIController".
WARNINGS!
Tests: 53, Assertions: 84, Warnings: 3. phpunit
```

```
Time: 1.16 minutes, Memory: 19.00MB

OK (53 tests, 84 assertions)

Generating code coverage report in HTML format ... done
```

ALERT LOGIC

Scan API Bundle Unit Tests Errors

Solution: The missing parameter was added to the constructor. The parameter is a mock of AttachmentStorage.

Stories solved

- DE5301
- DE5306
- DE5307
- DE5308

After the fix

```
Time: 163 ms, Memory: 15.25MB

OK (1 test, 2 assertions)
```

Before the fix

```
1) AlertLogic\ScanAPIBundle\Tests\Services\Datasource\AnnotationServiceTest::testGetAnnotationsList
Argument 2 passed to AlertLogic\ScanAPIBundle\Services\DataSource\AnnotationService::__construct() must be an instance of Al
ertLogic\ScanAPIBundle\Services\AttachmentStorage\AttachmentStorage, none given, called in /var/alertlogic/ui/src/AlertLogic
/ScanAPIBundle/Tests/Unit/Service/DataSource/AnnotationServiceTest.php on line 76 and defined

/var/alertlogic/ui/src/AlertLogic/ScanAPIBundle/Services/DataSource/AnnotationService.php:43
/var/alertlogic/ui/src/AlertLogic/ScanAPIBundle/Tests/Unit/Service/DataSource/AnnotationServiceTest.php:76

ERRORS!
Tests: 1, Assertions: 0, Errors: 1.
```

ALERT LOGIC

Scan API Bundle Unit Tests Errors

Before the fix

```
Failed asserting that two objects are equal.
--- Expected
+++ Actual
@@ @@
 Symfony\Component\HttpFoundation\Response Object (
     'headers' => Symfony\Component\HttpFoundation\ResponseHeaderBag
Object (...)
-    'content' => '{"annotations":
[{"annotation_id":1,"exposure_id":null,"created":
{"by":21976,"on":1450072800},"comment":"test","internal_comment":true
,"status":"accepted","attachment":
{"name":"File_test","type":"file","size":10}}]}'
+    'content' => '{"annotations":[{"annotation_id":1,"created":
{"by":21976,"on":1450072800},"comment":"test","internal_comment":true
,"status":"accepted","attachment":
{"name":"File_test","type":"file","size":10}}]}'
     'version' => '1.0'
     'statusCode' => 200
     'statusText' => 'OK'
     'charset' => null
)
```

After the fix

```
OK (1 test, 1 assertion)
```

Adding the missing parameters in the dumm request and replace the deprecated functions for newer ones.

Stories solved

DE5272

DE5305

ALERT LOGIC

Scan API Bundle Unit Tests Errors

Mock the correct methods and replace the deprecated functions for newer ones.

DE5291

Before the fix

```
AlertLogic\ScanAPIBundle\Tests\Controller\ScanAPIControllerTest::test
UpdateExposureListAction

BadMethodCallException: Method User::getCustomerId() does not exist
on this mock object

/var/alertlogic/ui/vendor/mockery/lib/Mockery/Mock.php:222
```

After the fix

```
OK (1 test, 2 assertions)
```

Coverage

Before the fixes

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		38.55%	705 / 1829		29.09%	32 / 110		28.57%	6 / 21
Controller		44.09%	533 / 1209		3.23%	1 / 31		0.00%	0 / 3
Services		27.74%	172 / 620		39.24%	31 / 79		29.41%	5 / 17
AlertLogicScanAPIBundle.php		100.00%	0 / 0		100.00%	0 / 0		100.00%	1 / 1

After the fixes

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		46.96%	849 / 1808		36.36%	40 / 110		28.57%	6 / 21
Controller		53.93%	652 / 1209		6.45%	2 / 31		0.00%	0 / 3
Services		32.89%	197 / 599		48.10%	38 / 79		29.41%	5 / 17
AlertLogicScanAPIBundle.php		100.00%	0 / 0		100.00%	0 / 0		100.00%	1 / 1

Thank you.

SCAN API BUNDLE COVERAGE GROWTH DEMO

Cloud Defender UI Interns Team

Wednesday, Sep 7th 2016

ALERT LOGIC

ScanAPIControllerTest– Setup Function.

US42267 ScanAPIControllerTest - mock objects to be reusable in order to test the ScanAPIController

```
public function setUp() {
    $this->scanAPIController = new ScanApiController();
    // Mock PermissionChecker
    $this->permissionChecker = m::mock('PermissionChecker');
    $this->permissionChecker->shouldReceive('assertUserAuthorization')->andReturn(true);
    // Mock SanitizeService
    $this->sanitizeService = m::mock('Sanitize');
    $this->sanitizeService->shouldReceive('sanitize')
                          ->times(5)
                          ->andReturn($this->customerId)
                          ->andReturn($this->status)
                          ->andReturn($this->orderBy)
                          ->andReturn($this->offset)
                          ->andReturn($this->limit);
    // Mock Customer Service
    $this->customerService = m::mock('Customer');
    $this->customerService->shouldReceive('getCustomerById')
                          ->andReturn(array('child_chain'=>array()));
}
```

Before the fix

ALERT LOGIC

ScanAPIControllerTest– Setup Function.

```
public function setUp() {
    // Mock Customer Service
    $this->customerService = $this->getMockBuilder('Customer')
                            ->setMethods(['getCustomerById',
                                          'getCustomersById',
                                          'anyChildHasAnyFeature',
                                          'updateCustomerName',
                                          'buildCustomer',
                                          'getChildCustomers',
                                          'hasFeatureEnabled',
                                          'isAWSCustomer',
                                          'getAWSSignupProduct',
                                          'checkRestrictHierarchy',
                                          'getChildChainAllowed',
                                          'getAllowedActiveCustomer'])
                            ->getMock();

    $this->scanDisputeId = 38;

    $this->navigationService = $this->getMockBuilder('Navigation')
                            ->setMethods(['getTabs','getHref'])
                            ->getMock();

