**Trabajo práctico administración de sistemas y redes**

**N°2 WebRTC**

*¿Qué es WebRTC?*

Es una tecnología de código abierto que les permite a los navegadores web y a las aplicaciones móviles comunicarse en tiempo real utilizando APIs simples. WebRTC tiene varias funciones desde aplicaciones web básicas que utilizan la cámara o el micrófono hasta aplicaciones de videollamada más avanzadas y uso compartido de pantalla.

Su misión es «permitir el desarrollo de aplicaciones [RTP](https://es.wikipedia.org/wiki/Protocolo_de_transporte_en_tiempo_real) avanzadas y de alta calidad para el navegador, plataformas móviles y dispositivos de [IoT](https://es.wikipedia.org/wiki/Internet_de_las_cosas), y permitir que todos ellos se comuniquen a través de un conjunto de [protocolos](https://es.wikipedia.org/wiki/Protocolo_de_comunicaciones) comunes».

WebRTC estándar abarca, en un nivel alto, dos tecnologías diferentes: dispositivos de captura de contenido multimedia y conectividad entre pares.

Los dispositivos de captura de contenido multimedia incluyen cámaras y micrófonos, pero también la captura de pantalla y dispositivos. En el caso de las cámaras y los micrófonos, usamos navigator.mediaDevices.getUserMedia() para capturar MediaStreams. Para la grabación de pantalla, se usa navigator.mediaDevices.getDisplayMedia().

La interfaz RTCPeerConnection controla la conectividad entre pares. Este es el punto central para establecer y controlar la conexión entre dos pares en WebRTC.

*Historia de WebRTC*

Google inició la idea de crear WebRTC en 2009, como una alternativa a Adobe Flash y a las aplicaciones de escritorio que no se podían ejecutar en el navegador y fue adoptado por varios proveedores de navegadores y líderes de la industria. Un ejemplo de un servicio creado a partir de esta tecnología es Hangouts. La estandarización en W3C e IETF comenzó en 2011, y en 2013 se demostraron videollamadas entre navegadores de Mozilla y Google. Aunque hubo desafíos en la interoperabilidad debido a diferencias de implementación, estos se resolvieron en gran medida con la finalización del estándar en años posteriores.

*WebRTC: ¿Cómo funciona?*

WebRTC estándar abarca, en un nivel alto, dos tecnologías diferentes: dispositivos de captura de contenido multimedia y conectividad entre pares.

Los dispositivos de captura de contenido multimedia incluyen cámaras y micrófonos, pero también la captura de pantalla y dispositivos. En el caso de las cámaras y los micrófonos, usamos navigator.mediaDevices.getUserMedia() para capturar MediaStreams. Para la grabación de pantalla, se usa navigator.mediaDevices.getDisplayMedia().

La interfaz RTCPeerConnection controla la conectividad entre pares. Este es el punto central para establecer y controlar la conexión entre dos pares en WebRTC.

WebRTC facilita el envío y recepción de transmisiones de audio, video y datos binarios a través de DataChannel. La funcionalidad de medios, incluida la codificación y decodificación de audio y video, constituye el núcleo de WebRTC, con soporte para códecs como Opus, VP8 y H.264. Las conexiones WebRTC son siempre encriptadas mediante DTLS y SRTP. Basado en estándares existentes, WebRTC se apoya en códecs de vídeo, protocolos de red y descripción de medios, enlazando más de 50 RFC.

*Información RFC 8835 y RFC 8826*

**Especificaciones de transporte y de Middlebox**

* Interfaces proporcionadas por el sistema

Las especificaciones de protocolo utilizadas aquí suponen que los siguientes protocolos están disponibles para las implementaciones de los protocolos WebRTC:

UDP: Es el protocolo principal para WebRTC, con capacidad para establecer puntos de código de servicios diferenciados (DSCP) para la priorización de paquetes.

TCP: Utilizado para HTTP/WebSockets y para conexiones TURN/TLS e ICE-TCP.

Compatibilidad IP: Las implementaciones deben soportar tanto IPv4 como IPv6.

Acceso a interfaces: Las plataformas deben proporcionar acceso a las interfaces de estos protocolos para ser compatibles con WebRTC.

Protocolos adicionales: Incluyen TURN, STUN, ICE, TLS, y DTLS, que son esenciales para la comunicación y la seguridad en WebRTC.

* Capacidad de utilizar IPv4 e IPv6

Las aplicaciones web que usen WebRTC deben poder operar con IPv4 e IPv6.

