TTC技術レポート
Technical Report

TR-1102

WebRTC に関する技術報告書
セキュリティ編

Technical Report on WebReal-Time Communication (WebRTC): Security

第 1 版

2023 年 07 月 13 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TTC Telecommunication Technology Committee

目次

＜参考＞

1. 国際勧告等の関連

本技術レポートは、RFC8826 および RFC8827 を調査したものである。

2. 上記国際勧告等に対する追加項目等

なし

3. 改版の履歴

版数	制定日	改版内容
第 1 版	2023 年 07 月 13 日	制定

4. 参考文献

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/RFC8826, January 2021, <https://www.rfc-editor.org/info/rfc8826>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, January 2021, <https://www.rfc-editor.org/info/rfc8827>.

5. 工業所有権

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、TTC ホームページでご覧になれます。

6. 技術レポート作成部門

第 1 版：企業ネットワーク専門委員会

I. 本技術レポートの概要

近年、テレワークの推進により、Web 会議システムが急速に普及してきた。Web 会議システムにおいてはパソコンやスマートフォン、タブレットなどデバイスを選ばず、Web ブラウザからアクセスすることにより、いつでもどこでも会議を行うことができるというメリットがある。Web 会議システムの通信プロトコルはシステムによって WebSocket であったり独自仕様であったりと様々なプロトコルが使用されている。その中でも近年特に注目されているのが WebRTC である。

WebRTC はブラウザ同士の双方向通信のために 2012 年に規格が策定され、様々な Web ブラウザで実装されてきた。その後テレワークの推進により、さらに注目を浴び、2021 年に IETF による標準化が行われた。

本報告書では TR-1095、TR-1101 に引き続き、IETF によって標準化された WebRTC に関する以下の RFC について日本語に翻訳する。

- RFC8826：WebRTC のセキュリティに関する考慮事項
- RFC8827：WebRTC セキュリティアーキテクチャ

RFC8826「ブラウザ脅威モデル」、「WebRTC アプリケーションのセキュリティ」と RFC8827 WebRTC セキュリティアーキテクチャの「詳細な技術的説明」の対応を以下に記載する。

RFC8826 3. ブラウザの脅威モデル	RFC8827 6. 詳細な技術的説明
3.1 .ローカルリソースへのアクセス	6.2. デバイス権限モデル
3.2. 同一源泉 (same-origin) ポリシー	6.1. 源泉 (origin) と Web セキュリティの問題
3.3. SOP のバイパス：CORS、WebSocket、および通信への同意	6.1. 源泉 (origin) と Web セキュリティの問題

RFC8826 4. WebRTC アプリケーションのセキュリティ	RFC8827 6. 詳細な技術的説明
4.1. ローカルデバイスへのアクセス	6.2. デバイス権限モデル
4.2. 通信同意の確認	6.3 .通信の承認 6.4. IP ロケーションプライバシー
4.3. 通信セキュリティ	6.5. 通信セキュリティ

TR-1095、TR-1101 にて翻訳した RFC を以下に記載する。

TR-1095	RFC8825	IETF によって標準化された WebRTC の仕様の概要
	RFC8834	WebRTC で使用される RTP の取り決め
TR-1101	RFC8835	WebRTC で使用されるデータ転送プロトコル
	RFC8828	WebRTC 実装における IP アドレスの処理方法 (プライバシーとメディアパフォーマンスのトレードオフの処理方法)
	RFC8831	WebRTC コンテキストにおける Stream Control Transmission Protocol (SCTP) の使用方法
	RFC8832	WebRTC データチャネル確立プロトコル

II. RFC8826 原文の著作権について

III. RFC8826 の和訳

RFC 8826　WebRTC のセキュリティに関する考慮事項

概要

WebRTC は、ブラウザに展開できるリアルタイムアプリケーション (「Web 上のリアルタイム通信」) で使用するためのプロトコルスイートある。この文書では、WebRTC 脅威モデルを定義し、そのモデルにおける WebRTC のセキュリティ脅威を分析する。

1. はじめに

Real-Time Communications on the Web (RTCWEB) Working Group は、一般に「WebRTC」[RFC8825] と呼ばれる、Web ブラウザ間のリアルタイム通信の標準化プロトコルを策定した。WebRTC テクノロジーの主なユースケースは、リアルタイムの音声/ビデオ通話、Web 会議、直接データ転送である。ほとんどの従来のリアルタイムシステム (例：SIP ベース [RFC3261] のソフトフォン) とは異なり、WebRTC 通信は何らかの Web サーバによって直接制御される。簡単なケースを以下に示す。

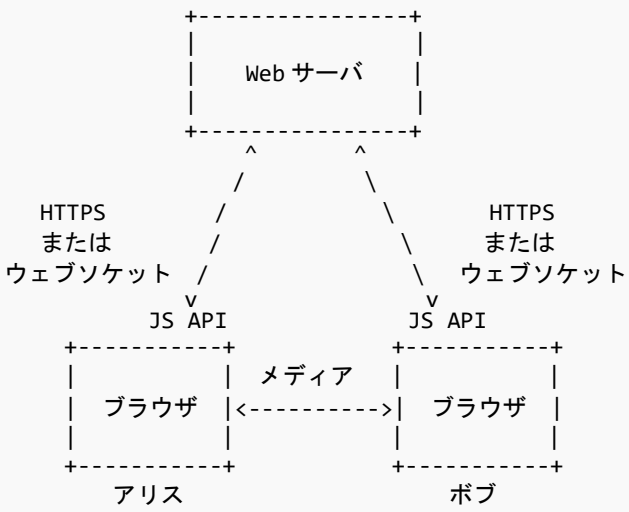


図 1：単純な WebRTC システム

図 1 のシステムでは、アリスとボブの両方が WebRTC 対応ブラウザを持っており、通話サービスを運用する何らかの Web サーバにアクセスする。それぞれのブラウザは、標準化された JavaScript (JS) 通話 API (ブラウザの組み込みとして実装) を公開しており、Web サーバがアリスとボブの間の通話をセットアップするために使用する。Web サーバは、ブラウザ間で制御メッセージを転送するためのシグナリングチャネルとしても機能する。このシステムは、従来の SIP ベースのシステム (Web サーバがシグナリングサービスとして機能し、ブラウザがソフトフォンとして機能する) にトポロジ的に似ているが、制御は中央の Web サーバに移動した。ブラウザは、通話サービスが使用する API ポイントを提供するだけである。他の Web アプリケーションと同様に、Web サーバはサーバとブラウザ内の JavaScript の間でロジックを移動できるが、コードが実行されている場所に関係なく、最終的にはサーバの制御下にある。

この種のシステムは、従来の Voice over IP (VoIP) システムを超える新しいセキュリティ上の課題を提起していることがすぐに明らかになるはずである。特に、悪意のある通話サービスに対処する必要がある。例えば、通話サービスによってブラウザが任意の呼び出し先に対していつでも呼び出しを行うことができ

る場合、この機能を使用して、ユーザが知らないうちに、何らかの録音サービスに呼び出しを行うだけで、ユーザのコンピュータにバグを発生させることができる。もっと微妙に言えば、公開された API によって、サーバがブラウザに任意のコンテンツを送信するように指示できるようになれば、ファイアウォールを迂回したり、サービス拒否 (DoS) 攻撃を仕掛けたりするのに利用できる。成功するシステムは、このような攻撃に対して耐性を持つ必要がある。

関連文書 [RFC8827] では、この文書で提起された問題に対処することを目的としたセキュリティアーキテクチャについて説明している。

2. 言語

この文書のキーワードである「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「NOT RECOMMENDED」、「MAY」、および「OPTIONAL」は、ここに示すように、すべて大文字で表示される場合にのみ、BCP 14 [RFC2119] [RFC8174] に記載されているように解釈される。

3. ブラウザの脅威モデル

WebRTC のセキュリティ要件は、ブラウザのジョブがユーザを保護するという要件に直接従っている。Huang et al. [huang-w2sp] は、コアなブラウザのセキュリティ保証を次のようにまとめている。

ユーザは、任意の Web サイトに安全にアクセスし、それらのサイトが提供するスクリプトを実行できる。

これには、任意の悪意のあるスクリプトをホストするサイトも含まれることを認識することが重要である。この要件の動機は単純であり、攻撃者がユーザを選択したサイトに誘導することは簡単である。例えば、攻撃者はディスプレイ広告を購入して、ユーザを自分のサイトに誘導することができ (自動またはユーザクリックで)、その時点でブラウザは攻撃者のスクリプトを実行する。そのため、任意の悪意のあるページでも安全に閲覧できることが重要である。もちろん、ブラウザには必然的にこの目標を達成できないバグがあるが、新しい WebRTC 機能はこの標準を満たすように設計されなければならない。この章の残りの部分では、既存の Web セキュリティモデルの背景について詳しく説明する。

このモデルでは、ブラウザはユーザから見てもサーバから見てもある程度は Trusted Computing Base (TCB) として機能する。サーバによって提供される HTML と JavaScript は、ブラウザにさまざまなアクションを実行させる可能性があるが、これらのスクリプトは、以下に詳述するように、ユーザのコンピュータと相互に分離するサンドボックスで動作する

従来は、ユーザを誘導してサイトを訪問させるがネットワークを制御できない Web 攻撃者と、ネットワークを制御できるネットワーク攻撃者のどちらかを指していた。ネットワーク攻撃者は [RFC3552] の「インターネット脅威モデル」に相当する。整合性保護を提供しないトランスポートプロトコルでは、ネットワークが通信ピアであるかのようにトラフィックを注入できるため、場合によってはネットワーク攻撃者も Web 攻撃者であることに注意すること。TLS、特に HTTPS はこれらの攻撃を防ぐが、HTTP 接続を分析する場合は、トラフィックが攻撃者に向かっていることを前提とする必要がある。

3.1. ローカルリソースへのアクセス

ブラウザはキー情報、ファイル、カメラ、マイクなどのローカルリソースにアクセスできるが、Web サーバがそれらの同じリソースにアクセスすることを厳しく制限または禁止する。例えば、ファイルのアップロードを許可する HTML フォームを作成することは可能であるが、スクリプトはユーザの同意なしに実

行することはできず、実際には特定のファイル (例：/etc/passwd) を提案することさえできない。ユーザは明示的にファイルを選択し、アップロードに同意する必要がある。(注意：多くの場合、ブラウザは「ここをクリックしてセキュリティチェックを回避する」という意味のダイアログを避けるように明示的に設計されている。広範な調査 [cranor-wolf] によると、このような状況ではユーザは同意する傾向があることがわかっている。)

同様に、Flash プログラム (SWF) [SWF] はカメラとマイクにアクセスできるが、それらは明示的にユーザがそのアクセスに同意する必要がある。さらに、一部のリソースはブラウザからまったくアクセスできない。例えば、特定の実行ファイルをスクリプトから直接実行する現実的な方法は存在しない (もちろん、実行ファイルをダウンロードして実行するように誘導することはできる)。

3.2. 同一源泉 (same-origin) ポリシー

他の多くのリソースはアクセス可能であるが、分離されている。例えば、スクリプトは fetch() API を介して HTTP 要求を行うことができるが ([fetch] を参照)、要求がスクリプトの送信元と同一源泉 (same-origin) 以外のサーバに対して行われた場合 [RFC6454]、応答を読み取ることはできない。Cross-Origin Resource Sharing (CORS) [fetch] および WebSocket [RFC6455] は、後述するように、この制限からのエスケープハッチを提供する。この同一源泉 (same-origin) ポリシー (SOP) は、サーバ A がユーザのブラウザを介してサーバ B に攻撃を仕掛けることを防ぎ、ユーザ (例えば、資格情報の不正使用など) とサーバ B (例えば、DoS 攻撃など) の両方を保護する。

より一般的には、SOP は各サイトのスクリプトを独自の分離されたサンドボックスで実行するように強制する。相互作用を可能にする技術はあるが、それらの相互作用は一般に (各サイトによる) 相互合意である必要があり、特定のチャネルに限定される。例えば、同一源泉 (same-origin) の複数のページ/ブラウザペインは互いの JS 変数を読み取ることができるが、異なる源泉 (origin) のページ (または同じページの異なる源泉 (origin) の IFRAME でさえ) は読み取ることができない。

3.3. SOP のバイパス：CORS、WebSocket、および通信への同意

SOP は重要なセキュリティ機能を果たす一方で、特定のクラスのアプリケーションを記述することを不便にしている。特に、源泉 A (origin A) のスクリプトが源泉 B (origin B) のリソースを使用するマッシュアップは、ある程度のハッキングでしか実現できない。W3C CORS spec [fetch] はこの要求に応えたものである。CORS では、源泉 A (origin A) からのスクリプトが安全でない可能性のある他源泉 (cross-origin) 要求を実行すると、ブラウザは代わりにターゲットサーバに接続して、A からの他源泉 (cross-origin) 要求を許可するかどうかを判断する。その場合、ブラウザは要求を許可する。この同意確認プロセスは、他源泉 (cross-origin) 要求を安全に許可するように設計されている。

CORS は他源泉 (cross-origin) HTTP 要求を許可するように設計されているが、WebSockets [RFC6455] は他源泉 (cross-origin) 透過チャネルの確立を許可する。スクリプトからサイトへの WebSocket 接続が確立されると、スクリプトは、一連の HTTP 要求/応答トランザクションとしてフレーミングする必要なく、任意のトラフィックを交換できる。CORS と同様に、WebSockets トランザクションは、スクリプトが別の源泉 (origin) に任意のデータを送信することを許可しないように、同意検証段階から開始される。

同意の検証は概念的には単純であり、実際のデータの交換を開始する前にハンドシェイクをするだけであるが、正しい同意の検証システムを設計することは困難であることが経験から示されている。特に、Huang et al. [huang-w2sp] は、既存の Java と Flash の同意検証技術、および WebSocket ハンドシェイクの簡易版に脆弱性があることを示した。CROSS-PROTOCOL 攻撃では、攻撃スクリプトが Web プロトコル以外のステートマシンに受け入れられるトラフィックを生成するため、注意が必要である。この形式の攻撃

に対抗するため、WebSocket には、ネットワーク上のビットをランダム化することを目的としたマスキング手法が組み込まれているため、特定のプロトコルに似たトラフィックを生成することがより困難になっている。

4. WebRTC アプリケーションのセキュリティ

4.1. ローカルデバイスへのアクセス

1 章で説明したように、任意のサイトが通話を開始することを許可することは、中核となる Web セキュリティ保証に違反する。ローカルデバイスへのアクセス制限がなければ、悪意のあるサイトはユーザをバグで攻撃するだけである。少なくとも、任意のサイトがユーザの同意なしに任意の場所への通話を開始してはならない[MUST NOT]。しかし、ユーザの同意の範囲をどうすべきかという問題がすぐに生じる。

ユーザが通話を許可するかどうか (したがって、カメラとマイクの入力がどこかにルーティングされるかどうか) をインテリジェントに判断するためには、誰がアクセスを要求しているか、メディアがどこに向かっているか、またはその両方を理解している必要がある。以下に詳述するように、2 つの基本的な概念モデルがある。

1. エンティティ A (例えば、あなたの母親) と通信するため、メディアをエンティティ A に送信している。
2. エンティティ A (例えば、通話サービス) は、メディアをエンティティ B (例えば、あなたの母親) に転送することを保証して、ユーザのデバイスにアクセスするように求める。