    $this->disputeService = $this->getMockBuilder('DisputeService')
                           ->setMethods(['getScanDisputeByPolicyLogId',
                                         'createPciDisputeContact',
                                         'updateScanDispute',
                                         'getDisputeContactObject',
```

After the fix

ALERT LOGIC

Scan API Bundle Unit Tests Services

US43062 CommunicationService - The desired goal is to reach 60% of coverage

Before the fix

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Services / Communications / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		0.00%	0 / 175		0.00%	0 / 7		0.00%	0 / 1
CommunicationService.php		0.00%	0 / 175		0.00%	0 / 7		0.00%	0 / 1

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Services / Communications / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		38.24%	39 / 102		42.86%	3 / 7		0.00%	0 / 1
CommunicationService.php		38.24%	39 / 102		42.86%	3 / 7		0.00%	0 / 1

After the fix

ALERT LOGIC

Scan API Bundle Unit Tests Services

US43059 AttachmentStorage Service - The desired goal is to reach 60% of coverage

Before the fix

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Services / AttachmentStorage / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		0.00%	0 / 60		0.00%	0 / 8		0.00%	0 / 1
AttachmentStorage.php		0.00%	0 / 60		0.00%	0 / 8		0.00%	0 / 1

Legend

Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

Generated by php-code-coverage 4.0.1 using PHP 5.6.23 with Xdebug 2.4.0 and PHPUnit 5.4.8 at Mon Aug 29 15:41:31 CDT 2016.

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Services / AttachmentStorage / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		100.00%	22 / 22		100.00%	7 / 7		100.00%	1 / 1
AttachmentStorage.php		100.00%	22 / 22		100.00%	7 / 7		100.00%	1 / 1

Legend

Low: 0% to 50% Medium: 50% to 90% High: 90% to 100%

Generated by php-code-coverage 4.0.1 using PHP 5.6.23 with Xdebug 2.4.0 and PHPUnit 5.4.8 at Mon Aug 29 15:57:25 CDT 2016.

After the fix

ALERT LOGIC

Scan API Bundle Unit Tests Services

US43064 PermissionChecker - The desired goal is to reach 60% of coverage

Before the fix

	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total		0.00%	0 / 1		40.00%	2 / 5	CRAP		15.62%	10 / 64
PermissionChecker		0.00%	0 / 1		40.00%	2 / 5	340.76		15.62%	10 / 64
__construct					100.00%	1 / 1	1		100.00%	6 / 6
assertUserAuthorization					0.00%	0 / 1	210		0.00%	0 / 32
actingUserHasRole					100.00%	1 / 1	1		100.00%	1 / 1
customerHasFeature					0.00%	0 / 1	6		0.00%	0 / 7
canManageCustomer					0.00%	0 / 1	19.47		16.67%	3 / 18

	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total		0.00%	0 / 1		60.00%	3 / 5	CRAP		65.62%	42 / 64
PermissionChecker		0.00%	0 / 1		60.00%	3 / 5	44.49		65.62%	42 / 64
__construct					100.00%	1 / 1	1		100.00%	6 / 6
assertUserAuthorization					0.00%	0 / 1	61.85		37.50%	12 / 32
actingUserHasRole					100.00%	1 / 1	1		100.00%	1 / 1
customerHasFeature					100.00%	1 / 1	2		100.00%	7 / 7
canManageCustomer					0.00%	0 / 1	5.03		88.89%	16 / 18

After the fix

ALERT LOGIC

Scan API Bundle Unit Tests Services

US43067 AnnotationService - The desired goal is to reach 60% of coverage

Before the fix

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total	0.00%	0 / 1		12.50%	1 / 8	CRAP	14.29%	3 / 21	
AnnotationService	0.00%	0 / 1		12.50%	1 / 8	48.30	14.29%	3 / 21	
__construct				100.00%	1 / 1	1	100.00%	3 / 3	
getAnnotationsList				0.00%	0 / 1	2	0.00%	0 / 3	
getAnnotationsHistory				0.00%	0 / 1	2	0.00%	0 / 2	
repealAnnotation				0.00%	0 / 1	2	0.00%	0 / 2	
getAnnotationSummary				0.00%	0 / 1	2	0.00%	0 / 3	
addAnnotation				0.00%	0 / 1	2	0.00%	0 / 3	
getAnnotationByIdentity				0.00%	0 / 1	2	0.00%	0 / 3	
claimUnassignedAnnotations				0.00%	0 / 1	2	0.00%	0 / 2	