Cuando se utiliza TURN y el servidor TURN tiene conectividad IPv4 o IPv6 con el par o con el servidor TURN del par, DEBEN admitirse candidatos de los tipos apropiados.

* Uso de direcciones IPv6 temporales

Se debe preferir el uso de direcciones IPv6 temporales sobre las permanentes para mejorar la privacidad.

En caso de direcciones temporales obsoletas, estas deben descartarse a menos que estén en uso por una conexión en curso.

* Funciones relacionadas con Middlebox

ICE: Es el mecanismo principal para tratar con NAT y firewalls, y debe ser compatible en su forma completa.

TURN: Debe ser compatible para manejar NATs y firewalls más restrictivos, y soportar tanto TCP como TLS sobre TCP.

STUN y TURN: Deben configurarse desde el navegador y la aplicación, y ser compatibles con varias extensiones y descubrimientos de servidores.

Proxies HTTP: El punto final WebRTC puede soportar acceso a través de proxies HTTP, incluyendo autenticación y encabezados específicos como "ALPN".

* Protocolos de transporte implementados

RTP seguro: Utilizado para el transporte de medios, incluyendo control de congestión y uso de un disyuntor.

DTLS-SRTP: Para el intercambio de claves en el transporte de medios.

SCTP sobre DTLS sobre ICE: Utilizado para el transporte de datos en canales de datos WebRTC.

Multiplexación: DTLS y RTP deben ser multiplexados sobre el mismo par de puertos, y todas las cargas útiles de la capa de aplicación deben ser paquetes SCTP.

Identificación de protocolo: Debe proporcionarse durante el protocolo de enlace DTLS.

**Priorización de medios**

La priorización en WebRTC es controlada a través de la API, donde la aplicación informa al punto final sobre las prioridades de los medios y datos.

Se usa el término "flujo" para referirse a las unidades a las que se les asigna una prioridad específica.

Los medios pueden ser "media flow" (flujos de media), que incluyen flujos de audio o video, considerados como "media source" (fuentes de media). Estas fuentes generan "flujos RTP de origen" y "flujos RTP de redundancia".

Se supone que todos los flujos de medios en WebRTC son interactivos ya que no hay soporte API del navegador para indicar si los medios son interactivos o no interactivos.

La prioridad asociada con un flujo de medios o un flujo de datos se clasifica como "muy baja", "baja", "media" o "alta". Sólo hay cuatro niveles de prioridad en la API.

**Priorización local**

La priorización local en WebRTC se realiza en el nodo local antes de enviar los paquetes, permitiendo un tratamiento diferenciado basado en el flujo al que pertenece cada paquete. Cuando un punto final WebRTC maneja múltiples flujos controlados por el mismo régimen de control de congestión, se debe priorizar de manera que cada flujo en un nivel de prioridad reciba el doble de capacidad de transmisión que el nivel siguiente. Por ejemplo, un flujo de alta prioridad puede enviar 8 veces más datos que uno de muy baja prioridad en condiciones de congestión.

Si hay un flujo de audio de alta prioridad y uno de video de baja prioridad con capacidad de salida para 5000 bytes, se podría enviar 4000 bytes de audio y 1000 bytes de video. En el caso inverso, con 2500 bytes de capacidad, se podrían enviar 2000 bytes de video y 500 bytes de audio.

**Consideraciones de seguridad**

* Amenazas al navegador:

Los requisitos de seguridad para WebRTC se basan en la necesidad de que el navegador proteja al usuario, permitiendo visitar sitios web arbitrarios y ejecutar sus scripts de forma segura, incluso si estos son maliciosos. Los navegadores deben diseñarse para que sea seguro interactuar con sitios maliciosos, ya que los usuarios pueden ser desviados fácilmente a ellos.

El navegador actúa como una base informática confiable (TCB), ejecutando scripts en un entorno limitado que los aísla del sistema del usuario y de otros scripts. Los atacantes pueden ser clasificados como atacantes web, que pueden inducir visitas a sus sitios pero no controlan la red, o atacantes de red, que sí controlan la red. Los navegadores usan HTTPS y TLS para proteger contra ataques de red, pero en conexiones HTTP, se debe asumir que el tráfico puede ser interceptado por atacantes.

