

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación



**DESARROLLO DE UN PROTOTIPO DE
APLICACIÓN PARA DETECCIÓN DE TÉCNICAS DE
IDENTIFICACIÓN Y SEGUIMIENTO DE USUARIOS
EN LA WEB**

TRABAJO FIN DE MÁSTER

Juan Daniel Márquez Lagalla

2018

Universidad Politécnica de Madrid
Escuela Técnica Superior de Ingenieros de Telecomunicación

**Máster Universitario en
Ingeniería de Redes y Servicios Telemáticos**

TRABAJO FIN DE MÁSTER

**DESARROLLO DE UN PROTOTIPO DE
APLICACIÓN PARA DETECCIÓN DE TÉCNICAS DE
IDENTIFICACIÓN Y SEGUIMIENTO DE USUARIOS
EN LA WEB**

Autor

Juan Daniel Márquez Lagalla

Tutor

José María del Álamo Ramiro

Departamento de Ingeniería de Sistemas Telemáticos

2018

Resumen

Una práctica que se encuentra establecida a lo largo de la web consiste en obtener datos acerca del terminal que emplea el usuario, como el software que hace la función de navegador web y el sistema operativo sobre el cual se ejecuta. Normalmente, los navegadores proveen información sobre la configuración de hardware y software del sistema, permitiendo a los sitios web hacer un mejor uso de los recursos disponibles y ajustar sus formatos de presentación. No obstante, existe la posibilidad de que se utilice esta información para otros fines, como la identificación y seguimiento del usuario en la web.

Usualmente, los métodos que permiten obtener la información de un dispositivo para identificarlo y monitorizarlo se manifiestan de forma que los usuarios finales de las aplicaciones web son inconscientes de su presencia. Por ello, en el presente trabajo de investigación se desarrolla un prototipo de aplicación para detección de técnicas de identificación y seguimiento de usuarios en la web, mediante la comprobación de criterios que determinan la existencia de esta práctica.

Para el desarrollo de la aplicación primero se realiza un estudio del estado del arte de los distintos mecanismos que permiten obtener datos de los usuarios, tanto aquellos que dependen del almacenamiento de información en la plataforma del usuario o dependientes del estado, como los que no dependen del estado y recolectan datos a través de los protocolos de aplicación y scripts. A los últimos también se les conoce como mecanismos de extracción de huellas.

Los criterios de detección implementados en el prototipo de aplicación se delimitan en torno a técnicas que figuran en el ámbito de la extracción activa de huellas, como el uso de ciertos objetos JavaScript para obtener atributos altamente identificadores o con valores de entropía elevados.

La aplicación se implementa sobre un entorno de Jupyter Notebook, integrando los scripts que contienen los criterios de detección desarrollados en lenguaje Python y un rastreador web que permite obtener los datos relevantes para el análisis. A lo largo de las fases de diseño y desarrollo se definen los principios que permiten extenderla e implementar futuros criterios de detección.

Finalmente, se llevan a cabo un conjunto de pruebas para validar la aplicación. De dos listas conformadas por 1.000 sitios web, la aplicación pudo detectar que 5,5% de los sitios de un conjunto y 2,6% del otro, llevan a cabo alguna técnica de extracción de huellas. Los resultados demuestran la utilidad técnica de la aplicación y permiten concientizar a la comunidad de usuarios y desarrolladores de aplicaciones web de la existencia de estas prácticas.

Abstract

A commonplace practice throughout the web consists to obtain data related to the user's device, such as the software serving as the web browser or the operative system on which it runs. Normally, browsers provide information about the system's hardware and software configurations, enabling websites to make a better use of the available resources and adjust their layouts. However, this information could also be used to identify and track users online.

Usually, the end users of the web applications are unaware of the ways in which the information related to their devices are extracted and used for tracking purposes. Thus, in the following dissertation, a prototype application is developed to detect identification and tracking techniques on the web, implementing several defining criteria related to these practices.

The development of the application begins with a research of the state of the art regarding mechanisms that allow obtaining user's data, including both those that rely on information stored in the device, also called stateful mechanisms, and those that collect data by exploiting the application level protocols or executing certain scripts, also called stateless mechanisms. The latter is commonly addressed as fingerprinting.

The detection criteria that the prototype application implements are centered around active fingerprinting techniques, such as those that use certain JavaScript objects to obtain highly identifying attributes, i.e., attributes with high entropy values.

The application is implemented on a Jupyter Notebook environment, integrating the scripts that contain the detection criteria, developed in Python, and a web crawler which allows the application to obtain the relevant data needed for the analysis of fingerprinting techniques. Throughout the design and development phases, the basics regarding the extendibility of the application and implementation of new detection criteria are defined.

Finally, a set of tests are conducted to validate the application. From two datasets made up of 1,000 websites, the application could detect that 5.5% of the sites of one dataset and 2.6% of the other one, were carrying out some fingerprinting technique. The results not only allow to assess the technical viability of the application but can also help to increase the awareness of these practices around the community of web applications' users and developers.

Índice general

	Pág.
RESUMEN	I
ABSTRACT	III
ÍNDICE GENERAL	V
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	XI
SIGLAS	XIII
1. INTRODUCCIÓN	1
1.1 Contexto.....	1
1.2 Objetivos	4
1.2.1 Objetivo general.....	4
1.2.2 Objetivos específicos	4
1.3 Estructura del documento.....	4
2. ESTADO DEL ARTE	6
2.1 HTTP y elementos que pueden introducir vulnerabilidades.....	6
2.2 HTTP sobre TCP/IP.....	7
2.3 Sintaxis del tráfico HTTP.....	8
2.3.1 Cabeceras HTTP	10
2.4 Seguimiento de usuario: información almacenada en cliente.....	11
2.4.1 Cookie HTTP.....	11
2.4.2 Etiqueta de entidad HTTP.....	12
2.4.3 Almacenamiento web	13
2.4.4 Objetos locales compartidos	13
2.5 Técnicas de seguimiento basada en cookies	14
2.5.1 Supercookies, evercookies o zombie cookies	14
2.5.2 Cookies de origen.....	14
2.5.3 Cookies de terceros	15
2.5.4 Sincronización de cookies	15
2.6 Identificación de usuarios: extracción de huellas	16
2.7 Extracción pasiva de huellas.....	16
2.8 Extracción activa de huellas.....	16
2.8.1 Basado en JavaScript.....	17
2.8.1.1 Canvas.....	18
2.8.1.2 WebGL	18
2.8.1.3 WebRTC.....	19

2.8.1.4	Audio.....	19
2.8.1.5	Batería	19
2.8.1.6	Extensiones del navegador.....	20
2.8.2	Basado en plugins	20
2.8.2.1	Lista de fuentes	20
2.9	Atributos de una huella.....	21
2.10	Herramientas de evaluación de huellas	24
2.10.1	Rastreador web	25
2.10.2	Instancias del rastreador web	25
2.10.3	Administrador de tareas.....	25
2.10.4	Recolector de datos	26
2.10.5	Herramientas disponibles	26
2.11	OpenWPM.....	27
2.12	Resumen y clasificación general de técnicas de seguimiento	29
3.	ANÁLISIS DE TÉCNICAS DE EXTRACCIÓN DE HUELLAS EN LA WEB..	30
3.1	Selección de las técnicas de seguimiento a detectar.	30
3.1.1	Presencia de la técnica en la web según estudios previos.	30
3.1.2	Valor de entropía del atributo que permite extraer la técnica.	31
3.1.3	Técnicas elegidas.	32
3.2	Criterios para la elaboración de scripts de análisis.	32
3.2.1	Detección de la técnica basada en objetos informativos.	33
3.2.2	Detección de la técnica basada en Canvas.	34
3.2.3	Detección de la técnica de fuentes basada en Canvas.	34
3.2.4	Detección de la técnica basada en WebRTC.	35
4.	DESARROLLO DE LA HERRAMIENTA DE DETECCIÓN.....	36
4.1	Decisiones de diseño.	36
4.2	Diseño.	36
4.3	Implementación de scripts de análisis y mejoras adicionales.....	38
4.4	Presentación de la aplicación prototipo.	39
4.5	Extensión de la aplicación prototipo.	39
5.	VALIDACIÓN Y EVALUACIÓN DE RESULTADOS	41
5.1	Definición de pruebas.....	41
5.2	Desarrollo de pruebas.....	42
5.3	Análisis de resultados.....	42
5.3.1	Resultados del conjunto de sitios web con base al ranking Alexa.	42

5.3.2	Resultados del conjunto de sitios web con base al ranking Majestic Million.	45
5.4	Otras pruebas desarrolladas.	48
5.4.1	Resultados del conjunto de sitios web de universidades públicas.....	48
5.4.2	Resultados del conjunto de sitios web de entidades del sector público... ..	48
6.	CONCLUSIONES Y LÍNEAS FUTURAS DE INVESTIGACIÓN.....	50
6.1	Conclusiones	50
6.2	Líneas futuras de investigación.....	52
	BIBLIOGRAFÍA	54
	ANEXOS	60
	Anexo 1. Ejemplo de resultado de script con técnica de Canvas [52].....	60
	Anexo 2. Ejemplo de <i>measureText</i> para la técnica de Canvas-Fuentes [53]	61
	Anexo 3. Entidades y atributos respectivos de la base de datos generada por OpenWPM.....	62
	Anexo 4. Capturas de pantalla de la aplicación prototipo.	64
	Anexo 5. Manual de desarrollo.	66
	Anexo 6. Configuraciones de OpenWPM.....	73
	Anexo 7. Marco regulatorio relacionado al <i>fingerprinting</i> web.....	75

Índice de figuras

Figura 1. Ejemplo de componente de aplicación web con HTML/CSS/JS [1].....	2
Figura 2. Extracto del resultado de Panoptick.	3
Figura 3. Cliente y Servidor HTTP	6
Figura 4. Comunicación HTTP sobre TCP/IP	7
Figura 5. Mecanismo de instanciación de cookies [10].	12
Figura 6. Proceso de sincronización de cookies [18].	15
Figura 7. Ejemplo de <i>fingerprinting</i> activo [19].....	17
Figura 8. Ejemplo de arquitectura de una herramienta de evaluación de <i>fingerprinting</i>	25
Figura 9. Clasificación de técnicas de seguimiento e identificación de usuarios en la web.	29
Figura 10. Detalle de la aplicación prototipo según arquitectura.	36
Figura 11. Diagrama de entidad-relación de la base de datos generada por OpenWPM.....	38
Figura 12. Proporción de técnicas de <i>fingerprinting</i> - Alexa.	43
Figura 13. Número de técnicas empleadas por script - Alexa.....	44
Figura 14. Número de sitios web que emplean técnicas de <i>fingerprinting</i> , por categoría - Alexa.	45
Figura 15. Proporción de técnicas de <i>fingerprinting</i> - Majestic Million.	46
Figura 16. Número de técnicas empleadas por script - Majestic Million.	47
Figura 17. Número de sitios web que emplean técnicas de <i>fingerprinting</i> , por categoría - Majestic Million.	47

Índice de tablas

Tabla 1. Formato de petición HTTP (<i>HTTP Request</i>).....	9
Tabla 2. Formato de respuesta HTTP (<i>HTTP Response</i>).....	9
Tabla 3. Atributos más significativos en una huella según Panopticlick y AmIUnique [31]......	21
Tabla 4. Estadísticas de zona horaria según AmIUnique en Enero de 2018 [33]	22
Tabla 5. Valores de entropía normalizados para los atributos en los conjuntos de datos de Panopticlick, AmIUnique y Gómez-Boix et al. [34].	23
Tabla 6. Comparativa entre herramientas de evaluación de <i>fingerprinting</i>	27
Tabla 7. Prevalencia de técnicas de <i>fingerprinting</i> según Englehardt y Narayanan [25].	30
Tabla 8. Resumen de técnicas elegidas.....	32
Tabla 9. Resumen de resultados – Alexa.....	43
Tabla 10. Resumen de resultados – Majestic Million.	46
Tabla 11. Resumen de resultados – universidades públicas.	48
Tabla 12. Resumen de resultados – entidades del sector público.	49

Siglas

API	Application Programming Interface
BD	Base de Datos
CSS	Cascading Style Sheets
CPU	Central Processing Unit
CSV	Comma-separated values
DNT	Do Not Track
DOM	Document Object Model
DFPM	Don't FingerPrint Me
DHCP	Dynamic Host Configuration Protocol
EFF	Electronic Frontier Foundation
ETag	Entity Tag
GPL	GNU General Public License
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
HTML5	HyperText Markup Language Version 5
PHP	Hypertext Preprocessor
HTTP	Hypertext Transfer Protocol
ID	Identifier/Identificador
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
ISP	Internet Service Provider
JS	JavaScript
KB	Kilobyte
LSO	Local Shared Object
NAT	Network Address Translation
OpenWPM	Open Web Privacy Measurement
RFC	Request for Comments
STUN	Session Traversal Utilities for NAT
SDK	Software Development Kit
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
VPN	Virtual Private Network
WebGL	Web Graphics Library
WebRTC	Web Real-Time Communication
WWW	World Wide Web
W3C	World Wide Web Consortium

1. Introducción

En este capítulo se describen los argumentos que en conjunto dan forma a la concepción del tema en estudio, las formas en que éste se manifiesta en su entorno, las características únicas que permiten su tratamiento como una situación de interés o problemática y la delimitación sobre los procesos a través de los cuáles se busca otorgar una respuesta a dicha problemática.

1.1 Contexto

La WWW o *World Wide Web*, es un medio que ha revolucionado las comunicaciones y que ha tenido una rápida evolución desde su introducción a finales de los años 80 e inicios de los años 90. La idea que movió a Tim Berners-Lee, fue desarrollar un método eficiente y rápido para intercambiar datos entre la comunidad científica. Para ello, combinó dos tecnologías ya existentes (el hipertexto y el protocolo de comunicaciones de Internet), creando un nuevo modelo de acceso a la información intuitivo e igualitario.

Actualmente, el crecimiento, difusión y expansión en el uso de la web ha incrementado de manera que se puede considerar parte como importante de la sociedad y uno de los medios principales de transmisión de todo tipo de información. De igual manera, el conjunto de tecnologías que permiten enriquecer la experiencia de los usuarios también han evolucionado a un ritmo acelerado.

Librerías JavaScript modernas o elementos innovadores de HTML5 han permitido el desarrollo de auténticas aplicaciones basadas en la web, habilitando la presentación de contenidos dinámicos que distan de los modelos estáticos predominantes del pasado y en general, redefiniendo el alcance y los límites de los elementos servidos a través de la web.

No obstante, las tecnologías que convenientemente hacen posible las aplicaciones web de hoy en día suelen ser usadas a expensas de la privacidad de los usuarios. Esto se debe a que para proveer servicios cada vez más diversos, complejos y eficientes, los sitios web requieren conocer datos acerca del terminal que permite al usuario acceder a estos contenidos, es decir, el software que hace la función de navegador web y el sistema operativo sobre el cual se ejecuta.

Normalmente, los navegadores proveen información sobre la configuración de hardware y software del sistema, permitiendo a los sitios web hacer un mejor uso de los recursos disponibles y ajustar sus formatos de presentación, como en el ejemplo que se muestra en la figura 1.

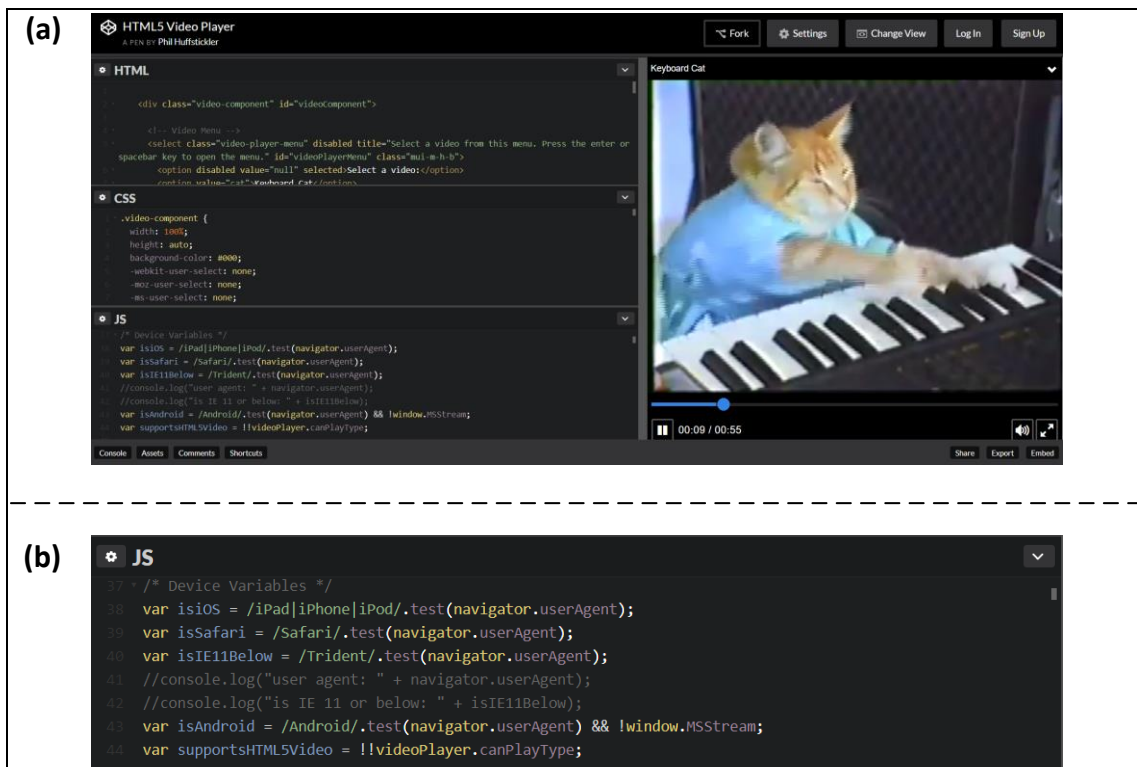


Figura 1. Ejemplo de componente de aplicación web con HTML/CSS/JS [1].

En el ejemplo se presenta un reproductor de vídeo que emplea elementos de HTML5 como `<video>`. En la figura 1a se presenta el código HTML, CSS y JavaScript empleado a la izquierda y a la derecha una muestra del reproductor de vídeo resultante. Nótese como en el código JavaScript (figura 1b) se requiere determinar el agente de usuario (`navigator.userAgent`) a través del cual se accede al contenido, con la finalidad de verificar la compatibilidad del navegador web con los elementos de HTML5 presentes.

Además del caso expuesto en el ejemplo de la figura 1 se debe considerar la posibilidad de que se utilice esta información para otros fines, como la identificación y seguimiento en la web o “*Web Tracking*”, práctica que consiste en recolectar distintos datos del dispositivo o terminal del usuario con el objetivo de monitorear su comportamiento, registrar su actividad y reconocerlo de forma unívoca al ingresar en un determinado sitio web.

Los motivos por los cuales puede darse la identificación y seguimiento de usuarios en la web son diversos, por ejemplo, algunas compañías pueden realizar esta práctica para dirigir anuncios publicitarios personalizados y campañas de marketing [2]. Indistintamente del objetivo, existe un riesgo a la privacidad del usuario ya que es posible obtener información del mismo sin su consentimiento.

Adoptar medidas para hacer frente al *web tracking* resulta complejo ya que las tecnologías que hacen posible esta práctica presentan una alta diversidad y estas, a su

vez, se han vuelto imprescindibles para los usuarios que desean acceder a las aplicaciones web de hoy en día. De acuerdo a Iglesias [3], existen distintas categorías de seguimiento en la web según el tipo de tecnologías usadas, como aquellas capaces de extraer las configuraciones de hardware y software del sistema a través de elementos del protocolo HTTP o por medio de scripts y otros tipos de código ejecutables por los navegadores en páginas web. Otras tecnologías incluyen las que emplean mecanismos de almacenamiento en el terminal cliente o las que permiten realizar un análisis del comportamiento de los usuarios.

A pesar de la complejidad, se han desarrollado herramientas que permiten detectar todo el conjunto de datos que un seguidor en la web puede extraer de un usuario. Por ejemplo, Panoptick [4] desarrollada por la EFF permite probar la protección de un navegador web ante técnicas de seguimiento e identificación en la web y provee una “huella digital” o *fingerprint* basado en la información extraída a través de dichas técnicas. Un ejemplo del resultado de esta herramienta es mostrado en la figura 2.

Test	Result
Is your browser blocking tracking ads?	✗ no
Is your browser blocking invisible trackers?	✗ no
Does your browser unblock 3rd parties that promise to honor Do Not Track?	✗ no
Does your browser protect from fingerprinting?	✗ your browser has a unique fingerprint

Note: because tracking techniques are complex, subtle, and constantly evolving, Panoptick does not measure all forms of tracking and protection.

Your browser fingerprint **appears to be unique** among the 1,427,402 tested so far.

Currently, we estimate that your browser has a fingerprint that conveys **at least 20.44 bits of identifying information.**

Figura 2. Extracto del resultado de Panoptick.

Basado en los principios en los que Panoptick y otras herramientas similares se basan, se plantea una propuesta fundamentada en un prototipo de aplicación para detección de técnicas de identificación y seguimiento de usuarios en la web. Dicho prototipo presenta como característica importante la capacidad de determinar, según criterios definidos, si un sitio web emplea una tecnología en particular con la finalidad de llevar cabo la práctica de web tracking.

1.2 Objetivos

Los objetivos representan la finalidad de la investigación, de forma que el cumplimiento de los mismos debe dar respuesta a la situación planteada. En los siguientes apartados se concretan tanto el objetivo general como los objetivos específicos asociados al tema en estudio.

1.2.1 Objetivo general

Desarrollar un prototipo de aplicación para detección de técnicas de identificación y seguimiento de usuarios en la web, mediante la comprobación de criterios que determinan la existencia de esta práctica.

1.2.2 Objetivos específicos

- Describir los mecanismos habituales de identificación y seguimiento de usuarios, mediante un estudio del estado del arte de las tecnologías habilitantes.
- Analizar los criterios empleados para la detección de técnicas de identificación y seguimiento de usuarios en la web, seleccionando aquellas que se consideren más relevantes.
- Desarrollar el prototipo de aplicación, tomando en cuenta decisiones de diseño en relación a las herramientas y técnicas definidas y según las condiciones de detección relacionadas a una tecnología en particular.
- Evaluar la ejecución del prototipo de aplicación, mediante la aplicación de pruebas sobre un conjunto de sitios web, así como la validación sobre la base de los resultados obtenidos.

1.3 Estructura del documento

El presente trabajo de investigación se encuentra distribuida en seis capítulos, distribuidos de la siguiente manera:

- Capítulo 1: Definición de la situación o tópico de estudio, que abarca el planteamiento del contexto de la investigación, así como los objetivos que permiten su resolución.
- Capítulo 2: Estado del arte, el cual incluye todas aquellas teorías o modelos pertinentes a la comprensión de la investigación, destacando el estado del arte de las distintas tecnologías y técnicas relacionadas a la identificación y seguimiento de usuarios en la web.
- Capítulo 3: Incluye el análisis de los criterios que caracterizan los métodos de identificación y seguimiento de usuarios a detectar.
- Capítulo 4: Describe el diseño y desarrollo de la aplicación prototipo, incluyendo la delimitación e implementación de los medios técnicos empleados para su funcionamiento.

- Capítulo 5: Presentación de los resultados obtenidos mediante la aplicación del prototipo propuesto.
- Capítulo 6: Exposición de las conclusiones y líneas futuras de investigación con relación al caso de estudio desarrollado.

2. Estado del arte

El presente capítulo presenta un análisis teórico en el cual se incluyen todas aquellas teorías o modelos pertinentes a la comprensión del tópico de estudio, tales como la definición de términos básicos., fundamentos teóricos, así como el estado actual de diversas técnicas y herramientas, que enmarcan y respaldan la perspectiva bajo la cual se aborda la investigación.