いずれの場合も、同意決定の中心には ID がある。さらに、ブラウザが意味のある形で強制できるのは、ブラウザが接続している相手の ID だけである。A に通話する場合、A は単にメディアを C に転送できる。同様に、A が B に通話する権限を与えると、A は代わりに C に通話することができる。どちらの場合も、ブラウザができることは、メディアの行く先を制御している人の承認を検証してチェックすることだけである。メディアのターゲットはもちろんセキュリティ/プライバシーポリシーをアドバタイズできるが、これはブラウザが強制できるものではない。それでも、さまざまな技術的同意メカニズムの動機となるさまざまな異なる同意シナリオがある。これらのメカニズムについては、以下の節で説明する。

ローカルデバイスへのアクセスに対する同意は、さまざまな種類のデータをネットワーク経由で送信することに対する同意とほぼ直交していることを理解することが重要である (4.2 節参照)。デバイスアクセスの同意は、主に悪意のあるサイトからユーザのプライバシーを保護する問題である。対照的に、ネットワークトラフィックの送信に同意することは、ユーザのブラウザがローカルネットワークへの攻撃に使用されるのを防ぐことである。そのため、サイトがカメラやマイクに全くアクセスできない場合でも、通信の同意を得る必要があり (したがって、WebSocket の同意メカニズム)、同様に、HTTP POST などの従来の HTTP ベースのネットワークメカニズムを介してデータをサイトに送り返す場合でも、サイトがユーザのカメラやマイクにアクセスすることを考慮する必要がある。

4.1.1. 画面共有による脅威

カメラやマイクへのアクセスに加えて、画面やアプリケーションの共有機能に対する需要があった。残念ながら、この機能のセキュリティへの影響は、カメラやマイクへのアクセスよりもユーザが直感的に分析することがはるかに困難である。 (詳細な分析については、
<https://lists.w3.org/Archives/Public/public-webrtc/2013Mar/0024.html> を参照すること。)

最も高い脅威は、単純に「過剰共有」の脅威である。つまり、ユーザは実際にはアプリケーションを共

有しているのに、ウィンドウを共有していると思い込んだり、画面全体、アイコン、通知などすべてを共有していることを忘れたりする可能性がある。これは、既存の画面共有技術ではすでに問題となっており、部分的に信頼されたサイトが、ユーザに提案させるのではなく、リソースの共有を求める責任がある場合は、やや悪化する。

低い脅威としては、画面共有が Web セキュリティモデルに与える影響がある。同一源泉(same-origin) ポリシーの重要な部分は、サイト A の HTML または JS がサイト B のコンテンツを参照し、ブラウザにそれをロードさせることができるが、 (明示的に許可されていない限り) 結果を見ることはできないということである。しかし、サイトからの Web アプリケーションがブラウザを画面共有している場合、これはこの不変条件に違反し、深刻なセキュリティ上の影響を及ぼす。例えば、攻撃者のサイトが画面共有を要求した後、ユーザの銀行または Web メールアカウントに新しいウィンドウを一時的に開き、画面共有を使用して結果として表示されるコンテンツを読み取る場合がある。より洗練された攻撃は、ソース表示ウィンドウをサイトに開き、画面共有の結果を使用して、アンチクロスサイト要求偽造トークンを表示することである。

これらの脅威は、画面/アプリケーションの共有には、カメラやマイクへのアクセスよりも高いレベルのユーザの同意が必要である可能性を示唆している。

4.1.2. 呼び出しシナリオとユーザの期待

可能な呼び出しシナリオは多数あるが、この節で説明するシナリオは、関連する同意の範囲を特定することの難しさの多くを示している。

4.1.2.1. 専用通話サービス

最初に考えるシナリオは、専用の通話サービスである。この場合、ユーザは通話サイトと関係を持ち、そのサイトで繰り返し通話を行う。ユーザは、各通話に許可を与えるのではなく、カメラとマイクへの長期的なアクセスを通話サービスに与えることを望む可能性がある。これは、長期的な同意メカニズム (例えば、通話サービスの許可を示すアプリストア「アプリケーション」をインストールする) に自然に適合する。専用通話サービスの変形として、ゲームサイト (例：ポーカーサイト) があり、プレイヤー同士が通話できる専用通話サービスをホストしている。

ユーザが同じサービスを使用して多くの異なる人と会話する可能性のあるどのような種類のサービスでも、ユーザが誰と話しているかを知ることができるかどうかという問題がある。サービス A に通話し自分の代わりに通話する許可を与えると、コンピュータが望むときにいつでもバグを起こす許可を暗黙的に与えていることになる。これは、サイトが特定のターゲットエンティティ (4.3.2 項および特に 4.3.2.3 項で説明されているように、メディアプレーン暗号メカニズムを介して特定される) に対してのみ通話を行う権限を与えられる別の同意モデルを示唆している。

ここでの同意の問題は、ピア ID の問題とは関連しているが、別の問題であることに注意してすること。一般的に、通話サイトが私に代わって通話を開始することを許可することもできるが、サイトがリッスンしていないことを確認できるサイト経由の通話もある。

4.1.2.2. 今いるサイトに通話する

もう 1 つの簡単なシナリオは、実際に訪問しているサイトに通話することである。ここでのパラダイムは、多くのショッピングサイトに表示される「ここをクリックして代表者と話す」ウィンドウである。この場合、ユーザは実際に訪問しているサイトに通話していることを期待する。しかし、彼らがそのようなサイトに一般的な同意を提供したいと考える可能性は低い。車の情報が欲しいからといって、いつでも好

きな時にマイクを起動できるようにしてほしいということではない。したがって、これは、特定の通話の間だけ同意を与える第 2 の同意メカニズムの必要性を示唆している。3.1 節で述べたように、ユーザがクリックスルーするのを避けるために、このインタフェースの設計には細心の注意を払う必要がある。また、ユーザがユーザエージェント自体と対話するための表現であるユーザインタフェースクロームは、通話サイトが無期限に放置しているだけで、そうでないことを意味する Web UI を表示する攻撃を避けるために、通話が継続していることを示す要素を明確に表示する必要があることに注意すること。

4.1.3. 源泉ベース (Origin-Based) のセキュリティ

呼び出しシナリオについて説明したので、セキュリティ要件についての理由付けを開始できる。

3.2 節で説明したように、Web サンドボックスの基本単位は源泉 (origin) であるため、源泉 (origin) への同意の範囲は当然である。具体的には、源泉 A (origin A) からのスクリプトは、ユーザがその源泉 (origin) に対して明示的にアクセスを許可している場合にのみ、通信の開始を許可されなければならない (したがって、カメラとマイクにアクセスすることも許可されなければならない) [MUST]。もちろん、技術的にはよりスコープの粗いアクセス許可を持つことが可能であるが、Web モデルのスコープは源泉 (origin) に設定されるため、これは困難なミスマッチを生み出す。

おそらく、源泉 (origin) は十分に細かい粒度ではない。アリスがサイトにアクセスし、1 回の通話を許可する状況を考えてみる。同意が源泉 (origin) に関してのみ表明された場合、その後そのサイトにアクセスすると (マッシュアップや広告ネットワーク経由で誘導されたものを含む)、そのサイトはアリスのコンピュータにバグを仕掛けたり、コンピュータを使って偽の通話ができるが、原則としてアリスは特権を付与してから取り消すことができるが、実際には特権は蓄積される。もし我々がこの攻撃を懸念しているなら、他の何かが必要である。この種の問題には多くの対策が考えられる。

個人の同意

呼び出しごとにユーザに許可を求める。

着信者指向の同意

特定のユーザへの呼び出しのみを許可する。

暗号化の同意

特定のピアキーイングマテリアルのセットまたは暗号で確立された ID への呼び出しのみを許可する。

残念ながら、これらのアプローチはいずれもすべてのケースで満足できるものではない。前述のように、個々の同意は、すべての通話の UI フローにユーザの承認を入れる。これはすぐに煩わしくなるだけでなく、ユーザが単に「OK」をクリックするように仕向けることができ、その時点で同意は役に立たなくなる。したがって、場合によっては個別の同意が必要になることがあるが、これは (例えば) 通話サービスのケースに適した解決策ではない。必要に応じて、インフローユーザインタフェースは、ユーザが盲目的にクリックスルーするリスクを避けるように慎重に設計する必要がある。

他の 2 つのオプションは、特定のターゲットへの呼び出しを制限するように設計されている。悪意のあるサイトは、ユーザが任意のユーザを呼び出していると主張できるため、通話サイトによって提供される着信者指向の同意はうまく機能しない。この問題の解決策の 1 つは、暗号で確立された ID に呼び出しを関連付けることである。すべてのケースに適しているわけではないが、この方法が役立つ場合もある。広告の場合を考えると、広告主が許可を得るためだけにホスティングサイトで IFRAME をインスタンス化することを要求するのは、特に便利ではない。より便利な方法は、広告主の証明書を通信に直接暗号で結びつけることである。ここではアクセス許可を源泉 (origin) に結び付けているが、Web 源泉 (web origin)

ではなくメディア源泉 (media origin) (および/または宛先) に結び付けている。[RFC8827] には、この種の同意を容易にするメカニズムが記載されている。

メディアレベルの暗号化 ID が意味を持つもう 1 つのケースは、ユーザが本当に通話サイトを信頼していない場合である。例えば、通話サービスがコンピュータにバグを起こそうとするのではないかと心配になるかもしれないが、友人に便利に通話できるようにしたいとも思っている。同意が特定の通信エンドポイントに結びついている場合、リスクは限定される。当然ながら、この種のポリシーを表現する UI プリミティブの設計はやや困難である。この問題は、マルチユーザ通話のケースではさらに困難になる。

4.1.4. 呼び出し元ページのセキュリティプロパティ

源泉ベース (Origin-Base) のセキュリティは、Web 攻撃者に対するセキュリティを目的としている。ただし、ネットワーク攻撃者の場合も考慮する必要がある。例えば、<http://calling-service.example.com> のように、HTTP スキームを持つ源泉 (origin) によって通話サービスに許可を与えた場合を考えてみる。セキュリティで保護されていないネットワーク (例：ホットスポット、または自宅のワイヤレスネットワークが安全でない場合) でコンピュータを使用し､任意の HTTP サイトを参照した場合､攻撃者はコンピュータにバグを作成できる。攻撃は次のように進行する。

1. <http://anything.example.org/> に接続する。このサイトは、通話サービスとは無関係であることに注意すること。
2. 攻撃者は、私の HTTP 接続を変更して IFRAME (またはリダイレクト) を <http://calling-service.example.com> に注入する。
3. 攻撃者は､<http://calling-service.example.com/> からの応答を偽造して JS を挿入し、自身への呼び出しを開始する。

この攻撃は、メディアがセキュアでないことに依存しないことに注意すること。攻撃者に対する通話であるため、攻撃者に対しても暗号化される。さらに、すぐに実行する必要はない。攻撃者は、半永久的に (例えば、Web ワーカーの場合や、メインウィンドウの下に隠れているポップアップウィンドウの場合など) 源泉 (origin) を「感染」させることができるため、感染したネットワークから離れてから長い時間が経ってもバグを起こすことができる。このリスクは、HTTP 経由で取得されたページからの呼び出しを全て許可することで発生する。

HTTPS [RFC2818] サイトからの呼び出しのみが可能であっても、それらのサイトに信頼されていないサイトからのアクティブなコンテンツ (例：JavaScript) が含まれている場合、その JavaScript はページのセキュリティコンテキストで実行される。これにより、親ページが安全であっても、呼び出しが侵害される可能性がある。注意：この問題は、信頼できないコンテンツを含むページに限定されない。特定の源泉 (origin) のページが攻撃者から JavaScript をロードされた場合、その攻撃者がブラウザのその源泉 (origin) の概念に半永久的に感染する可能性がある。

4.2. 通信同意の確認

3.3 節で説明されているように、Web アプリケーションにブラウザを介した無制限のネットワークアクセスを許可すると、トポロジ的に制限されているため (例：ファイアウォールや NAT の背後)、本来なら悪意のあるサイトにアクセスできないマシンに対する攻撃プラットフォームとしてブラウザを使用するリスクが生じる。この形式の攻撃とクロスプロトコル攻撃を防止するには、トラフィックのターゲットが問題のトラフィックを受信することに明示的に同意する必要がある。特定のエンドポイントに対してその同意が検証されるまで、そのエンドポイントに同意ハンドシェイク以外のトラフィックを送信してはなら

ない[MUST NOT]。

同意の検証は、ネットワークリソースの過剰使用を防ぐには十分ではない。WebRTC を使用すると、Web サイトはユーザの同意なしに 2 つのブラウザインスタンス間のデータフローを作成できるため、悪意のあるサイトは、別のユーザにそのようなチャネルを設定することで、大きなコストをかけずにユーザの帯域幅を大量に消費する可能性がある。しかし、実際問題として、データソースとして機能する Web サイトが多数存在するため、攻撃者は少なくとも既存の Web API でダウンリンク帯域幅を使用できる。しかし、この潜在的な DoS ベクトルは、WebRTC プロトコルがユーザの帯域幅に対する他の要求と公平に一致することを保証するために、適切な輻輳制御の必要性を強化する。

4.2.1. ICE

受信者の同意を確認するには、何らかの明示的なハンドシェイクが必要であるが、NAT ホールパンチングを行うには、便利なことに、すでにハンドシェイクが必要である。Interactive Connectivity Establishment (ICE) [RFC8445] には、受信要素が送信者からのトラフィックの受信を希望していることを確認するためのハンドシェイクが含まれている。ここで忘れてはならないのは、ICE を開始するサイトは悪意があると推定されるということである。ハンドシェイクを安全にするために、受信要素は、サイトで使用できない何らかの値の受信/知識を証明する必要がある[MUST]。 (これにより、サイトが応答を偽造するのを防ぐことができる)。ICE でこの目的を達成するために、Session Traversal Utilities for NAT (STUN) トランザクション ID は、ブラウザによって生成される必要があり、診断インタフェースを介しても、開始スクリプトで使用できるようにしてはならない[MUST NOT]。 受信者の同意を確認するには、受信者が特定の送信者からのトラフィックを受信する必要があることも確認する必要がある。例えば、悪意のあるサイトは、他のセッションに ICE を使用している既知のサーバに対して、単に ICE を試みる可能性がある。ICE は、STUN クレデンシャルをセッション共有秘密の形式として使用することで、この検証も提供する。これらの資格情報は Web アプリケーションで認識されているが、STUN 受信要素でも認識され、使用されている必要がある。

また、トラフィックのターゲットが引き続き受信を希望していることをブラウザが確認するためのメカニズムも必要である。ICE キープアライブは指示であるため、ここでは機能しない。

[RFC7675] は、同意の鮮度を提供するメカニズムを記述している。

4.2.2. マスキング

いったん同意が確認されると、Huang et al. [huang-w2sp] によって説明されているように、誤解による攻撃が懸念される。ICE ハンドシェイクは受信者の同意を強制し、Huang が研究したタイプのパッシブ DTLS プロキシの証拠はほとんどないため、これは DTLS には深刻な問題ではないと思われる。ただし、RTCWEB は TCP 上で動作するため、攻撃者が SCTP へのプレーンテキスト入力を制御することによって暗号文を制御する可能性があるという懸念がある。このリスクは、SCTP スタックがパケットのフレーミングを制御するという事実によって部分的に軽減されるだけである。

