

Desplegar la herramienta "Zeek IDS" y su posterior explotación para el análisis de actividades sospechosas en la red

Xavier Farré López

Máster Universitario en Seguridad de las Tecnologías
de la Información y de las Comunicaciones
TFM - Hacking

Borja Guaita Pérez
Víctor García Font

31/12/2019



Esta obra está sujeta a una licencia de Reconocimiento-SinObraDerivada [3.0 España de Creative Commons](https://creativecommons.org/licenses/by-nc/3.0/es/)

FICHA DEL TRABAJO FINAL

Título del trabajo:	<i>Desplegar la herramienta "Zeek IDS" y su posterior explotación para el análisis de actividades sospechosas en la red</i>
Nombre del autor:	<i>Xavier Farré López</i>
Nombre del consultor/a:	<i>Borja Guaita Pérez</i>
Nombre del PRA:	<i>Víctor García Font</i>
Fecha de entrega (mm/aaaa):	31/12/2019
Titulación:	<i>Máster Universitario en Seguridad de las Tecnologías de la Información y de las Comunicaciones</i>
Área del Trabajo Final:	<i>TFM - Hacking</i>
Idioma del trabajo:	<i>Castellano</i>
Palabras clave	<i>IDS, análisis, monitorización</i>
<p>Resumen del Trabajo (máximo 250 palabras): <i>Con la finalidad, contexto de aplicación, metodología, resultados i conclusiones del trabajo.</i></p>	
<p>El trabajo recoge el proceso de construcción de un sistema de monitorización que, a partir del tráfico de red, identifica actividades sospechosas que podrían indicar una vulneración de la seguridad de los sistemas informáticos en una red de comunicaciones.</p> <p>La finalidad del trabajo era la de disponer de un sistema completo de monitorización de bajo coste, capaz de detectar tráfico de red sospechoso a partir de listas de reputación.</p> <p>El trabajo se ha basado en el uso del Zeek IDS, como sistema de detección de intrusos, encargado del análisis del tráfico de red; el stack Elastic, para la monitorización activa del tráfico de red; y MineMeld como herramienta de gestión de listas de reputación.</p> <p>Para el desarrollo del trabajo se ha utilizado una metodología de tipo waterfall, dividida en diferentes fases propias de un proyecto de desarrollo de software: definición, análisis, construcción y entrega. Se han realizado además seguimientos periódicos de avance del trabajo.</p> <p>El resultado ha sido un sistema completo de monitorización, en el cual se ha simulado tráfico de red que ha quedado registrado en el dashboard de monitorización, cuyas actividades sospechosas, en base a las listas de reputación, han quedado identificadas para un posterior análisis.</p> <p>Se concluye que, con las herramientas utilizadas, se puede obtener un completo sistema de monitorización, fácilmente adaptable y extensible a las</p>	

necesidades de cualquier ámbito de aplicación, tanto empresarial como particular, y que el uso de listas de reputación es clave para la identificación de actividades maliciosas en este tipo de sistemas.

Abstract (in English, 250 words or less):

The work covers the process of building a full monitoring system that, based on network traffic, identifies suspicious activities that could indicate a breach of the security of computer systems in a communications network.

The purpose of the work was to have a complete low-cost monitoring system capable of detecting suspicious network traffic from reputation lists.

The work was based on the use of Zeek IDS, as an intrusion detection system, responsible for analyzing network traffic; the Elastic stack, for active monitoring of network traffic; and MineMeld as a reputation list management tool.

For the development of the work, a waterfall type methodology has been used, divided into different phases of a software development project: definition, analysis, construction and delivery. In addition, periodic progress reporting has been carried out.

The result has been a complete monitoring system, in which network traffic has been simulated and registered in the monitoring dashboard, whose suspicious activities, based on reputation lists, have been identified for later analysis.

It is concluded that, with the tools used, a complete monitoring system can be obtained, easily adaptable and extensible to the needs of any field of application, both business and private, and that the use of reputation lists is key to the identification of malicious activities in this type of systems.

Índice

1. Introducción	1
1.1 Contexto y justificación del Trabajo.....	1
1.2 Objetivos del Trabajo	2
1.3 Enfoque y método seguido	3
1.4 Planificación del Trabajo.....	4
1.5 Análisis de Riesgos	7
1.6 Estado del Arte	8
1.7 Breve resumen de productos obtenidos.....	10
1.8 Breve descripción de los otros capítulos de la memoria.....	10
2. Fase de Análisis	12
2.1. Componentes principales	12
2.2 Enfoque de la solución	13
2.3 Inventario software	14
2.4 Topología de red.....	14
2.5 Captura de datos	15
2.6 Listas de reputación.....	17
2.7 Ingesta de información.....	20
2.8 Visualización de datos	22
2.9 Arquitectura final de la solución	23
2.10 Plan de pruebas.....	25
3. Construcción del sistema	26
3.1 Componentes software	26
3.1.1 Zeek IDS.....	26
3.1.2 Minemeld	30
3.1.3 Python	42
3.1.4 Stack Elastic	44
3.1.5 Logstash	48
3.2 Monitorización	55
3.2.1 Creación patrones de índices	55
3.2.2 Consulta de información	56
3.2.3 Construcción Dashboards.....	57
4. Validaciones	72
4.1 Validación integración Zeek IDS con stack Elastic	72
4.2 Validación gestión listas de reputación	73
4.3 Validación detección actividades maliciosas.....	75
4.4 Validación dashboard de monitorización.....	76
5. Conclusiones	78
5.1 Conclusiones del trabajo.....	78
5.2 Revisión objetivos.....	79
5.3 Seguimiento y metodología	80
5.4 Dificultades encontradas.....	81
5.5 Líneas de trabajo futuras	82
6. Glosario	83
7. Referencias.....	86
8. Anexos.....	88

Anexos

Anexo I. Inventario software.....	88
Sistema operativo	88
Software de virtualización	88
Zeek IDS.....	89
Stack Elastic	89
Minemeld	90
Python	90
Anexo II. Setup máquinas virtuales.....	91
Instalación.....	91
Configuración.....	96
Validación	98
Anexo III. Scripts Python.....	100
Script direcciones IP	100
Script dominios	101
Script URL's.....	102
Script Ficheros.....	102
Módulo de utilidades	103
Anexo IV. Configuración Logstash.....	105
Fichero zeek.conf.....	105
Fichero zeek-patterns	107
Anexo V. Template Elasticsearch	108
Configuración del template.....	108
Anexo VI. Script Zeek	109
Script Zeek. Ampliar información files.log.....	109
Anexo VII. Script y ficheros Pruebas.....	110
Script generador tráfico. generate_traffic.sh.....	110
Lista de URLs lícitas. good_traffic.txt	110
Lista de URLs maliciosas. bad_traffic.txt.....	113
Lista de URLs de ficheros lícitos. good_files.txt	115
Lista de URLs de ficheros maliciosos. bad_files.txt.....	115

Lista de figuras

Figura 1. Brechas de Seguridad y datos expuestos en EEUU.	1
Figura 2. Diagrama de Gantt con la planificación del trabajo.	6
Figura 3. Relación componentes stack Elastic.....	13
Figura 4. Topología de red de la solución.....	15
Figura 5. Configuración tarjeta red en software virtualización.....	16
Figura 6. Captura de pantalla de la plataforma Minemeld.....	19
Figura 7. Relación logs Zeek IDS y listas Minemeld.....	21
Figura 8. Esquema de la arquitectura de la solución.....	24
Figura 9. Log de arranque del sistema Zeek IDS.....	28
Figura 10. Validación estado de servicio Zeek IDS.....	28
Figura 11. Ejemplo contenido fichero de trazas http.log de Zeek IDS.....	29
Figura 12. Nodos de entrada dados de alta para el sistema desarrollado.....	40
Figura 13. Validación estado de la herramienta Minemeld (2).....	40
Figura 14. Validación estado de los nodos Minemeld.....	41
Figura 15. Validación acceso a feed Minemeld de tipo IP desde navegador.....	41
Figura 16. Validación acceso a feed Minemeld de tipo Dominio desde navegador.....	41
Figura 17. Validación acceso a feed Minemeld de tipo URL desde navegador.....	42
Figura 18. Validación acceso a feed Minemeld de tipo MD5 desde navegador.....	42
Figura 19. Página de bienvenida de Kibana.....	46
Figura 20. Validación acceso a servicio Elasticsearch.....	47
Figura 21. Validación estado servicio Logstash.....	54
Figura 22. Listado de índices generados en Elasticsearch.....	55
Figura 23. Pantalla de creación de patrón de índice en Kibana – Paso 1.....	56
Figura 24. Pantalla de creación de patrón de índice en Kibana – Paso 2.....	56
Figura 25. Ejemplo de consulta de información a través de Kibana.....	56
Figura 26. Ejemplo de traza recuperada en Kibana después de consulta.....	57
Figura 27. Mapa de calor del mundo con conexiones destino.....	58
Figura 28. Gráfico circular con Países de las conexiones destino.....	58
Figura 29. Total bytes de datos de entrada.....	59
Figura 30. Total bytes de datos de salida.....	59
Figura 31. Gráfico de barras con servicios detectados.....	60
Figura 32. Total conexiones maliciosas detectadas.....	60
Figura 33. Lista de las conexiones maliciosas detectadas.....	61
Figura 34. Dashboard de monitorización “Visión Global”.....	61
Figura 35. Total de peticiones http realizadas.....	62
Figura 36. Gráfico de barras con códigos de respuesta y totales.....	62
Figura 37. Top 10 de URLs visitadas.....	63
Figura 38. Total peticiones http maliciosas detectadas.....	63
Figura 39. Dashboard de monitorización “Análisis URL”.....	64
Figura 40. Total de consultas dns realizadas.....	65
Figura 41. Top 10 de consultas dns realizadas.....	66
Figura 42. Total consultas dns maliciosas detectadas.....	66
Figura 43. Lista de consultas dns maliciosas detectadas.....	67
Figura 44. Dashboard de monitorización “Análisis Dominios”.....	67
Figura 45. Total de ficheros descargados.....	68
Figura 46. Total bytes de ficheros descargados.....	69
Figura 47. Top 10 de extensiones de ficheros descargadas.....	69
Figura 48. Total ficheros maliciosos descargados.....	70
Figura 49. Lista de ficheros maliciosos descargados.....	70
Figura 50. Dashboard de monitorización “Análisis Ficheros”.....	71
Figura 51. Trazas de fichero http.log en primera fase pruebas.....	72
Figura 52. Trazas de fichero dns.log en primera fase pruebas.....	72

Figura 53. Trazas de fichero files.log en primera fase pruebas	72
Figura 54. Registro de logs en Kibana de registros DNS en primera fase de pruebas	72
Figura 55. Contenido del feed de IPs de Minemeld.....	73
Figura 56. Contenido del feed de Dominios de Minemeld	73
Figura 57. Contenido del feed de URLs de Minemeld.....	74
Figura 58. Contenido del feed de Hashes de Minemeld.....	74
Figura 59. Ficheros generados por los programas Python	74
Figura 60. Validación de no existencia de dominio en fichero domains.yml	75
Figura 61. Validación de no existencia de dominio en fichero domains.yml	75
Figura 62. Validación de existencia de dominio en fichero domains.yml	75
Figura 63. Validación registro dns de consulta de dominio lícito	76
Figura 64. Validación registro dns de consulta de dominio malicioso.....	76
Figura 65. Ficheros para validación cuarta fase pruebas.	77
Figura 66. Dashboard Visión Global con actividad maliciosa.	77
Figura 67. Validación integridad Imagen SO en local.....	88
Figura 68. Validación integridad software VM en local.	89
Figura 69. Pantalla creación máquina virtual	91
Figura 70. Pantalla creación disco duro virtual.....	92
Figura 71. Pantalla información máquina virtual.....	93
Figura 72. Carga imagen ISO de sistema operativo.....	93
Figura 73. Pantalla configuración adaptador de red.....	94
Figura 74. Consola Linux de la máquina virtual instalada	95
Figura 75. Lista de máquinas virtuales creadas	96
Figura 76. Validación configuración red máquina uoc-server-ids	98
Figura 77. Validación configuración nombre máquina uoc-server-ids	98
Figura 78. Validación configuración red máquina uoc-server-elk	98
Figura 79. Validación configuración nombre máquina uoc-server-elk	98
Figura 80. Validación configuración red máquina uoc-server-minemeld	99
Figura 81. Validación configuración nombre máquina uoc-server-minemeld.....	99
Figura 82. Validación configuración red máquina uoc-server-host	99
Figura 83. Validación configuración nombre máquina uoc-server-host	99

Lista de tablas

Tabla 1. Riesgos del trabajo	8
Tabla 2. Ficheros log de Zeek IDS utilizados en el trabajo.....	16
Tabla 3. Configuración nodos miner en Minemeld	19
Tabla 4. Relación nodos processor y nodos miner en Minemeld	19
Tabla 5. Prototipos utilizados en nodos processor en Minemeld.....	20
Tabla 6. Relación nodos output y nodos processos en Minemeld.....	20
Tabla 7. Url de los feed generados en Minemeld	20
Tabla 8. Relación índices elasticsearch, ficheros de trazas y listas de reputación	22
Tabla 9. Relación nodos processor con nodos miner en Minemeld.....	39
Tabla 10. Relación nodos output con nodos processor en Minemeld.....	40

1. Introducción

1.1 Contexto y justificación del Trabajo

El volumen de incidentes de seguridad se ha incrementado de forma considerable en los últimos años. La profesionalización de equipos dedicados a vulnerar los sistemas tanto de organizaciones, como de particulares, hace que cada vez sea más complicado disponer de contramedidas que puedan protegernos frente a los ataques, lo cual acaba generando pérdidas de información, extorsiones y otro tipo de delitos con el propósito principal de obtener un beneficio económico de ello.

Las cifras así lo indican. En Estados Unidos, el volumen de brechas de seguridad reportadas anualmente entre los años 2016 y 2018, superaba los mil millones al año, representando una fuga de información de 446 millones de registros en el año 2018.

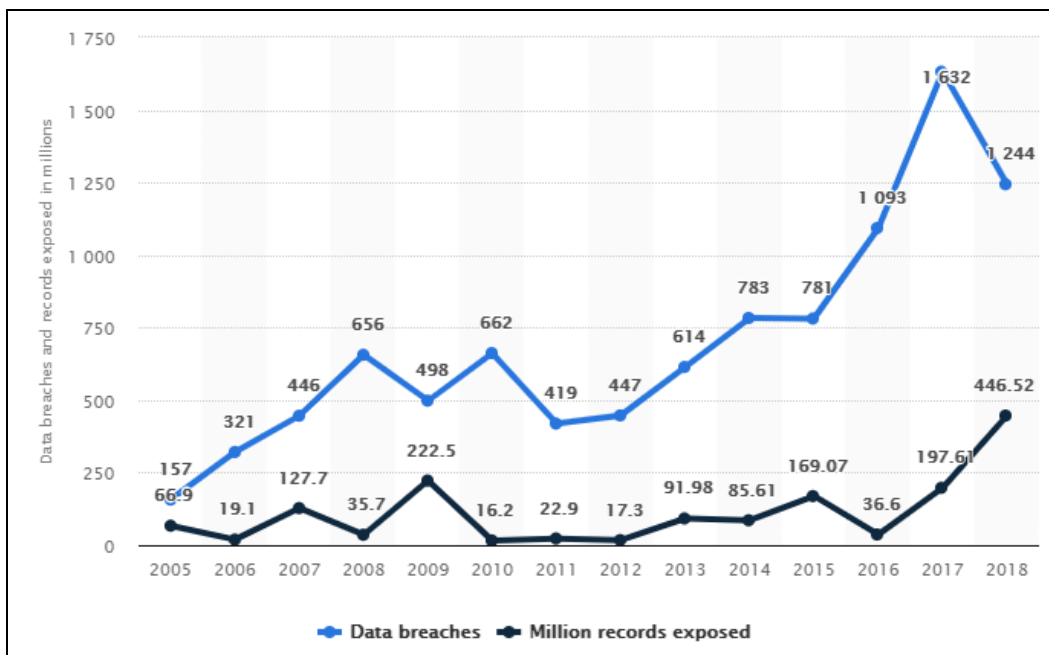


Figura 1. Brechas de Seguridad y datos expuestos en EEUU.

Fuente: Statista 2019, "Annual number of data breaches and exposed records in the United States from 2005 to 2018 (in millions)."

Y el objetivo de los ataques no son siempre grandes organizaciones. Al contrario. Según indica el portal de noticias Small Business Trends (referencia 1), en el año 2017 se identificó que el 43% de los ataques a nivel mundial estaba dirigido a pequeñas empresas, la mayoría de las cuales no dispone de dispositivos de seguridad adecuados para una detección proactiva de incidentes de seguridad. De hecho, según se menciona en la misma fuente, sólo el 14% de las organizaciones indicaba estar preparada para mitigar riesgos de seguridad y vulnerabilidades de forma efectiva, en parte por los altos costes que supone la adquisición y mantenimiento de equipos de seguridad.

Otro dato relevante, según el portal de noticias ZDNet (referencia 2), es que el promedio de detección de brechas de seguridad se sitúa entorno a los 6 meses, lo cual sugiere que: o bien las organizaciones no disponen de sistemas de seguridad que permitan la detección proactiva de esos incidentes hasta que el riesgo se ha materializado; o bien que disponen de dichos sistemas, pero debido a la gran cantidad de información que se recopila, no son capaces de diferenciar el tráfico lícito del que no lo es.

El propósito de este trabajo es precisamente hacer más accesible a las pequeñas organizaciones un sistema de seguridad que permita la detección ágil de posibles incidentes, que consistirá en el diseño una arquitectura de seguridad compuesta por un sistema de detección de intrusos, IDS, que analice el tráfico de red, y cuya información será ingesta en una herramienta de monitorización, que se configurará para poder detectar actividades sospechosas haciendo matching con listas de reputación que contienen direcciones IP de equipos categorizados como parte de sistemas de malware/botnets.

El resultado del trabajo será un sistema de seguridad pasivo, totalmente funcional, de bajo coste, e integrado con una plataforma de monitorización, que permitirá a cualquier organización disponer de una solución con la que poder detectar comportamientos anómalos que sugieran que han sufrido una brecha de seguridad, para poder así reducir el tiempo de reacción de cara a la mitigación del incidente.

Referencias:

1. <https://smallbiztrends.com/2017/01/cyber-security-statistics-small-business.html#targetText=43%20percent%20of%20cyber%20attacks,months%20of%20a%20cyber%20attack>.
2. <https://www.zdnet.com/article/businesses-take-over-six-months-to-detect-data-breaches/>

1.2 Objetivos del Trabajo

El objetivo principal de Trabajo de Final de Master consiste en desplegar un sistema de detección de intrusos, basado en la solución Zeek IDS, cuya información será ingesta en el stack de Elastic y visualizada en un dashboard de monitorización continua que permita detectar comportamientos anómalos en función del tráfico de entrada/salida en base a listas de reputación.

Para lograr el objetivo principal del trabajo, será necesario alcanzar los siguientes subobjetivos:

- Analizar el sistema de detección de intrusos Zeek IDS, e identificar las capacidades que nos ofrecen este tipo de soluciones para el propósito del proyecto.

- Conocer el funcionamiento del stack Elastic para la monitorización de información, y determinar cómo este puede complementarse con otros sistemas de seguridad.
- Conocer los diferentes tipos de listas de reputación existentes, sus propósitos, e identificar aquellas que podrán ser utilizadas en el trabajo para la identificación de tráfico sospechoso.
- Llevar a cabo la construcción de la solución, integrando los diferentes componentes software, y validar el correcto funcionamiento del sistema en su conjunto.
- Identificar qué otras aplicaciones reales podrían derivarse del uso de los componentes utilizados, para la detección de otros tipos de amenazas.

1.3 Enfoque y método seguido

Se identifican dos posibles enfoques para la realización del trabajo:

La primera es realizar un desarrollo a medida de un sistema de detección de intrusos, que permita capturar el tráfico de red y exportarlo en ficheros de trazas para posteriormente explorarlo con una herramienta también a medida de visualización de información, con capacidad de poder comparar la información con listas de reputación y representar gráficamente los datos, además de disparar alertas. El problema de esta estrategia es que el coste para el desarrollo del software necesario sería muy costoso en tiempo, además que el resultado obtenido, debido a la inexperiencia en el desarrollo de soluciones de alto rendimiento como la que se requiere, resultaría en un producto ineficiente y no adecuado para entornos productivos.

La otra estrategia posible es utilizar productos existentes, que ofrezcan unas funcionalidades que de forma global cubran con los objetivos definidos para el trabajo. El uso de productos existentes asegura una buena estabilidad del sistema, y un rendimiento propio de soluciones como la que se espera, además que son productos que están en constante evolución y se incluyen nuevas capacidades alineadas con las últimas tendencias en materia de ciberseguridad.

Tras evaluar ambas estrategias, se ha considerado como mejor opción para el propósito del trabajo el uso de productos existentes, y se ha elegido el producto Zeek IDS como sistema de detección de intrusos, y el stack Elastic para el tratamiento y representación gráfica de la información.

Para el despliegue de los productos, se empleará una máquina con capacidad de virtualización, donde se aprovisionará una máquina virtual con sistema Operativo Linux en la cual se realizará la instalación y despliegue del producto Zeek IDS, y también del stack Elastic.

El sistema operativo Linux utilizado será Ubuntu 18.04 server 64 bits, sin entorno gráfico, con las siguientes especificaciones hardware:

- Memoria RAM: 4 GB 2600 MHz DDR4

- Disco Duro: 40GB formato ext4 y estructura de particiones estándar.
- CPU: AMD Ryzen con 2 núcleos, y 2 hilos por núcleo.

La máquina virtual tendrá conectividad a Internet, a través de la red LAN compartida con la máquina anfitrión, necesario para la ejecución de las pruebas de validación del trabajo y la conectividad con las listas de reputación.

Actualmente ya se dispone de una máquina con las especificaciones hardware necesarias para levantar la máquina virtual donde se realizará el trabajo, por lo que no es necesario adquirir hardware ni realizar ninguna inversión económica.

Se descarta la contratación de ninguna máquina en el Cloud y también la compra de una nueva máquina física para el trabajo, por disponer de capacidad de virtualización en una máquina física existente.

1.4 Planificación del Trabajo

Para la gestión del trabajo, se seguirá una metodología de tipo *waterfall*, en la que se han definido una serie de tareas que se han organizado en diferentes fases.

Las tareas que conforman cada una de las fases son:

Fase de definición:

- Explicación detallada del problema al cual el trabajo quiere dar respuesta y definición de los objetivos que se quieren alcanzar.
- Descripción de la metodología que se seguirá en el desarrollo del trabajo y tareas a realizar para alcanzar los objetivos descritos.
- Planificación temporal del trabajo a realizar y sus dependencias.
- Identificación de riesgos del trabajo y acciones correctivas.
- Revisión del estado del arte.

El resultado de esta fase dará lugar al Plan de proyecto, y corresponderá a la entrega de la PEC 1 del Trabajo de Final de Máster.

Fase de análisis:

- Realizar un análisis de la solución Zeek IDS y de la información que ésta permite recopilar a partir del tráfico de red, para cada uno de los protocolos soportados.
- Identificar qué información recopilada por Zeek IDS en los diferentes ficheros de trazas va a ser necesaria ingestar en Elastic para su posterior explotación.
- Realizar un análisis del stack Elastic e identificar de qué forma se puede realizar la ingesta de la información a partir de los ficheros de trazas generado por Zeek IDS.

- Identificar listas de reputación que puedan ser utilizadas determinar si el tráfico de red analizado puede ser generado por algún equipo infectado con malware.
- Determinar en qué componente se realizará la integración con la lista de reputación para asegurar que la tenga buen rendimiento y sea escalable.
- Identificar qué información se utilizará para hacer el matching con la lista de reputación y definir a nivel técnico cómo se realizará la integración.
- Definir el diseño final de la arquitectura de la solución, integrando todos los componentes necesarios para disponer de la monitorización continua en Elastic.
- Definir dashboard de monitorización que permita identificar tráfico sospechoso y determinar qué alertas se dispararán para notificar de posibles incidentes de seguridad.
- Definición del plan de pruebas y el set de datos que se utilizarán para validar la solución y de los resultados esperados que confirmarán el correcto funcionamiento del sistema.

El resultado de esta fase será el diseño completo de la arquitectura a implementar, habiendo identificado el mecanismo para integrar ambas herramientas y la definición de la solución en su conjunto. El alcance completo del trabajo estará definido, pendiente de su implementación. La entrega de esta fase corresponderá a la PEC 2 del Trabajo Final de Máster.

Fase de construcción y validación:

- Aprovisionar el entorno (máquina virtual) donde se instalará los componentes que definen la arquitectura de la solución.
- Llevar a cabo la instalación de los componentes software.
- Validación de que todos ellos funcionan según lo esperado y que la información recopilada por Zeek IDS se ingesta de forma correcta en Elastic.
- Documentar el paso a paso del proceso de instalación de la solución y de la integración entre los diferentes componentes.
- Implementar la integración con la lista de reputación en el componente de la arquitectura seleccionado en la fase de análisis.
- Construir el dashboard de monitorización y creación de las alertas de notificación de tráfico sospechoso.
- Desarrollo de un set de datos para validación del dashboard y de las alertas.
- Generación de tráfico de red y verificación de la correcta ingesta en Elastic, y visualización de los datos en el dashboard desarrollado.
- Ejecución de las pruebas de validación de las alertas a partir del set de datos desarrollado, y comprobación de los resultados obtenidos respecto a los esperados según plan de pruebas.

El resultado de esta fase será la implementación de la solución, donde se dispondrá de un sistema completo y funcional que cumplirá con las necesidades del trabajo a realizar y por ende con el objetivo principal del trabajo. Se incluye en esta fase la validación de la solución mediante la

ejecución del plan de pruebas elaborado en la fase de análisis. La entrega de esta fase corresponderá a la PEC 3 del Trabajo Final de Máster.

Fase de entrega:

- Validación de la consecución de todos los objetivos definidos en el plan de trabajo, así como las tareas descritas.
- Redacción de la memoria del trabajo y de la presentación.
- Elaboración de vídeo demostración de la solución.
- Empaquetado de los recursos software utilizados en el trabajo.
- Elaboración vídeo final del trabajo.
- Entrega de memoria, vídeo del trabajo y de recursos.
- Defensa del trabajo ante el tribunal asignado.

El resultado de esta fase será la elaboración y entrega del resultado del trabajo, así como la defensa de este ante el tribunal asignado para su posterior valoración. Esta fase se corresponde a la fase de Entrega Final y Defensa del Trabajo de final de Máster.

A continuación, se muestra diagrama de Gantt con la planificación de las tareas, y las dependencias entre ellas, así como los hitos relevantes del trabajo.

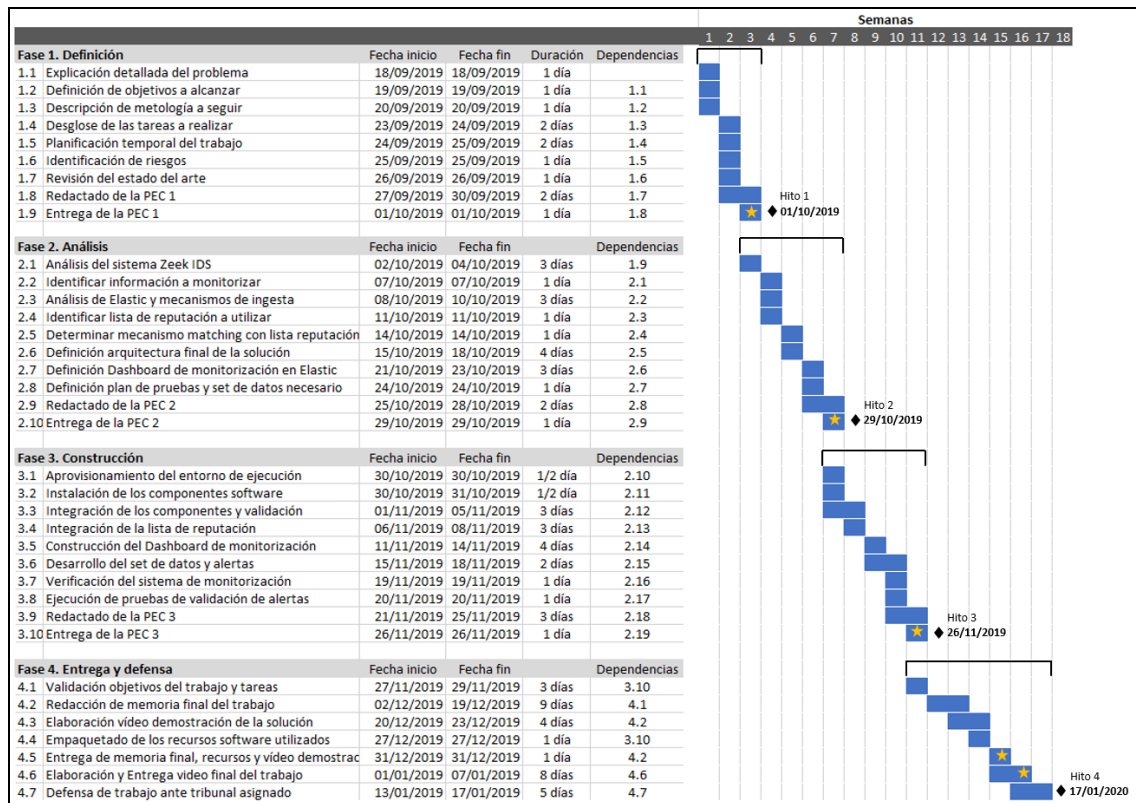


Figura 2. Diagrama de Gantt con la planificación del trabajo.

Se ha definido 4 hitos principales en el trabajo, que corresponden a cada una de las fases que componen la ejecución del proyecto. Cada hito finaliza con la entrega de uno o más entregables, según la fase.

Los entregables que se realizarán en cada una de las fases, y representados en el diagrama de Gantt con una estrella, son:

- Entregable 1. Entrega del plan de trabajo. 01/10/2019.
- Entregable 2. Entrega del análisis realizado. 29/10/2019.
- Entregable 3. Entrega del desarrollo de la solución. 26/11/2019.
- Entregable 4. Entrega de la memoria final. 31/12/2019.
- Entregable 5. Entrega del video final del trabajo. 07/01/2020.

1.5 Análisis de Riesgos

A continuación, se definen los riesgos identificados en el trabajo, y las acciones de mitigación propuestas para minimizar el impacto en la planificación y cumplir así con las fechas de entrega de cada uno de los hitos.

Categoría	Descripción	Acción mitigación
Técnicos	Pérdida de información por error hardware	Se realizará un backup diario en el Cloud de la información generada en el proyecto para su recuperación en caso de desastre.
Técnicos	Recursos hardware para virtualización insuficientes	Se solicitará la contratación de un servidor en Cloud con los recursos necesarios para el despliegue del sistema en lugar de utilizar la máquina física disponible.
Técnicos	Complejidad para la integración con listas de reputación públicas	En caso de no poder integrar con una lista de reputación pública, se elaborará un fichero con una lista de reputación customizada para poder validar la solución.
Técnicos	Producto obtenido diferente al esperado	Para asegurar que el resultado final de la solución, se definirá un plan de pruebas en la fase de análisis que se validará al finalizar la fase de construcción.
Externos	Cambio de licenciamiento de los productos utilizados	En caso de que las licencias de los productos cambien a comerciales en nuevas versiones, se seleccionarán versiones antiguas de los productos que permitan un uso no comercial para la elaboración de este trabajo.
Externos	Indisponibilidad de servicio de internet	Se utilizaría una conexión móvil en el equipo físico para disponer de conectividad a Internet tanto para la búsqueda de información, como para la conectividad con listas de reputación públicas.

Externos	Corte de suministro eléctrico máquina física	Se dispondrá de un equipo portátil con capacidad de virtualización para poder desplegar el sistema, y poder desarrollarlo en una ubicación física con suministro eléctrico.
Planificación	Tareas más costosas de lo previsto	Se ha planificado el proyecto para disponer de un margen de seguridad entre tareas, para evitar que el retraso en una de ellas pueda impactar en el resto de las planificaciones.
Planificación	Enfermedad	Se ha planificado el proyecto para disponer de un margen de seguridad entre tareas, por lo que, en caso de enfermedad, se dispondría de un colchón que permitiría volver a estar en planificación y asegurar la fecha de entrega de los entregables.
Planificación	Festividades	Se tendrá en cuenta para la planificación de las tareas los diferentes días festivos que habrá hasta la finalización del trabajo.

Tabla 1. Riesgos del trabajo

1.6 Estado del Arte

El trabajo está basado principalmente en el uso de dos componentes.

El primero de ellos es el sistema de detección de intrusos Zeek IDS, anteriormente denominado Bro IDS, que es un elemento de seguridad que se despliega en uno o diferentes puntos de una red, y que analiza el tráfico y aplica filtros para la monitorización de comportamientos que se desean detectar.

Existen otros IDS con unas funcionalidades similares. A continuación, se listan alguno de los más relevantes.

IDS Open Source:

- Snort. Es un Sistema de Detección de Intrusos basado en red (NIDS), open source. Cuenta con un lenguaje de creación de reglas en el que se pueden definir los patrones que se utilizarán a la hora de monitorizar el sistema. Al igual que Zeek IDS, puede ser utilizado como sniffer.
- Suricata. Es un sistema IDS basado en Snort, gratuito y open source, con capacidad para actuar también como sistema de prevención de intrusos, IPS. Al igual que Snort y Zeek IDS, la monitorización la realiza a nivel de red (NIDS).
- OSSEC. Este sistema funciona a nivel de máquina (HIDS), no de red, y permite la monitorización del sistema de ficheros, los registros de sistemas operativos Windows y detección de rootkits entre otras funcionalidades.

- Kismet. Este sistema está diseñado para la detección de intrusos en redes inalámbricas, tanto Wi-Fi como bluetooth, permitiendo la detección de accesos y de puntos de acceso no autorizados. Actúa además como sniffer.

IDS Comerciales

- Juniper Networks. Serie SRX Gateways. Dispositivos físicos con funcionalidades de IDS y también de IPS, con entornos gráficos de monitorización y capacidad para monitorizar diferentes redes de forma concurrente.
- DarkTrace. Cyber AI Platform. Producto que utiliza Inteligencia Artificial, concretamente machine learning, para crear modelos de tráfico que son considerados legítimos dentro de los sistemas donde está implantado, facilitando así la detección de tráfico anómalo.

Por lo que se ha podido comprobar, la tendencia a futuro será la de incorporar soluciones de inteligencia artificial, basados en el machine learning, para detectar modelos de comportamiento anómalos, lo cual complementará a los sistemas de detección actuales, consiguiendo unas soluciones mucho más completas y adecuadas a las necesidades actuales.

El segundo componente utilizado es el stack Elastic, que permite la ingesta y tratamiento de información, y su posterior visualización mediante el uso de componentes visuales, así como la búsqueda de información de forma ágil y eficiente.

Existen otras plataformas similares, como podrían ser:

- Splunk. Producto comercial ofrecido como Software as a Service que permite la ingesta de información desde diferentes fuentes, y ofrece unos componentes visuales para la visualización de información mediante el uso de gráficos y otros elementos.
- Devo. Herramienta comercial, categorizada como SIEM, que permite la correlación de eventos, y ofrece unas funcionalidades de búsqueda sobre una cantidad muy grande de información de forma ágil y visual. Ofrece además la posibilidad de generar alertas, y de construir dashboards de monitorización.
- Graylog. Herramienta que se ofrece de forma Open Source, y también Enterprise, considerada como un gestor de logs, que permite la ingesta de diferentes fuentes de información y su posterior explotación de forma visual. La versión Enterprise ofrece sobre la versión gratuita funcionalidades de correlador de eventos.