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		100.00%	1 / 1		100.00%	8 / 8	CRAP	100.00%	21 / 21
AnnotationService		100.00%	1 / 1		100.00%	8 / 8	8	100.00%	21 / 21
__construct					100.00%	1 / 1	1	100.00%	3 / 3
getAnnotationsList					100.00%	1 / 1	1	100.00%	3 / 3
getAnnotationsHistory					100.00%	1 / 1	1	100.00%	2 / 2
repealAnnotation					100.00%	1 / 1	1	100.00%	2 / 2
getAnnotationSummary					100.00%	1 / 1	1	100.00%	3 / 3
addAnnotation					100.00%	1 / 1	1	100.00%	3 / 3
getAnnotationByIdentity					100.00%	1 / 1	1	100.00%	3 / 3
claimUnassignedAnnotations					100.00%	1 / 1	1	100.00%	2 / 2

After the fix

ALERT LOGIC

ScanAPIBundle Unit Tests DisputeEvidenceController

Before the fix

US43467
US43468
US43465

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Controller / DisputeEvidenceController.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		0.00%	0 / 6	CRAP	0.00%	0 / 206
DisputeEvidenceController		0.00%	0 / 1		0.00%	0 / 6	1332	0.00%	0 / 206
evaluatePreflightAction					0.00%	0 / 1	2	0.00%	0 / 7
getDisputeEvidenceListAction					0.00%	0 / 1	110	0.00%	0 / 50
addDisputeEvidenceAction					0.00%	0 / 1	110	0.00%	0 / 68
deleteDisputeEvidenceAction					0.00%	0 / 1	20	0.00%	0 / 27
getExposureIdsByScanDispute					0.00%	0 / 1	6	0.00%	0 / 9
getExposureIdsByReportOutput					0.00%	0 / 1	90	0.00%	0 / 45

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Controller / DisputeEvidenceController.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		16.67%	1 / 6	CRAP	27.17%	47 / 173
DisputeEvidenceController		0.00%	0 / 1		16.67%	1 / 6	536.70	27.17%	47 / 173
evaluatePreflightAction					100.00%	1 / 1	1	100.00%	5 / 5
getDisputeEvidenceListAction					0.00%	0 / 1	13.98	65.85%	27 / 41
addDisputeEvidenceAction					0.00%	0 / 1	110	0.00%	0 / 59
deleteDisputeEvidenceAction					0.00%	0 / 1	4.67	65.22%	15 / 23
getExposureIdsByScanDispute					0.00%	0 / 1	6	0.00%	0 / 7
getExposureIdsByReportOutput					0.00%	0 / 1	90	0.00%	0 / 38

After the fix

ALERT LOGIC

ScanAPIBundleControllerTest

US43471 getScanDisputeHistoryAction – New Unit Tests were created, The desired goal is to reach 60% of coverage

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Controller / ScanApiController.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		3.85%	1 / 26	CRAP	40.63%	453 / 1115
ScanApiController		0.00%	0 / 1		7.41%	2 / 27	10915.72	40.63%	453 / 1115
getScanDisputesListAction					0.00%	0 / 1	24.64	85.42%	82 / 96
getScanDisputeAction					0.00%	0 / 1	20.05	85.71%	72 / 84
getScanDisputeHistoryAction					0.00%	0 / 1	42	0.00%	0 / 30
deleteScanDisputeHistoryAction					0.00%	0 / 1	42	0.00%	0 / 30
updateScanDisputeAction					0.00%	0 / 1	82.31	41.67%	35 / 84
getExposureListAction					0.00%	0 / 1	12.21	88.68%	94 / 106
updateExposureListAction					0.00%	0 / 1	1056	0.00%	0 / 103
_annotateExposures					0.00%	0 / 1	240	0.00%	0 / 57
_unlinkAttachment					0.00%	0 / 1	6	0.00%	0 / 5
getExposureAnnotationAttachmentAction					0.00%	0 / 1	25.13	35.56%	16 / 45
_switchExposuresLock					0.00%	0 / 1	12	0.00%	0 / 12
_importPreviousResponses					0.00%	0 / 1	20	0.00%	0 / 13
getExposureAnnotationListAction					0.00%	0 / 1	6.16	83.67%	41 / 49
getReportSummaryAction					0.00%	0 / 1	72	0.00%	0 / 38
_getReportSummary					0.00%	0 / 1	342	0.00%	0 / 114
getUserDataAction					0.00%	0 / 1	4.11	80.77%	21 / 26
filterTabs					0.00%	0 / 1	9.45	82.35%	14 / 17

Before the fix

ALERT LOGIC

ScanAPIBundleControllerTest

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / Controller / ScanApiController.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		3.85%	1 / 26	CRAP	53.63%	598 / 1115
ScanApiController		0.00%	0 / 1		7.41%	2 / 27	5317.71	53.63%	598 / 1115
getScanDisputesListAction					0.00%	0 / 1	24.64	85.42%	82 / 96
getScanDisputeAction					0.00%	0 / 1	20.05	85.71%	72 / 84