原則として、攻撃者は WebAudio API と非常に厳密なタイミング制御の組み合わせを使用して、Secure Real-time Transport Protocol (SRTP) パケットをある程度制御できることに注意する必要がある。ここでの主なリスクは、NAT (TURN) TCP の周りのリレーを使用したトラバーサル上で SRTP を運ぶことと思われる。しかし、SRTP パケットには非常に特徴的なパケットヘッダーがあるため、最も攻撃的な仲介者以外が、別のアプリケーション層プロトコルが使用されていると勘違いする可能性は低いと思われる。

4.2.3. 下位互換性

注意：RTCWEB WG は最終的に ICE を要求することを決定した。ここでは、その決定の背景について説明する。

ICE を使用する必要があるため、従来の非 ICE クライアントとの互換性が制限される。いくつかのチェックの要件を完全に削除するのは安全ではないと思われる。提案されたすべてのチェックには、ブラウザが候補トラフィックの受信者に何らかのメッセージを送信し、そのメッセージが返信されるまで他のトラフィックの送信を拒否するという共通の機能がある。メッセージ/応答ペアは、Web アプリケーションを制御する攻撃者が偽造できないような方法で生成する必要がある。一般的には、メッセージに組み込む必要のある秘密の値 (例えば、echo、hashed into など) が含まれている必要がある。このロールの ICE 以外の候補 (レガシーエンドポイントにパブリックアドレスがある場合) には、次のものがある。

- ICE を使用しない STUN チェック（つまり、非 RTC Web エンドポイントは STUN レスポンダを設定する）。
- 暗黙的な到達可能性チェックとしての RTP Control Protocol (RTCP) の使用。

RTCP アプローチでは、WebRTC エンドポイントは、同意を受け取る前に限られた数の RTP パケットを送信できる。これにより、短い攻撃ウィンドウが可能になる。さらに、一部のレガシーエンドポイントは RTCP をサポートしていないため、このようなエンドポイントでは非常に高価なソリューションとなり、ICE の実装が容易になる可能性がある。これらの 2 つの理由から、RTCP ベースのアプローチではセキュリティの問題に十分に対処できないと思われる。

STUN アプローチでは、WebRTC エンドポイントは、受信者が何らかの STUN エンドポイントを実行していることを確認できるが、STUN 応答側が ICE ユーザ名/パスワード確立システムと統合されていない限り、WebRTC エンドポイントは、受信者がこの特定の呼び出しに同意していることを確認できない。これは、既存の STUN サーバが帯域幅ベースの攻撃を処理できないアドレスで運用されている場合に問題になる可能性がある。したがって、このアプローチも満足できるものとは思えない。

システムが緊密に統合されている場合 (つまり、STUN エンドポイントは ICE クレデンシャルで認証された応答で応答する)、この問題は発生しない。ただし、このような設計は ICE-Lite の実装に非常に近い (確かに、間違いなくそのものである)。中間的なアプローチとしては、WebRTC チェックに応答しているが ICE クレデンシャルに基づく整合性チェックを計算していないことを示す STUN 拡張を使用することが考えられる。これにより、レガシーSTUN サーバと混同するリスクなしにスタンドアロン STUN サーバを使用できるようになる。ICE 以外のレガシーソリューションが必要な場合は、これがおそらく最適な選択である。

最初の同意が確認されたら、継続的な同意も確認する必要がある。これは、2 人のユーザが一時的に IP を共有し (例：インターネットカフェの NAT の背後)、攻撃者がネットワークへの大規模で停止不可能なトラフィックフローを手配してから離れる攻撃を避けるためである。ここでの適切な技術は、最初の同意のための技術とかなり似ているが、脅威がそれほど深刻ではないため、おそらく弱い。

4.2.4. IP ロケーションプライバシー

着信者が ICE 候補を送信するとすぐに、発信者は着信者の IP アドレスを学習することに注意する必要がある。着信者のサーバ再帰アドレスは、着信者の位置に関する多くの情報を明らかにする。追跡を避けるために、実装では、着信者が応答するまで ICE ネゴシエーションの開始を抑制したい場合がある。さらに、どちらの側も、すべてのトラフィックを強制的に TURN サーバ経由にすることで、相手側から自分の位置を完全に隠したい場合がある。

通常の運用では、サイトはブラウザの IP アドレスを学習するが、Tor <https://www.torproject.org> や VPN などのメカニズムによって隠されることもある。ただし、サイトによってブラウザが IP アドレスを提供する可能性があるため、これにより、ユーザが IP アドレスをマスクする VPN の背後にいる場合でも、サイトがユーザのネットワーク環境について学習するメカニズムが提供される。

実装では、ユーザが特定の種類の VPN、特に Tor などのプライバシー指向システムを使用している場合に、すべての非 VPN 候補を抑制する設定を提供したい場合がある。詳細は [RFC8828] を参照。

4.3. 通信セキュリティ

最後に、SIP の世界でよく知られている問題である通信セキュリティについて考察する。明らかな理由により、通信当事者は、メッセージ回復とメッセージ変更の両方に対して安全なチャネルを確立することが可能でなければならない[MUST] (詳細は [RFC5479] を参照)。このサービスは、データと音声/ビデオの両方に対して提供される必要がある。理想は、両方のタイプのコンテンツに同じセキュリティメカニズムを使用する。このサービスを提供する技術 (例えば、SRTP [RFC3711]、DTLS [RFC6347]、DTLS-SRTP [RFC5763]) はよく理解されている。ただし、脅威モデルが多少異なる WebRTC のコンテキストで、この技術を検証する必要がある。

一般に、従来の SIP プロキシとは異なり、通話サービス (Web サーバ) は、通信エンドポイント間のチャネルだけでなく、ユーザのブラウザ上で実行されるアプリケーションも制御することを理解することが重要である。原理的には、ブラウザが通話サービスをループから切り離し、信頼できる情報を直接提示する (そしておそらく同意を得る) ことは可能であるが、最近のブラウザでは、可能な限りこれを避けることが実践されている。特定の行動への同意をユーザに要求する「In-flow」モーダルダイアログは、特に好まれていない。なぜなら、ヒューマンファクターの研究によると、それらが著しく侵害的でない限り、ユーザが実際には意識的に同意を与えることなく、単にそれらに同意するからである。したがって、ほとんどすべての UI はブラウザによってレンダリングされる必要があるが、通話サービスの制御下にある。これには、ピアの ID 情報が含まれている可能性がある。これは結局のところ、何らかの通話サービスのコンテキストでのみ意味を持つ。

この制限は、通話サービスによる攻撃を防ぐことが完全に絶望的であることを意味するものではない。ただし、次の 2 つの攻撃クラスを区別する必要がある。

通話サービスの遡及的な妥協:

通話サービスは、通話中は悪意のないサービスであるが、その後に侵害され、古い通話を攻撃しようとする (「パッシブ攻撃」と呼ばれる)。

通話サービスによる通話中の攻撃:

攻撃対象の通話中に、通話サービスが侵害される (「アクティブ攻撃」と呼ばれる)。

前者のタイプの攻撃に対するセキュリティの提供は、4.3.1項で説明されている手法を使用して実用的である。しかし、中間者攻撃 (MITM) を仕掛けたり、通話を完全に迂回させたりすることによって、信頼できるが悪意のある通話サービスが積極的にユーザの通話を攻撃するのを防ぐことは非常に困難である。(この攻撃は、通話サービスへの通信がセキュリティで保護されていない場合、ネットワーク攻撃者にも同様に適用されることに注意する必要がある)。いくつかの潜在的なアプローチについては、4.3.2項で説明する。

4.3.1. 遡及的妥協からの保護

レトロスペクティブ攻撃では、通話中に通話サービスが侵害されなかったが、その後、攻撃者は通話の

内容を回復しようとする。攻撃者は、保護されたメディアストリームにアクセスできるだけでなく、通話サービスの完全な制御権を持っていると仮定する。

通話サービスがトラフィックの鍵材料にアクセスできる場合 (Security Descriptions (SDES) [RFC4568] など)、レトロスペクティブ攻撃は些細なことである。Web サービスでは広範なログと監視を実行するのが標準的な手法であるため、この形式の攻撃は Web コンテキストで特に深刻である。したがって、トラフィックキーが HTTP 要求の一部である場合、どこかでログに記録され、その後の侵害の対象となる可能性が高くなる。WebRTC に自動の公開鍵ベースの鍵交換メカニズムを必須とするのはこの考慮事項であり (これはあらゆる通信セキュリティシステムに適したアイデアである)、このメカニズムは Forward Secrecy (FS) を提供すべきである[SHOULD]。このメカニズムの認証には、シグナリングチャネル/通話サービスを使用できる。

さらに、エンドツーエンドの鍵が使用されている場合、システムは、長期的な鍵材料を抽出したり、保存されているトラフィックキーに直接アクセスしたりするための API を提供してはならない[MUST NOT]。そうしないと、その後に通話サービスを侵害した攻撃者が、これらの API を使用してトラフィックキーを回復し、トラフィックを侵害できる可能性がある。

4.3.2. 通話中攻撃からの保護

通話中の攻撃から保護することは、より困難な提案である。 (前項で推奨されているように) 通話サービスが鍵材料に直接アクセスできない場合でも、ボブとボブに電話していることをアリスに伝え、実際には通話サービスが通話ブリッジとして機能し、すべてのトラフィックをキャプチャすることで接続に対して中間者攻撃を仕掛けることができる。この形式の攻撃から保護するには、明示的な帯域外鍵検証 (例えば、フィンガープリントによるもの) や、[RFC8827] で説明されているサードパーティの ID サービスなど、リモートエンドポイントの肯定的な認証が必要である。

4.3.2.1. 鍵の連続性

自然なアプローチの 1 つは、「鍵の連続性」を使用することである。悪意のある通話サービスは、選択した任意の ID をユーザに提示できるが、特定の公開鍵にマップする秘密鍵を生成することはできない。したがって、ブラウザは特定ユーザの公開鍵を記録し、そのユーザの鍵が変更されるたびにアラームを生成することができる。Secure Shell (SSH) プロトコル [RFC4251] も同様の手法を使用している。 (すべての通話で明示的なユーザの同意を避ける必要があるため、ブラウザはピアの鍵をすぐに手動でチェックする必要がないことに注意が必要である)

残念ながら、この種の鍵の連続性メカニズムは WebRTC のコンテキストではあまり役に立たない。まず、WebRTC (及びあらゆる Web アプリケーション) の長所の多くは、特定のクライアントソフトウェアに縛られないことである。したがって、ユーザが異なるコンピュータ上で複数のブラウザを使用することが可能なだけでなく、当然ながら異なる鍵材料を持つことになる (Secure Available Credentials (SACRED) [RFC3760] にもかかわらず)。そのため、ユーザは実際には完全に正当な鍵の不一致に頻繁に警告され、その結果、単にそれらをクリックするように訓練される。ユーザが日常的にはるかに悲惨な警告 [cranor-wolf] をクリックすることが知られているように、鍵の連続性メカニズムが単に煩わしいだけでなく有効である可能性は極めて低いと思われる。

さらに、この種のメカニズムでさえバイパスするのは簡単である。SSH の場合とは異なり、ブラウザはピアの ID をユーザから直接取得することはない。代わりに、通話サービスによって提供される。このタイプのメカニズムを有効にする場合でも、通話サービスがブラウザに「これはユーザ X への通話です」と通知できるようにする API が必要になる。鍵の継続性の警告が発生しないようにするために必要なのは、

Y が X と紛らわしい場合に、「これはユーザ Y への通話です」とブラウザに伝えることだけである。ユーザが実際に相手の名前を確認したとしても (利用可能なすべての証拠がそうではないことを示す)、これには (a) ブラウザが名前を提供するために信頼された UI を使用すること、および (b) ユーザが似たような名前に騙されないことが必要である。

4.3.2.2. 短い認証文字列

ZRTP [RFC6189] は、鍵共有プロトコルから派生した「Short Authentication String」(SAS) を使用する。この SAS は、ユーザ (例えば、音声チャネルで読み上げたり帯域外チャネルで送信する) が比較するように設計されており、双方が確認すれば MITM 攻撃を防ぐことができる。これは、SAS が 1 回使用され、その後で鍵の連続性 (前述のメカニズムとは異なる) が使用されることを意味する。

残念ながら、SAS は、危険にさらされた通話サービスの問題に対する実用的な解決策を提供していない。特定の話者の声を模倣する「音声複製」システムは活発な研究分野であり [deepfakes-ftc]、現実世界での攻撃 [deepfakes-fraud] にすでに使用されている。これらの攻撃は、特にユーザが電話をかけたいだけの環境では、今後改善される可能性が高い。したがって、SAS が現在有効であったとしても、それはそれほど長くは続かない可能性が高い。

さらに、ユーザが実際に SAS を使用するかどうかは不明である。前述のように、ブラウザ UI の制約により、通話を完了する前に SAS 交換を要求することはできないため、SAS 交換は任意である必要がある。多くの場合、ブラウザは SAS がまだチェックされていないことを示す UI インジケータを表示する。しかし、オプションのセキュリティメカニズムに直面したとき、多くのユーザが単にそれらを無視する [whitten-johnny] ことはよく知られている。

ユーザが SAS を 1 回確認したら、通話のたびに確認する必要がないように、鍵の連続性が必要である。しかしながら、これは 4.3.2.1 項に示された理由により問題がある。原則として、通話が認証されていない一連の鍵材料を使用していることを示すために、別の UI 要素をレンダリングすることはもちろん可能であるが (攻撃者がわずかに異なる名前を提示するだけで、新しいデバイスまたは以前に発信したことのない相手への通話と同じ UI を示すことができる)、実際的な問題として、混合コンテンツ警告のかなり悲惨なケースでも、ユーザは単にそのようなインジケータを無視する。

4.3.2.3. サードパーティ ID

通信 ID を提供する従来のアプローチは、エンドポイントを認証するために何らかのサードパーティ ID システム (例：PKI) を使用することであった。このようなメカニズムは、一般的なユーザが使用するには煩雑すぎることが証明されている (また、管理者にとってはほとんど面倒である)。しかし、新世代の Web ベースの ID プロバイダ (BrowserID、Federated Google Login、Facebook Connect、OAuth [RFC6749]、OpenID [OpenID]、WebFinger [RFC7033]) が開発され、Web 技術を使用して軽量な (ユーザの観点から) サードパーティ認証トランザクションを提供している。このタイプのシステムを使用して WebRTC 発信を認証し、既存の ID のユーザ概念 (例：Facebook の隣接関係) にリンクすることができる。具体的には、サードパーティの ID システムを使用して、ユーザの ID を暗号化された鍵材料にバインドし、それを使用して発信側エンドポイントを認証する。この方法で認証された通話は、通話サイトによるアクティブな MITM 攻撃に対しても自然に耐性がある。

PKI スタイルの証明書が実用的な解決策を提供する特殊なケースが 1 つあることに注意する必要がある。それは、エンドユーザから大規模サイトへの通話である。例えば、Amazon.com に電話をかける場合、Amazon はウェブトラフィックを認証するための証明書を取得するのと同様に、メディアトラフィックを認証するための証明書を簡単に取得できる。これは、通話サイトとメディアピアが同じである場合には追