La tendencia es la de ingestar todo tipo de información en este tipo de sistemas, tanto de tráfico de red, como logs aplicativos, de sistemas, etc, por lo que el rendimiento y la escalabilidad de estas soluciones será todo un reto para las empresas que ofrecen los servicios, así como quienes optan por desplegar estas soluciones *on premise*.

1.7 Breve resumen de productos obtenidos

El producto obtenido tras la ejecución del trabajo es un sistema completo de monitorización de actividades sospechosas, basado en soluciones de código abierto y de libre distribución.

Por un lado, se dispondrá de un sistema de Detección de Intrusos basado en la solución Zeek IDS, que monitorizará la actividad de red, y que generará un volcado de información a ficheros de trazas.

Los ficheros de trazas se ingestarán en una solución Elastic mediante Logstash, donde previamente se determinará, en base a la información recogida en los registros, si se trata de actividades sospechosas haciendo matching de los datos con ficheros diccionario generados a partir de los feeds que se darán de alta en el gestor de listas de reputación Minemeld de la empresa PaloaltoNetworks, el cual se configurará para recuperar y agregar datos de actividades maliciosas de diferentes fuentes.

La información ingestada en Elastic será explotada mediante unos Dashboards de Monitorización continua donde, de forma visual, se podrán detectar actividades sospechosas para su posterior análisis.

1.8 Breve descripción de los otros capítulos de la memoria

El capítulo 2, Fase de Análisis, incluye todas las tareas de análisis del producto a implementar, comenzando por el enfoque de la solución; la selección del software que se utilizará, versiones, y mecanismos de instalación; la topología de red y arquitectura final de la solución; la información del sistema Zeek IDS que se monitorizará y las listas de repuración que se utilizarán en cada caso; y también se define cómo se realizará la ingesta de los datos y su posterior explotación en el stack Elastic mediante Dashboards de monitorización; por último se plantea un plan de pruebas para validar todo el producto.

El capítulo 3, Construcción del sistema, está dividido en dos bloques: El primero, Componentes Software, incluye una descripción paso a paso del proceso de instalación de todos los componentes software utilizados en el sistema. También se detalla la configuración a aplicar en cada uno de ellos, y las tareas de validación previas para asegurar que los diferentes elementos funcionan correctamente. El segundo bloque, Monitorización, contiene el proceso de configuración del stack Elastic, y las instrucciones para la construcción de los diferentes Dashboards de monitorización.

En capítulo 4, Validaciones, contiene el detalle de la ejecución del plan de pruebas realizado cuyo objetivo es probar todo el sistema en su conjunto y validar el correcto funcionamiento de todas las piezas de la solución de forma integrada.

El capítulo 5, Conclusiones, recoge las principales conclusiones del trabajo, y el grado de consecución de los objetivos planteados al inicio de este. A su vez, se indican las dificultades con las que se ha tenido que lidiar durante la fase de construcción del mismo, y cómo han sido resueltas para obtener un producto totalmente funcional y alineado con el planteamiento inicial. También en este capítulo se listan posibles evoluciones de la solución actual.

El capítulo 6, Glosario, incluye el glosario de términos utilizados en el trabajo y referenciados en los diferentes apartados.

El capítulo 7, Referencias, recoge los recursos utilizados para la elaboración del trabajo, desde páginas de documentación de los diferentes productos, hasta blogs de terceros con información relevante acerca de los componentes software utilizados.

El capítulo 8, Anexos, contiene ficheros de configuración utilizados en los diferentes componentes software, así como los scripts Python que generan los ficheros con la información necesaria para la detección de actividades sospechosas.

2. Fase de Análisis

2.1 Componentes principales

A continuación, se lista los diferentes componentes software que conforman la solución y una breve descripción de su uso dentro del sistema:

Zeek IDS

Sistema de detección de intrusos opensource y de libre distribución, que ofrece un framework extensible para el análisis de tráfico de red, cuya información es recogida en ficheros de trazas para su posterior explotación. Actualmente se dispone de la versión estable 3.0.0 del software (23 de septiembre de 2019). Si bien el producto en si ya nos ofrece las funcionalidades requeridas para el trabajo, el framework que incorpora permite extender las funcionalidades básicas del sistema para la elaboración de scripts de detección sofisticados, como escaneos de puertos, detección de tráfico con características propias de ciertos tipos de malware, extender los protocolos soportados, agregar y modificar información de los ficheros de trazas, etc.

El uso que se dará a Zeek IDS será el de monitorizar la actividad de red de la máquina host, desde la cual se generará el tráfico para probar el sistema. Zeek generará los ficheros de trazas que será recuperados por Logstash para su posterior tratamiento e ingesta en el stack Elastic.

Página web del producto: <https://www.zeek.org/>

Stack Elastic

Grupo de herramientas opensource, formadas por Elasticsearch, Logstash y Kibana, que de forma conjunta permiten la ingesta de información de diferentes fuentes para poder realizar búsquedas, análisis y visualización de datos en tiempo real.

- Elasticsearch: Motor de búsqueda distribuido, accesible mediante una interfaz API RESTful, basado en el motor de búsqueda Apache Lucene. Está desarrollado en Java y su funcionalidad principal es la de buscar e indexar información en diferentes formatos.
- Logstash: Motor de recolección de información que agrega, normaliza y distribuye información, en este caso a Elasticsearch. Logstash permite el uso de plugins, que dotan al sistema de una gran potencia en el proceso de agregación y normalización de los datos.
- Kibana: Es una herramienta web para la visualización y representación de datos, especializada para trabajar con un gran volumen de información en tiempo real. Permite la elaboración de Dashboards de monitorización, y la ejecución de consultas sobre el motor de búsqueda Elasticsearch, entre otras funcionalidades.

- **Beats:** Al stack también se suma el componente Beats, que permite la recolección e ingesta de información de forma descentralizada mediante la instalación de unos agentes en las máquinas a monitorizar.

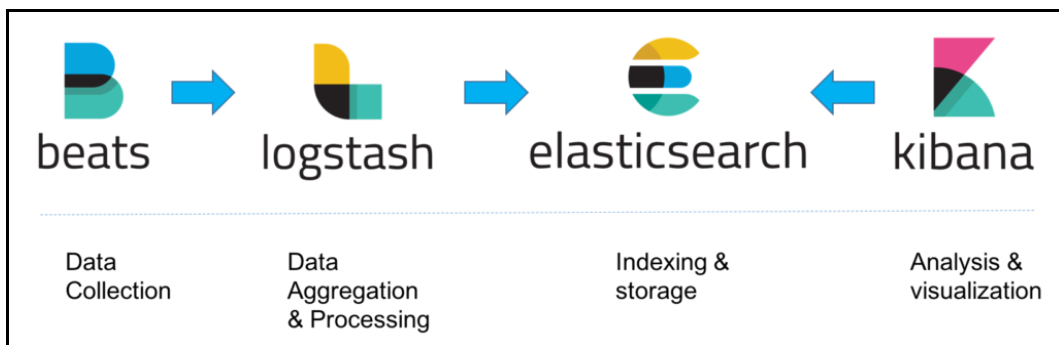


Figura 3. Relación componentes stack Elastic

El uso que se hará del stack Elastic será múltiple:

- Logstash para recuperar y normalizar la información generada por Zeek.
- A su vez, se aplicará la lógica para detectar actividades sospechosas a partir de ficheros diccionario generados por otro componente del sistema, Minemeld.
- Por último, se ingestará la información en Elasticsearch para su posterior tratamiento mediante la herramienta de visualización Kibana.

Página web del producto: <https://www.elastic.co>

Minemeld

Herramienta Web opensource, de la empresa PaloaltoNetworks, que facilita el tratamiento de indicadores, habitualmente listas de reputación, permitiendo poder agregarlos y transformarlos con el objetivo de ser consumidos por herramientas de terceros. La herramienta dispone de múltiples conectores, tanto para los datos de entrada, como para los datos de salida, que facilitan la integración con otros sistemas. Es una herramienta extensible, lo que permite crear nuevos conectores a sistemas no soportados de base.

El uso que se hará de Minemeld será el de proveer la información de listas de reputación de terceros, para poder ser consumidas por Logstash (stack Elastic) de cara a la detección de actividades sospechosas.

Página web del producto: <https://www.paloaltonetworks.com/products/secure-the-network/subscriptions/minemeld>

2.2 Enfoque de la solución

Se propone diseñar un sistema distribuido en diferentes máquinas virtuales, con el objetivo de aislar los componentes y que la arquitectura permita

escalado vertical u horizontal en caso de ser necesario. Además, con esta arquitectura distribuida, se garantiza un mayor gobierno del consumo de recursos de cada uno de los componentes y permitirá poder realizar acciones para asegurar capacity con mayor seguridad.

Por tanto, la solución requerirá de base dos máquinas:

- *Máquina 1: hostname uoc-server-ids*, donde se instalará el software Zeed IDS y se realizará la captura del tráfico de red a ser analizado.
- *Máquina 2: hostname uoc-server-elk*, donde se instalará el stack Elastic para la monitorización continua de la información.

Así mismo, se dispondrá de una máquina adicional que se encargará de gestionar las listas de reputación. En esta máquina, se instalará la plataforma Minemeld.

- *Máquina 3: hostname uoc-server-minemeld*.

Para completar la solución, se dispondrá de una máquina adicional para la realización de las pruebas de conectividad a servicios maliciosos.

- *Máquina 4: hostname upc-server-host*.

2.3 Inventario software

A continuación, se listan las herramientas y versiones utilizadas en el trabajo:

- Sistema operativo: Ubuntu Server 18.04 LTS.
- Software virtualización: Oracle VM Virtualbox 6.0.
- Sistema detección intrusos: Zeek IDS 3.0.0.
- Monitorización: Stack Elastic 7.4.0.
- Gestor listas reputación: Minemeld 0.9.64.
- Python versión 2.7.

El detalle de software, tamaños, repositorios y verificaciones de integridad de cada herramienta están disponibles en el Anexo I del trabajo.

2.4 Topología de red

Para llevar a cabo la implementación del sistema en su conjunto, es necesario definir una topología de red que incluya por un lado las diferentes máquinas con las herramientas que confirman la solución, y por otro lado la máquina host que vamos a monitorizar.

Para ello, se ha definido una red de área local (LAN), con un rango de

direcciones IP 192.168.1.0/24. Dentro de la red, estarán conectadas las diferentes máquinas con configuraciones de IP fijas.

Al ser una solución diseñada para validar el objetivo principal del trabajo, se ha optado por no incorporar al esquema de red ningún tipo de firewall, por lo que todas las máquinas tendrán visibilidad entre ellas y conectividad a Internet a través de la puerta de enlace con dirección IP privada 192.168.1.1.

A continuación, se detalla el esquema de la topología de red del sistema:

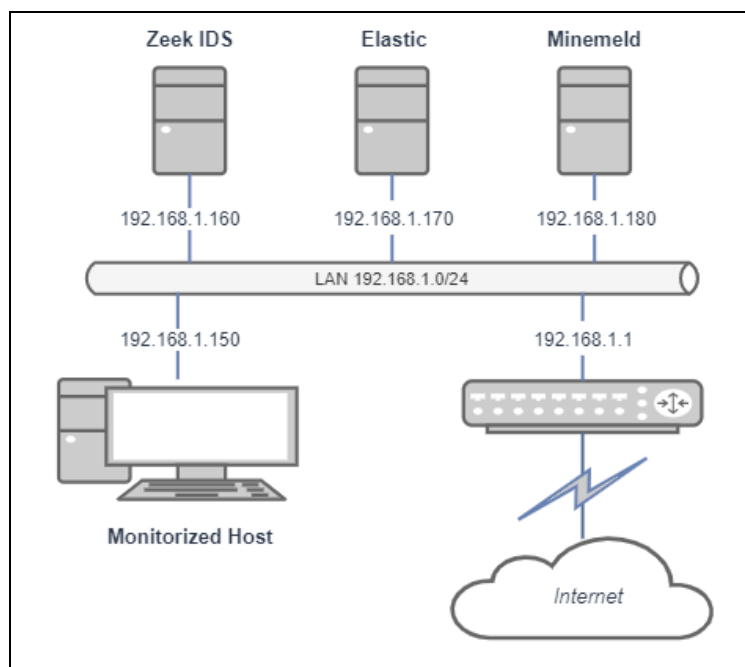


Figura 4. Topología de red de la solución

2.5 Captura de datos

El sistema se basa en la recopilación de información llevada a cabo por el sistema de detección de intrusos Zeek IDS.

Para la captura de datos, se desplegará una máquina virtual con sistema operativo Ubuntu Server 18.04, donde se realizará una instalación de Zeek IDS y se configurará para capturar el tráfico de la red de área local 192.168.1.0/24.

Para que el IDS pueda capturar el tráfico, se configurará la tarjeta de red de la máquina en modo promiscuo de forma que podamos tener acceso a todos los paquetes que viajen a través de la red. La configuración en modo promiscuo de la tarjeta de red se realiza mediante el software de virtualización Oracle VM VirtualBox, dentro de las características de red de la máquina virtual:



Figura 5. Configuración tarjeta red en software virtualización

La máquina donde estará instalado Zeek IDS tendrá el nombre *uoc-server-ids* y se le asignará la dirección IP 192.168.1.160.

Tras la configuración de red de la máquina virtual, y del arranque del sistema de detección de intrusos una vez aplicada la configuración, ya tendremos disponibles los logs de trazas que nuestro sistema está recopilando en base al tráfico de red analizado.

Zeek IDS dispone de multitud de ficheros de log disponibles, en función del tipo de tráfico que queremos monitorizar.

La lista completa de logs disponibles se encuentra en la documentación oficial del sistema: <https://docs.zeek.org/en/stable/script-reference/log-files.html>

Para el objetivo del trabajo, utilizaremos un subconjunto de los ficheros de log disponibles, que serán:

Fichero	Tipo	Descripción
conn.log	Red	Conexiones de protocolos tcp, udp, icmp
dns.log	Red	Peticiones a servicios dns
http.log	Red	Peticiones/Respuestas protocolo http
files.log	Ficheros	Resultado de análisis de ficheros

Tabla 2. Ficheros log de Zeek IDS utilizados en el trabajo

El contenido de cada uno de los ficheros de trazas se encuentra disponible en la página de documentación de Zeek IDS. La ingesta que se realizará en el stack de Elastic será de todas las trazas al completo de cada uno de los registros de los ficheros de log.

Se descarta el uso de logs del framework NetControl al no requerir el uso de hardware de red específico como switches, routers o firewalls en la arquitectura de la solución.

También se descarta el uso de logs del framework Detection dado que la implementación de la inteligencia en la detección de eventos maliciosos será realizada en Logstash.

La ingesta de logs en el Stack de Elastic será llevada a cabo mediante Logstash, que será instalado en la máquina donde reside el sistema IDS y que tendrá acceso a todos los ficheros de log generados.

2.6 Listas de reputación

Las listas de reputación son fuentes de información recopiladas por terceros, como organizaciones, comunidades, empresas e incluso particulares, que recogen diferente información para aplicar lo que comúnmente se conoce como *threat intelligence*, para la detección de actividades sospechosas en redes informáticas, y que pueden ser utilizadas por terceros para ser incorporadas en las herramientas de seguridad de los sistemas informáticos.

Las listas de reputación pueden servir para diferentes propósitos, en función de la información que contienen.

Para el desarrollo del sistema de detección objeto de este trabajo, se ha optado por el uso de las siguientes listas de reputación externas:

- **Alienvault IP reputation**
Fuente abierta de la empresa AlienVault, que incluye hosts considerados como maliciosos.
Url: <http://reputation.alienvault.com/reputation.data>
- **BBcan177 DNSBL**
Fuente abierta de la empresa Patreon con listado de dominios involucrados en actividades maliciosas.
Url: <https://gist.githubusercontent.com/BBcan177/4a8bf37c131be4803cb2/raw>
- **C&C Domains**
Fuente abierta de la empresa Bambenek Consulting con listado de dominios utilizados por diferentes malwares.
Url: <http://osint.bambenekconsulting.com/feeds/c2-dommasterlist.txt>
- **C&C IPs**
Fuente abierta de la empresa Bambenek Consulting con listado de ips utilizadas por diferentes malwares.
Url: <http://osint.bambenekconsulting.com/feeds/c2-ipmasterlist.txt>
- **Infosec**
Fuente abierta de la organización CERT de la Administración Pública Italiana con lista de urls que han servido ficheros binarios maliciosos.
Url: <https://infosec.cert-pa.it/analyze/listurls.txt>
- **Virusshare hashes md5**
Fuente abierta de la empresa virusshare con listado de hashes md5 de muestras de malware.
Url: https://virusshare.com/hashes/VirusShare_APT1.md5

Las listas de reputación indicadas, incluyen información que debe ser unificada de cara a poder ser utilizada con Logstash para aplicar el matching sobre los ficheros de log generados por Zeek IDS, además, se necesita de un sistema que permita una actualización periódica de la información para estar siempre actualizada.

Adicionalmente, al objeto de poder validar la solución del sistema y evitar realizar peticiones a máquinas consideradas como maliciosas durante las pruebas, se incluirán dos listas de reputación adicionales con un inventario de IP y dominios controlados, accesibles por http en la máquina *uoc-server-ids*.

- Lista de IPs para pruebas
Url: <http://192.168.1.160/badips.txt>
- Lista de Dominios para pruebas
Url: <http://192.168.1.160/baddomains.txt>
- Lista de URLs para pruebas
Url: <http://192.168.1.160/badurls.txt>
- Lista de Hash MD5 para pruebas
Url: <http://192.168.1.160/badhashes.txt>

Para facilitar la gestión de las listas de reputación, se ha optado por el uso de la herramienta Minemeld de la empresa Paloalto Networks, que provee de una interfaz gráfica mediante la cual se pueden configurar feeds de entrada de diferentes listas de reputación, y generar unos endpoints de salida con la información unificada para ser utilizados por otras herramientas.

La máquina donde estará instalado Minemeld tendrá el nombre *uoc-server-minemeld* y se le asignará la dirección IP 192.168.1.180.

El funcionamiento de Minemeld se basa en la definición de nodos, que pueden ser de diferentes tipos:

- Miner. Nodos de entrada que se dan de alta para recuperar los datos de las listas de reputación. Hay un nodo miner por cada lista de reputación a ingestar, y la carga de información se realiza mediante protocolo http.
- Processor. Nodos que reciben la información de uno o varios nodos miner, y permiten aplicar filtros sobre la información recibida.
- Output. Nodos que reciben información de processors y que permiten exportar o expone la información para ser consumida por otros sistemas.

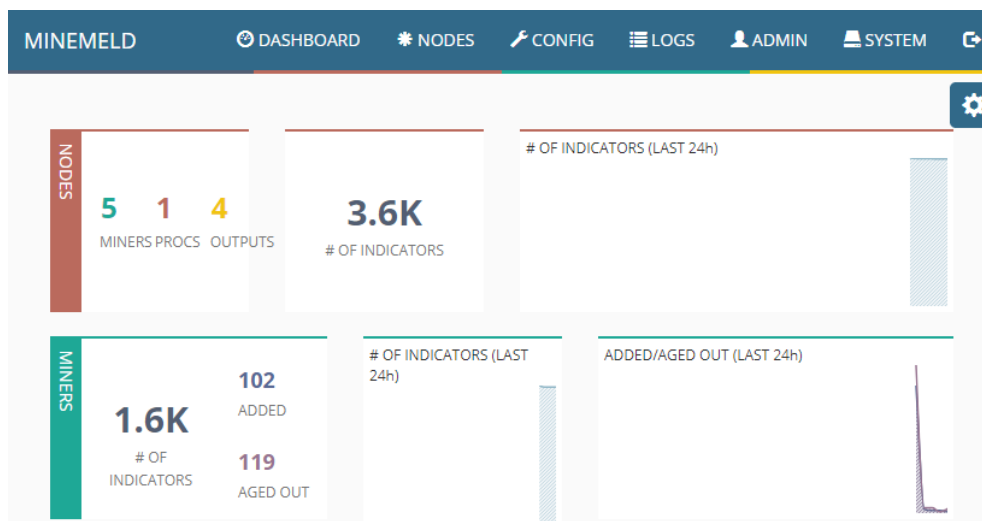


Figura 6. Captura de pantalla de la plataforma Minemeld

La configuración de nodos que se aplicará en Minemeld para el propósito del trabajo es:

Nodos Miner

Se darán de alta los siguientes nodos Miner, tanto como listas de reputación se van a utilizar en el sistema.

Nombre	Tipo	Caducidad (seg)
alienvault-ip-input	Miner	14.400 (4 horas)
bbcan-domain-input	Miner	14.400
cc-domain-input	Miner	14.400
cc-ip-input	Miner	14.400
infosec-url-input	Miner	14.400
virusshare-hash-input	Miner	14.400
uoc-ip-input	Miner	60
uoc-domain-input	Miner	60
uoc-url-input	Miner	60
uoc-hash-input	Miner	60

Tabla 3. Configuración nodos miner en Minemeld

El refresco por defecto se ha configurado en 14.400 segundos (4 horas) para los feeds externos, y 60 segundos para los feeds internos.

Nodos Processor

Se darán de alta los siguientes nodos Processor:

Nombre	Nodos Miner
ips-processor	alienvault-ip-input, cc-ip-input, uoc-ip-input
domains-processor	bbcan-domain-input, cc-domain-input, uoc-domain-input
urls-processor	infosec-url-input, uoc-url-input
md5-processor	virusshare-hash-input, uoc-hash-input

Tabla 4. Relación nodos processor y nodos miner en Minemeld

Los prototipos que se utilizarán para cada uno de los nodos processor son:

Nombre	Prototype
ips-processor	stdlib.aggregatorIPv4Generic
domains-processor	stdlib.aggregatorDomain
urls-processor	stdlib.aggregatorURL
md5-processor	stdlib.aggregatorMD5

Tabla 5. Prototipos utilizados en nodos processor en Minemeld

Todos los nodos tendrán por defecto activada la propiedad drop con valor all, para realizar un procesado completo de los datos proveniente de los nodos Miner.

Nodos Output

Se darán de alta los siguientes nodos Output:

Nombre	Nodos Processor
ips-output	ips-processor
domains-output	domains-processor
urls-output	urls-processor
hash-output	md5-processor

Tabla 6. Relación nodos output y nodos processos en Minemeld

Los nodos output volcarán los datos en una base de datos de tipo Redis, cuya información podrá ser consumida a través de la invocación a los siguientes endpoints http:

Nombre	Url
ips-output	https://uoc-server-minemeld/feeds/ips-output
domains-output	https://uoc-server-minemeld/feeds/domains-output
url-output	https://uoc-server-minemeld/feeds/url-output
hash-output	https://uoc-server-minemeld/feeds/hash-output

Tabla 7. Url de los feed generados en Minemeld

Todos los nodos tendrán por defecto activada la propiedad drop con valor all, para realizar un output completo de los datos proveniente de los nodos processor, evitando que se acumule información repetida.

El acceso web a la plataforma Minemeld estará securizado por el mecanismo de autenticación y autorización incluido en la propia herramienta. Para este trabajo se habilitará el acceso de un usuario con permisos de administración.

2.7 Ingesta de información

La ingesta de la información será realizada por el componente Logstash, que estará disponible en la máquina uoc-server-ids, y cuya función esencial para el trabajo será la de recopilar la información de Zeek IDS e ingestarla en Elasticsearch, previa ejecución del proceso de *matching* de los registros con las listas de reputación generadas con Minemeld.

Para el *matching* de los registros con las listas de reputación, se utilizará el plugin de filtrado de traducción de Logstash (Translate filter plugin), que permitirá la inyección de datos vía sistema de ficheros en formato diccionario, y generar una salida a partir de los datos de entrada, en este caso de Zeek IDS, incluyendo ciertas marcas en los registros que permitan identificar el tráfico de red malicioso.

Para que Logstash pueda realizar el *matching*, es necesario volcar los feeds generados por Minemeld al sistema de ficheros. Para realizar dicho volcado, se implementarán 4 programas en Python que invocarán a las URLs de cada uno de los feeds y volcará los contenidos en diferentes ficheros que serán los que se utilizarán como diccionarios en el sistema.

En función del feed, los datos de los ficheros generados por los scripts tendrán diferente contenido:

- Fichero de direcciones ips – nombre: ip-dictionary: contendrá una lista de direcciones IP separadas por un salto de línea.
- Fichero de dominios – nombre: domain-dictionary: contendrá una lista de dominios separadas por un salto de línea.
- Fichero de urls – nombre: url-dictionary: contendrá una lista de urls separadas por un salto de línea.
- Fichero de hashes md5 – nombre: file-dictionary: contendrá una lista de hashes md5 separadas por un salto de línea.

Se configurará el sistema para ejecutar cada uno de los scripts con una periodicidad de 4 horas, con la posibilidad de ejecutar los scripts de forma manual para la ejecución de las pruebas de validación.

Para la ingesta de los datos en elasticsearch, se utilizará el plugin de salida de Elasticsearch provisto con Logstash, que realizará la ingesta de los datos mediante invocaciones http en el puerto por defecto de elasticsearch 9200.

La relación entre logs de Zeek IDS y las listas de reputación será la siguiente:

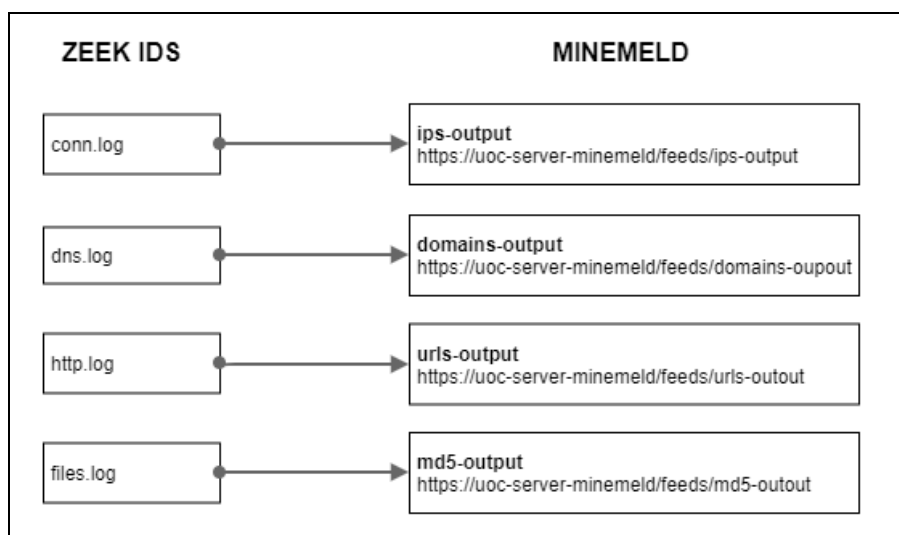


Figura 7. Relación logs Zeek IDS y listas Minemeld

Para poder identificar los registros maliciosos, cuando haya un match en cualquiera de los registros con las listas de reputación, se incluirá un nuevo tag en el registro de entrada para así informarlo, de forma que se consultable mediante Elasticsearch y Kibana.

El tag que se añadirá tendrá el nombre “malicious”.

2.8 Visualización de datos

La visualización de los datos se realizará en un dashboard que se basará en el stack Elastic. En concreto, se utilizarán los componentes Elasticsearch, como motor de búsqueda basado en Lucene, y Kibana, para la representación gráfica de la información.

La máquina donde se instalará Elastic tendrá el nombre *uoc-server-elk* y se le asignará la dirección IP 192.168.1.170.

Una vez que la información es ingestada en Elasticsearch, se podrán dar de alta los diferentes índices en Kibana para la explotación de la información.

Se darán de alta tantos índices como relaciones existan entre los logs de Zeek IDS y los ficheros de las listas de reputación generados. En total, 4 índices diferentes.

Índice Elasticsearch	Fichero Traza	Lista reputación
zeek-conn-logs	conn.log	ips-output
zeek-dns-logs	dns.log	domains-output
zeek-http-logs	http.log	urls-output
zeek-files-logs	files.log	md5-output

Tabla 8. Relación índices elasticsearch, ficheros de trazas y listas de reputación

Para el trabajo, se habilitarán una serie de dashboards de monitorización continua, donde se visualizarán datos para cada uno de los logs ingestados, y se habilitará una zona para detectar registros marcados como maliciosos.

En total, se habilitarán 4 dashboards:

- Dashboard Visión Global:
 - Mapa de calor del mundo con conexiones destino.
 - Gráfico circular con Países de las conexiones destino.
 - Total bytes de datos de entrada.
 - Total bytes de datos de salida.
 - Gráfico de barras con protocolos detectados.
 - Total conexiones maliciosas detectadas.
 - Lista de conexiones maliciosas detectadas.

Con este primer dashboard, detectaremos tráfico malicioso a partir de la dirección IP origen/destino de las diferentes conexiones, para cualquiera de los protocolos soportados.

- Dashboard Dominios:
 - Total de consultas dns realizadas.
 - Top 10 de consultas dns realizadas.
 - Total resoluciones dns maliciosas detectadas.
 - Lista de consultas dns maliciosas detectadas.

En este segundo dashboard, se monitorizará la consulta a servicios DNS para la resolución de nombres de dominio, y mostraremos consultas de nombres consideradas como maliciosas.

- Dashboard URLs:
 - Total de peticiones http realizadas.
 - Gráfico de barras con códigos de respuesta y totales.
 - Top 10 de URLs visitadas.
 - Total peticiones http maliciosas detectadas.
 - Lista de peticiones http maliciosas detectadas.

Este dashboard monitorizará el tráfico web, y se mostrará las peticiones realizadas y los códigos de respuesta obtenidos, y también el total de peticiones maliciosas detectadas y las últimas 10 detectadas.

- Dashboard Ficheros:
 - Total de ficheros descargados.
 - Total bytes de ficheros descargados.
 - Top 10 de extensiones de ficheros descargadas.
 - Total ficheros maliciosos descargados.
 - Lista ficheros maliciosos descargados.

En este último dashboard se mostrará información de la descarga de ficheros, y mostraremos datos de la descarga de ficheros considerados como maliciosos según las fuentes de reputación utilizadas.

2.9 Arquitectura final de la solución

Una vez completado el análisis de los diferentes componentes que formarán la solución a implementar, y los mecanismos de integración entre todos ellos, ya es posible definir lo que será la arquitectura completa de la solución, que será la siguiente:

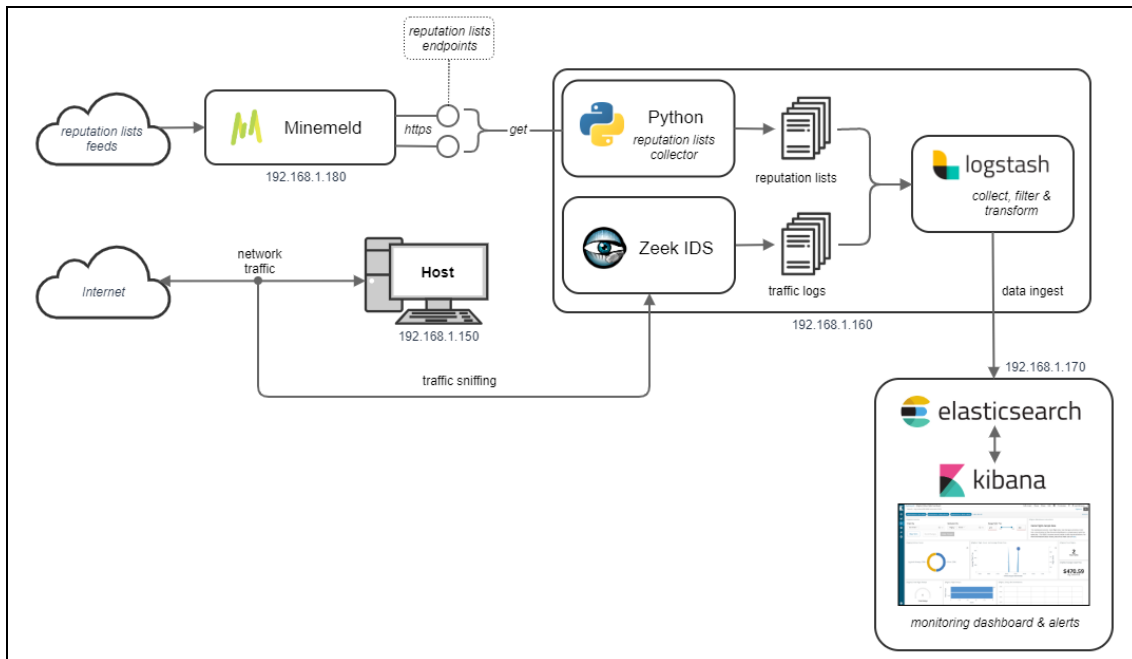


Figura 8. Esquema de la arquitectura de la solución

En el esquema, se muestran los componentes que se dispondrán en cada una de las máquinas que conforman el sistema en su conjunto.

El proceso que se seguirá será el cliente:

1. Minemeld recupera la información de las listas de reputación y las ofrece a través de endpoints http que son invocados por los programas Python que se ejecutarán periódicamente mediante un cron y volcará los datos al sistema de ficheros.
2. En paralelo, Zeek IDS recuperará la información del tráfico de red y volcará los datos en los ficheros de log al filesystem de la máquina.
3. Mediante Logstash, se cargarán los ficheros de trazas de Zeek IDS, y se realizará el *matching* con los ficheros generados por Python con datos provenientes de las listas de reputación.
4. Logstash al detectar matchings, incluirá un tag adicional en los datos que se ingestarán en elasticsearch para que posteriormente se puedan filtrar y visualizar en Kibana.
5. Los datos ingestados en elasticsearch se visualizarán en unos dashboards de monitorización para la detección ágil de anomalías en el tráfico de red.
6. La máquina host será la encargada de simular tráfico con el objetivo de validar el correcto funcionamiento de todo el sistema de forma integrada.

2.10 Plan de pruebas

El Plan de pruebas se ha definido en varias fases:

- En la primera fase del plan de pruebas, se validará la integración de Zeek IDS con el stack Elastic. Por tanto, se llevará a cabo una primera prueba que consistirá en generar tráfico a partir de la máquina host, y ver que la información es recopilada por Logstash e ingestada en Elasticsearch para su consulta a través de Kibana.

En esta primera fase del plan de pruebas, no se tendrán en cuenta los datos provenientes de las listas de reputación.

La generación del tráfico de muestra en la máquina Host se realizará mediante un script Python que realizará diferentes conexiones a máquinas destino.

- En la segunda fase del plan de pruebas, se validará el circuito de las listas de reputación, que implicará la correcta generación de los datos de salida a partir de las listas de reputación seleccionadas, y la generación de los ficheros de salida con los programas Python desarrollados.

A su vez, se validará que los datos generados en los ficheros de salida se van actualizando correctamente. Para ello, se introducirán registros de forma controlada en los feeds de pruebas para comprobar que en los ficheros de salida los registros de prueba con direcciones IP, dominios, urls y hash de ficheros, se han recuperado correctamente.

- La tercera fase del plan de pruebas validará la parte del matching de los registros de Zeek IDS con los ficheros generados a partir de las listas de reputación, y la correcta ingesta en Elasticsearch y visualización en Kibana.

La generación del tráfico de muestra en la máquina Host se realizará mediante la ejecución del mismo programa Python creado para las pruebas.

