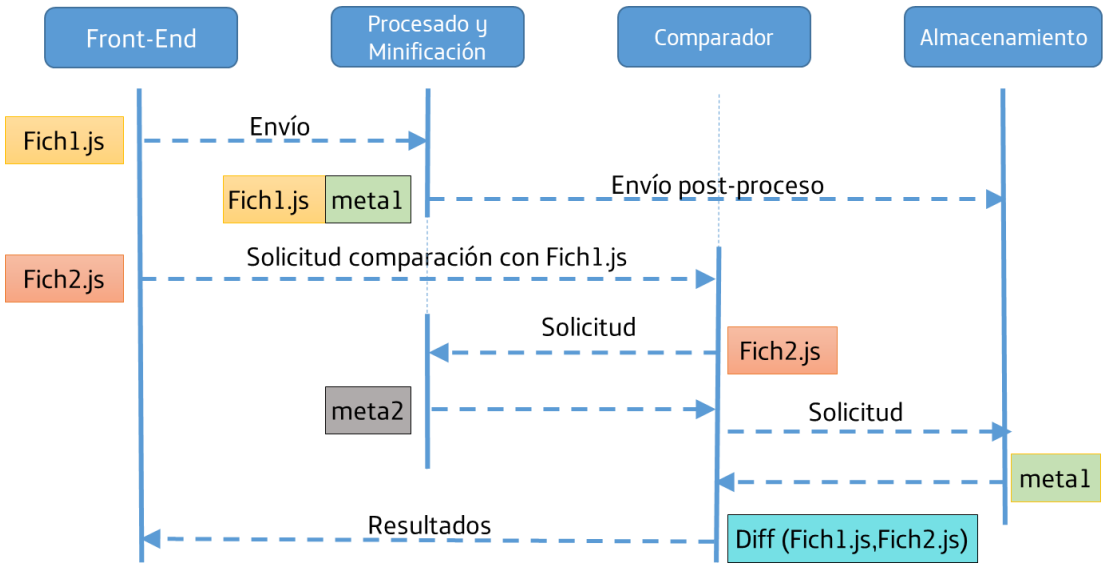
Reto 1

Programa de Tutorización Universitaria para la Transferencia Orientada mediante Retos de Innovación Avanzada (TUTORÍA) para la Red de Cátedras Telefónica: 1ª edición 2018

Reto 1. Análsis morfológico y de flujos funcionales en librerías JavaScript	
INFORMACIÓN DEL TUTOR	
Departamento	Área de Innovación y Laboratorio de ElevenPaths. https://www.elevenpaths.com/ Telefónica Digital España
Responsable	Marcos Arjona Fernández. marcos.arjona@11paths.com Director de proyectos de investigación e innovación
INFORMACIÓN GENERAL	
Carácter	Investigación científico-técnica con desarrollo experimental
Tipo	Componente de valor, el proyecto podría sustituir un elemento de un producto.
Recursos de apoyo	☐ Repositorios de software, APIs y conectores necesarios, librerías, etc. ☒ Orientación técnica relativa a la productivización e industrialización para mercado, consejos y metodologías de desarrollo, orientación técnica, sugerencias de configuraciones, etc. ☐ Licencias de productos y plataformas ElevenPaths, ☒ Servicios de asistencia y soporte especialista a la investigación y el desarrollo.
DETALLES DE LA PROPUESTA	
Contexto y Motivación	El área de innovación y laboratorio de ElevenPaths supone un motor constante de generación de tecnologías, prototipos y elementos de innovación capaces de proporcionar características diferenciales a la carta de productos y soluciones de seguridad de Telefónica. Una de las plataformas de ciberseguridad destinadas a la protección de usuarios durante la navegación centra su estudio en las elevadas capacidades y en muchos casos privilegiadas, que poseen las librerías JavaScript que son importadas y ejecutadas en los equipos de los usuarios. Suponiendo un punto de entrada de malware a distintos niveles y cuyas amenazas de seguridad pueden ser críticas para los individuos. Ante esta circunstancia, dotar a los usuarios de medidas de protección eficaces resulta prioritario, siempre y cuando la experiencia de navegación no sea mermada, las demandas a los usuarios no