加のセキュリティ価値を提供しないが、サードパーティ (例えば、広告ネットワークや小売業者) が通話を手配しても参加しない場合に役立つことがある。

4.3.2.4. メディアへのページアクセス

ファーメディアエンドポイントの ID を識別することは、メディアセキュリティを提供するための必要条件であるが十分条件ではない。WebRTC では、メディアフローは HTML5 MediaStream にレンダリングされ、通話サイトで操作できる。明らかに、サイトがメディアを変更または表示できる場合、ユーザはピアを認証できると期待されるレベルの保証を得られない。多くの場合、ユーザはサイトからの完全なセキュリティよりもサイトベースの特殊効果を重視するため、これは許容される。ただし、サイトが干渉できないことをユーザが知りたい場合もある。これを促進するためには、サイトがメディアストリームへのアクセスを検証可能に放棄できる機能を提供する必要がある。この検証は、ローカル側とリモート側の両方から可能である必要がある。すなわち、ユーザは、呼び出された人がセキュアメディアモードを使用していることを確認できなければならない (4.3.3 項を参照)。これを実現するには、ローカルメディアアクセスポリシーの表示を、前項で説明した暗号認証手順に暗号的にバインドする必要がある。

このセキュアメディアモードの使用は、サイトの裁量に委ねられていることに注意する必要がある。このようなモードが使用されている場合、ブラウザは、関連付けられたメディアが識別されたユーザからのものとして認証されたことを示す指標をユーザに提供する必要がある。これにより、エンドツーエンドのセキュリティを要求する WebRTC サービスは、ユーザが簡単に検証できる方法でそれを行うことができる。このモデルでは、3 章で説明するように、リモートパーティのブラウザが TCB に含まれている必要がある。

4.3.3. 悪意のあるピア

通常、防止しようとしない攻撃の 1 つのクラスは、悪意のあるピアである。例えば、どのような秘密保持措置を講じても、通話を録音してインターネットに公開する可能性がある。同様に、音声や動画の処理技術を使用して、外観を隠したり変更したりすることを防止できない。これらの問題に対処しようとする技術 (デジタル著作権管理 (DRM) など) は存在するが、一般的なオープンシステムとの互換性はなく、WebRTC はそれらに対処していない。

同様に、いたずら電話やその他の迷惑電話を防止する試みも実施していない。一般に、これは通話サイトの範囲内であるが、WebRTC は強力な認証をいくつかの形式で提供しているため、そのような攻撃に対する防御の一部として役立つ可能性がある。

4.4. プライバシーに関する考慮事項

4.4.1. 匿名呼び出しの相関関係

永続的なエンドポイント識別子は、有用なセキュリティ機能となる可能性があるが (4.3.2.1 項を参照)、ユーザが匿名を希望する設定ではプライバシーの脅威になる可能性もある。WebRTC は、DTLS 証明書 (接続間で再利用される場合) や RTCP CNAME ([RFC7022] のプライバシー保護モードではなく [RFC6222] に従って生成される場合) など、いくつかの永続識別子を提供する。このタイプの相関関係を防ぐために、ブラウザはこれらの識別子をリセットするメカニズムを提供する必要がある (例えば、クッキーと同じ寿命とする)。さらに、API は、匿名呼び出しを目的としたサイトが新しい識別子を強制的に作成できるようにするメカニズムを提供する必要がある。さらに、IP アドレスを通話連携の発信元にすることもできる [RFC8828]。

4.4.2. ブラウザのフィンガープリント

どのような新しい API のセットでも、ブラウザのフィンガープリントのリスクがあり、WebRTC も例外ではない。具体的には、サイトは特定のデバイスの有無をブラウザのフィンガープリントとして使用できる。一般的に、API は機能とフィンガープリントリスク増加とのバランスを取る必要がある。[Fingerprinting] を参照。

5. セキュリティに関する考慮事項

この文書全体はセキュリティに関するものである。

6. IANA に関する考慮事項

このドキュメントには IANA アクションはない。

7. 参考資料

7.1. 標準参照

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. 参考文献

[abarth-rtcweb] Barth, A., "Prompting the user is security failure", RTC-Web Workshop, September 2010, <http://rtc-web.alvestrand.com/home/papers/barth-security-prompt.pdf?attredirects=0>.

[cranor-wolf] Sunshine, J., Egelman, S., Almuhimedi, H., Atri, N., and L. Cranor, "Crying Wolf: An Empirical Study of SSL Warning Effectiveness", Proceedings of the 18th USENIX Security Symposium, August 2009, <https://www.usenix.org/legacy/event/sec09/tech/full_papers/sunshine.pdf>.

[deepfakes-fraud] Statt, N., "Thieves are now using AI deepfakes to trick companies into sending them money", September 2019, <https://www.theverge.com/2019/9/5/20851248/deepfakes-ai-fake-audio-phone-calls-thieves-trick-companies-stealing-money>.

[deepfakes-ftc] Lyons, K., "FTC says the tech behind audio deepfakes is getting better", January 2020, <https://www.theverge.com/2020/1/29/21080553/ftc-deepfakes-audio-cloning-joe-rogan-phone-scams>.

[fetch] van Kesteren, A., "Fetch", <https://fetch.spec.whatwg.org/>.

[finer-grained] Jackson, C. and A. Barth, "Beware of Finer-Grained Origins", Web 2.0 Security and Privacy (W2SP 2008), July 2008.

[Fingerprinting] Doty, N., Ed., "Mitigating Browser Fingerprinting in Web Specifications", March 2019, <https://www.w3.org/TR/fingerprinting-guidance/>.

[huang-w2sp] Huang, L-S., Chen, E.Y., Barth, A., Rescorla, E., and C. Jackson, "Talking to Yourself for Fun and Profit", Web 2.0 Security and Privacy (W2SP 2011), May 2011.

[OpenID] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC3760] Gustafson, D., Just, M., and M. Nystrom, "Securely Available Credentials (SACRED) - Credential Server Framework", RFC 3760, DOI 10.17487/RFC3760, April 2004, <https://www.rfc-editor.org/info/rfc3760>.

[RFC4251] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251, DOI 10.17487/RFC4251, January 2006, <https://www.rfc-editor.org/info/rfc4251>.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <https://www.rfc-editor.org/info/rfc4568>.

[RFC5479] Wing, D., Ed., Fries, S., Tschofenig, H., and F. Audet, "Requirements and Analysis of Media Security Management Protocols", RFC 5479, DOI 10.17487/RFC5479, April 2009, <https://www.rfc-editor.org/info/rfc5479>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/info/rfc5763>.

[RFC6189] Zimmermann, P., Johnston, A., Ed., and J. Callas, "ZRTP: Media Path Key Agreement for Unicast Secure RTP", RFC 6189, DOI 10.17487/RFC6189, April 2011, <https://www.rfc-editor.org/info/rfc6189>.

[RFC6222] Begen, A., Perkins, C., and D. Wing, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 6222, DOI 10.17487/RFC6222, April 2011, <https://www.rfc-editor.org/info/rfc6222>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/info/rfc6454>.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/info/rfc6455>.

[RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <https://www.rfc-editor.org/info/rfc7022>.

[RFC7033] Jones, P., Salgueiro, G., Jones, M., and J. Smarr, "WebFinger", RFC 7033, DOI 10.17487/RFC7033, September 2013, <https://www.rfc-editor.org/info/rfc7033>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015,

<https://www.rfc-editor.org/info/rfc7675>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <https://www.rfc-editor.org/info/rfc8825>.

[RFC8827] Rescorla, E., "WebRTC Security Architecture", RFC 8827, DOI 10.17487/RFC8827, January 2021, <https://www.rfc-editor.org/info/rfc8827>.

[RFC8828] Uberti, J. and G. Shieh, "WebRTC IP Address Handling Requirements", RFC 8828, DOI 10.17487/RFC8828, January 2021, <https://www.rfc-editor.org/info/rfc8828>.

[SWF] "SWF File Format Specification Version 19", April 2013, <https://www.adobe.com/content/dam/acom/en/devnet/pdf/swf-file-format-spec.pdf>.

[whitten-johnny] Whitten, A. and J.D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0", Proceedings of the 8th USENIX Security Symposium, August 1999, <https://www.usenix.org/legacy/publications/library/proceedings/sec99/whitten.html>.

IV. RFC8827 原文の著作権について

V. RFC8827 の和訳

RFC 8827　WebRTC セキュリティアーキテクチャ

概要

この文書では、WebRTC のセキュリティアーキテクチャを定義している。WebRTC は、ブラウザに展開できるリアルタイムアプリケーションで使用することを目的としたプロトコルスイートで、「Web 上のリアルタイム通信」である。

1. はじめに

Real-Time Communications on the Web (RTCWEB) Working Group は、一般に「WebRTC」[RFC8825] と呼ばれる、Web ブラウザ間のリアルタイム通信の標準化プロトコルを策定した。WebRTC テクノロジーの主なユースケースは、リアルタイムの音声/ビデオ通話、Web 会議、直接データ転送である。ほとんどの従来のリアルタイムシステム (例：SIP ベース [RFC3261] のソフトフォン) とは異なり、WebRTC 通信は、図 1 に示すように、JavaScript (JS) API を介して、何らかの Web サーバによって直接制御される。

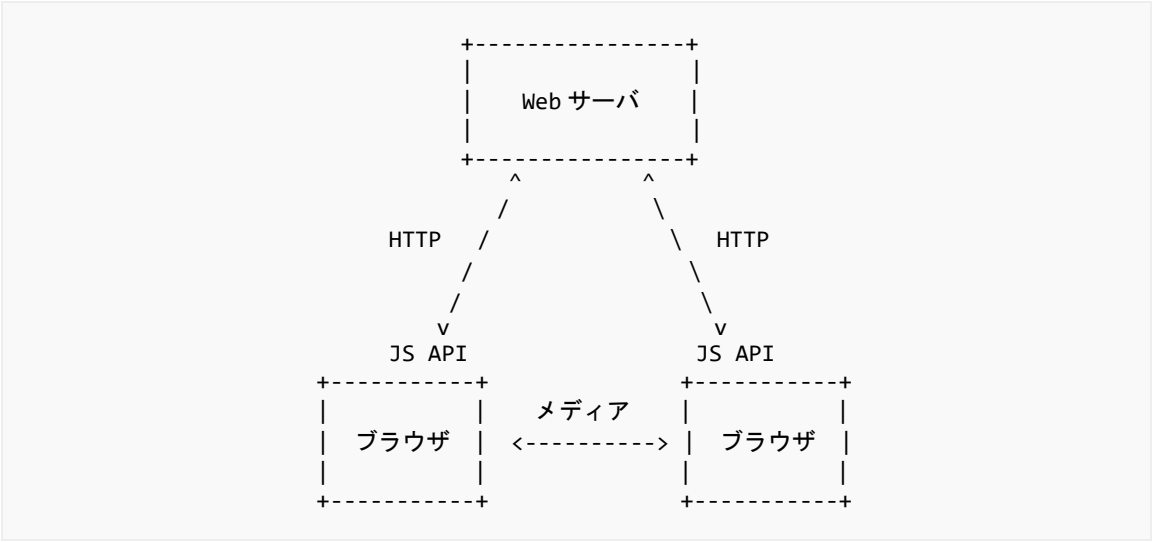


図 1：単純な WebRTC システム

図 2 に示すように、より複雑なシステムではドメイン間通話が可能になる場合がある。ドメイン間で使用されるプロトコルは WebRTC によって標準化されていないが、インストールベースと WebRTC API の形式を考えると、SIP のような SDP ベースのものか、Extensible Messaging and Presence Protocol (XMPP) [RFC6120] のようなものになる可能性が高い。

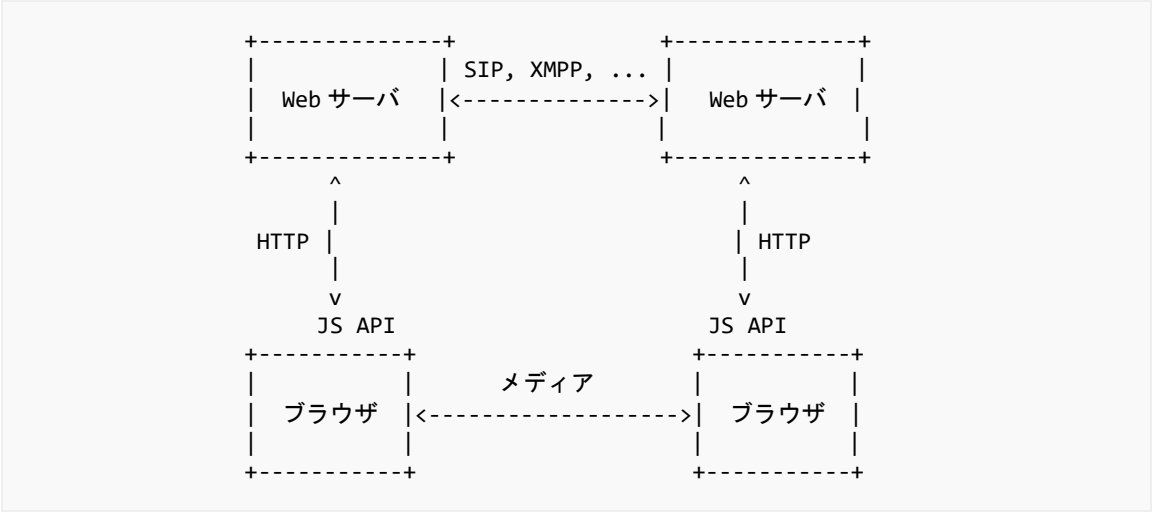


図 2：マルチドメイン WebRTC システム

このシステムでは、多くの新しいセキュリティ上の課題が提示され、[RFC8826] で分析されている。この文書では、[RFC8826] に記述されている脅威と要件に対処する WebRTC のセキュリティアーキテクチャについて説明する。

2. 用語

この文書のキーワードである「MUST」、「MUST NOT」、「REQUIRED」、「SHALL」、「SHALL NOT」、「SHOULD」、「SHOULD NOT」、「RECOMMENDED」、「NOT RECOMMENDED」、「MAY」、および「OPTIONAL」は、ここに示すように、すべて大文字で表示される場合にのみ、BCP 14 [RFC2119] [RFC8174] に記載されているように解釈される。

3. 信頼モデル

このアーキテクチャの基本的な前提は、ネットワークリソースがブラウザをルートとする信頼の階層に存在することであり、これはユーザの Trusted Computing Base (TCB) として機能する。ユーザが適用を希望するセキュリティプロパティは、最終的にブラウザによって (または、ブラウザが検証する何らかのプロパティによって推移的に) 保証される必要がある。逆に、もしブラウザが侵害された場合、セキュリティは保証されない。ユーザがブラウザを信頼できない場合 (例:インターネットキオスク) もあることに注意すること。このような場合、提供されるセキュリティのレベルは、ブラウザをどれだけ信頼するかによって制限される。

最適なのは、ブラウザ以外のエンティティへの信頼に依存しないことである。しかし、残念ながらこれは機能的なシステムを持ちたい場合には不可能である。その他のネットワーク要素は、ブラウザによって認証できるため機密性の高いリソースへのアクセス許可を付与できるものと、認証できないため信頼できないものの 2 つに分類される。

3.1. 認証済みエンティティ

システムには、認証済みエンティティの 2 つの主要なクラスがある。

通話サービス: 発信元を確認できるウェブサイト(最適には HTTPS 経由だが、場合によっては、ファイアウォールの内側などのトポロジ的に制限されたネットワーク上にあり、ファイアウォールの動作から認証を推測できるウェブサイト)。