- La cuarta y última fase del plan de pruebas consistirá en visualizar la información en tiempo real del tráfico capturado a través de los Dashboards de visualización, y verificar que se muestra de forma correcta los datos de tráfico malicioso detectados en cada uno de ellos.

La generación del tráfico de muestra en la máquina Host se realizará mediante la ejecución de un Shell script, incluido en el Anexo VII del trabajo.

3. Construcción del sistema

3.1 Componentes software

En este apartado se describe el proceso de instalación de los diferentes componentes software que componen el sistema, así como las diferentes configuraciones que aplican a cada uno de ellos.

Para cada componente software, se detallará:

- El proceso de instalación en base a las versiones de los componentes definidos en la fase de análisis del trabajo.
- Configuraciones que se aplicarán en cada componente acordes al servicio que van a dar dentro del sistema.
- Validaciones preliminares de la correcta instalación y configuración de cada uno de los componentes.

La instalación y configuración de los componentes se ha ordenado de forma que facilite la construcción del sistema, evitando en la medida de lo posible dependencias entre ellos.

El detalle de la instalación, configuración y validación del sistema operativo en las diferentes máquinas virtuales se ha detallado en el Anexo II del trabajo.

3.1.1 Zeek IDS

3.1.1.1 Instalación

La instalación del software Zeek IDS se realizará a partir del repositorio de paquetes de opensuse.org, que ya ofrece un paquete zeek instalable a través del gestor de paquetes APT para la versión del sistema operativo Ubuntu instalado.

A continuación, se listan los comandos a realizar para la instalación:

```
$ sudo sh -c "echo 'deb
http://download.opensuse.org/repositories/security:/zeek/xUbuntu_18.04/ /' >
/etc/apt/sources.list.d/security:zeek.list"
$ wget -nv
https://download.opensuse.org/repositories/security:zeek/xUbuntu_18.04/Release.key -O Release.key
$ sudo apt-key add - < Release.key
$ sudo apt-get update
$ sudo apt-get install zeek
```

Una vez ejecutado el comando de instalación, el último de todos, se requerirá introducir una serie de parámetros de configuración para la funcionalidad de envío de correos. Esta configuración no será relevante para el propósito del trabajo puesto que no se ha definido ninguna funcionalidad específica que requiera el envío de correos electrónicos desde Zeek.

La instalación del software Zeek IDS se realizará en el siguiente directorio,

```
$ /opt/zeek
```

donde encontraremos los ejecutables del software, así como ficheros de configuración, ficheros de log, y otros ficheros requeridos por Zeek para su correcto funcionamiento.

3.1.1.2 Configuración

A continuación, se procede a la configuración de Zeek.

En primer lugar, indicaremos la red que queremos monitorizar. Para ello, se editará el fichero,

```
$ /opt/zeek/etc/networks.cfg
```

e introduciremos la siguiente configuración:

```
192.168.1.0/24      Private IP space
```

En segundo lugar, indicaremos la interfaz de red que se utilizará para la detección del tráfico de red a analizar. Para ello, se editará el fichero,

```
$ /opt/zeek/etc/nodes.cfg
```

e introduciremos la siguiente configuración:

```
[zeek]
type=standalone
host=localhost
interface=enp0s3
```

En tercer lugar, para facilitar las pruebas y acotar la información generada en los logs, indicaremos a Zeek qué máquina en concreto queremos monitorizar. Para ello, se editará el fichero,

```
$ /opt/zeek/etc/zeekctl.cfg
```

e introduciremos al final de todo el siguiente parámetro:

```
# Custom configuration
ZeekArgs = -f 'host 192.168.1.150'
```

Una vez finalizada la configuración, procedemos a aplicar los cambios y a inicializar el sistema de detección de intrusos.

El comando para aplicar los cambios de configuración es:

```
$ zeekctl deploy
```

Este comando, aplicará los cambios en las políticas, y además arrancará el sistema. Se adjunta log obtenido una vez ejecutado el comando:

```
root@uoc-server-ids:/opt/zeek/etc# zeekctl deploy
checking configurations ...
installing ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/site ...
removing old policies in /opt/zeek/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
stopping ...
stopping zeek ...
starting ...
starting zeek ...
root@uoc-server-ids:/opt/zeek/etc#
```

Figura 9. Log de arranque del sistema Zeek IDS

En último lugar, configuraremos cron para lanzar una tarea cada 5 minutos, que verificará si el sistema zeek está arrancado correctamente, y en caso contrario, que se proceda a su arranque, para garantizar así continuidad en el servicio.

El alta del cron se realizará mediante el comando,

```
$ sudo crontab -e
```

y la tarea que se dará de alta será la siguiente:

```
*/5 * * * * /opt/zeek/bin/zeekctl cron
```

Una vez demos de alta la tarea cron, el sistema de detección de intrusos estará completamente configurado y listo para utilizarse para la detección de tráfico de red.

3.1.1.3 Validación

Una vez realizado el deploy y arranque de Zeek, verificamos con el comando

```
$ zeekctl status
```

que el sistema está correctamente arrancado. El resultado ejemplo del comando debería ser una salida por consola indicando un estado de “running”.

```
root@uoc-server-ids:/opt/zeek/etc# zeekctl status
Name      Type      Host      Status  Pid   Started
zeek      standalone localhost running 7249 19 Nov 17:55:14
root@uoc-server-ids:/opt/zeek/etc#
```

Figura 10. Validación estado de servicio Zeek IDS

A continuación, verificaremos que el sistema Zeek genera los logs de salida que necesitamos para el propósito del trabajo, que son:

- conn.log
- http.log
- dns.log
- files.log

Para ello, realizaremos unas invocaciones de prueba desde la máquina host (*uoc-server-host*) a diferentes servicios web, concretamente se ejecutarán los siguientes comandos:

```
$ curl http://google.es

$ wget http://file-examples.com/wp-
content/uploads/2017/10/file_example_JPG_100kB.jpg
```

Tras la ejecución de las invocaciones de prueba, validamos que los logs se han generado correctamente. Para ello, accedemos a la carpeta

```
$ /opt/zeek/Logs/current
```

y listamos los ficheros del directorio, cuyo resultado nos debería dar una lista de ficheros que debería incluir al menos los ficheros conn.log, dns.log, files.log y http.log que son los ficheros que se utilizarán para la monitorización.

En la siguiente figura, a modo de ejemplo. se muestra los registros de fichero http.log:

```
#separator \x09
#set_separator
#empty_field (empty)
#unset_field
#path http
#open 2019-11-19-18-09-44
#fields ts uid id.orig_h id.orig_p id.resp_h id.resp_p trans_depth method host uri refe
#types time string addr port addr port count string string string string string string count coun
1574183384.711142 CQYbXQ18r0Pk0YJ0Tf 192.168.1.150 46216 216.58.211.35 80 1 GET google.es /
1574183410.503144 CxIdUX2EbbwPo9HN03 192.168.1.150 55244 185.135.88.81 80 1 GET file-examples.com
```

Figura 11. Ejemplo contenido fichero de trazas http.log de Zeek IDS

3.1.1.4 Script custom

Para incrementar los datos registrados en el fichero de trazas files.log generado por Zeek, se ha implementado un script custom de Zeek para informar, para cada registro, el nombre del fichero detectado, la URI y el hostname remoto desde el cual se ha descargado.

Esta información es de mucha utilidad para el diagnóstico de actividades sospechosas relacionadas con la descarga de ficheros, y las trazas por defecto facilitadas por Zeek no ofrecían toda la información necesaria para llevar a cabo los diagnósticos.

Por ello, se ha desarrollado un script que amplía la información generada en el fichero de trazas files.log para incluir, el nombre del fichero, la extensión, el dominio de la máquina remota, y la URI.

El script se ha añadido al siguiente fichero:

```
/opt/zeek/share/zeek/site/Local.zeek
```

El bloque de código está disponible en el Anexo VI del trabajo.

3.1.2 Minemeld

3.1.2.1 Instalación

La instalación de Minemeld se realizará utilizando el motor de automatización Ansible, a partir del código fuente de Minemeld a disposición de los usuarios en el repositorio GitHub, disponible en la siguiente dirección:

- Repositorio:
<https://github.com/PaloAltoNetworks/minemeld-ansible.git>

La empresa PaloAltoNetworks ofrece un playbook de Ansible para llevar a cabo la instalación del software con todas las dependencias.

A continuación, se listan los comandos a ejecutado para llevar a cabo el proceso de instalación de Minemeld, así como el software adicional necesario.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install -y gcc git python-minimal python2.7-dev libffi-dev
libssl-dev make
$ wget https://bootstrap.pypa.io/get-pip.py
$ sudo -H python get-pip.py
$ sudo -H pip install ansible
$ git clone https://github.com/PaloAltoNetworks/minemeld-ansible.git
$ cd minemeld-ansible
$ ansible-playbook -K -i 127.0.0.1, Local.yml
```

Una vez finalizada la instalación, el software estará instalado y arrancado, y verificaremos su estado mediante el comando,

```
$ service minemeld status
```

que dará como resultado el estado “active (running)”.

Por último, se deberá definir las credenciales de acceso de la herramienta web para el usuario admin.

Para ello, instalaremos el paquete apache2-utils:

```
$ sudo apt install apache2-utils
```

A continuación, procederemos a definir la nueva contraseña para el usuario admin mediante el siguiente comando:

```
$ sudo htpasswd /opt/minemeld/Local/config/api/wsgi.htpasswd admin
```

```
New password:  
Re-type new password:  
Updating password for user admin
```

Tras el cambio, se reiniciaría el servicio de Minemeld, y verificaríamos de nuevo el estado del proceso para asegurarnos que está “active (running)”.

```
$ service minemeld restart
```

Para validar el acceso a la herramienta web, y la correcta configuración de la contraseña del usuario admin, quedaría entrar, mediante un navegador web, a la IP de la máquina donde hemos realizado la instalación y nos autenticaríamos con las credenciales definidas.

3.1.2.2 Configuración

Una vez instalado Minemeld, lo que haremos en primer lugar es eliminar todos los nodos creados por defecto en la herramienta, ya que partiremos de una configuración limpia para incluir únicamente las listas de reputación indicadas en la fase de análisis.

Para ello, accederemos a la sección *Config*, y sobre la tabla de nodos, eliminaremos todos los registros hasta que la tabla quede vacía. Una vez eliminados, haremos clic en el botón *Commit* para aplicar los cambios, y verificaremos en la sección de Dashboard de que todos los indicadores están a 0, para todos los nodos.

A continuación, procederemos a dar de alta los nuevos prototipos a partir de los cuales crearemos los nodos de nuestro sistema. El alta de nuevos nodos se realiza desde el apartado de *Config*.

Dentro del apartado *Config*, haremos click sobre el botón “*browse prototypes*” que mostrará todos los prototipos existentes en la herramienta. El alta de los prototipos para el sistema se realizará partiendo de los ya existentes en la herramienta.

Es necesario que cada vez que se realice el alta de un nuevo prototipo, o nodo, o cualquier cambio en el sistema, se realice un *Commit* para aplicar los cambios. De lo contrario, la configuración podrá dar error.

Nodos Miner

En función de las listas de reputación utilizadas, y el formato de la información contenida, los prototipos de nodos Miner los podremos clasificar de dos tipos:

- Nodos de tipo CSV: La información de las listas de reputación de origen tienen un formato CSV, es decir, que para cada registro hay diferente información que está separada por un carácter de separación, habitualmente comas.

Para estos nodos, se utilizará la clase Minemeld `minemeld.ft.csv.CSVFT`, que permitirá recuperar datos a partir de peticiones HTTP, y con el formato por registro ya mencionado.

- Nodos de tipo HTTP estándar: La información de los registros para las listas de reputación origen en este caso sólo contienen un dato por fila.

Para estos nodos, se utilizará la clase Minemeld `minemeld.ft.http.HttpFT`, que permitirá recuperar los datos a partir de peticiones HTTP.

A continuación, se especifica la configuración para cada uno de los nodos Miner dados de alta en el sistema. Para todos ellos, como se parte de un prototipo existente, cuando se seleccione el prototipo existente, se hará clic en la opción de `new`, para poder introducir la configuración que corresponde:

Nodo `alientvault-ip-input`

- Prototipo origen: `alienvault.reputation`
- Clase: `minemeld.ft.csv.CSVFT`
- Nombre: `alientvault-ip-input`
- Indicator types: IPv4
- Config:

```
attributes:
  confidence: 80
  share_level: green
  type: IPv4
delimiter: '#'
fieldnames:
- indicator
interval: 14400
source_name: alienvault
url: http://reputation.alienvault.com/reputation.data
```

Nodo `bbcان-domain-input`

- Prototipo origen: `blocklist_de.all`
- Clase: `minemeld.ft.http.HttpFT`
- Nombre: `bbcان-domain-input`
- Indicator types: domain
- Config:

```
attributes:
  confidence: 80
  share_level: green
  type: domain
ignore_regex: ^#
interval: 14400
source_name: bbcان177
url: https://gist.githubusercontent.com/BBcan177/4a8bf37c131be4803cb2/raw
```

Nodo cc-domain-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: cc-domain-input
- Indicator types: domain
- Config:

```
attributes:  
  confidence: 80  
  share_level: green  
  type: domain  
ignore_regex: ^#  
interval: 14400  
source_name: bambenekconsulting  
url: http://osint.bambenekconsulting.com/feeds/c2-dommasterlist.txt
```

Nodo cc-ip-input

- Prototipo origen: alienvault.reputation
- Clase: minemeld.ft.csv.CSVFT
- Nombre: cc-ip-input
- Indicator types: IPv4
- Config:

```
attributes:  
  confidence: 80  
  share_level: green  
  type: IPv4  
delimiter: ','  
fieldnames:  
- indicator  
interval: 14400  
source_name: bambenekconsulting  
url: http://osint.bambenekconsulting.com/feeds/c2-ipmasterlist.txt
```

Nodo infosec-url-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: infosec-url-input
- Indicator types: URL
- Config:

```
attributes:  
  confidence: 80  
  share_level: green  
  type: URL  
ignore_regex: ^#  
interval: 14400  
source_name: infosec  
url: https://infosec.cert-pa.it/analyze/listurls.txt
```

Nodo virusshare-hash-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: virusshare-hash-input
- Indicator types: md5
- Config:

```
attributes:  
  confidence: 80  
  share_level: green  
  type: md5  
ignore_regex: ^#  
interval: 14400  
source_name: virtusshare-md5  
url: https://virusshare.com/hashes/VirusShare_APT1.md5
```

Nodo uoc-ip-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: uoc-ip-input
- Indicator types: IPv4
- Config:

```
attributes:  
  confidence: 80  
  share_level: green  
  type: IPv4  
ignore_regex: ^#  
interval: 60  
source_name: local-ip  
url: http://192.168.1.160/badips.txt
```

Nodo uoc-domain-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: uoc-domain-input
- Indicator types: domain
- Config:

```
attributes:  
  confidence: 80  
  share_level: green  
  type: domain  
ignore_regex: ^#  
interval: 60  
source_name: local-domains  
url: http://192.168.1.160/baddomains.txt
```

Nodo uoc-url-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: uoc-url-input
- Indicator types: URL
- Config:

```
attributes:  
  confidence: 100  
  share_level: green  
  type: URL  
ignore_regex: ^#  
interval: 60  
source_name: local-urls  
url: http://192.168.1.160/badurls.txt
```

Nodo uoc-hash-input

- Prototipo origen: blocklist_de.all
- Clase: minemeld.ft.http.HttpFT
- Nombre: uoc-hash-input
- Indicator types: md5
- Config:

```
attributes:  
  confidence: 100  
  share_level: green  
  type: md5  
ignore_regex: ^#  
interval: 60  
source_name: local-md5  
url: http://192.168.1.160/badhashes.txt
```

Nodos Processor

Para el alta de los nodos processor, se utilizarán las diferentes librerías que Minemeld pone a disposición en función del tipo de información que manejan.

Para este trabajo, se utilizarán cuatro tipos de librerías processor, que son:

- stdlib.aggregatorIPv4Generic: para agregar los nodos input de tipo IPv4.
- stdlib.aggregatorDomain: para agregar los nodos input de tipo domain.
- stdlib.aggregatorURL: para agregar los nodos input de tipo URL
- stdlib.aggregatorMD5: para agregar los nodos input de tipo md5.

A continuación, se especifica la configuración para cada uno de los nodos processor que se darán de alta en el sistema:

Nodo ips-processor

- Librería origen: stdlib.aggregatorIPv4Generic

- Clase: minemeld.ft.ipop.AggregateIPv4FT
- Nombre: ips-processor
- Indicator types: IPv4
- Config:

```

infilters:
- actions:
  - accept
  conditions:
  - __method == 'withdraw'
  name: accept withdraws
- actions:
  - accept
  conditions:
  - type == 'IPv4'
  name: accept IPv4
- actions:
  - drop
  name: drop all

```

Nodo domains-processor

- Librería origen: stdlib.aggregatorDomain
- Clase: minemeld.ft.op.AggregateFT
- Nombre: domains-processor
- Indicator types: domain
- Config:

```

infilters:
- actions:
  - accept
  conditions:
  - __method == 'withdraw'
  name: accept withdraws
- actions:
  - accept
  conditions:
  - type == 'domain'
  name: accept domain
- actions:
  - drop
  name: drop all

```

Nodo urls-processor

- Librería origen: stdlib.aggregatorURL
- Clase: minemeld.ft.op.AggregateFT
- Nombre: urls-processor
- Indicator types: URL
- Config:

```

infilters:
- actions:
  - accept

```

```

conditions:
- __method == 'withdraw'
name: accept withdraws
- actions:
- accept
conditions:
- type == 'URL'
name: accept URL
- actions:
- drop
name: drop all

```

Nodo md5-processor

- Librería origen: stdlib.aggregatorMD5
- Clase: minemeld.ft.op.AggregateFT
- Nombre: md5-processor
- Indicator types: md5
- Config:

```

infilters:
- actions:
- accept
conditions:
- __method == 'withdraw'
name: accept withdraws
- actions:
- accept
conditions:
- type == 'md5'
name: accept MD5
- actions:
- drop
name: drop all

```

Nodos Output

El proceso de alta de los nodos de salida, es exactamente el mismo que en el resto de nodos. Utilizaremos un prototipo de nodo de salida existente a partir del cual crearemos los prototipos para el sistema.

Para todos los nodos de salida, se dispondrá de un feed con acceso a través de protocolo HTTP seguro, e internamente se utilizará una clase de tipo minemeld.ft.redis.RedisSet, que internamente creará una base de datos en memoria y se manejará la información mediante este sistema.

Nodo ips-output

- Librería origen: stdlib.feedHCGreen
- Clase: minemeld.ft.redis.RedisSet
- Nombre: ips-output
- Indicator types: IPv4
- Config:

```

infilters:
- actions:
  - accept
  conditions:
  - __method == 'withdraw'
  name: accept withdraws
- actions:
  - accept
  conditions:
  - confidence > 75
  - share_level == 'green'
  name: accept confidence > 80 and share level green
- actions:
  - drop
  name: drop all

```

Nodo domains-output

- Librería origen: stdlib.feedHCGreen
- Clase: minemeld.ft.redis.RedisSet
- Nombre: domains-output
- Indicator types: domain
- Config:

```

infilters:
- actions:
  - accept
  conditions:
  - __method == 'withdraw'
  name: accept withdraws
- actions:
  - accept
  conditions:
  - confidence > 75
  - share_level == 'green'
  name: accept confidence > 80 and share level green
- actions:
  - drop
  name: drop all

```

Nodo urls-output

- Librería origen: stdlib.feedHCGreen
- Clase: minemeld.ft.redis.RedisSet
- Nombre: urls-output
- Indicator types: URL
- Config:

```

infilters:
- actions:
  - accept
  conditions:
  - __method == 'withdraw'
  name: accept withdraws
- actions:

```

```

- accept
conditions:
- confidence > 75
- share_level == 'green'
name: accept confidence > 80 and share level green
- actions:
- drop
name: drop all

```

Nodo hash-output

- Librería origen: `stdlib.feedHCGreen`
- Clase: `minemeld.ft.redis.RedisSet`
- Nombre: `hash-output`
- Indicator types: URL
- Config:

```

infilters:
- actions:
- accept
conditions:
- __method == 'withdraw'
name: accept withdraws
- actions:
- accept
conditions:
- confidence > 75
- share_level == 'green'
name: accept confidence > 80 and share level green
- actions:
- drop
name: drop all

```

Una vez completada la generación de los nodos prototipo para el sistema, tendremos que ir a cada uno de los prototipos creados, y dentro de ellos, darle a la acción de clonar (*CLONE*) y asignar el correspondiente nombre de nodo para cada uno de ellos.

Es necesario que cada vez que se realice el alta de un nuevo prototipo, o nodo, o cualquier cambio en el sistema, se realice un *Commit* para aplicar los cambios. De lo contrario, la configuración podrá dar error.

Para los nodos processor, se deberán seleccionar los diferentes nodos Miner según el tipo de objeto gestionado, según la siguiente tabla:

Nodos Processor	Nodos Miner
ips-processor	alienvault-ip-input, cc-ip-input, uoc-ip-input
domains-processor	bbcan-domain-input, cc-domain-input, uoc-domain-input
urls-processor	infosec-url-input, uoc-url-input
md5-processor	virusshare-hash-input, uoc-hash-input

Tabla 9. Relación nodos processor con nodos miner en Minemeld

Y para los nodos output, se deberán seleccionar como entrada los nodos que correspondan en función del tipo de objeto gestionado, según la siguiente tabla:

Nodos Output	Nodos Processor
ips-output	ips-processor
domains-output	domains-processor
urls-output	urls-processor
hash-output	md5-processor

Tabla 10. Relación nodos output con nodos processor en Minemeld

Al finalizar la configuración, en el apartado *Config* tendremos todos los nodos creados en nuestro sistema, y los nodos relacionados, y en este punto veremos que el sistema empieza a recuperar la información de las diferentes listas de reputación que hemos indicado.

NAME	TYPE	STATE	INDICATORS	ADD/REM/AG	UPDATES	WITHDRAWNS
ellenvault-ips-input	MINER	STARTED	87379	ADDED: 9815 AGED OUT: 6420	RX: 0 PROCESSED: 0 TX: 87379	RX: 0 PROCESSED: 0 TX: 6420
bbcan177-domain-input	MINER	STARTED	16992	ADDED: 0 REMOVED: 0	RX: 0 PROCESSED: 0 TX: 17372	RX: 0 PROCESSED: 0 TX: 0
cc-domain-input	MINER	STARTED	810	ADDED: 141 AGED OUT: 164	RX: 0 PROCESSED: 0 TX: 811	RX: 0 PROCESSED: 0 TX: 164
cc-ips-input	MINER	STARTED	549	ADDED: 0 REMOVED: 0	RX: 0 PROCESSED: 0 TX: 0	RX: 0 PROCESSED: 0 TX: 0

Figura 12. Nodos de entrada datos de alta para el sistema desarrollado

3.1.2.3 Validación

La primera validación a realizar es la de verificar el estado del servicio. Dado que ya se ha realizado la configuración de los nodos, y ésta debe realizarse a través de la herramienta web, el servicio ya se asume que debería estar activo. No obstante, se indica cómo realizar la validación de igual forma para que quede debidamente documentado.

El estado que se espera del servicio es *active (running)*:

```

illogyc@uoc-server-minemeld:~$ service minemeld status
● minemeld.service - Process Monitoring and Control Daemon
   Loaded: loaded (/lib/systemd/system/minemeld.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2019-11-24 13:22:44 CET; 7h ago
     Process: 914 ExecStart=/opt/minemeld/engine/current/bin/supervisord -c /opt/minemeld/supervisor
     Process: 899 ExecStartPre=/bin/chown -R minemeld:minemeld /var/run/minemeld/ (code=exited, sta
     Process: 894 ExecStartPre=/bin/mkdir /var/run/minemeld (code=exited, status=0/SUCCESS)
   Main PID: 1150 (supervisord)
     Tasks: 27 (limit: 2318)

```

Figura 13. Validación estado de la herramienta Minemeld (2)

La siguiente validación es la de verificar, a través de la herramienta Web de Minemeld, que los diferentes nodos de nuestro sistema tienen un estado de *Started*.

NAME	TYPE	STATE	INDICATORS	ADD/REM/AO	UPDATES
alienvault-ip-input	MINER	STARTED	87379	ADDED: 9815 AGED OUT: 6420	RX: 0 PROCESSED: 0 TX: 87379
bbcan177-domain-input	MINER	STARTED	16992	ADDED: 0 REMOVED: 0	RX: 0 PROCESSED: 0 TX: 17372
cc-domain-input	MINER	STARTED	810	ADDED: 141 AGED OUT: 164	RX: 0 PROCESSED: 0 TX: 811

Figura 14. Validación estado de los nodos Minemeld

La última validación a realizar es que las URLs correspondientes a cada uno de los nodos de salida está respondiendo correctamente y que ofrecen la información que se espera en cada caso. Para ello, se introducirá en el navegador web las diferentes URLs indicadas y se verificará la información de salida en todos ellos.

Verificación nodo salida IPs

Url: <https://192.168.1.180/feeds/ips-output>

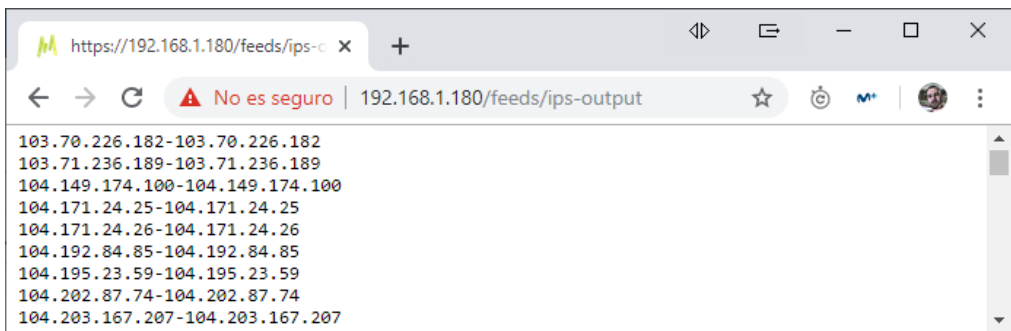


Figura 15. Validación acceso a feed Minemeld de tipo IP desde navegador

Verificación nodo salida Dominios

Url: <https://192.168.1.180/feeds/domains-output>

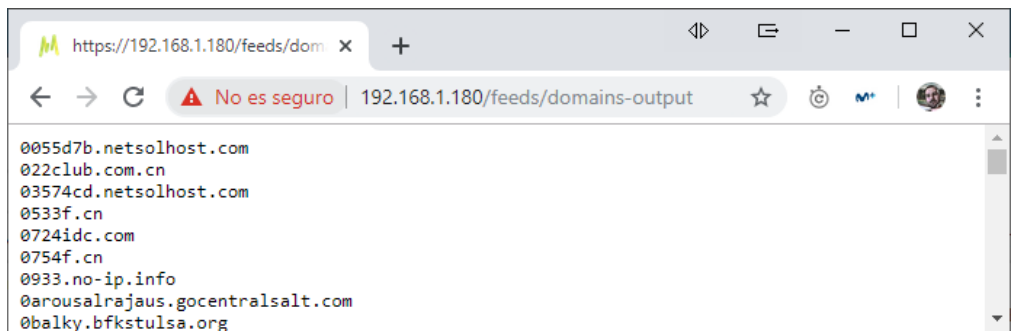


Figura 16. Validación acceso a feed Minemeld de tipo Dominio desde navegador

Verificación nodo salida URLs

Url: <https://192.168.1.180/feeds/urls-output>

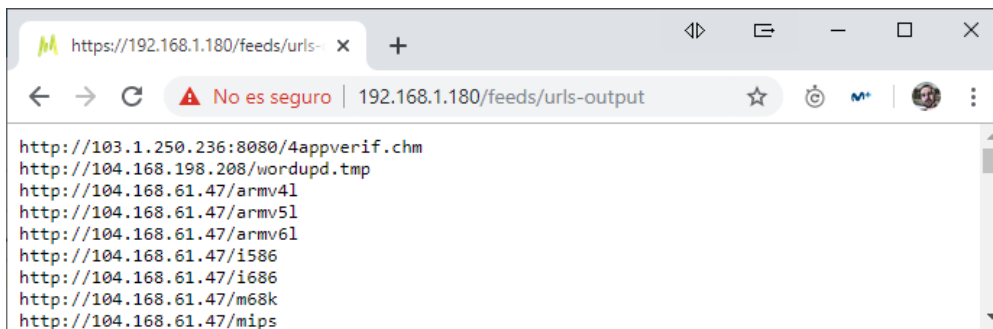


Figura 17. Validación acceso a feed Minemeld de tipo URL desde navegador

Verificación nodo salida Hashes

Url: <https://192.168.1.180/feeds/hash-output>

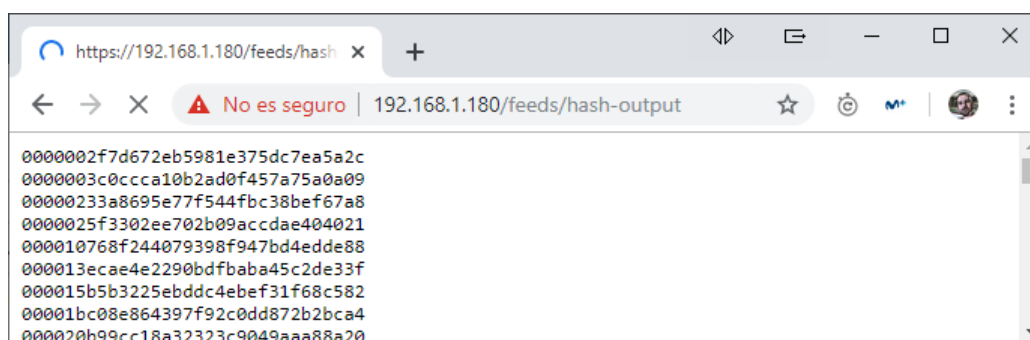


Figura 18. Validación acceso a feed Minemeld de tipo MD5 desde navegador

3.1.3 Python

3.1.3.1 Instalación

La instalación del entorno de ejecución Python, necesario para ejecutar los scripts que generarán los ficheros de diccionario para logstash a partir de los feeds generados en Minemeld, la realizaremos mediante el gestor de paquetes apt en la máquina uoc-server-ids. Los comandos a ejecutar son los siguientes:

```
$ sudo apt install python python-pip
```

A continuación, se copiarán los scripts facilitados en el sistema de ficheros de la máquina. Los scripts están incluidos en la memoria, dentro del Anexo III, y deberán ubicarse en el siguiente directorio:

```
/opt/zeek/minemeld
```

El directorio donde se crearán los ficheros de volcado con la información de los feed de Minemeld será el siguiente:

```
/opt/zeek/dictionary
```

Por último, será necesario instalar mediante pip la librería *requests* utilizada por los diferentes scripts. El comando a ejecutar es:

```
$ pip install requests
```

3.1.3.2 Configuración

A nivel de configuración, lo que se requiere es programar unas tareas Cron para ejecutar los scripts cada cierto tiempo, de forma que podamos asegurar de que los ficheros diccionario tienen la información actualizada de forma automática, sin requerir actualizaciones manuales.

A modo de ejemplo, se darán de alta las tareas para que sean ejecutadas cada cuatro horas, aunque para facilitar las pruebas, se realizarán ejecuciones de los scripts de forma manual para actualizar con mayor frecuencia los cambios realizados en los feeds Minemeld.

Para la configuración de las tareas, ejecutaremos el siguiente comando:

```
$ sudo crontab -e
```

Y daremos de alta las siguientes tareas:

```
* */4 * * * python /opt/zeek/minemeld/minemeld-parser-domain.py >>
/var/log/minemeld-parser-domain-Log
* */4 * * * python /opt/zeek/minemeld/minemeld-parser-ip.py >>
/var/log/minemeld-parser-ip-Log
* */4 * * * python /opt/zeek/minemeld/minemeld-parser-url.py >>
/var/log/minemeld-parser-url-Log
* */4 * * * python /opt/zeek/minemeld/minemeld-parser-hash.py >>
/var/log/minemeld-parser-hash-Log
```

3.1.3.3 Validación

La validación la realizaremos lanzando los scripts Python de forma manual, y revisando el contenido de los ficheros generados.

La ejecución de los scripts la realizaremos ejecutando los siguientes comandos:

Una vez realizados los pasos, podremos ejecutar ya los diferentes scripts Python mediante los siguientes comandos (Minemeld debe estar activo para poder realizar la prueba):

```
$ sudo python /opt/zeek/minemeld/minemeld-parser-ip.py
$ sudo python /opt/zeek/minemeld/minemeld-parser-domain.py
$ sudo python /opt/zeek/minemeld/minemeld-parser-hash.py
$ sudo python /opt/zeek/minemeld/minemeld-parser-url.py
```

Por consola, verificaremos que las trazas de salida indican que el proceso ha finalizado, y el número de registros que se han tratado, como en el siguiente ejemplo:

```

$ sudo python /opt/zeek/minemeld/minemeld-parser-url.py
2019-11-20 21:38:52 -----
2019-11-20 21:38:52 Starting minemeld parser -> url
2019-11-20 21:38:52
2019-11-20 21:38:52 Configuration:
2019-11-20 21:38:52     feed_url: https://192.168.1.180/feeds/urls-output
2019-11-20 21:38:52     temp_output: temp/urls-output-temp
2019-11-20 21:38:52     dest_output: /opt/zeek/dictionary/urls.yml
2019-11-20 21:38:52
2019-11-20 21:38:52 Loading...
2019-11-20 21:38:52 Finished! Items Loaded: 2280

```

Y también verificaremos que el fichero de salida contiene información de los registros obtenidos de Minemeld:

```

$ more /opt/zeek/dictionary/urls.yml
"103.1.250.236:8080/4appverif.chm": malicious
"103.207.38.15:1010/get": malicious
"104.148.19.229/ys53a": malicious
"104.168.198.208/wordupd.tmp": malicious
"104.168.61.47/armv4l": malicious
"104.168.61.47/armv5l": malicious
"104.168.61.47/armv6l": malicious
"104.168.61.47/i586": malicious

```

3.1.4 Stack Elastic

3.1.4.1 Instalación

La instalación del stack Elastic se realizará en la máquina uoc-server-elk. De los tres componentes que componen el stack, tan sólo se instalarán Kibana y Elasticsearch en esta máquina, puesto que Logstash se instalará en la máquina donde está desplegado Zeek IDS.

Los comandos a ejecutar para la instalación son:

```

$ sudo apt install openjdk-8-jre apt-transport-https nginx
$ sudo sh -c "echo 'deb https://artifacts.elastic.co/packages/7.x/apt stable
main' > /etc/apt/sources.list.d/elastic.list"
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-
key add -
$ sudo apt-get update
$ sudo apt-get install elasticsearch kibana

```

3.1.4.2 Configuración

La primera configuración a aplicar será la de indicar el host sobre el cual podrá recibir peticiones elasticsearch. Por defecto es localhost, pero en nuestro caso, el acceso se realizará desde la máquina uoc-server-ids, dentro de la LAN.

Para poder configurar correctamente elasticsearch, se editará el fichero

```
/etc/kibana/kibana.yml
```

y se informará la propiedad `elasticsearch.hosts` con el siguiente parámetro

```
elasticsearch.hosts: ["http://192.168.1.170:9200"]
```

A continuación, también se requiere editar la configuración de `elasticsearch` para informar de forma correcta el host por el cual se podrán recibir las peticiones. Para ello, se editará el fichero

```
/etc/elasticsearch/elasticsearch.yml
```

y se informarán las siguientes propiedades:

```
network.host: 192.168.1.170
discovery.seed_hosts: ["127.0.0.1", "192.168.1.170"]
```

Una vez aplicados los cambios, procederemos a reiniciar los servicios de `Elasticsearch` y `Kibana` para que apliquen la nueva configuración.