Acceso a Recursos Locales

El navegador limita estrictamente el acceso a recursos locales como archivos, cámaras y micrófonos. Los scripts no pueden acceder a estos recursos sin el consentimiento del usuario. Por ejemplo, los formularios HTML pueden permitir la carga de archivos, pero solo si el usuario selecciona y consiente explícitamente. Programas como Flash también requieren consentimiento para acceder a la cámara y el micrófono. No se permite a los scripts ejecutar directamente archivos ejecutables.

Política del Mismo Origen (SOP)

La SOP evita que scripts de diferentes orígenes interactúen entre sí, protegiendo al usuario y a otros servidores de ataques. Los scripts pueden hacer solicitudes HTTP a otros servidores, pero no pueden leer las respuestas a menos que se utilice CORS (Cross-Origin Resource Sharing) o WebSockets, que permiten la comunicación entre orígenes bajo ciertas condiciones de verificación de consentimiento.

Omitir SOP: CORS, WebSockets y Consentimiento para Comunicarse

CORS permite que los scripts realicen solicitudes HTTP de origen cruzado tras verificar el consentimiento del servidor de destino. WebSockets, por otro lado, establecen canales de comunicación transparentes entre orígenes, permitiendo el intercambio de cualquier tráfico deseado tras una verificación de consentimiento inicial. La verificación de consentimiento es esencial para evitar que los scripts envíen datos arbitrarios a otros orígenes, pero diseñar un sistema de verificación adecuado es complejo. WebSockets incluye técnicas de enmascaramiento para dificultar la generación de tráfico malicioso que se asemeje a otros protocolos.

* Seguridad para aplicaciones WebRTC:

Acceso a Dispositivos Locales

El acceso a dispositivos locales como la cámara y el micrófono debe restringirse para evitar que sitios maliciosos molesten al usuario. No se deben iniciar llamadas sin el consentimiento del usuario, quien necesita comprender quién solicita acceso y hacia dónde se dirigen los medios.

Existen dos modelos básicos para la gestión del consentimiento:

1.El usuario envía medios a una entidad específica (p. ej., una llamada a su madre).

2.Una entidad solicita acceso a los dispositivos del usuario para transferir los medios a otra entidad.

El consentimiento se centra en la identidad de la entidad que controla los medios. Los navegadores solo pueden verificar y autorizar esta identidad, aunque los medios pueden ser redirigidos posteriormente. La protección de la privacidad y la prevención de ataques a la red local son esenciales. El consentimiento para acceder a dispositivos locales y para transmitir datos a través de la red son cuestiones distintas.

Amenazas por Compartir Pantalla

El compartir pantalla puede ser problemático, ya que los usuarios pueden compartir más de lo que pretenden, exponiendo información sensible. Además, compartir pantalla puede violar la política del mismo origen, permitiendo que sitios maliciosos vean contenido de otros sitios cargados en el navegador. Esto podría facilitar ataques como el robo de información bancaria.

Escenarios de Llamadas y Expectativas del Usuario

Servicios de Llamadas Dedicados

Los usuarios pueden preferir dar acceso a largo plazo a servicios de llamadas específicos, como una aplicación de llamadas instalada desde una tienda de aplicaciones. Sin embargo, es importante asegurar que el servicio solo realice llamadas autorizadas.

Llamar al Sitio en el que estás

El usuario puede querer iniciar una llamada desde un sitio visitado, como en los servicios de atención al cliente. Este consentimiento debe ser temporal y visible para evitar que el sitio siga activo indefinidamente sin el conocimiento del usuario.

Seguridad Basada en el Origen

Limitar el consentimiento al origen del script es crucial para proteger contra atacantes web. Sin embargo, esto puede ser insuficiente ya que los privilegios pueden acumularse y ser explotados en visitas futuras. Existen varias estrategias para mitigar esto:

* Consentimiento individual: Solicitar permiso para cada llamada.
* Consentimiento orientado al destinatario: Permitir llamadas sólo a usuarios específicos.
* Consentimiento criptográfico: Vincular llamadas a identidades establecidas criptográficamente.

Propiedades de Seguridad de la Página de Llamada

Es necesario proteger contra atacantes de red, permitir llamadas desde páginas HTTP puede exponer a los usuarios a ataques si la conexión es manipulada. Usar sitios HTTPS es una solución, pero no es infalible si se incluye contenido no confiable. El contenido activo de fuentes no confiables puede comprometer la seguridad de la llamada, incluso si la página principal es segura.