2.1 HTTP y elementos que pueden introducir vulnerabilidades

El Protocolo de transferencia de hipertexto o *Hypertext transfer protocol* (HTTP) es el protocolo fundamental de la web, diseñado para operar sobre una arquitectura distribuida cliente/servidor. Es un protocolo simple que originalmente fue desarrollado para obtener recursos de texto estáticos. Su versión original fue HTTP/0.9 presentada en 1991 por Tim Berners-Lee y le siguieron las versiones HTTP/1.0 publicada en el RFC 145 del año 1996 y HTTP/1.1, la versión vigente del protocolo, publicada por primera vez en el RFC 2068 del año 1997. El desarrollo del estándar es llevado a cabo principalmente por la IETF y el W3C [5].

HTTP utiliza un modelo basado en mensajes donde el terminal cliente envía un mensaje de petición y el servidor devuelve un mensaje de respuesta, como se muestra en la figura 3.

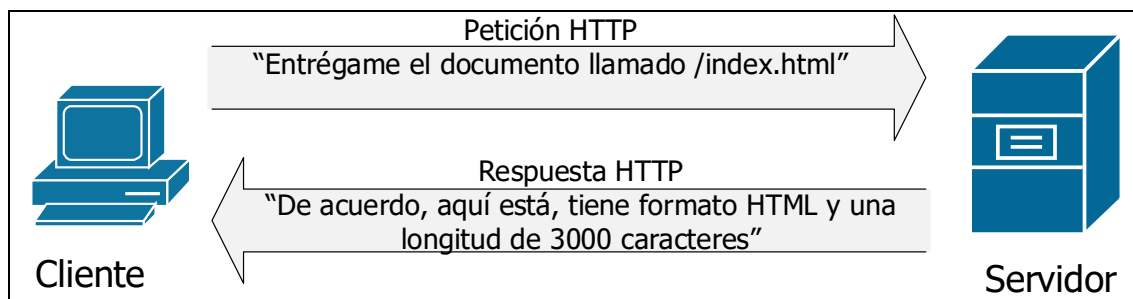


Figura 3. Cliente y Servidor HTTP

Entre las principales características de HTTP se tiene:

- HTTP es un protocolo sin estado: La petición actual no conoce lo que se ha llevado a cabo en peticiones anteriores.
- HTTP permite la negociación del tipo de datos y representación. Una representación es la instanciación de un recurso o elemento solicitado por el cliente que se encuentra en el servidor.
- Cada recurso se identifica por medio de un identificador de recursos uniforme o *Uniform Resource Identifier* (URI). Un ejemplo de URI es el siguiente: <http://www.dit.upm.es/~posgrado/muirst/>. Un *Uniform Resource Name*

(URN), que define un nombre y el *Uniform Resource Locator* (URL), que identifica un recurso por su localización y medios de acceso, son tipos de URI.

2.2 HTTP sobre TCP/IP

HTTP es un protocolo de aplicación que se ejecuta típicamente sobre una conexión TCP/IP, como se ilustra en la figura 4.

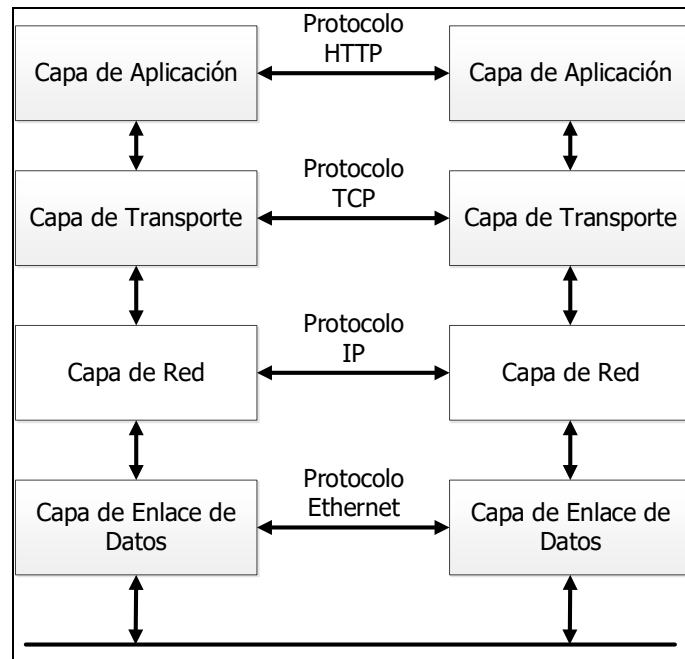


Figura 4. Comunicación HTTP sobre TCP/IP

TCP/IP (*Transmission Control Protocol/Internet Protocol*) representa un conjunto de protocolos de capa de transporte y de red a través de los cuales los nodos de una red pueden comunicarse entre sí.

TCP es un protocolo de capa de transporte, responsable de establecer conexiones entre nodos de una red. TCP es un protocolo confiable, donde cada paquete es asignado un número de secuencia y se espera una confirmación de recepción por cada paquete enviado. Un paquete es retransmitido si no ha llegado a su destino.

El protocolo IP se encuentra en la capa de red y se encuentra asociado al direccionamiento y encaminamiento. En una red IP, cada nodo tiene asignado una dirección IP única y el protocolo de encaminamiento es el encargado de redirigir un mensaje de una dirección IP origen a una dirección IP destino. Existen dos versiones vigentes del protocolo IP, IPv4 de 32 bits de longitud e IPv6 de 128 bits de longitud. A su vez, una dirección IP puede ser de dos tipos, públicas y privadas. Una dirección pública es la que permite al cliente acceso directo a Internet mientras que una dirección privada no apunta a Internet y es usada en redes de ámbitos locales o internas.

Una dirección IP es el principal elemento de capas inferiores que puede ser usado para la identificación y el seguimiento o *tracking* de usuarios en la web. La mayoría de los servidores web registran todas las direcciones IP privadas que los visitan, así como la fecha y la hora de cada visita, entre otras informaciones. Por medio de la dirección IP pública de un usuario en la web se puede determinar su proveedor de servicios de Internet o *Internet Service Provider* (ISP) y la ubicación física del usuario dentro de una ciudad y un país. No obstante, en ocasiones la información sobre la ubicación de un usuario según su dirección IP puede no ser correcta ya que un ISP puede poseer un conjunto de direcciones distribuidas a lo largo de distintos países [6].

Además de la información sobre la dirección IP que se almacena en servidores web, se debe tomar en cuenta el rol de los ISP en la gestión de las direcciones de sus usuarios. En una conexión a Internet, un cliente puede utilizar la misma dirección IP pública en cada acceso a la red del ISP (IP estática) o una dirección diferente para cada acceso (IP dinámica). Una dirección IP estática es asignada por el ISP y permanece sin cambios en el transcurso del tiempo, a diferencia de las direcciones IP dinámicas que se caracterizan por ser variables y son asignadas a partir de un rango predefinido por medio de un protocolo llamado *Dynamic Host Configuration Protocol* (DHCP).

Independientemente de si el cliente está usando una dirección IP estática o conectado a través de un dispositivo que cumple una función de traducción de direcciones de red *Network Address Translation* (NAT), un ISP puede registrar todas las actividades de navegación web y almacenarlas en una base de datos vinculada a la identidad real del cliente. Esto es posible ya que el ISP conoce que dirección IP ha sido asignada a cada cliente y en qué instante.

2.3 Sintaxis del tráfico HTTP

Un cliente inicia una sesión HTTP estableciendo una conexión TCP con el servidor, típicamente al puerto 80, y posteriormente envía una petición o *request* que contiene la localización del recurso solicitado o URL. En respuesta (*response*), el servidor entrega el recurso solicitado, y en el caso más rudimentario, finaliza posteriormente la conexión TCP de manera inmediata.

Todos los mensajes HTTP (petición y respuesta) consisten en una o más cabeceras, cada una en una línea por separado, seguidas por una línea en blanco obligatoria y por un cuerpo de mensaje opcional. Un ejemplo de una petición HTTP se muestra en la tabla 1 a continuación:

Tabla 1. Formato de petición HTTP (HTTP Request)

Petición HTTP
<p>GET https://moodle.lab.dit.upm.es/pluginfile.php/13/user/icon/standard/f1?rev=1 HTTP/1.1 Accept: text/html, application/xhtml+xml, application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8 Accept-Encoding: gzip, deflate, br Accept-Language: es-ES,es;q=0.9,en-US;q=0.8,en;q=0.7 Cookie: MoodleSession=j7pgqrnbkntntv9pqi3auke0; MOODLEID1_=%259A%25100%2508%259E%2522N%25FE%2514g DNT: 1 Referer: https://moodle.lab.dit.upm.es/user/profile.php?id=2 Upgrade-Insecure-Requests: 1 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.97 Safari/537.36 Vivaldi/1.94.1008.44</p>

La primera línea de toda petición HTTP contiene los siguientes tres elementos, separados por espacios:

- Un verbo indicando el método HTTP. El método más usado es GET, cuya función es obtener un recurso de un servidor web. Las peticiones GET no presentan un cuerpo de mensaje, por lo que no se añade ningún dato adicional después de la línea en blanco que sigue a las cabeceras. Otros métodos HTTP son: POST, HEAD, OPTIONS, PUT, DELETE, TRACE y CONNECT [7].
- La URL solicitada. Una URL puede contener una cadena de solicitud opcional que contenga parámetros que el cliente este transfiriendo al recurso. En el ejemplo de la tabla 1 existe un parámetro con el nombre *rev* y valor *1*.
- La versión de HTTP utilizada. En el caso del ejemplo anterior, HTTP versión 1.1.

Posteriormente se observan las distintas cabeceras HTTP incluidas. En la tabla 1 se observan las siguientes cabeceras: *Accept*, *Accept-Encoding*, *Accept-Language*, *Cookie*, *Referer*, *Upgrade-Insecure-Requests* y *User-Agent*.

Un mensaje de respuesta HTTP, por su parte, presenta en el formato mostrado en la tabla 2:

Tabla 2. Formato de respuesta HTTP (HTTP Response)

Respuesta HTTP
<p>HTTP/1.1 200 OK Accept-Ranges: bytes Cache-Control: private, max-age=31536000, no-transform Content-Disposition: inline; filename="f1.png" Content-Length: 14830 Content-Type: image/png Date: Wed, 24 Jan 2018 15:48:16 GMT Etag: "33c139ade5fd3872429d47d366ef1b3c0de0459b" Expires: Thu, 24 Jan 2019 15:48:16 GMT Last-Modified: Wed, 26 Oct 2011 17:04:50 GMT Pragma: Server: Apache/2.2.22 (Ubuntu) X-Powered-By: PHP/5.3.10-1ubuntu3.26</p>

La primera línea de toda respuesta HTTP contiene los siguientes tres elementos, separados por espacios:

- La versión HTTP utilizada.
- Un código de estado numérico indicando el resultado de la petición. El código 200 es el más común de los estados, indicando que la petición fue exitosa y que el recurso solicitado ha sido entregado.
- Una frase en forma de texto indicando el estado de la respuesta (ej. OK) [8].

Al igual que el mensaje de petición, el mensaje de respuesta lista a continuación el conjunto de cabeceras utilizadas. En la tabla 2 se pueden observar cabeceras tales como: *Accept-Ranges*, *Cache-Control*, *Content-Disposition*, *Content-Length*, *Content-Type: Date*, *Etag*, *Expires*, *Last-Modified*, *Pragma*, *Server* y *X-Powered-By*.

2.3.1 Cabeceras HTTP

Las cabeceras HTTP pueden ser utilizadas para el seguimiento y obtención de información de usuarios en la web. Se destacan las siguientes cabeceras [9]:

- **Accept:** Utilizada para anunciar los tipos de datos que soporta el navegador web del usuario. Además, un navegador puede declarar una preferencia sobre tipos de datos específicos.
- **Accept-Language:** Es similar a la cabecer *Accept*. La cabecera *Accept-Language* es usada para anunciar una preferencia por un idioma en particular.
- **Accept-Encoding:** Es usada para anunciar las capacidades de codificación del navegador web del usuario.
- **Accept-Charset:** Anuncia los conjuntos de caracteres aceptados por el navegador web y preferencias.
- **Connection:** Especifica que ocurrirá con la conexión después de que la petición haya sido respondida. Valores de esta cabecera incluyen *keep-alive* y *close*.
- **User-Agent:** Contiene información sobre el navegador web y la plataforma del usuario. La información en muchos casos es larga y ampulosa, conteniendo diversas capacidades y números de versiones completos (ej. Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.97 Safari/537.36 Vivaldi/1.94.1008.44), pero también puede ser pequeña y proveer poca información.
- **ETag:** Una ETag o entity tag es parte de un mecanismo de HTTP que provee validación de la caché web y su objetivo es controlar el tiempo que un fichero puede permanecer en la caché del cliente.
- **Cookie:** Mecanismo que permite al servidor enviar datos al cliente, que posteriormente el cliente almacena y reenvía en peticiones posteriores al

servidor, sin ninguna acción en particular requerida por la aplicación o el usuario. Son frecuentemente usadas como medio para explotar vulnerabilidades.

2.4 Seguimiento de usuario: información almacenada en cliente

Los métodos descritos a continuación están relacionados con la capacidad que presentan los clientes de almacenar un valor recibido desde un servidor. Para cada método de almacenamiento, es posible para el cliente detectar el valor establecido o recuperado y evitarlo o modificarlo. La dificultad y posibilidad de que esto ocurra es diferente dependiendo de la ubicación del almacenamiento, dando a cada uno características particulares.

2.4.1 Cookie HTTP

Las cookies son archivos principalmente almacenados en el terminal del usuario a través del navegador. Usualmente se encuentran cifradas (sólo el sitio web que crea las cookies puede leerlas) y son empleadas para el manejo de sesiones, almacenamiento de las preferencias del sitio web, autenticación e identificación de clientes.

Las cookies se crean cuando un navegador web envía una petición a un servidor. El servidor puede solicitar la instanciación de una cookie en el terminal del cliente como respuesta. Si el navegador acepta la solicitud, la cookie es almacenada y enviada al servidor dentro de una cabecera HTTP cuando el cliente realice una nueva petición al mismo dominio.

El proceso detallado de creación y manejo de cookies se muestra en la figura 5. La primera vez que un usuario visita un sitio web, el servidor web no conoce nada sobre el usuario (figura 5a). El servidor web espera que este mismo usuario regresé, por lo que instanciará una cookie unívoca en el usuario para poder identificarlo en el futuro. La cookie contiene una lista arbitraria de información en formato *nombre=valor*, y es adosada al usuario utilizando las cabeceras *Set-Cookie* o *Set-Cookie2* en la respuesta HTTP.

Las cookies pueden contener cualquier información, pero usualmente solo contienen un único número de identificación, generado por el servidor con finalidades de *tracking*. Por ejemplo, en la figura 5b, el servidor instancia una cookie en el usuario que indica *id="34294"*. El servidor puede utilizar este número para consultar en la base de datos información que el servidor acumula sobre sus visitantes.

El navegador web tiene presente los contenidos de la cookie que fueron enviados por el servidor en las cabeceras *Set-Cookie* o *Set-Cookie2*, almacenando el conjunto de cookies en una base de datos de cookies del navegador. Cuando el usuario regresa al mismo sitio web en un futuro (figura 5c), el navegador seleccionará las cookies que fueron

instanciadas por ese servidor y las enviará de nuevo en una cabecera *Cookie* en la petición HTTP [10].

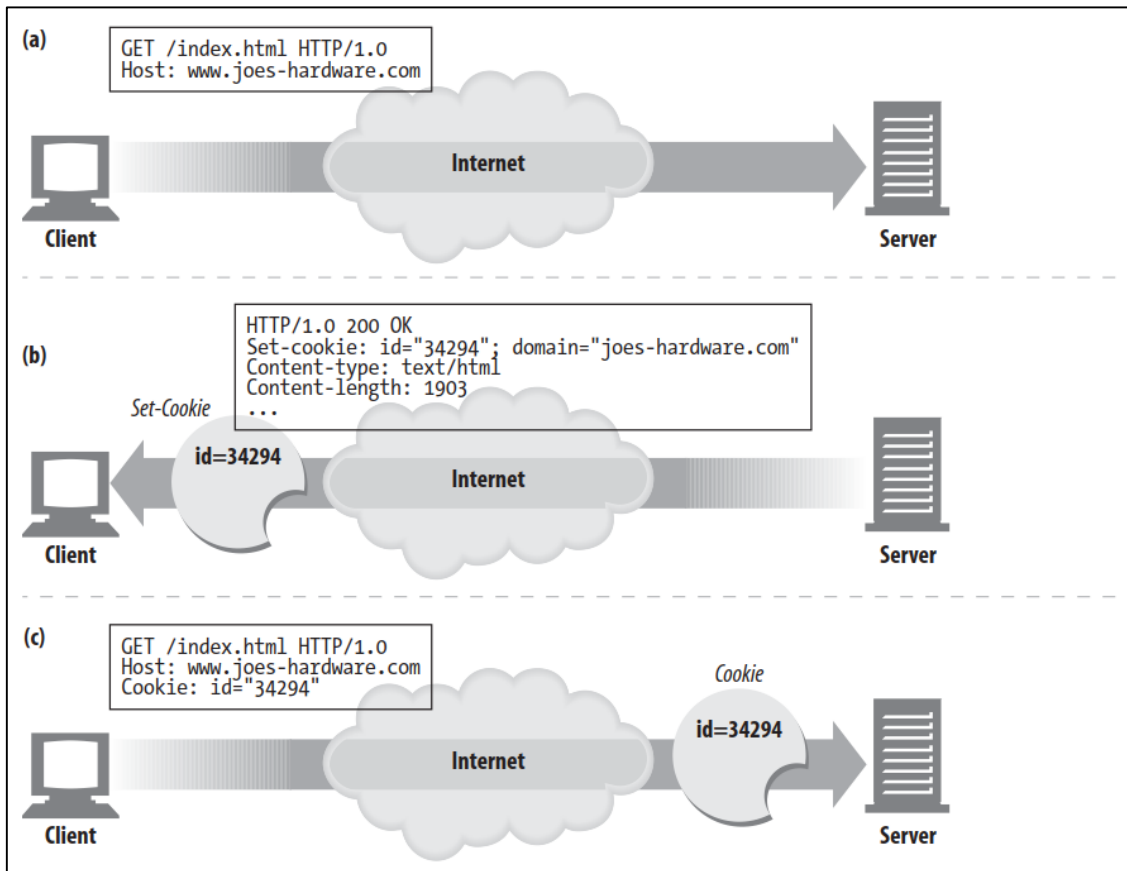


Figura 5. Mecanismo de instanciación de cookies [10].

Existen dos categorías principales de cookies: cookies de sesión y cookies persistentes. Las cookies de sesión o transitorias se almacenan en una ubicación temporal en el navegador del usuario y se eliminan cuando éste cierra el navegador o cierra su sesión. Las cookies persistentes son aquellas que pueden incluir mecanismos de rastreo por parte de terceros.

2.4.2 Etiqueta de entidad HTTP

Las etiquetas de entidad o ETags permiten que un navegador web evite cargar los mismos recursos de forma duplicada, como archivos de música o imágenes. En la primera visita al sitio web, el servidor envía al cliente un archivo dado para su descarga y un ETag. Cuando el navegador web del cliente solicite el sitio nuevamente, el servidor puede informar que el archivo no ha cambiado, haciendo que el navegador use la copia del archivo almacenada en su caché. Si el servidor genera una ETag distinta, quiere decir que existe una versión más nueva del archivo que debe descargar el cliente.

A pesar de que los ETag deben ser utilizados como firmas que validan si el caché o almacenamiento web de un cliente ha sido actualizado, también han sido empleados

para almacenar y recuperar información como agente de *tracking* de usuarios. A diferencia de la cabecera HTTP *Last-Modified*, el contenido y formato de un ETag es completamente arbitrario. A pesar de que la mayoría de los navegadores web legítimos proveen ya sea una versión del archivo o un hash del recurso representado, no existe un requerimiento para hacerlo. Esto hace que los ETags sean útiles para almacenar un identificador y cookies regenerativas.

Esta técnica ha sido utilizada por diversos sitios web para el *tracking* de usuarios., con efectos persistentes incluso después de que el usuario elimine sus cookies. Un ejemplo de ello es el portal Hulu, que estaba siguiendo a sus usuarios almacenando información de identificación en ETags [11].

Cuando el valor del ETag es enviado al servidor que realiza el *tracking*, es manejado de forma similar a una cookie HTTP tradicional. A diferencia de las cookies normales, los ETags son más difíciles de falsificar en clientes HTTP estándar y por lo tanto puede asumirse que son más confiables. El valor puede ser eliminado con facilidad limpiando la caché del navegador web, y un usuario puede bloquear el seguimiento por medio de ETags completamente deshabilitando el caché o utilizando un cliente que no soporta funcionalidades de caché.

2.4.3 Almacenamiento web

El almacenamiento web o *web storage* es una solución de base de datos sencilla implementada por primera vez por Mozilla en Firefox 1.5 y eventualmente adoptada por la especificación HTML5. El diseño actual de *web storage* depende de dos objetos JavaScript simples: *localStorage* y *sessionStorage*. Ambos objetos proveen una API simple e idéntica para crear, recuperar y borrar pares de *nombre=valor* en la base de datos gestionada por un navegador web.

El objeto *localStorage* implementa un almacenamiento persistente de origen específico que sobrevive al apagado del navegador, mientras que *sessionStorage* está vinculado a la ventana abierta de un navegador y provee un mecanismo de caché temporal que es destruido al finalizar la sesión de navegación. A pesar de que la especificación indica que tanto *localStorage* como *sessionStorage* deben estar asociados a una tupla protocolo-host-puerto, las implementaciones en algunos navegadores no siguen este lineamiento, introduciendo potenciales vulnerabilidades. Un ejemplo de ello es el navegador Internet Explorer 8, donde el protocolo no es tomado en cuenta al procesar el origen [12].

2.4.4 Objetos locales compartidos

Los objetos locales compartidos o *Local shared objects* (LSO), también conocidos como cookies flash, son una recopilación de datos similar al de una cookie tradicional creada por un sitio web que ejecuta Adobe Flash. Las cookies flash sobrepasan las limitaciones

de las cookies HTTP, presentando un tamaño por defecto de 100 KB, a diferencia de los 4 KB de las cookies anteriores, y con la posibilidad de almacenarse en su propia carpeta dentro del terminal del usuario, ignorando la instrucción de borrado de cookies presente en los navegadores. Por lo tanto, pueden almacenar una mayor cantidad de información del usuario y acceder a múltiples navegadores en una terminal de usuario [6].

2.5 Técnicas de seguimiento basada en cookies

Las técnicas de seguimiento o *tracking* descritas a continuación hacen uso de diversos tipos de cookies, tales como:

2.5.1 Supercookies, evercookies o zombie cookies

Una supercookie puede ser definida como una cookie que se almacena en diferentes ubicaciones de respaldo afuera de su ubicación regular designada por la configuración del navegador web. Las supercookies también son llamadas como evercookies, en referencia a la implementación basada en JavaScript desarrollada por Samy Kamkar [13].

Las supercookies pueden ser utilizadas como un mecanismo de seguimiento adicional a las cookies HTTP tradicionales o de forma independiente como un mecanismo más resistente y difícil de detectar. También pueden ser empleadas para el respaldo de cookies HTTP, de forma que cuando se elimina una cookie de este tipo, estas pueden recrearse desde los lugares de respaldo disponibles. Este último método también es llamado “zombie cookie” [14].

Las ubicaciones de respaldo pueden incluir [15]:

- Cookies HTTP
- Cookies Flash
- Almacenamiento aislado de Silverlight
- Almacenamiento web
- Historial web
- El caché del navegador
- La propiedad DOM `window.name`

En general, suelen utilizarse los términos supercookies, evercookies y zombie cookies como sinónimos para referirse a las cookies caracterizadas por su persistencia en el terminal del usuario.

2.5.2 Cookies de origen

Dentro de cada cookie, se especifica un dominio que es el sitio web que establece una cookie en el cliente. Las cookies de origen o *First-party* son aquellas que se crean por el sitio web con el que el usuario interactúa directamente. Si un usuario visita un sitio web,

por ejemplo abc.com, el nombre de dominio que presentará la cookie será abc.com. El término “*First-party*” simplemente se refiere al origen de la cookie [16].