	supongan acciones intrusivas y los requisitos computacionales no repercutan negativamente en los sistemas finales. Debido a esto, dichos ficheros JavaScript se convierten en los elementos más importantes de cara a incorporar medidas de seguridad tanto preventivas como paliativas y para ello es necesario disponer de mecanismos optimizados, eficaces, computacionalmente avanzados y precisos para clasificar, procesar y comparar dichos ficheros o librerías JavasScripts.
Descripción	La resolución de este reto parte por tanto de la necesidad de una tecnología capaz de procesar un fichero o código JavaScript, almacenarlo y realizar una indexación eficiente, optimizada y ágil para determinados procesos comparativos posteriores. Teniendo dichos ficheros procesados sería posible comparar diferentes versiones del mismo fichero y encontrar cambios morfológicos e inserciones anómalas de elementos que podrían ser indicativos de intenciones maliciosas. Para ello es necesario el almacenaje de acuerdo a distintos criterios de indexación componiendo un enfoque con varios niveles de abstracción, que son descritos a continuación, cada uno con un nivel de complejidad superior al anterior. Nivel Global (Nivel 1): Un fichero o código JavaScript supone un hash propio en su conjunto, lo que permitiría localizar diferencias a nivel general sin inspección profunda. Las expectativas de este nivel incluyen las capacidades de realizar comparaciones entre ficheros JavaScript a nivel de función, de esta forma se podría concretar y acotar los cambios detectados, algo que podría realizarse mediante hashes anidados relativos a cambios sintácticos en el código. Nivel Granular (Nivel 2): La comparación a nivel de código supone un factor adicional en cuanto a las capacidades de detección, aislamiento y aprendizaje (si lo hubiese). El objetivo en este caso es poder identificar los cambios entre tramos de código para mostrar las diferencias morfológicas, reduciendo a su vez diferencias de estilo incorporadas que no respondan a factores críticos en la ejecución de código, tales como tabulaciones, espacios, saltos de línea, etc. Nivel Funcional (Nivel 3): Este es el nivel de mayor precisión y el que requiere unas capacidades de análisis y preprocesado más elevadas, debiendo ser capaz de analizar el comportamiento del software para reflejar los flujos funcionales de forma que puedan ser comparados con otras versiones similares del código y poder detectar no solamente diferencias sino determinados indicadores anómalos que podrían catalogarse como elementos maliciosos. Estos flujos relativos al código JavaScript también deben ser almacenados de forma efectiva utilizando para ello modelos algebraicos eficaces que faciliten la navegabilidad y por tanto los procesos comparativos. Con esas capacidades, la tecnología desarrollada debe ser capaz de recibir ficheros, tramas de código y otros fragmentos de JavaScript para su procesado y almacenaje. Posteriormente llamadas recurrentes podrían solicitar comparaciones tanto sintácticas como funcionales sobre ficheros ya almacenados donde será necesario localizar de manera precisa los cambios, extrayendo las diferencias y en determinados casos mostrando los cambios de comportamiento acontecidos entre dos ficheros/fragmentos similares. (Extra) De forma opcional, este reto podría complementarse con un motor de aprendizaje capaz de detectar anomalías de índole maliciosa en las diferencias encontradas entre dos pares de ficheros, bien sean algunos indicadores textuales, patrones de comportamiento o elementos que hayan sido catalogados por ElevenPaths como indicadores maliciosos.