* Verificación del Consentimiento de las Comunicaciones:

Permitir que aplicaciones web accedan a la red sin restricciones puede convertir al navegador en una plataforma de ataque contra máquinas que están protegidas por firewalls o NAT. Para evitar estos ataques, es esencial que el objetivo del tráfico dé su consentimiento explícito antes de recibir cualquier dato. Esta verificación inicial no es suficiente para prevenir el uso indebido del ancho de banda, lo que refuerza la necesidad de controles adecuados de congestión en WebRTC.

ICE

El protocolo ICE verifica el consentimiento del receptor, utilizando ID de transacción generadas por el navegador que no deben estar accesibles para scripts maliciosos. ICE asegura que el receptor desea recibir tráfico específico en un momento específico mediante credenciales STUN compartidas.

Enmascaramiento

Aunque ICE verifica el consentimiento y reduce los riesgos de ataques, existen preocupaciones sobre la manipulación del texto cifrado en WebRTC sobre TCP. El SRTP minimiza estos riesgos debido a sus características distintivas, aunque aún puede ser vulnerable si se utiliza una combinación de Web Audio API y un control de tiempo estricto.

Privacidad de la Ubicación IP

El envío de candidatos ICE revela la dirección IP del llamante, lo que puede proporcionar información sobre su ubicación. Para proteger la privacidad, se puede retrasar la negociación ICE hasta que el destinatario de la llamada responda, o forzar el tráfico a través de un servidor TURN. Los sitios web pueden obtener información sobre el entorno de red del usuario incluso detrás de una VPN, por lo que las implementaciones pueden suprimir candidatos no VPN si el usuario utiliza VPNs orientadas a la privacidad como Tor.

* Seguridad de las Comunicaciones:

En WebRTC, es crucial garantizar la seguridad de las comunicaciones para proteger contra la recuperación y modificación de mensajes. Se deben utilizar mecanismos como SRTP, DTLS y DTLS-SRTP para proporcionar un canal seguro para datos, voz y video. Sin embargo, en el contexto de WebRTC, donde el servicio de llamadas controla tanto el canal como la aplicación en el navegador del usuario, se presentan desafíos adicionales.

Protección contra el Compromiso Retrospectivo

Para prevenir el acceso posterior al contenido de la llamada, es esencial implementar un intercambio automático de claves basado en clave pública y asegurarse de que el material de codificación no sea accesible a través de API directas.

Identidad de Terceros

Los sistemas de identidad de terceros basados en la web, como PKI y OAuth, pueden proporcionar una autenticación sólida de los puntos finales de las llamadas. Estos sistemas son resistentes incluso a los ataques Man-In-The-Middle (tipo de ataque en el que alguien intercepta la comunicación entre dos dispositivos conectados en la red) activos por parte del servicio de llamadas.

Acceso a los Medios

Es necesario proporcionar funciones para verificar que el sitio de llamadas no pueda interferir con los flujos de medios. Esto implica vincular criptográficamente la política de acceso a los medios y proporcionar indicadores al usuario sobre la autenticidad de los medios.

Compañeros Maliciosos

Aunque WebRTC ofrece medidas de seguridad, no puede prevenir completamente el grabado o manipulación de llamadas por parte del interlocutor. Sin embargo, las tecnologías de autenticación sólida pueden ser útiles para proteger contra llamadas no deseadas.

* Consideraciones de privacidad:

Correlación de Llamadas Anónimas

Aunque los identificadores persistentes de puntos finales pueden ser útiles para la seguridad, también representan una amenaza para la privacidad en situaciones donde el usuario desea permanecer anónimo. WebRTC proporciona identificadores persistentes como certificados DTLS y CNAME RTCP, que pueden ser utilizados para correlacionar llamadas. Para evitar esta correlación y proteger la privacidad, los navegadores deben ofrecer mecanismos para restablecer estos identificadores y permitir a los sitios crear nuevos identificadores. Además, las direcciones IP también pueden vincular llamadas, por lo que se deben considerar medidas adicionales.

*Ventajas de WebRTC*

* No requiere instalación previa: No es necesario instalar ningún programa. Simplemente inicia tu navegador, conéctate a tu espacio personal y estará listo para funcionar.
* Flexibilidad: WebRTC es una tecnología versátil que admite una variedad de casos de uso, desde simples llamadas de voz hasta videoconferencias multiparte y transmisiones en vivo. Esto permite que los empleados puedan trabajar desde cualquier lugar.
* Bajos costos: Debido a que está alojado directamente en la Web ya que no requiere el uso de servidores dedicados ni licencias costosas.
* Seguridad: WebRTC utiliza cifrado de extremo a extremo para garantizar la seguridad y privacidad de las comunicaciones.
* Compatibilidad: WebRTC es compatible con la mayoría de los navegadores web modernos, incluyendo Chrome, Firefox, Edge y Opera, lo que significa que los usuarios no tienen que instalar software adicional para utilizar aplicaciones basadas en WebRTC.