2.5.3 Cookies de terceros

Las cookies de terceros *Third-party* hacen referencia a cookies implantadas por sitios web distintos a los que el usuario decide interactuar. Si un usuario en particular visita un sitio web, por ejemplo, abc.com y las cookies de dominio almacenadas en su navegador son distintas a las del sitio abc.com, esa cookie se considera una cookie *Third-party*. Las cookies *Third-party* se establecen cuando una petición se redirige de un sitio web a un nombre de dominio que no corresponde al visitado por el usuario.

2.5.4 Sincronización de cookies

La sincronización de cookies es el proceso mediante el cual dos sitios web comparten un valor de identificación o ID, típicamente almacenado en cookies, otorgado a un usuario en particular [17]. En la figura 6 se ilustra el proceso:

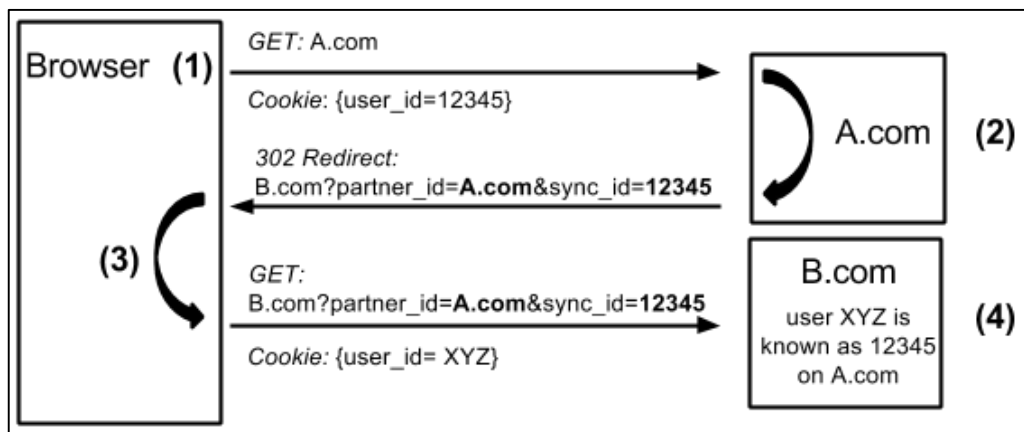


Figura 6. Proceso de sincronización de cookies [18].

Según la figura 6, la sincronización de cookies se desarrolla como sigue [18]:

- El proceso inicia cuando un usuario visita un sitio web en particular, que incluye a “A.com” como un *Third-party tracker*.
- El navegador realiza una petición a A.com e incluida en esta petición se encuentra una cookie de seguimiento establecida por A.com (1).
- A.com recoge la ID de la cookie, y redirige al navegador hacia “B.com”, codificando la ID dentro de la URL (2).
- El navegador posteriormente realiza una petición a B.com, que incluye la URL completa que A.com redirigió, así como la cookie de seguimiento de B.com (3).
- B.com puede entonces vincular la ID que asignó al usuario con la ID de A.com. Todo el proceso ocurre sin conocimiento por parte del usuario (4).

Una vez que dos dominios o sitios web han sincronizado cookies, estos pueden intercambiar datos del usuario a través de sus servidores. Estos datos pueden incluir desde historiales de navegación hasta información personalmente identificable.

2.6 Identificación de usuarios: extracción de huellas

Una huella o *fingerprint* es un conjunto de información relacionada al terminal o dispositivo del usuario, incluyendo desde el hardware hasta el sistema operativo, el navegador web y su configuración. Por ende, la extracción de huellas o *fingerprinting* se refiere al proceso de recolectar información a través de un navegador web para construir la huella de un dispositivo.

A diferencia de otros mecanismos de identificación y *tracking* como las cookies que requieren de un identificador único del usuario (ID) almacenado directamente dentro del navegador web (mecanismo dependiente del estado o *stateful*), la extracción de huellas es una técnica sin estado o *stateless*, ya que no deja ningún tipo de trazas y no requiere del almacenamiento de información en el dispositivo. Si la huella contiene suficiente información, es muy probable que la huella sea única para ese usuario, por lo que puede ser utilizada para identificarlo de forma unívoca por un sitio web a través de sus características expuestas [19].

Para recolectar la información del entorno del usuario se hace uso de dos tipos de mecanismos de extracción de huellas:

- *Fingerprinting* pasivo.
- *Fingerprinting* activo.

2.7 Extracción pasiva de huellas

En el *fingerprinting* pasivo los paquetes de datos recibidos en los servidores web son analizados. En este caso el comportamiento del servidor es similar al de un *sniffer*¹ y no se coloca ningún tipo de tráfico adicional en la red. La información extraída de los paquetes puede incluir direcciones IP a nivel de red y cabeceras HTTP nivel de aplicación. Recibe el nombre de pasivo ya que no implica la comunicación directa con el dispositivo sobre el cual se aplica el seguimiento.

2.8 Extracción activa de huellas

El *fingerprinting* activo es aquel que depende de la ejecución de scripts dentro del navegador para recolectar la información del usuario. A diferencia del método pasivo,

¹ Un *sniffer* es una herramienta que permite interceptar, capturar y analizar todos los paquetes que circulan a través de una red o parte de una red.

la extracción activa de huellas requiere transmitir paquetes de datos al usuario remoto desde el servidor web. Este tipo de *fingerprinting* se ilustra en la figura 7:

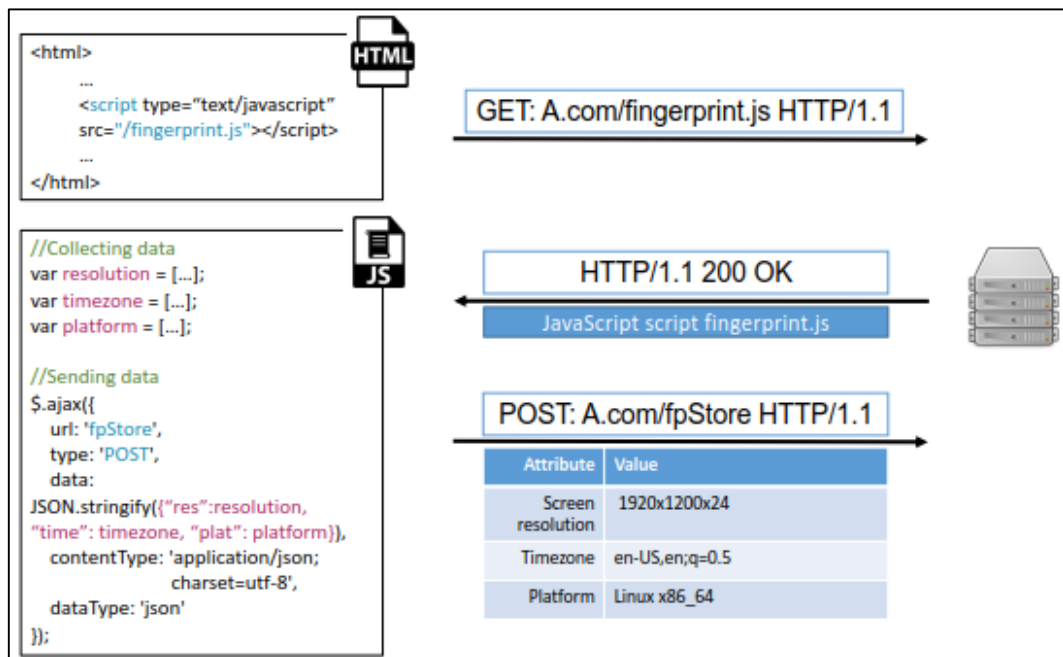


Figura 7. Ejemplo de *fingerprinting* activo [19].

En el ejemplo de la figura 7, el script “fingerprint.js” no se encuentra embebido en la página, por lo que el navegador intenta obtenerlo. Cuando el script es recibido, este se ejecuta directamente y recoge la información consultando diferentes componentes del navegador. Toda la información recogida se envía posteriormente al mismo servidor remoto.

Entre los principales mecanismos de *fingerprinting* activo se tienen los siguientes [20] [21]:

- Basado en JavaScript.
 - Canvas.
 - WebGL.
 - WebRTC.
 - Audio.
 - Batería.
 - Extensiones del navegador.
- Basado en plugins.
 - Lista de fuentes.

2.8.1 Basado en JavaScript

JavaScript es un lenguaje de scripting orientado a objetos y multiplataforma que ha sido utilizado para diferentes fines tales como contenidos web interactivos,

comunicaciones asíncronas (ej. Validación de campos de entrada sin la necesidad de refrescar la página por completo), creación de animaciones sobre la marcha, modificación de contenidos en documentos de forma dinámica, etc. [22].

JavaScript provee APIs para acceder a propiedades del navegador y del dispositivo que lo ejecuta, una característica usada para la extracción de huellas. Las APIs de JavaScript que han sido más utilizadas con esta finalidad son los objetos informativos *navigator* y *screen* [20].

Algunos elementos informativos obtenidos por medio de JavaScript incluyen:

- Lista de plugins.
- Cookies habilitadas (*cookies enabled*).
- Zona horaria.
- Resolución de pantalla
- Profundidad de color.

Entre los elementos que emplean APIs de JavaScript para la extracción de huellas de un usuario se tiene:

2.8.1.1 Canvas

Canvas es un elemento HTML5 originalmente desarrollado por Apple y es utilizado para dibujar gráficos (líneas, figuras, texto, imágenes) y animaciones (por ejemplo, juegos y publicidades estilo banner) en páginas web utilizando un API JavaScript. Las características de Canvas pueden ser explotadas por anunciantes para obtener información de los navegadores y perfilar a los usuarios.

Para extraer la huella de un usuario, primero se dibuja un contexto arbitrario (usualmente un pangrama o frase holoalfabética²) en un elemento Canvas. Posteriormente, se recoge los datos de imagen del elemento Canvas (píxeles), que dependen de las configuraciones de hardware y software del sistema (librerías de fuentes, hardware de gráficos y sistema operativo). Finalmente, se genera un hash con los datos recolectados. De esta forma se obtienen datos de imagen para diferentes combinaciones de navegadores y sistemas operativos, pudiendo usarse para la identificación unívoca de la plataforma del usuario [23].

2.8.1.2 WebGL.

WebGL es un API de gráficos desarrollado por Khronos Group que puede procesar objetos 3D interactivos en el navegador y manipularlos a través de JavaScript sin

² Frase que utiliza cada una de las letras existentes en un idioma dado. Ejemplo: El veloz murciélago hindú comía feliz cardillo y kiwi. La cigüeña tocaba el saxofón detrás del palenque de paja.

necesidad de plugins. Depende del elemento Canvas pero utiliza su propio contexto de renderizado (*WebGLRenderingContext*). Si una tarjeta de gráficos o GPU se encuentra disponible, el navegador puede hacer uso de la misma para procesar escenas 3D complejas. Si la renderización a través de hardware no está disponible, el navegador puede llamar al CPU con un renderizador de software como SwiftShader que emplea el navegador Chrome [19].

WebGL puede ser utilizado para la extracción de huellas de usuarios. Por un lado, se puede emplear como herramienta para probar las distintas posibilidades de renderizado según la GPU del dispositivo del usuario. Por otro lado, se puede obtener información sobre el nombre del fabricante y renderizador (GPU o CPU) que emplea WebGL a través de scripts.

2.8.1.3 WebRTC

WebRTC es un API para comunicaciones en tiempo real entre pares que permite a los navegadores realizar llamadas de voz o vídeo y compartir ficheros sin necesidad de plugins adicionales. WebRTC implementa STUN (Session Traversal Utilities for NAT) [24], un protocolo de red que permite a los hosts finales descubrir las direcciones IP de las interfaces locales de red (como ethernet o WiFi), así como las direcciones del lado público del NAT [25].

Una aplicación web puede acceder al conjunto de direcciones IP a través de JavaScript. La ventaja de emplear un filtrado de direcciones a través de WebRTC es que los resultados son precisos incluso si el usuario se conecta a Internet a través de una VPN o un proxy, por lo que no pueden ser falsificados con facilidad [26].

2.8.1.4 Audio

Es una técnica de *fingerprinting* cuyo funcionamiento se basa en la interfaz *AudioContext* del API Web Audio. Por medio de la interfaz mencionada, un script puede recopilar datos de la máquina del usuario, lo que potencialmente permite identificar al dispositivo. Existen incluso scripts más sofisticados que pueden procesar una señal de audio para generar una huella digital. Esta técnica de extracción de huellas usa el aspecto de las señales de audio en diferentes navegadores, pudiendo diferir de un sistema a otro por las distintas características de hardware y software de cada plataforma [21].

2.8.1.5 Batería

Esta técnica de *fingerprinting* utiliza el API de *Battery Status*, introducida con HTML5. Esta API permite a un sitio web consultar al navegador del usuario sobre el estado actual de la batería del dispositivo o su nivel de carga. Dado que los niveles de carga y tiempos

de descarga de una batería presentan un número elevado de estados, estos pueden ser utilizados como un identificador a corto plazo [25].

2.8.1.6 Extensiones del navegador

Las extensiones son programas escritos en JavaScript para añadir nuevas funcionalidades a un navegador. Estas extensiones pueden ser utilizadas para la extracción de huellas, ya que permiten la obtención de información del navegador del usuario.

Si una extensión, por ejemplo, añade un botón en el portal Youtube para proveer nuevos controles sobre un vídeo, el botón añadido es detectable por medio del análisis del DOM del sitio web (el Modelo en Objetos para la Representación de Documentos o DOM representa la estructura de la página web). La detección de un bloqueador de anuncios es similar ya que su función es evitar la aparición de publicidad y contenido emergente. Dado que la extensión modifica un elemento de la página que el usuario visita, este comportamiento es detectable y puede llevar a la identificación de la extensión instalada [27].

2.8.2 Basado en plugins

Un plugin es un componente de software que provee características adicionales al navegador web. Plugins como Flash, Java o Silverlight permiten crear, entregar y mostrar contenido multimedia interactivo en un navegador y proveen APIs para acceder a las propiedades de hardware y software, permitiendo a los desarrolladores obtener información sobre el sistema donde se ejecutan los plugins [28].

Una técnica importante de *fingerprinting* que hace uso de plugins es la detección de fuentes instaladas en el sistema.

2.8.2.1 Lista de fuentes

Técnica de *fingerprinting* relacionada con la obtención de las fuentes instaladas en el sistema y la forma como se presentan dentro de las aplicaciones. A diferencia de otros elementos, las fuentes dependen del sistema operativo por lo que pueden ser utilizadas para vincular a un usuario con diferentes navegadores web que se ejecutan en un mismo dispositivo [29].

Existen diversas maneras para obtener una lista de las fuentes instaladas en el sistema, pero las más usuales emplean plugins. Flash permite acceder a dicha lista por medio de una llamada a la función *Font.enumerateFonts*, mientras que un *applet* Java puede utilizar el método *getDisponibleFontFamilyNames*, contenido dentro de la clase *GraphicsEnvironment* del lenguaje Java. Silverlight puede proporcionar una lista de fuentes por medio de la propiedad *SystemTypefaces* de la clase *System.Windows.Media.Fonts* [14].

Es posible obtener una lista de fuentes por medio de una alternativa basada en JavaScript que consiste en probar cada fuente una por una, pero este método carece de la rapidez y del acceso directo provisto por los plugins mencionados [19].

2.9 Atributos de una huella

Según el proyecto Panopticlick de la Electronic Frontier Foundation (EFF), se ha demostrado que la combinación de distintos atributos del entorno del usuario puede dar lugar a una huella de alta precisión [30]. El algoritmo de Panopticlick originalmente identificó 10 atributos clave, a los cuales los desarrolladores de otra plataforma de *fingerprinting*, AmiUnique, agregaron 7 atributos adicionales. Los atributos más determinantes que forman parte de una huella según los dos proyectos mencionados se presentan en la tabla 3.

Tabla 3. Atributos más significativos en una huella según Panopticlick y AmiUnique [31].

#	Atributo	Mecanismo	Tipo	Ejemplo
1	User Agent	Cabecera HTTP	Pasivo	Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/41.0.2272.118 Safari/537.36
2	Cabecera Accept	Cabecera HTTP	Pasivo	text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
3	Codificación de contenido	Cabecera HTTP	Pasivo	gzip, deflate, sdch
4	Idioma del contenido	Cabecera HTTP	Pasivo	en-us,en;q=0.5
5	Lista de plugins	JavaScript	Activo	Plugin 1: Chrome PDF Viewer. Plugin 2: Chrome Remote Desktop Viewer. Plugin 3: Native Client. Plugin 4: Shockwave Flash...
6	<i>Cookies enabled</i>	JavaScript	Activo	Sí
7	Uso de almacenamiento local/sesión	JavaScript	Activo	Sí
8	Zona horaria	JavaScript	Activo	-60 (UTC+1)
9	Resolución de pantalla y profundidad de color	JavaScript	Activo	1920x1200x24
10	Lista de fuentes	JavaScript, Plugins	Activo	Abyssinica SIL,Aharoni CLM,AR PL UMing CN,AR PL UMing HK,AR PL UMing TW...
11	Lista de cabeceras HTTP	Cabeceras HTTP	Pasivo	Referer X-Forwarded-For Connection Accept Cookie Accept-Language Accept-Encoding User-Agent Host
12	Plataforma	JavaScript	Activo	Linux x86_64
13	<i>Do no Track</i>	JavaScript	Activo	Sí
14	Canvas	JavaScript	Activo	Cwm fjordbank glyphs vext quiz, 😊
15	Fabricante WebGL	JavaScript	Activo	NVIDIA Corporation
16	Renderizador WebGL	JavaScript	Activo	GeForce GTX 650 Ti/PCIe/SSE2
17	Uso de un bloqueador de anuncios	JavaScript	Activo	No

Los atributos del 1 al 10 de la tabla 3 fueron los utilizados originalmente en el algoritmo de Panopticlick, mientras que los atributos desde el número 11 al 17 fueron los incorporados por AmIUnique.

Para evaluar la importancia de los atributos en una huella, los desarrolladores de Panopticlick y AmIUnique calcularon los valores de entropía de cada atributo, para lo cual hicieron uso de la entropía de Shannon [32]. Sea H el valor de entropía, x una variable aleatoria discreta con posibles valores $x_1; \dots; x_i$ y $P(x)$ una función de probabilidad. La entropía se expresa a través de la siguiente ecuación:

$$H(X) = - \sum_{i=0}^n P(x_i) \log_b P(x_i) \quad (1)$$

En la ecuación de entropía de Shannon $b = 2$ y el resultado se expresa en bits. Un bit de entropía reduce a la mitad la probabilidad de ocurrencia de un evento, por lo que a medida que el valor de entropía es mayor, la huella será más única e identificable.

Por lo tanto, para calcular el valor de entropía de un atributo es necesario conocer la probabilidad de que dicho atributo presente un valor dado. Para ello es necesario almacenar los valores recolectados de una cantidad considerable de usuarios y analizarlos para obtener las proporciones necesarias para calcular la entropía.

Un ejemplo de cálculo de entropía se puede desarrollar con el atributo de zona horaria. Las distribuciones de los posibles valores del atributo obtenidas por AmIUnique en enero de 2018 de un total de 29.328 registros se presenta a continuación en la tabla 4.

Tabla 4. Estadísticas de zona horaria según AmIUnique en Enero de 2018 [33]

Valor	Uso (%)
UTC+1	29,5
UTC+0	12,5
UTC-2	12,4
UTC-5	5,9
UTC+3	4,2
UTC+2	3,9
UTC-8	3,3
Sin especificar	14,7
Otros	13,6

Con los valores de la tabla 4 se puede calcular la entropía de la siguiente manera:

$$H = -[0,295 \cdot \log_2 0,295 + 0,125 \cdot \log_2 0,125 + 0,124 \cdot \log_2 0,124 + 0,059 \cdot \log_2 0,059 + 0,042 \cdot \log_2 0,042 + 0,039 \cdot \log_2 0,039 + 0,033 \cdot \log_2 0,033] + 0,147 \cdot \log_2 0,147 + 0,136 \cdot \log_2 0,136$$

Obteniéndose el resultado de $H = 2,844$ bits.

Es posible comparar los valores de entropía de obtenidos por Panoptick y AmIUnique con valores de entropía normalizados. Para normalizar dichos valores se aplica la siguiente ecuación [34]:

$$\frac{H(X)}{H_M} \quad (2)$$

H_M representa el peor escenario donde el valor de entropía es máximo y todos los valores de un atributo son únicos. Siendo N el número de huellas o muestras en un conjunto de datos o *dataset*, H_M viene dado por [34]:

$$H_M = \log_2(N) \quad (3)$$

El estudio desarrollado por Gómez-Boix et al. [34] compara los valores de entropía normalizados (Norm.) para los atributos de Panoptick y AmIUnique. Además, agrega los valores de entropía obtenidos para estos mismos atributos en un *dataset* construido por los investigadores y conformado por atributos extraídos tanto de plataformas móviles (Android, iOS, Windows Phone) como de escritorio (Windows, MacOS, Linux). Los resultados se muestran a continuación en la tabla 5.

Tabla 5. Valores de entropía normalizados para los atributos en los conjuntos de datos de Panoptick, AmIUnique y Gómez-Boix et al. [34].

Atributo	Panoptick		AmIUnique		Dataset	
	Entropía	Norm.	Entropía	Norm.	Entropía	Norm.
Plataforma	-	-	2,310	0,137	1,200	0,057
Do no Track	-	-	0,944	0,056	1,919	0,091
Zona horaria	3,040	0,161	3,338	0,198	0,164	0,008
Lista de plugins	15,400	0,817	11,060	0,656	9,485	0,452
Uso de almacenamiento local/sesión	-	-	0,405	0,024	0,043	0,002
Uso de un bloqueador de anuncios	-	-	0,995	0,059	0,045	0,002
Fabricante WebGL	-	-	2,141	0,127	2,282	0,109
Renderizador WebGL	-	-	3,406	0,202	5,541	0,264
Lista de fuentes	13,900	0,738	8,379	0,497	6,904	0,329
Canvas	-	-	8,278	0,491	8,546	0,407
Cabecera Accept	-	-	1,383	0,082	0,729	0,035
Codificación de contenido	-	-	1,534	0,091	0,382	0,018
Idioma del contenido	-	-	5,918	0,351	2,716	0,129
User Agent	10,000	0,531	9,779	0,580	7,150	0,341
Resolución de pantalla y profundidad de color	4,830	0,256	4,889	0,290	4,847	0,231
Lista de cabeceras HTTP	-	-	4,198	0,249	1,783	0,085
Cookies enabled	0,353	0,019	0,253	0,015	0,000	0,000
H_M (peor escenario)	18,843		16,860		20,980	
Número de huellas	470.161		118.934		2.067.942	

Tomando en cuenta los valores normalizados de la tabla 5, los atributos con mayor entropía o que mejor permiten identificar de forma unívoca el entorno del usuario según los tres conjuntos de datos comparados son:

- Panopticklick:
 - Lista de plugins: 0,817.
 - Lista de fuentes: 0,738.
 - *User Agent*: 0,531.
 - Resolución de pantalla y profundidad de color: 0,256.
- AmIUnique:
 - Lista de plugins: 0,656.
 - *User Agent*: 0,580.
 - Lista de fuentes: 0,497.
 - Canvas: 0,491.
- Dataset de Gómez-Boix et al. [34]:
 - Lista de plugins: 0,452.
 - Canvas: 0,407.
 - *User Agent*: 0,341.
 - Lista de fuentes: 0,329.

Se observa que los atributos de lista de plugins, lista de fuentes y el agente de usuario o *user agent* aparecen en todos los conjuntos de datos entre los atributos con mayor valor de entropía. El atributo *Canvas*, que no fue evaluado en el dataset de Panopticklick, se encuentra entre los más identificadores según AmIUnique y el conjunto de datos de Gómez-Boix et al., ubicándose en cada uno con el cuarto y segundo mayor valor de entropía respectivamente.

De acuerdo Cao et al. [35], desarrolladores de otra plataforma para el estudio del *fingerprinting* llamada uniquemachine.org, AmIUnique es capaz de extraer huellas unívocas del 90,84% de los usuarios, con una entropía agregada para todos los atributos de 10,82. En el mismo estudio, Cao et al. alegan que su enfoque multi-navegador puede identificar hasta 99,24% de usuarios, demostrando que existen técnicas de *fingerprinting* todavía más eficientes.