```
$ sudo service elasticsearch restart
$ sudo service kibana restart
```

Por último, quedará por configurar `nginx` para apuntar a la aplicación `Kibana`. Esta configuración requiere dar de alta un nuevo site en `Nginx`. Para ello, se creará un fichero con nombre `kibana` en el siguiente directorio,

```
/etc/nginx/sites-available
```

que contendrá la siguiente configuración:

```
server {
    listen 80;

    server_name uoc-zeek-elk.com;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

A continuación, se eliminará el site por defecto de `nginx`, y crearemos un enlace simbólico al fichero creado anteriormente dentro del directorio de sites activos,

```
$ sudo rm /etc/nginx/sites-enabled/default
$ sudo ln -s /etc/nginx/sites-available/kibana /etc/nginx/sites-enabled/kibana
```

Y para aplicar la nueva configuración, reiniciaremos el servicio nginx,

```
$ sudo service nginx restart
```

3.1.4.3 Validación

La validación de la correcta instalación de elasticsearch y kibana se realizará en varios pasos.

En primer lugar, validamos que ambos servicios están activos mediante los comandos,

```
$ sudo service elasticsearch status  
$ sudo service kibana status
```

Que deben mostrar que los servicios están *active (running)*.

En segundo lugar, verificaremos mediante el acceso a través de un navegador web de que Kibana está funcionando correctamente y que hemos dado de alta la configuración del site en nginx de forma adecuada.

La url de acceso sería: <http://192.168.1.170>

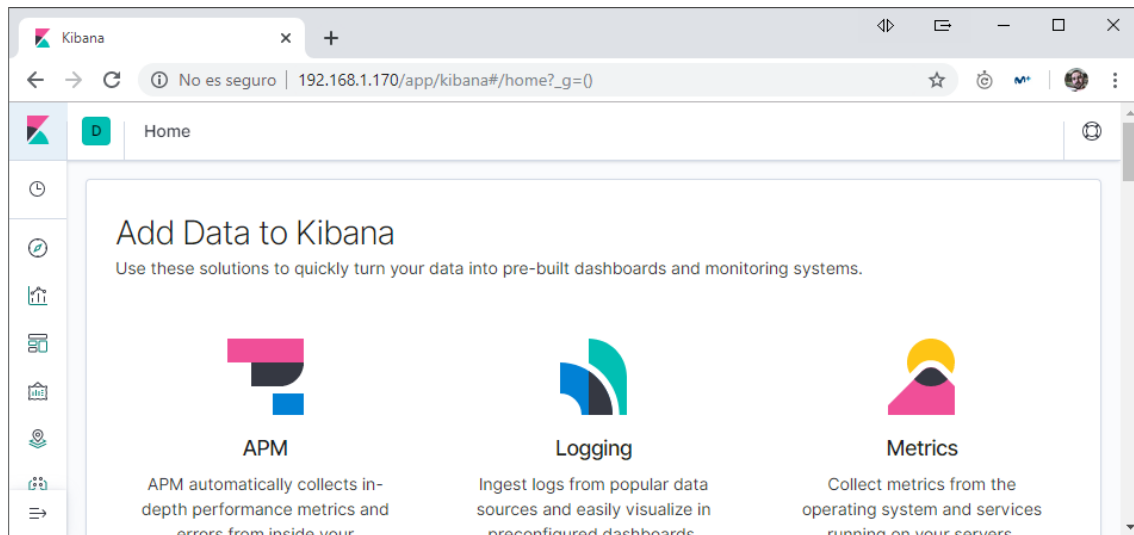


Figura 19. Página de bienvenida de Kibana

Por último, validaremos también mediante la invocación a una URL de elasticsearch a través del navegador web, de que el motor está funcionando correctamente y que permite recibir respuestas desde máquinas externas.

La url de prueba sería: <http://192.168.1.170:9200/>



Figura 20. Validación acceso a servicio Elasticsearch

3.1.4.4 Template

Una de las funcionalidades que se requieren para la elaboración de los dashboards es la de geoposicionamiento, cuyo objetivo es poder pintar un mapa del mundo con los puntos de las conexiones remotas hacia la máquina host. A su vez, también se requerirá construir objetos de visualización que realizarán sumatorios sobre ciertos campos de los ficheros de trazas.

Debido a que no se utilizarán nombres de índices en elasticsearch estándar de logstash (nomenclatura logstash-*), sino que utilizaremos la nomenclatura zeek-*, hay ciertas configuraciones que aplican a geoposicionamiento y al tipo de ciertos campos que por defecto no estarían bien definidos.

Para solucionarlo, es necesario que, antes de realizar la ingesta de información en elastic search, se de de alta un nuevo template para índices zeek-*, donde se indicará el correcto mapeo para los campos de geoposicionamiento que se van a utilizar y los tipos de datos para ciertos campos que se utilizarán para sumatorios.

El alta del nuevo template de mapeo se realizará accediendo al menú *Dev Tools* de Kibana y ejecutando el script definido en el Anexo IV del trabajo, que está basado en el mismo template que el utilizado para índices con nomenclatura logstash-*.

Si el template se ha creado de forma satisfactoria, nos aparecerá un mensaje de acknowledge de este tipo:

```
{
  "acknowledged" : true
}
```

Una vez completada el alta del nuevo template, se puede proceder con la instalación, configuración y arranque de logstash.

3.1.5 Logstash

3.1.5.1 Instalación

La instalación de Logstash se realizará en la máquina *uoc-server-ids*, ya que será desde esta máquina donde se generarán las trazas que posteriormente ingestaremos en el elasticsearch instalado en la máquina elk.

Los comandos a ejecutar para la instalación son:

```
$ sudo apt install openjdk-8-jre apt-transport-https
$ sudo sh -c "echo 'deb https://artifacts.elastic.co/packages/7.x/apt stable main' > /etc/apt/sources.list.d/elastic.list"
$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add -
$ sudo apt-get update
$ sudo apt-get install logstash
```

3.1.5.2 Configuración

La ubicación de los ficheros de configuración está en el directorio,

```
/etc/logstash
```

En primer lugar, crearemos un nuevo fichero de configuración, denominado,

```
/etc/logstash/conf.d/zeek.conf
```

La estructura del fichero de configuración estará compuesta por tres bloques:

- *input*: donde se configurarán las entradas de tipo *file* y se indicará la ubicación de los ficheros de trazas generados por zeek para su lectura. A la vez, se especificará un tipo mediante el atributo *type* para tener categorizadas las entradas para ser utilizadas en las posteriores configuraciones.

```
input {
  [...]
}
```

- *filter*: en este punto se aplicarán los filtros a cada uno de los nodos de entrada definidos en el bloque *input*, que se resumen en:
 - aplicar unos patrones utilizando el plugin *grok* para poder parsear los logs de entrada de forma correcta.
 - aplicar el plugin de traducción (*translate*) para la detección de actividades sospechosas a partir de los ficheros de diccionario generados por los scripts Python.
 - activar la funcionalidad de geolocalización para poder utilizar mapas de calor en Kibana.

```
filter {
  [...]
}
```

- *output*. último bloque donde se especifica la ingesta que se va a realizar para cada uno de los datos de entrada hacia Elasticsearch, y el nombre de índice que se utilizará.

```
output {
  [...]
}
```

Antes de proceder con la definición de las configuraciones en el fichero *zeek.conf* es necesario realizar un paso previo, para definir los patrones de los ficheros que se trataran de Zeek, con el objetivo de poder determinar las etiquetas y el tipo para cada dato registrado en los ficheros de log.

Patrones ficheros Zeek

Para que Logstash pueda parsear de forma correcta los ficheros de trazas de Zeek, es necesario definir las estructuras de información de los registros de cada uno de los ficheros de entrada.

En este caso, se dará de alta el fichero *zeek-patterns* en la siguiente ruta,

```
/etc/logstash/templates/zeek-patterns
```

y dentro del fichero, agregaremos las expresiones regulares de parseado para cada uno de los tipos de fichero que se darán de alta, que en el caso de este trabajo serán los siguientes: *connLog*, *dnsLog*, *httpLog*, *filesLog*, *sslLog*.

El contenido del fichero *zeek-patters* está definido en el Anexo IV del trabajo.

Una vez guardado el fichero, procederemos a la configuración de cada uno de los bloques del fichero *zeek.conf*.

Bloque input

La configuración para indicar a logstash el origen de los ficheros de zeek es la siguiente:

```
input {
  file {
    path => "/opt/zeek/Logs/current/conn.Log"
    type => "conn"
  }
  file {
    path => "/opt/zeek/Logs/current/dns.Log"
    type => "dns"
  }
  file {
    path => "/opt/zeek/Logs/current/http.Log"
    type => "http"
  }
}
```

```

}
file {
  path => "/opt/zeek/Logs/current/files.Log"
  type => "files"
}
}

```

Bloque filter

En este punto, agregaremos la configuración específica para cada fichero zeek que se ha indicado en el bloque de entrada, toda ella a incluirse dentro del nodo *filter*.

Fichero conn.log

Para este fichero, se configurará en el plugin de *translate* el fichero de IPs maliciosas generado por el script Python, y el campo del fichero de trazas de Zeek que se utilizará para el matching será *resp_h* que contendrá la dirección IPv4 remota de las conexiones realizadas con la máquina host.

También se indicará el uso del plugin *grok* para el parseado de los datos de entrada según el fichero de patrones creado anteriormente.

```

if [path] == "/opt/zeek/Logs/current/conn.Log" {
  grok {
    patterns_dir => "/etc/Logstash/patterns"
    match => {
      message => "%{connLog}"
    }
  }
  translate {
    field => "resp_h"
    add_field => { "malicious" => "yes" }
    dictionary_path => "/opt/zeek/dictionary/ips.yml"
  }
}

```

Fichero http.log

Para este fichero, se define de nuevo el uso del plugin *grok* para el parseado del fichero de entrada, y en el plugin de traducción se especificará el fichero de URLs maliciosas generado a partir de Minemeld.

Como el fichero de URLs maliciosas utiliza para cada registro un patrón de tipo hostname(IP o FQDN)+URI, y en el fichero de trazas http.log de Minemeld no hay ningún campo con la información estructurada de esa forma, lo que se requiere es incluir un campo adicional a la información que se ingesta en Elasticsearch con la información en el formato que se requiere. Ese nuevo campo se define mediante el atributo *mutate*.

```

if [path] == "/opt/zeek/Logs/current/http.Log" {
  grok {
    patterns_dir => "/etc/Logstash/patterns"

```

```

    match => {
        message => "%{httpLog}"
    }
}
mutate {
    add_field => { "url" => "%{server_name}%{uri}" }
}
translate {
    field => "url"
    add_field => { "malicious" => "yes" }
    dictionary_path => "/opt/zeek/dictionary/urls.yml"
}
}

```

Fichero dns.log

Para este fichero, además del plugin *grok*, se especificará en el filtro de traducción el uso del fichero diccionario *domains.yml* con la lista de dominios maliciosos, y el dato por el cual se hará matching será el campo *query*.

```

if [path] == "/opt/zeek/logs/current/dns.log" {
    grok {
        patterns_dir => "/etc/logstash/patterns"
        match => {
            message => "%{dnsLog}"
        }
    }
    translate {
        field => "query"
        add_field => { "malicious" => "yes" }
        dictionary_path => "/opt/zeek/dictionary/domains.yml"
    }
}
}

```

Fichero files.log

Para este fichero, se define el plugin *grok*, y a la vez se especificará en el filtro de traducción el uso del fichero diccionario *files.yml* con la lista de hashes md5 de ficheros maliciosos. El dato por el cual se hará matching será el campo *md5*.

```

if [path] == "/opt/zeek/logs/current/files.log" {
    grok {
        patterns_dir => "/etc/logstash/patterns"
        match => {
            message => "%{filesLog}"
        }
    }
    translate {
        field => "md5"
        add_field => { "malicious" => "yes" }
        dictionary_path => "/opt/zeek/dictionary/hashes.yml"
    }
}
}

```

Geoposicionamiento

Adicionalmente a los plugins de parseado y traducción utilizados para cada uno de los ficheros de trazas, se define una configuración para poder utilizar el plugin de geoposicionamiento ofrecido por Logstash para poder obtener información adicional respecto a las conexiones identificadas por Zeek para su posterior explotación en Kibana.

Como en la definición de los patrones de parseado, se ha asignado el nombre *resp_h* a la información relativa a las direcciones IPv4 de las conexiones remotas, utilizaremos estos campos como datos origen del plugin de posicionamiento, y los datos adicionales resueltos se almacenarán en el atributo *geoip*.

```
geoip {
  source => "resp_h"
}
```

Valores por defecto

En el caso de datos que se van a utilizar para realizar sumatorios, zeek asigna a esos datos un valor por defecto de "-" en caso de que no haya un valor numérico que se pueda asignar.

Este valor por defecto, provoca una serie de errores en la ingesta de información. Para solventar la problemática, se agregan una serie de filtros de tipo *mutate* para que en caso de llegar de zeek valores "-", éstos sean sustituidos por 0.

La configuración a aplicar es la siguiente:

```
if [orig_bytes] == "-" {
  mutate {
    replace => ["orig_bytes", "0"]
  }
}

if [orig_ip_bytes] == "-" {
  mutate {
    replace => ["orig_ip_bytes", "0"]
  }
}

if [orig_pkts] == "-" {
  mutate {
    replace => ["orig_pkts", "0"]
  }
}

if [resp_bytes] == "-" {
  mutate {
    replace => ["resp_bytes", "0"]
  }
}
```

```

if [resp_ip_bytes] == "-" {
  mutate {
    replace => ["resp_ip_bytes", "0"]
  }
}

if [resp_pkts] == "-" {
  mutate {
    replace => ["resp_pkts", "0"]
  }
}
}

```

Bloque output

En el bloque output, generaremos una configuración de salida utilizando el plugin de elasticsearch para cada tipo de log y, además, generaremos una salida adicional con la información de todos los logs.

Para cada configuración de salida, se especifica el host de elasticsearch, y además se indica el nombre del índice que se utilizará en cada caso.

```

output {
  elasticsearch {
    hosts => "192.168.1.170"
    index => "zeek-all-logs"
  }

  if [type] == "conn" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-conn-logs"
    }
  }
  if [type] == "http" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-http-logs"
    }
  }
  if [type] == "dns" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-dns-logs"
    }
  }
  if [type] == "files" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-files-logs"
    }
  }
}
}

```

Una vez realizada la configuración completa del fichero de configuración de Logstash, realizaremos un reinicio del servicio para aplicar los cambios mediante el siguiente comando:

```
$ sudo service logstash restart
```

3.1.5.3 Validación

La validación de la correcta instalación y configuración de Logstash se divide en tres partes.

En la primera, se validará que la configuración aplicada en el fichero *zeek.conf* es correcta, y que Logstash inicia correctamente con los cambios aplicados. Para ello, una vez reiniciado el servicio, consultaremos el estado del mismo para asegurarnos que muestra *active (running)* como se espera.

```
root@uoc-server-ids:/# service logstash status
● logstash.service - logstash
   Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: enabled)
   Active: active (running) since Sun 2019-11-24 12:01:21 CET; 19s ago
     Main PID: 8376 (java)
       Tasks: 13 (limit: 4660)
    CGroup: /system.slice/logstash.service
            └─8376 /usr/bin/java -Xms1g -Xmx1g -XX:+UseConcMarkSweepGC -XX:CMSInitiatingOccupanc
```

Figura 21. Validación estado servicio Logstash

En la segunda, validaremos que la información recuperada a partir del fichero de trazas se ingesta de forma correcta en Elasticsearch, y que a través de Kibana, visualizamos los índices configurados en el bloque de salida (output) del fichero de configuración *zeek.conf*.

Para validar este punto, se requiere arrancar la máquina *uoc-server-elk*, e iniciar los servicios *elasticsearch* y *kibana*. Además, para que los índices se den de alta correctamente, será necesario generar tráfico de red para cada uno de los ficheros de trazas a ingestar.

Para generar el tráfico de red, que provoquen la generación de todos los índices, lanzaremos los siguientes comandos desde la máquina *uoc-server-host*:

```
$ curl http://google.es

$ wget http://file-examples.com/wp-content/uploads/2017/10/file_example_JPG_100kB.jpg
```

Una vez ejecutados, accederemos a la herramienta web de Kibana mediante navegador Web, y accederemos a la sección de configuración (*Management*). En este punto, accederemos a la funcionalidad de gestión de índices (*Index Management*) de la sección de Elasticsearch, y comprobaremos que se han detectado los diferentes índices configurados en Logstash.

The screenshot shows the 'Index Management' page in Elasticsearch. It features a search bar, a 'Reload indices' button, and a table of indices. The table has columns for Name, Health, Status, Primaries, Replicas, Docs count, and Storage size. There are six indices listed, all with a 'yellow' health status and 'open' status.

<input type="checkbox"/>	Name	Health	Status	Primaries	Replicas	Docs count	Storage size
<input type="checkbox"/>	zeek-all-logs	● yellow	open	1	1	27047	18.4mb
<input type="checkbox"/>	zeek-http-logs	● yellow	open	1	1	1127	1mb
<input type="checkbox"/>	zeek-conn-logs	● yellow	open	1	1	13233	18.3mb
<input type="checkbox"/>	zeek-dns-logs	● yellow	open	1	1	11848	8mb
<input type="checkbox"/>	zeek-files-logs	● yellow	open	1	1	777	926.5kb

Figura 22. Listado de índices generados en Elasticsearch

3.2 Monitorización

Una vez completada la instalación, configuración y validación de todos los componentes software que conforman el sistema, el siguiente punto es construir el sistema de monitorización activo en Kibana que se utilizará para la detección de actividades sospechosas.

Para ello, se darán de alta todos los índices Elasticsearch en Kibana. Tras esta alta, ya se estará en disposición de consultar información de todas las trazas generadas por nuestro sistema Zeek.

A continuación, se construirán los diferentes Dashboards de monitorización para cada índice, donde podremos tener una visión completa de las conexiones realizadas en tiempo real, detectando actividades maliciosas de forma ágil.

3.2.1 Creación patrones de índices

La creación de patrones de índices Kibana es requerido para poder utilizar las herramientas de búsqueda y tener toda la información correctamente mapeada, incluyendo los campos de los ficheros de trazas, tipo de datos, formatos, etc.

Para ello, se accederá a la sección de Configuración (*Management*) de la herramienta web Kibana, y se accederá al apartado de *Index Patterns*.

A continuación, crearemos un patrón de índice por cada índice existente en elasticsearch, y que previamente hemos visualizado en la gestión de índices del apartado de Elasticsearch.

La creación de cada patrón de índice se realiza en dos pasos. En el primero, se requiere indicar el nombre del índice de elasticsearch.

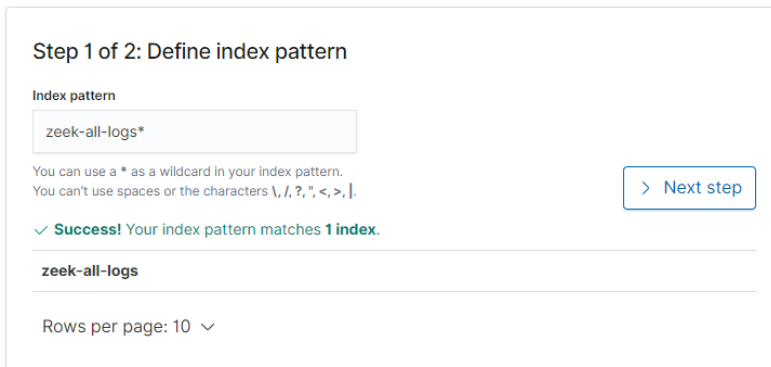


Figura 23. Pantalla de creación de patrón de índice en Kibana – Paso 1

En el segundo paso, se solicitará indicar el nombre del campo que se utilizará para el filtrado basado en tiempo, que se informará con el campo @timestamp.

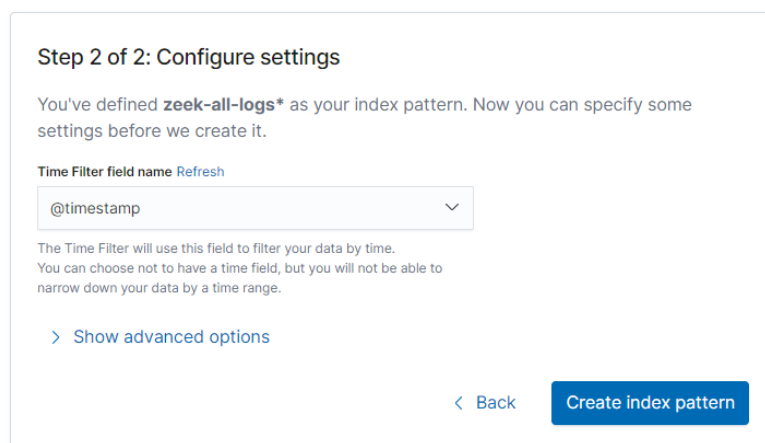


Figura 24. Pantalla de creación de patrón de índice en Kibana – Paso 2

Una vez creado el patrón de índice, se realizará la misma acción para el resto de índices hasta tenerlos todos dados configurados.

3.2.2 Consulta de información

Habiendo creado los patrones de índices, ya podremos acceder a la sección de *Discover* de Kibana, y ejecutar algunas consultas sobre los diferentes índices.

Para validar esta parte, seleccionaremos el índice *zeek-dns-logs*, y filtraremos por consultas realizadas del dominio *google.es*.

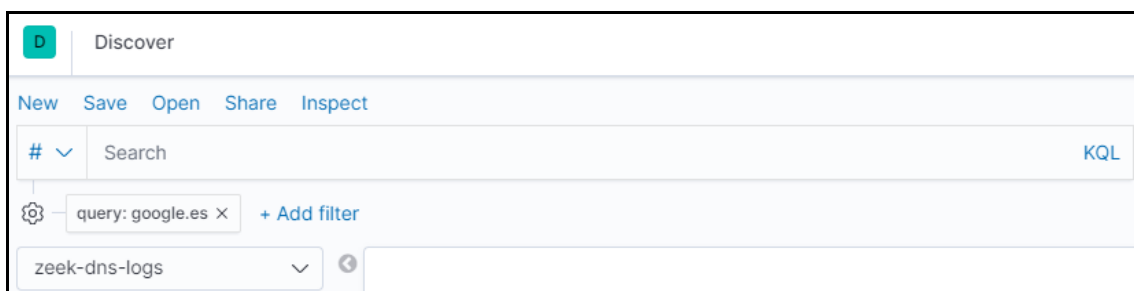


Figura 25. Ejemplo de consulta de información a través de Kibana

Una de las trazas obtenidas para dicha consulta sería la siguiente,

```
Time ▾      _source
> Nov 24, 2019 @ 13:07:00.896  query: google.es  geoup_resp.country_name: United States
                                geoup_resp.ip: 8.8.8.8  geoup_resp.location.lon: -97.822
                                geoup_resp.location.lat: 37.751  geoup_resp.longitude: -97.822
                                geoup_resp.country_code3: US  geoup_resp.timezone: America/Chicago
                                geoup_resp.latitude: 37.751  geoup_resp.continent_code: NA
```

Figura 26. Ejemplo de traza recuperada en Kibana después de consulta

donde se muestra información de la consulta realizada (campo *query*), así como información adicional como los datos de geoposicionamiento (campos *geoup_resp*, *geoup_orig*), si la consulta es maliciosa (campo *malicious*), servidor dns utilizado (campo *resp_h*), etc.

3.2.3 Construcción Dashboards

Para la construcción de los dashboards, se deberán realizar dos tipos de procesos en función de los datos a visualizar:

- En el caso de gráficas, métricas, o tablas de contaje, se requerirá dar de alta objetos de visualización en la sección *Visualize* de Kibana.
- En el caso de listados de últimos registros, será necesario realizar unas búsquedas de información, estableciendo las columnas a visualizar, y estas búsquedas serán guardadas en Kibana para poder ser agregadas en los diferentes dashboards de monitorización.

3.2.3.1 Visión Global

El dashboard Visión Global contendrá la siguiente información:

- Mapa de calor del mundo con conexiones destino.
- Gráfico circular con Países de las conexiones destino.
- Total bytes de datos de entrada.
- Total bytes de datos de salida.
- Gráfico de barras con protocolos detectados.
- Total conexiones maliciosas detectadas.
- Lista de las últimas conexiones maliciosas detectadas.

A continuación, se describe las características de cada objeto de visualización:

Mapa de calor del mundo con conexiones destino

- Nombre: [Global] Cordinates map
- Tipo visualización: Coordinate Map
- Datos origen: zeek-conn-logs
- Campo de coordenadas: geoup.location

Una vez completada la configuración, aplicamos los campos, y verificamos que el mapa muestra ya los puntos de las conexiones de las máquinas remotas.

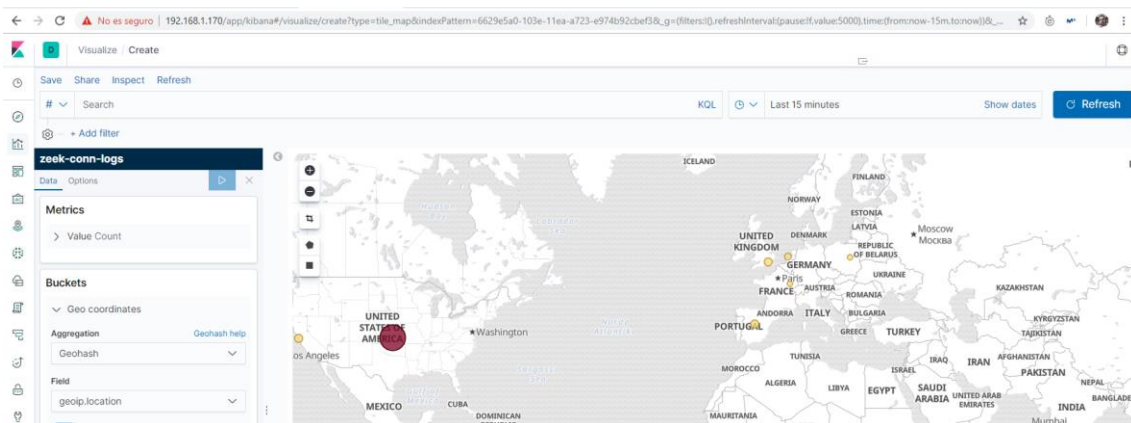


Figura 27. Mapa de calor del mundo con conexiones destino

Gráfico circular con Países de las conexiones destino

- Nombre: [Global] Country pie chart
- Tipo visualización: Pie chart
- Datos origen: zeek-conn-logs
- Metrics - Aggregation: Count
- Buckets:
 - Split slices
 - Aggregation: Terms
 - Field: geoup.country_name.keyword

Tras aplicar los cambios, obtenemos el objeto en cuestión:

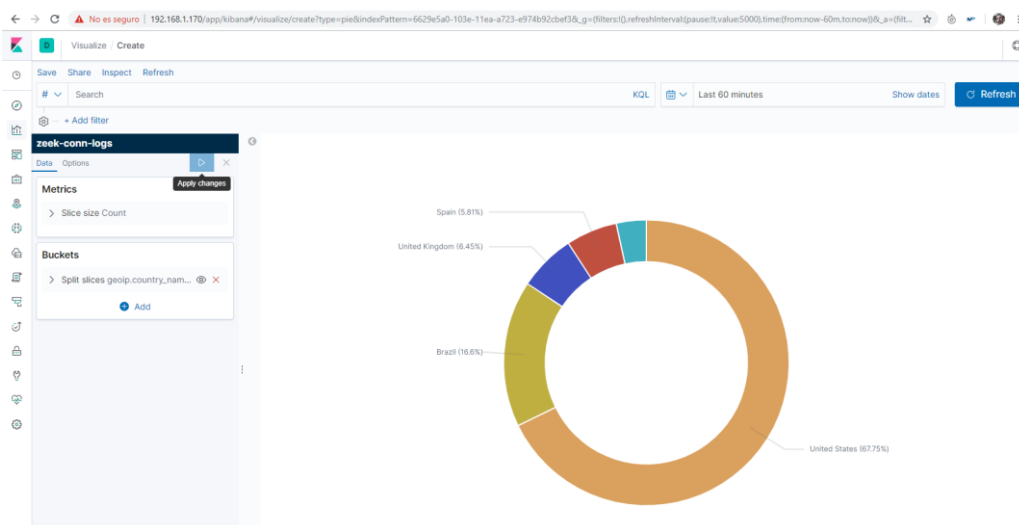


Figura 28. Gráfico circular con Países de las conexiones destino

Total bytes de datos de entrada

- Nombre: [Global] Input bytes
- Tipo visualización: Metric

- Datos origen: zeek-conn-logs
- Metrics – Aggregation: Sum
- Field: resp_bytes
- Custom label: bytes

Tras aplicar los cambios, obtenemos el objeto en cuestión:

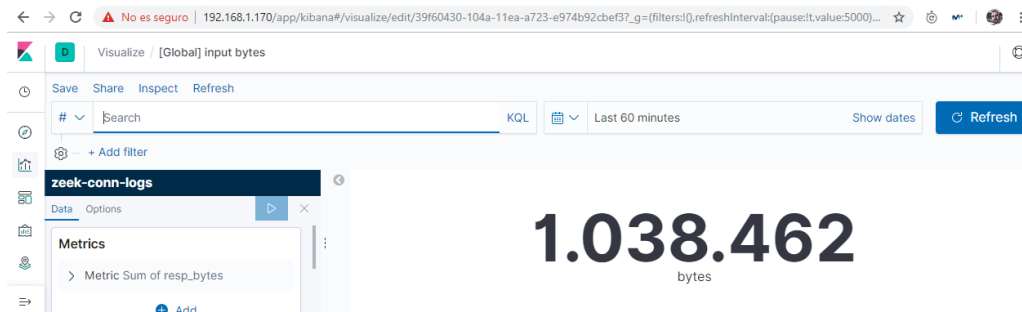


Figura 29. Total bytes de datos de entrada

Total bytes de datos de salida

- Nombre: [Global] Output bytes
- Tipo visualización: Metric
- Datos origen: zeek-conn-logs
- Metrics – Aggregation: Sum
- Field: orig_bytes
- Custom label: bytes

Tras aplicar los cambios, obtenemos el objeto en cuestión:

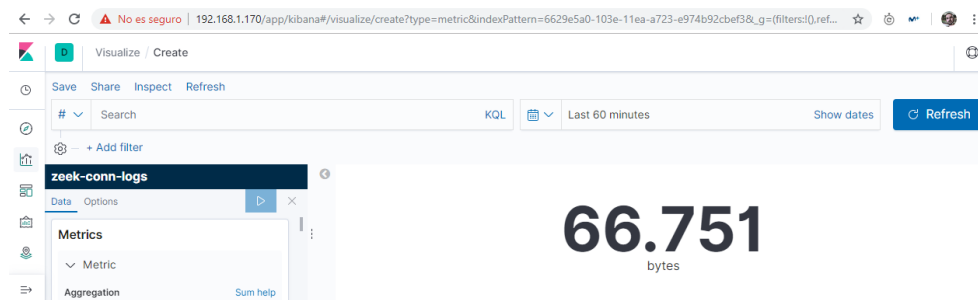


Figura 30. Total bytes de datos de salida

Gráfico de barras con servicios detectados

- Nombre: [Global] Services
- Tipo visualización: Vertical Bar
- Datos origen: zeek-conn-logs
- Metrics – Aggregation: Count
- Buckets:
 - X-axis
 - Aggregation: Terms
 - Field: service.keyword
 - Order by: Metric : Count

Tras aplicar los cambios, obtenemos el objeto en cuestión:

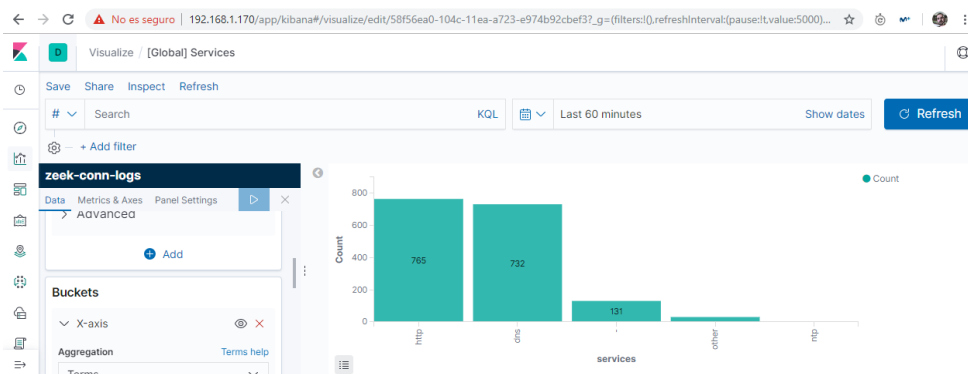


Figura 31. Gráfico de barras con servicios detectados

Total conexiones maliciosas detectadas

- Nombre: [Global] Total malicious
- Tipo visualización: Metric
- Datos origen: zeek-conn-logs
- Metrics – Aggregation: Count
- Buckets:
 - Aggregation: Filters
 - Filter 1: “malicious : yes”

Tras aplicar los cambios, obtenemos el objeto en cuestión:

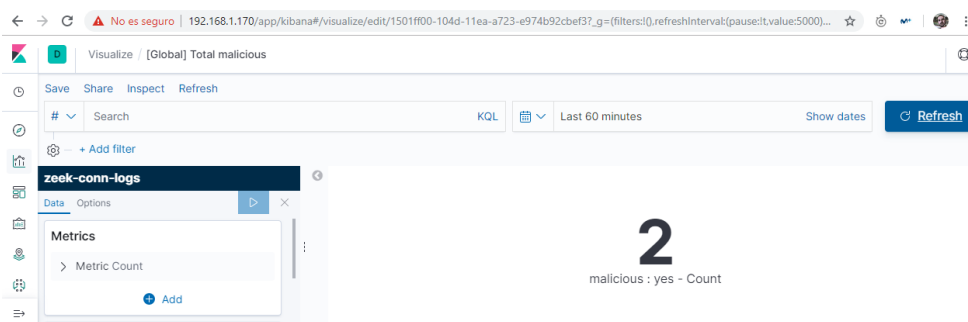


Figura 32. Total conexiones maliciosas detectadas

Lista de las conexiones maliciosas detectadas

Para este indicador, se requiere únicamente la construcción de una búsqueda a partir de los logs del índice zeek-conn-logs. Para ello, se accederá a la sección de *Discover*, y se realizará la siguiente configuración:

- Filtrar por el campo malicious para un valor “yes”.
- Se agregarán las siguientes columnas al listado: resp_h (dirección IP del host remoto), geoip.country_name, geoip.city_name, proto (protocolo) y service (servicio).

Una vez aplicada la configuración, guardaremos la búsqueda como “[Search]”

Malicious connections” y posteriormente la agregaremos al dashboard para su visualización.