*Desventajas*

* Complejidad de desarrollo: Implementar correctamente WebRTC puede ser complejo, especialmente para desarrolladores menos experimentados. La tecnología requiere un conocimiento profundo de los protocolos de red y puede presentar desafíos en términos de configuración y gestión de conexiones.
* Interoperabilidad limitada: Aunque WebRTC es una tecnología estándar y abierta, puede haber problemas de interoperabilidad entre diferentes implementaciones o versiones de la tecnología. Esto puede dificultar la comunicación entre aplicaciones o sistemas que utilizan versiones diferentes de WebRTC.
* Rendimiento: En algunas situaciones, WebRTC puede experimentar problemas de rendimiento, especialmente en redes con ancho de banda limitado o con alta latencia. Esto puede resultar en una calidad de audio o video inferior, retrasos en la comunicación o incluso desconexiones.
* Dependencia de la red: La calidad de la experiencia de usuario en WebRTC está estrechamente relacionada con la calidad de la red. Las conexiones inestables o de baja velocidad pueden afectar negativamente la experiencia del usuario, especialmente en aplicaciones que requieren comunicación en tiempo real, como videoconferencias o juegos en línea.
* Limitaciones en la transmisión de datos: Aunque WebRTC está diseñado principalmente para la transmisión de audio y video en tiempo real, puede no ser la mejor opción para la transmisión de grandes cantidades de datos. En comparación con otros protocolos, como HTTP o FTP, WebRTC puede no ser tan eficiente para la transferencia de archivos grandes o la transmisión de datos en tiempo no real.

*Vulnerabilidades*

* WebRTC emplea UDP como protocolo de comunicación. Si bien esto simplifica la comunicación, también compromete la seguridad en términos de confiabilidad. A diferencia del TCP, que valida la recepción de datos, UDP transmite datos sin verificar si han sido recibidos por la otra parte, lo que puede generar incertidumbre sobre la disponibilidad de los datos.
* Todas las soluciones WebRTC presentan incompatibilidad entre sí. Aunque el estándar describe métodos para la transmisión de video y audio, deja aspectos como el direccionamiento de suscriptores y el intercambio de mensajes al desarrollador. Esto puede dificultar la comunicación entre diferentes aplicaciones WebRTC.
* WebRTC descubre las direcciones IP reales de los usuarios, lo que compromete su privacidad. Aunque se pueden emplear servicios VPN o servidores TURN para ocultar la IP, la revelación de la dirección IP del usuario sigue siendo inherente a WebRTC, lo que puede ser un problema en términos de anonimato.

*WebRTC Hoy en dia*

WebRTC es el segundo protocolo de vídeo más popular después del patentado por Zoom. WebRTC supera todos los demás protocolos estándar (H.323 y SIP) y patentados (Microsoft Teams y Cisco Webex).

Información:

* <https://www.ringover.es/blog/webrtc>
* <https://web.dev/articles/webrtc-standard-announcement?hl=es#:~:text=Google%20inici%C3%B3%20la%20idea%20de,sobre%20tecnolog%C3%ADa%20patentada%20con%20licencia>.
* <https://www.voiper.es/que-es-webrtc-es-segura-que-ventajas-tiene-su-uso-voiper/#:~:text=Ventajas%20de%20usar%20WebRTC&text=Fuerte%20enfoque%20de%20seguridad%3A%20todas,en%20la%20transferencia%20de%20datos>.
* <https://flussonic.com/es/blog/news/webrtc/> (vulnerabilidades)
* [https://trueconf.com/es-es/webrtc.html](https://trueconf.com/es-es/webrtc.html#:~:text=Hoy%20en%20d%C3%ADa%2C%20WebRTC%20es,Microsoft%20Teams%20y%20Cisco%20Webex)
* <https://webrtc.org/getting-started/media-devices?hl=es-419>
* <https://datatracker.ietf.org/doc/html/rfc8835> (RFC)
* <https://datatracker.ietf.org/doc/html/rfc8826#name-security-for-webrtc-applica> (RFC - consideraciones de seguridad)