2.10 Herramientas de evaluación de huellas

Se entiende por herramienta de evaluación de *fingerprinting* al conjunto de software que realiza visitas a sitios web, simulando la actividad de un navegador común, para llevar a cabo diversas verificaciones sobre los posibles mecanismos de seguimiento con estado y/o sin estado (*fingerprinting*) que pueden estar presentes.

Los componentes principales de una herramienta de evaluación de *fingerprinting* se muestran en la figura 8.

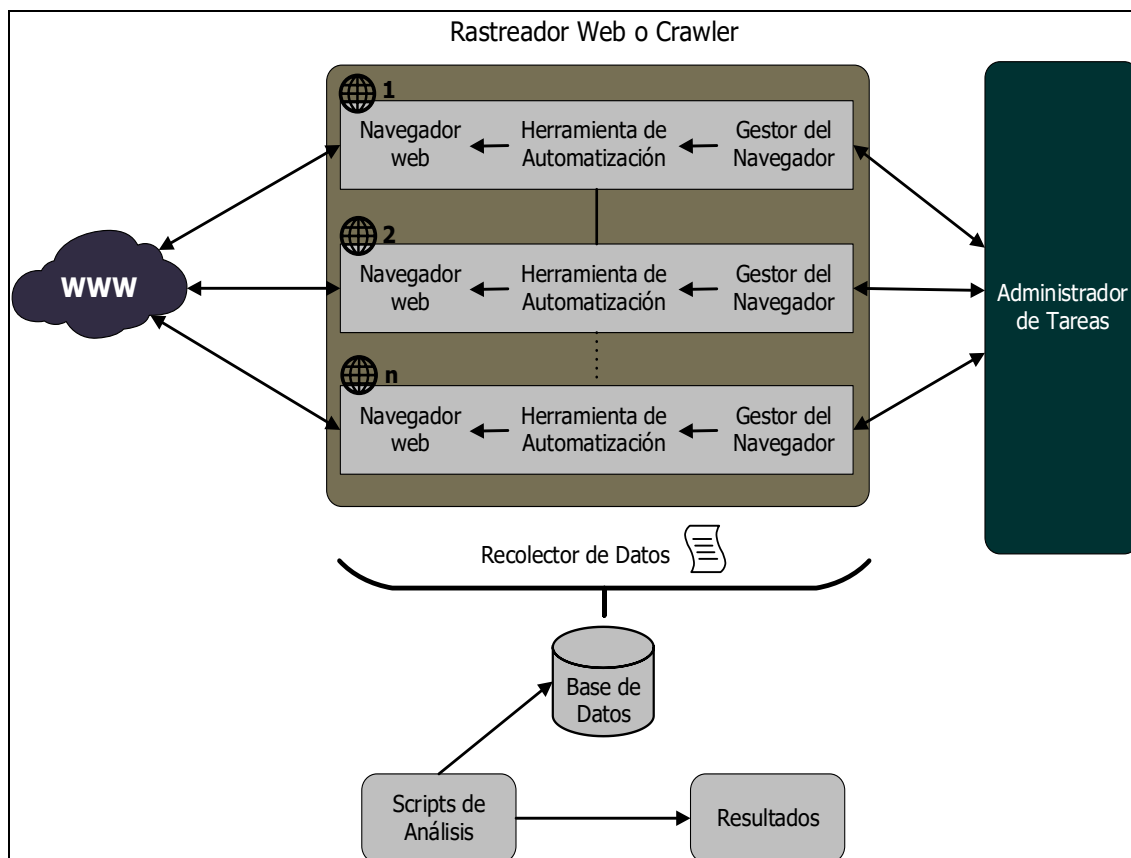


Figura 8. Ejemplo de arquitectura de una herramienta de evaluación de *fingerprinting*.

2.10.1 Rastreador web

También conocido como *crawler*, indexador o araña web, es un componente normalmente empleado en la construcción de colecciones de páginas web para motores de búsqueda. En el contexto de la extracción de huellas un rastreador web permite recolectar datos de eventos orientados a construir una huella del usuario, utilizando para ello un componente de navegación web ligero o personalizado según el tipo de exploración a realizar.

2.10.2 Instancias del rastreador web

Consta de tres elementos: Navegador web, herramienta de automatización y gestor del navegador. La ejecución de las distintas instrucciones para la detección de posibles eventos de extracción de huellas en el navegador web es coordinada por medio de un programa que permite la automatización de tareas tales como extracción, etiquetado y almacenamiento de los datos recolectados.

2.10.3 Administrador de tareas

Provee una interfaz para controlar distintas instancias del rastreador web al mismo tiempo, encargándose del manejo de errores y del envío de instrucciones predefinidas.

2.10.4 Recolector de datos

Este componente permite procesar, serializar y homogeneizar los registros obtenidos para su presentación. Dependiendo del tipo de herramienta, los resultados del *crawler* pueden consultarse a través del acceso a una base de datos o por medio de una interfaz destinada para tal fin.

Finalmente, las herramientas pueden ser compatibles con scripts u otros elementos de análisis para evaluar los registros con base a reglas establecidas, y determinar la presencia de técnicas particulares de *fingerprinting*.

2.10.5 Herramientas disponibles

Existen diversas herramientas disponibles públicamente, orientadas a investigadores y que han sido empleadas en otros estudios o proyectos, que funcionan con un modelo que comparte la mayoría de los componentes mostrados en la figura 8, tales como:

- FourthParty [36].
- FPDetective [37].
- OpenWPM [25].

Por otro lado, existen herramientas que proveen una funcionalidad similar y que presentan un modelo de funcionamiento más sencillo, ya que se encuentran alojadas en la web y pueden ser utilizadas de manera práctica por cualquier tipo de usuario sin necesidad de instalar software adicional. Ejemplos de este tipo de herramientas son:

- SecurityHeaders [38].
- Webbkoll [39].
- PrivacyScore [40].

Finalmente, existen extensiones disponibles para navegadores que permiten detectar eventos de *fingerprinting* y ofrecen información sobre los mecanismos utilizados por un sitio web para la obtención de los atributos. Como ejemplos se tiene:

- Chameleon [41].
- Don't FingerPrint Me (DFPM) [42].

A continuación, se presenta en la tabla 6 una comparación entre las distintas herramientas de evaluación de *fingerprinting* mencionadas.

Tabla 6. Comparativa entre herramientas de evaluación de *fingerprinting*.

Herramienta	Integración	Lenguajes	Licencia	Técnicas compatibles
FourthParty	<ul style="list-style-type: none"> • Mozilla Firefox: Mozilla Add-on SDK y extensión. • BD: SQLite. 	<ul style="list-style-type: none"> • JavaScript • Python 	GPLv3	<ul style="list-style-type: none"> • <i>Fingerprinting</i> pasivo: Peticiones y respuestas HTTP, cabeceras. • <i>Fingerprinting</i> activo: Llamadas JavaScript.
FPDetective	<ul style="list-style-type: none"> • <i>Crawler</i>: Chromium y PhantomJS. • Automatización: CasperJS, Selenium. • BD: MySQL. 	<ul style="list-style-type: none"> • JavaScript • Python • C++ 	GPLv3	<ul style="list-style-type: none"> • <i>Fingerprinting</i> pasivo: Peticiones y respuestas HTTP, cabeceras. • <i>Fingerprinting</i> activo: Llamadas JavaScript, fuentes cargadas/bloqueadas. • Almacenamiento: Local Shared Objects - LSO (Flash).
OpenWPM	<ul style="list-style-type: none"> • <i>Crawler</i>: Mozilla Firefox. • Automatización: Selenium. • BD: SQLite. 	<ul style="list-style-type: none"> • JavaScript • Python 	GPLv3	<ul style="list-style-type: none"> • <i>Fingerprinting</i> pasivo: Peticiones y respuestas HTTP, cabeceras. • <i>Fingerprinting</i> activo: Llamadas JavaScript. • Almacenamiento: LSO (Flash), cookies de origen, cookies de terceros, sincronización de cookies., evercookies.
SecurityHeaders	<ul style="list-style-type: none"> • Framework web: CodeIgniter. 	<ul style="list-style-type: none"> • PHP 	GPLv3	<ul style="list-style-type: none"> • <i>Fingerprinting</i> pasivo: Peticiones y respuestas HTTP, cabeceras.
Webbkoll	<ul style="list-style-type: none"> • Framework web: Phoenix. • Automatización: PhearJS (hace uso de PhantomJS). 	<ul style="list-style-type: none"> • Elixir 	Licencia MIT	<ul style="list-style-type: none"> • <i>Fingerprinting</i> pasivo: Peticiones y respuestas HTTP, cabeceras. • Almacenamiento: Cookies HTTP, cookies de origen, cookies de terceros, sincronización de cookies.
PrivacyScore	<ul style="list-style-type: none"> • Framework web: Django. 	<ul style="list-style-type: none"> • Python • JavaScript 	GPLv3	<ul style="list-style-type: none"> • <i>Fingerprinting</i> pasivo: Peticiones/respuestas HTTP, cabeceras. • Almacenamiento: LSO (Flash), Cookies de origen, cookies de terceros, sincronización de cookies.
Chameleon	<ul style="list-style-type: none"> • Extensión para navegadores web basados en Chromium. 	<ul style="list-style-type: none"> • JavaScript 	MPL 2.0	<ul style="list-style-type: none"> • <i>Fingerprinting</i> activo: Llamadas JavaScript.
DFPM	<ul style="list-style-type: none"> • Extensión para navegadores web basados en Chromium. 	<ul style="list-style-type: none"> • JavaScript 	Licencia MIT	<ul style="list-style-type: none"> • <i>Fingerprinting</i> activo: Llamadas JavaScript.

2.11 OpenWPM

Open Web Privacy Measurement (OpenWPM) es un *framework* para la medición de la privacidad en la web orientada a simplificar la recolección de datos en estudios donde se analizan sitios web en gran escala. La herramienta surgió de la necesidad de estandarizar las mediciones obtenidas en diversos estudios en este campo, de forma que pudieran ser comparables y contrastables [43]. OpenWPM fue desarrollado por

miembros del *Web Transparency & Accountability Project* de la Universidad de Princeton [44].

OpenWPM posee ventajas importantes con respecto a otras herramientas similares, entre las cuales destacan [25]:

- **Arquitectura:** OpenWPM está construido sobre Mozilla Firefox (navegador web usado por el componente rastreador o crawler) y provee la automatización a través del Selenium *web driver*. La interacción por parte del usuario se lleva a cabo a través del administrador de tareas, que provee un conjunto de instrucciones que permiten personalizar las instancias del rastreador o *crawler*.

A diferencia de otras herramientas, como aquellas alojadas en la web o basadas únicamente en extensiones para navegadores, OpenWPM presenta un nivel de personalización y flexibilidad con una arquitectura que facilita el desarrollo de componentes adicionales.

- **Lenguaje de desarrollo:** La necesidad de estandarización se refleja también en el lenguaje de desarrollo, ya que todo el *framework* de la herramienta se encuentra implementado en Python y emplea librerías de Python.
- **Escalabilidad:** OpenWPM se desarrolló para llevar a cabo operaciones de rastreo web con facilidad en una escala de miles a millones de sitios. Esto es posible ya que la herramienta permite usar una lista de URLs como entrada para las actividades del administrador de tareas. Otras herramientas mencionadas en este estudio no presentan una funcionalidad equivalente.
- **Relevancia:** OpenWPM se encuentra en continuo desarrollo. El último lanzamiento estable por parte de los autores fue la versión 0.8.0 el 9 de octubre de 2017 y actualmente existe una versión 0.9.0 en desarrollo. Aunado a los diversos cambios consolidados a lo largo del 2018 [45], se evidencia que la herramienta se mantiene operativa en entornos modernos, con librerías y dependencias actualizadas. Herramientas con arquitecturas similares como FourthParty y FPDetective, no reciben actualizaciones desde el año 2015.
- **Facilidad de instalación y uso:** OpenWPM se desarrolló y probó sobre Ubuntu 14.04/16.04. Junto con la herramienta se provee un script (*install.sh*) que permite instalar en el sistema todas las dependencias necesarias de forma automática. Después de instalar las dependencias, solo es necesario ejecutar un script Python (ej. *demo.py*) para iniciar el administrador de tareas [45].

2.12 Resumen y clasificación general de técnicas de seguimiento

Existen diversas técnicas que permiten realizar la identificación y seguimiento de usuarios en la web. Es posible agrupar las técnicas descritas en apartados anteriores en dos grandes grupos:

- Con estado (*stateful*).
- Sin estado (*stateless*).

En la figura 9 presentada a continuación se clasifican las técnicas en estos dos grupos.

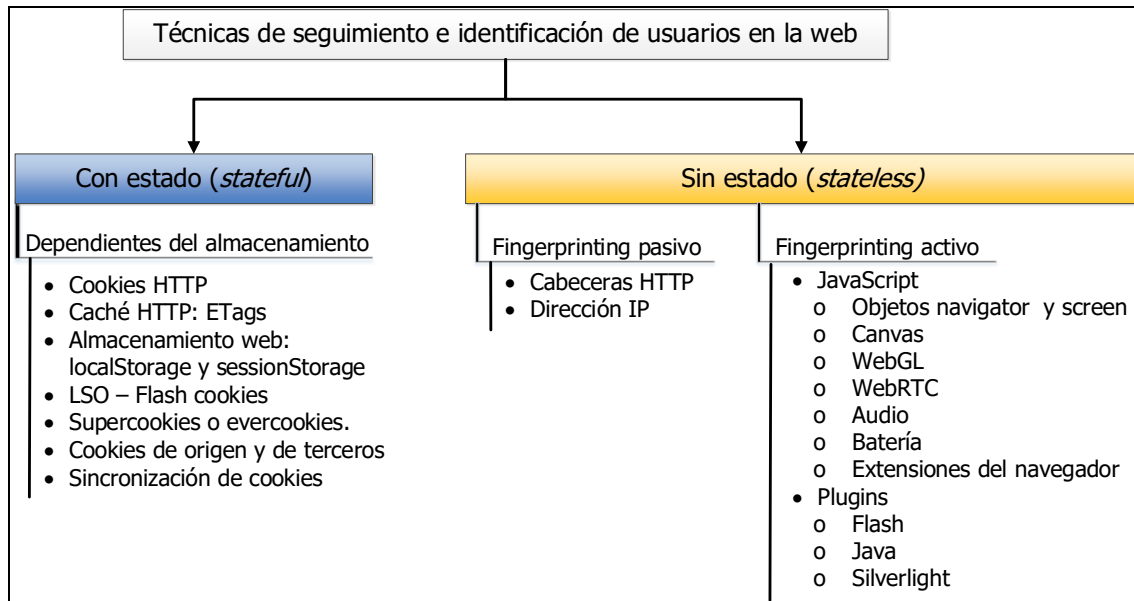


Figura 9. Clasificación de técnicas de seguimiento e identificación de usuarios en la web.

Para el presente estudio interesa delimitar el prototipo de aplicación para detección de técnicas de identificación y seguimiento de usuarios en la web en un grupo particular de la clasificación presentada. Considerando las herramientas disponibles que pueden servir de base para la aplicación (FPDetective, OpenWPM, Chameleon, DFPM, etc.), se puede observar que presentan una alta compatibilidad para trabajar con diversas técnicas de la categoría *stateless* o sin estado (*fingerprinting*). El prototipo de aplicación se define, entonces, en función de este último grupo, y específicamente en torno a técnicas de *fingerprinting* activo.

3. Análisis de técnicas de extracción de huellas en la web.

Existen diversas técnicas de *fingerprinting* presentes en la web. Como se mencionó previamente en el presente estudio, el grupo de técnicas que interesa seleccionar dentro de la delimitación de la aplicación prototipo son aquellas que figuran dentro de la categoría de *fingerprinting* activo. En el presente capítulo, se analizan dichas técnicas para seleccionar aquellas consideradas de mayor relevancia y determinar una metodología de análisis y detección según corresponda.

3.1 Selección de las técnicas de seguimiento a detectar.

Para la selección se toman en cuenta las siguientes condiciones:

- Presencia de la técnica en la web según estudios previos.
- Valor de entropía del atributo que permite extraer la técnica.

3.1.1 Presencia de la técnica en la web según estudios previos.

De acuerdo al estudio de Englehardt y Narayanan, que en Enero de 2016 analizaron el ranking Alexa³ para evaluar la presencia de *fingerprinting*, las técnicas de Canvas, Canvas para extracción de fuentes (Canvas-Fuentes) y WebRTC se presentaron con las proporciones mostradas en la tabla 7 a continuación:

Tabla 7. Prevalencia de técnicas de *fingerprinting* según Englehardt y Narayanan [25].

Intervalos	% de sitios		
	Canvas	Canvas-Fuentes	WebRTC
0 - 1.000	5,10%	2,50%	0,60%
1.000 - 10.000	3,91%	1,98%	0,42%
10.000 - 100.000	2,45%	0,86%	0,19%
100.000 - 1.000.000	1,31%	0,25%	0,06%

La técnica de Canvas orientada a la generación de gráficos presenta la mayor presencia en los sitios que conformaban el ranking Alexa para la fecha del estudio, seguido de Canvas aplicado a la evaluación de fuentes del sistema y luego WebRTC. Entre otras técnicas evaluadas, el estudio determinó que el *fingerprinting* basado en el procesamiento de señales de audio es muy poco utilizado.

Por otro lado, Khademi et al. [46] develaron que entre los primeros 10 mil sitios web del ranking Alexa en Julio de 2015, existían 101 sitios (1,010%) sospechosos de llevar a cabo la técnica de *fingerprinting* basada en los objetos informativos de JavaScript como *navigator* y *screen*.

Un caso particular en el grupo de *fingerprinting* activo es la técnica basada en el API JavaScript de *Battery Status*. Existe incertidumbre en lo que respecta a su presencia y

³ Conformado por 1 millón de sitios web que presentan el mayor tráfico de datos a nivel mundial en función de visitantes únicos y vistas o número de peticiones a un sitio [63].

evolución en la web, ya que en octubre de 2016 Mozilla ha eliminado el acceso a esta API en contenidos web a partir de la versión 52 de Firefox. Posterior a Mozilla, el acceso a esta API también se ha eliminado de WebKit, motor de renderizado de páginas web del navegador Safari de Apple [47]. En el caso de otros navegadores, la característica no se ha implementado (Edge) o la forma en que el API opera aunado a los permisos requeridos se encuentran bajo consideración (Chrome).

Por otro lado, las técnicas basadas en plugins dependen de la compatibilidad del navegador web con Flash, Java, Silverlight, entre otros. Además de la compatibilidad, plugins como Flash se están empleando con menor frecuencia en la web por lo que se han adoptado otras medidas, como aquellas basadas en JavaScript, para extraer atributos como la lista de fuentes [35].

3.1.2 Valor de entropía del atributo que permite extraer la técnica.

Teniendo en cuenta los valores de entropía de diversos atributos (Tabla 5), atributos como el *User Agent* y la lista de plugins se colocan de forma consistente entre aquellos que presentan los más altos valores de entropía. Estos atributos se pueden extraer a través de propiedades del objeto JavaScript *navigator*.

De igual forma, el atributo de Canvas presenta el cuarto mayor valor en el conjunto de datos de AmIUnique y el segundo mayor en el conjunto de datos de Gómez-Boix et al. [34]. La lista de fuentes es otro atributo que presenta un alto valor de entropía, ya que entra entre los cuatro mayores valores en los dos conjuntos de datos mencionados, así como en el de Panopticlick.

A pesar de que en los conjuntos de datos mencionados no se hace referencia a un atributo relacionado a técnicas de *fingerprinting* de audio, Englehardt y Narayanan [25] determinaron de una muestra de 713 huellas distintas que el atributo relacionado a una señal de audio presentaba una entropía de 5,4 bits, lo que equivale a un valor normalizado de 0,570.

Otro atributo de interés a considerar además de los incluidos en los conjuntos de datos mencionados es el de la dirección IP, específicamente aquellas asignadas en la parte pública de un router NAT, para la identificación de un dispositivo a través de *fingerprinting*. A pesar de que las direcciones IP puede obtenerse de forma pasiva, la técnica basada en WebRTC en el *fingerprinting* activo permite obtener este atributo. De acuerdo a Blakemore [48], la dirección IP pública presenta una entropía de 8,52 bits en una muestra de 531 navegadores, que corresponde con un valor normalizado de 0,941, indicando que es un dato altamente identificador con respecto a otros atributos.

No obstante, a pesar del alto valor de entropía de una dirección IP, se debe tomar en cuenta que es un atributo que cambia de forma regular y puede resultar deficiente para

identificar a un usuario a largo plazo. En este sentido, la dirección IP no suele emplearse como un identificador único y es usualmente combinado con atributos extraídos con otras técnicas [48].

Otros atributos ligados a técnicas de *fingerprinting* activo presentan valores menores de entropía. Por ejemplo, el atributo de “Uso de un bloqueador de anuncios” que se relaciona a la técnica basada en extensiones del navegador, presenta una entropía normalizada de 0,024 y 0,002 en los conjuntos de datos de AmIUnique y de Gómez-Boix et al. [34]. De igual forma, los atributos relacionados a WebGL como fabricante y renderizador, presenta valores de entropía comparativamente bajos. Según los dos datasets mencionados, fabricante WebGL presenta valores de 0,127 (AmIUnique) y 0,109 (Gómez-Boix et al.), mientras que renderizador WebGL obtuvo valores de 0,202 (AmIUnique) y 0,264 (Gómez-Boix et al.).

3.1.3 Técnicas elegidas.

Considerando las condiciones descritas, las técnicas elegidas para su detección a través de la aplicación prototipo son las siguientes: informativos (*navigator* y *screen*) de JavaScript, Canvas, Canvas-Fuentes y WebRTC. En la tabla 8 se presenta un resumen de las características de las técnicas mencionadas.

Tabla 8. Resumen de técnicas elegidas.

Técnica	Atributo relacionado	Entropía (norm.) del atributo	Prevalencia en la web
Objetos informativos (<i>navigator</i> y <i>screen</i>)	<i>User agent</i> , lista de plugins, resolución de pantalla, etc.	Lista de plugins (Mayor valor): <ul style="list-style-type: none"> • Panopticlick: 0,817 • AmIUnique: 0,656 • Gómez-Boix et al.: 0,452 <i>User Agent</i> (segundo mayor valor): <ul style="list-style-type: none"> • Panopticlick: 0,531 • AmIUnique: 0,580 • Gómez-Boix et al.: 0,341 	1,010% ^b
Canvas	Hash de los datos de imagen (Canvas)	<ul style="list-style-type: none"> • AmIUnique: 0,491 • Gómez-Boix et al.: 0,407 	3,91% ^b
Canvas-Fuentes	Lista de fuentes	<ul style="list-style-type: none"> • Panopticlick: 0,738 • AmIUnique: 0,497 • Gómez-Boix et al.: 0,329 	1,98% ^b
WebRTC	Direcciones IP	0,941 ^a	0,42% ^b
^a Según Blakemore [48]. ^b En relación a 10 mil de sitios web.			

3.2 Criterios para la elaboración de scripts de análisis.

En general, una metodología que permita la detección de una técnica de *fingerprinting* se basa en criterios tales como:

- Identificación de los diversos objetos JavaScript que uno o varios scripts de un sitio web pueden emplear para obtener datos o atributos del sistema y navegador web del usuario.

- Presencia de una manera inusual o sospechosa de emplear dichos objetos JavaScript (ej. Un objeto que se invoca un número elevado de veces dentro de un mismo script, pase de parámetros específicos a través de un objeto, etc.)
- Combinación de diferentes tipos de objetos JavaScript para obtener diversos datos, así como también para procesar o transformar atributos extraídos con anterioridad en el mismo script.

El razonamiento para determinar la presencia de una técnica de *fingerprinting* en un sitio web se encuentra relacionado a factores de cantidad y relevancia de los datos extraídos por medio de uno o varios scripts. Al verificar que todos los criterios descritos se cumplen, es decir, que una cantidad de llamadas a un objeto particular se han producido o que ciertas propiedades presentan valores mayores a un umbral específico, es posible filtrar y descartar scripts que tienen poca probabilidad de presentar la complejidad o contenido suficiente para extraer una huella significativa.