getScanDisputeHistoryAction					0.00%	0 / 1	8.30	60.00%	18 / 30
deleteScanDisputeHistoryAction					0.00%	0 / 1	42	0.00%	0 / 30
updateScanDisputeAction					0.00%	0 / 1	82.31	41.67%	35 / 84
getExposureListAction					0.00%	0 / 1	12.21	88.68%	94 / 106
updateExposureListAction					0.00%	0 / 1	117.39	56.31%	58 / 103
_annotateExposures					0.00%	0 / 1	19.10	73.68%	42 / 57
_unlinkAttachment					0.00%	0 / 1	6	0.00%	0 / 5
getExposureAnnotationAttachmentAction					0.00%	0 / 1	25.13	35.56%	16 / 45
_switchExposuresLock					0.00%	0 / 1	12	0.00%	0 / 12
_importPreviousResponses					0.00%	0 / 1	20	0.00%	0 / 13
getExposureAnnotationListAction					0.00%	0 / 1	6.16	83.67%	41 / 49
getReportSummaryAction					0.00%	0 / 1	9.55	71.05%	27 / 38
_getReportSummary					0.00%	0 / 1	342	0.00%	0 / 114
getUserDataAction					0.00%	0 / 1	4.11	80.77%	21 / 26
filterTabs					0.00%	0 / 1	9.45	82.35%	14 / 17

After the fix

ALERT LOGIC

Coverage

Before the fixes

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		34.53%	700 / 2027		26.36%	34 / 129		26.09%	6 / 23
Controller		35.95%	518 / 1441		2.56%	1 / 39		0.00%	0 / 4
Services		31.06%	182 / 586		36.67%	33 / 90		27.78%	5 / 18
AlertLogicScanAPIBundle.php		100.00%	0 / 0		100.00%	0 / 0		100.00%	1 / 1

After the fixes

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		53.87%	1036 / 1923		42.64%	55 / 129		30.43%	7 / 23
Controller		51.21%	721 / 1408		5.13%	2 / 39		0.00%	0 / 4
Services		61.17%	315 / 515		58.89%	53 / 90		33.33%	6 / 18
AlertLogicScanAPIBundle.php		100.00%	0 / 0		100.00%	0 / 0		100.00%	1 / 1

ALERT LOGIC

Thank you.

CORE BUNDLE AND SCANAPI BUNDLE COVERAGE INCREMENT DEMO

Cloud Defender UI Interns Team

Wednesday, Sep 21th 2016

ALERT LOGIC

US43801

As AlertLogic we want the unit test for CoreBundle/Services/API/HttpRequest to be created

Before

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / API / HttpRequest.php

	Code Coverage								
	Classes and Traits			Functions and Methods				Lines	
Total		0.00%	0 / 1		0.00%	0 / 18	CRAP	0.00%	0 / 198

After

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / API / HttpRequest.php

	Code Coverage								
	Classes and Traits			Functions and Methods				Lines	
Total		0.00%	0 / 1		44.44%	8 / 18	CRAP	59.06%	88 / 149

ALERT LOGIC

US43802

As AlertLogic we want the unit test for CoreBundle/Services/API/API to be created

Before

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / API / API.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		0.00%	0 / 13	CRAP	0.00%	0 / 125

After

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / API / API.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		30.77%	4 / 13	CRAP	49.60%	62 / 125

ALERT LOGIC

US43803

As AlertLogic we want the unit test for CoreBundle/Services/API/APIService to be created

Before

APIService.php		0.00%	0 / 37		0.00%	0 / 2		0.00%	0 / 1

After

APIService.php		32.43%	12 / 37		0.00%	0 / 2		0.00%	0 / 1

ALERT LOGIC

US43806

As AlertLogic we want the unit test for CoreBundle/Services/API/APICache to be created

Before

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / API / APICache.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		0.00%	0 / 6	CRAP		0.00%	0 / 47

After

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / API / APICache.php

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		50.00%	3 / 6	CRAP		61.76%	21 / 34

ALERT LOGIC

US43807

As Alert Logic we want the unit tests EntitlementService to be created

Before

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		25.00%	1 / 4	CRAP	74.42%	32 / 43

After

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		0.00%	0 / 4	CRAP	0.00%	0 / 61

ALERT LOGIC

US43810

As AlertLogic we want the unit test for CoreBundle/Services/Datasource/SourcesService to be created