その他のユーザ: 発信元を暗号的に検証できない WebRTC ピア(最適には DTLS-SRTP 経由)。

認証されるだけでは、これらのエンティティは信頼されないことに注意すること。例えば、<https://www.example.org/> が Dr. Evil によって所有されていることを確認できるからといって、Dr. Evil がカメラとマイクにアクセスすることを信頼できるわけではない。ただし、Dr. Evil を信頼するかどうかを判断する機会はユーザに与えられる。結局のところ、もし彼らが Dr. Evil に連絡を取りたいのであれば (おそらく身代金の支払いを手配するため)、通話のために一時的にカメラとマイクにアクセスできるようにすることに問題はないが、Dr. Evil が通話中以外にカメラとマイクにアクセスできるようにしたくない。ここでのポイントは、他の要素を信頼するかどうか、またどの程度信頼するかを決定する前に、まずそれらを特定しなければならないということである。さらに、適用するポリシーを知る前に、通信相手を特定する必要がある場合もある。

3.2. 認証されていないエンティティ

上記のエンティティ以外では、一般的に他のネットワーク要素を特定することはできず、信用できない。これは、彼らとの交流ができないということではなく、彼らが悪意を持って行動することを想定し、安全なシステムを設計しなければならないということである。

4. 概要

この章では、一般的な WebRTC セッションについて説明し、さまざまなセキュリティ要素がどのように相互作用し、どのような保証がユーザに提供されるかを示す。この章の例は、最大量のユーザ認証とメディアプライバシーを、通話サービスの最小限の信頼レベルで提供する「ベストケース」のシナリオである。セキュリティレベルの低い単純なバージョンも可能であり、該当する場合は本文に記載される。また、セキュリティ (またはパフォーマンス) とプライバシーの間の緊張関係を認識することも重要である。ここで示す例は、プライバシーよりも安全な通話を重視する設定を対象としているが、後述するように、さまざまなトレードオフを行う必要がある設定もある。このアーキテクチャは、これらの設定と互換性がある。

この例では、以下の図 (図 3) に示すトポロジを想定する。このトポロジは図 1 に示すトポロジから派生しているが、アリスとボブの ID をシグナリングのプロセスから分離している。具体的には、アリスとボブは、他の関係者に自分の ID を証明するために使用できるプロトコル (OpenID Connect など) をサポートする一部の ID プロバイダ (IdP) と関係を持っている。例えば、アリスがソーシャルネットワークのアカウントを持っていて、それを使って他の Web サイトのアカウントを明示的に持っていなくても、それらのサイトへの認証を行うことができる。これは Web 上ではかなり一般的なパターンである。7.1 節では、IdP の概要と関連用語について説明する。アリスとボブは、異なる IdP とも関係を持つ場合がある。注意：ここで説明する IdP メカニズムは、広く採用されていない。IdP ベースの認証のステータスの詳細については、7 章を参照すること。

この ID の提供とシグナリングの分離は、アリスとボブが同じソーシャルネットワーク上のユーザであり、そのドメインに基づく ID を持つ 「クローズドワールド」 のケースでは特に重要ではない (図 3)。ただし、フェデレーション (あるドメインから別のドメインへの通話、図 4 を参照) や、特定のソーシャルネットワークを介して関係を持っている 2 人のユーザが、ポーカーサイトなどの信頼されていない別のサイトで互いに通話したい場合など、そうではない重要な設定がある。

サーバ自体も、外部 ID サービスである SSL/TLS 証明書インフラストラクチャ (図には示されていない) によって認証されることに注意する。Web で慣習的に行われているように、すべての ID は最終的にそのシステムに根ざしている。例えば、IdP が ID アサーションを作成すると、そのアサーションを使用するリライングパーティは、HTTPS 経由で IdP に接続できるため、検証できる。

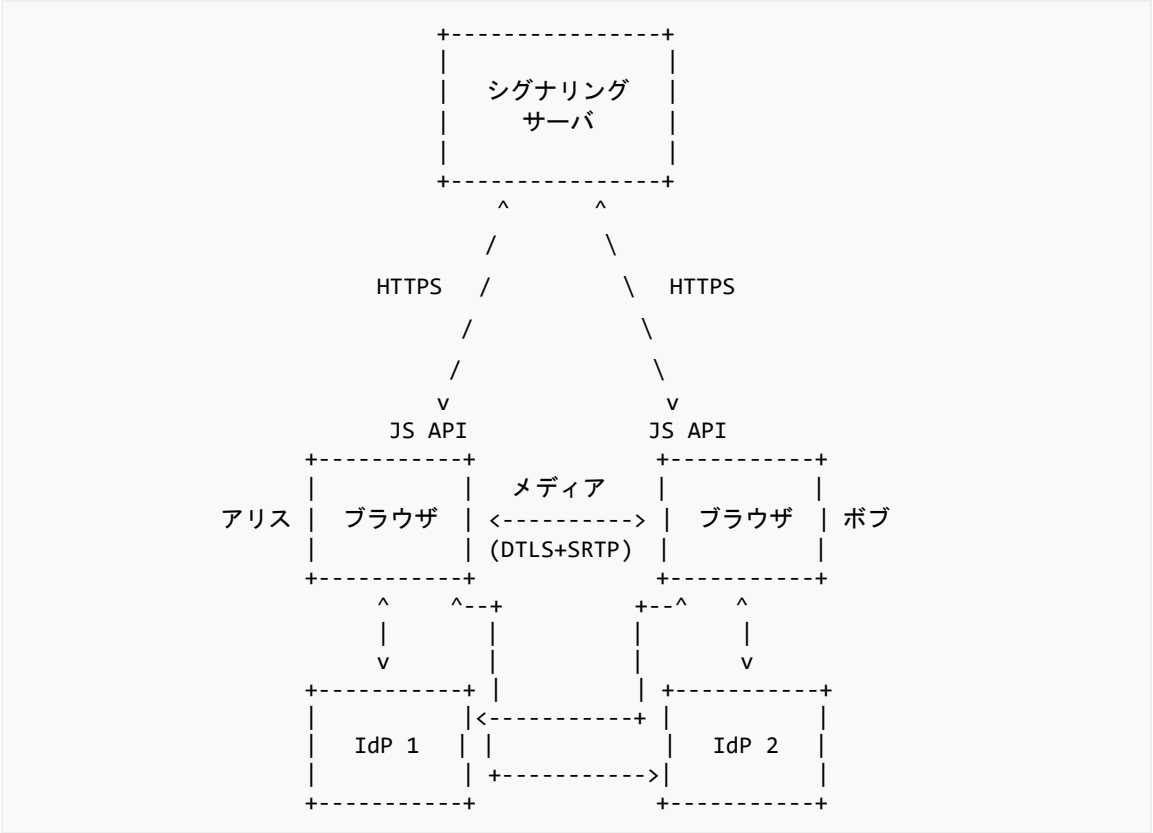


図 3：IdP ベースの ID を使用した通話

図 4 は、基本的に同じ呼び出しシナリオを示しているが、図 2 と同様に、2 つの異なるドメイン (すなわち、フェデレーションのケース) 間の通話を使用している。前述のように、ドメインは何らかの不特定のプロトコルで通信し、個別のシグナリングと ID を提供することで、ドメイン間プロトコルの詳細に関係なく通話を認証できるようになる。

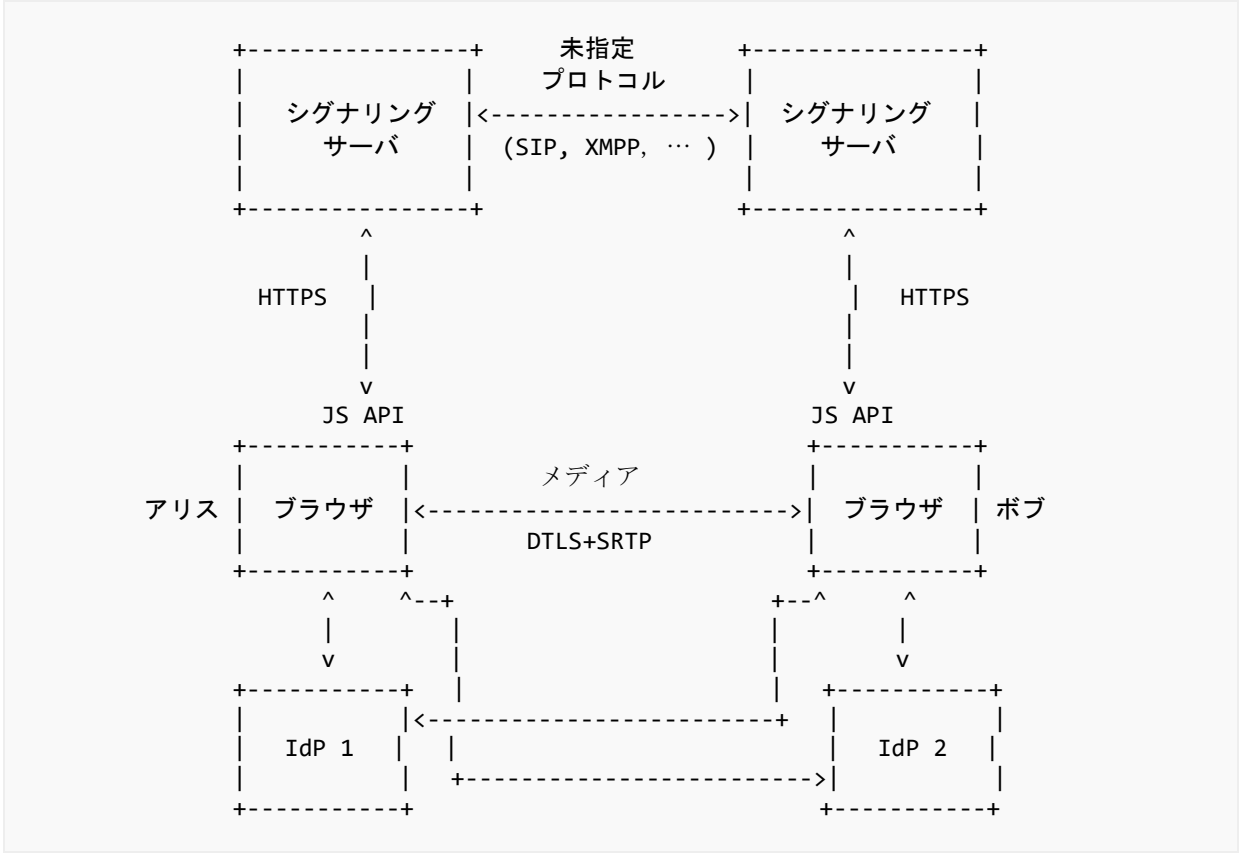


図 4：IdP ベースの ID を使用したフェデレーション通話

4.1. 初期シグナリング

簡単にするため、図 3 のトポロジを想定する。アリスとボブはどちらも共通通話サービスのユーザである。どちらも、通話サービスが通話を実行することを承認している (デバイスのアクセス許可については、後で説明する)。どちらも HTTPS 経由で通話サービスに接続されているため、ある程度の信頼性を持って発信元を知ることができる。また、一部の IdP のアカウントも持っている。この種の ID サービスは Web 環境 (Federated Google Login、Facebook Connect、OAuth、OpenID、WebFinger などのテクノロジーを使用) で一般的になりつつあり、ユーザの通常のアカウントの副作用サービスとして何らかのサービスとともに提供されることが多い。この例では、アリスとボブが別の ID サービスを使用しているが、ID サービスが通話サービスと同じエンティティであったり、ID サービスがまったく存在しない場合もある。

アリスは通話サービスにログインし、ボブに呼び出す。彼女は、通話サービスから彼がオンラインであることを確認でき、通話サービスは、「通話」と書かれたボブの名前の横にあるボタンの形式で JS UI を提示する。アリスがボタンをクリックすると、PeerConnection オブジェクトをインスタンス化する JS コールバックが開始される。これはセキュリティチェックを必要としない。どのような発信元からの JS でもここまでは許可される。

PeerConnection が作成されたら、通話サービス JS はいくつかのメディアを設定する必要がある。これは音声/ビデオ通話であるため、2 つの MediaStreamTrack を持つ MediaStream が作成される。1 つは音声入力に接続され、もう 1 つはビデオ入力に接続される。この時点で、最初のセキュリティチェックが必要になる。信頼できない発信元はカメラとマイクにアクセスできないため、ブラウザはアリスに許可を求める。

現在の W3C API では、いくつかのストリームが追加されると、アリスのブラウザ+JS によって次の内容を含むシグナルメッセージ [RFC8829] が生成される。

- メディアチャネル情報
- Interactive Connectivity Establishment (ICE) [RFC8445] の候補
- 通信をキーペア [RFC5763] に紐づけた ”fingerprint” 属性。このキーは、この呼び出しのために一時的に生成されるか、このドメイン固有のものである場合があり、アリスにはこのようなキーが多数存在する場合があることに注意すること。

シグナリングメッセージを送信する前に、PeerConnection コードは ID サービスに接続し、アリスの ID をフィンガープリントに紐づけるするアサーションを取得する。正確な詳細は ID サービスに依存するが (ただし、7 章で説明するように、PeerConnection は ID サービスに依存しない場合もある)、現時点では OAuth トークンと考えるのが最も簡単である。アサーションは、フィンガープリント以外の他の情報を ID にバインドする場合があるが、少なくともフィンガープリントをバインドする必要がある。

このメッセージは、例えば、fetch() [fetch] または WebSockets [RFC6455] によって、TLS [RFC8446] を介してシグナリングサーバに送信される。シグナリングサーバはアリスのブラウザからのメッセージを処理し、これがボブへの呼び出しであると判断して、ボブのブラウザにシグナリングメッセージを送信する (繰り返すが、形式は現在定義されていない)。ボブのブラウザ上の JS がそれを処理し、ボブに着信とアリスの身元を警告する。この場合、アリスが ID アサーションを提供しているため、ボブのブラウザはアリスの IdP (これも一般的な方法で行われるため、ブラウザは IdP に関する特定の知識を持たない) に接続してアサーションを検証する。7.1 節で説明するように、ブラウザが特定の信頼関係を持つ IdP を持つこともできる。これにより、ブラウザは、クロームに信頼できる要素を表示して、アリスからの呼び出しであることを示す。アリスがボブのアドレス帳にある場合、このインタフェースには彼女の本名や写真なども含まれることがある。また、通話サイトは、ボブが呼び出しに応答できるようにするためのいくつかのユーザインタフェース要素 (例：ボタン) も提供するが、これは信頼された UI の一部ではない可能性が高い。

ボブが同意すると、アリス側からのメッセージで PeerConnection がインスタンス化される。次に、アリスのブラウザと同様のプロセスが発生する。ボブのブラウザがデバイスの許可を求め、メディアストリームが作成され、メディア情報、ICE 候補、およびフィンガープリントを含むシグナリングメッセージが、シグナリングサービスを介してアリスに返される。ボブが IdP と関係を持っている場合、メッセージには ID アサーションも付属する。