Con base a lo anterior, se definen los criterios específicos a considerar para la detección, a través de la aplicación prototipo, de las técnicas de *fingerprinting* activo seleccionadas. Los criterios se basan en metodologías empleadas en estudios previos.

3.2.1 Detección de la técnica basada en objetos informativos.

Para la detección de la técnica de *fingerprinting* que emplea los objetos informativos JavaScript de *navigator* y *screen*, se detectarán como sospechosos los sitios que realizan una o ambas de las prácticas definidas por Khademi et al. [46] de la siguiente manera:

1. El sitio web realiza solicita más del 80% de las propiedades de los objetos informativos. Considerando que OpenWPM permite instrumentar 20 propiedades distintas de los objetos informativos, 18 para *navigator* (*appName*, *appVersion*, *buildID*, *cookieEnabled*, *doNotTrack*, *geolocation*, *language*, *languages*, *onLine*, *oscpu*, *platform*, *product*, *productSub*, *userAgent*, *vendorSub*, *vendor*, *getBattery*) y 2 para *screen* (*pixelDepth*, *colorDepth*). Descartando la propiedad de *navigator getBattery* cuyo uso se encuentra orientado a una técnica de *fingerprinting* no seleccionada (batería), se tendrían 19 propiedades. Los sitios que cumplan el criterio deben solicitar más de 15 propiedades distintas.
2. El sitio web solicita todas las propiedades relacionadas a *navigator.plugins* y *navigator.mimeTypes*. En OpenWPM se recogen 5 propiedades distintas del primer objeto (*name*, *filename*, *description*, *version*, *length*) y 3 del segundo (*description*, *suffixes*, *type*).

En el estudio de Khademi et al. se asignaron hasta 3 niveles de sospecha según el número de accesos a las propiedades de los objetos informativos por un sitio web. En los

resultados se expone que la mayoría de los sitios web (57,4 %) presentó una cantidad de accesos menor al promedio (más de 5 propiedades del objeto *navigator* y más de 3 propiedades del objeto *screen*) [46].

Dentro de los criterios definidos para esta técnica, se establece como intento de *fingerprinting* a un acceso del 80% de las propiedades instrumentadas por OpenWPM de los objetos informativos. Esto se debe a que, en el estudio referenciado, este criterio arroja un nivel de sospecha de segundo nivel. El umbral máximo que determina un nivel 3 de sospecha se consigue cuando el sitio web solicita todas las propiedades de los objetos *navigator* y *screen*, así como de *navigator.plugins* y *navigator.mimeTypes*. En el caso de estas últimas, Khademi et al. indican que únicamente considerando estas dos propiedades es posible identificar una instancia en el conjunto de datos del estudio con un 90% de precisión y que el acceso a todas las propiedades sobre plugins y tipos MIME son evidencias características de intentos de *fingerprinting* [46].

3.2.2 Detección de la técnica basada en Canvas.

En la detección de la técnica de Canvas se recogen los criterios definidos por Englehardt y Narayanan [25]:

1. Las propiedades de alto y ancho del elemento Canvas no deben ser inferiores a 16 x 16 píxeles.
2. El texto debe anotarse en el Canvas con al menos dos colores o al menos 10 caracteres distintos.
3. El script no debe llamar los métodos *save*, *restore* o *addEventListener*.
4. El script debe extraer la imagen con *toDataURL* o con una llamada a *getImageData* que especifique un área de tamaño mínimo de 16 x 16 píxeles.

Los elementos Canvas con dimensiones inferiores a 16 x 16 píxeles o que presentan textos dibujados con menos colores de los indicados, presentan menos complejidad y por consiguiente no se consideran relevantes para ser empleados en la extracción de huellas. Según Englehardt y Narayanan, solo se encontraron 4 falsos negativos de 3.493 scripts identificados en 1 millón de sitios web [25].

En el anexo 1 del presente trabajo de investigación se presenta una comparativa a modo de ejemplo de los diferentes resultados que se pueden obtener con un script que aplica la técnica de Canvas en 3 navegadores web distintos ejecutados en el mismo sistema.

3.2.3 Detección de la técnica de fuentes basada en Canvas.

A pesar de que existen otras alternativas en JavaScript para extraer la lista de fuentes, se opta por detectar la técnica descrita por Englehardt y Narayanan [25] que emplea la

interfaz *CanvasRenderingContext2D*. En dicho estudio, esta técnica se identificó como “Canvas-Fuentes” y debe cumplir las dos condiciones descritas a continuación:

1. El script establece la propiedad *Font* de la interfaz *CanvasRenderingContext2D* en 50 valores distintos.
2. El script emplea el método *measureText* de dicha interfaz al menos 50 veces sobre una cadena de caracteres.

En otras palabras, se obtiene información significativa para una huella si el elemento Canvas emplea más de 50 tipos de fuentes distintas (propiedad *Font*) y realiza una medición del ancho del texto (*measureText*) por cada fuente probada. Según Englehardt y Narayanan, no se identificaron falsos negativos a través de estos criterios en el millón de sitios web analizados [25].

En el anexo 2 del presente trabajo de investigación se presenta una comparativa a modo de ejemplo de los diferentes resultados que se pueden obtener con un script que emplea el método *measureText* en 3 navegadores web distintos ejecutados en el mismo sistema.

3.2.4 Detección de la técnica basada en WebRTC.

Finalmente, para la técnica basada en WebRTC se adaptan los siguientes criterios, basados en los desarrollados por Englehardt y Narayanan [25]:

1. El script emplea las siguientes APIs pertenecientes a *RTCPeerConnection*: *createDataChannel*, *createOffer*, y *onIceCandidate*.
2. El script extrae direcciones IP válidas por medio de los objetos mencionados en el criterio anterior.
3. El script debe emplear las direcciones IP extraídas en un contexto de seguimiento. Específicamente, el script debe contener alguna otra técnica de *fingerprinting* (Objetos informativos, Canvas o Canvas-Fuentes).

Englehardt y Narayanan identificaron a través de los criterios desarrollados en su estudio que un total de 715 sitios web del millón del Alexa obtenían direcciones IP sin interacción por parte del usuario [25].

Los criterios de detección definidos para cada técnica de *fingerprinting* constituyen la base del diseño y desarrollo del prototipo de aplicación, específicamente, del componente de análisis que permite, sobre la base de los datos recolectados por una herramienta que actúe como rastreador web o *crawler*, distinguir y seleccionar a los sitios que emplean scripts con la finalidad de extraer huellas de los usuarios.

4. Desarrollo de la herramienta de detección

En este apartado se diseña y desarrolla la aplicación prototipo para detección de técnicas de *fingerprinting* activo. Se parte de la selección, aplicación y modificación de una de las herramientas presentadas con anterioridad en el presente estudio. Posteriormente, con base al conjunto de técnicas seleccionadas para su detección, se implementan los distintos componentes de la aplicación prototipo sobre la base de la herramienta seleccionada.

4.1 Decisiones de diseño.

Para el desarrollo del prototipo de aplicación se plantea utilizar como base una herramienta de evaluación de *fingerprinting* existente, para luego complementar e incorporar las funcionalidades de detección deseadas en función de las características de la herramienta.

De las herramientas que asisten en la evaluación de técnicas de fingerprinting, se selecciona a OpenWPM como base para el desarrollo de la aplicación prototipo.

4.2 Diseño.

La aplicación prototipo abarca el uso de OpenWPM como base, además de diversos componentes que permitan la detección y obtención de resultados de técnicas de *fingerprinting* activo. Una arquitectura de alto nivel se muestra en la figura 10.

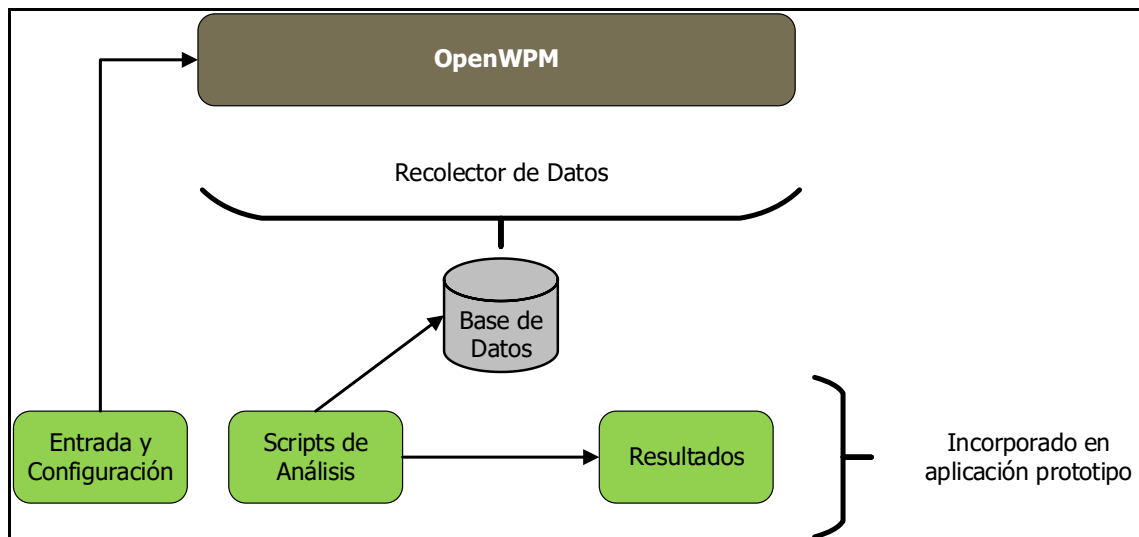


Figura 10. Detalle de la aplicación prototipo según arquitectura.

El recolector de datos almacena la información obtenida de las distintas instancias del *crawler* web de OpenWPM en una estructura relacional. La herramienta utiliza SQLite como sistema de gestión de base de datos, lo cual tiene como principal ventaja el bajo consumo de recursos del sistema. Sin embargo, existe la desventaja de que el sistema de gestión de base de datos es limitado con respecto a las funciones soportadas [43].

Partiendo de la arquitectura (Fig. 10) bajo la cual se adapta el *framework* de OpenWPM (rastreador web, administrador de tareas y recolector de datos), es necesario una aplicación que permita suplir la desventaja en el sistema de gestión de la base de datos y analizar con el detalle requerido los resultados obtenidos de la herramienta.

En este contexto, se desarrolla un componente de análisis para detectar la presencia de técnicas de *fingerprinting* activo según criterios predefinidos, y mostrar los resultados finales detallando los sitios web que se ajustan a los patrones que caracterizan a una técnica en particular.

La base de datos generada por OpenWPM y sobre la cual se aplica el componente de análisis que brinda la aplicación prototipo contiene tablas y atributos tales como [44]:

- **Crawl:** Opciones del navegador, tiempo de inicio, tiempo de competencia.
- **Peticiones HTTP:** *Referrer*, cabeceras, métodos, marca temporal, URL del sitio visitado.
- **Respuestas HTTP:** *Referrer*, cabeceras, métodos, estado, localización (para redirecciones), marca temporal, URL del sitio visitado.
- **Cookies HTTP:** Dominio, nombre, valor, expiración, tiempo de acceso.
- **Cookies flash:** URL del sitio web, nombre del fichero, clave, contenido.
- **localStorage:** Alcance, valor.
- **Historial del Crawl:** Comandos, argumentos, estados de éxito, marcas temporales.
- **JavaScript:** URL del script, funciones, valores, argumentos, operaciones, marcas temporales.

Dependiendo de la configuración, OpenWPM puede agregar más tablas a la base de datos. Una vista completa de las diferentes tablas y sus respectivos atributos se presenta en el anexo 3.

En general, un diagrama de entidad-relación de la base de datos generada por OpenWPM se muestra en la figura 11. En el diagrama se representan las siguientes entidades:

- *profile_cookies*
- *flash_cookies*
- *localStorage*
- *site_visits*
- *CrawlHistory*
- *crawl*
- *task*.

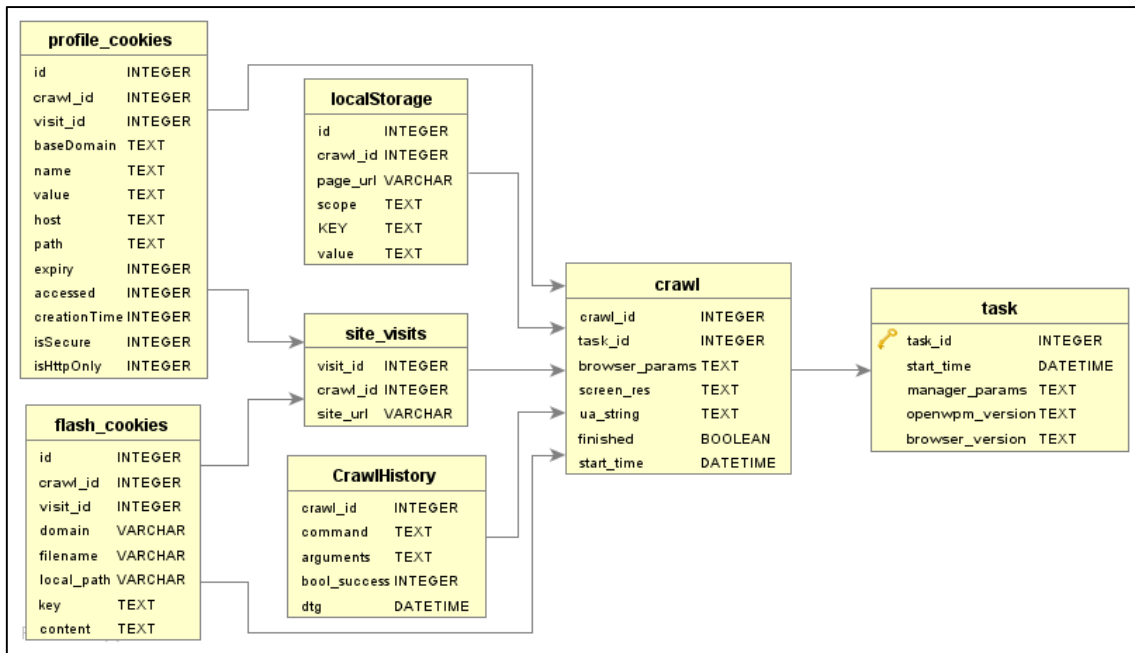


Figura 11. Diagrama de entidad-relación de la base de datos generada por OpenWPM.

Con base al modelo relacional presentado, se destacan las claves primarias y claves foráneas o ajenas descritas a continuación:

- **id**: Identificador único que se asigna a cada registro en la base de datos.
- **crawl_id**: Identificador que se asigna a la instancia del rastreador o *crawler*. Por ejemplo, si se configuran 3 instancias en el administrador de tareas, los valores posibles de *crawl_id* serán 1, 2 y 3.
- **visit_id**: Identificador que se asigna a cada sitio web visitado, por cada instancia del rastreador o *crawler*. Por ejemplo, si se indicó una lista de 10 URLs y se configuraron 3 instancias, los valores posibles de *visit_id* estarán comprendidos entre 1 y 30 inclusive.

Es importante conocer el modelo relacional de la base de datos ya que una consulta a través del sistema de gestión puede ayudar en filtrar y manipular los datos antes de aplicarles el componente de análisis.

Siguiendo las sugerencias de la documentación de OpenWPM [44], el componente de análisis se conformaría de scripts desarrollados en Python que permiten ejecutar una consulta a la base de datos y evaluar los distintos registros según los criterios que se definan en relación a una técnica de *fingerprinting* en particular.

4.3 Implementación de scripts de análisis y mejoras adicionales.

Basado en los criterios de detección definidos y en las características de los componentes de OpenWPM, se implementaron en Python los siguientes scripts de análisis:

- Objetos informativos.
- Canvas.
- Canvas-Fuentes.
- WebRTC.

Además de los scripts desarrollados para ser usados con los resultados de OpenWPM, se implementó una mejora en el script que permite inicializar las distintas instancias del *crawler*. Dicha mejora se encuentra orientada a una mejor gestión de los sitios evaluados, permitiendo proporcionar a la aplicación prototipo una lista de sitios en formato CSV [49].

4.4 Presentación de la aplicación prototipo.

Para integrar los componentes desarrollados con la herramienta OpenWPM en una forma práctica e intuitiva, se desarrolló una interfaz interactiva a través de la herramienta Jupyter Notebook.

Por consiguiente, se implementó la aplicación prototipo sobre un entorno de Jupyter Notebook donde se guía al usuario a través de tres pasos:

- Ingresar la lista de sitios web.
- Iniciar el rastreo o *crawling* de sitios web.
- Mostrar los resultados por técnica de *fingerprinting*.

En el anexo 4 se incluyen capturas de pantalla de la aplicación prototipo, donde se pueden observar los pasos descritos. El código fuente de la aplicación prototipo se encuentra disponible en GitHub [50].

4.5 Extensión de la aplicación prototipo.

Además de los scripts desarrollados para la detección de *fingerprinting* por medio de objetos informativos JavaScript, Canvas, Canvas-Fuentes y WebRTC, es posible extender la herramienta para evaluar nuevas técnicas o métodos de seguimiento de usuarios en la web. Un marco de trabajo a considerar para ello es el siguiente:

- Desarrollar un análisis de los criterios (ej. Número de llamadas a un objeto JavaScript, valores relacionados a una propiedad o interfaz, etc.) que permitan detectar un comportamiento típico de *fingerprinting*. Los lineamientos para describir tales criterios se han planteado previamente en la presente investigación.
- Conocer la estructura bajo la cual OpenWPM almacena los datos, lo que permite desarrollar los scripts de análisis en función de las entidades y sus respectivas relaciones en la base de datos.

- En caso de que OpenWPM no implemente una forma de almacenar los parámetros de un objeto JavaScript asociado a una nueva técnica de *fingerprinting* a analizar, modificar e incorporar directamente sobre OpenWPM el mecanismo para instrumentar tal objeto [51].
- Implementar los scripts de análisis desarrollados para las nuevas técnicas a implementar, mediante su incorporación en el entorno de Jupyter Notebook empleado para presentar el prototipo de aplicación.

Un mayor detalle de los temas de extensibilidad e implementación de nuevos scripts de análisis en la aplicación se presente en el anexo 5, un manual de desarrollo.

5. Validación y evaluación de resultados

Habiendo obtenido la versión prototipo de la herramienta de detección de técnicas de identificación y seguimiento de usuarios en la web, el presente capítulo se presenta con el interés de validar el funcionamiento de dicha aplicación. Para ello, se procede a llevar a cabo un conjunto de pruebas que permiten demostrar la existencia de las técnicas de *fingerprinting* activo caracterizadas en sitios web reales.

5.1 Definición de pruebas.

Para validar al prototipo de aplicación se definió un escenario de pruebas enmarcado dentro de los siguientes puntos:

- Detección de cuatro técnicas de *fingerprinting* activo: Se emplean todos los scripts basados en los criterios de detección desarrollados para las técnicas seleccionadas (Objetos informativos, Canvas, Canvas-Fuentes y WebRTC). Se deben obtener resultados para todas las técnicas referidas.
- Conjunto de datos: Los conjuntos de datos empleados por la aplicación se basan en las listas de ranking de sitios web disponibles públicamente. Se utilizan dos listas distintas:
 - Alexa.
 - Majestic Million⁴.

Considerando que ambas emplean criterios distintos para la construcción del ranking, existen diferencias en los sitios web que conforman ambos conjuntos.

- Relevancia de los datos: A su vez, con base a los rankings mencionados, se realizó una lista para cada uno conformada por los primeros 1.000 sitios web con el dominio de nivel superior geográfico “.es” correspondiente a España.

En cuanto a los resultados, interesa evaluar la relevancia de los mismos en función de los siguientes criterios:

- Proporción de sitios web que emplean alguna técnica de *fingerprinting*,
- Proporción de cada técnica dentro del conjunto de sitios que aplican *fingerprinting*.
- Categoría de los sitios web: Se evalúa la categoría o tipo de sitio web (ej. Universidades y centros de formación, banca, noticias y medios, etc.) donde se detecte la presencia de una técnica de *fingerprinting*.

⁴ Conformado por 1 millón de sitios web ordenados en función del número de subredes que contienen referencias (enlaces) a un dominio en la web [64].

5.2 Desarrollo de pruebas.

Se procede a ejecutar las pruebas bajo los criterios definidos. Para ello, se desarrollan los siguientes pasos:

- Preparación de los conjuntos de datos: Se emplearon los datos disponibles públicamente de Alexa y Majestic Million de fecha 15 de mayo de 2018. Las listas de sitios web generadas con base a los servicios mencionados se cargan en la aplicación prototipo en formato CSV.
- Ejecución de OpenWPM: Para cada lista se ejecuta OpenWPM para llevar a cabo el rastreo web o *crawling*. En las configuraciones de la herramienta se debe habilitar la instrumentación de los objetos JavaScript, con la finalidad de almacenar en la base de datos los elementos relacionados a técnicas de *fingerprinting* activo. Los detalles de la configuración, incluyendo números de versión de la herramienta (OpenWPM) y del navegador (Mozilla Firefox), número de instancias ejecutadas, y estado de otros parámetros además del mencionado anteriormente, se presenta en el anexo 6.
- Resultados: Se verifica para ambos casos que se obtiene el fichero de base de datos en formato .sqlite y se procede, por medio de la ejecución de los distintos scripts de análisis, a obtener las listas de sitios que cumplen con los criterios de *fingerprinting* asociados a las técnicas de objetos informativos de JavaScript, Canvas, Canvas-Fuentes y WebRTC.

5.3 Análisis de resultados.

Los resultados se presentan a continuación, para cada uno de los conjuntos de sitios web evaluados en las pruebas.

5.3.1 Resultados del conjunto de sitios web con base al ranking Alexa.

De los 1.000 sitios web rastreados, se detectaron 55 (5,5%) dominios únicos que emplean, al menos, una técnica de *fingerprinting* activo. Sin embargo, el número de intentos de *fingerprinting* es de 102, comprobándose que ciertos sitios web:

- Aplican 2 o más técnicas de *fingerprinting* distintas.
- Aplican una técnica en particular más de una vez.

Un resumen de los resultados obtenidos de las pruebas aplicadas sobre el conjunto de datos de Alexa se presenta en la tabla 9.

Tabla 9. Resumen de resultados - Alexa.

Número de sitios web rastreados	1.000
Número de sitios web que emplean <i>fingerprinting</i>	55 (5,5%)
Número de intentos de <i>fingerprinting</i>	102
Intentos por Canvas	46 (45%)
Intentos por Objetos Informativos	33 (32%)
Intentos por WebRTC	14 (14%)
Intentos por Canvas-Fuentes	9 (9%)

De las técnicas de fingerprinting analizadas, Canvas se presentó con la mayor frecuencia, siendo utilizada en el 45% de los intentos de *fingerprinting*. A Canvas le siguen objetos informativos, WebRTC y, por último, Canvas-Fuentes, como se muestra en el gráfico de tarta de la figura 12.