Factores de investigación e innovación	- Mecanismos de análisis, saneamiento, minificación, parseamiento, abstracción, anonimización, etc. que sean capaces de realizar una captura adecuada y precisa de los flujos funcionales del código Javascript. - Arquitectura adecuada para el almacenaje, acceso, administración y gestión de las diferencias entre librerías de forma optimizada. Escalable y modular. - Modelos algebraicos (probablemente complejos) para reflejar los flujos funcionales y el comportamiento de los ficheros JavaScript, que a su vez supongan beneficios operacionales de lectura y comparación. - Estudio sobre la problemática de las librerías JavaScript respecto a las distintas modalidades de importación que existen en el lado cliente y cómo factores como las cookies afectan al tipo y personalización del conjunto de librerías importadas. - (Extra)Motor de aprendizaje para realizar tareas de búsqueda de indicadores, patrones y secuencias, tanto textuales como funcionales sobre ficheros JavaScript.
Hitos	1) Estudio científico de la problemática de almacenaje de ficheros JavaScript por el ecosistema encontrado en páginas web dinámicas. 2) Estudio científico respecto a la problemática algebraica y computacional del almacenamiento estructurado que perseguimos en esta propuesta y los distintos modelos y topologías que pueden acoger este reto, tanto para el Backend como para la indexación de elementos de las librerías y el código. 3) Desarrollo del Backend capaz de sustentar las necesidades tecnológicas del reto 4) Capacidades funcionales del Nivel 1 y 2 con pruebas que sustenten dichas capacidades, incluyendo pruebas relativas a las capacidades comparativas de ficheros. 5) Desarrollo de los modelos algebraicos necesarios para recoger los flujos funcionales. 6) Capacidades tecnológicas del Nivel 3 relativas al almacenamiento de la funcionalidad del código JavaScript procesado y minificado. 7) Capacidades tecnológicas del Nivel 3 relativas a la comparación de flujos funcionales. 8) Pruebas de estabilidad y capacidades del motor de comparación del nivel 3. 9) (Extra) Motor de aprendizaje capaz de realizar la detección de elementos maliciosos en las diferencias localizadas a lo largo de los diferentes procesos comparativos de este reto.
Caso de Uso	Supongamos que queremos realizar una comparativa entre dos ficheros: Fich1.js y Fich2.js. Para ello inicialmente se hace una subida del fichero desde el Front-End, que procesa y minifica el fichero generando un compendio de información adicional, hashes, información de flujo relativa a la funcionalidad del código y finalmente almacena este conjunto en el Back-End. Tras eso, una solicitud para comparar ese fichero previo y el nuevo no necesariamente requiere un nuevo almacenamiento, sino únicamente la generación del fichero con los flujos (metadatos) necesarios sobre Fich2.js y compararlos con los correspondientes al Fich1.js. El siguiente diagrama mostraría este posible caso de uso.

	
Alcance e Impacto	Esta tecnología tiene una aplicabilidad directa sobre múltiples disciplinas de la seguridad destinadas a la localización de elementos sospechosos para discernir si contienen elementos maliciosos o no y la índole de los mismos. Dotar a estos sistemas de capacidades avanzadas como comparadores a nivel de flujos funcionales garantiza unas capacidades comparativas de detección de patrones de comportamiento con un elevado potencial y que facilitaría el proceso de detección de comportamientos a gran escala, unos factores de alto valor en paquetes software destinados a la prevención y alerta temprana.
Enlaces de Interés	- JAWS. https://blog.elevenpaths.com/2018/12/jaws-javascript-plugin-ciberseguridad.html - Uso de JavaScript inyectado en navegadores con usos maliciosos desapercibidos para el usuario. https://blogs.protegerse.com/2017/09/25/tecnicas-de-minado-de-criptomonedas-usadas-por-los-delincuentes/ - British Airways site had credit card skimming code injected https://arstechnica.com/information-technology/2018/09/british-airways-site-had-credit-card-skimming-code-injected/ - Card-Skimming Malware Campaign Hits Dozens of Sites Daily https://www.bankinfosecurity.com/online-skimming-50-to-60-sites-hacked-per-day-a-11451
TRL Objetivo	TRL 4: Validación de componente y/o disposición de los mismos en entorno de laboratorio.