Before

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		0.00%	0 / 13 CRAP		0.00%	0 / 275

After

	Code Coverage								
	Classes and Traits			Functions and Methods			Lines		
Total		0.00%	0 / 1		15.38%	2 / 13 CRAP		60.34%	143 / 237

US43811

As AlertLogic we want the unit test for CoreBundle/Services/Datasource/StatsService to be created

Before

	Code Coverage								
	Classes and Traits			Functions and Methods				Lines	
Total		0.00%	0 / 1		0.00%	0 / 6	CRAP	0.00%	0 / 120

After

	Code Coverage								
	Classes and Traits			Functions and Methods				Lines	
Total		0.00%	0 / 1		83.33%	5 / 6	CRAP	92.94%	79 / 85

ALERT LOGIC

US43812

As AlertLogic we want the unit tests for CoreBundle/Services/Insight/CloudExplorerClientService to be created

Before

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / Insight / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		0.00%	0 / 442		0.00%	0 / 32		0.00%	0 / 7
Exception		0.00%	0 / 15		0.00%	0 / 2		0.00%	0 / 2
CloudExplorerClientService.php		0.00%	0 / 26		0.00%	0 / 2		0.00%	0 / 1

After

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		16.22%	61 / 376		21.88%	7 / 32		57.14%	4 / 7
Exception		100.00%	13 / 13		100.00%	2 / 2		100.00%	2 / 2
CloudExplorerClientService.php		100.00%	20 / 20		100.00%	2 / 2		100.00%	1 / 1

US43813

As AlertLogic we want the unit test for CoreBundle/Services/Datasource/EdgarService to be created

Before

	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total		0.00%	0 / 1		0.00%	0 / 5	CRAP		0.00%	0 / 72
EdgarService		0.00%	0 / 1		0.00%	0 / 5	272		0.00%	0 / 72

After

	Code Coverage									
	Classes and Traits			Functions and Methods				Lines		
Total		0.00%	0 / 1		60.00%	3 / 5	CRAP		32.76%	19 / 58
EdgarService		0.00%	0 / 1		60.00%	3 / 5	93.83		32.76%	19 / 58

ALERT LOGIC

US43819

As Alertlogic we want the unit test for CoreBundle/Services/Datasource/ClydeService to be created

Before

ClydeService.php		0.00%	0 / 71		0.00%	0 / 4		0.00%	0 / 1

After

ClydeService.php		98.00%	49 / 50		75.00%	3 / 4		0.00%	0 / 1

ALERT LOGIC

US43820

As AlertLogic we want the unit tests for CoreBundle/Services/Insight/StrawbossClientService to be created

Before

StrawbossClientService.php		0.00%	0 / 37		0.00%	0 / 3		0.00%	0 / 1

After

StrawbossClientService.php		100.00%	28 / 28		100.00%	3 / 3		100.00%	1 / 1

ALERT LOGIC

US43828

As AlertLogic we want the unit test for CoreBundle/Services/User and CoreBundle/Services/Insight exceptions to be created

After

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / User / (Dashboard)

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total		4.95%	18 / 364		18.18%	6 / 33		75.00%	6 / 8
Exception		100.00%	18 / 18		100.00%	6 / 6		100.00%	6 / 6
MyAccount.php		0.00%	0 / 118		0.00%	0 / 10		0.00%	0 / 1

/var/alertlogic/ui/src/AlertLogic / CoreBundle / Services / Insight / (Dashboard)

	Code Coverage							
	Lines			Functions and Methods			Classes and Traits	
Total		2.95%	13 / 440		6.25%	2 / 32		28.57%	2 / 7
Exception		100.00%	13 / 13		100.00%	2 / 2		100.00%	2 / 2

ALERT LOGIC

Comparison coverage CoreBundle

Before

/var/alertlogic/ui/src/AlertLogic / CoreBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		1.75%	98 / 5589		0.45%	2 / 446		4.04%	4 / 99
Controller		8.34%	93 / 1115		1.32%	1 / 76		0.00%	0 / 12

After

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		12.70%	682 / 5372		9.80%	44 / 449		14.00%	14 / 100
Controller		8.34%	93 / 1115		1.32%	1 / 76		0.00%	0 / 12

ALERT LOGIC

US43800

- As AlertLogic we want the Code Coverage of the ScanApiBundle to reach 60%

Before

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		53.87%	1036 / 1923		42.64%	55 / 129		30.43%	7 / 23
Controller		51.21%	721 / 1408		5.13%	2 / 39		0.00%	0 / 4
Services		61.17%	315 / 515		58.89%	53 / 90		33.33%	6 / 18