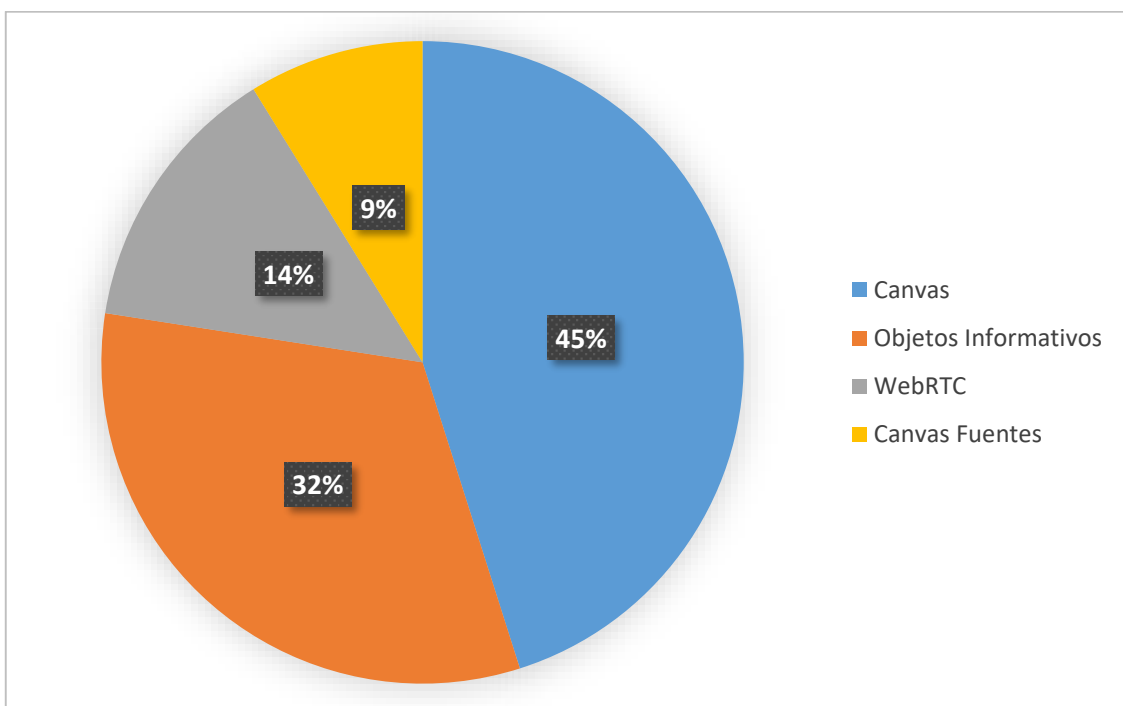


Figura 12. Proporción de técnicas de *fingerprinting* - Alexa.

Por otro lado, también se recolectaron las URL de origen de los scripts cargados por los sitios web para aplicar alguna técnica de *fingerprinting*. Las URL obtenidas permiten comprobar lo siguiente:

- Los sitios web pueden emplear scripts provenientes de terceros.
- Un mismo script puede ser empleado por diversos sitios web.
- Un script puede aplicar más de una técnica de *fingerprinting*.

En el gráfico de barras de la figura 13, se pueden observar la cantidad de técnicas asociadas a un URL de un script. En este gráfico solo se muestran los URL de los scripts que aplican 2 o más técnicas.

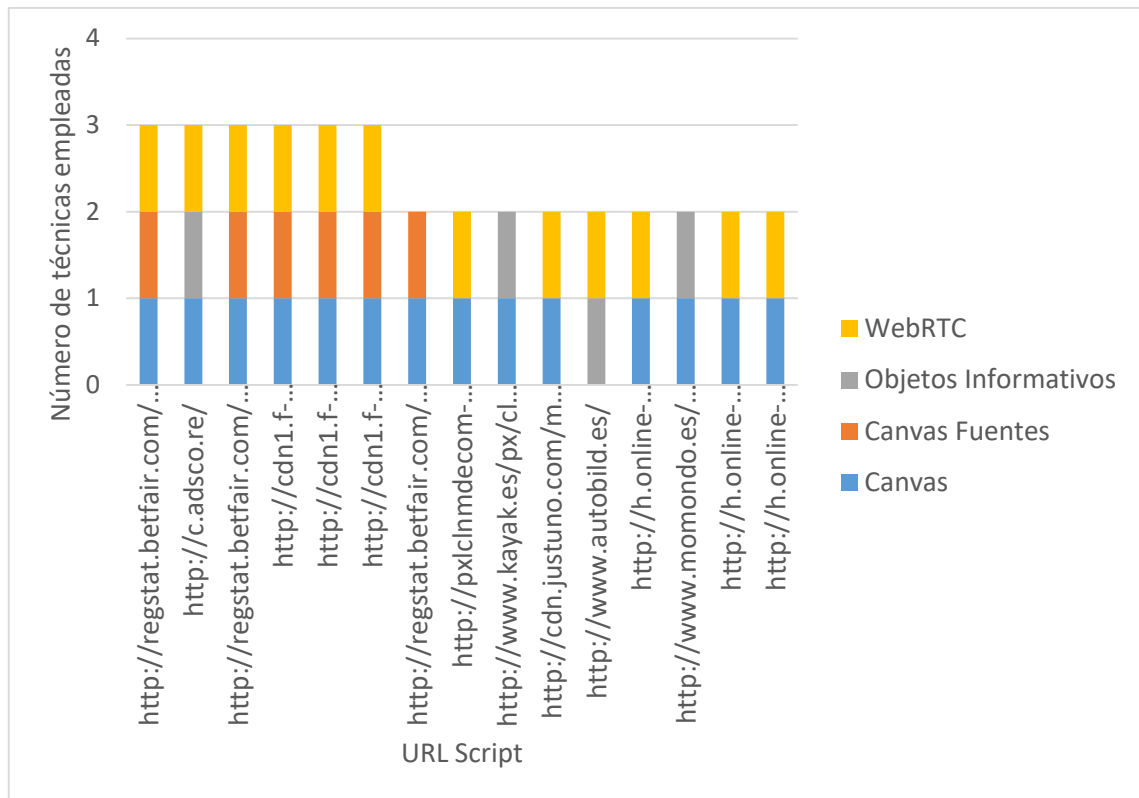


Figura 13. Número de técnicas empleadas por script - Alexa.

Se destacan los siguientes scripts:

- `check.js`: Encontrado en las URL que contienen “`http://regstat.betfair.com`” y “`http://cdn1.f-cdn.com`”. El script aplica 3 técnicas de *fingerprinting* distintas (Canvas, Canvas-Fuentes y WebRTC). De igual forma, existen otras versiones de “`check.js`” que aplican solo 2 técnicas (Canvas y Canvas-Fuentes). Por ejemplo, la versión de “`check.js`” encontrada en “`http://h.online-metrix.net/`” aplica las técnicas de Canvas y WebRTC.
- `main.min.js` y `mwgt_3.6.js`: Aplican 2 técnicas de *fingerprinting*. El primero, “`main.min.js`”, puede encontrarse en la URL que inicia por “`http://www.kayak.es/`” y aplica Canvas junto a objetos informativos. El segundo, “`mwgt_3.6.js`”, en la URL que contiene “`http://cdn.justuno.com/`” y aplica Canvas con WebRTC.

El código encontrado en los scripts mencionados se encuentra ofuscado, lo que dificulta su lectura y revisión. Además, existen casos donde el código JavaScript se encuentra embebido con el resto del código HTML del sitio web.

Por último, desde la perspectiva de las categorías dentro de las que se pueden clasificar a los sitios web, en el gráfico de barras de la figura 14 se presenta el número de sitios que aplican al menos una técnica de *fingerprinting* en función de tales categorías.

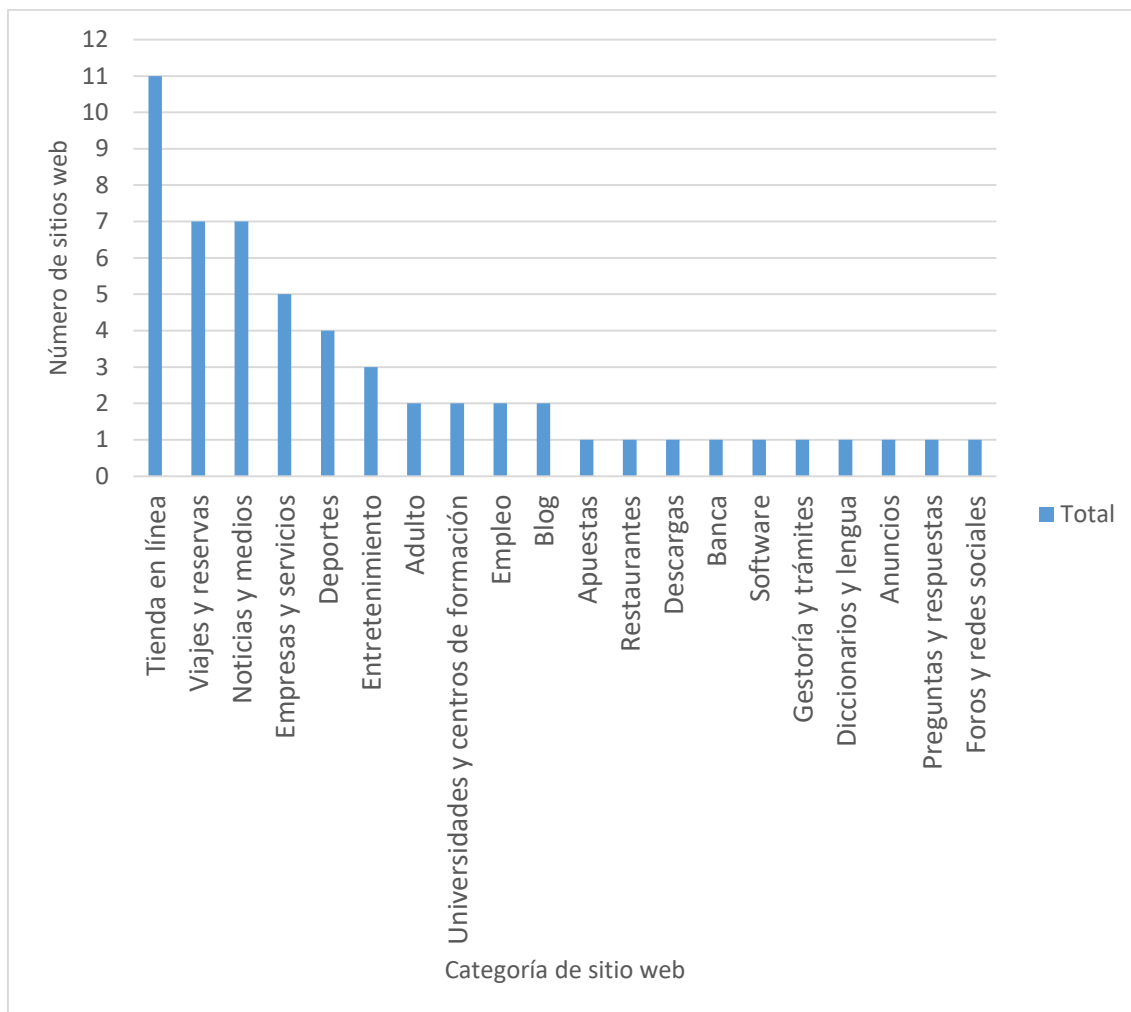


Figura 14. Número de sitios web que emplean técnicas de *fingerprinting*, por categoría – Alexa.

Las categorías de “Tienda en línea” (11 sitios) “Viajes y reservas” (7 sitios) y “Noticias y medios” (7 sitios), presentan la mayor cantidad de sitios web que emplean *fingerprinting*. Otras categorías que se destacan son “Universidades y centros de formación” (2 sitios) y “Banca” (1 sitio), donde la presencia de *fingerprinting* puede considerarse como inusual por motivos de condiciones de uso, privacidad, entre otros.

5.3.2 Resultados del conjunto de sitios web con base al ranking Majestic Million.

En lo que respecta a los 1.000 sitios web con dominio .es pertenecientes al ranking de Majestic Million, se detectaron que 26 (2,6%) dominios únicos emplean alguna técnica de *fingerprinting* activo. Por otro lado, el número de intentos de *fingerprinting* es de 34.

Tanto el número de sitios web únicos como de intentos de *fingerprinting* son menores en comparación a los obtenidos en el conjunto de sitios web basado en el ranking Alexa. No obstante, la observación realizada sobre la diferencia entre ambos indicadores también es aplicable en el caso del conjunto de Majestic Million.

Un resumen de los resultados obtenidos de las pruebas aplicadas sobre el conjunto de datos de Alexa se presenta en la tabla 10.

Tabla 10. Resumen de resultados - Majestic Million.

Número de sitios web rastreados	1.000
Número de sitios web que emplean <i>fingerprinting</i>	26 (2,6%)
Número de intentos de <i>fingerprinting</i>	34
Intentos por Canvas	16 (47%)
Intentos por Objetos Informativos	13 (38%)
Intentos por WebRTC	3 (9%)
Intentos por Canvas-Fuentes	2 (6%)

De igual forma, también se puede comprobar que la proporción en la que se presentan las técnicas de *fingerprinting* es similar al caso de Alexa, siendo la técnica más empleada la de Canvas en 47% de los intentos de *fingerprinting*. En orden de uso, le siguen objetos informativos, WebRTC y Canvas-Fuentes.

En el gráfico de tarta de la figura 15 se detallan las proporciones en que se presentaron las técnicas analizadas y encontradas.

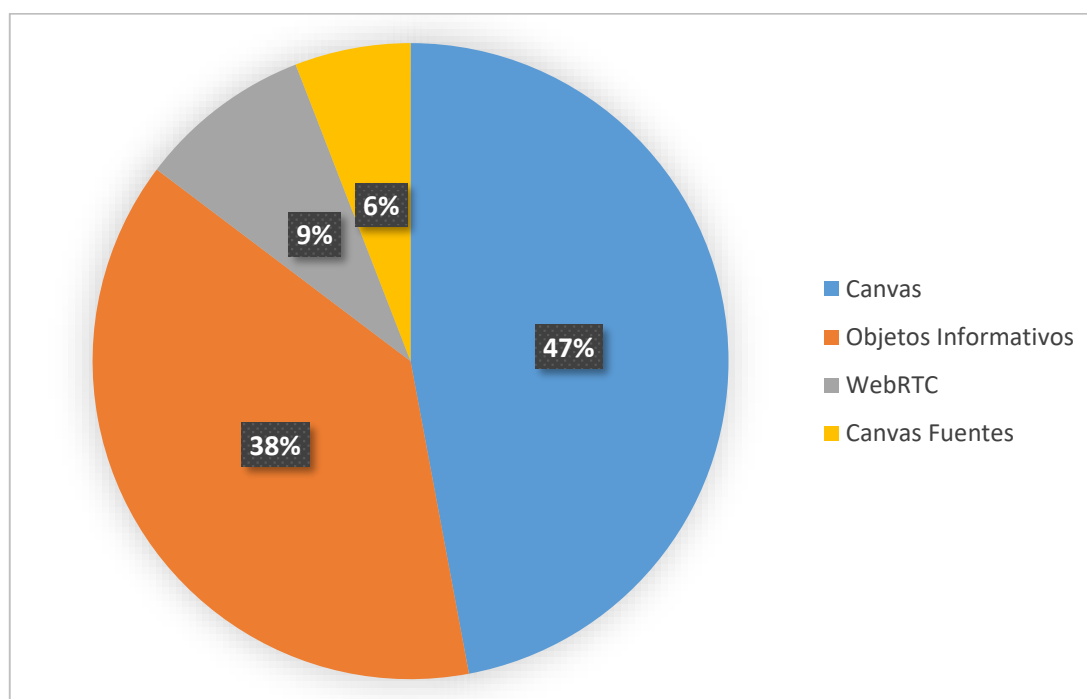


Figura 15. Proporción de técnicas de *fingerprinting* - Majestic Million.

Siguiendo con el análisis de la cantidad de técnicas de *fingerprinting* que pueden estar contenidas en un script, en el gráfico de barras de la figura 16 se pueden observar los URL de los ficheros JavaScript cuyo código implementa 2 o más técnicas.

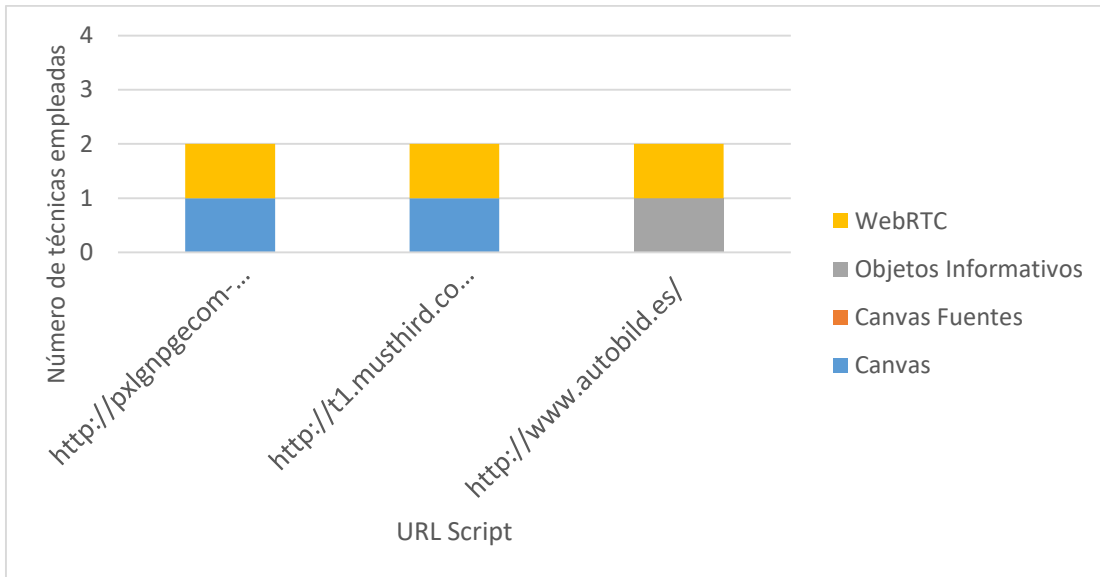


Figura 16. Número de técnicas empleadas por script - Majestic Million.

El script “check.js” vuelve a detectarse, en este caso en la URL que inicia por “http://t1.musthird.com”, abarcando las técnicas de Canvas y WebRTC. Aplicando las mismas técnicas, aparece un nuevo script llamado “browserfp.min.js”, el cual se encuentran en la URL que contiene “http://pxlgnpgecom-a.akamaihd.net/”.

En cuanto al número de sitios web que aplican *fingerprinting* en función de su categoría o clasificación, se presenta el gráfico de barras de la figura 17.

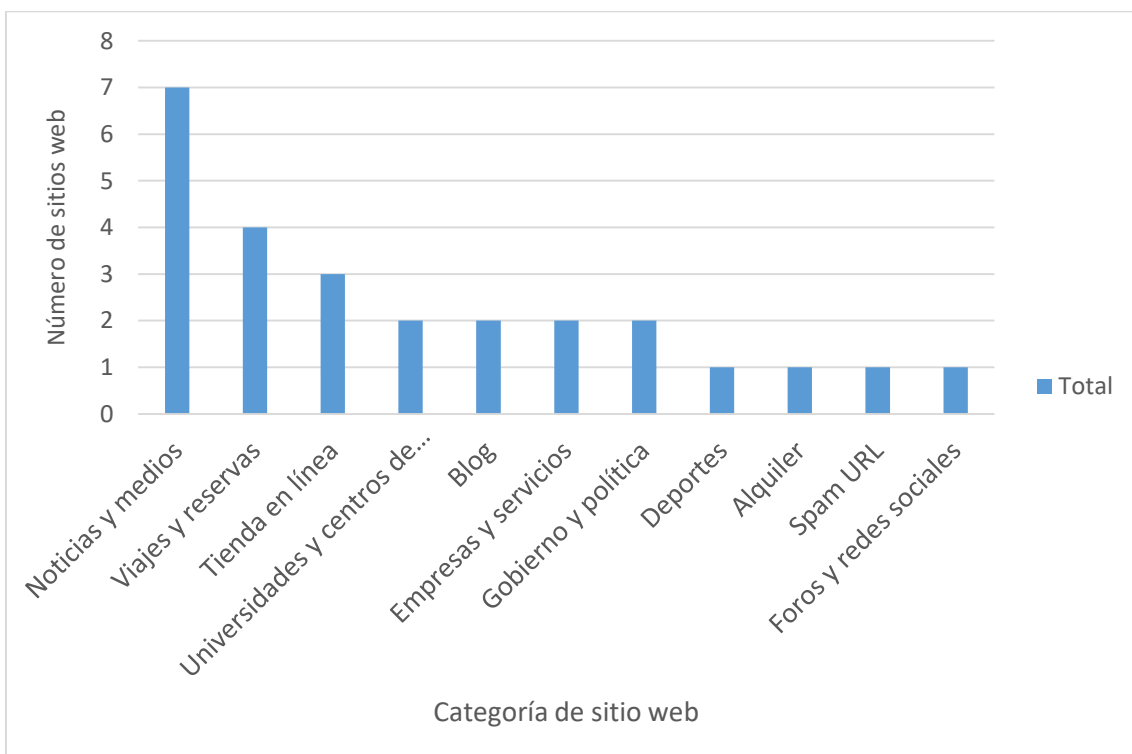


Figura 17. Número de sitios web que emplean técnicas de *fingerprinting*, por categoría - Majestic Million.

De acuerdo al gráfico de la figura 17, las primeras tres categorías reflejadas en el ranking Alexa también se presentan en este ranking con el mayor número de sitios web, con “Noticias y medios” en primer lugar con 7 sitios, “Viajes y reservas” en segundo lugar con 4 sitios y “Tienda en línea” de tercero con 3 sitios.

Finalmente, entre las categorías que se destacan por ser inusuales en cuanto a presencia de *fingerprinting* se refieren, se encuentran “Universidades y centros de formación” donde se detectaron los mismos 2 sitios que en el ranking Alexa y “Gobierno y política”, igualmente con 2 sitios.

5.4 Otras pruebas desarrolladas.

Aunado a los casos de prueba principales expuestos previamente en este capítulo, se diseñaron y realizaron casos de prueba conservando los criterios definidos, pero empleando conjuntos de datos más específicos donde todos los sitios web pertenecen a una determinada categoría. De esta manera, se crearon dos listados de sitios web:

- Sitios web de universidades públicas de España [52].
- Sitios web de entidades del sector público estatal de España [53]

Las pruebas desarrolladas con estos sitios web y sus respectivos resultados se describen a continuación.

5.4.1 Resultados del conjunto de sitios web de universidades públicas.

Se analizaron 50 sitios web pertenecientes a universidades públicas españolas. No se encontró la presencia de ninguna de las herramientas de *fingerprinting* implementadas en la aplicación prototipo (Tabla 11).

Tabla 11. Resumen de resultados – universidades públicas.

Número de sitios web rastreados	50
Número de sitios web que emplean <i>fingerprinting</i>	0
Número de intentos de <i>fingerprinting</i>	0
Intentos por Canvas	0
Intentos por Objetos Informativos	0
Intentos por WebRTC	0
Intentos por Canvas-Fuentes	0

5.4.2 Resultados del conjunto de sitios web de entidades del sector público.

En esta categoría se definió una lista de 368 sitios web, de los cuales solo 4 (1,087%) sitios presentaban un uso de los objetos JavaScript acorde a los criterios de detección de *fingerprinting* definidos. Los resultados detallados de esta prueba se presentan en la tabla 12.

Tabla 12. Resumen de resultados – entidades del sector público.

Número de sitios web rastreados	368
Número de sitios web que emplean <i>fingerprinting</i>	4 (1,087%)
Número de intentos de <i>fingerprinting</i>	4
Intentos por Canvas	1 (25%)
Intentos por Objetos Informativos	3 (75%)
Intentos por WebRTC	0
Intentos por Canvas-Fuentes	0

En los resultados obtenidos se destaca el bajo número de sitios web de esta categoría que emplean alguna técnica de extracción de huellas. De los sitios detectados, 3 (75%) empleaban *fingerprinting* por objetos informativos de JavaScript, mientras que 1 (25%) sitio web aplicaba la técnica de Canvas. En este caso, el número de intentos de coincide con el número de sitios web donde se presentan las técnicas. No se encontraron casos de *fingerprinting* por WebRTC o Canvas-Fuentes.

6. Conclusiones y líneas futuras de investigación

En el presente capítulo se presentan los resultados más destacables del presente trabajo de investigación, resumiendo además los métodos utilizados para concretar los distintos aportes realizados. Dichos resultados parciales permitieron la consecución del objetivo general de la investigación, que consistió en el desarrollo de un prototipo de aplicación para detección de técnicas de identificación y seguimiento de usuarios en la web.

Por otro lado, se ofrecen las bases para llevar a cabo proyectos e investigaciones basadas en las teorías y métodos empleados en la presente investigación, así como en los resultados obtenidos.

6.1 Conclusiones

Existen diversas técnicas, interfaces y herramientas que permiten identificar y monitorizar usuarios en la web, desde aquellas que aprovechan las características del protocolo HTTP para capturar datos de los terminales conectados a la web como otras basadas en la ejecución subrepticia de código embebido en páginas web. Específicamente, aquellos procedimientos que emplean medios como los descritos anteriormente y que no dependen del estado del dispositivo, se definen por el término *fingerprinting*.