この時点で、アリスとボブはそれぞれ、相手が自分との安全な通話を望んでいることを知っている。純粋にシグナリングサーバによって提供されるインタフェースに基づいて、呼び出しがアリスからボブへのものであるとシグナリングサーバが主張していることがわかる。このレベルのセキュリティは、メッセージにフィンガープリントが含まれ、そのメッセージがシグナリングサーバから安全に受信されるだけで提供される。終端がメッセージとともに ID アサーションを送信したため、これが IdP からも検証可能であることがわかる。図 4 に示すように、呼び出しがフェデレートされている場合、アリスは自身のシグナリングサーバまたはボブのいずれによっても仲介されない方法でボブの ID を確認できる。むしろ、ボブの IdP で直接検証する。

もちろん、アリスまたはボブのいずれかが IdP と関係を持たない場合、低いレベルの保証しか得られないため、この呼び出しは完全に機能する。つまり、発信者/着信者の ID に関して発信サイトが主張する情報を持っているだけとなる。さらに、アリスは匿名通話サイトを通じて匿名の呼び出しを行うことを望むかもしれない。その場合、アリスはもちろん、ID アサーションを提供しないだけで、通話サイトはボブから彼女の ID を隠蔽する。

4.2. メディア同意検証

[RFC8826] の 4.2 節で説明されているように、メディア同意検証は ICE 経由で提供される。したがって、アリスとボブは相互に ICE チェックを実行する。これらのチェックが完了すると、ICE 以外のデータを送信できるようになる。

この時点で、アリスは、 (a) ボブ (自身の IdP 経由で検証されていると仮定)、またはシグナリングサービスが主張しているボブが彼女とトラフィックを交換する意思があると主張している他の誰かがいること、および (b) ボブが ICE 経由で検証した IP アドレスにいるか、またはトラフィックを迂回してその IP アドレスへのパス上にいる攻撃者がいることを認識している。アリスとボブの間のパス上にいても、シグナリングサービスに接続していない攻撃者は、ICE 資格情報を持っていないため、なりすましすることはできない。ボブはアリスに関して同じセキュリティ保証を持つことになる。

4.3. DTLS ハンドシェイク

必要な ICE チェックが完了すると、アリスとボブはセキュリティで保護されたチャネルを設定できる。これは DTLS [RFC6347] および DTLS-SRTP [RFC5763] キーイング (メディアチャネルのための SRTP [RFC3711] とデータチャネルのための Stream Control Transmission Protocol (SCTP) over DTLS [RFC8261]) を介して実行される。具体的には、アリスとボブは ICE によって確立されたすべてのコンポーネントに対して DTLS ハンドシェイクを実行する。チャネルの総数は、多重化の量によって異なる。最も可能性の高いケースでは、RTP/RTCP の多重化と、同じチャネル上の複数のメディアストリームの多重化の両方を使用している。この場合、DTLS ハンドシェイクは 1 つのみとなる。DTLS ハンドシェイクが完了すると、キーはエクスポートされ [RFC5705]、メディアチャネルの SRTP のキー設定に使用される。

この時点で、アリスとボブは、セキュアなデータと第三者の攻撃者が知らないキーを持つメディアチャネルのセットを共有していることを知っている。アリスとボブがそれぞれの IdP を介して認証された場合、シグナリングサービスがトラフィックに対して man-in-the-middle 攻撃を仕掛けていないこともわかる。IdP を使用しない場合でも、man-in-the-middle 攻撃を実行できないシグナリングサービスを最低限信頼している限り、シグナリングサービスに対しても通信が安全であることがわかる。 (つまり、シグナリングサービスは通信に対してパッシブ攻撃を実行でできない)

4.4. コミュニケーションと同意の鮮度

セキュリティの観点からは、ここから先のすべては少し逆境的となる。アリスとボブは DTLS でネゴシエートされたキーによって保護されたデータを交換する。前章で説明したセキュリティ保証のため、彼らは通信が暗号化および認証されることがわかっている。

確立する必要がある残りの 1 つのセキュリティプロパティは、「同意の鮮度」である。つまり、アリスが突然オフラインになったエンティティに大量のトラフィックを送信し続けないように、ボブがまだ彼女の通信を受信する準備ができていることをアリスが確認できるようにする。ICE は、メディアがフローしていない場合に限り、定期的な Session Traversal Utilities for NAT (STUN) キープアライブを指定する。ここでは同意の問題がより困難であるため、WebRTC の実装では、[RFC7675] で指定されている同意鮮度メカニズムを使用してキープアライブを定期的に送信する必要がある。キープアライブが失敗し、新しい ICE チャネルを確立できない場合、セッションは終了する。

5. SDP "identity" 属性

SDP "identity" 属性は、エンドポイントが ID アサーションを相手に伝えるために使用するセッションレベルの属性である。Identity-assertion 値は、[RFC4648] の 4 章で説明されているとおり、base64 としてエンコードされる。

この章の手順は、エンドポイントの ID アサーションがエンドポイントのフィンガープリントにバインドされているという前提に基づいている。これは、エンドポイントにアサーションをバインドする代替手段の定義を妨げるものではないが、そのような手段はこの仕様の範囲外である。

オファーまたはアンサー内の複数の "identity" 属性のセマンティクスは定義されていない。実装は、オファーまたはアンサーに単一の "identity" 属性のみを含めるべき[SHOULD]であり、リライングパーティは、最初の "identity" 属性以外はすべて無視することを選択してもよい[MAY]。

Name: identity

Value: identity-assertion

Usage Level: session

Charset Dependent: no

Default Value: N/A

構文:

```
identity-assertion          = identity-assertion-value
                              * (SP identity-extension)
identity-assertion-value    = base64
identity-extension          = extension-name [ "=" extension-value ]
extension-name              = token
extension-value             = 1*(%x01-09 / %x0b-0c / %x0e-3a / %x3c-ff)
                              ; byte-string from [RFC4566]

<ALPHA and DIGIT as defined in [RFC4566]>
<base64 as defined in [RFC4566]>
```

例:

```
a=identity:\
  eyJpZHAiOnsiZG9tYWluIjoiZXhhbXBsZS5vcmciLCJwcm90b2NvbCI6ImJvZ3Vz\
  In0sImFzc2VydGlvbiI6IntcImlkZW50aXR5XCI6XCJib2JAZXhhbXBsZS5vcmdc\
  IixcImNvbnRlbnRzXCI6XCJhYmNkZWZnaGlqa2xtbm9wcXJzdHV2d3l6XCIsXCJz\
  aWduYXR1cmVcIjpcIjAxMDIwMzA0MDUwNlwifSJ9
```

この例の長い行は、この文書の列幅の制約を満たすように折りたたまれていることに注意すること。行末のバックスラッシュ（「\」）、それに続く改行、空白は無視する。

この仕様では、属性の拡張は定義されていない。

identity-assertion 値は、JSON エンコードされた文字列である [RFC8259]。JSON オブジェクトには、"assertion" と "idp" の 2 つのキーが含まれている。"assertion" キー値には、IdP によって消費される不透明な文字列が含まれている。"idp" キー値には、IdP を識別する 1 つまたは 2 つの追加の値を持つ辞書が含まれている。詳細は 7.6 節を参照。

5.1. オファー/アンサーの考慮事項

この節では、SDP "identity" 属性の SDP オファー/アンサー [RFC3264] に関する考慮事項を定義する。

このセクション内の「初期オファー」は、SDP "identity" 属性を含む SDP セッション内の最初のオファーを指す。

5.1.1. 初期 SDP オファーの生成

申込者がオファーを送信すると、相手に ID アサーションを提供するために、オファーに "identity" 属性が含まれる。さらに、オファーには 1 つ以上の SDP "fingerprint" 属性が含まれる。"identity" 属性は、セッション記述内のすべての "fingerprint" 属性へバインドしなければならない[MUST]。

5.1.2. SDP アンサーの生成

回答者が "identity" 属性を含めることを選択した場合、5.1.1 項と同じ手順に従う。回答者は、申込者側が行ったかどうかに関係なく、"identity" 属性を個別に含めるか省略するかを選択できる。

5.1.3. SDP オファーまたはアンサーの処理

エンドポイントが "identity" 属性を含むオファーまたはアンサーを受信すると、回答者は属性情報を使用して IdP に連絡し、ピアの ID を確認できる。7.1 節で説明されているように、ID にサードパーティの IdP が必要な場合は、その IdP を特別に設定する必要がある。もし本人認証に失敗した場合、回答者はオファーまたはアンサーを不正な形式として破棄しなければならない[MUST]。

5.1.4. セッションの変更

セッションを変更するときに、フィンガープリントのセットが変更されていない場合、送信者は同じ "identity" 属性を送信できる[MAY]。この場合、確立された ID は、既存の DTLS 接続だけでなく、これらのフィンガープリントのいずれかを使用して確立された新しい接続にも適用されなければならない[MUST]。[RFC8829] の 5.2.1 項では、各メディアセクションで同じフィンガープリントのセットを使用する必要があることに注意する。新しい "identity" 属性を受信した場合、受信者はその ID を既存のすべての接続に適用しなければならない[MUST]。

フィンガープリントのセットが変更された場合、送信者は新しい "identity" 属性を送信するか、まったく送信してはならない[MUST]。フィンガープリントの変更によって新しい DTLS 接続も確立されるため、受信者は以前に確立された ID をすべて破棄しなければならない[MUST]。

6. 詳細な技術的説明

6.1. 源泉 (origin) と Web セキュリティの問題

WebRTC の許可の基本単位は、源泉 (origin) [RFC6454] である。源泉 (origin) のセキュリティは、その源泉 (origin) からのコンテンツを認証できるかどうかに依存するため、HTTPS [RFC2818] を介してデータを転送する場合にのみ、源泉 (origin) を安全に確立できる。したがって、クライアントは HTTP と HTTPS の源泉 (origin) を異なる権限ドメインとして扱う必要がある[MUST]。注意：これは、源泉 (origin) セキュリティモデルから直接従うものであり、ここでは単に明確にするために記載している。

現在、多くの Web ブラウザでは、HTTPS ページでアクティブな混在 (HTTP と HTTPS) コンテンツをデフォルトで禁止している。つまり、JavaScript が HTTP 源泉 (HTTP origin) から HTTPS ページにロードされると、エラーが表示され、ユーザがエラーをオーバーライドしない限り HTTP コンテンツは実行されない。このようなポリシーを適用するブラウザは、混在コンテンツページから WebRTC 機能へのアクセスも許可しない (混在コンテンツは表示されないため)。アクティブな混合コンテンツを許可するブラウザは、混合コンテンツ設定で WebRTC 機能を無効にする必要がある。

なお、混合コンテンツではなかったページは、通話中に混合コンテンツになることがある。ここでの大きなリスクは、新しく到着した安全でない JS によって、攻撃者が制御する場所にメディアがリダイレクトさ

れる可能性があることである。実装は、その時点で呼び出しを終了するか、警告を表示するかを選択する必要がある[MUST]。

また、セキュリティアーキテクチャは、源泉 (origin) 間を移動することができないキーマテリアルに依存する。ただし、ID アサートはページを気にする誰にでも渡すことができると想定されている。

6.2. デバイス権限モデル

実装は、カメラおよび/またはマイクへのアクセスを提供する前に、明示的なユーザの同意を得る必要がある[MUST]。実装は、HTTPS 源泉 (HTTPS origin) に対して少なくとも次の 2 つのアクセス許可モデルをサポートする必要がある[MUST]。

- 1 回限りのカメラ/マイクアクセスを要求。
- 常にアクセスできることを要求。

HTTP 源泉 (HTTP origin) はネットワーク攻撃者に対して安全に確立できないため、実装は HTTP 源泉 (HTTP origin) に対するすべての権限付与を拒否する必要がある[MUST]。

さらに、認められたところからのメディアが単一の通信ピアに送信されることを約束するアクセスからの要求をサポートする必要がある[SHOULD] (明らかに、他のピアに対する他の要求がある可能性がある) (例：「Call customerservice@example.org」)。この要求の意味は、カメラとマイクからのメディアストリームは、指定された ID に関連付けられていることが暗号的に検証 (DTLS-SRTP ハンドシェイクの IdP メカニズムまたは X.509 証明書を介して) された接続を介してのみルーティングされるということである。ブラウザに X.509 証明書がある可能性は低いが、サーバにはある可能性があることに注意する必要がある。そのような要求にサービスを提供するブラウザは、許可を求めるときに、その ID をユーザに明確に示す必要がある[SHOULD]。このタイプのアクセス許可の背景にある考え方は、ユーザがコミュニケーションを取ろうとしているピアのかなり狭いリストを持っている可能性があるということである。例えば、「Facebook 上の誰か」ではなく「私の母」である。許可される権限が狭い場合、ブラウザはその強制を行うことができる。

API の要件: API は、要求元の JS がメディアを表示または変更する機能を放棄するためのメカニズムを提供しなければならない[MUST] (例：MediaStream.record() 経由)。通信するピアの安全な認証を組み合わせることで、ユーザは通話サイトが変換しようとしたり、変更したりしていないことを確認できる。

UI の要件: UI は、ユーザのカメラとマイクがいつ使用されているかを明確に示す必要がある[MUST]。この表示は、JS によって抑制できてはならず[MUST NOT]、デバイスアクセスを終了する方法を明確に示し、JS がそれを防ぐことができずにカメラ/マイク入力をすぐに停止する UI 手段を提供しなければならない[MUST]。

UI の要件: カメラ/マイク使用の UI 表示がブラウザに表示され、ブラウザウィンドウを最小化すると表示が非表示になるか、オーバーラップするウィンドウを作成する JS によって表示が非表示になる場合、ブラウザは表示が非表示になったときにカメラとマイクの入力を停止する必要がある[SHOULD]。 (注意：これは、Windows ベースではないが、電話などの優れた通知サポートがあるシステムでは必要ない場合がある。)

- ブラウザは、JS のアクセス許可要求への応答として、恒久的な画面またはアプリケーション共有のアクセス許可をインストールすることを許可してはならない[MUST NOT]。代わりに、サイトにアクセス許可を付与するには、アクセス許可の設定やアプリケーションのインストール体験など、その他のユーザアクションが必要である。

- ブラウザは、カメラとマイクのアクセス許可の要求と同時に画面/アプリケーション共有アクセス許可の要求が行われた場合でも、画面/アプリケーション共有アクセス許可の個別のダイアログ要求を提供する必要がある[MUST]。
- ブラウザは、現在何らかの明確な方法で共有されているウィンドウを示さなければならない[MUST]。表示されないウィンドウは、アプリケーションが共有されていても共有してはならない[MUST NOT]。画面が共有されている場合は、それを示す必要がある[MUST]。

ブラウザは、ユーザの直接の承認なしにデータチャネルの形成を許可してもよい[MAY]。サイトは常にサーバを介してデータをトンネリングできるため、データチャネルのさらなる制限は追加のセキュリティを提供しない。 (関連する問題については 6.3 節を参照。)

何らかの形式の直接ユーザ認証をサポートする実装では、ユーザが特定の通信ピアに対してのみ呼び出しを許可できるポリシーも提供する必要がある[SHOULD]。具体的には、実装は以下のインタフェース/コントロールを提供すべきである[SHOULD]。

- この検証されたユーザへの将来の呼び出しを許可する。
- システムアドレス帳 (これはもちろんアドレス帳の統合でしか使えない) に登録されている認証済みユーザへの将来の呼び出しを許可する。