El aspecto de la tabla una vez aplicados los cambios es la siguiente:

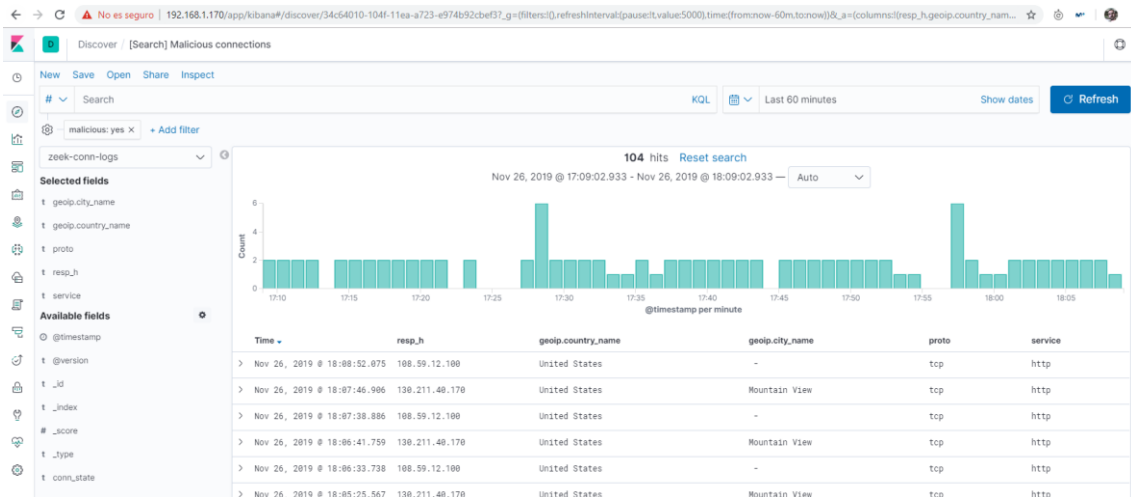


Figura 33. Lista de las conexiones maliciosas detectadas

Construcción dashboard

Una vez que ya tenemos todos los objetos de visualización, se dará de alta un nuevo dashboard, con nombre “Visión Global”, e iremos añadiendo todos los componentes visuales al Canvas disponible.

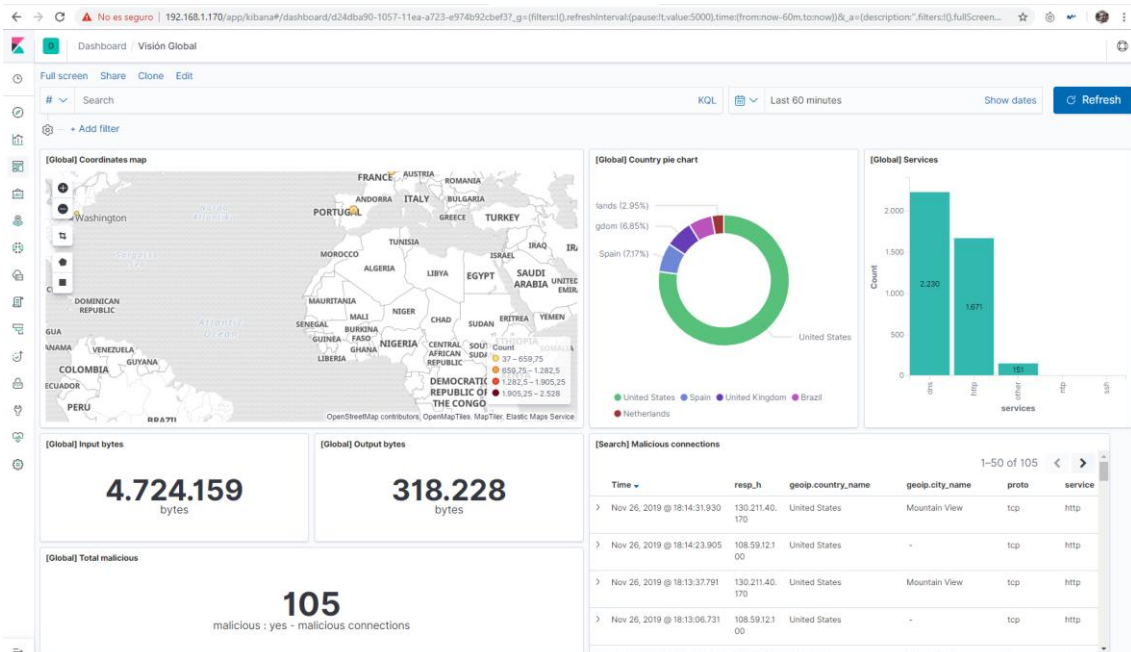


Figura 34. Dashboard de monitorización “Visión Global”

3.2.3.2 URLs

El dashboard de URLs contendrá la siguiente información:

- Total de peticiones http realizadas.
- Gráfico de barras con códigos de respuesta y totales.
- Top 10 de URLs visitadas.
- Total peticiones http maliciosas detectadas.
- Lista de peticiones http maliciosas detectadas.

A continuación, se describe las características de cada objeto de visualización:

Total de peticiones http realizadas

- Nombre: [URL] Total HTTP requests
- Tipo visualización: Metric
- Datos origen: zeek-http-logs
- Metrics – Aggregation: Count

Tras aplicar los cambios, obtenemos el objeto en cuestión:

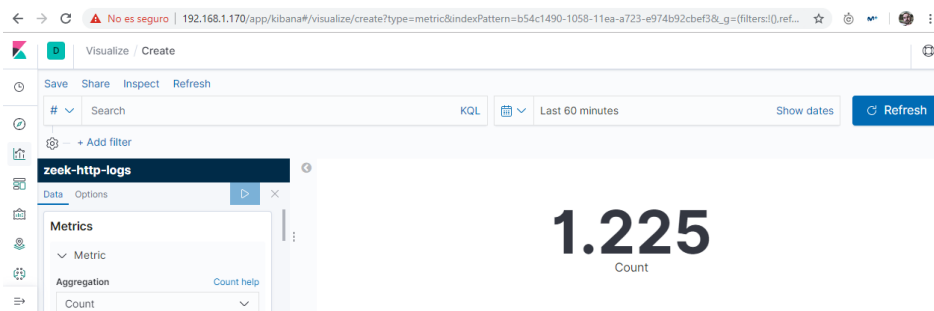


Figura 35. Total de peticiones http realizadas

Gráfico de barras con códigos de respuesta y totales

- Nombre: [URL] HTTP response codes
- Tipo visualización: Vertical Bar
- Datos origen: zeek-http-logs
- Metrics – Aggregation: Count
- Buckets:
 - X-axis
 - Aggregation: Terms
 - Field: status_msg.keyword
 - Order by: Metric : Count

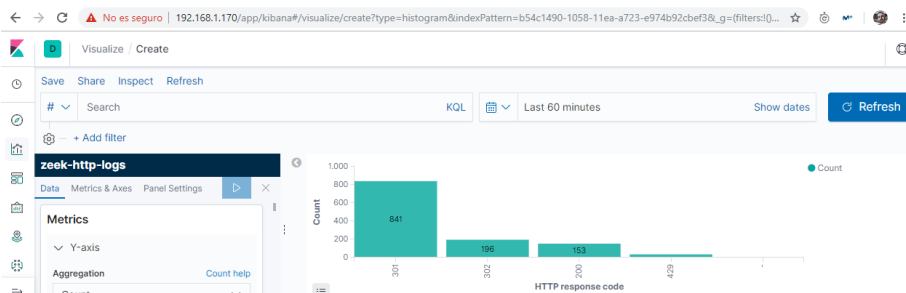


Figura 36. Gráfico de barras con códigos de respuesta y totales

Top 10 de URLs visitadas

- Nombre: [URL] Top 10 HTTP requests
- Tipo visualización: Data Table
- Datos origen: zeek-http-logs
- Metrics – Aggregation: Count
- Buckets:
 - Split rows
 - Aggregation: Terms
 - Field: url.keyword
 - Order by: Metric : Count
 - Order: Descending
 - Size: 10

Tras aplicar los cambios, obtenemos el objeto en cuestión:

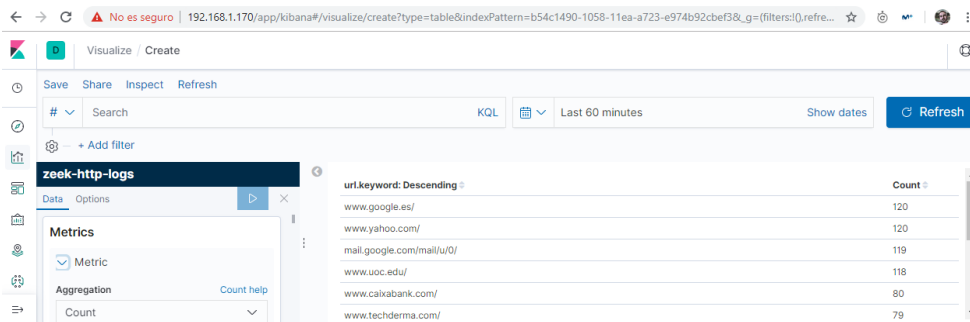


Figura 37. Top 10 de URLs visitadas

Total peticiones http maliciosas detectadas

- Nombre: [URL] Total HTTP malicious requests
- Tipo visualización: Metric
- Datos origen: zeek-http-logs
- Metrics – Aggregation: Count
- Buckets:
 - Split group
 - Aggregation: Filters
 - Filter 1: “malicious : yes”

Tras aplicar los cambios, obtenemos el objeto en cuestión:

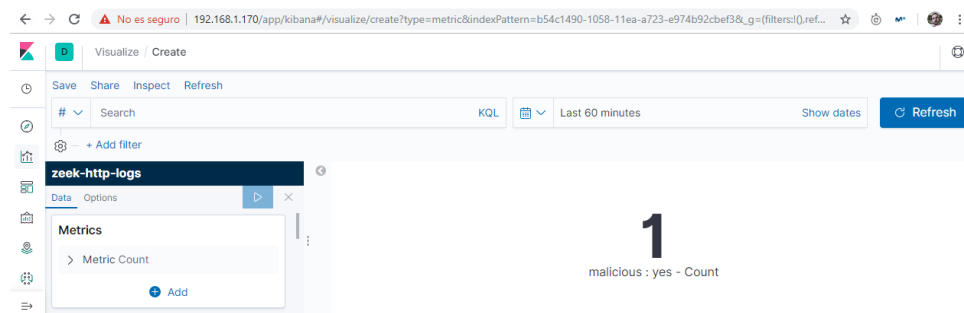


Figura 38. Total peticiones http maliciosas detectadas

Lista de peticiones http maliciosas detectadas

Para este indicador, se requiere únicamente la construcción de una búsqueda a partir de los logs del índice zeek-http-logs. Para ello, se accederá a la sección de *Discover*, y se realizará la siguiente configuración:

- Filtrar por el campo *malicious* para un valor “yes”.
- Se agregarán las siguientes columnas al listado: *resp_h* (dirección IP del host remoto), *URI*, *resp_p* (puerto del host remoto), *method* (método HTTP utilizado) y *geop.country_name*.

Una vez aplicada la configuración, guardaremos la búsqueda como [Search] *Malicious HTTP requests*” y posteriormente la agregaremos al dashboard para su visualización.

Construcción dashboard

Una vez que ya tenemos todos los objetos de visualización, se dará de alta un nuevo dashboard, con nombre “Análisis URL”, e iremos añadiendo todos los componentes visuales al Canvas disponible.

Tras agregar los elementos visuales, el resultado del dashboard obtenido es el siguiente:

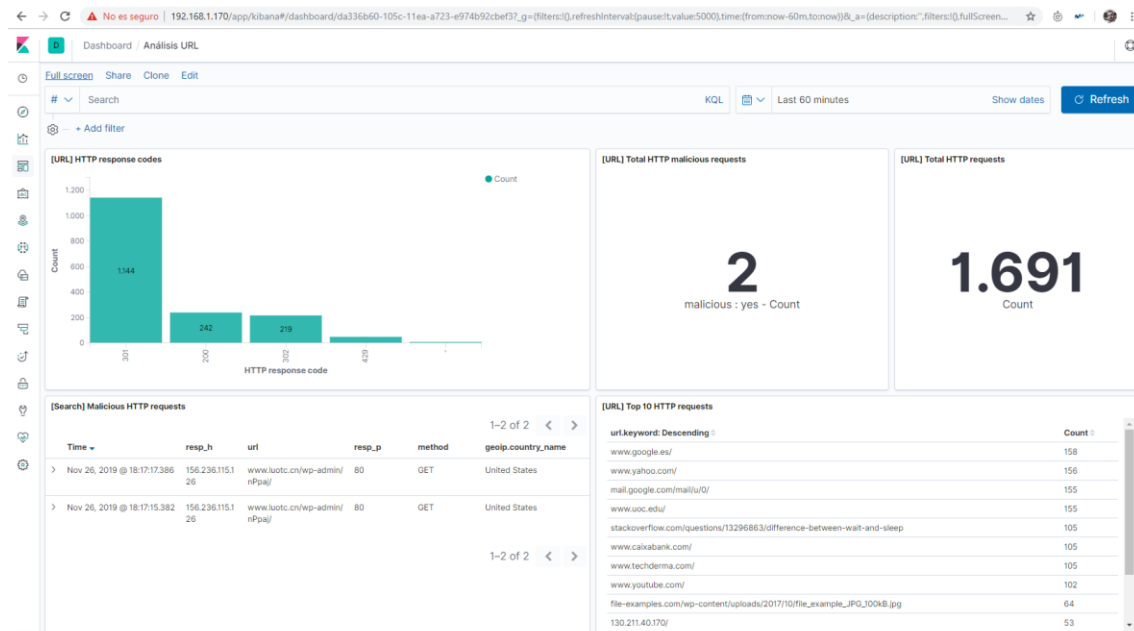


Figura 39. Dashboard de monitorización “Análisis URL”

3.2.3.3 Dominios

El dashboard de Dominios contendrá la siguiente información:

- Total de consultas dns realizadas.
- Top 10 de consultas dns realizadas.
- Total resoluciones dns maliciosas detectadas.

- Lista de consultas dns maliciosas detectadas.

A continuación, se describe las características de cada objeto de visualización:

Total de consultas dns realizadas

- Nombre: [Domains] Total DNS requests
- Tipo visualización: Metric
- Datos origen: zeek-dns-logs
- Metrics – Aggregation: Count

Tras aplicar los cambios, obtenemos el objeto en cuestión:

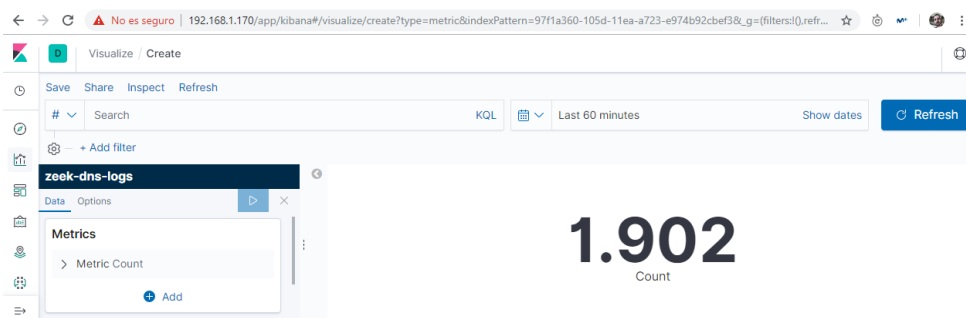


Figura 40. Total de consultas dns realizadas

Top 10 de consultas dns realizadas

Para este indicador, se requiere previamente la construcción de una búsqueda a partir de los logs del índice zeek-dns-logs. Para ello, se accederá a la sección de *Discover*, y se realizará la siguiente configuración:

- Filtrar por el campo query para un valor:
NOT query: *\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00

Una vez aplicado el filtro, procedemos a guardar la búsqueda como “[Search] DNS requests”.

A continuación, se dará de alta el objeto de visualización:

- Nombre: [Domains] Top 10 dns requests
- Tipo visualización: Data Table
- Datos origen: [Search] DNS requests
- Metrics – Aggregation: Count
- Buckets:
 - Split rows
 - Aggregation: Terms
 - Field: query.keyword
 - Order by: Metric : Count
 - Order: Descending
 - Size: 10

Tras aplicar los cambios, obtenemos el objeto en cuestión:

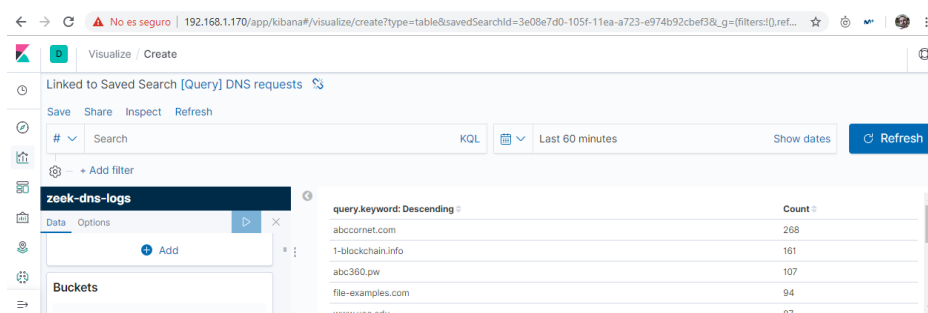


Figura 41. Top 10 de consultas dns realizadas

Total consultas dns maliciosas detectadas

- Nombre: [Domain] Total DNS malicious requests
- Tipo visualización: Metric
- Datos origen: zeek-dns-logs
- Metrics – Aggregation: Count
- Buckets:
 - Split group
 - Aggregation: Filters
 - Filter 1: “malicious : yes”

Tras aplicar los cambios, obtenemos el objeto en cuestión:

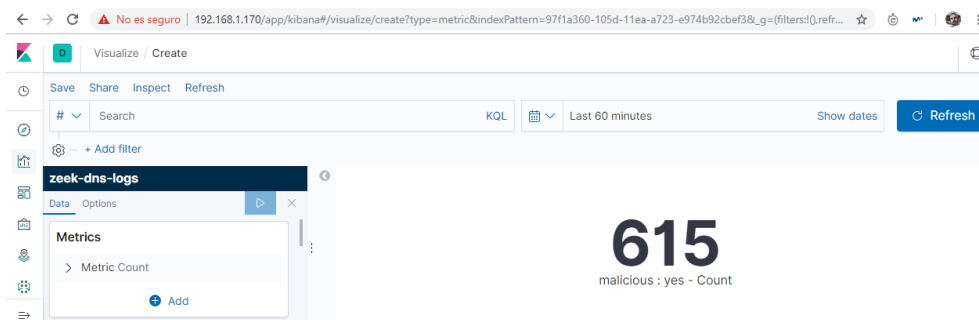


Figura 42. Total consultas dns maliciosas detectadas

Lista de consultas dns maliciosas detectadas

Para este indicador, se requiere únicamente la construcción de una búsqueda a partir de los logs del índice zeek-http-logs. Para ello, se accederá a la sección de *Discover*, y se realizará la siguiente configuración:

- Filtrar por el campo malicious para un valor “yes”.
- Se agregarán las siguientes columnas al listado: query, resp_h (servidor dns que ha dado respuesta), proto (protocolo), rcode_name (código de respuesta)

Una vez aplicada la configuración, guardaremos la búsqueda como [Search] Malicious DNS requests” y posteriormente la agregaremos al dashboard para su visualización.

El aspecto de la tabla una vez aplicados los cambios es la siguiente:

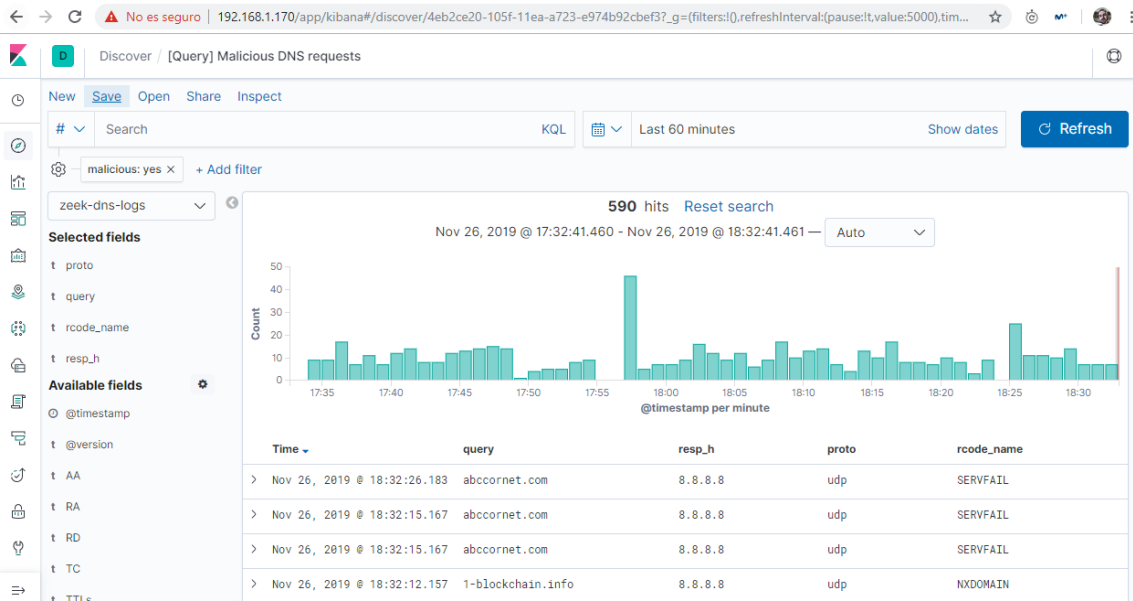


Figura 43. Lista de consultas dns maliciosas detectadas

Construcción dashboard

Una vez que ya tenemos todos los objetos de visualización, se dará de alta un nuevo dashboard, con nombre “Análisis Dominios”, e iremos añadiendo todos los componentes visuales al Canvas disponible.

Tras agregar los elementos visuales, el resultado del dashboard obtenido es el siguiente:

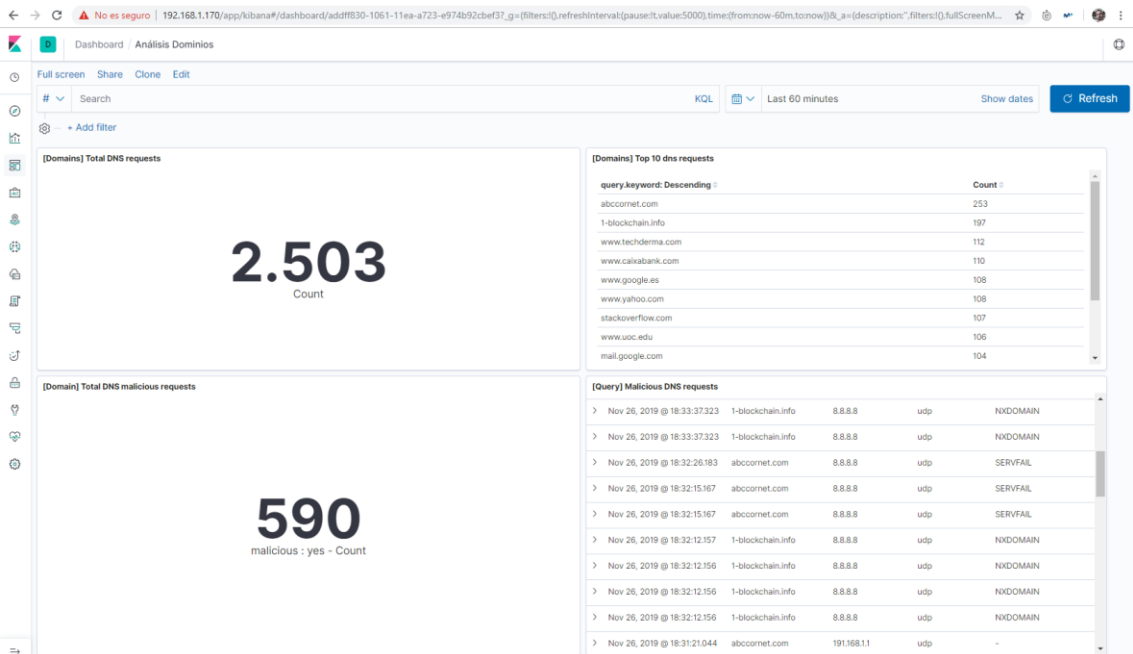


Figura 44. Dashboard de monitorización “Análisis Dominios”

3.2.3.4 Ficheros

El dashboard de Ficheros contendrá la siguiente información:

- Total de ficheros descargados.
- Total bytes de ficheros descargados.
- Top 10 de extensiones de ficheros descargadas.
- Total ficheros maliciosos descargados.
- Lista de los últimos ficheros maliciosos descargados.

A continuación, se describe las características de cada objeto de visualización:

Total de ficheros descargados

- Nombre: [Files] Total Files requests
- Tipo visualización: Metric
- Datos origen: zeek-files-logs
- Metrics – Aggregation: Count

Tras aplicar los cambios, obtenemos el objeto en cuestión:

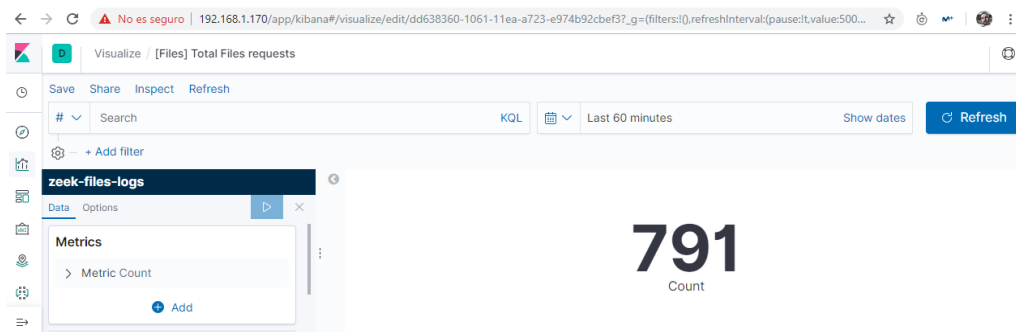


Figura 45. Total de ficheros descargados

Total bytes de ficheros descargados

- Nombre: [Files] Download bytes
- Tipo visualización: Metric
- Datos origen: zeek-files-logs
- Metrics – Aggregation: Sum
- Field: total_bytes
- Custom label: bytes

Tras aplicar los cambios, obtenemos el objeto en cuestión:

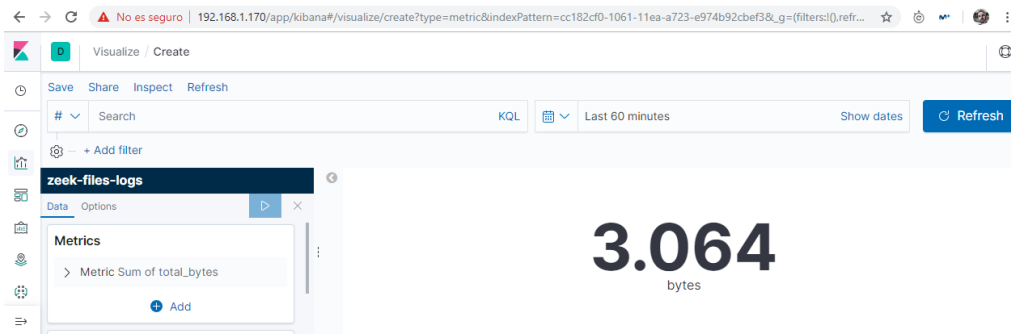


Figura 46. Total bytes de ficheros descargados

Top 10 de extensiones de ficheros descargadas

- Nombre: [Files] Top 10 file extensions
- Tipo visualización: Data Table
- Datos origen: zeek-files-logs
- Metrics – Aggregation: Count
- Buckets:
 - Split rows
 - Aggregation: Terms
 - Field: extension.keyword
 - Order by: Metric : Count
 - Order: Descending
 - Size: 10

Tras aplicar los cambios, obtenemos el objeto en cuestión:

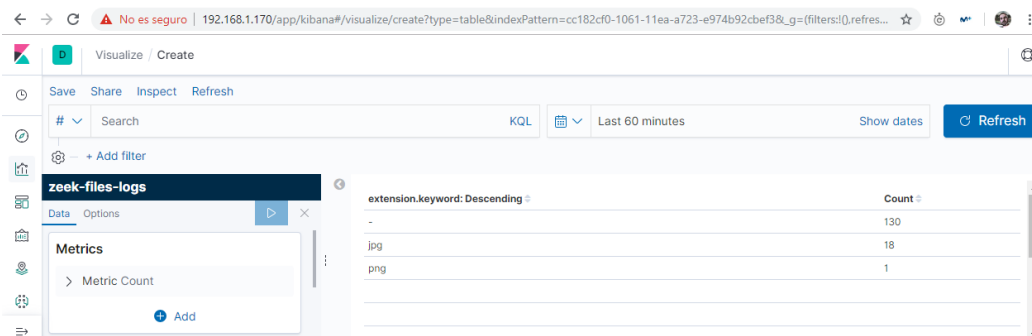


Figura 47. Top 10 de extensiones de ficheros descargadas

Total ficheros maliciosos descargados

- Nombre: [Files] Total malicious files
- Tipo visualización: Metric
- Datos origen: zeek-files-logs
- Metrics – Aggregation: Count
- Buckets:
 - Split group
 - Aggregation: Filters
 - Filter 1: "malicious : yes"

Tras aplicar los cambios, obtenemos el objeto en cuestión:

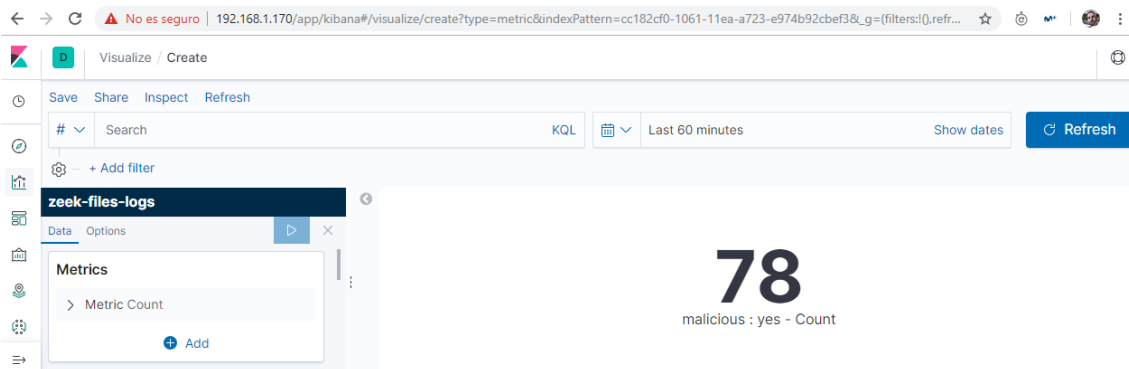


Figura 48. Total ficheros maliciosos descargados

Lista de ficheros maliciosos descargados

Para este indicador, se requiere únicamente la construcción de una búsqueda a partir de los logs del índice zeek-files-logs. Para ello, se accederá a la sección de *Discover*, y se realizará la siguiente configuración:

- Filtrar por el campo malicious para un valor “yes”.
- Se agregarán las siguientes columnas al listado: filename, hostname, uri, md5. El resto se mantendrá ocultas.

Una vez aplicada la configuración, guardaremos la búsqueda como “[Search] Files malicious downloads” y posteriormente la agregaremos al dashboard para su visualización.

El aspecto de la tabla una vez aplicados los cambios es la siguiente:

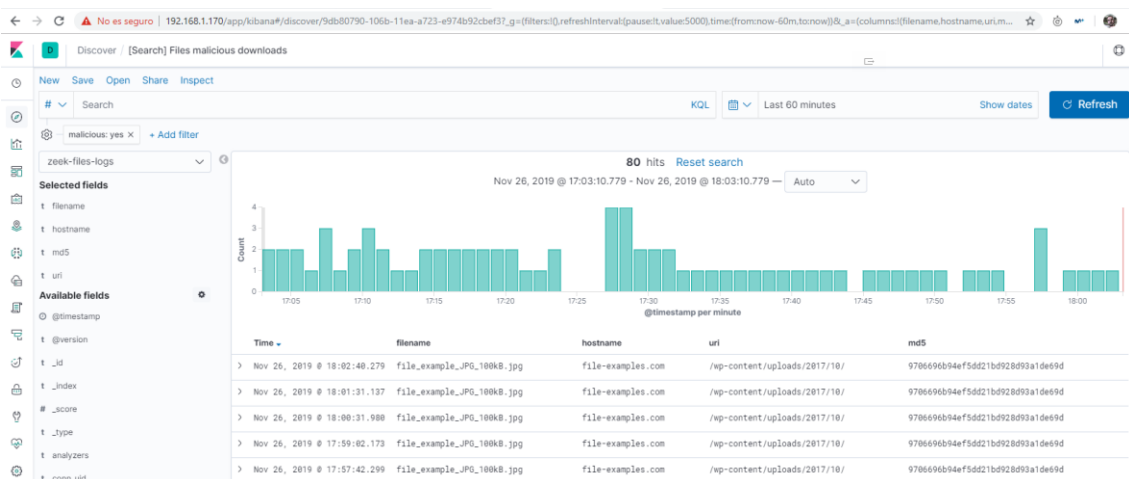


Figura 49. Lista de ficheros maliciosos descargados

Construcción dashboard

Una vez que ya tenemos todos los objetos de visualización, se dará de alta un nuevo dashboard, con nombre “Análisis Ficheros”, e iremos añadiendo todos los componentes visuales al Canvas disponible.

Tras agregar los elementos visuales, el resultado del dashboard obtenido es el siguiente:

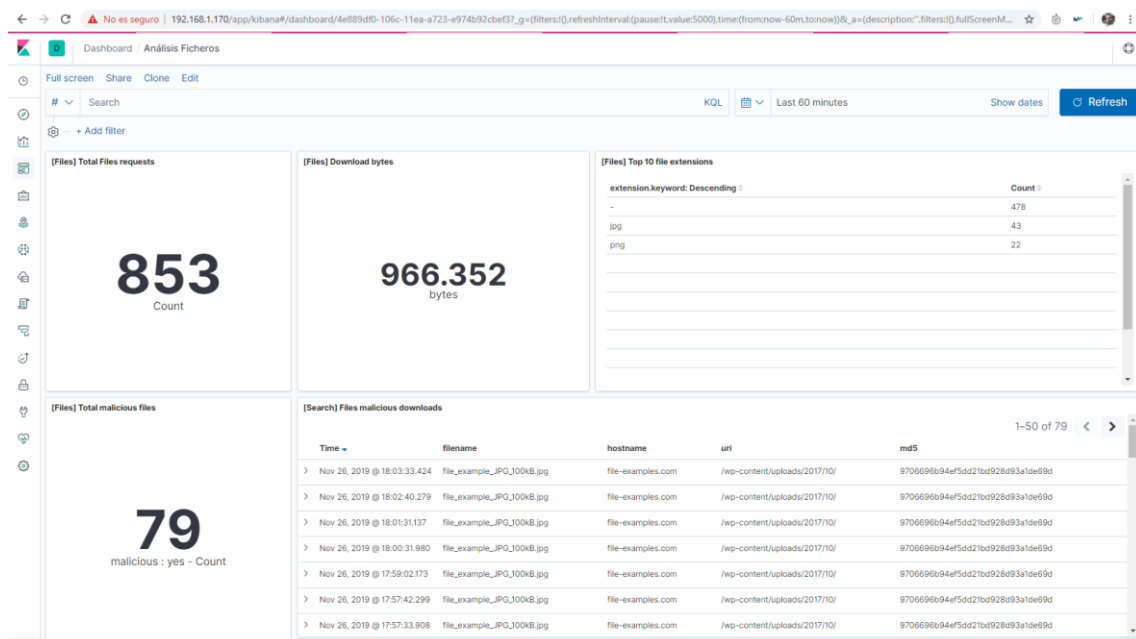


Figura 50. Dashboard de monitorización “Análisis Ficheros”

4. Validaciones

En este apartado se detallan los resultados del plan de pruebas definido en la fase de análisis. Para la ejecución del mismo, se requiere tener todas las máquinas virtuales arrancadas, y todos los servicios activados.