En la presente investigación, se desarrolló un prototipo de aplicación para detectar la presencia de las técnicas basadas en *fingerprinting* activo, específicamente aquellas que emplean objetos informativos como *navigator* y *screen* de JavaScript, así como elementos de gráficos y fuentes renderizados con el API Canvas, y filtración de direcciones IP a través de WebRTC. Se priorizó en la implementación de criterios de detección de estas técnicas, considerando los valores de entropía que un atributo extraído por estos métodos puede aportar a una huella, así como su presencia en la web según estudios previos. Por ejemplo, el valor de entropía relacionado al atributo generado a raíz de las imágenes generadas por Canvas presenta un valor de 0,491 en el conjunto de datos originales de AmIUnique (cuarto valor más elevado) y la técnica en sí se presenta en el 3,91 % de los primeros 10.000 sitios web del ranking Alexa según el estudio de Englehardt y Narayanan del año 2016.

La aplicación prototipo se encuentra conformada, a alto nivel, por un componente que funciona como rastreador web o *crawler*, como la herramienta OpenWPM, cuyos resultados almacenados en una base de datos son analizados por medio de scripts de detección de *fingerprinting* implementados en lenguaje Python y basados en los criterios definidos para cada técnica. Todo ello se integra en un componente de presentación basado en un entorno de Jupyter Notebook.

Para evaluar la herramienta desarrollada se emplearon dos conjuntos de datos distintos, cada uno basado en los primeros 1.000 sitios web con dominio .es según rankings web disponibles públicamente. Uno de los conjuntos se elaboró según en el ranking de Alexa y otro con el de Majestic Million.

Los resultados obtenidos permiten validar, en primer lugar, la factibilidad técnica de la herramienta en relación a la capacidad de analizar datos provenientes directamente de sitios web. En general, el presente trabajo de investigación determinó que un 5,5% de los primeros 1.000 sitios web con dominio .es llevan a cabo algún proceso de extracción de huellas.

A pesar de que las pruebas se aplicaron sobre sitios con dominio .es, es posible realizar una comparación con el estudio de los desarrolladores de OpenWPM Englehardt y Narayanan. En dicho estudio, la técnica de Canvas fue la más común en los sitios web del ranking Alexa, seguida por Canvas-Fuentes y WebRT. En el presente trabajo de investigación, la prevalencia de las técnicas se mantuvo en el mismo orden para ambos conjuntos de datos, con la técnica de Canvas resultando nuevamente como la más predominante, abarcando el 45% de los intentos de *fingerprinting* en el conjunto basado en Alexa y 47% en el conjunto basado en Majestic Million.

No obstante, se observó que en lugar de Canvas-Fuentes, la segunda técnica más común fue la basada en objetos informativos de JavaScript, no incluida en el anterior estudio y presente en el 32% de los intentos de *fingerprinting* en el ranking Alexa y 38% en el de Majestic Million. WebRTC fue la tercera más común (14% Alexa, 9% Majestic Million) y por último se ubicó Canvas-Fuentes (9% Alexa, 6% Majestic Million).

A través de la aplicación de la herramienta fue posible encontrar que un sitio web puede emplear diversos scripts para aplicar diversas técnicas de *fingerprinting* a la vez o incluso, un mismo tipo de técnica más de una vez. Es por ello que a diferencia del estudio de Englehardt y Narayanan donde se contabiliza la presencia de una técnica según los sitios web únicos que la aplican, en el presente estudio se optó por contabilizar el número de intentos totales de *fingerprinting* para determinar la prevalencia de las distintas técnicas en la web.

Sin embargo, se empleó el criterio de sitios web únicos para evaluar las categorías donde es más común la presencia de *fingerprinting*, encontrándose que son las de noticias y medios, viajes y reservas y tienda en línea. En el conjunto de datos basado en el ranking Alexa, 25 de 55 sitios web únicos que emplean las técnicas detectadas pertenecen a estas categorías, mientras que en el de Majestic Million son 14 de 26 sitios web únicos. A pesar de que el presente trabajo de investigación no profundiza en las implicaciones que tiene esta práctica desde el punto de vista de la privacidad de los

usuarios o los tipos de perfiles que se pueden desarrollar de los usuarios en combinación con información identificadora o incluso sensible, desde esa perspectiva se destacan categorías como universidades y centros de formación, gobierno y política, así como banca, donde se detectaron sitios web que emplean *fingerprinting*.

En general, la aplicación prototipo permite obtener resultados contrastables y congruentes en relación a estudios previos. El uso y entendimiento de esta herramienta y los resultados que permite obtener puede ayudar a incrementar la conciencia y conocimiento sobre las prácticas de *fingerprinting* en la web y permitir a una comunidad, tanto de usuarios como de desarrolladores, evaluar el uso sospechoso, excesivo o inusual de los objetos JavaScript. Los criterios definidos e implementados permiten un claro entendimiento de los métodos, llamadas y propiedades asociados a cada técnica en particular, e implementar con una metodología similar, nuevos criterios de detección.

6.2 Líneas futuras de investigación

Se proponen las siguientes líneas de investigación, como formas de extender el alcance de la aplicación propuesta:

- Implementar criterios de detección de nuevas técnicas de *fingerprinting*, tanto pasivas como activas, sobre la aplicación prototipo.
- Desarrollar mecanismos y herramientas de defensa ante técnicas de *fingerprinting* activo como aquellas basadas en objetos informativos, Canvas, Canvas-Fuentes y WebRTC, de acuerdo a los criterios de detección definidos e implementados en la aplicación prototipo.
- Estudiar el impacto que tiene en la privacidad del usuario la práctica de *fingerprinting* llevada a cabo por sitios web, con énfasis en aquellos dominios donde el seguimiento de usuarios pueda ser considerado como un caso especialmente atípico, desde la perspectiva de su ámbito o actividad y el tipo de información que manejan en relación a sus visitantes.
- Determinar la legalidad del empleo de técnicas de extracción de huellas según el marco regulatorio existente en materia de la protección y privacidad de los datos. Una reflexión sobre este tópico se ofrece en el anexo 7.
- Desarrollar pruebas de rendimiento de la aplicación prototipo frente a otras aplicaciones similares.
- Evaluar la portabilidad de los scripts de análisis desarrollados en la presente investigación, tomando en cuenta la posibilidad de emplear un rastreador o *crawler* alternativo, un lenguaje distinto a Python, entre otras posibilidades.
- Desarrollar un modelo de aprendizaje automático, con base a un conjunto de datos elaborado a través de la aplicación propuesta en la presente

investigación, con la finalidad de predecir la presencia de técnicas de *fingerprinting* en la web.

- Definir criterios de detección de *fingerprinting* en otros contextos (ej. Aplicaciones en dispositivos inteligentes, servicios de marketing por correo electrónico, etc.)
- Determinar el empleo e impacto de las técnicas de *fingerprinting* en ciertos procedimientos comunes en aplicaciones en línea, como *Single Sign-On*, geolocalización, etc.
- Implementar criterios de detección para técnicas de *fingerprinting* que permiten asociar distintos dispositivos a un mismo usuario.
- Estudiar criterios de detección bajo una modalidad de lista negra, basado en los scripts y direcciones web de terceros que la aplicación propuesta ha detectado como positivos en el empleo de técnicas de *fingerprinting*.

Bibliografía

- [1] P. Huffstickler, «CodePen,» [En línea]. Disponible: <https://codepen.io/philmanheim/pen/gWepON>.
- [2] A. Baxter, «The Truth About Data Mining: How Online Trackers Gather Your Info and What They See,» *Observer*, 21 Julio 2016. [En línea]. Disponible: <http://observer.com/2016/07/the-truth-about-data-mining-how-online-trackers-gather-your-info-and-what-they-see/>.
- [3] P. Iglesias, «Web Tracking: Cómo nos identifican y monitorizan en Internet,» 2016. [En línea]. Disponible: <https://www.pabloyglesias.com/wp-content/uploads/2013/03/Especial-VIII-Web-Tracking-pabloyglesias.pdf>.
- [4] Electronic Frontier Foundation (EFF), «Panopticklick 3.0,» EFF, [En línea]. Disponible: <https://panopticklick.eff.org>.
- [5] E. Barolli, L. Nebiaj y G. Tyxhari, «Data Traffic and Security over Internet via Monitoring and Analyzing the HTTP Protocol,» *International Journal of Engineering & Technology IJET-IJENS*, vol. 14, n° 16, pp. 44-50, 2014.
- [6] N. Hassan y R. Hijazi, *Digital Privacy and Security Using Windows: A Practical Guide*, Apress, 2017.
- [7] F. Velázquez, K. Lyngstøl, T. Fog y J. Renard, *The Varnish Book*, Varnish Software AS, 2017.
- [8] D. Stuttard y M. Pinto, *The Web Application Hacker's Handbook*, Segunda ed., Indianapolis: John Wiley & Sons, Inc., 2011.
- [9] R. Broenink, «Using Browser Properties for Fingerprinting Purposes,» *16th biennial Twente Student Conference on IT*, 2012.
- [10] G. Gourley y B. Totty, *HTTP: The Definitive Guide*, Sebastopol, California: O'Reilly Media, Inc., 2002.
- [11] W. Huba, *An Analysis of various web tracking methods*, Rochester Institute of Technology, 2012.

- [12] M. Zalewski, *The Tangled Web: A Guide to Securing Modern Web Applications*, San Francisco: No Starch Press, Inc., 2012.
- [13] S. Kamkar, «evercookie,» 20 Septiembre 2010. [En línea]. Disponible: <https://samy.pl/evercookie/>.
- [14] A. Prášil, *Website user tracking*, Praga: Fakulta informačních technologií ČVUT, 2016.
- [15] N. Schmücker, «Web Tracking,» *SNET2 Seminar Paper-Summer Term*, 2011.
- [16] M. Osakwe, «Cookies: The Secret to Online Tracking,» *NextAdvisor*, 5 Febrero 2018. [En línea]. Disponible: <https://www.nextadvisor.com/blog/2018/02/05/cookies-the-secret-to-online-tracking/>.
- [17] G. Acar, C. Eubank, S. Englehardt, M. Juarez, A. Narayanan y C. Diaz, «The Web Never Forgets: Persistent Tracking Mechanisms in the Wild,» *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pp. 674-689, 2014.
- [18] S. Englehardt, «The hidden perils of cookie syncing,» 7 Agosto 2014. [En línea]. Disponible: <http://freedom-to-tinker.com/2014/08/07/the-hidden-perils-of-cookie-syncing/>.
- [19] P. Laperdrix, *Browser Fingerprinting: Exploring Device Diversity to Augment Authentication and Build Client-Side Countermeasures*, Institut National des Sciences Appliquées de Rennes, 2017.
- [20] A. F. Khademi, *Browser Fingerprinting: Analysis, Detection, and Prevention at Runtime*, Ontario: Queen's University's School of Computing, 2014.
- [21] S. Hussain, M. Aleem, A. Islam y A. Ishtiaq, «User tracking mechanisms and counter-measures,» *International Journal of Applied Mathematics, Electronics and Computers*, vol. 5, n° 2, pp. 33-40, 2017.
- [22] G. Richards, S. Lebresne, B. Burg y J. Vitek, «An Analysis of the Dynamic Behavior of JavaScript Programs,» *ACM Sigplan Notices*, vol. 45, n° 6, pp. 1-12, 2010.

- [23] A. F. Khademi, M. Zulkernine y K. Weldemariam, «An Empirical Evaluation of Web-Based Fingerprinting,» *IEEE Software*, vol. 32, n° 4, pp. 46-52, 2015.
- [24] V. Bernardo y D. Domingos, «Web-based Fingerprinting Techniques,» *Proceedings of the 13th International Joint Conference on e-Business and Telecommunications (ICETE 2016)*, vol. 4, pp. 271-282, 2016.
- [25] S. Englehardt y A. Narayanan, «Online Tracking: A 1-million-site Measurement and Analysis,» *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1388--1401, 2016.
- [26] X. Liu, Q. Liu, X. Wang y Z. Jia, «Fingerprinting Web Browser for Tracing Anonymous Web Attackers,» *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*, pp. 222-229, 2016.
- [27] O. Starov y N. Nikiforakis, «XHOUND: Quantifying the Fingerprintability of Browser Extensions,» *2017 IEEE Symposium on Security and Privacy*, pp. 941-956, 2017.
- [28] N. Kaur, S. Azam, K. Kannoopatti y K. C. Yeo, «Browser Fingerprinting as user tracking technology,» *2017 11th International Conference on Intelligent Systems and Control (ISCO)*, pp. 103-111, 2017.
- [29] K. Boda, Á. Máté Földes, G. György Gulyás y S. Imre, «User tracking on the web via cross-browser fingerprinting,» *Nordic Conference on Secure IT Systems*, p. 31-46, 2011.
- [30] P. Eckersley, «How Unique Is Your Web Browser?,» *PETS'10 Proceedings of the 10th international conference on Privacy enhancing technologies*, pp. 1-18, 2010.
- [31] P. Laperdrix, W. Rudametkin y B. Baudry, «Beauty and the Beast: Diverting modern web browsers to build unique browser fingerprints,» *2016 IEEE Symposium on Security and Privacy*, pp. 878-894, 2016.
- [32] C. E. Shannon, «Prediction and entropy of printed English,» *The Bell System Technical Journal*, vol. 30, n° 1, pp. 50-64, 1951.
- [33] Am I Unique?, «Global Statistics,» [En línea]. Disponible: <https://amiunique.org/stats>.

- [34] A. Gómez-Boix, P. Laperdrix y B. Baudry, «Hiding in the Crowd: an Analysis of the Effectiveness of Browser Fingerprinting at Large Scale,» *WWW 2018: The 2018 Web Conference*, pp. 309-318, 2018.
- [35] Y. Cao, S. Li y E. Wijmans, «(Cross-)Browser Fingerprinting via OS and Hardware Level Features,» *NDSS Symposium 2017*, 2017.
- [36] J. Mayer, «FourthParty,» The Center for Internet and Society - Stanford Law School, [En línea]. Disponible: <http://fourthparty.info>.
- [37] G. Acar, M. Juarez, N. Nikiforakis, C. Diaz, S. Gürses, F. Piessens y B. Preneel, «FPDetective: Dusting the Web for Fingerprinters,» *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 1129-1140, 2013.
- [38] S. Helme, «SecurityHeaders,» [En línea]. Disponible: <https://securityheaders.io>.
- [39] A. Andersdotter y A. Jensen-Urstad, «Evaluating Websites and Their Adherence to Data Protection Principles: Tools and Experiences,» *Privacy and Identity Management. Facing up to Next Steps. Privacy and Identity 2016. IFIP Advances in Information and Communication Technology*, vol. 498, pp. 39-51, 2016.
- [40] «PrivacyScore,» [En línea]. Disponible: <https://privacyscore.org>.
- [41] «Chameleon,» GitHub, Inc., [En línea]. Disponible: <https://github.com/ghostwords/chameleon>.
- [42] «Don't FingerPrint Me,» GitHub, Inc., [En línea]. Disponible: <https://github.com/freethenation/DFPM>.
- [43] C. Gonçalves, Who is Tracking Me on the Web?, Instituto Superior de Engenharia do Porto, 2016.
- [44] WebTAP, «Princeton Web Transparency & Accountability Project,» Princeton University, [En línea]. Disponible: <https://webtap.princeton.edu>.
- [45] Princeton Web Transparency & Accountability Project, «OpenWPM,» GitHub, Inc., [En línea]. Disponible: <https://github.com/citp/OpenWPM>.

- [46] A. F. Khademi, M. Zulkernine y K. Weldemariam, «FPGuard: Detection and Prevention of Browser Fingerprinting,» *29th IFIP Annual Conference on Data and Applications Security and Privacy (DBSEC)*, pp. 293-308, 2015.
- [47] L. Olejnik, S. Englehardt y A. Narayanan, «Battery Status Not Included: Assessing Privacy in Web Standards,» *2017 International Workshop on Privacy Engineering*, 2017.
- [48] C. Blakemore, *Fingerprinting for Web Applications: from Devices to Related Groups*, Instituto Superior Técnico, Universidade de Lisboa, 2016.
- [49] «itdelatrisu/OpenWPM,» GitHub, Inc., [En línea]. Disponible: <https://github.com/itdelatrisu/OpenWPM>.
- [50] jdanml, «Web Fingerprinting Detection Tool,» GitHub, Inc., [En línea]. Disponible: <https://github.com/jdanml/Web-Fingerprinting-Detection-Tool>.
- [51] J. Kyrölä, «Tapping the web to measure its privacy,» *Proceedings of the Seminar in Computer Science: Internet, Data and Things (CS-E4000)*, pp. 109-121, 2018.
- [52] Estudios y Universidades en España, «Listado de Universidades Públicas de España,» [En línea]. Disponible: <http://www.estudiosyuniversidades.com/universidades/universidades-publicas-espana.html>.
- [53] Transparencia Internacional España, «Entidades del sector público estatal,» [En línea]. Disponible: https://webantigua.transparencia.org.es/entes_publicas_en_espana/listado_detallado_entidades_sector_publico_estatal.pdf.
- [54] Jon B, «CodePen,» [En línea]. Disponible: <https://codepen.io/jon/pen/rwBbgQ>.
- [55] W3Schools, «HTML canvas measureText() Method,» [En línea]. Disponible: https://www.w3schools.com/tags/canvas_measuretext.asp.
- [56] Directiva 95/46/CE del Parlamento Europeo y del Consejo relativa a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos, Diario Oficial n° L 281/31, 1995.

- [57] Reglamento (UE) 2016/679 del Parlamento Europeo y del Consejo relativo a la protección de las personas físicas en lo que respecta al tratamiento de datos personales y a la libre circulación de estos datos y por el que se deroga la Directiva 95/46/CE (Reglamento general de protección de datos), Diario Oficial n° L 119/1, 2016.
- [58] Article 29 Working Party (WP29), Opinion 9/2014 on the application of Directive 2002/58/EC to device fingerprinting, 2014.
- [59] Article 29 Working Party (WP29), Opinion 01/2017 on the Proposed Regulation for the ePrivacy Regulation (2002/58/EC), 2017.
- [60] Directiva 2002/58/CE del Parlamento Europeo y del Consejo relativa al tratamiento de los datos personales y a la protección de la intimidad en el sector de las comunicaciones electrónicas (Directiva sobre la privacidad y las comunicaciones electrónicas), Diario Oficial n° L 201/37, 2002.
- [61] Propuesta de Reglamento del Parlamento Europeo y el Consejo sobre el respeto de la vida privada y la protección de los datos personales en el sector de las comunicaciones electrónicas y por el que se deroga la Directiva 2002/58/CE (Reglamento sobre la privacidad y las comunicaciones electrónicas), 2017/0003(COD), 2017.
- [62] TermsFeed, «Legal requirements for device fingerprinting,» 16 Octubre 2016. [En línea]. Disponible: <https://termsfeed.com/blog/legal-requirements-device-fingerprinting/>.
- [63] Alexa Internet, Inc., «How are Alexa's traffic rankings determined?,» [En línea]. Disponible: <https://support.alexa.com/hc/en-us/articles/200449744-How-are-Alexa-s-traffic-rankings-determined->.
- [64] Majestic, «El Majestic Million,» [En línea]. Disponible: <https://es.majestic.com/reports/majestic-million>.

Anexos

Anexo 1. Ejemplo de resultado de script con técnica de Canvas [54].

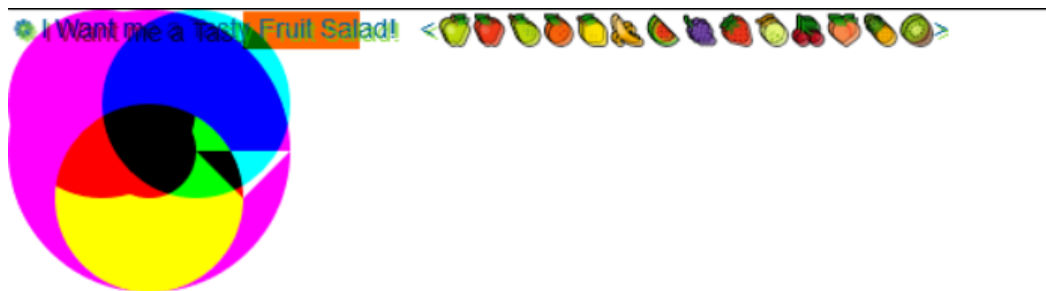
En este ejemplo, nótese que el valor del hash generado en cada navegador es distinto.

- Google Chrome (Versión 67.0.3396.79):



01f8eee9f61d956518da20e20577bbb43778771ec3921e9f20bb79ff52a5890a

- Mozilla Firefox (Versión 60.0.2):



910635fb24ec01a703d906fa9833c756206c8f6b43f13af7ef80631d636a4c2f

- Microsoft Edge (Versión 42.17134.1.0):

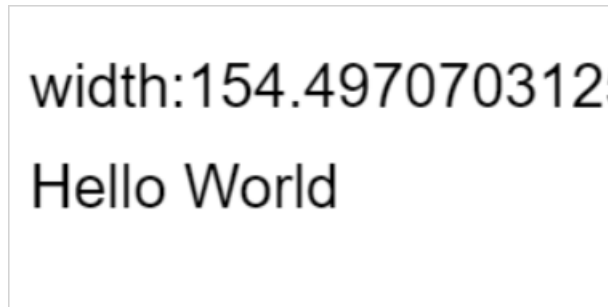


746acc31b00a4d03c88c1ffb47bff353424b45e72c6a257168a2540a2f5e5992

Anexo 2. Ejemplo de *measureText* para la técnica de Canvas-Fuentes [55]

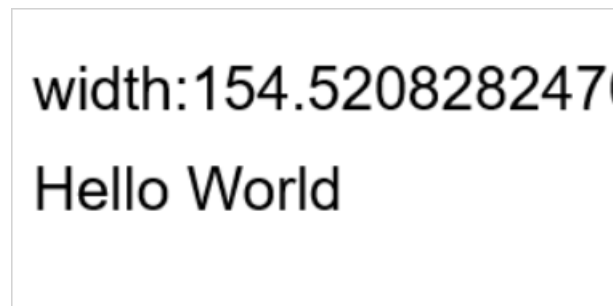
En este ejemplo, nótese que la anchura del texto medida en cada navegador es distinta. En las 3 imágenes, el tipo de fuente (“Arial”) y el tamaño (30) se conservaron constantes.

- Google Chrome (Versión 67.0.3396.79):



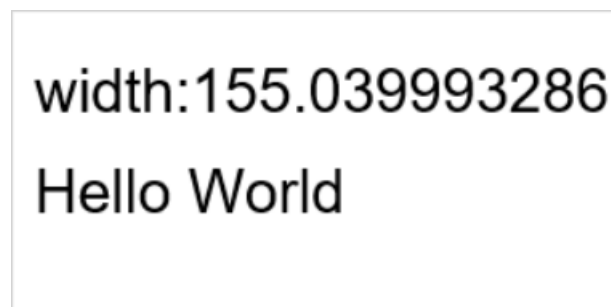
A screenshot of a browser window showing the text "Hello World" in a black serif font. Above the text, the measured width is displayed as "width:154.4970703125px".

- Mozilla Firefox (Versión 60.0.2):




A screenshot of a browser window showing the text "Hello World" in a black serif font. Above the text, the measured width is displayed as "width:154.520828247px".


- Microsoft Edge (Versión 42.17134.1.0):





A screenshot of a browser window showing the text "Hello World" in a black serif font. Above the text, the measured width is displayed as "width:155.039993286px".