AlertLogicScanAPIBundle.php		100.00%	0 / 0		100.00%	0 / 0		100.00%	1 / 1

After

/var/alertlogic/ui/src/AlertLogic / ScanAPIBundle / (Dashboard)

	Code Coverage								
	Lines			Functions and Methods			Classes and Traits		
Total		60.69%	1167 / 1923		44.19%	57 / 129		30.43%	7 / 23
Controller		60.51%	852 / 1408		10.26%	4 / 39		0.00%	0 / 4
Services		61.17%	315 / 515		58.89%	53 / 90		33.33%	6 / 18
AlertLogicScanAPIBundle.php		100.00%	0 / 0		100.00%	0 / 0		100.00%	1 / 1

Thank you.

Date	Nov 21th 2016
Developer	Fair Andres Tarapues H.
IT Involve Link	DE5550

Fix Description (Before and After fix behavior)

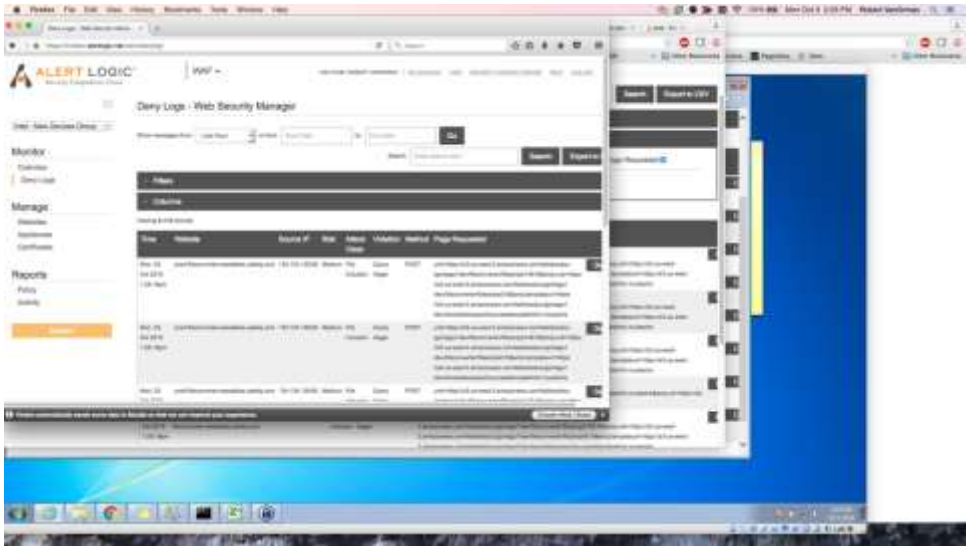
DE5550: Resizing of browser windows below a certain width causes the Detail button to disappear from the right side of the Deny Logs page on the WAF module.

Description:

When resizing the browser window on the 'Deny Logs' Page of the WAF Module, the Detail button disappears from view. When this occurs the user is unable to scroll horizontally to view the Detail button.

Monitor Resolution set to 1920x1080
The versions tested are as follows:
Firefox - 49.0.1
Chrome - 51.0.2704.103 (64-bit)
Internet Explorer - 11.0.9600.18449

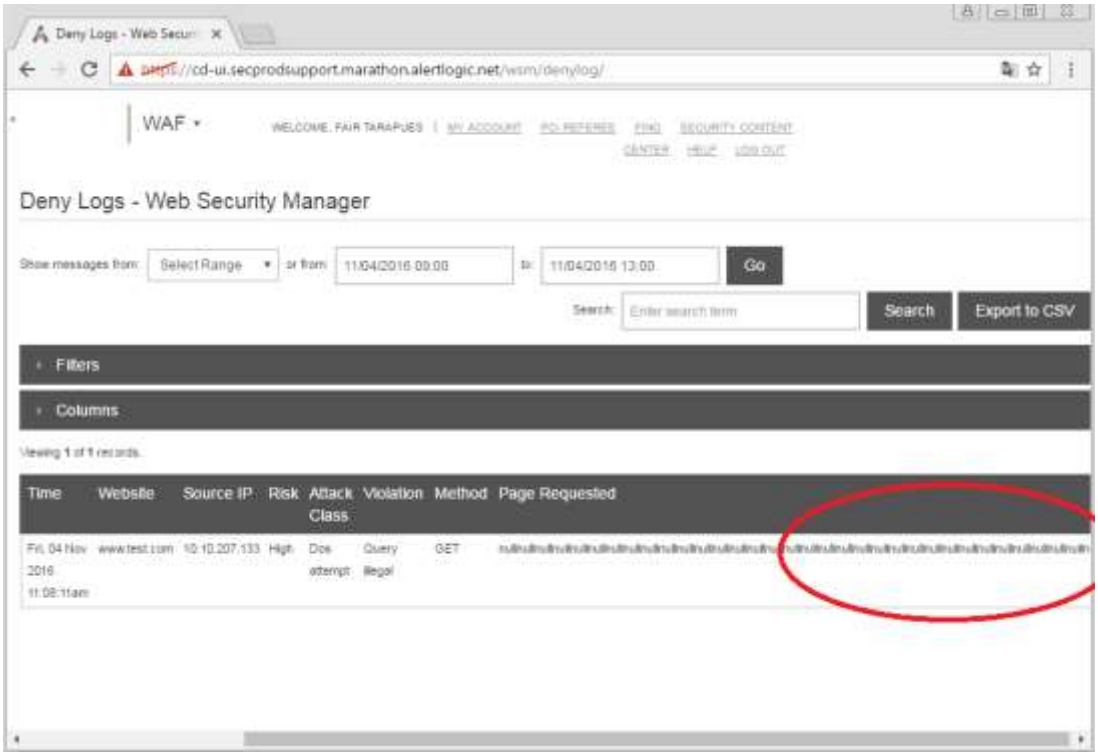
I have tested this in Internet Explorer, Chrome and Firefox (in order from Back to Front in the below picture):



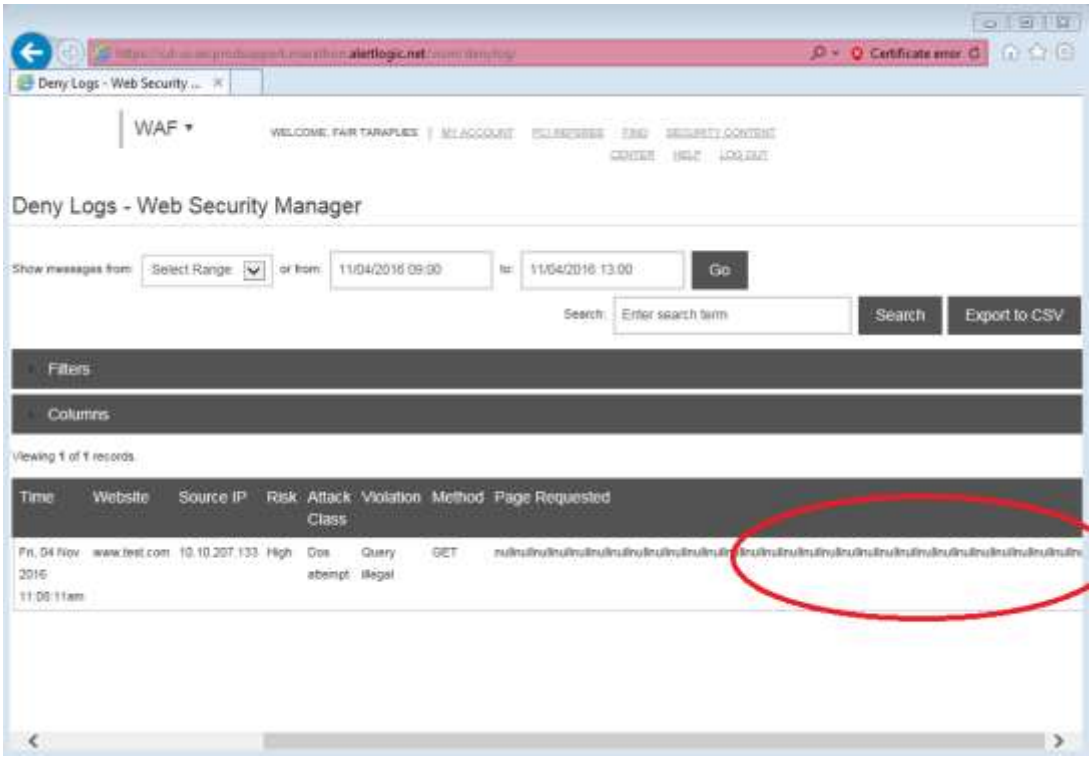
Solution: "overflow-x: auto;" property was added to the "#wsm_content_holder" style locate in Resources/public/css/wsm.css

Before the fix

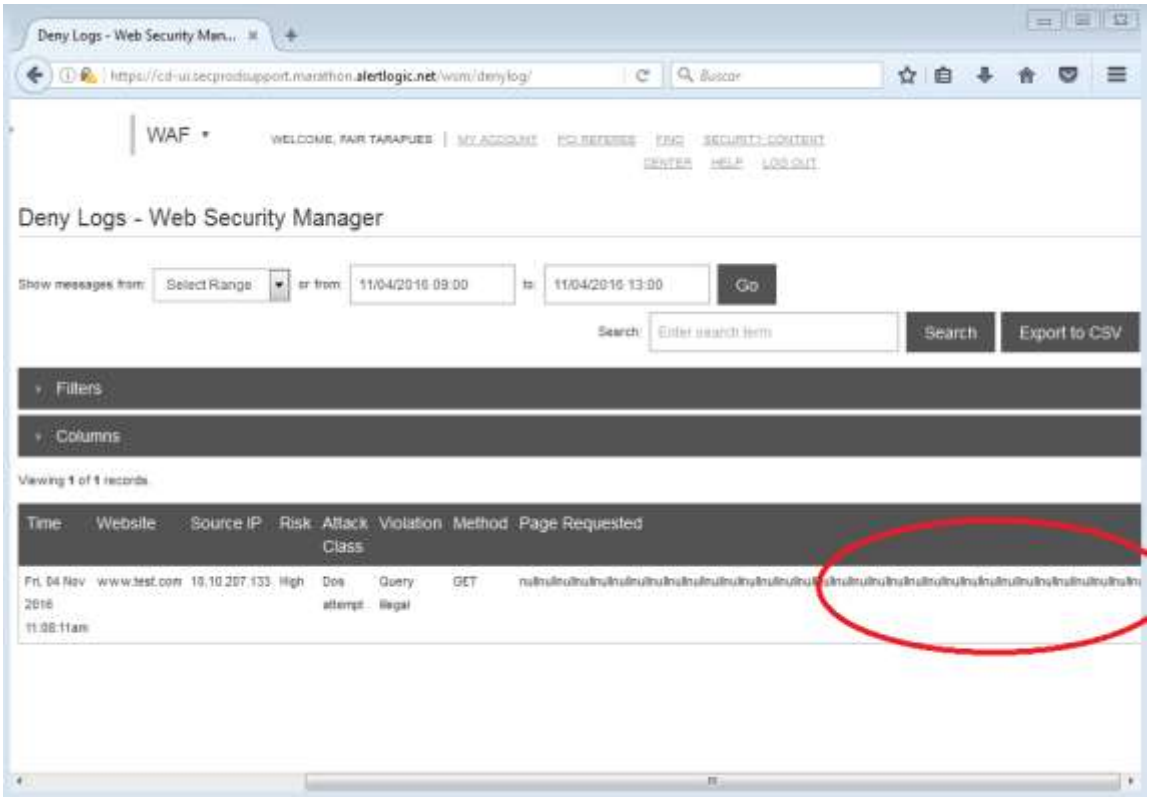
Google Chrome test.



Internet Explorer test.

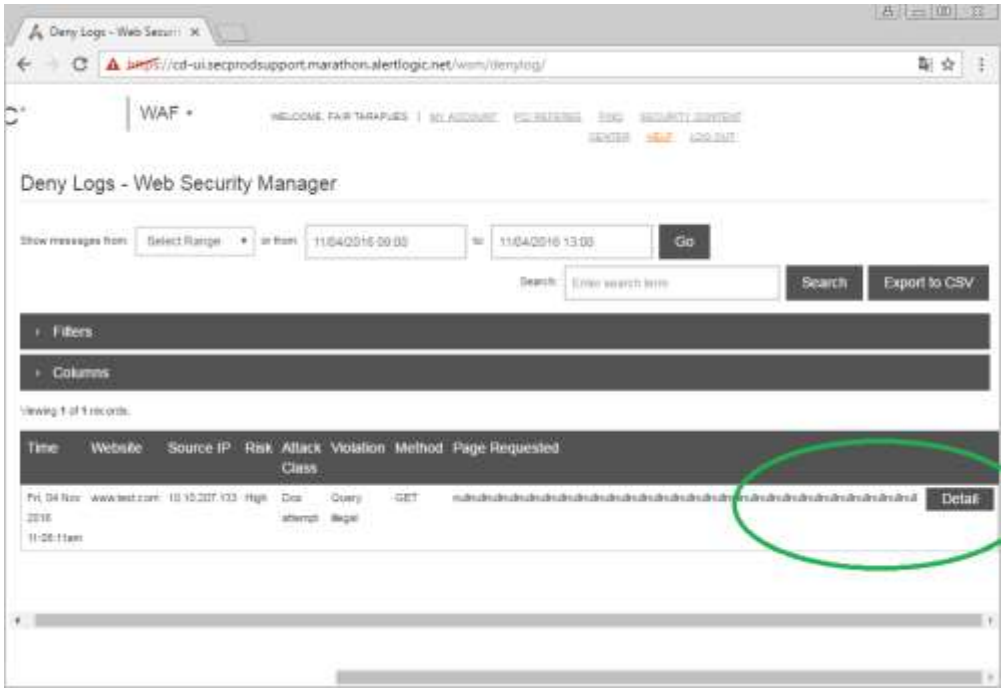


Mozilla Firefox test.

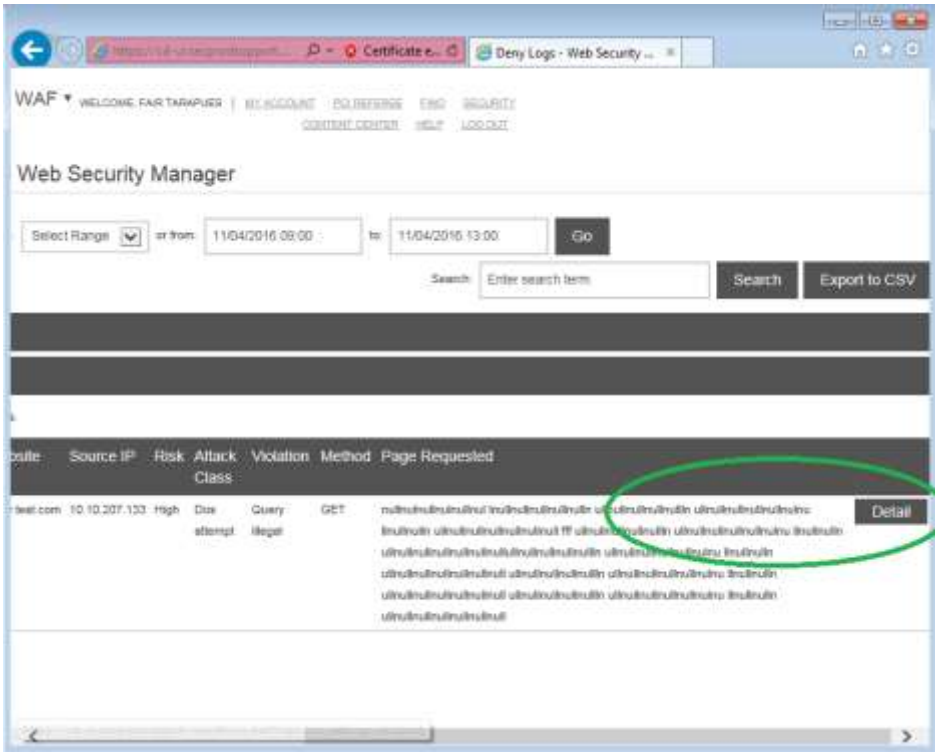


After the fix

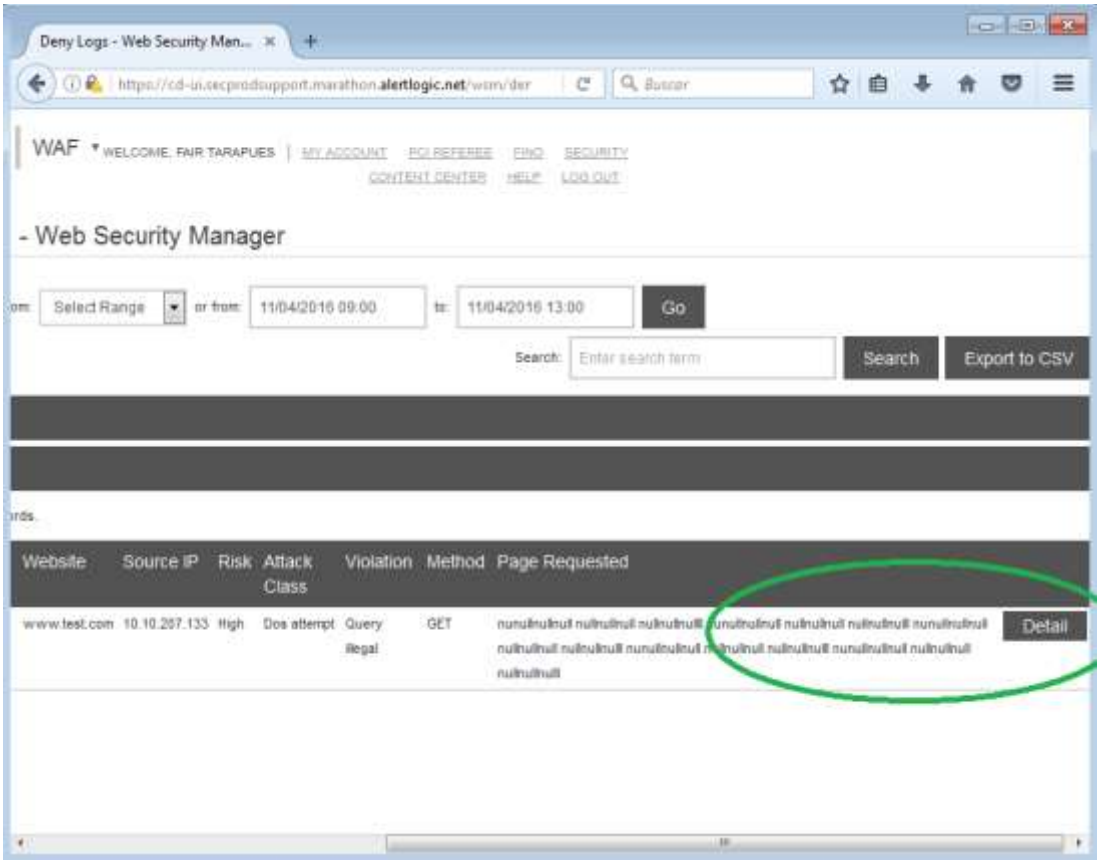
Google Chrome test.



Internet Explorer test.



Mozilla Firefox test.



Date	Jan 17th 2016
Developer	Fair Andres Tarapues H.
IT Involve Link	DE5863

Fix Description (Before and After fix behavior)

DE5863: Unable to see manage_user.php and assets menu.

Description:

User can see the manage_sensor.php page because this line
https://algithub.pd.alertlogic.net/alertlogic/php-interface/blob/master/webroot/manage_sensors.php#L51

return this

Array

(

[13645] => Anahuac National Bank

[54900] => Classic Bank

[16033] => Community National Bank of Hondo

[52910] => First Liberty National Bank

[12772] => First National Bank of Bastrop

[35694] => First State Bank of Bedias

[48365] => First State Bank of Chico

[11210] => Normangee State Bank

[14347] => Plains State Bank

[14552] => Post Oak Bank

[17110] => Security State Bank of Anahuac

[12866] => SouthStar Bank

[9728] => State Bank of Texas

[11088] => The Hondo National Bank

)

when the only customers that should return should be

Classic Bank
First State Bank of Bedias
First State Bank of Chico
State Bank of Texas

for that reason this function

https://algithub.pd.alertlogic.net/alertlogic/php-interface/blob/master/webroot/manage_sensors.php#L67

returns false and it will show the error

print_r(User_Permission::isAllowedForThisCustomer(33864, 13645)); return empty

==========

Speaking with Clint over the phone he still did not have the 'Assets' section of the the Management module available. I noticed that the customer's user account had zone limitations, to test functionality, we removed the zone limitations for his user account momentarily (and set them back up after the test was completed per his request). After the zone limitations were removed from his user account, he was able to access https://console.clouddefender.alertlogic.com/manage_user.php.

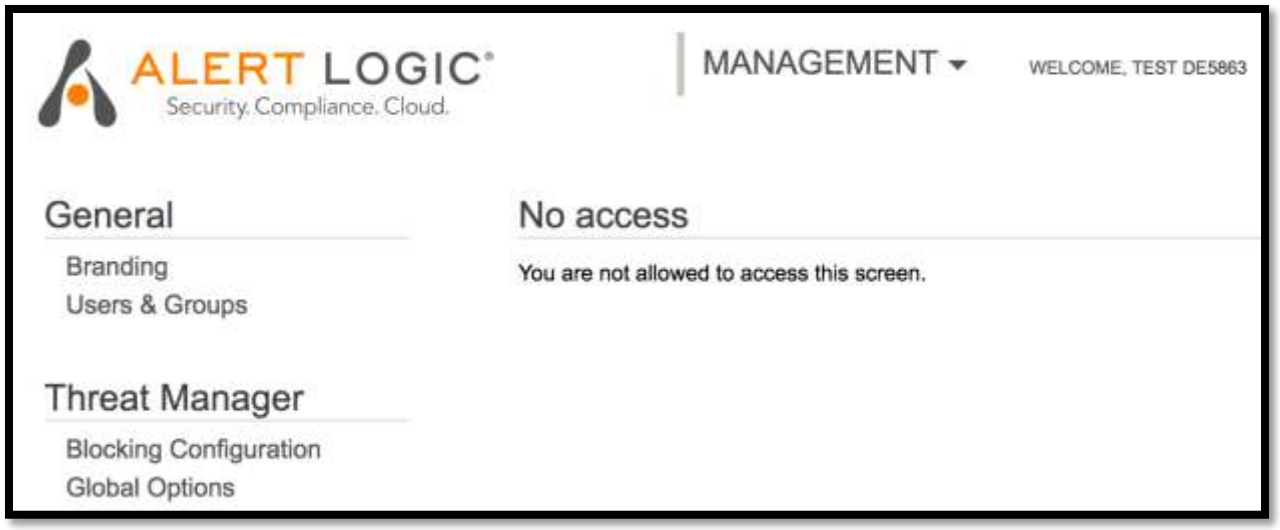
Zone limitations for his accounts were/are set to:
Classic Bank
First State Bank of Bedias
First State Bank of Chico