実装では、ID が直接検証可能なユーザへの呼び出しが進行中の場合に、異なるユーザインタフェースの表示も提供する必要がある[MUST]。これについては 6.5 節に詳しい説明がある。

6.3. 通信の承認

WebRTC のブラウザクライアントの実装は ICE を実装しなければならない[MUST]。パブリック IP アドレスでのみ動作するサーバゲートウェイの実装では、完全な ICE または ICE-Lite [RFC8445] のいずれかを実装する必要がある[MUST]。

ブラウザの実装では、ICE 以外のパケットを特定の宛先に送信する前に、ICE を介した到達可能性を検証する必要がある[MUST]。実装は、トランザクション (つまり、ICE スタックがそのトランザクションの新しい応答を受け入れる期間) の有効期間中に ICE トランザクション ID を JavaScript に提供してはならない[MUST NOT]。JS はもちろんそれを知っているが、ローカルの ufrag (ユーザ名) とパスワードを制御することを許可されてはならない[MUST NOT]。

継続的な同意が必要だが、ICE [RFC8445] (Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal)、セクション 11 キープアライブは STUN Binding Indication を使用するが、これは一方向であるため十分ではない。現在の WG の同意は、継続的な同意の鮮度のために ICE Binding Requests を使用することである。ICE は、実装がそのような要求に応答することを既に要求しているため、このアプローチは最大限に互換性がある。使用する ICE タイマーについては別の文章で説明する。[RFC7675] を参照すること。

6.4. IP ロケーションプライバシー

デフォルトの ICE 動作の副作用は、ピアが自分の IP アドレスを学習し、大量の位置情報が漏洩することである。 これは、状況によってはプライバシーに悪影響を及ぼす。 このセクションの API 要件は、この問題を軽減することを目的としている。これらの要件は、悪意のあるサイトからユーザの IP アドレスを保護するためのものではないことに注意すること。 一般に、サイトは HTTP トランザクションから少なくともユーザのサーバ再帰アドレスを学習する。むしろ、これらの要件は、サイトがユーザと協力して、通話の相

手からユーザの IP アドレスを隠蔽できるようにすることを目的としている。ユーザの IP アドレスをサーバから隠すには、クライアントである種の明示的なプライバシー保護メカニズムが必要である。 (例：Tor Browser <https://www.torproject.org/projects/torbrowser.html.en>) が必要であるが、この仕様の範囲外である。

API の要件: API は、ユーザが呼び出しに応答することを決定するまで、JS が ICE ネゴシエーションを抑制できるようにするメカニズムを提供する必要がある[MUST] (ただし、候補者の収集を可能にするため)。 (注意：呼び出しがいつ応答されたかの判断は、JS の問題である。) これにより、ユーザは、呼び出しに応答しないことを選択した場合にピアが自分の IP アドレスを学習したり、ユーザがオンラインかどうかを学習したりするのを防ぐことができる。

API の要件: API は、通話アプリケーション JS が TURN 候補のみを使用することを示すメカニズムを提供しなければならない[MUST]。これにより、ピアは自分の IP アドレスをまったく学習できなくなる。このメカニズムは、関連するアドレスフィールドの抑制も許可する必要がある[MUST]。これは、ローカルアドレスが漏れるためである。

API の要件: API は、通話アプリケーションが既存の呼び出しを再構成して非 TURN 候補を追加するためのメカニズムを提供する必要がある[MUST]。この要件と以前の要件を合わせると、着信呼び出し通知時に ICE ネゴシエーションをすぐに開始できるため、ダイヤル後の遅延を減らすことができるが、応答を決定するまでユーザの IP アドレスを公開しないようにすることもできる。また、通話中は IP アドレスを完全に隠すこともできる。最後に、ユーザが IP アドレスを非表示にする必要がなくなったと判断した場合に、アクティブな呼び出し中に非 TURN 候補を許可するように再設定することで、ユーザがパフォーマンスを最適化するメカニズムを使用できる。

企業によっては、内部の IP アドレスを外部から隠すように設計されたプロキシや NAT を運用している場合がある。WebRTC は、この機能を許可する明示的なメカニズムを提供しない。このような企業では、HTTP/HTTPS をプロキシして SDP や JS を変更する必要があるか、サイトの設定に関係なく「TURN-only」ポリシーを設定するためのブラウザサポートが必要である。

注意：これらの要件は、サイトがユーザの IP アドレスをピアから隠すことを目的としている。ユーザの IP アドレスを通話サイトから隠す方法については、[RFC8828] を参照すること。

6.5. 通信セキュリティ

実装は SRTP [RFC3711] をサポートしなければならない[MUST]。実装では、SRTP キーイングのために DTLS [RFC6347] と DTLS-SRTP [RFC5763] [RFC5764] をサポートする必要がある[MUST]。実装は SCTP over DTLS [RFC8261] をサポートしなければならない[MUST]。

すべてのメディアチャネルは、SRTP および Secure Real-time Transport Control Protocol (SRTCP) を介して保護する必要がある[MUST]。メディアトラフィックはプレーン (暗号化されていない) RTP または RTCP で送信してはならない[MUST NOT]。つまり、実装は NULL 暗号化モードで暗号スイートをネゴシエートしてはならない[MUST NOT]。すべてのメディアチャネルに対して DTLS-SRTP を提供する必要がある[MUST]。WebRTC 実装は、SDP セキュリティの説明 [RFC4568] を提供するか、提供されている場合はそれを選択してはならない[MUST NOT]。SRTP マスターキー識別子 (MKI) を使用してはならない[MUST NOT]。

すべてのデータチャネルは DTLS 経由で保護する必要がある[MUST]。

すべての実装で TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA 256 暗号スイートおよび P-256 カーブ [FIPS186] を伴う DTLS 1.2 をサポートする必要がある。TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA、およびこの記事の執筆時点では、一部の実装は DTLS 1.2 をサポートしていない。DTLS 1.2 のみをサポー

トするエンドポイントでは、相互運用性の問題が発生する可能性がある。DTLS-SRTP 保護プロファイル SRTP に対して SRTP_AES 128_CM_HMAC_SHA 1_80 がサポートされている必要がある[MUST]。実装は、Forward Secrecy (FS) をサポートする暗号スイートを非 FS 暗号スイートよりも支持しなければならず[MUST]、非 AEAD 暗号スイートよりも Authenticated Encryption with Associated Data (AEAD) を支持すべきである[SHOULD]。注意：IETF は DTLS 1.3 [TLS-DTLS] の標準化を進めている。

実装は DTLS 再ネゴシエーションを実装してはならず[MUST NOT]、提示された場合は"no_renegotiation"アラートで拒否しなければならない[MUST]。

エンドポイントは TLS False Start [RFC7918] を実装してはならない[MUST NOT]。

API の要件: API は、デフォルトで新しい呼び出しごとに新しい認証キーペアを生成する必要がある[MUST]。これはリンク不可能性を考慮したものである。

API の要件: API は、呼び出しのためにキーペアを再利用する手段を提供する必要がある[MUST]。これは、キー連続性ベースの認証を有効にするために使用でき、キー生成コストを償却するために使用できる。

API の要件: ユーザが特に外部キーペアを設定しない限り、発信元ごとに異なるキーペアを使用する必要がある[MUST]。 (これにより、スーパークッキーの作成が回避される。)

API の要件: DTLS-SRTP を使用する場合、API は JS がネゴシエートされたキーイングマテリアルを取得することを許可してはならない[MUST NOT]。この要件により、メディアのエンドツーエンドのセキュリティが維持される。

UI の要件: ユーザ指向のクライアントは、ユーザがメディアの「セキュリティ特性」を決定できるようにする「インスペクター」インタフェースを提供しなければならない[MUST]。

以下のプロパティは、ブラウザクロームに「up-front (正直) 」で表示される必要がある[SHOULD]。つまり、ユーザがそれらを要求する必要はない。

- クライアントは、ユーザが現在表示されているオーディオおよびビデオストリームの「セキュリティ特性」を判断できるユーザインタフェースを提供しなければならない[MUST]。
- クライアントは、ユーザがマイクオーディオとカメラビデオの送信のための「セキュリティ特性」を決定できるユーザインタフェースを提供しなければならない[MUST]。
- サードパーティの検証可能な X.509 証明書または Web IdP メカニズム (7 章を参照) を介して、遠端エンドポイントが直接検証された場合、「セキュリティ特性」には検証された情報が含まれている必要がある[MUST]。X.509 ID と Web IdP ID は同様のセマンティクスを持つため、同様の方法で表示する必要がある。

次のプロパティは、ユーザからの何らかの「ドリルダウン (詳細な分析) 」を必要とする可能性が高くなる。

- 「セキュリティ特性」は、使用中の暗号アルゴリズム (例：「AES-CBC」) を示さなければならない[MUST]。
- 「セキュリティ特性」は、FS が提供されているかどうかを示さなければならない[MUST]。
- 「セキュリティ特性」には、証明書フィンガープリントや Short Authentication String (SAS) など、ピアの帯域外検証を可能にする何らかのメカニズムが含まれている必要がある[MUST]。これらはピアによって比較され、相互に認証される。

7. Web ベースのピア認証

注意:この章で説明するメカニズムは、RTCWEB プロセスの比較的初期に設計されたものである。振り返ってみると、WG はこの種のメカニズムに対する熱意について楽観的すぎた。発表時点では、広く採用・実施されていない。この文章では、この執筆時点での最新技術の説明として記載されている。

多くの場合、エンドポイント (すなわち、ブラウザ) は、接続先のシグナリングサービスを信頼せずに、相手側のエンドポイントを直接識別できることが望ましい。例えば、ユーザは、相手側の直接認証を取得するために、フェデレーションシステムを介して通話を行う場合がある。または、最低限信頼しているサイト (ポーカーサイトなど) で呼び出ししているが、信頼しているサイト (ソーシャルネットワークなど) で ID を持っている人に呼び出しをかけている場合もある。

最近、多くの Web ベースの ID 技術 (OAuth、Facebook Connect など) が開発されている。詳細は様々だが、これらの技術に共通するのは、アリスのアイデンティティを証明するウェブベースの (すなわち、HTTP/HTTPS) IdP を持っているということである。例えば、アリスが example.org のアカウントを持っている場合、アリスは example.org IdP を使用してアリスが alice@example.org であることを他のユーザに証明できる。これらの技術の開発により、通話と ID 提供を分離することができる。アリスはポーカーサイトであなたを呼び出すことができるが、自分自身を alice@example.org と識別する。

基盤となる技術が何であれ、一般的な原則は、認証される相手はシグナリングサイトではなく、むしろユーザ (とそのブラウザ) であるということである。同様に、リライングパーティ (Relying Party) はブラウザであり、シグナリングサイトではない。したがって、ブラウザは IdP アサーションプロセスへの入力を生成し、通話サイトが模倣できない方法で検証プロセスの結果をユーザに表示する必要がある[MUST]。

この文章で定義されているメカニズムでは、ブラウザが特定の ID プロトコルを実装したり、特定の IdP をサポートしたりする必要はない。代わりに、この文章では、任意の IdP が実装できる汎用インタフェースを提供する。したがって、新しい IdP とプロトコルは、ブラウザまたは通話サービスのいずれかに変更することなく導入できる。これにより、特定の ID プロトコルにコミットする必要がなくなるが、優れたパフォーマンスや UI プロパティを提供するために、ブラウザがいくつかの ID プロトコルを直接実装することを選択する場合もある。

7.1. 信頼関係：IdP、AP、および RP

フェデレーション ID プロトコルには、次の 3 つの主要な参加者がある。

認証パーティ (AP) : アイデンティティを確立しようとしているエンティティ。

ID プロバイダ (IdP) : AP の ID を保証しているエンティティ。

リライングパーティ (RP) : AP の ID を検証しようとしているエンティティ。

AP と IdP には、ある種のアカウント関係がある。AP は IdP に登録し、その後 IdP に直接認証できる (例：パスワード付き)。これは、ユーザがアカウント関係を持つ IdP をブラウザが何らかの方法で知る必要があることを意味する。これには、ユーザがブラウザに設定するものと、通話サイトで設定され、通話サイトの Web アプリケーションによって PeerConnection に提供されるものがある。この情報をブラウザに設定するユースケースは、ユーザがブラウザに「ログイン」して何らかの ID にバインドすることである。これは新しいブラウザで一般的になりつつある。ただし、IdP 情報は、通話アプリケーションによって単に提供されることも可能である必要がある。

高レベルでは、2 種類の IdP がある。

権限: ID 空間の一部のセクションの検証可能な制御を持つ IdP。例えば、電子メールの領域では、「example.com」の演算子が「@example.com」で終わる名前空間を完全に制御している。したがって、 「alice@example.com」はオペレータが言う相手である。信頼できる IdP を持つシステムの例としては、SIP の ID システムである DNSSEC ([RFC8224] を参照)、Facebook Connect (Facebook の ID は Facebook システムのコンテキスト内でのみ意味を持つ) などがある。

サードパーティ: IdP は、ID 空間の自分のセクションを制御できず、代わりに何らかの不特定のメカニズムを介してユーザの ID を検証し、それを証明する。IdP は実際には名前空間を制御しないため、RP は IdP が AP ID を正しく検証していることを信頼する必要があり、ID 空間の同じセクションを証明する複数の IdP が存在する可能性がある。サードパーティの IdP の最もよく知られた例は、SSL/TLS 証明書であり、多数の認証局 (CA) があり、そのすべてが任意のドメイン名を証明できる。

AP が権限のある IdP を介して認証している場合、RP は IdP の信頼を明示的に設定する必要はまったくない。ID メカニズムは、IdP が実際に関連する ID アサーション (この文章のメカニズムによって提供される機能) を作成したことを直接検証でき、それが権限を持つ ID について行うすべてのアサーションは直接検証可能である。これは、IdP が嘘をつかない可能性があるという意味ではなく、ユーザが ID を確認するときに行うことができる信頼性の判断であることに注意すること。

対照的に、AP がサードパーティの IdP を介して認証している場合、RP はその IdP を明示的に信頼する必要がある (そのため、PKI ベースの SSL/TLS クライアントでは明示的なトラストアンカーリストが必要である)。信頼できる IdP のリストは、ユーザによって、または場合によってはブラウザの製造元によって、ブラウザに直接設定する必要がある。これは、権限のある IdP の大きな利点であり、サードパーティの IdP をサポートする場合は、潜在的な数をかなり少なくする必要があることを意味する。