Anexo 3. Entidades y atributos respectivos de la base de datos generada por OpenWPM.


http_requests	
 id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
url	TEXT
top_level_url	TEXT
method	TEXT
referrer	TEXT
headers	TEXT
channel_id	TEXT
is_XHR	BOOLEAN
is_frame_load	BOOLEAN
is_full_page	BOOLEAN
is_third_party_channel	BOOLEAN
is_third_party_window	BOOLEAN
triggering_origin	TEXT
loading_origin	TEXT
loading_href	TEXT
req_call_stack	TEXT
content_policy_type	INTEGER
post_body	TEXT
time_stamp	TEXT


javascript_cookies	
 id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
change	TEXT
creationTime	DATETIME
expiry	DATETIME
is_http_only	INTEGER
is_session	INTEGER
last_accessed	DATETIME
raw_host	TEXT
expires	INTEGER
host	TEXT
is_domain	INTEGER
is_secure	INTEGER
name	TEXT
path	TEXT
policy	INTEGER
status	INTEGER
value	TEXT

http_responses	
 id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
url	TEXT
method	TEXT
referrer	TEXT
response_status	INTEGER
response_status_text	TEXT
is_cached	BOOLEAN
headers	TEXT
channel_id	TEXT
location	TEXT
time_stamp	TEXT
content_hash	TEXT

javascript	
 id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
script_url	TEXT
script_line	TEXT
script_col	TEXT
func_name	TEXT
script_loc_eval	TEXT
call_stack	TEXT
symbol	TEXT
operation	TEXT
value	TEXT
arguments	TEXT
time_stamp	TEXT


profile_cookies	
id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
baseDomain	TEXT
name	TEXT
value	TEXT
host	TEXT
path	TEXT
expiry	INTEGER
accessed	INTEGER
creationTime	INTEGER
isSecure	INTEGER
isHttpOnly	INTEGER

http_redirects	
 id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
old_channel_id	TEXT
new_channel_id	TEXT
is_temporary	BOOLEAN
is_permanent	BOOLEAN
is_internal	BOOLEAN
is_sts_upgrade	BOOLEAN
time_stamp	TEXT


content_policy	
 id	INTEGER
crawl_id	INTEGER
content_type	INTEGER
content_location	TEXT
request_origin	TEXT
mime_type_guess	TEXT
page_id	INTEGER
visit_id	INTEGER

flash_cookies	
id	INTEGER
crawl_id	INTEGER
visit_id	INTEGER
domain	VARCHAR
filename	VARCHAR
local_path	VARCHAR
key	TEXT
content	TEXT

crawl	
crawl_id	INTEGER
task_id	INTEGER
browser_params	TEXT
screen_res	TEXT
ua_string	TEXT
finished	BOOLEAN
start_time	DATETIME

xpath		
 id	INTEGER	
name	VARCHAR	
url	VARCHAR	
xpath	VARCHAR	
absolute_xpath	VARCHAR	
ctime	DATETIME	

localStorage		
id	INTEGER	
crawl_id	INTEGER	
page_url	VARCHAR	
scope	TEXT	
KEY	TEXT	
value	TEXT	

task		
 task_id	INTEGER	
start_time	DATETIME	
manager_params	TEXT	
openwpm_version	TEXT	
browser_version	TEXT	

CrawlHistory		
crawl_id	INTEGER	
command	TEXT	
arguments	TEXT	
bool_success	INTEGER	
dtg	DATETIME	

site_visits		
visit_id	INTEGER	
crawl_id	INTEGER	
site_url	VARCHAR	

sqlite_sequence		
name	N/A	
seq	N/A	

Anexo 4. Capturas de pantalla de la aplicación prototipo.

- Paso 1: Se brindan opciones para ingresar la lista de sitios web, ya sea a través de un fichero CSV o de forma manual.

Detección de Fingerprinting en sitios web

1. Ingrese la lista de sitios

Indique un conjunto de sitios web a explorar por OpenWPM. Existen dos opciones:

- **CSV:** Permite la carga de un fichero CSV con la presentación indicada.
- **Manual:** Presenta un recuadro en el que es posible escribir las distintas URL de los sitios web.



Presione el botón **Cargar** para buscar un fichero CSV en su disco.

El contenido del fichero CSV debe presentar el siguiente patrón:

- 1,sitio1.com
- 2,sitio2.net
- 3,sitio3.org

Nota: Si ha ingresado sitios web empleando la opción "Manual", asegúrese que el botón "Confirmar lista" no esté activado

Detección de Fingerprinting en sitios web

1. Ingrese la lista de sitios

Indique un conjunto de sitios web a explorar por OpenWPM. Existen dos opciones:

- **CSV:** Permite la carga de un fichero CSV con la presentación indicada.
- **Manual:** Presenta un recuadro en el que es posible escribir las distintas URL de los sitios web.



Ingrese las URL a rastrear en el recuadro de la izquierda.

Instrucciones:

- Incluya el protocolo (http, https) en cada dirección.
- Separe cada elemento con una coma.

Por ejemplo:

https://www.sitio1.com, https://www.sitio2.net, http://www.sitio3.org

- Presione el botón **Confirmar** para activar el uso del conjunto ingresado.

- Paso 2: Indica que se debe ejecutar la herramienta OpenWPM.

2. Rastrear sitios web

Presione el botón **Ejecutar OpenWPM** para iniciar el rastreador web con la lista de sitios ingresada.

Cuando la ejecución de la herramienta finalice, debe observar el siguiente mensaje: **Ejecución finalizada**.

Ejecutar OpenWPM

- Paso 3: Se brindan botones para mostrar los resultados generales y por cada técnica de *fingerprinting* implementada, así como un botón para borrar los resultados.

3. Resultados

Presione el botón "Resumen General" para mostrar una tabla de los sitios web que aplican por lo menos una de las siguientes técnica des *fingerprinting*:
Objetos informativos, Canvas, Camvas-Fuentes y WebRTC.

Resumen General

Resultados detallados

Para obtener la lista de los sitios que aplican una técnica de *fingerprinting* en particular, además del URL del script correspondiente, presione uno o varios de los siguientes botones en función de la técnica o técnicas de su interés.

Objetos Informativos

Canvas

Canvas-Fuentes

WebRTC

Por defecto se guarda una base de datos donde se acumulan todos los datos obtenidos a través de OpenWPM. Si desea partir en limpio para la siguiente ejecución, presione el botón **Borrar Resultados**.

Borrar Resultados

© 2018 Juan D. Márquez Lagalla

Herramienta de detección de *fingerprinting*

Manual de Desarrollo

Índice

- [Introducción](#)
- [Requisitos](#)
- [OpenWPM](#)
- [Implementando un script de análisis](#)
- [Recomendaciones adicionales](#)
- [Comentarios](#)

Introducción

Para la detección de técnicas de extracción de huellas o *fingerprinting* activo, se implementan diversos scripts que consultan la base de datos en busca de parámetros como objetos JavaScript para verificar el cumplimiento de los criterios que caracterizan dichas técnicas (ej. Canvas, Objetos informativos, Canvas-Fuentes, WebRTC, etc.)

Este documento describe el entorno de la aplicación e incluye las formas en que el desarrollador puede implementar nuevos criterios de detección basado en la metodología aplicada a los scripts de análisis existentes.

Requisitos

Sistema Operativo

Se recomienda Ubuntu 16.04/17.10/18.04.

Versión de Python

- Python 3.5 o superior
- Pip

Dependencias

- OpenWPM (<https://github.com/citp/OpenWPM>). Para instalar OpenWPM, ejecuté el script **install.sh**. Si desea desarrollar la extensión que emplea OpenWPM para instrumentar o ejecutar pruebas específicas a esta herramienta, emplee el script **install-dev.sh**.
- Anaconda (Recomendado) (<https://docs.anaconda.com/anaconda/install/>). Incluye los paquetes Pandas y Jupyter Notebook.
- Opcional. Para la correcta visualización de los elementos del notebook se recomienda:
 - Jupyter notebook extensions (https://github.com/ipython-contrib/jupyter_contrib_nbextensions)
 - Habilite las siguientes extensiones:
 - `jupyter nbextension enable init_cell/main`
 - `jupyter nbextension enable hide_input/main`

Manejo de archivos

- El notebook de la aplicación está diseñado para ejecutarse dentro de la misma carpeta donde se encuentra OpenWPM.
- Los scripts que ejecutan OpenWPM almacenan los resultados en la misma carpeta donde se encuentra la herramienta.
- El nombre por defecto del fichero que contiene la base de datos es "crawl-data.sqlite". Los scripts de análisis implementados asumen por defecto que este fichero se encuentra en la carpeta con este nombre.

OpenWPM

La herramienta empleada como *crawler*. Accede a diversos sitios web y almacena en una base de datos sqlite todos los elementos instrumentados. Consideraciones con respecto a OpenWPM:

- La documentación se encuentra disponible en <https://github.com/citp/OpenWPM/wiki>.
- Para habilitar la tabla "javascript" en la base de datos y almacenar los contenidos JavaScript de los sitios web, se debe habilitar la instrumentación de objetos JavaScript. Para ello, edite el fichero **automation/default_browser_params.json** y modifique la línea "**js_instrument**": **false** para establecerla en "**js_instrument**": **true**.
- Por defecto, OpenWPM ejecuta tres instancias del navegador.
- La tabla "javascript" de la base de datos presenta las siguientes columnas:

id	crawl_id	visit_id	script_url	script_line	script_col
func_name	script_loc_eval	call_stack	symbol	operation	value
arguments	time_stamp				

- Otra tabla de la base de datos que es de relevancia para la realización de consultas es "site_visits". Esta tabla nos permite conocer la URL del sitio web original (site_url) que llama al script.

	visit_id	crawl_id	site_url
0	1	1	http://google.es
1	2	2	http://google.es
2	3	3	http://google.es
3	4	2	http://blogspot.com.es
4	5	3	http://blogspot.com.es

Implementando un script de análisis

El desarrollo de un script de análisis puede descomponerse en 4 pasos:

1. Conexión con la base de datos.
2. Definición de variables.
3. Consulta de la base de la base de datos.
4. Representación de resultados.

1. Conexión con la base de datos.

Para conectarse a la base de datos, se realiza lo siguiente:

```
#import sqlite3 as lite
wpm_db = "crawl-data.sqlite"
conn = lite.connect(wpm_db)
cur = conn.cursor()
```

2. Definición de variables.

Se recomienda definir como variables los objetos JavaScript que permiten comprobar las condiciones de *fingerprinting*. Por ejemplo, para la técnica de Canvas se tendría:

```
canvas_h = "HTMLCanvasElement.height"
canvas_w = "HTMLCanvasElement.width"
canvas_cf= "CanvasRenderingContext2D.fillStyle"
canvas_cs= "CanvasRenderingContext2D.strokeStyle"
canvas_ff= "CanvasRenderingContext2D.fillText"
canvas_fs= "CanvasRenderingContext2D.strokeText"
canvas_save= "CanvasRenderingContext2D.save"
canvas_rest= "CanvasRenderingContext2D.restore"
canvas_el= "HTMLCanvasElement.addEventListener"
canvas_data= "HTMLCanvasElement.toDataURL"
canvas_img= "CanvasRenderingContext2D.getImageData"
```

También se recomienda definir estructuras de datos para almacenar resultados parciales de cada criterio a verificar. Se destacan:

- Sets: Contienen valores únicos y sin ordenar. La primera característica es útil ya que permite evitar duplicados por ejecutar diferentes instancias del navegador.
- Lists: Conjuntos modificables de valores. Pueden contener elementos duplicados.

Un ejemplo de lo mencionado para la técnica de Canvas es el siguiente:

```
fp_1 = set()
fp_2 = set()
fp_3 = set()
fp_4 = set()
veri_1 = set()
veri_2 = set()
fp_sites = set()
```

- fp_1, fp_2, fp_3, fp_4: Almacenan resultados parciales por cada criterio asociado a la técnica de Canvas.
- veri_1, veri_2: Empleados para verificaciones parciales dentro de criterios específicos.
- fp_sites: Para almacenar los resultados finales.

3. Consulta de la base de la base de datos.

Se empieza por ejecutar una consulta para buscar en la base de datos los objetos JavaScript de interés. Por ejemplo, para la verificación de la técnica de Canvas se puede ejecutar una consulta como la siguiente:

```
for url, symbol, op, val, arg, top_url in cur.execute("SELECT distinct
j.script_url, j.symbol, j.operation, j.value, j.arguments, v.site_url
FROM javascript as j JOIN site_visits as v ON j.visit_id = v.visit_id
WHERE j.symbol LIKE '%Canvas%' ORDER BY v.site_url;"):
```

Si ejecutáramos la consulta directamente a la base de datos, obtendríamos un resultado como el siguiente:

	script_url	symbol	operation	value	arguments	site_url
0	https://rs.20m.es/videoplayer/jw8/jwplayer.js?...	HTMLCanvasElement.height	set	1	None	http://20minutos.es
1	https://rs.20m.es/videoplayer/jw8/jwplayer.js?...	HTMLCanvasElement.width	set	1	None	http://20minutos.es
2	https://rs.20m.es/videoplayer/jw8/jwplayer.js?...	HTMLCanvasElement.getContext	call		{"0": "2d"}	http://20minutos.es

	script_url	symbol	operation	value	arguments	site_url
3	https://rs.20m.es/videoplayer/jw8/jwplayer.js?...	CanvasRenderingContext2D.fillStyle	set	#000000	None	http://20minutos.es
4	https://rs.20m.es/videoplayer/jw8/jwplayer.js?...	CanvasRenderingContext2D.fillRect	call		{"0":0,"1":0,"2":1,"3":1}	http://20minutos.es

En este caso, recorreremos toda la tabla resultante, verificando los criterios relacionados a la técnica de *fingerprinting* analizada. En el ejemplo que se presenta a continuación para la técnica de Canvas, se siguen los siguientes pasos:

- Se compara el valor de las variables que hacen referencia a objetos JavaScript con los valores de la columna symbol.
- Al coincidir se verifica, según sea el caso, que los valores de la columna "value" o de la columna "arguments" se corresponden con los umbrales conocidos de un criterio que define la técnica analizada.
- Se emplean conjuntos como sets para almacenar resultados parciales en formato "URL_Sitio URL_Script".

```

if canvas_h in symbol:
    if val != "null" and float(val) >= 16:
        veri_1.add(symbol + url)
        if (canvas_w + url) in veri_1:
            fp_1.add(top_url + ' ' + url)
elif canvas_w in symbol:
    if val != "null" and float(val) >= 16:
        veri_1.add(symbol + url)
        if (canvas_h + url) in veri_1:
            fp_1.add(top_url + ' ' + url)

if canvas_cf in symbol:
    veri_2.add(symbol + url + val)
    if (canvas_cs + url + val) in veri_2:
        fp_2.add(top_url + ' ' + url)
    elif (sum((canvas_cf + url) in s for s in veri_2)) > 1:
        fp_2.add(top_url + ' ' + url)
elif canvas_cs in symbol:
    veri_2.add(symbol + url + val)
    if (canvas_cf + url + val) in veri_2:
        fp_2.add(top_url + ' ' + url)
    elif (sum((canvas_cs + url) in s for s in veri_2)) > 1:
        fp_2.add(top_url + ' ' + url)
elif canvas_ff in symbol and
(len(set(arg[arg.find('0":') + 4:arg.find(', "1"') - 1]))) >= 10:
    fp_2.add(top_url + ' ' + url)
elif canvas_fs in symbol and
(len(set(arg[arg.find('0":') + 4:arg.find(', "1"') - 1]))) >= 10:

```

```

fp_2.add(top_url + ' ' + url)

if canvas_save in symbol:
    fp_3.add(top_url + ' ' + url)
elif canvas_rest in symbol:
    fp_3.add(top_url + ' ' + url)
elif canvas_el in symbol:
    fp_3.add(top_url + ' ' + url)

if canvas_data in symbol:
    fp_4.add(top_url + ' ' + url)
elif canvas_img in symbol:
    if float(arg[arg.find('"2":')+4:arg.find(', "3"')]) >= 16:
        if float(arg[arg.find('"3":')+4:arg.find('}')] >= 16:
            fp_4.add(top_url + ' ' + url)

```

Finalmente, se combinan los conjuntos de resultados parciales en uno que almacene los resultados finales. Para ello se emplean operaciones como unión, intersección, diferencia y diferencia simétrica. En el ejemplo que se presenta de la técnica de Canvas, interesa que el conjunto final almacene todos los sitios que cumplen con los criterios 1, 2 y 4 (intersección de fp_1, fp_2 y fp_4) y no cumplan con el criterio 3 (diferencia de fp_3):

```
fp_sites = (fp_1 & fp_2 & fp_4) - fp_3
```

4. Representación de resultados.

Se emplea un dataframe (pandas) para manejar y mostrar los resultados finales. Para ello se realizan dos cosas:

- Se separan los sets obtenidos en tuplas. Considérese los valores guardados y los caracteres de separación utilizados.
- Se construye el dataframe con las tuplas obtenidas.

Para el ejemplo desarrollado de Canvas, se realiza lo siguiente:

```

c, d = zip(*(s.split(' ') for s in fp_sites))
df2 = pd.DataFrame({'URL Sitio Web': c, 'URL Script': d})
df2 = df2[['URL Sitio Web', 'URL Script']]

```

Con lo anterior, es posible mostrar el dataframe con los resultados. Se puede definir una función para escribir hipervínculos por cada URL, como se muestra en el caso desarrollado a continuación:

```

def make_clickable(val):
    return '<a href="{}">{}</a>'.format(val, val)
#pd.set_option('max_colwidth', -1)
print('Los sitios web que aplican Canvas Fingerprinting según los
criterios analizados son los siguientes:')
display(df2.style.format(make_clickable))

```

Recomendaciones adicionales

- Puede reutilizar los widgets (elementos gráficos, botones, etc.) empleados en el notebook copiando las celdas. En el caso de los botones que ejecutan una acción, nótese que su función es ejecutar la celda de código siguiente. Ejemplo:

```
def run_cell(ev):

display(Javascript('IPython.notebook.execute_cell_range(IPython.notebook.
get_selected_index()+1, IPython.notebook.get_selected_index()+2)'))

button = widgets.Button(button_style='info',description="Ejecutar
OpenWPM")
button.on_click(run_cell)
display(button)
```

- Para evitar errores, puede verificar que el fichero de base de datos existe. Una manera de hacerlo se presenta a continuación:

```
if os.path.isfile("crawl-data.sqlite"):
```

- Dependiendo del tamaño de la base de datos y de la consulta realizada, el tiempo que tarda un script de análisis desde la ejecución hasta la presentación de los resultados finales puede variar entre unos pocos segundos a varios minutos. Puede agregar un mensaje para indicar que se esta ejecutando la operación. Una manera de hacerlo se muestra a continuación:

```
print("Procesando...")
clear_output(wait=True)
```

- Es recomendable verificar que los sets que contienen los resultados finales no estén vacíos. Por ejemplo, en el caso del análisis de Canvas:

```
if len(fp_sites) == 0:
    print('Ninguno de los sitios evaluados aplica esta técnica de
fingerprinting')
```

Comentarios

- Para seleccionar el fichero CSV, se emplea una ventana de diálogo de Tkinter.
- Para mayor información sobre los widgets empleados, así como de los widgets en general, se recomienda consultar el siguiente enlace: [https://ipywidgets.readthedocs.io/en/latest/examples/Widget Basics.html](https://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Basics.html)

© 2018 Juan D. Márquez Lagalla

Anexo 6. Configuraciones de OpenWPM.

TaskManager - INFO -

OpenWPM Version: v0.8.0-137-gb0a8e00

Firefox Version: 52.8.0

===== Manager Configuration =====

```
{
  "aggregator_address": [
    "127.0.0.1",
    46699
  ],
  "data_directory": "/home/OpenWPM",
  "database_name": "/home/OpenWPM/crawl-data.sqlite",
  "failure_limit": null,
  "ldb_address": [
    "127.0.0.1",
    52731
  ],
  "log_directory": "/home/OpenWPM",
  "log_file": "/home/OpenWPM/openwpm.log",
  "logger_address": [
    "127.0.0.1",
    36003
  ],
  "num_browsers": 3,
  "screenshot_path": "/home/OpenWPM/screenshots",
  "source_dump_path": "/home/OpenWPM/sources",
  "testing": false
}
```

===== Browser Configuration =====

Keys:

```
{
  "crawl_id": 0,
  "adblock-plus": 1,
  "bot_mitigation": 2,
  "browser": 3,
  "cookie_instrument": 4,
  "cp_instrument": 5,
  "disable_flash": 6,
  "disconnect": 7,
  "donottrack": 8,
  "extension_enabled": 9,
  "ghostery": 10,
  "headless": 11,
  "http_instrument": 12,
  "https-everywhere": 13,
  "js_instrument": 14,
  "prefs": 15,
  "random_attributes": 16,
  "save_all_content": 17,
  "save_javascript": 18,
  "tp_cookies": 19,
  "tracking-protection": 20,
  "ublock-origin": 21
}
```

0	1	2	3	4	5	6	7	8
1	False	False	firefox	False	False	False	False	False
2	False	False	firefox	False	False	False	False	False
3	False	False	firefox	False	False	False	False	False

0	9	10	11	12	13	14	15	16	17
1	True	False	True	True	False	True	{}	False	True
2	True	False	False	True	False	True	{}	False	True
3	True	False	False	True	False	True	{}	False	True

0	18	19	20	21
1	False	always	False	False
2	False	always	False	False
3	False	always	False	False

=====
 Input profile tar files
 No profile tar files specified

=====
 Output (archive) profile dirs
 No profile archive directories specified

Anexo 7. Marco regulatorio relacionado al *fingerprinting* web.

A continuación, se propone una breve interpretación argumentada del marco regulatorio relacionado a la extracción de huellas o *fingerprinting* en la web, a través de la revisión de los documentos con rango de ley, así como recomendaciones y borradores aplicables al campo de la protección y privacidad de datos.

El Grupo de trabajo del artículo 29 (WP29), creado a partir de la Directiva de Protección de Datos o Directiva 95/46/CE adoptada en el año 1995, era un cuerpo constituido por los representantes de las autoridades en materia de protección de datos de cada uno de los Estados miembros de la Unión Europea [56]. Fue reemplazado por el Comité Europeo de Protección de Datos (CEPD) el 25 de mayo de 2018, fecha en que entró en vigencia el Reglamento General de Protección de Datos (RGPD) [57].

El WP29 publicó una serie de recomendaciones y dictámenes que a menudo se convertían en regulaciones en el futuro. En lo que respecta al tema de *fingerprinting*, se destacan las siguientes publicaciones del ente mencionado:

- WP29 Dictamen 9/2014: Los terceros que procesen huellas generadas mediante procesos que consisten en adquirir acceso al terminal de un usuario o en el almacenamiento de información en el mismo, **requieren un consentimiento válido por parte de los usuarios**. También establece que **las huellas digitales de los dispositivos pueden considerarse datos personales** [58].
- WP29 Dictamen 1/2017: Prohibición de las prácticas que consisten en denegar el acceso al usuario a un sitio web o aplicaciones a menos que estos acepten en ser monitoreados. El acceso por parte del usuario a servicios en línea **no debe estar condicionado a la aceptación de una actividad de seguimiento**, como *fingerprinting*, inyección de identificadores únicos, uso de cookies, etc. [59].

Por otro lado, se destaca el reglamento de privacidad electrónica o e-Privacy, que actualmente se encuentra en forma de borrador y que busca reemplazar a la Directiva 2002/58/CE [60]. El borrador, publicado por la Comisión Europea en octubre de 2017, menciona específicamente en el considerando 20 la posibilidad de recopilar información por medio de técnicas como la “huella digital del dispositivo”, considerándola como una amenaza a la privacidad y establece que este tipo de prácticas solo deben permitirse bajo el **consentimiento del usuario e incluyendo, en todo caso, de forma específica y transparente los fines asociados** a la realización de dicha práctica [61].

Basado en las publicaciones referenciadas, se destacan las siguientes consideraciones con respecto a las prácticas de *fingerprinting* o extracción de huellas:

1. Debe existir un consentimiento explícito por parte del usuario para poder extraer una huella digital de su dispositivo en cualquier aplicación y servicio en la web donde esta práctica se realice.
2. Cuando una aplicación o sitio web solicita el consenso al usuario, debe expresar de forma detallada y transparente el contexto en que la práctica de *fingerprinting* se lleva a cabo, indicando los datos a recolectar, las técnicas de recolección y la finalidad para la que estos datos se utilizarán.
3. Permitir al usuario acceder a los servicios web en los que está interesado, incluso si de forma explícita no otorga su consentimiento a ser monitorizado o seguido por medio de técnicas de *fingerprinting*. Para los casos en que un usuario haya otorgado inicialmente su consentimiento, presentar la opción de revocar el mismo en cualquier momento [62].