4.1 Validación integración Zeek IDS con stack Elastic

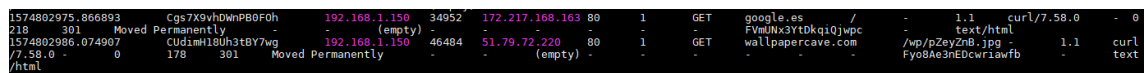
En esta primera fase, se validará la integración de Zeek IDS con el stack Elastic, con lo que se verificará que Zeek está funcionando correctamente, así como la ingesta de las trazas en Elasticsearch.

El primer paso consiste en ejecutar varias peticiones HTTP en la máquina uoc-server-host para activar la monitorización.

Para ello, ejecutaremos los siguientes comandos por consola:

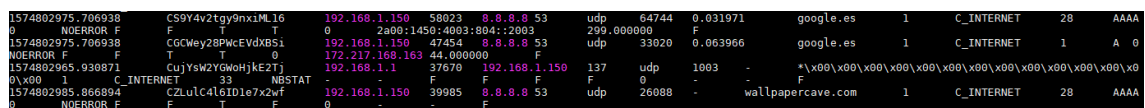
```
$ curl -s http://google.es --max-time 2 > /dev/null
$ curl -s http://wallpapercave.com/wp/pZeyZnB.jpg --max-time 2 > /dev/null
```

En primer lugar, validaremos que se han generado los ficheros de trazas Zeek y que contienen los registros correspondientes a las pruebas ejecutadas.



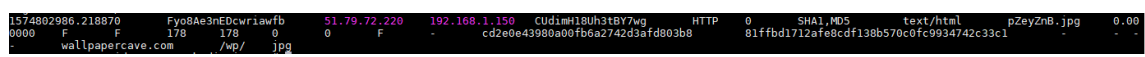
```
1574802975.866893 Cgs7X9vhDwnPB0Foh 192.168.1.150 34952 172.217.168.163 80 1 GET google.es / 1.1 curl/7.58.0 0
218 301 Moved Permanently (empty) - - - - -
1574802986.074907 CUDimH18Uh3tBY7wg 192.168.1.150 46484 51.79.72.220 80 1 GET wallpapercave.com /wp/pZeyZnB.jpg 1.1 curl
/7.58.0 0 178 301 Moved Permanently (empty) - - - - -
/html Fyo8Ae3nEdCwr1awfb - - - - -
```

Figura 51. Trazas de fichero http.log en primera fase pruebas



```
1574802975.706938 CS9Y4v2tgy9nxiML16 192.168.1.150 58023 8.8.8.8 53 udp 64744 0.031971 google.es 1 C_INTERNET 28 AAAA
0 NOERROR F T T 2a00:1450:4003:804::2003 299.000000 F
1574802975.706938 CCGwey28PwceVdXBSi 192.168.1.150 47454 8.8.8.8 53 udp 33020 0.063966 google.es 1 C_INTERNET 1 A 0
NOERROR F T T 0 172.217.168.163 44.000000 F
1574802965.938871 CujYsW2YGwohJKEZTj 192.168.1.1 37678 192.168.1.150 137 udp 1803 *x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00
0x00 1 C_INTERNET 33 NBSTAT F F F F F 0
1574802985.866894 CZLuLc4l6ID1e7x2wf 192.168.1.150 39985 8.8.8.8 53 udp 26088 wallpapercave.com 1 C_INTERNET 28 AAAA
0 NOERROR F T F 0 - - - - -
```

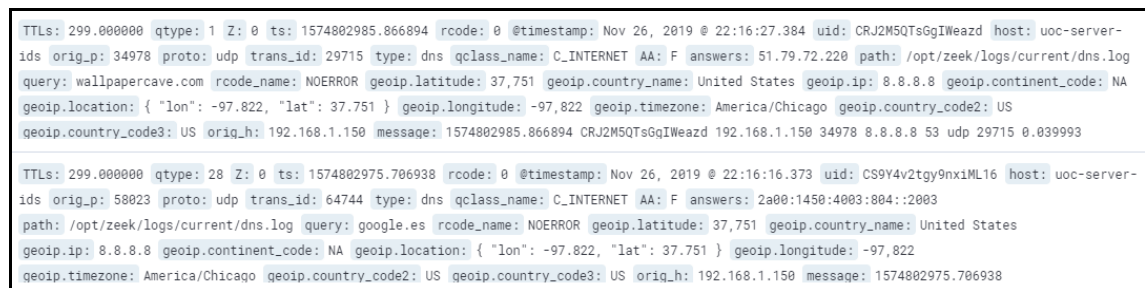
Figura 52. Trazas de fichero dns.log en primera fase pruebas



```
1574802986.218870 Fyo8Ae3nEdCwr1awfb 51.79.72.220 192.168.1.150 CUDimH18Uh3tBY7wg HTTP 0 SHA1_MDS text/html pZeyZnB.jpg 0.00
0000 F F 178 178 0 cd2ee043990a00fbae2742d3af8093b8 81ffbdl712afe8cdf138b570c0fc9934742c33c1
wallpapercave.com /wp/ jpg
```

Figura 53. Trazas de fichero files.log en primera fase pruebas

En segundo lugar, validaremos que las trazas están disponibles en Elastic, y accesibles mediante la herramienta de *Discover* de Kibana.



```
TTLs: 299.000000 qtype: 1 Z: 0 ts: 1574802985.866894 rcode: 0 @timestamp: Nov 26, 2019 @ 22:16:27.384 uid: CRJ2M5QTsGgIWeazd host: uoc-server-ids orig_p: 34978 proto: udp trans_id: 29715 type: dns qclass_name: C_INTERNET AA: F answers: 51.79.72.220 path: /opt/zeek/logs/current/dns.log query: wallpapercave.com rcode_name: NOERROR geoup.latitude: 37,751 geoup.country_name: United States geoup.ip: 8.8.8.8 geoup.continent_code: NA geoup.location: { "lon": -97.822, "lat": 37.751 } geoup.longitude: -97,822 geoup.timezone: America/Chicago geoup.country_code2: US geoup.country_code3: US orig_h: 192.168.1.150 message: 1574802985.866894 CRJ2M5QTsGgIWeazd 192.168.1.150 34978 8.8.8.8 53 udp 29715 0.039993

TTLs: 299.000000 qtype: 28 Z: 0 ts: 1574802975.706938 rcode: 0 @timestamp: Nov 26, 2019 @ 22:16:16.373 uid: CS9Y4v2tgy9nxiML16 host: uoc-server-ids orig_p: 58023 proto: udp trans_id: 64744 type: dns qclass_name: C_INTERNET AA: F answers: 2a00:1450:4003:804::2003 path: /opt/zeek/logs/current/dns.log query: google.es rcode_name: NOERROR geoup.latitude: 37,751 geoup.country_name: United States geoup.ip: 8.8.8.8 geoup.continent_code: NA geoup.location: { "lon": -97.822, "lat": 37.751 } geoup.longitude: -97,822 geoup.timezone: America/Chicago geoup.country_code2: US geoup.country_code3: US orig_h: 192.168.1.150 message: 1574802975.706938
```

Figura 54. Registro de logs en Kibana de registros DNS en primera fase de pruebas

En este punto, se ha validado que la ingesta de información se está realizando correctamente en el stack Elastic, tanto para los registros dns, como para el resto de ficheros de trazas (conn.log, http.log, files.log)

4.2 Validación gestión listas de reputación

En la segunda fase del plan de pruebas, se validará el circuito de las listas de reputación, desde la generación de los datos de salida a partir de las listas de reputación configuradas y la generación de los ficheros diccionario con los programas Python desarrollados.

También se validará que, al modificar las listas de reputación, la información actualizada se genera de forma correcta y está disponible en los ficheros diccionario para su uso por Logstash.

En primer lugar, se validará que todas las URL de los feeds provistos por Minemeld están funcionando correctamente y devuelven información. Para ello, introduciremos cada URL en el navegador web y comprobaremos la información de salida.

Feed IPs

Url: <https://192.168.1.180/feeds/ips-output>

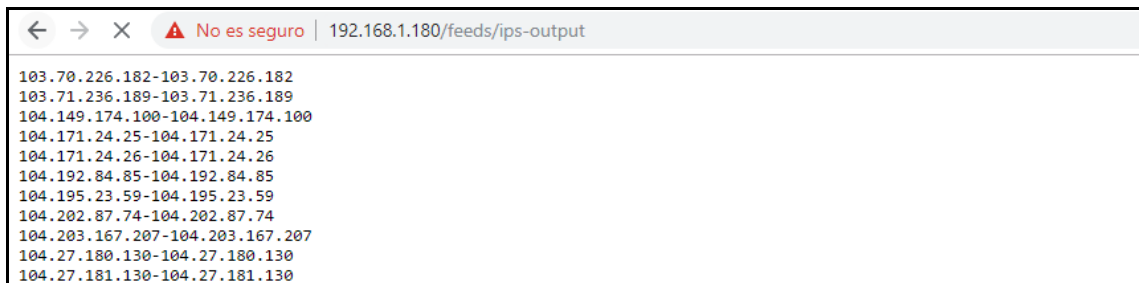


Figura 55. Contenido del feed de IPs de Minemeld

Feed Dominios

Url: <https://192.168.1.180/feeds/domains-output>



Figura 56. Contenido del feed de Dominios de Minemeld

Feed URLs

Url: <https://192.168.1.180/feeds/urls-output>

```
← → ↻ No es seguro | 192.168.1.180/feeds/urls-output
http://103.1.250.236:8080/4appverif.chm
http://104.168.198.208/wordupd.tmp
http://104.168.61.47/armv4l
http://104.168.61.47/armv5l
http://104.168.61.47/armv6l
http://104.168.61.47/i586
http://104.168.61.47/i686
http://104.168.61.47/m68k
http://104.168.61.47/mips
http://104.168.61.47/mipse1
http://104.168.61.47/powerpc
```

Figura 57. Contenido del feed de URLs de Minemeld

Feed Hashes

Url: <https://192.168.1.180/feeds/hash-output>

```
← → ↻ No es seguro | 192.168.1.180/feeds/hash-output
001dd76872d80801692ff942308c64e6
002325a0a67fded0381b5648d7fe9b8e
00dbb9e1c09dbdafb360f3163ba5a3de
0149b7bd7218aab4e257d28469fddb0d
01e0dc079d4e33d8edd050c4900818da
02c65973b6018f5d473d701b3e7508b2
034374db2d35cf9da6558f54cec8a455
0496e3b17cf40c45f495188a368c203a
052ec04866e4a67f31845d656531830d
065e63afdfa539727f63af7530b22d2f
0908d8b3e459551039bade50930e4c1b
```

Figura 58. Contenido del feed de Hashes de Minemeld

Una vez comprobado que los feeds de Minemeld están funcionando correctamente, se procede a la ejecución de los diferentes scripts Python para el volcado de la información al sistema de ficheros en la máquina uoc-server-ids.

Para ello, se accede al directorio,

```
/opt/zeek/minemeld
```

y se ejecuta los 4 programas Python mediante los siguientes comandos:

```
$ python minemeld-parser-ip.py
$ python minemeld-parser-domain.py
$ python minemeld-parser-url.py
$ python minemeld-parser-hash.py
```

Tras la ejecución de los programas, se aprecia que se han generado los 4 ficheros de salida:

```
root@uoc-server-ids:/opt/zeek/dictionary# ll
total 3148
-rw-r--r-- 1 root root 705064 nov 26 22:36 domains.yml
-rw-r--r-- 1 root root 13677 nov 26 22:36 hashes.yml
-rw-r--r-- 1 root root 2380861 nov 26 22:36 ips.yml
-rw-r--r-- 1 root root 112888 nov 26 22:36 urls.yml
```

Figura 59. Ficheros generados por los programas Python

Ahora, se validará que los cambios realizados sobre las listas de reputación origen, acaban actualizando la información generada en los feeds de Minemeld

y que los cambios son visibles en los ficheros generados por los programas Python para su uso.

Para validar este punto, se realizará una prueba controlada con uno de los feeds, que será del de urls.

En primer lugar, validaremos que no existe, en el fichero diccionario generado anteriormente por el programa Python de dominios, el dominio google.ru.

```
root@uoc-server-ids:/opt/zeek/dictionary#  
root@uoc-server-ids:/opt/zeek/dictionary# grep google.ru domains.yml  
root@uoc-server-ids:/opt/zeek/dictionary#
```

Figura 60. Validación de no existencia de dominio en fichero domains.yml

A continuación, se modificará el fichero baddomains.txt para incluir el dominio google.ru. La ubicación del fichero, en la máquina uoc-server-ids, sería la siguiente:

```
/var/www/html/baddomains.txt
```

Una vez introducido el dominio, accederemos a Minemeld, y forzaremos un refresco del nodo de entrada. Al hacerlo, ya se visualiza que para ese nodo, aparece un indicador, cuando anteriormente marcaba 0.

uoc-domain-input NODE	
STATUS	
CLASS	minemeld.ft.http.HttpFT
PROTOTYPE	minemeldlocal.uoc-domain-input
STATE	STARTED
LAST RUN	2019-11-26 22:47:24 +0100 SUCCESS
# INDICATORS	1

Figura 61. Validación de no existencia de dominio en fichero domains.yml

A continuación, lanzamos de nuevo el programa minemeld-parser-domain.py, y de nuevo lanzamos la consulta del dominio google.ru sobre el fichero diccionario de dominios.

```
root@uoc-server-ids:/opt/zeek/dictionary#  
root@uoc-server-ids:/opt/zeek/dictionary# grep google.ru domains.yml  
"google.ru": malicious  
root@uoc-server-ids:/opt/zeek/dictionary#
```

Figura 62. Validación de existencia de dominio en fichero domains.yml

Tras la búsqueda, se aprecia que ahora el dominio google.ru sí que aparece en el fichero diccionario, por lo que esto valida que el circuito completo de la gestión de las listas de reputación funciona correctamente.

4.3 Validación detección actividades maliciosas

En esta fase, se validará que el sistema de matching aplicado en Logstash para la detección de actividades sospechosas está funcionando correctamente.

Para ello, se accederá a la sección *Discover* de Kibana, y se seleccionará el índice `zeek-dns-index`.

A continuación, se lanzará una consulta desde la máquina `host`, para verificar que el registro correspondiente a tráfico lícito, aparece en los resultados de la búsqueda.

El comando a ejecutar sería el siguiente:

```
$ curl -s http://google.es --max-time 2 > /dev/null
```

Y el registro correspondiente que aparecería en Kibana es:

Time	message	malicious
> Nov 26, 2019 @ 22:59:36.372	1574805575.022914 C3Txrv148HoFoRx9Uj 192.168.1.150 47209 8.8.8.8 53 udp 39737 0.032012 google.es 1 - C_INTERNET 1 A 0 NOERROR F F T T 0 216.58.211.35 24.000000 F	

Figura 63. Validación registro dns de consulta de dominio lícito

A continuación, lanzaremos de nuevo otro comando desde la máquina `host`, pero en este caso utilizaremos el dominio que previamente hemos marcado como malicioso.

```
$ curl -s http://google.ru --max-time 2 > /dev/null
```

Tras la ejecución, el registro aparece en Kibana, y en este caso, se aprecia que se ha informado un nuevo atributo que indica que corresponde a una actividad maliciosa.

Time	message	malicious
> Nov 26, 2019 @ 23:02:15.767	1574805734.246862 CWHr103PrzI9jbrssa 192.168.1.150 36349 8.8.8.8 53 udp 38995 0.032038 google.ru 1 yes C_INTERNET 1 A 0 NOERROR F F T T 0 172.217.16.227 299.000000 F	

Figura 64. Validación registro dns de consulta de dominio malicioso.

Tras la prueba realizada, se valida que el `matching` de las trazas de Zeek con los ficheros diccionario están funcionando correctamente, y que la información se traslada a Elasticsearch y Kibana para su posterior explotación.

4.4 Validación dashboard de monitorización

En la cuarta fase, se validará que los dashboards de monitorización muestran la información en tiempo real del tráfico de red, y que las actividades sospechosas son monitorizadas también en base a las listas de reputación.

Para ello, se ha creado un Shell script, y unos ficheros diccionario de diferentes URLs, tanto lícitas como maliciosas, que ejecutará peticiones HTTP de forma continua para validar que la información representada en los dashboards se va actualizando según se espera.

Sobre la máquina `host`, se copiarán los siguientes ficheros:

```

root@uoc-server-host:/home/ilogyc# ll
total 32
-rw-r--r-- 1 root root 129 nov 26 21:52 bad_files.txt
-rw-r--r-- 1 root root 9565 nov 26 20:29 bad_traffic.txt
-rwxr-xr-x 1 root root 447 nov 26 21:34 generate_traffic.sh
-rw-r--r-- 1 root root 1224 nov 26 21:51 good_files.txt
-rw-r--r-- 1 root root 4245 nov 26 21:38 good_traffic.txt

```

Figura 65. Ficheros para validación cuarta fase pruebas.

- El fichero good_traffic.txt, contiene una lista de URLs lícitas.
- El fichero bad_traffic.txt, contiene una lista de URLs maliciosas, obtenida a partir de los feeds de Minemeld.
- El fichero good_files.txt, contiene una lista de URLs para la descarga de ficheros lícitos.
- El fichero bad_files.txt, contiene una lista de URLs para la descarga de ficheros maliciosos.
- Y el fichero generate_traffic.sh es el script que ejecuta las peticiones HTTP partiendo del contenido del resto de fichero.

Antes de ejecutar el script, accederemos al Dashboard de Visión Global y nos aseguraremos que todos los indicadores y tablas muestran 0 resultados.

En ese momento, lanzaremos el script y esperaremos 5 minutos a que se haya generado el tráfico suficiente como para que todos los indicadores muestren información. El comando para la ejecución del script:

```
$ sh generate_traffic.sh
```

Pasados los 5 minutos, accederemos a los diferentes dashboards de monitorización y verificaremos que todos los indicadores muestran información.

Se adjunta una captura de pantalla del dashboard Visión Global con los indicadores actualizados, y donde se aprecia que se ha detectado tráfico malicioso.

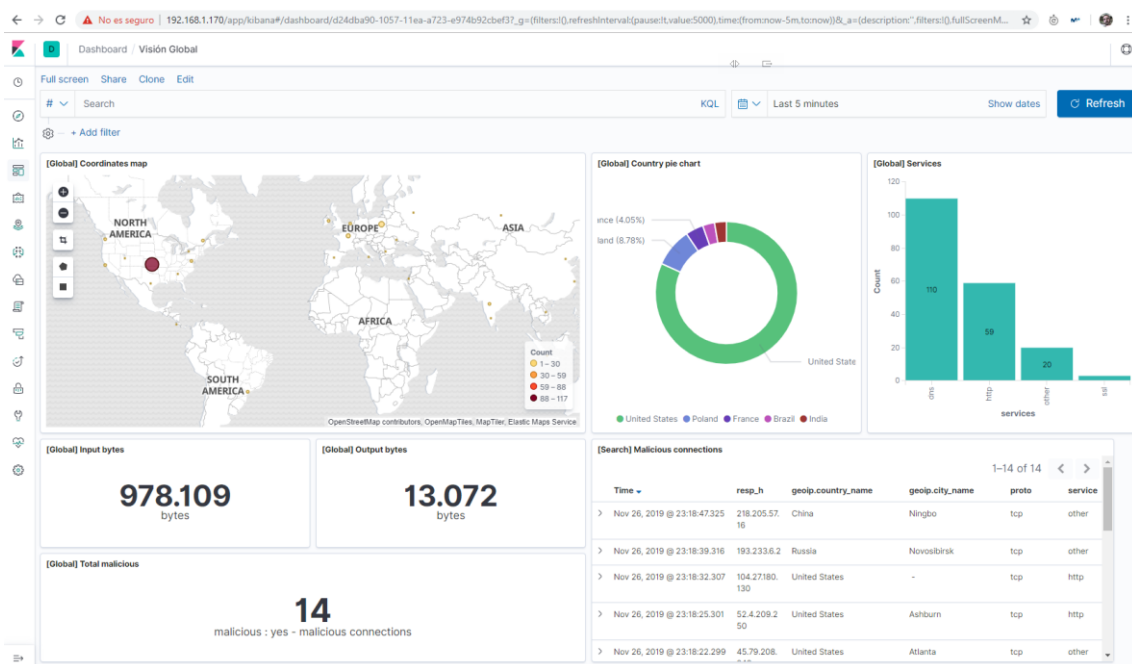


Figura 66. Dashboard Visión Global con actividad maliciosa.

5. Conclusiones

5.1 Conclusiones del trabajo

Una vez finalizada la construcción del trabajo, se puede concluir que se ha cumplido con el objetivo principal del mismo, que indicaba:

“El objetivo principal de Trabajo de Final de Master consiste en desplegar un sistema de detección de intrusos, basado en la solución Zeek IDS, cuya información será ingestada en el stack de Elastic y visualizada en un dashboard de monitorización continua que permita detectar comportamientos anómalos en función del tráfico de entrada/salida en base a listas de reputación.”

El sistema dispone de todos los componentes requeridos para llevar a cabo la monitorización y detección de actividades sospechosas, y sienta las bases para el diseño de un sistema de mayor embergadura que permita dar uso a redes con un mayor tráfico de red, más propias del ámbito empresarial, a un coste reducido por el hecho de utilizar soluciones de código abierto y de libre distribución.

El uso de listas de reputación es clave para el sistema, y se ha podido observar durante la fase de análisis que existen multitud de ellas, tanto de empresas privadas como de entes públicas, con el objetivo de fortalecer los sistemas de seguridad y compartir datos cruciales para minimizar en la medida de lo posible ataques informáticos a diferentes escalas.

El desempeño del software Zeek IDS, y las posibilidades que ofrece, certifican que ha sido una buena elección como motor de detección de intrusos, si bien cabe destacar que se han detectado ciertas limitaciones en cuanto al registro de tráfico originado a partir del protocolo HTTP seguro (https). Para este tipo de tráfico, Zeek no deja registro en el fichero http.log ni en el fichero files.log, por lo que la detección de actividades sospechosas en este caso se ha delegado a otros tipos de ficheros, como el de conexiones y el de consultas dns.

En cuanto al uso del stack de Elastic, para la ingesta, tratamiento, búsqueda y visualización de información, destacar que es una herramienta muy completa y que ha cumplido con las expectativas del trabajo. Se ha podido tratar la información aplicando una configuración mínima en los componentes, y a nivel visual ofrece muchas herramientas para la generación de Dashboards muy completos, con una interfaz gráfica amigable e intuitiva, que facilita la monitorización requerida en este tipo de sistemas.

Por último, se concluye que la elección de Minemeld también ha sido acertada, porque se ha podido completar de forma ágil toda la gestión de listas de reputación y se han elaborado los feeds de salida acorde a las necesidades del trabajo sin requerir demasiado esfuerzo. A destacar que la herramienta dispone de poca documentación funcional, por lo que podría suponer un problema en caso de requerir implementar escenarios más complejos.

5.2 Revisión objetivos

En el apartado de conclusiones del trabajo ya se confirma el cumplimiento del objetivo principal del trabajo, que se demuestra con la construcción del sistema de monitorización de actividades sospechosas basado en los componentes Zeek IDS y stack Elastic.

No obstante, se identificaron adicionalmente una serie de subobjetivos cuyo grado de cumplimiento pasa de detallarse a continuación.

Subobjetivo 1

Definición: *Analizar el sistema de detección de intrusos Zeek IDS, e identificar las capacidades que nos ofrecen este tipo de soluciones para el propósito del proyecto.*

Para este subobjetivo, el grado de consecución ha sido total. Se ha podido analizar con bastante detalle las funcionalidades ofrecidas para el propósito del trabajo, y además se ha ahondado en las capacidades propias del framework para ampliar las funcionalidades ofrecidas de base, mediante la creación de un script adicional para poder registrar en el fichero files.log los nombres de los ficheros analizados.

Adicionalmente, se estudió las posibilidades para que el análisis de actividades sospechosas se realizara directamente sobre Zeek, y no sobre Logstash como se ha implementado, mediante el uso del framework de *threat intelligence* incluido en el sistema.

Subobjetivo 2

Definición: *Conocer el funcionamiento del stack Elastic para la monitorización de información, y determinar cómo este puede complementarse con otros sistemas de seguridad.*

Para este subobjetivo, el grado de consecución también ha sido completo. El stack Elastic, por sus características, permite poder tratar información de diferentes fuentes, y las funcionalidades de tratamiento, normalización y consulta de información es realmente útil para poder explotar cualquier tipo de dato. Además, se permite poder definir dashboards de monitorización que se adaptan a las necesidades de cada caso.

Subobjetivo 3

Definición: *Conocer los diferentes tipos de listas de reputación existentes, sus propósitos, e identificar aquellas que podrán ser utilizadas en el trabajo para la identificación de tráfico sospechoso.*

Este subobjetivo también ha sido logrado en su totalidad. En la fase de análisis se identificaron numerosas listas de reputación, tanto públicas como privadas, con información diversa para la detección de actividades sospechosas.

Se identificaron listas globales de direcciones IP de máquinas utilizadas para actividades maliciosas, y algunas listas con información específica incluso para

ciertos tipos de Malware, como Emote, WannaCry, Kovter y otros. Además de listas de direcciones IP, también se identificaron listas de dominios con mala reputación, hashes MD5 y SHA256 de ficheros maliciosos y URLs maliciosas. El reto fue identificar listas de reputación actualizadas y fiables.

A destacar que, durante la fase de análisis, se detectó que muchas de las listas de reputación son alimentadas por información de terceros a partir de la detección de actividades sospechosas en los propios sistemas de seguridad de empresas o personales, que recopilan la información y la envían a estos recolectores de información para poner a disposición de terceros los datos actualizados.

Subobjetivo 4

Definición: Llevar a cabo la construcción de la solución, integrando los diferentes componentes software, y validar el correcto funcionamiento del sistema en su conjunto.

Este subobjetivo, muy alineado con el objetivo principal del trabajo, también se ha cumplido en su totalidad. El apartado de construcción del trabajo indica el proceso a paso a paso para el desarrollo del sistema, desde la instalación del software necesario, hasta su configuración y validación. A su vez, la construcción de los dashboards de monitorización y la ejecución del plan de pruebas han podido validar la efectividad del sistema para la detección de actividades sospechosas.

Subobjetivo 5

Definición: Identificar qué otras aplicaciones reales podrían derivarse del uso de los componentes utilizados, para la detección de otros tipos de amenazas.

El trabajo ha consistido en la detección básica de amenazas, a partir de los ficheros de trazas generados por Zeek IDS, y contrastando la información con las listas de reputación públicas, pero el sistema podría detectar otros tipos de amenazas, como serían:

- Detección de ataques de fuerza bruta.
- Detección de tráfico malicioso para otros protocolos (ntp, rdp, ssl)
- Detección de ataques de tipo SQLinjection.
- Detección de ataques de Denegación de Servicio.
- Detección de escaneos de puertos.
- Detección de ataques de tipo Cross Site Scripting (XSS)

Y también se podría utilizar la información recolectada para llevar a cabo análisis forenses.

5.3 Seguimiento y metodología

La ejecución del trabajo durante las primeras fases del mismo se ha realizado según estaba planificado. La ejecución de las diferentes tareas planteadas en el Plan de Trabajo se llevó a cabo en tiempos y sin desvíos en la planificación.

Sin embargo, en la fase de construcción, se detectó que Zeek IDS carecía de ciertas funcionalidades requeridas para el análisis de tráfico mediante el protocolo HTTP seguro, lo cual requirió un tiempo adicional de investigación no previsto inicialmente.

Esta situación provocó un desajuste en la planificación, que pudo mitigarse mediante los márgenes de contingencia previstos en la planificación de las diferentes tareas, para asegurar completar la entrega en fechas.

La metodología planteada para el proyecto fue adecuada por el tipo de trabajo desarrollado, con las fases de análisis, construcción y pruebas, como los habituales proyectos de desarrollo de software. Si bien es cierto que se hubiera requerido algo más de tiempo para la fase de construcción, al requerir un mayor esfuerzo que en el resto de fases del trabajo, lo cual habría permitido profundizar algo más en las capacidades de Zeek IDS para haber implementado el sistema según se había previsto en la fase de análisis.

Durante la ejecución del proyecto, se ha compartido con el tutor el planteamiento y decisiones tomadas en la definición de la solución del sistema, lo cual ha permitido garantizar que el diseño de la solución era correcto, anticipando así posibles problemas de compatibilidad o carencias. A su vez, se ha tratado de informar semanalmente de los avances del desarrollo en base a la planificación definida.

5.4 Dificultades encontradas

Las principales dificultades encontradas han sido las siguientes:

- En el caso de Zeek IDS, se detectó que para peticiones a páginas web seguras (HTTPS), los ficheros de trazas `http.log` y `files.log` no registraban el tráfico generado, lo cual suponía un problema porque este tipo de tráfico no quedaba registrado y por tanto impedía poder ser monitorizado como estaba previsto.
- Se tomó la decisión de basarnos en los ficheros de trazas de `conn.log` y `dns.log` para la detección de tráfico malicioso para peticiones HTTP seguras.
- En el caso de Minemeld, la falta de una documentación completa del funcionamiento de la herramienta hizo que el avance en esta parte de la fase de construcción fuera algo más lenta de lo previsto inicialmente. Sin embargo, al ser una herramienta intuitiva, y dado que los casos de uso no eran demasiado complejos, se pudo aplicar las configuraciones para disponer de un sistema totalmente funcional y alineado con el propósito del trabajo.
- En el caso del stack Elastic, la dificultad ha radicado principalmente en familiarizarse con la herramienta para la construcción de los dashboards. No obstante, la documentación disponible es muy completa y se dispone

de mucha información adicional en foros de terceros, lo cual permitió poder realizar el desarrollo según estaba previsto.

5.5 Líneas de trabajo futuras

Como líneas de trabajo futuras se identifican las siguientes:

- Implementar en Zeek unos scripts que permitan generar trazas de log para peticiones HTTP seguras en los ficheros http.log y files.log, para asegurar que el sistema monitorizar todo el tráfico requerido por este tipo de soluciones.
- Implementar en Zeek una detección de malware basado en firmas, y mecanismos de detección para otros tipos de ataques informáticos, como XSS, SQLInjection o DDoS.
- Realizar una comparativa en detalle de los diferentes sistemas de intrusión open source disponibles, como Suricata o Snort.
- Aplicar los mecanismos de seguridad necesarios para restringir el acceso a las diferentes herramientas Web del sistema.
- Habilitar un servicio firewall en cada una de las máquinas del sistema para restringir el tráfico a nivel interno, y bloquear cualquier tipo de conexión desde máquinas externas no permitidas.
- Conectar el sistema con un cortafuegos para que, tras la detección de tráfico malicioso, se puedan aplicar políticas de restricción de tráfico en tiempo real, aportando un sistema de protección automático. Incluso, se podría plantear que este firewall aplicara las reglas a partir de las listas de reputación generadas por Zeek.
- Hacer uso de las capacidades predictivas del stack Elastic para, mediante el modelado de tráfico a partir de técnicas de machine learning, poder detectar anomalías en el tráfico de red monitorizado, que no pueda ser detectado por Zeek.
- Incorporar al sistema un sistema de detección de intrusos basado en equipos (HIDS), e integrarlo también en Elastic para la detección de actividades sospechosas sobre los equipos de una red o personal de una organización.

6. Glosario

- **LAN:** Una red de área local o LAN (por las siglas en inglés de Local Area Network) es una red de computadoras que abarca un área reducida a una casa, un departamento o un edificio.
- **Cloud:** La computación en la nube (del inglés cloud computing), conocida también como servicios en la nube, informática en la nube, nube de cómputo, nube de conceptos o simplemente «la nube», es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.
- **Waterfall:** En Ingeniería de software el desarrollo en cascada (waterfall), también llamado secuencial o ciclo de vida de un programa (denominado así por la posición de las fases en el desarrollo de esta, que parecen caer en cascada “por gravedad” hacia las siguientes fases), es el enfoque metodológico que ordena rigurosamente las etapas del proceso para el desarrollo de software, de tal forma que el inicio de cada etapa debe esperar a la finalización de la etapa anterior.
- **IDS:** Un sistema de detección de intrusiones (o IDS de sus siglas en inglés Intrusion Detection System) es un programa de detección de accesos no autorizados a un computador o a una red.

El IDS suele tener sensores virtuales (por ejemplo, un sniffer de red) con los que el núcleo del IDS puede obtener datos externos (generalmente sobre el tráfico de red). El IDS detecta, gracias a dichos sensores, las anomalías que pueden ser indicio de la presencia de ataques y falsas alarmas.

- **NIDS:** NIDS (Network Intrusion Detection System), Sistema de detección de intrusos en una Red. Busca detectar anomalías que inicien un riesgo potencial, tales como ataques de denegación de servicio, escaneadores de puertos o intentos de entrar en un ordenador, analizando el tráfico en la red en tiempo real. Para ello, analiza todos los paquetes, buscando en ellos patrones sospechosos. Los NIDS no sólo vigilan el tráfico entrante, sino también el saliente o el tráfico local, ya que algunos ataques podrían ser iniciados desde el propio sistema protegido. A pesar de la vigilancia, su influencia en el tráfico es casi nula.
- **HIDS:** HIDS, Sistema de detección de intrusos en un Host. Busca detectar anomalías que indican un riesgo potencial, revisando las actividades en la máquina (host). Puede tomar medidas protectoras.
- **IPS:** Un sistema de prevención de intrusos (o por sus siglas en inglés IPS) es un software que ejerce el control de acceso en una red informática para proteger a los sistemas computacionales de ataques y abusos. La tecnología de prevención de intrusos es considerada por

algunos como una extensión de los sistemas de detección de intrusos (IDS), pero en realidad es otro tipo de control de acceso, más cercano a las tecnologías cortafuegos.

- **SIEM:** Un sistema de gestión de información y eventos de seguridad (en inglés, security information and event management, SIEM) es un sistema que centraliza el almacenamiento y la interpretación de los datos relevante de seguridad. De esta forma, permite un análisis de la situación en múltiples ubicaciones desde un punto de vista unificado que facilita la detección de tendencias y patrones no habituales.
- **Open Source:** El código abierto (*open source* en inglés) es un modelo de desarrollo de software basado en la colaboración abierta. Se enfoca más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.
- **On premise:** El término on-premise o en local se refiere al tipo de instalación de una solución de software. Esta instalación se lleva a cabo dentro del servidor y la infraestructura (TIC) de la empresa. Es el modelo tradicional de aplicaciones empresariales.
- **Hostname:** Un nombre de equipo es un nombre único y relativamente informal que se le da a un dispositivo conectado a una red informática. Puede ser un ordenador, un servidor de ficheros, un dispositivo de almacenamiento por red, una máquina de fax, impresora, etc.
- **IP:** El Protocolo de Internet (en inglés Internet protocol o IP) es un protocolo de comunicación de datos digitales clasificado funcionalmente en la capa de red según el modelo internacional OSI.
- **Threat Intelligence:** Threat Intelligence is an emerging technology discipline that helps organizations aggregate, correlate, and analyze threat data from multiple sources in real time to support defensive actions.
- **MD5:** En criptografía, MD5 (abreviatura de Message-Digest Algorithm 5, Algoritmo de Resumen del Mensaje 5) es un algoritmo de reducción criptográfico de 128 bits ampliamente usado. Uno de sus usos es el de comprobar que algún archivo no haya sido modificado.
- **DNS:** El sistema de nombres de dominio (Domain Name System o DNS, por sus siglas en inglés) es un sistema de nomenclatura jerárquico descentralizado para dispositivos conectados a redes IP como Internet o una red privada. Este sistema asocia información variada con nombre de dominio asignado a cada uno de los participantes. Su función más importante es "traducir" nombres inteligibles para las personas en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.