State Bank of Texas

Problem:
Network Security Services has several child customers under their parent account. They have contractual obligations that prevent certain employees from seeing certain customer's data/environments. For example, Clint is only supposed to have access to the 4 accounts that he has access to above and some of those accounts have legacy IDS devices. As it stands now, when the customer has zone limitations set in place, he can not manage legacy appliances.

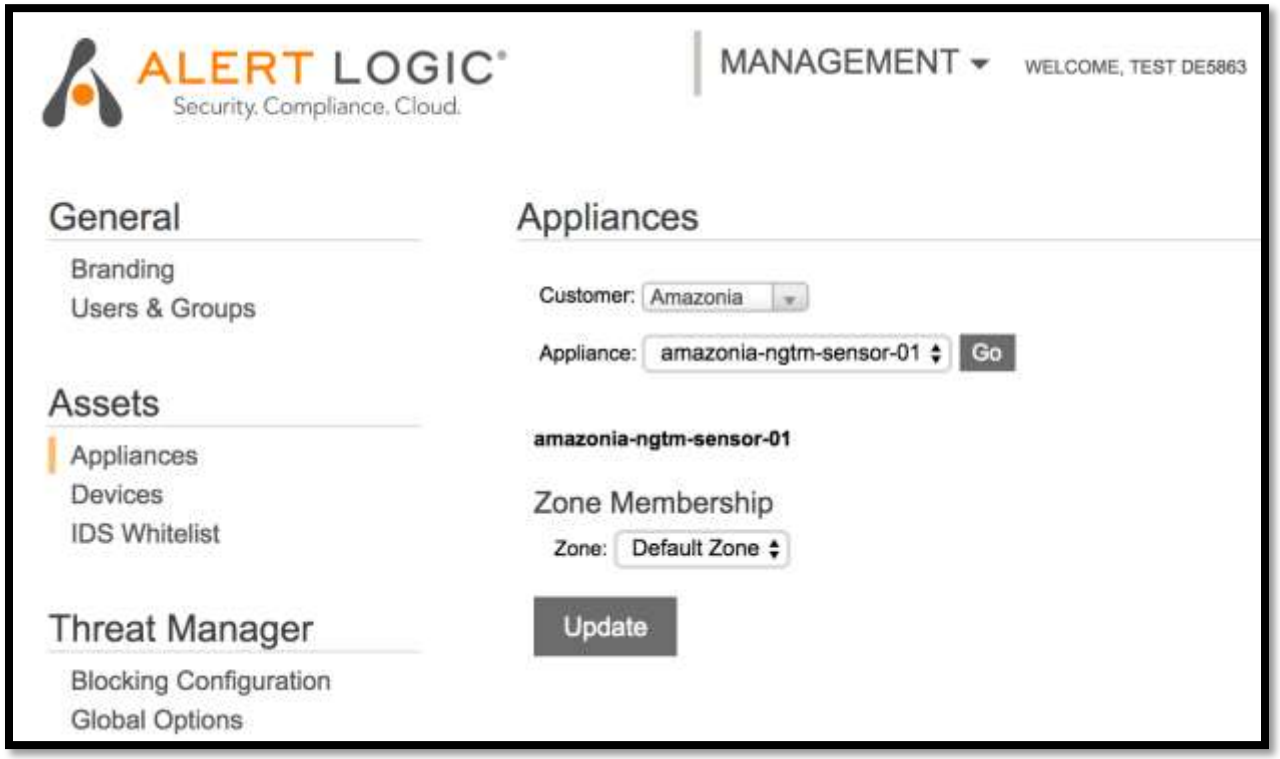
As he currently can not manage the legacy appliances, I have asked that he contact customer care whenever they have to make changes to the appliances protected networks.

Solution: the getAllowedCustomerIDs function was added in lib/class.customer.php.

Before Fix



After Fix



Date	Feb 17th 2017
Developer	Fair Andres Tarapues H.
IT Involve Link	DE6047

Fix Description (Before and After fix behavior)

DE6120: As a Alertlogic I need to modify the behaviour of "Modify log policy" permission option in Log Manager

Description:

If the user has disabled the "Modify Log Policy" permission option, in Log Manager the S3 and Azure pages should not have access (Access denied), but they have it..

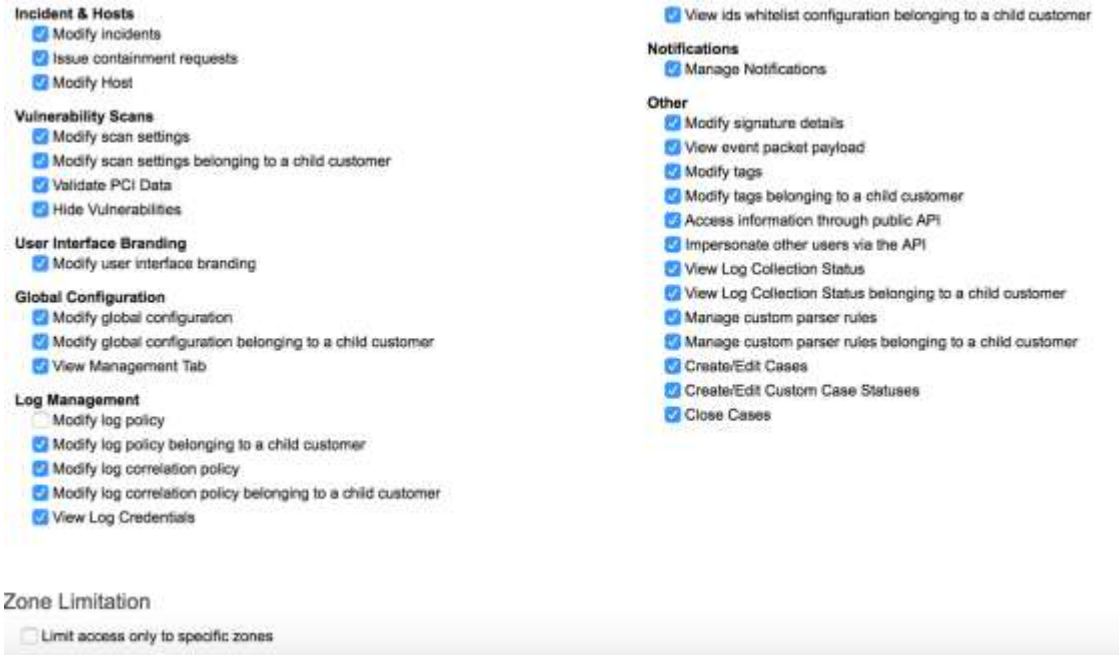
"**Action Taken**
The following behavior was verified as a bug by Barry Skidmore in the ticket #56361. This is merely the requested separate ticket requested to be passed to VTT for tracking this bug.

"3. Please confirm the respective results from the actions described below are proper behaviors.
・Log Management>Uncheck "Modify log policy"
Result：In Log Manager>Policies, S3 stayed visible while Flat File, Syslog, and Windows Eventlog were removed from the view.
Answer: This is definitively a bug, S3 should not have remained visible, Azure should also disappear."

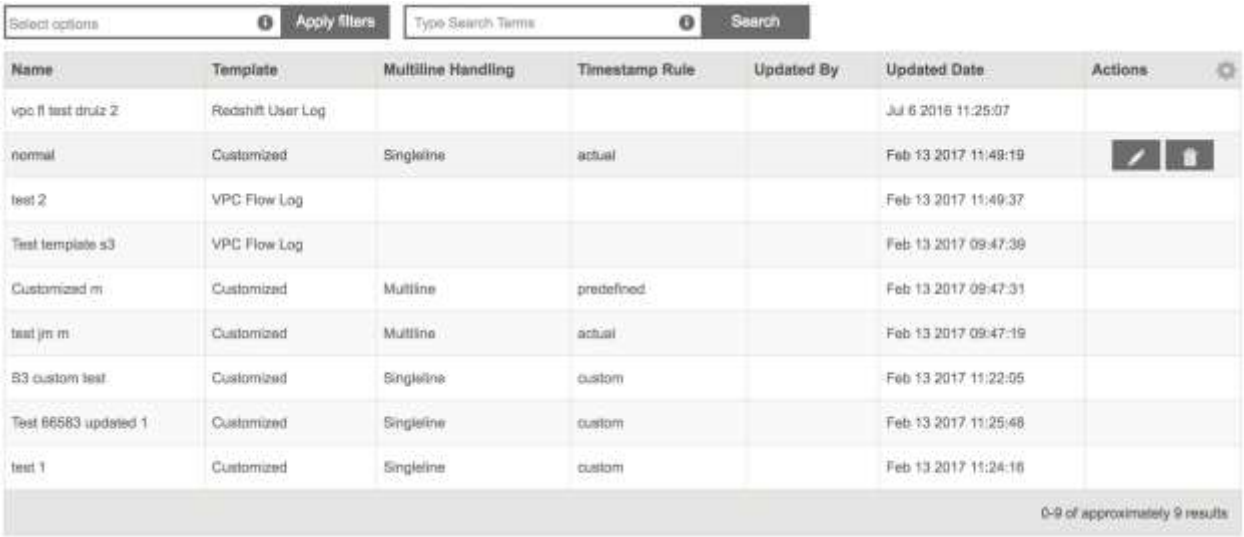
Solution: a conditional was added in S3PolicyController, redirect to AlertLogicCoreBundle_misc_access_denied if the user doesn't have permissions.

Before Fix

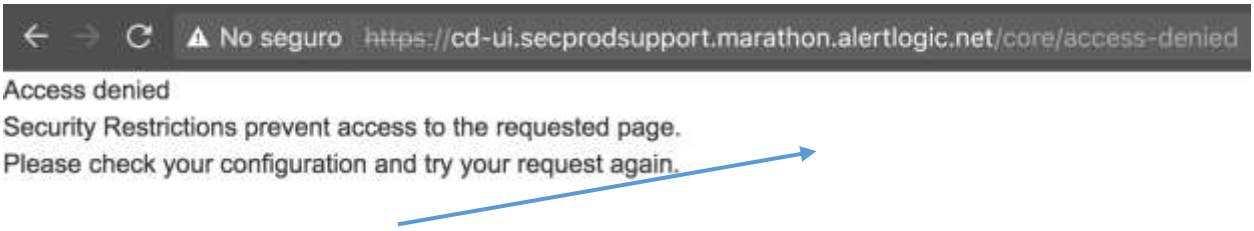
when the user doesn't have the "Modify log policy" option selected.



This form would have to disappear.



After Fix



the user doesn’t have access to S3 form.

Date	Feb 13th 2017
Developer	Fair Andres Tarapues H.
IT Involve Link	DE6120

Fix Description (Before and After fix behavior)

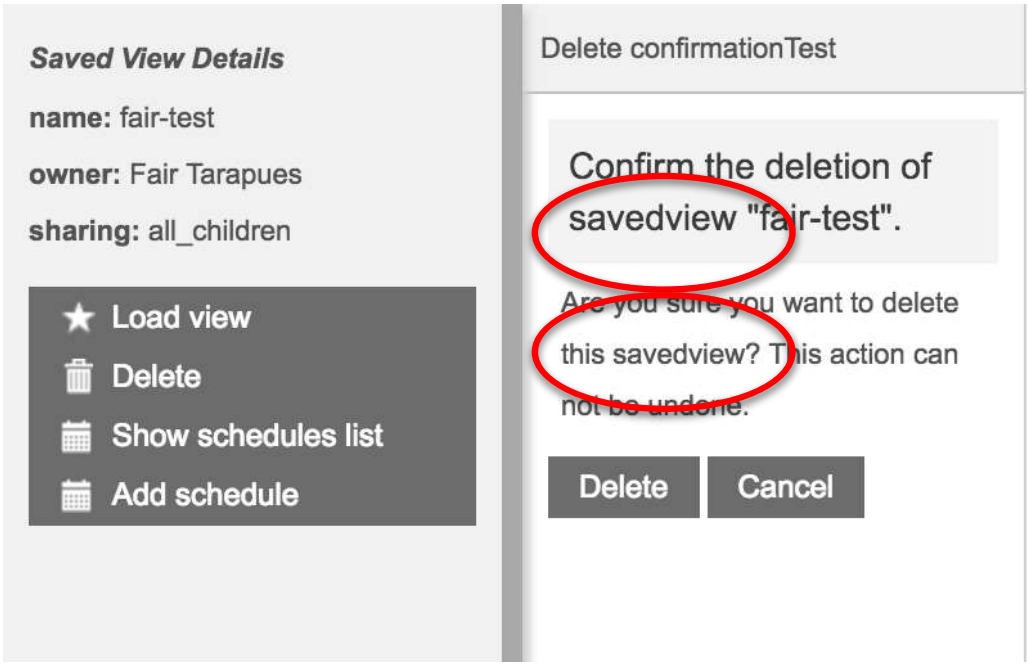
DE6120: As a developer I want that the delete saved view change from savedview to saved view.

Description:

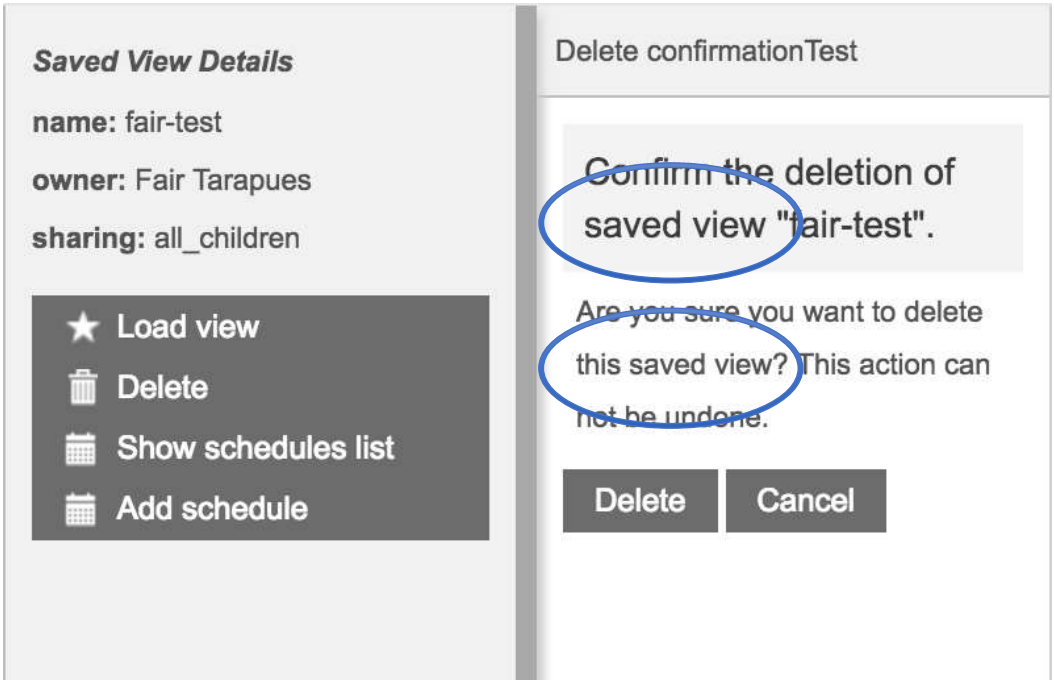
In saved view when you are going to delete a save view change the word savedview by saved view.

Solution: the ObjectType variable was changed in SaveViewsController

Before Fix



After Fix



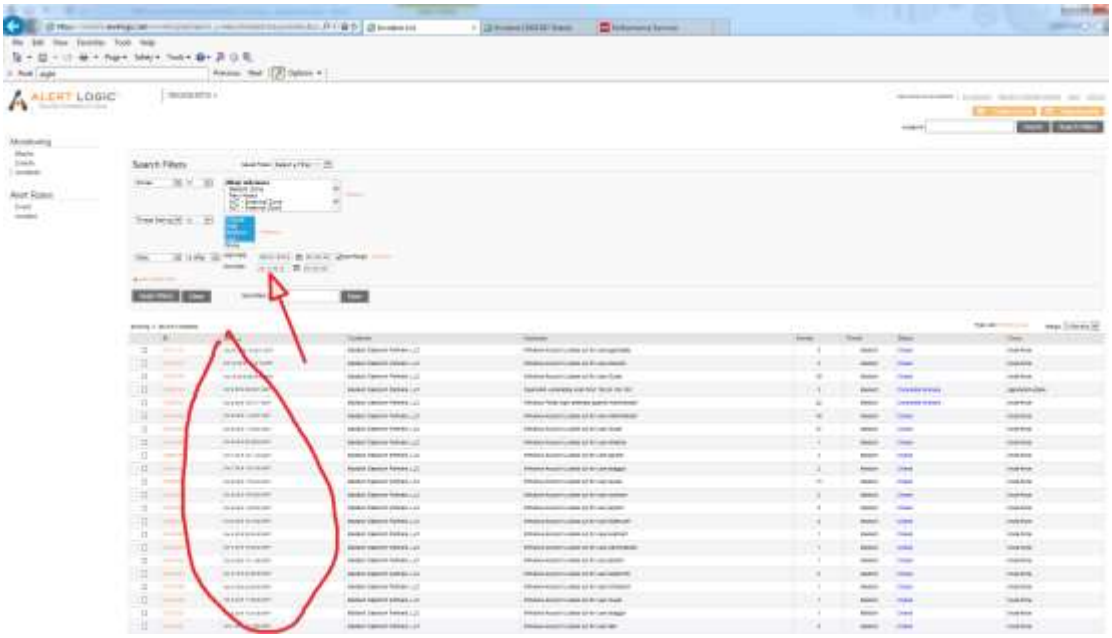
Date	Nov 24th 2016
Developer	Fair Andres Tarapues H.
IT Involve Link	DE47651

Fix Description (Before and After fix behavior)

DE47651: As Altair Advisers I want the date filters to work as expected with single digits for days.

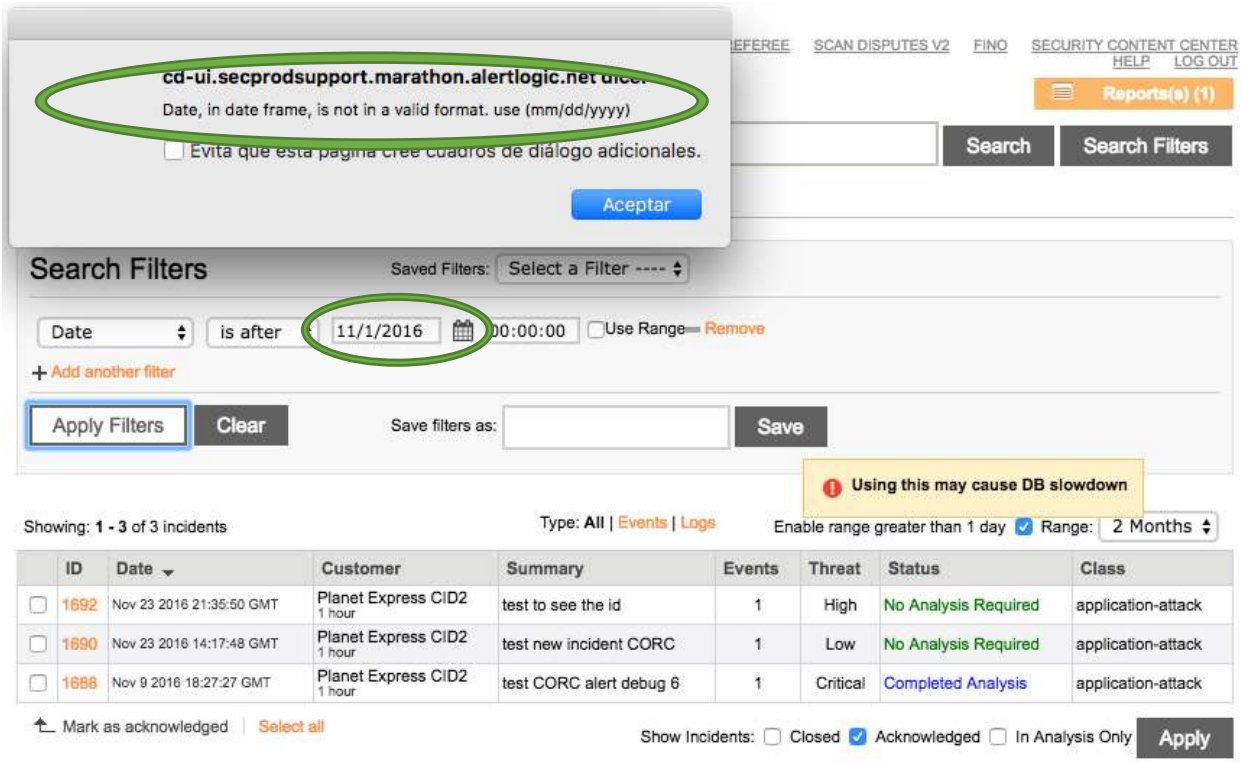
Description:

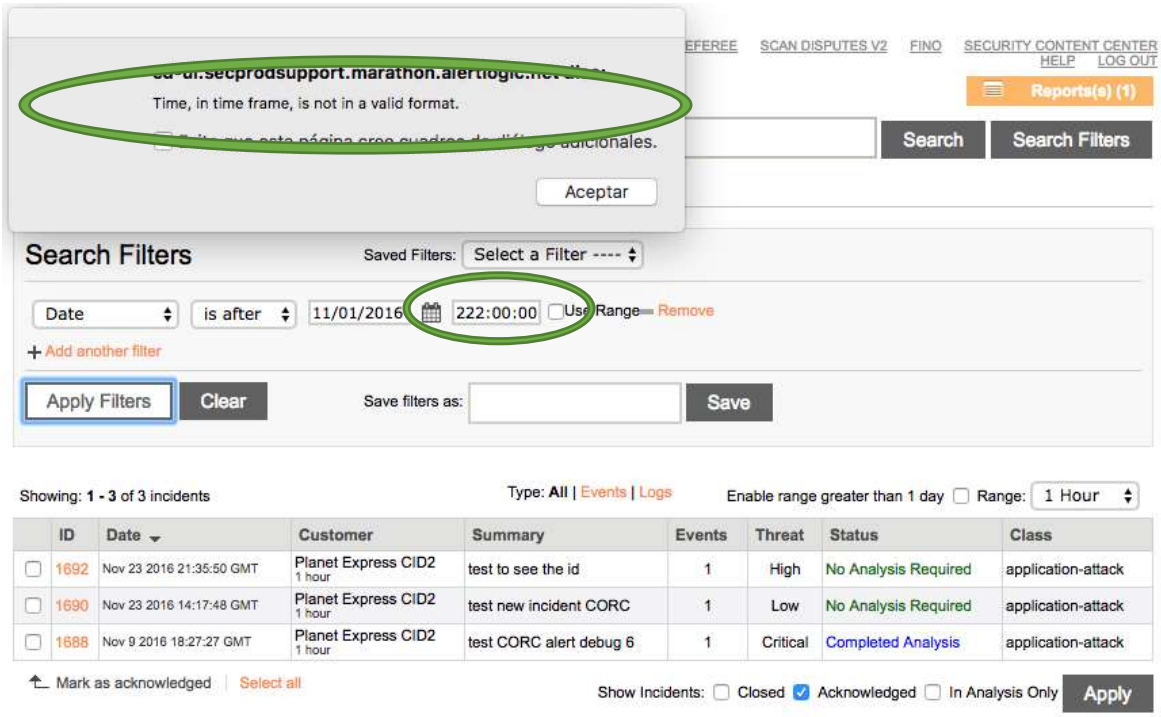
Customer is experiencing unreliable results when filtering by date and single digits are entered for the day value.
A validation or conversion must be developed for this filter.



Solution: A validation function was added to "webroot/js/monitor.js" to check the date and time frame for the Date filter.

After the fix





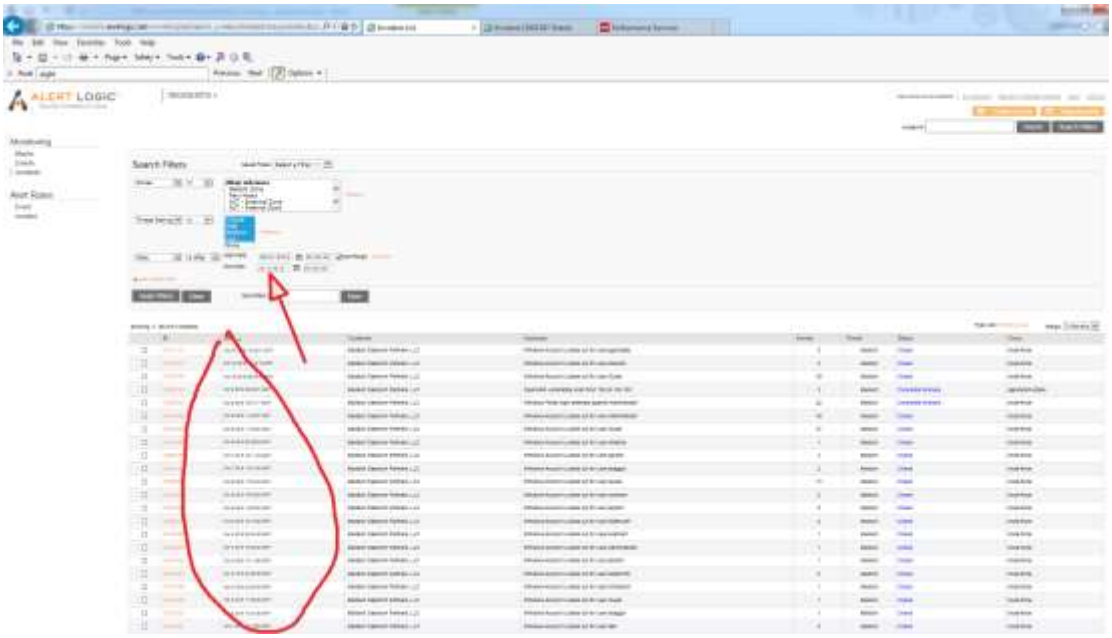
Date	Jan 10th 2016
Developer	Fair Andres Tarapues H.
IT Involve Link	DE47651

Fix Description (Before and After fix behavior)

DE47651: As Altair Advisers I want the date filters to work as expected with single digits for days.

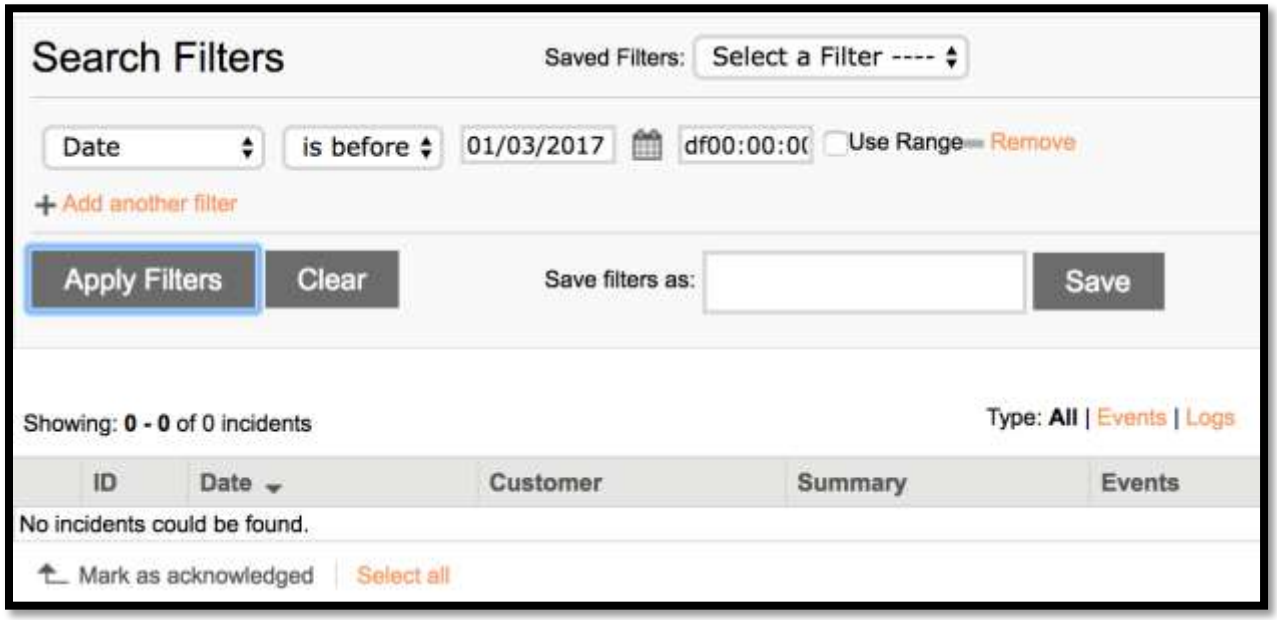
Description:

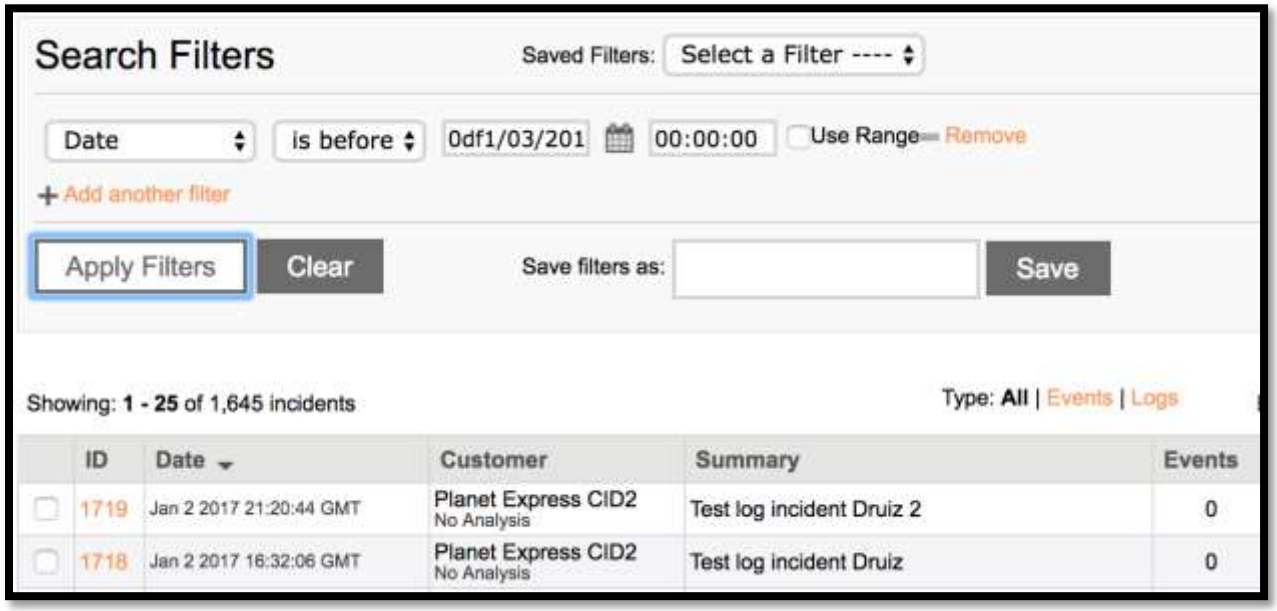
Customer is experiencing unreliable results when filtering by date and single digits are entered for the day value.
A validation or conversion must be developed for this filter.



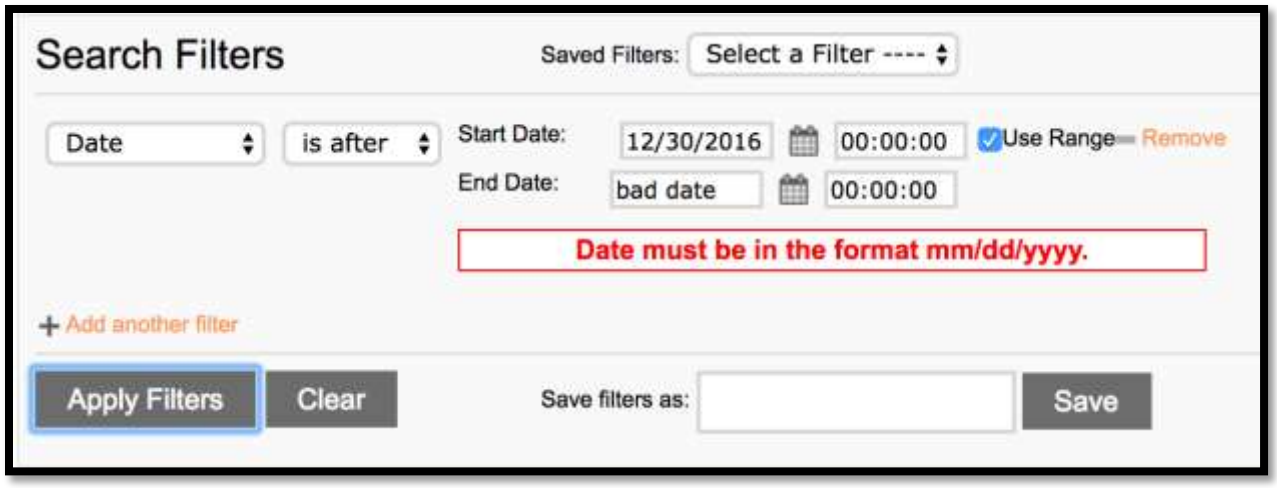
Solution: A validation function was added to "webroot/js/monitor.js" to check the date and time frame for the Date filter.

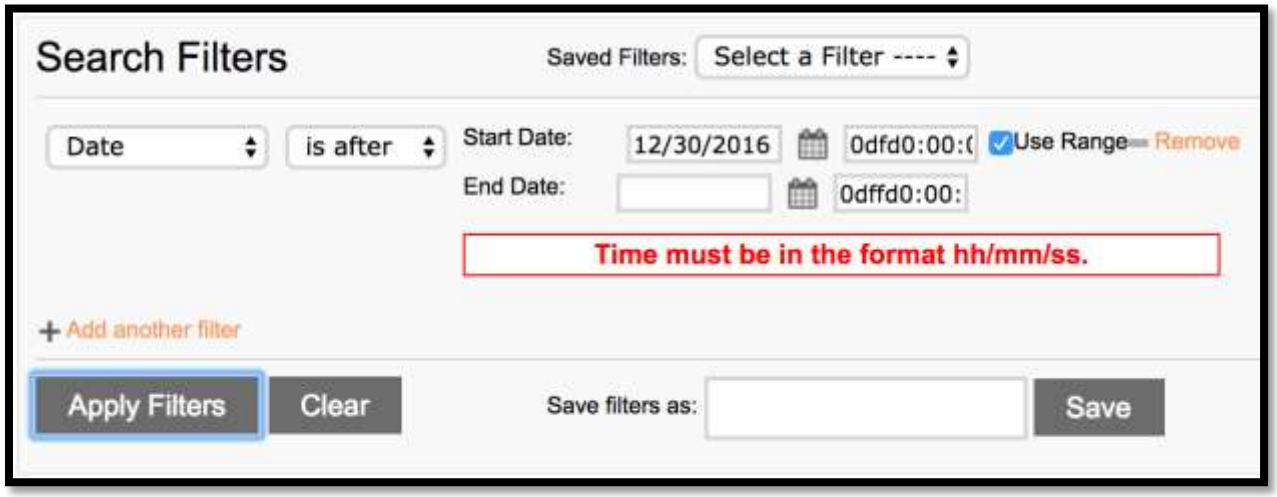
Before Fix

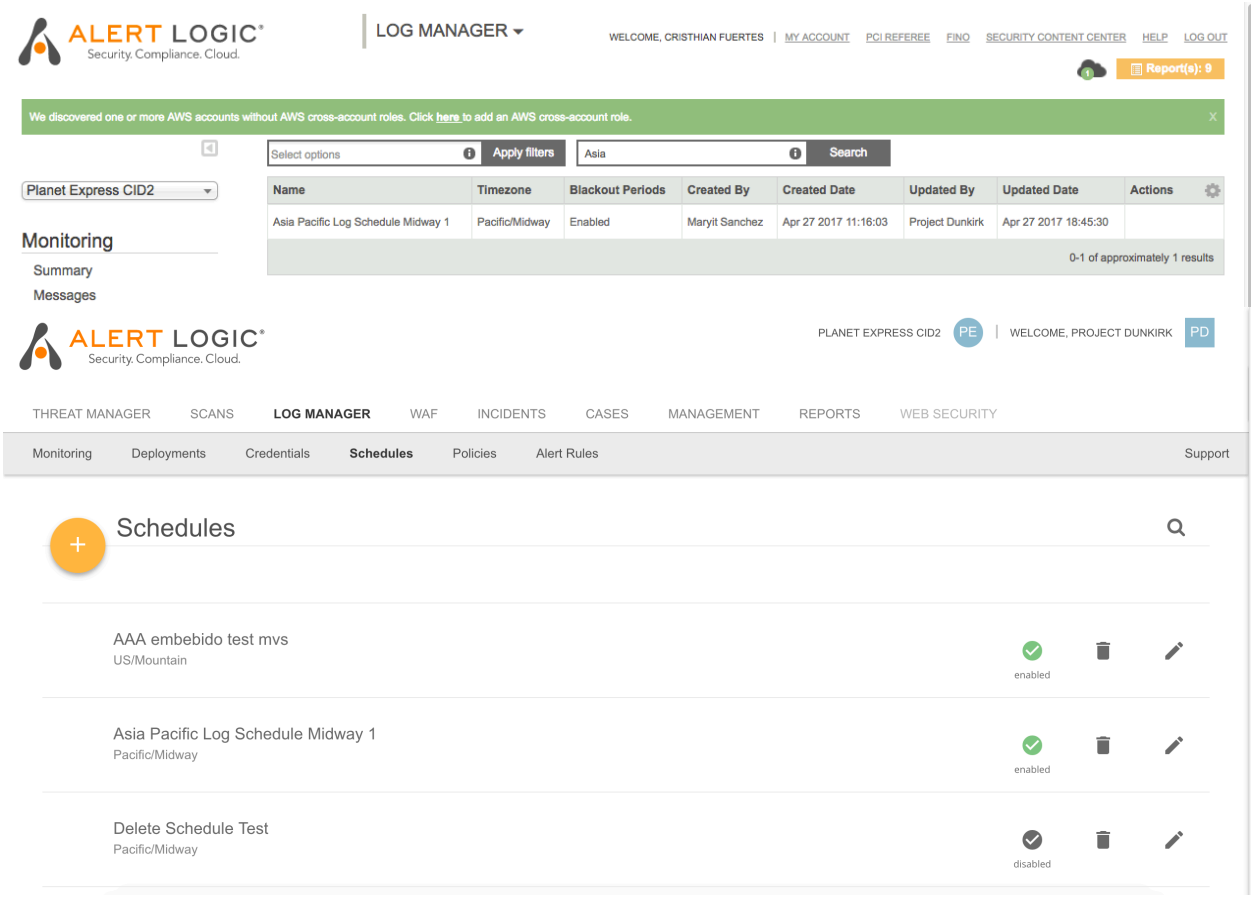

Search Filters
Saved Filters: Select a Filter ----
Date
is before
01/03/2017
df00:00:0(
Use Range
Remove
Add another filter
Apply Filters
Clear
Save filters as:
Save
Showing: 0 - 0 of 0 incidents
Type: All | Events | Logs
ID
Date
Customer
Summary
Events
No incidents could be found.
Mark as acknowledged
Select all

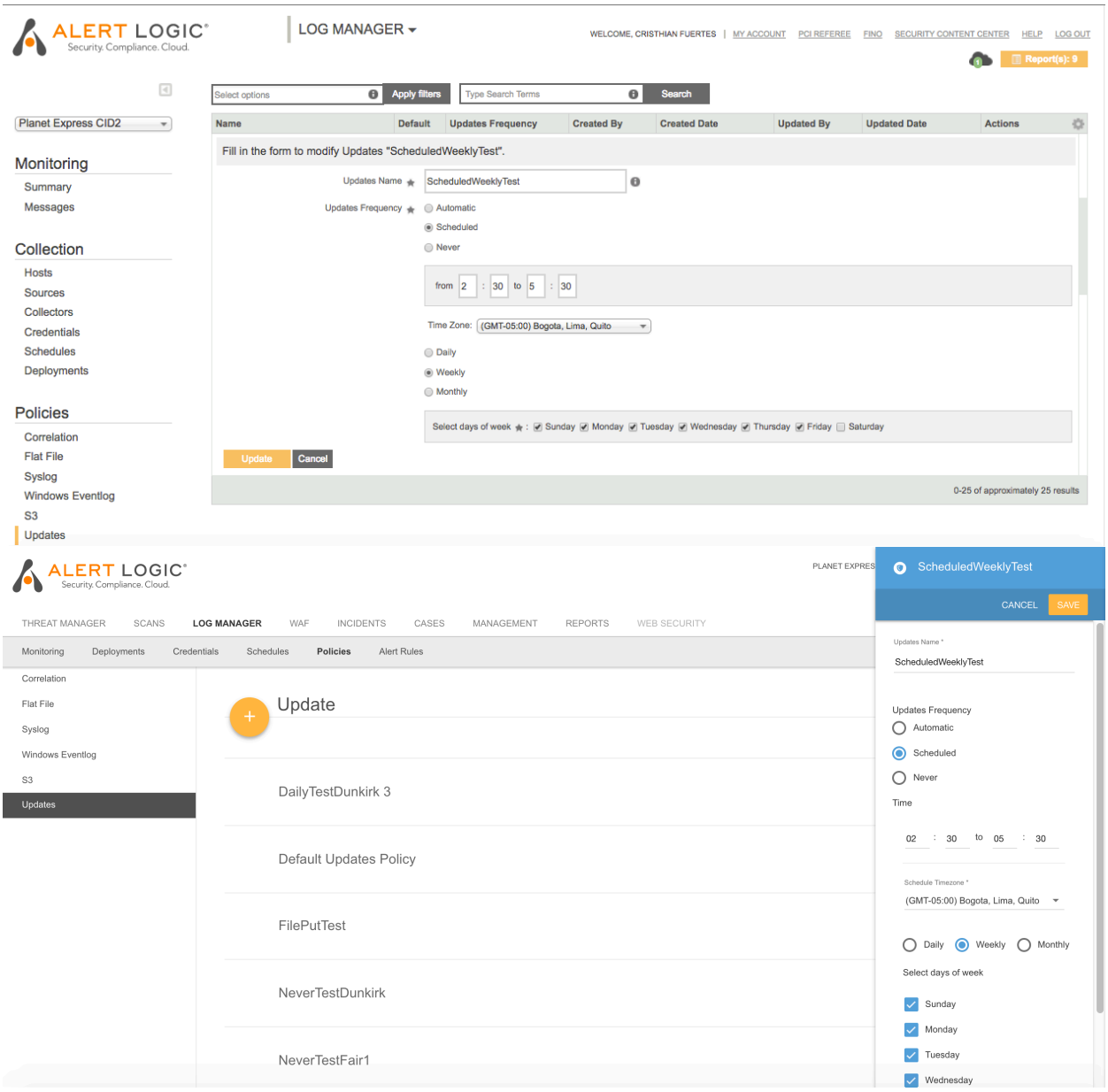

Search Filters
Saved Filters: Select a Filter ----
Date
is before
0df1/03/201
00:00:00
Use Range
Remove
Add another filter
Apply Filters
Clear
Save filters as:
Save
Showing: 1 - 25 of 1,645 incidents
Type: All | Events | Logs
ID
Date
Customer
Summary
Events
1719
Jan 2 2017 21:20:44 GMT
Planet Express CID2
No Analysis
Test log incident Druiz 2
0
1718
Jan 2 2017 16:32:06 GMT
Planet Express CID2
No Analysis
Test log incident Druiz
0

After the fix






ALERT LOGIC
Security. Compliance. Cloud.
LOG MANAGER
WELCOME, CRISTHIAN FUERTES | MY ACCOUNT PCI REFEREE FINO SECURITY CONTENT CENTER HELP LOG OUT
Report(s): 9
We discovered one or more AWS accounts without AWS cross-account roles. Click here to add an AWS cross-account role.
Select options
Apply filters
Asia
Search
Planet Express CID2
Monitoring
Summary
Messages
Name
Timezone
Blackout Periods
Created By
Created Date
Updated By
Updated Date
Actions
Asia Pacific Log Schedule Midway 1
Pacific/Midway
Enabled
Maryit Sanchez
Apr 27 2017 11:16:03
Project Dunkirk
Apr 27 2017 18:45:30
0-1 of approximately 1 results
ALERT LOGIC
Security. Compliance. Cloud.
PLANET EXPRESS CID2
PE
WELCOME, PROJECT DUNKIRK
PD
THREAT MANAGER
SCANS
LOG MANAGER
WAF
INCIDENTS
CASES
MANAGEMENT
REPORTS
WEB SECURITY
Monitoring
Deployments
Credentials
Schedules
Policies
Alert Rules
Support
Schedules
AAA embebido test mvs
US/Mountain
enabled
Asia Pacific Log Schedule Midway 1
Pacific/Midway
enabled
Delete Schedule Test
Pacific/Midway
disabled


ALERT LOGIC
Security. Compliance. Cloud.
LOG MANAGER
WELCOME, CRISTHIAN FUERTES | MY ACCOUNT PCI REFEREE FINO SECURITY CONTENT CENTER HELP LOG OUT
Report(s): 9
Select options
Apply filters
Type Search Terms
Search
Planet Express CID2
Monitoring
Summary
Messages
Collection
Hosts
Sources
Collectors
Credentials
Schedules
Deployments
Policies
Correlation
Flat File
Syslog
Windows Eventlog
S3
Updates
Name
Default
Updates Frequency
Created By
Created Date
Updated By
Updated Date
Actions
Fill in the form to modify Updates "ScheduledWeeklyTest".
Updates Name
ScheduledWeeklyTest
Updates Frequency
Automatic
Scheduled
Never
from 2 : 30 to 5 : 30
Time Zone: (GMT-05:00) Bogota, Lima, Quito
Daily
Weekly
Monthly
Select days of week : Sunday Monday Tuesday Wednesday Thursday Friday Saturday
Update
Cancel
0-25 of approximately 25 results
ALERT LOGIC
Security. Compliance. Cloud.
PLANET EXPRES
THREAT MANAGER
SCANS
LOG MANAGER
WAF
INCIDENTS
CASES
MANAGEMENT
REPORTS
WEB SECURITY
Monitoring
Deployments
Credentials
Schedules
Policies
Alert Rules
Correlation
Flat File
Syslog
Windows Eventlog
S3
Updates
Update
DailyTestDunkirk 3
Default Updates Policy
FilePutTest
NeverTestDunkirk
NeverTestFair1
ScheduledWeeklyTest
CANCEL
SAVE
Updates Name *
ScheduledWeeklyTest
Updates Frequency
Automatic
Scheduled
Never
Time
02 : 30 to 05 : 30
Schedule Timezone *
(GMT-05:00) Bogota, Lima, Quito
Daily
Weekly
Monthly
Select days of week
Sunday
Monday
Tuesday
Wednesday