- **Cron:** En el sistema operativo Unix, cron es un administrador regular de procesos en segundo plano (demonio) que ejecuta procesos o guiones a intervalos regulares (por ejemplo, cada minuto, día, semana o mes). Los procesos que deben ejecutarse y la hora en la que deben hacerlo se especifican en el fichero crontab.
- **ISO:** Una imagen ISO es un archivo informático donde se almacena una copia o imagen exacta de un sistema de archivos. Se rige por el estándar ISO 9660, que le da nombre. Algunos de los usos más comunes incluyen la distribución de sistemas operativos, por ejemplo: GNU/Linux, FISCH, BSD y Live CD.
- **Ansible:** Ansible es una plataforma de software libre para configurar y administrar ordenadores. Combina instalación multi-nodo (es decir: permite desplegar configuraciones de servidores y servicios por lotes), ejecuciones de tareas ad hoc y administración de configuraciones. Adicionalmente, Ansible es categorizado como una herramienta de orquestación.
- **GitHub:** GitHub es una forja (plataforma de desarrollo colaborativo) para alojar proyectos utilizando el sistema de control de versiones Git. Se utiliza principalmente para la creación de código fuente de programas de ordenador.
- **HTTP:** El Protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP) es el protocolo de comunicación que permite las transferencias de información en la World Wide Web.
- **HTTPS:** El Protocolo seguro de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol Secure o HTTPS) es un protocolo de aplicación basado en el protocolo HTTP, destinado a la transferencia segura de datos de hipertexto, es decir, es la versión segura de HTTP.
- **CSV:** Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo para representar datos en forma de tabla, en las que las columnas se separan por comas (o punto y coma en donde la coma es el separador decimal como en Chile, Perú, Argentina, España, Brasil, entre otros) y las filas por saltos de línea.
- **URL:** Un localizador de recursos uniforme (más conocido por las siglas URL, del inglés Uniform Resource Locator) es un identificador de recursos uniforme (Uniform Resource Identifier, URI) cuyos recursos referidos pueden cambiar, esto es, la dirección puede apuntar a recursos variables en el tiempo. Están formados por una secuencia de caracteres de acuerdo a un formato modélico y estándar que designa recursos en una red como, por ejemplo, Internet.
- **Hash:** Un hash es el resultado de una función hash, la cual es una operación criptográfica que genera identificadores únicos e irrepetibles a partir de una información dada.

7. Referencias

- Instalar paquete security:zeek / zeek
Fecha consulta: 22/10/2019
http://download.opensuse.org/repositories/security:/zeek/xUbuntu_18.04/
- If you Bro you should ElasticSearch ! [Part 1] | etz69
Fecha consulta: 22/10/2019
<http://etz69.blogspot.com/2015/03/if-you-bro-you-should-elasticsearch.html>
- Intelligence Framework — Zeek User Manual v3.0.0
Fecha consulta: 29/9/2019
<https://docs.zeek.org/en/stable/frameworks/intel.html>
- Installing — Zeek User Manual v3.0.0
Fecha consulta: 13/10/2019
<https://docs.zeek.org/en/stable/install/install.html>
- Quick Start Guide — Zeek User Manual v3.0.0
Fecha consulta: 12/11/2019
<https://docs.zeek.org/en/stable/quickstart/index.html#zeek-scripts>
- Example for Bro file extraction with domain name in the filename if the file was grabbed over HTTP
Fecha consulta: 12/11/2019
<https://gist.github.com/sethhall/8221692>
- Home · PaloAltoNetworks/minemeld Wiki
Fecha consulta: 01/10/2019
<https://github.com/PaloAltoNetworks/minemeld/wiki>
- Ansible playbook for installing MineMeld on Linux
Fecha consulta: 06/10/2019
<https://github.com/PaloAltoNetworks/minemeld-ansible>
- PANW Firewall Visualisations using Elastic Stack
Fecha consulta: 01/10/2019
<https://github.com/sm-biz/paloalto-elasticstack-viz>
- Network Security Monitoring on Raspberry Pi type devices
Fecha consulta: 15/10/2019
<https://github.com/TravisFSmith/SweetSecurity>
- How to configure static IP address on Ubuntu 18.04 Bionic Beaver Linux
Fecha consulta: 13/10/2019
<https://linuxconfig.org/how-to-configure-static-ip-address-on-ubuntu-18-04-bionic-beaver-linux>
- Install ELK On Ubuntu 18.04 Bionic Beaver Linux
Fecha consulta: 23/07/2019
<https://linuxconfig.org/install-elk-on-ubuntu-18-04-bionic-beaver-linux>

- MineMeld Live Community
Fecha consulta: 10/10/2019
<https://live.paloaltonetworks.com/t5/MineMeld/ct-p/MineMeld>
- Free threat intelligence feeds
Fecha consulta: 22/10/2019
<https://threatfeeds.io/>
- How to Install Bro IDS on Ubuntu 16.04
Fecha consulta: 23/07/2019
https://www.alibabacloud.com/blog/how-to-install-bro-ids-on-ubuntu-16-04_594935
- Integrating Bro IDS with the Elastic Stack
Fecha consulta: 30/07/2019
<https://www.elastic.co/es/blog/bro-ids-elastic-stack>
- GeolIP in the Elastic Stack
Fecha consulta: 12/11/2019
<https://www.elastic.co/es/blog/geoip-in-the-elastic-stack>
- Starting Elasticsearch
Fecha consulta: 17/11/2019
<https://www.elastic.co/guide/en/elasticsearch/reference/current/starting-elasticsearch.html>
- Logstash Configuration Examples
Fecha consulta: 30/07/2019
<https://www.elastic.co/guide/en/logstash/current/config-examples.html>
- Cyber crime: reported damage to the IC3 2018
Fecha consulta: 01/10/2019
<https://www.statista.com/statistics/267132/total-damage-caused-by-by-cyber-crime-in-the-us/>
- U.S. data breaches and exposed records 2018
Fecha consulta: 01/10/2019
<https://www.statista.com/statistics/273550/data-breaches-recorded-in-the-united-states-by-number-of-breaches-and-records-exposed/>
- Wikipedia
Fecha consulta: 12/12/2019
<https://es.wikipedia.org/>

8. Anexos

Anexo I. Inventario software

Sistema operativo

Las herramientas que conforman la solución serán desplegadas en sistemas operativos Linux. La distribución seleccionada es Ubuntu Server 18.04 LTS, en su versión de 64 bits, al tratarse de un sistema operativo liviano, con buen rendimiento, y con requerimientos hardware que se ajustan a las capacidades de la máquina host donde se virtualizarán todos los sistemas.

A continuación, información de la versión de la distribución Linux seleccionada:

- URL Descarga:
<http://old-releases.ubuntu.com/releases/18.04.2/>
- Fichero:
Ubuntu-18.04.2-server-amd64.iso
- Tamaño:
925.892.608 bytes
- Verificación integridad: Checksum SHA256

```
a2cb36dc010d98ad9253ea5ad5a07fd6b409e3412c48f1860536970b073c98f5
```

Fuente: <http://old-releases.ubuntu.com/releases/18.04.2/SHA256SUMS>

- Validación integridad en local:

```
[2019-10-22 12:02.40] /drives/d/SO_Images  
[xavi.DESKTOP-T85BEJU] > sha256sum ubuntu-18.04.2-server-amd64.iso  
a2cb36dc010d98ad9253ea5ad5a07fd6b409e3412c48f1860536970b073c98f5  ubuntu-18.04.2-server-amd64.iso
```

Figura 67. Validación integridad Imagen SO en local.

Software de virtualización

Las máquinas donde se instalarán los sistemas operativos Ubuntu 18.04 serán provisionados mediante el software de virtualización Oracle VM VirtualBox en su versión 6.0, que se instalará en una máquina host con sistema operativo Windows 10.

A continuación, se provee la información de la versión utilizada:

- URL Descarga:
<https://download.virtualbox.org/virtualbox/6.0.12/>
- Fichero:
VirtualBox-6.0.12-133076-Win.exe

- Tamaño:
170.458.632 bytes
- Verificación integridad: Checksum SHA256

```
2780be12c139858412e789ed01700ea9041522a7b5fad2a27923f30c850f0198
```

Fuente: <https://download.virtualbox.org/virtualbox/6.0.12/SHA256SUMS>

- Validación integridad en local:

```
[2019-10-22 12:02.54] /drives/d/S0_Images
[xavi.DESKTOP-T85BEJU] > sha256sum VirtualBox-6.0.12-133076-Win.exe
2780be12c139858412e789ed01700ea9041522a7b5fad2a27923f30c850f0198 VirtualBox-6.0.12-133076-Win.exe
```

Figura 68. Validación integridad software VM en local.

Zeek IDS

El software de detección de intrusos Zeek IDS que se utilizará en el sistema será la versión 3.0.0, y su instalación se realizará mediante el gestor de paquetes apt de la distribución Linux instalada.

- Repositorio:
http://download.opensuse.org/repositories/security:/zeek/xUbuntu_18.04/
- Nombre paquete:
zeek
- Tamaño:
29.252.635 bytes (comprimido)
- Verificación integridad: PGP

Mediante la utilidad `api-key` del gestor de paquetes `apt`, se verificará la integridad del paquete a la hora de realizar la instalación. Para ello, se instalará en `apt` la clave facilitada por Zeek.

http://download.opensuse.org/repositories/security:/zeek/xUbuntu_18.04/Release.key

Stack Elastic

La instalación del stack Elastic se realizará mediante el gestor de paquete apt. La versión de los componentes que se utilizarán es la 7.4.0.

- Repositorio:
<https://artifacts.elastic.co/packages/7.x/apt>
- Nombre paquetes:
kibana elasticsearch logstash
- Tamaño:
 - *kibana: 259.102.054 bytes (comprimido)*
 - *elasticsearch: 291.729.210 bytes (comprimido)*
 - *logstash: 175.075.400 bytes (comprimido)*

- Verificación integridad: PGP

Mediante la utilidad `api-key` del gestor de paquetes `apt`, se verificará la integridad del paquete a la hora de realizar la instalación. Para ello, se instalará en `apt` la clave facilitada por Elastic.

<https://artifacts.elastic.co/GPG-KEY-elasticsearch>

Minemeld

El gestor de listas de reputación que se utilizará en el sistema es Minemeld, de la compañía Palo Alto Networks. La versión que se desplegará será la última disponible en el repositorio de código git oficial en el momento de instalación del sistema, que para este trabajo corresponde a la versión 0.9.64.

- Repositorio:
<https://github.com/PaloAltoNetworks/minemeld-ansible.git>

El motivo por el cual se ha optado por realizar una instalación de Minemeld a partir del código fuente es debido a la no disponibilidad de un paquete de instalación para el sistema operativo utilizado (Ubuntu Server 18.04).

Python

Para la implementación del sistema, se requerirá el desarrollo varios programas Python, lo cual requerirá de la instalación del entorno de ejecución en su versión 2.7 que se realizará mediante el gestor de paquetes `apt`.

- Repositorio:
<https://artifacts.elastic.co/packages/7.x/apt>
- Nombre paquetes:
kibana elasticsearch logstash
- Tamaño:
139.968 bytes (comprimido)
- Verificación integridad: Checksum SHA256

```
9be87952c519f94d24129911e03a24035ce47358cb528492f4a53b2e31efc6d4
```

- Validación integridad en local:

La validación de la integridad de los paquetes se realizará de forma automática por el gestor de paquetes `apt` en el momento de la instalación.

Anexo II. Setup máquinas virtuales

Instalación

Se utilizará como sistema operativo la distribución Ubuntu Server 18.04 LTS.

El fichero descargado será una imagen ISO que se cargará en el software de virtualización Oracle VM VirtualBox para proceder a su instalación.

La instalación del sistema operativo será estándar, sin requerir elementos software adicionales a los incluidos en la instalación básica.

A continuación, se describe el proceso de instalación de una de las máquinas virtuales. El proceso de instalación sería el mismo para las diferentes máquinas que componen el sistema, por lo que habiendo instalado una, el resto serán clones de la primera máquina creada, y la clonación la realizaremos utilizando las funcionalidades facilitadas por el software de virtualización.

Paso 1. Creación de la máquina virtual

En este primer paso, seleccionaremos la opción *Máquina > Nueva* del menú de opciones de VirtualBox, e introduciremos los datos básicos de la máquina en la ventana emergente que nos aparece.

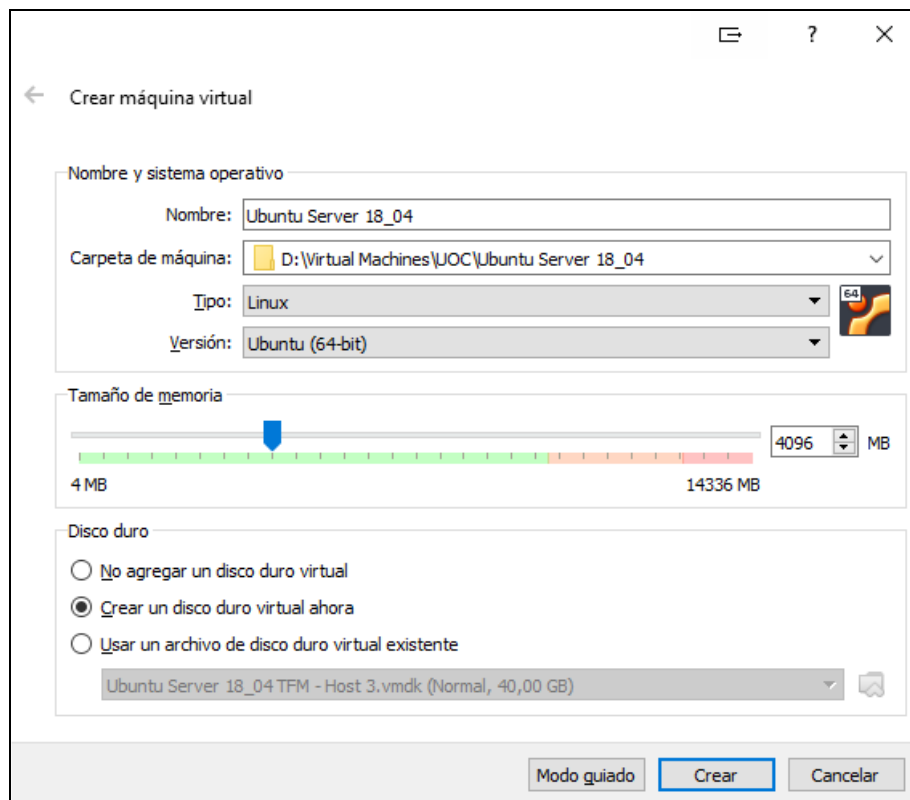


Figura 69. Pantalla creación máquina virtual

- Nombre y sistema operativo: Indicaremos nombre y ubicación de la máquina, y seleccionaremos Tipo Linux y Versión Ubuntu (64-bit)
- Tamaño de memoria: Seleccionaremos un tamaño de memoria acorde a las capacidades hardware de la máquina host. Para este trabajo, se seleccionará un tamaño de memoria de 4GB por máquina
- Disco duro: Se seleccionará la opción de crear un disco duro virtual ahora, para reservar ya el espacio que necesitará nuestra máquina en los siguientes pasos de creación de la máquina virtual.

A continuación, damos clic en el botón *Crear*, y nos aparece una nueva ventana donde se solicita información para la creación del disco duro virtual.

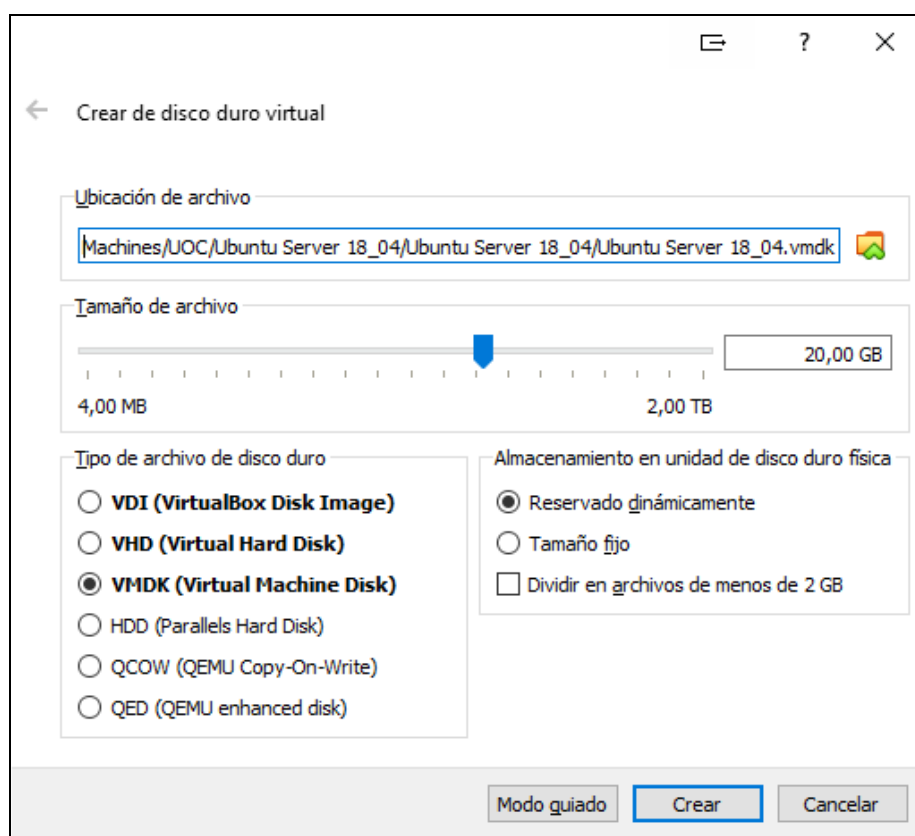


Figura 70. Pantalla creación disco duro virtual

- Tamaño de archivo: Seleccionaremos 20,00 GB de espacio.
- Tipo de archivo de disco duro: Seleccionaremos la opción VMDK (Virtual Machine Disk) por ser uno de los formatos de disco usados en el estándar Open Virtualization Format de cara a facilitar la migración de las imágenes a otros sistemas de virtualización si fuera necesario.
- Almacenamiento en unidad de disco física: Seleccionaremos la reserva dinámica de espacio en disco para evitar ocupar espacio innecesario de inicio en el disco duro de la máquina host.

A continuación, damos clic en el botón *Crear*, y automáticamente se procede a la creación de la máquina virtual con las características seleccionadas.

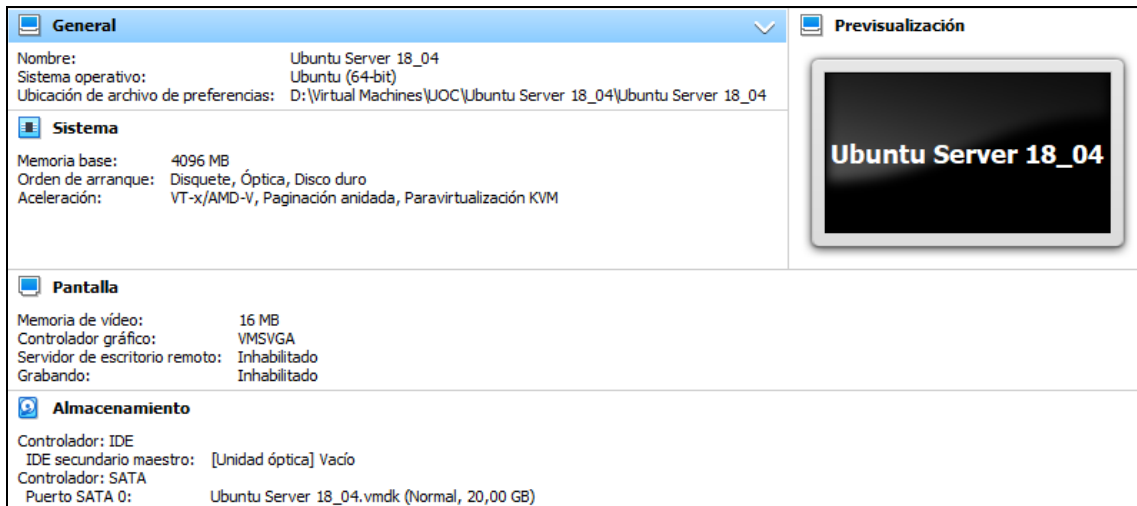


Figura 71. Pantalla información máquina virtual

Paso 2. Carga de la imagen de instalación del sistema operativo

El siguiente paso es cargar la imagen ISO del sistema operativo en la máquina virtual que se ha dado de alta, para que, a la hora de arrancar la imagen, se cargue el menú de instalación de Ubuntu.

Para ello, se debe acceder a la *Configuración* de la máquina virtual, apartado *Almacenamiento*, y allí seleccionar el Controlador: IDE Vacío existente. A continuación, se requiere seleccionar la opción “*Selecione archivo de disco duro virtual*” para asociarlo al Controlador: IDE. En este punto, seleccionaremos la imagen ISO descargada.

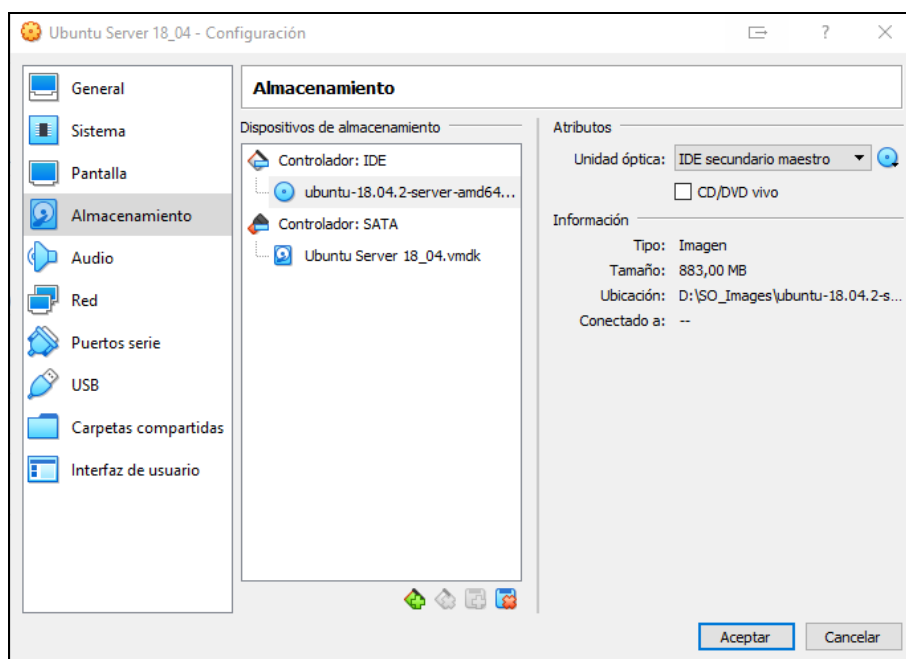


Figura 72. Carga imagen ISO de sistema operativo

Paso 3. Configuración de red

El siguiente paso consiste en configurar el adaptador de red de la máquina virtual. Para ello, accederemos dentro de la configuración de la máquina, al apartado *Red*.

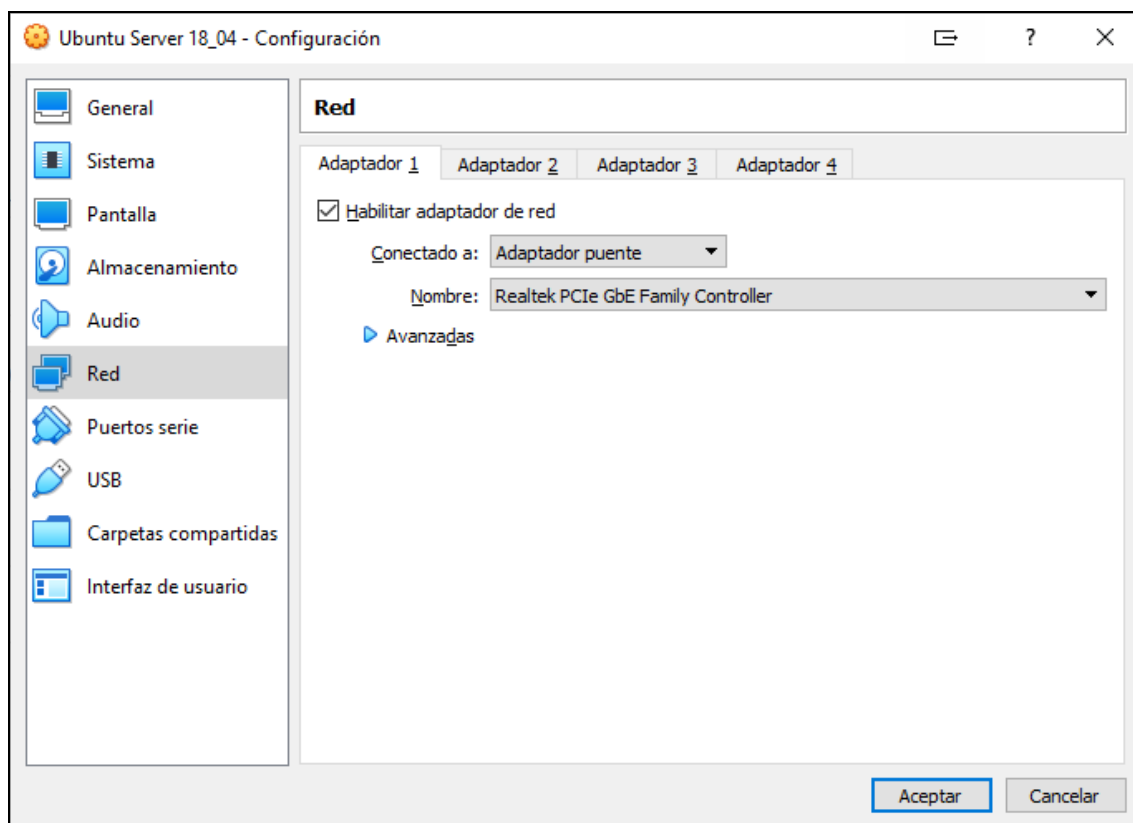


Figura 73. Pantalla configuración adaptador de red

- Conectado a: Seleccionaremos la opción de Adaptador puente, para que todas las máquinas estén conectadas a la misma LAN que la máquina Host y con acceso a Internet.

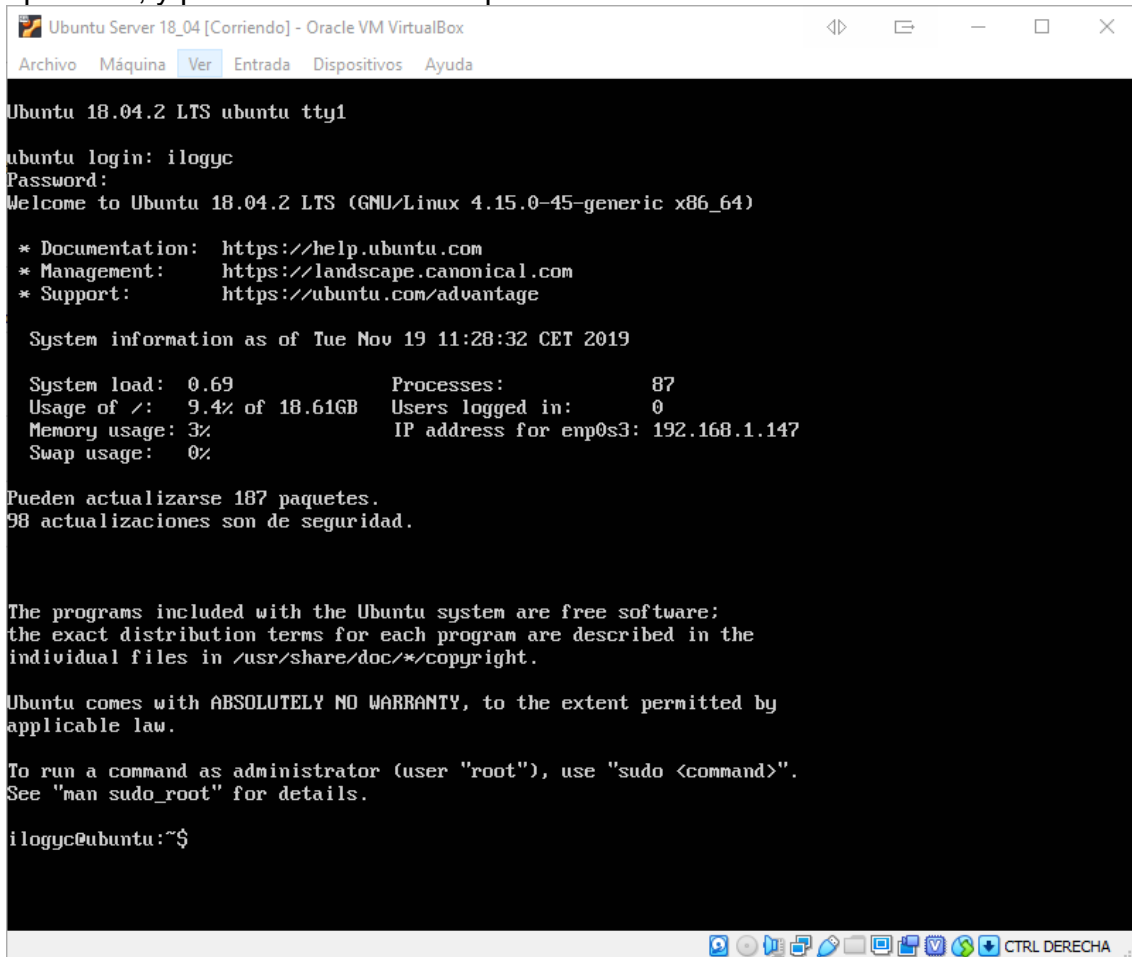
Paso 4. Instalación del sistema operativo

Una vez completada la configuración de la máquina virtual, procedemos a iniciar la máquina.

Al arrancar, aparecerá un primer menú de Ubuntu donde se seleccionará el idioma de la interfaz, y acto seguido, otro menú donde seleccionaremos la opción de "Instalar Ubuntu Server".

A continuación, se inicia el proceso de instalación del sistema operativo. Los detalles acerca de la configuración de ubicación, teclado, idioma, particionamiento, usuarios y otros aspectos del sistema operativo quedan fuera del alcance de este trabajo.

Una vez finalizada la instalación, se desmonta la imagen ISO del sistema operativo, y procedemos al arranque del mismo.



```
Ubuntu Server 18_04 [Corriendo] - Oracle VM VirtualBox
Archivo  Máquina  Ver  Entrada  Dispositivos  Ayuda

Ubuntu 18.04.2 LTS ubuntu tty1

ubuntu login: ilogyc
Password:
Welcome to Ubuntu 18.04.2 LTS (GNU/Linux 4.15.0-45-generic x86_64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Tue Nov 19 11:28:32 CET 2019

System load:  0.69          Processes:            87
Usage of /:   9.4% of 18.61GB Users logged in:      0
Memory usage: 3%          IP address for emp0s3: 192.168.1.147
Swap usage:  0%

Pueden actualizarse 187 paquetes.
98 actualizaciones son de seguridad.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ilogyc@ubuntu:~$
```

Figura 74. Consola Linux de la máquina virtual instalada

Paso 5. Actualización de paquetes

Por último, antes de iniciar la configuración de las diferentes máquinas virtuales, ejecutaremos una actualización de los paquetes instalados en el sistema.

Para ello, ejecutaremos los siguientes comandos:

```
$ sudo apt update
```

y a continuación

```
$ sudo apt upgrade -y
```

En este punto, el sistema operativo está correctamente instalado y actualizado.

A continuación, apagamos la máquina, y procedemos a la clonación del resto de máquinas a partir de la que acabamos de crear e instalar, y asignamos los nombres correspondientes a todas ellas para una mejor identificación.

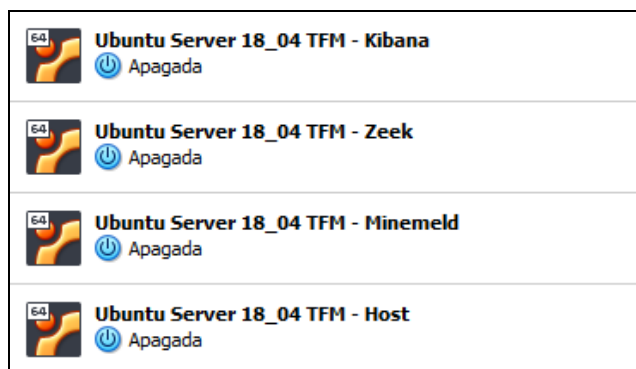


Figura 75. Lista de máquinas virtuales creadas

Y con esta última acción finaliza el proceso de instalación del sistema operativo, y procedemos a su configuración para cada una de las máquinas.

Configuración

La fase de configuración de los sistemas operativos se limitará por un lado a la configuración de la interfaz de red, en base a la topología de red definida en la fase de análisis, y en segundo lugar a asignar el nombre de máquina correcto.

La asignación de la dirección IP estática se llevará a cabo mediante netplan, herramienta disponible en el sistema operativo Ubuntu, que permite aplicar las configuraciones de red partiendo de un fichero de configuración.

El fichero de configuración de netplan está ubicado en la siguiente ruta:

```
$ /etc/netplan/01-netcfg.yaml
```

Las configuraciones que se aplicarán para cada máquina serán las siguientes:

Configuración de red máquina uoc-server-ids

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.1.160/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1, 8.8.8.8]
```

Configuración de red máquina uoc-server-elk

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
```

```
addresses: [192.168.1.170/24]
gateway4: 192.168.1.1
nameservers:
  addresses: [192.168.1.1,8.8.8.8]
```

Configuración de red máquina uoc-server-minemeld

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.1.180/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1,8.8.8.8]
```

Configuración de red máquina uoc-server-host

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [192.168.1.150/24]
      gateway4: 192.168.1.1
      nameservers:
        addresses: [192.168.1.1,8.8.8.8]
```

Una vez realizada la modificación sobre el fichero de configuración de cada máquina, sólo queda ejecutar netplan para que se apliquen los cambios oportunos. El comando a ejecutar sería el siguiente:

```
$ sudo netplan apply
```

Por último, quedaría pendiente asignar los nombres de host a las diferentes máquinas. Para ello, será necesario editar el fichero *resolv.conf* para indicar el nombre correcto asignado según la arquitectura final de la solución

La ubicación del fichero a editar es la siguiente:

```
$ /etc/hostname
```

Y los nombres que se aplicarán en cada máquina son:

- uoc-server-ids
- uoc-server-elk
- uoc-server-minemeld
- uoc-server-host

Validación

A continuación, se describen las diferentes validaciones realizadas sobre cada una de las máquinas.

uoc-server-ids

Validación de la configuración de red:

```
ilogyc@uoc-server-ids:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.160 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fela:af91 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:1a:af:91 txqueuelen 1000 (Ethernet)
    RX packets 30338 bytes 44900573 (44.9 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2256 bytes 176461 (176.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 76. Validación configuración red máquina uoc-server-ids

Validación del nombre de la máquina:

```
ilogyc@uoc-server-ids:~$ uname -a
Linux uoc-server-ids 4.15.0-65-generic #74-Ubuntu SMP Tue Sep 17 17:06:04 UTC 2019
ilogyc@uoc-server-ids:~$
```

Figura 77. Validación configuración nombre máquina uoc-server-ids

uoc-server-elk

Validación de la configuración de red:

```
ilogyc@uoc-server-elk:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.170 netmask 255.255.255.0 broadcast 192.168.1.255
    inet6 fe80::a00:27ff:fec0:332e prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:c0:33:2e txqueuelen 1000 (Ethernet)
    RX packets 5306 bytes 7325968 (7.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 1477 bytes 144432 (144.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Figura 78. Validación configuración red máquina uoc-server-elk

Validación del nombre de la máquina:

```
ilogyc@uoc-server-elk:~$ uname -a
Linux uoc-server-elk 4.15.0-65-generic #74-Ubuntu SMP Tue Sep 17 17:06:04 UTC 2019
ilogyc@uoc-server-elk:~$
```

Figura 79. Validación configuración nombre máquina uoc-server-elk

uoc-server-minemeld

Validación de la configuración de red:

```

ilogyc@uoc-server-minemeld:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.1.180 netmask 255.255.255.0 broadcast 192.168.1.255
  inet6 fe80::a00:27ff:fe31:f4e9 prefixlen 64 scopeid 0x20<link>
  ether 08:00:27:31:f4:e9 txqueuelen 1000 (Ethernet)
  RX packets 16804 bytes 8066896 (8.0 MB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 132683 bytes 280573353 (280.5 MB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figura 80. Validación configuración red máquina uoc-server-minemeld

Validación del nombre de la máquina:

```

ilogyc@uoc-server-minemeld:~$ uname -a
Linux uoc-server-minemeld 4.15.0-65-generic #74-Ubuntu SMP Tue Sep 17 17:06:04 UTC 2019
ilogyc@uoc-server-minemeld:~$

```

Figura 81. Validación configuración nombre máquina uoc-server-minemeld

uoc-server-host

Validación de la configuración de red:

```

ilogyc@uoc-server-host:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
  inet 192.168.1.150 netmask 255.255.255.0 broadcast 192.168.1.255
  inet6 fe80::a00:27ff:fe06:8d6a prefixlen 64 scopeid 0x20<link>
  ether 08:00:27:06:8d:6a txqueuelen 1000 (Ethernet)
  RX packets 5224 bytes 7275372 (7.2 MB)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 1886 bytes 153268 (153.2 KB)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

```

Figura 82. Validación configuración red máquina uoc-server-host

Validación del nombre de la máquina:

```

ilogyc@uoc-server-host:~$ uname -a
Linux uoc-server-host 4.15.0-65-generic #74-Ubuntu SMP Tue Sep 17 17:06:04 UTC 2019
ilogyc@uoc-server-host:~$

```

Figura 83. Validación configuración nombre máquina uoc-server-host

Anexo III. Scripts Python

A continuación, se incluye el código fuente de los cuatro scripts Python desarrollados para el sistema, encargados de recuperar los contenidos de los feeds de Minemeld que hemos creado, y volcarlos a fichero con el formato correcto para que posteriormente se puedan utilizar como diccionarios en el filtro Translate de Logstash para la ingesta en el motor de Elasticsearch.

Para cada script, se indicará la URL de Minemeld que se consulta, junto con el nombre del fichero de salida generado y ubicación.

Junto con los scripts, se incluye además un módulo de utilidades.

Script direcciones IP

- URL feed Minemeld: <https://192.168.1.180/feeds/ips-output>
- Fichero de salida: /opt/zeek/dictionary/ips.yml
- Nombre script: minemeld-parser-ip.py

```
import requests
import parserutil

feed_url = "https://192.168.1.180/feeds/ips-output"
temp_output = "temp/ips-output-temp"
dest_output = "/opt/zeek/dictionary/ips.yml"

parserutil.Log("-----")
parserutil.Log("Starting minemeld parser -> IPv4")
parserutil.Log("")
parserutil.Log("Configuration:")
parserutil.Log("    feed_url: " + feed_url)
parserutil.Log("    temp_output: " + temp_output)
parserutil.Log("    dest_output: " + dest_output)
parserutil.Log(" ")
parserutil.Log(" Loading...")

r = requests.get(feed_url, verify=False)

if r.status_code != 200:
    exit(-1)

feed = r.text

feed_iter = iter(feed.splitlines())

out = parserutil.open_file(temp_output)

count = 0

for line in feed_iter:
    item = line.split("-")
    print >>out, "\"" + item[0] + "\": malicious"
    count += 1
```

```

out.close()

parserutil.Log(" Finished! Items Loaded: " + str(count))

parserutil.copy_file(temp_output, dest_output)

```

Script dominios

- URL feed Minemeld: <https://192.168.1.180/feeds/domains-output>
- Fichero de salida: /opt/zeek/dictionary/domains.yml
- Nombre script: minemeld-parser-domain.py

```

import requests
import parserutil

feed_url = "https://192.168.1.180/feeds/domains-output"
temp_output = "temp/domains-output-temp"
dest_output = "/opt/zeek/dictionary/domains.yml"

parserutil.Log("-----")
parserutil.Log("Starting minemeld parser -> domain")
parserutil.Log("")
parserutil.Log("Configuration:")
parserutil.Log("    feed_url: " + feed_url)
parserutil.Log("    temp_output: " + temp_output)
parserutil.Log("    dest_output: " + dest_output)
parserutil.Log(" ")
parserutil.Log(" Loading...")

try:
    r = requests.get(feed_url, verify=False)
except requests.ConnectionError, e:
    parserutil.Log("ERROR HTTP connection error "+str(e.message)+" |
"+feed_url)
    exit(-1)

if r.status_code != 200:
    parserutil.Log("ERROR HTTP incorrect response code "+r.status_code+" |
"+feed_url)
    exit(-1)

feed = r.text

feed_iter = iter(feed.splitlines())

out = parserutil.open_file(temp_output)

count = 0

for line in feed_iter:
    item = line.split(",")
    print >>out, "\"" + item[0] + "\"": malicious"
    count += 1

out.close()

```

```
parserutil.Log(" Finished! Items Loaded: " + str(count))

parserutil.copy_file(temp_output, dest_output)
```

Script URL's

- URL feed Minemeld: <https://192.168.1.180/feeds/urls-output>
- Fichero de salida: /opt/zeek/dictionary/urls.yml
- Nombre script: minemeld-parser-url.py

```
import requests
import parserutil

feed_url = "https://192.168.1.180/feeds/urls-output"
temp_output = "temp/urls-output-temp"
dest_output = "/opt/zeek/dictionary/urls.yml"

parserutil.Log("-----")
parserutil.Log("Starting minemeld parser -> url")
parserutil.Log("")
parserutil.Log("Configuration:")
parserutil.Log("    feed_url: " + feed_url)
parserutil.Log("    temp_output: " + temp_output)
parserutil.Log("    dest_output: " + dest_output)
parserutil.Log(" ")
parserutil.Log(" Loading...")

r = requests.get(feed_url, verify=False)

if r.status_code != 200:
    exit(-1)

feed = r.text

feed_iter = iter(feed.splitlines())

out = parserutil.open_file(temp_output)

count = 0

for line in feed_iter:
    line = line.replace("https", "").replace("http", "").replace("://", "")
    print >>out, "\"" + line + "\": malicious"
    count += 1

out.close()

parserutil.Log(" Finished! Items Loaded: " + str(count))

parserutil.copy_file(temp_output, dest_output)
```

Script Ficheros

- URL feed Minemeld: <https://192.168.1.180/feeds/hash-output>

- Fichero de salida: /opt/zeek/dictionary/hashes.yml
- Nombre script: minemeld-parser-hash.py

```

import requests
import parserutil

feed_url = "https://192.168.1.180/feeds/hash-output"
temp_output = "temp/hashes-output-temp"
dest_output = "/opt/zeek/dictionary/hashes.yml"

parserutil.Log("-----")
parserutil.Log("Starting minemeld parser -> hash md5")
parserutil.Log("")
parserutil.Log("Configuration:")
parserutil.Log("    feed_url: " + feed_url)
parserutil.Log("    temp_output: " + temp_output)
parserutil.Log("    dest_output: " + dest_output)
parserutil.Log(" ")
parserutil.Log(" Loading...")

r = requests.get(feed_url, verify=False)

if r.status_code != 200:
    exit(-1)

feed = r.text

feed_iter = iter(feed.splitlines())

out = parserutil.open_file(temp_output)

count = 0

for line in feed_iter:
    print >>out, "\"" + line + "\": malicious"
    count += 1

out.close()

parserutil.Log(" Finished! Items Loaded: " + str(count))

parserutil.copy_file(temp_output, dest_output)

```

Módulo de utilidades

- Nombre script: parserutil.py

```

import os
import errno
import shutil
import time
import datetime

def open_file(filename):
    create_dir_if_not_exist(filename)
    out = open(filename, "w")

```



```

    return out

def copy_file(source_filename, dest_filename):
    create_dir_if_not_exist(dest_filename)

    try:
        shutil.copy2(source_filename, dest_filename)
    except OSError as ex:
        log("ERROR copying file " + ex.message)
        raise

    return True

def create_dir_if_not_exist(filename):
    if not os.path.exists(os.path.dirname(filename)):
        try:
            os.makedirs(os.path.dirname(filename))
        except OSError as ex:
            if ex.errno != errno.EEXIST:
                log("ERROR creating dir " + ex.message)
                raise

def log(text):
    st = datetime.datetime.fromtimestamp(time.time()).strftime('%Y-%m-%d
%H:%M:%S')

```

Anexo IV. Configuración Logstash

Fichero zeek.conf

```
input {
  file {
    path => "/opt/zeek/Logs/current/conn.Log"
    type => "conn"
  }
  file {
    path => "/opt/zeek/Logs/current/dns.Log"
    type => "dns"
  }
  file {
    path => "/opt/zeek/Logs/current/http.Log"
    type => "http"
  }
  file {
    path => "/opt/zeek/Logs/current/files.Log"
    type => "files"
  }
}

filter {
  if [path] == "/opt/zeek/Logs/current/conn.Log" {
    grok {
      patterns_dir => "/etc/logstash/patterns"
      match => {
        message => "%{connLog}"
      }
    }
    translate {
      field => "resp_h"
      add_field => { "malicious" => "yes" }
      dictionary_path => "/opt/zeek/dictionary/ips.yml"
    }
  }
  if [path] == "/opt/zeek/Logs/current/http.Log" {
    grok {
      patterns_dir => "/etc/logstash/patterns"
      match => {
        message => "%{httpLog}"
      }
    }
    mutate {
      add_field => { "url" => "%{server_name}%{uri}" }
    }
    translate {
      field => "url"
      add_field => { "malicious" => "yes" }
      dictionary_path => "/opt/zeek/dictionary/urls.yml"
    }
  }
  if [path] == "/opt/zeek/Logs/current/dns.Log" {
    grok {
      patterns_dir => "/etc/logstash/patterns"
      match => {
```

```

    message => "%{dnsLog}"
  }
}
translate {
  field => "query"
  add_field => { "malicious" => "yes" }
  dictionary_path => "/opt/zeek/dictionary/domains.yml"
}
}
if [path] == "/opt/zeek/logs/current/files.log" {
  grok {
    patterns_dir => "/etc/logstash/patterns"
    match => {
      message => "%{filesLog}"
    }
  }
  translate {
    field => "md5"
    add_field => { "malicious" => "yes" }
    dictionary_path => "/opt/zeek/dictionary/ hashes.yml"
  }
}

geoiip {
  source => "resp_h"
  target => "geoiip_resp"
}

geoiip {
  source => "orig_h"
  target => "geoiip_orig"
}
}

output {
  elasticsearch {
    hosts => "192.168.1.170"
    index => "zeek-all-logs"
  }

  if [type] == "conn" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-conn-logs"
    }
  }
  if [type] == "http" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-http-logs"
    }
  }
  if [type] == "dns" {
    elasticsearch {
      hosts => "192.168.1.170"
      index => "zeek-dns-logs"
    }
  }
}
}

```

```

if [type] == "files" {
  elasticsearch {
    hosts => "192.168.1.170"
    index => "zeek-files-logs"
  }
}
}

```

Fichero zeek-patterns

```

connLog (?<ts>\d+\.\d+)\t(?<uid>(\w+/-
))\t(?<orig_h>[^\t]+\t(?<orig_p>[^\t]+\t(?<resp_h>[^\t]+\t(?<resp_p>[^\t]+
)\t(?<proto>[^\t]+\t(?<service>[^\t]+\t(?<duration>[^\t]+\t(?<orig_bytes>[
^\t]+\t(?<resp_bytes>[^\t]+\t(?<conn_state>[^\t]+\t(?<local_orig>[^\t]+\t
(?<local_resp>[^\t]+\t(?<missed_bytes>[^\t]+\t(?<history>[^\t]+\t(?<orig_p
kts>[^\t]+\t(?<orig_ip_bytes>[^\t]+\t(?<resp_pkts>[^\t]+\t(?<resp_ip_bytes
>[^\t]+\t(?<tunnel_parents>.*))

```

```

dnsLog (?<ts>\d+\.\d+)\t(?<uid>(\w+/-
))\t(?<orig_h>[^\t]+\t(?<orig_p>[^\t]+\t(?<resp_h>[^\t]+\t(?<resp_p>[^\t]+
)\t(?<proto>[^\t]+\t(?<trans_id>[^\t]+\t(?<rtt>[^\t]+\t(?<query>[^\t]+\t(
?<qclass>[^\t]+\t(?<qclass_name>[^\t]+\t(?<qtype>[^\t]+\t(?<qtype_name>[^\
t]+\t(?<rcode>[^\t]+\t(?<rcode_name>[^\t]+\t(?<AA>[^\t]+\t(?<TC>[^\t]+\t
(?<RD>[^\t]+\t(?<RA>[^\t]+\t(?<Z>[^\t]+\t(?<answers>[^\t]+\t(?<TTLs>[^\t]
+)\t(?<rejected>.*))

```

```

httpLog (?<ts>\d+\.\d+)\t(?<uid>(\w+/-
))\t(?<orig_h>[^\t]+\t(?<orig_p>[^\t]+\t(?<resp_h>[^\t]+\t(?<resp_p>[^\t]+
)\t(?<trans_depth>[^\t]+\t(?<method>[^\t]+\t(?<server_name>[^\t]+\t(?<uri>
[^\t]+\t(?<referrer>[^\t]+\t(?<version>[^\t]+\t(?<user_agent>[^\t]+\t(?<r
equest_body_len>[^\t]+\t(?<response_body_len>[^\t]+\t(?<status_code>[^\t]+
)\t(?<status_msg>[^\t]+\t(?<info_code>[^\t]+\t(?<info_msg>[^\t]+\t(?<tags>[
^\t]+\t(?<username>[^\t]+\t(?<password>[^\t]+\t(?<proxied>[^\t]+\t(?<orig
_fuids>[^\t]+\t(?<orig_filenames>[^\t]+\t(?<orig_mime_types>[^\t]+\t(?<res
p_fuids>[^\t]+\t(?<resp_filenames>[^\t]+\t(?<resp_mime_types>.*))

```

```

filesLog (?<ts>\d+\.\d+)\t(?<fuid>(\w+/-
))\t(?<orig_h>[^\t]+\t(?<resp_h>[^\t]+\t(?<conn_uid>[^\t]+\t(?<source>[^\t
]+\t(?<depth>[^\t]+\t(?<analyzers>[^\t]+\t(?<mime_type>[^\t]+\t(?<filenam
e>[^\t]+\t(?<duration>[^\t]+\t(?<local_orig>[^\t]+\t(?<is_orig>[^\t]+\t(?
<seen_bytes>[^\t]+\t(?<total_bytes>[^\t]+\t(?<missing_bytes>[^\t]+\t(?<ove
rflow_bytes>[^\t]+\t(?<timed_out>[^\t]+\t(?<parent_fuid>[^\t]+\t(?<md5>[^\
t]+\t(?<sha1>[^\t]+\t(?<sha256>[^\t]+\t(?<extracted>[^\t]+\t(?<extracted_
cutoff>[^\t]+\t(?<extracted_size>.*)\t(?<hostname>.*)\t(?<uri>.*)\t(?<extens
ion>.*))

```

Anexo V. Template Elasticsearch

Configuración del template

```
PUT _template/zeek-template
{
  "index_patterns" : "zeek-*",
  "version" : 100000,
  "settings" : {
    "index.refresh_interval" : "5s",
    "number_of_shards": 1
  },
  "mappings" : {
    "dynamic_templates" : [ {
      "message_field" : {
        "path_match" : "message",
        "match_mapping_type" : "string",
        "mapping" : {
          "type" : "text",
          "norms" : false
        }
      }
    }
  ], {
    "string_fields" : {
      "match" : "*",
      "match_mapping_type" : "string",
      "mapping" : {
        "type" : "text", "norms" : false,
        "fields" : {
          "keyword" : { "type": "keyword", "ignore_above": 256 }
        }
      }
    }
  } ],
  "properties" : {
    "@timestamp": { "type": "date"},
    "@version": { "type": "keyword"},
    "geoip" : {
      "dynamic": true,
      "properties" : {
        "ip": { "type": "ip" },
        "location" : { "type" : "geo_point" },
        "latitude" : { "type" : "half_float" },
        "longitude" : { "type" : "half_float" }
      }
    },
    "orig_bytes" : { "type": "Long"},
    "orig_ip_bytes" : { "type": "Long"},
    "orig_pkts" : { "type": "Long"},
    "resp_bytes" : { "type": "Long"},
    "resp_ip_bytes" : { "type": "Long"},
    "resp_pkts" : { "type": "Long"},
    "duration" : { "type": "Long"},
    "total_bytes" : { "type": "Long"}
  }
}
```

Anexo VI. Script Zeek

Script Zeek. Ampliar información files.log

```
redef record Files::Info += {
  hostname: string &log &optional;
  uri: string &log &optional;
  extension: string &log &optional;
};

event file_over_new_connection(f: fa_file, c: connection, is_orig: bool)
{
  if (c?$http)
  {
    f$info$hostname = (c$http?$host) ? c$http$host : "-";

    if (c$http?$uri)
    {
      local uri_split = split_string(c$http$uri, /\//);

      f$info$hostname = c$http$host;

      local uri = "";
      local total = 0;

      # iterate over the vector to get total of items
      for (item in uri_split) {
        total += 1;
      }

      local iteration = 0;

      while( iteration < total-1) {
        if (iteration > 0) {
          uri += "/" + uri_split[iteration];
        }

        iteration += 1;
      }

      uri += "/";

      f$info$uri = uri;
      f$info$filename = uri_split[total-1];

      local filename_split = split_string(f$info$filename, /\../);
      f$info$extension = filename_split[1];
    }
  }
}
```

Anexo VII. Script y ficheros Pruebas

Script generador tráfico. generate_traffic.sh

```
#!/bin/bash

while :
do

for i in {1..30}
do
    url=$(shuf -n 1 good_traffic.txt)
    curl -s $url --max-time 2 > /dev/null
    sleep 0.25
done

url=$(shuf -n 1 bad_traffic.txt)
curl -s $url --max-time 2 > /dev/null

for i in {1..2}
do
    url=$(shuf -n 1 good_files.txt)
    curl -s $url --max-time 2 > /dev/null
    sleep 0.25
done

url=$(shuf -n 1 bad_files.txt)
curl -s $url --max-time 2 > /dev/null

# systemd-resolve --flush-caches > /dev/null

done
```

Lista de URLs lícitas. good_traffic.txt

```
http://www.youtube.com
http://www.facebook.com
http://www.baidu.com
http://www.yahoo.com
http://www.amazon.com
http://www.wikipedia.org
http://www.qq.com
http://www.google.co.in
http://www.twitter.com
http://www.live.com
http://www.taobao.com
http://www.bing.com
http://www.instagram.com
http://www.weibo.com
http://www.sina.com.cn
http://www.linkedin.com
http://www.yahoo.co.jp
http://www.msn.com
http://www.vk.com
http://www.google.de
http://www.yandex.ru
http://www.hao123.com
http://www.google.co.uk
http://www.reddit.com
http://www.ebay.com
http://www.google.fr
http://www.t.co
http://www.tmall.com
http://www.google.com.br
http://www.360.cn
```

<http://www.sohu.com>
<http://www.amazon.co.jp>
<http://www.pinterest.com>
<http://www.netflix.com>
<http://www.google.it>
<http://www.google.ru>
<http://www.microsoft.com>
<http://www.google.es>
<http://www.wordpress.com>
<http://www.gmw.cn>
<http://www.tumblr.com>
<http://www.paypal.com>
<http://www.blogspot.com>
<http://www.imgur.com>
<http://www.stackoverflow.com>
<http://www.aliexpress.com>
<http://www.naver.com>
<http://www.ok.ru>
<http://www.apple.com>
<http://www.github.com>
<http://www.chinadaily.com.cn>
<http://www.imdb.com>
<http://www.google.co.kr>
<http://www.fc2.com>
<http://www.jd.com>
<http://www.blogger.com>
<http://www.163.com>
<http://www.google.ca>
<http://www.whatsapp.com>
<http://www.amazon.in>
<http://www.office.com>
<http://www.tianya.cn>
<http://www.google.co.id>
<http://www.youku.com>
<http://www.rakuten.co.jp>
<http://www.craigslist.org>
<http://www.amazon.de>
<http://www.nicovideo.jp>
<http://www.google.pl>
<http://www.soso.com>
<http://www.bilibili.com>
<http://www.dropbox.com>
<http://www.xinhuanet.com>
<http://www.outbrain.com>
<http://www.pixnet.net>
<http://www.alibaba.com>
<http://www.alipay.com>
<http://www.microsoftonline.com>
<http://www.booking.com>
<http://www.googleusercontent.com>
<http://www.google.com.au>
<http://www.popads.net>
<http://www.cntv.cn>
<http://www.zhihu.com>
<http://www.amazon.co.uk>
<http://www.diplly.com>
<http://www.coccoc.com>
<http://www.cnn.com>
<http://www.bbc.co.uk>
<http://www.twitch.tv>
<http://www.wikia.com>
<http://www.google.co.th>
<http://www.go.com>
<http://www.google.com.ph>
<http://www.doubleclick.net>
<http://www.onet.pl>
<http://www.googleadservices.com>
<http://www.accuweather.com>
<http://www.googleweblight.com>
<http://www.answers.yahoo.com>
<http://eonline.com>
<http://about.me>
<http://wiley.com>
<http://house.gov>
<http://sina.com.cn>
<http://state.gov>
<http://newyorker.com>
<http://stores.jp>
<http://icann.org>
<http://people.com>
<http://apachefriends.org>
<http://bluehost.com>
<http://thedailybeast.com>
<http://inc.com>
<http://xing.com>

<http://cbslocal.com>
<http://scientificamerican.com>
<http://geocities.jp>
<http://ca.gov>
<http://tabelog.com>
<http://answers.com>
<http://answers.yahoo.com>
<http://stuff.co.nz>
<http://sputniknews.com>
<http://thenextweb.com>
<http://nyu.edu>
<http://narod.ru>
<http://uspto.gov>
<http://liveinternet.ru>
<http://discovery.com>
<http://cocolog-nifty.com>
<http://imgur.com>
<http://thefreedictionary.com>
<http://slate.com>
<http://digitaltrends.com>
<http://ustream.tv>
<http://entrepreneur.com>
<http://scoop.it>
<http://pastebin.com>
<http://lifehacker.com>
<http://go.co>
<http://arstechnica.com>
<http://calameo.com>
<http://groups.yahoo.com>
<http://vkontakte.ru>
<http://bp1.blogger.com>
<http://symantec.com>
<http://skype.com>
<http://kinja.com>
<http://flickr.com>
<http://redhat.com>
<http://calendar.google.com>
<http://allrecipes.com>
<http://mail.yahoo.com>
<http://www.nhs.uk>
<http://daum.net>
<http://livescience.com>
<http://wisc.edu>
<http://tmz.com>
<http://seattletimes.com>
<http://gesetze-im-internet.de>
<http://ucoz.ru>
<http://espn.go.com>
<http://wattpad.com>
<http://archives.gov>
<http://billboard.com>
<http://megaupload.com>
<http://disney.go.com>
<http://ameblo.jp>
<http://dictionary.com>
<http://arxiv.org>
<http://ieee.org>
<http://www.canalblog.com>
<http://usgs.gov>
<http://etsy.com>
<http://one.com>
<http://usa.gov>
<http://storify.com>
<http://plos.org>
<http://amazon.ca>
<http://politico.com>
<http://newscientist.com>
<http://foursquare.com>
<http://teamviewer.com>
<http://quantcast.com>
<http://evernote.com>
<http://feedproxy.google.com>
<http://nicovideo.jp>
<http://seesaa.net>
<http://fifa.com>
<http://springer.com>
<http://xfinity.com>
<http://excite.co.jp>
<http://indiegogo.com>
<http://khanacademy.org>
<http://nikkei.com>
<http://ap.org>
<http://prezi.com>
<http://web.fc2.com>
<http://prnewswire.com>

Lista de URLs maliciosas. bad_traffic.txt

<http://104.168.61.47/sparc>
<http://www.thermadorapplianceservice.com/rtqh/ZyzXzTiD/>
<http://89.42.133.29/Netflix.arm6>
<http://5.206.227.65/arm5.tsunami>
<http://104.168.190.82/8arm78>
<http://91.211.153.251/rrtn/WizyCrypt.exe>
<http://gigantic-friends.com/11>
<http://185.163.47.144/Lucky/dspy.exe>
<http://indoroyalseafood.com/br/ijsk.exe>
<http://205.185.114.16/bins/a.arm>
<http://193.70.36.193/i686>
<http://185.164.72.176/razor/r4z0r.arm6>
<http://185.112.249.122/packets.arm4>
<http://103.1.250.236:8080/4appverif.chm>
<http://pro-rec.event-pro.com.ua/wp-admin/8a6g28460/>
<http://adhesive.bengalgroup.com/bivgg/5o7bg/>
<http://217.73.62.206/hqLw/win32s.exe>
<http://23.254.224.213/earyzq>
<http://quangcaogiaodich.com/wp-content/upgrade/jzkowiu4uobwywynyj7/>
<http://205.185.114.16/bins/shibui.x86>
<http://skilmu.com/9ar12/>
<http://185.101.105.115/mipsel>
<http://tellselltheme.com/cgi-bin/a/>
<http://35.181.60.96/7/9704116.jpg>
<http://35.181.60.96/8/590741.jpg>
http://159.203.92.58/dark_bins/dark.sh4
<http://cdn.discordapp.com/attachments/630911118843576320/643723679376605184/bbuil1d.exe>
<http://www.alberolandia.it/wp-admin/yHRE0qfAg/>
<http://157.52.211.142/priv8/putty.exe>
<http://46.166.187.151/bins/a.x86>
<http://46.166.187.151/bins/shibui.spc>
<http://194.15.36.41/bins/orphic.mpsl>
<http://www.portoghesefilippo.it/wp-content/themes/sketch/rss.exe>
<http://178.33.83.75/armv5L>
<http://104.168.190.82/8mpsl8>
<http://lifesaverbottledirect.com/wp-includes/ID3/908rgg/4rx0yqfay/2c.jpg>
<http://185.112.249.39/bins/akemi.arm7>
<http://185.112.249.39/bins/akemi.x86>
<http://201.153.28.86:17562/.i>
<http://104.168.133.5/bins/hoho.spc>
<http://192.99.167.213/a-r.m-7.SNOOPY>
<http://esportcenter.pl/br/kv.exe>
<http://104.168.61.47/m68k>
<http://151.80.197.109/eBxUk/ddtp>
<http://selfhelpstartshere.com/wp-admin/1>
<http://185.212.130.42/updater.exe>
<http://sleuth.energy/5c0.msi>
<http://178.33.181.19/snype.ppc>
<http://xvcvxcxf.ru/nsdfvjhgk.exe>
<http://rui-chan.net/.well-known/pki-validation/payments/rhhr3zvK0/2c.jpg>
http://179.43.149.12/updating_32zs6f54f6rg1543tg32/ku.i686
<http://178.156.202.100/armv6L>
<http://www.gelisimcizgisi.com/articles/wxpg6fk/>
<http://www.kapdabazaar.com/installo/n8u18/>
<http://192.227.176.116/bins/yakuza.mpsl>
<http://wwwhelper.com/comm/moneymakers/css/xzm96/>
<http://ring1.ug/exe/starticon0.exe>
<http://interset-idf.org/prive/payreceipt.exe>
<http://192.227.176.122/bins/yakuza.arm5>
http://gray-yame-8073.holy.jp/nice/BBNN_Protected.exe
<http://185.112.249.62/bins/Wolfz.arm7>
<http://pingup.ir/wp-content/uploads/2019/11/home/aaaa.png>
<http://62.101.62.66:47163/.i>
<http://199.195.254.59/dope/fd.i686>
<http://13.54.13.60/C/25960.jpg>
<http://104.168.190.82/8sh48>
<http://pvg31z3n20.ooshuspeisekarte.com>
<http://kruisskyf.marylandevictionsonline.com>
<http://cdnrfv.com>
<http://gatec.iexemasktang8.net>
<http://turnbyov.uk>
<http://d6-6oy.kaeyucoolenjoy.net>
<http://ayomide1.ddns.net>
<http://mdb.phetaearlbox.net>
<http://satel.saejiactievandedag.net>
<http://zx.impressionoa.xyz>
<http://kh.eiyoojuegosparawindows.net>
<http://hidroplano.sukol.com>
<http://a20z5m.upahrcreadordeingresos.com>
<http://successionfinst.co.nz>
<http://lasopo3ik.Lowpokjsioep.tk>

<http://componisteresaddle.unapologeticnails.com>
<http://toyra.ahdoonetfisher.net>
<http://service-providers-of-communications2-v72.net.pl>
<http://sacto.ahchivirtual.net>
<http://dev.radiusfinancial.net>
<http://casazip.com>
<http://njrzae5r3ns.ohsohmoviez.com>
<http://www.paypalobjects.com>
<http://werlwas.antiagingbeauty.com>
<http://karakujianfis.hotelportauxbasque.com>
<http://wsucmwkccgaiwkuq.org>
<http://rvw2j26avk.ibohpjavme.com>
<http://revivelondon.org>
<http://dzw9-lcm4.shepelsmar.com>
<http://suihkuttavathervorschaut.sindytennesrealtor.com>
<http://dispunishable.pussypump.be>
<http://panthersoccerclub.org>
<http://jwqg.pheikmajide.com>
<http://sufficinglyskrofulose.detoxcenterssouthcarolina.com>
<http://kj.bestfitnessactivitytracker.com>
<http://curso.ohghaiqpic.com>
<http://po564kl.io23i4ikjsah.xyz>
<http://trade.ahkohzjgwy.com>
<http://wd7rw18lww.ahshoescream.net>
<http://traff.eefoupollstream.net>
<http://footed.stevendaluz.com>
<http://zu.ohsohmoviez.com>
<http://treck.glaclierguns.com>
<http://scienceweek.scieron.com>
<http://torrent.gotgeeks.com>
<http://osidoribyleerveraenderliche.detoxcenternorthcarolina.com>
<http://nk15y6io76.zinoiosijek031.net>
<http://imfar.iedoobuzzlike.com>
<http://xz3g4i25su.ibohpjavme.com>
<http://comun.saeghsamaanonline.net>
<http://bestamour4u.net>
<http://mykon.eitiee-marketpost.net>
<http://xablopefgr.ru>
<http://yib3oa.useshbestofvids.com>
<http://wanpisuamyLomyces.stixjeweler.com>
<http://blog.thefurnituremarket.co.uk>
<http://linqalldgood.net>
<http://neartikuliranim1.pardosrpublicadjusters.com>
<http://lua9y6ev2c.looqucoventrybuildingsociety.com>
<http://qhvofnudibriut.com>
<http://rusticflagcrafts.com>
<http://bt8x.guphobizovo.net>
<http://115.218.63.174>
<http://186.90.41.210>
<http://68.183.50.203>
<http://46.252.214.32>
<http://183.156.225.238>
<http://116.14.38.158>
<http://5.152.204.140>
<http://5.88.245.229>
<http://182.160.39.66>
<http://27.74.245.47>
<http://220.141.17.59>
<http://222.140.70.164>
<http://115.50.90.12>
<http://122.121.23.199>
<http://200.7.124.2>
<http://192.115.165.54>
<http://113.172.46.253>
<http://218.205.57.16>
<http://92.44.149.154>
<http://196.64.19.14>
<http://209.204.46.141>
<http://157.147.136.152>
<http://50.248.158.158>
<http://177.124.102.81>
<http://123.20.109.48>
<http://106.51.69.185>
<http://180.148.5.178>
<http://27.188.43.72>
<http://41.233.188.95>
<http://187.37.46.65>
<http://117.39.30.166>
<http://203.87.133.219>
<http://116.22.30.76>
<http://106.104.140.76>
<http://115.49.237.173>
<http://201.190.184.31>
<http://114.33.221.10>
<http://189.50.134.244>

<http://112.87.144.119>
<http://101.51.86.92>
<http://103.25.132.70>
<http://120.253.206.248>
<http://68.183.151.194>
<http://41.78.174.227>
<http://123.4.79.244>
<http://115.49.4.119>
<http://111.241.30.239>
<http://200.98.128.92>
<http://123.4.153.73>
<http://121.158.234.154>
<http://110.248.242.115>
<http://203.150.131.130>
<http://42.235.34.222>
<http://143.137.60.98>
<http://181.123.9.160>
<http://81.214.120.207>
<http://178.59.109.217>

Lista de URLs de ficheros lícitos. good_files.txt

http://file-examples.com/wp-content/uploads/2018/04/file_example_AVI_480_750kB.avi
http://file-examples.com/wp-content/uploads/2017/11/file_example_MP3_700KB.mp3
http://file-examples.com/wp-content/uploads/2017/02/file-sample_100kB.doc
http://file-examples.com/wp-content/uploads/2017/02/file-sample_100kB.docx
http://file-examples.com/wp-content/uploads/2017/10/file-sample_150kB.pdf
http://file-examples.com/wp-content/uploads/2017/10/file_example_PNG_500kB.png
http://file-examples.com/wp-content/uploads/2017/10/file_example_GIF_500kB.gif
http://file-examples.com/wp-content/uploads/2017/02/file_example_CSV_5000.csv
http://file-examples.com/wp-content/uploads/2017/02/zip_2MB.zip
<http://images.unsplash.com/photo-1519501025264-65ba15a82390?ixlib=rb-1.2.1&w=1000&q=80>
<http://images.unsplash.com/photo-1511447333015-45b65e60f6d5?ixlib=rb-1.2.1&w=1000&q=80>
<http://wallpapercave.com/wp/pZeyZnB.jpg>
<http://www.africau.edu/images/default/sample.pdf>
<https://www.w3.org/WAI/ER/tests/xhtml/testfiles/resources/pdf/dummy.pdf>
<http://www.pdf995.com/samples/pdf.pdf>
<http://sample-videos.com/zip/10mb.zip>
<http://sample-videos.com/img/Sample-jpg-image-50kb.jpg>
http://sample-videos.com/video123/mp4/720/big_buck_bunny_720p_1mb.mp4

Lista de URLs de ficheros maliciosos. bad_files.txt

http://file-examples.com/wp-content/uploads/2017/10/file_example_JPG_100kB.jpg

La lista de ficheros maliciosos no contiene un archivo malicioso como tal. Se ha indicado esta URL a efectos de validación del sistema, para comprobar que la detección en base al hash MD5 aplicado sobre los ficheros descargados funcionaba de forma correcta.