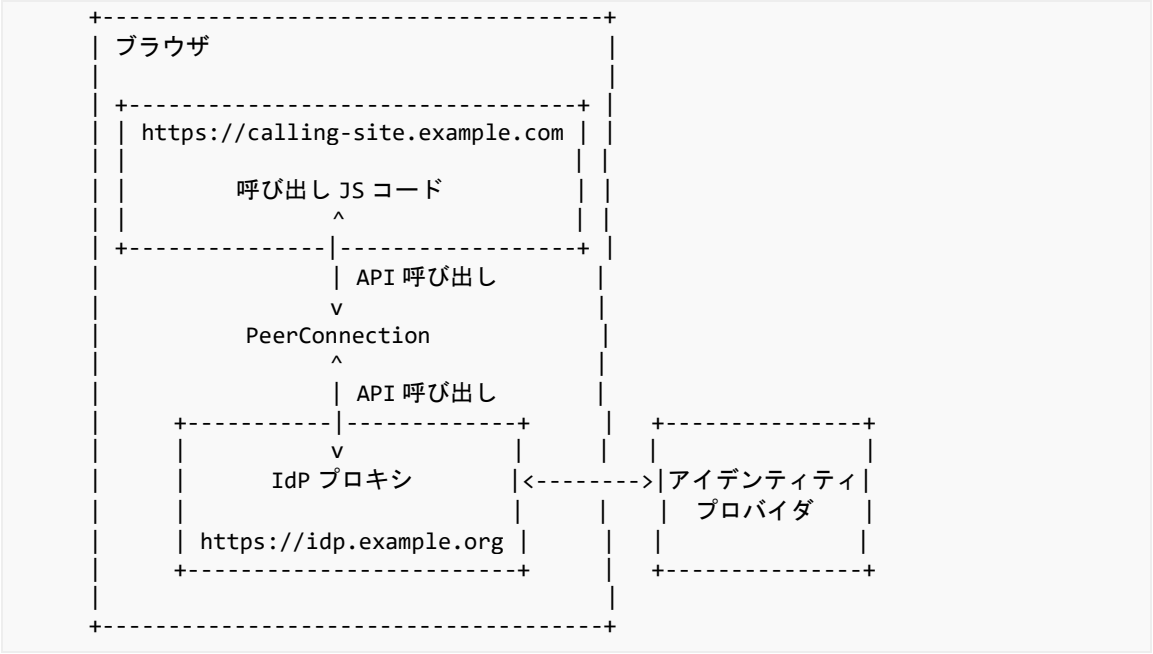
7.2. 運用概要

通話サイトを信頼せずにセキュリティを提供するには、ブラウザの PeerConnection コンポーネントが IdP と直接対話する必要がある。メカニズムの詳細は W3C API 仕様に記載されているが、一般的な考え方は、PeerConnection コンポーネントが IdP ドメイン名によって指定された IdP 上の特定の場所から JS をダウンロードするというものである。その JS (「IdP プロキシ」) はブラウザ内の分離されたセキュリティコンテキストで実行され、PeerConnection は安全なメッセージパッシングチャネルを介してそれと通信する。

ここで、2 つの論理的に分離された関数があることに注意すること。

- ID アサーションの生成。
- ID アサーションの検証。

両方の機能に同じ IdP JS「endpoint」が使用されるが、当然ながら、指定された IdP は異なる動作をし、新しい JS をロードしていずれかの機能を実行する場合がある。



PeerConnection オブジェクトが IdP と対話する場合、イベントのシーケンスは次のとおりである。

1. ブラウザ (PeerConnection コンポーネント) は IdP プロキシをインスタンス化する。これにより、IdP は必要な JS をプロキシにロードできる。生成されたコードは、IdP のセキュリティコンテキストで実行される。
2. IdP は、[webrtc-api] で定義されている API に準拠するオブジェクトをブラウザに登録する。
3. ブラウザは、IdP プロキシによって登録されたオブジェクトのメソッドを呼び出して、ID アサーションを作成または検証する。

このアプローチにより、ブラウザを特定の IdP から切り離すことができる。ブラウザは、IdP の JavaScript (IdP の ID に基づいて場所が決定される) をロードする方法と、ID アサーションを要求および検証するための汎用 API を呼び出す方法を知っているだけで済む。IdP は、汎用プロトコルを IdP の特定の要件にブリッジするために必要なあらゆるロジックを提供する。したがって、単一のブラウザは、ブラウザが作成された時点では存在しなかった IdP との上位互換性を含め、任意の数の ID プロトコルをサポートできる。

7.3. 標準化項目

この作業には 2 つの部分がある。

- ユーザの ID に暗号的にバインドする必要があるシグナリングメッセージからの正確な情報と、JavaScript Session Establishment Protocol (JSEP) メッセージでアサーションを伝達するメカニズム。これは 7.4 節で規定されている。
- 付属の W3C WebRTC API 仕様 [webrtc-api] で定義されている IdP へのインタフェース。

WebRTC API 仕様では、通話アプリケーションが使用する IdP を指定するために使用できる JavaScript インタフェースも定義されている。この API は、アサーション生成機能と検証プロセスのステータスへのアクセスも提供する。

7.4. JSEP オファー/アンサートランザクションへの ID アサーションのバインド

ID アサーションは、ユーザの ID (IdP によってアサートされる) を SDP オファー/アンサー交換、特にメディアにバインドする。これを実現するには、PeerConnection が ID にバインドされる DTLS-SRTP フィンガープ

リントを提供する必要がある。これは、次に示すように、1 つの "fingerprint" キーを持つ JavaScript オブジェクト (辞書またはハッシュとも呼ばれる連想配列) として提供される。

```
{
  "fingerprint":
    [
      { "algorithm": "sha-256",
        "digest": "4A:AD:B9:B1:3F:...:E5:7C:AB" },
      { "algorithm": "sha-1",
        "digest": "74:E9:76:C8:19:...:F4:45:6B" }
    ]
}
```

"fingerprint" 値はオブジェクトの配列である。配列内の各オブジェクトには "algorithm" 値と"digest" 値が含まれており、これらは SDP の "fingerprint" 属性のアルゴリズム値とダイジェスト値に直接対応している [RFC8122]。

このオブジェクトは、IdP に渡すための JSON [RFC8259] 文字列にエンコードされる。"identity" 属性でエンコードされた IdP によって返される ID アサーションは、7.4.1 項で説明されているようにエンコードされた JSON オブジェクトである。

この構造体は、IdP または IdP プロキシによって解釈される必要はない。RP のブラウザだけが使用する。IdP は、証明すべき不透明な値として扱うだけである。したがって、IdP を変更せずに新しいパラメータをアサーションに追加できる。

7.4.1. ID アサーションの伝送

IdP が生成したアサーション (7.6 節を参照) は、SDP オファー/アンサーメッセージに添付される。これは、SDP に新しい "identity" 属性を追加することによって行われる。この値の唯一の内容は、ID アサーションである。IdP によって生成された ID アサーションは、UTF-8 の JSON テキストにエンコードされた後、base64 エンコード [RFC4648] によって、この文字列が生成される。例えば、

```
v=0
o=- 1181923068 1181923196 IN IP4 ua1.example.com
s=example1
c=IN IP4 ua1.example.com
a=fingerprint:sha-1 \
   4A:AD:B9:B1:3F:82:18:3B:54:02:12:DF:3E:5D:49:6B:19:E5:7C:AB
a=identity: \
   eyJpZHAiOnsiZG9tYWluIjoiZXhhbXBsZS5vcmciLCJwcm90b2NvbCI6ImJvZ3Vz\
   In0sImFzc2VydGlvbiI6IntcImlkZW50aXR5XCI6XCJib2JAZXhhbXBsZS5vcmdc\
   IixcImNvbnRlbnRzXCI6XCJhYmNkZWZnaGlqa2xtbm9wcXJzdHV2d3l6XCIsXCJz\
   aWduYXR1cmVcIjpcIjAxMDIwMzA0MDUwNlwifSJ9
a=...
t=0 0
m=audio 6056 RTP/SAVP 0
a=sendrecv
...
```

この例の長い行は、この文書の列幅の制約を満たすように折りたたまれていることに注意する。行末のバックスラッシュ (「\」)、それに続くキャリッジリターン、空白は無視する。

"identity" 属性は、セッション記述内のすべての "fingerprint" 属性を証明する。したがって、これはセッションレベルの属性である。

複数の "fingerprint" 値を使用して、ピアの代替証明書を提供できる。"identity" 属性には、セッション記述の "fingerprint" 属性に含まれるすべての "fingerprint" 値を含める必要がある[MUST]。

RP ブラウザは、アサーションを検証するときに、DTLS 接続の使用中の証明書が IdP から返されたフィンガープリントのセットに含まれていることを確認する必要がある[MUST]。

7.5. IdP URI の決定

IdP が、ドメイン所有者のサーバ (例えば、共有ホスティングシナリオなど) にアカウントを持っているだけの人ではなく、ドメイン所有者の管理下にあることを保証するために、IdP JavaScript は、IdP のドメイン名に基づいて決定される場所でホストされる。各 IdP プロキシインスタンスは、次の 2 つの値に関連付けられている。

authority: IdP のサービスがホストされている権限 [RFC3986] 。

protocol: IdP が使用している特定の IdP プロトコル。これは完全に不透明な IdP 固有の文字列であるが、IdP は 2 つのプロトコルを並行して実装できる。この値は空の文字列である場合がある。protocol の値が指定されていない場合は、"default" の値が使用される。

各 IdP は、well-known URI [RFC8615] から初期エントリページ (つまり、IdP プロキシによってロードされたもの) を提供する必要がある[MUST]。IdP プロキシの well-known URI は、次の URI コンポーネントから形成される。

1. スキーム「https:」。IdP は、HTTPS [RFC2818] を使用してロードする必要がある[MUST]。
2. 権限 [RFC3986] 。前述のように、権限にはデフォルト以外のポート番号または userinfo サブコンポーネントを含めることができる[MAY]。どちらも、アサートされた ID が IdP の名前と一致するかどうかを判断するときに削除される。
3. 「/.well-known/idp-proxy/」で始まり、IdP プロトコルが付加されたパス。攻撃者が制御された「/.well-known/」 プレフィックスの外部に要求を送信できないように、区切り文字 「/」(%2F) と「\」(%5C) をプロトコルフィールドに許可してはならないことに注意すること[MUST NOT]。クエリとフラグメントの値は、「?」または「#」文字を含めることで使用できる[MAY]。

例えば、IdP 「identity.example.com」とプロトコル「example」の場合、URL は次のようになる。

```
https://identity.example.com/well-known/idp-proxy/example
```

IdP は要求をこの URL にリダイレクトできるが[MAY]、「https:」スキームを保持する必要がある[MUST]。これにより、IdP の有効な発信元は変更されるが、IdP がアサートおよび検証を許可されている ID のドメインは変更されない。つまり、IdP は依然として元のドメインに対して権限があると見なされる。

7.5.1. 認証パーティ (Authenticating Party)

AP が適切な IdP ドメインを決定する方法は、この仕様の範囲外である。ただし、この ID は IdP が証明しているものであるため、一般に、AP は IdP と何らかの実際のアカウント関係を持っている。したがって、AP は何らかの方法でブラウザに IdP 情報を提供する。考えられるメカニズムには次のようなものがある。

- ユーザが直接提供。
- 通話サイトが認識している IdP のセットから選択される (例 :「Facebook Connect 経由で認証する」と表示されるボタン)。

7.5.2. リライングパーティ (Relying Party)

AP とは異なり、RP は IdP と特定の関係を持つ必要はない。むしろ、AP から提供されるすべてのアサーションを処理できる必要がある。アサーションには、JSON エンコードされたオブジェクト (7.6 節を参照) の "idp" フィールドに IdP の ID が含まれているため、URI をアサーションから直接構築できるため、RP はユーザの操作なしで、アサーションの技術的な妥当性を直接検証できる。権限のあるアサーションは、検証可能である必要がある。また、サードパーティのアサーションは、8.1 節で説明されているように、ローカルポリシーに対して検証する必要がある[MUST]。

7.6. アサーションの要求

ID アサーション生成プロセスへの入力は、ブラウザが使用しようとしている証明書フィンガープリントのセットを含む、7.4 節で説明されている JSON エンコードされたオブジェクトである。この文字列は、IdP の観点からは不透明として扱われる。

また、ブラウザは PeerConnection が実行されている源泉 (origin) を識別するため、IdP は誰がアサーションを要求しているかに基づいて決定を下すことができる。

アプリケーションは、IdP を指定するときに、オプションでユーザ識別子ヒントを提供できる。この値は、IdP が複数の ID から選択するため、不要な ID のアサーションを提供しないようにするために使用できるヒントである。「username」は、IdP 以外のエンティティに対して意味を持たない文字列である。これには、正しくアサーションを生成するために IdP が必要とする任意のデータを含めることができる。

IdP によって正常に提供される ID アサーションは、次の情報で構成される。

idp: IdP のドメイン名とプロトコル文字列。これは、アサーションを生成したものとは異なる IdP またはプロトコルを識別する場合がある[MAY]。

assertion: アサーション自体を含む不透明な値。これは、識別された IdP またはクライアントで実行されている IdP コードによってのみ解釈可能である。

図 5 に、JSON 形式のアサーションの例を示す。この場合、メッセージは IdP が後で検証できるように何らかの方法でデジタル署名/MAC 処理されていると考えられるが、これは実装の詳細であり、この文書の範囲外である。

```
{
  "idp":{
    "domain": "example.org",
    "protocol": "bogus"
  },
  "assertion": "{\"identity\":\"bob@example.org\",
                 \"contents\":\"abcdefghijklmnopqrstuvwyz\",
                 \"signature\":\"010203040506\"}"
}
```

図 5：アサーションの例

シグナリングで使用するために、アサーションは JSON にシリアル化され、base64 エンコード [RFC4648] され、"identity" 属性の値として使用される。IdP は、生成するすべてのアサーションが別のコンテキストで解釈されないようにする必要がある[SHOULD]。例えば、異なる形式を使用するか、またはアサーション生成とその他の目的のために別々の暗号鍵を持つ必要がある。改行は読みやすくするためだけに挿入される。

7.7. ユーザログインの管理

ID アサーションを生成するために、IdP はユーザの ID の証明を必要とする。一般的には、 (パスワードまたは多要素認証を使用して) ユーザを認証し、その後の交換に Cookie [RFC6265] または HTTP 認証 [RFC7617] を使用する。

IdP プロキシは、IdP 発信元のセキュリティコンテキストで動作するため、Cookie、HTTP 認証データ、またはその他の永続セッションデータにアクセスできる。したがって、ユーザがログインしている場合、IdP には、アサーションの生成に必要なすべての情報が含まれている可能性がある。

ユーザがログインしていない場合、または IdP がユーザと対話してより多くの情報を取得してから、アサーションを生成する必要がある場合、IdP プロキシはアサーションを生成できない。IdP がアサーションを生成する前にユーザと対話する必要がある場合、IdP プロキシはアサーションの生成に失敗し、代わりにログインを続行する URL を示すことができる。

その後、アプリケーションは提供された URL をロードして、ユーザが資格情報を入力できるようにすることができる。アプリケーションと IdP 間の通信については、[webrtc-api] で説明されている。

8. アサーションの検証

ID 検証への入力は、デコードされた "identity" 属性から取得されたアサーション文字列である。

IdP プロキシはアサーションを検証する。ID プロトコルによっては、プロキシが IdP サーバまたは他のサーバに接続する場合がある。例えば、OAuth ベースのプロトコルでは、IdP をオラクルとして使用する必要があるが、署名ベースのスキームでは、関連する公開鍵がキャッシュされていれば、IdP に接続せずにアサーションを検証できる可能性がある。

メカニズムに関係なく、検証が成功した場合、IdP プロキシからの正常な応答は次の情報で構成される。

identity: IdP から見た AP の ID。詳細については、8.1 節.を参照。

contents: AP がアサーション生成プロセスへの入力として提供する、変更されていない元の文字列。

図 6 は、JSON 形式の応答の例を示している。

```
{
  "identity": "bob@example.org",
  "contents": "{\"fingerprint\":[ ... ]}"
}
```

図 6：検証結果の例

8.1. ID 形式

IdP から RP ブラウザに提供される ID は、ユーザの ID を表す文字列で構成されている必要がある[MUST]。この文字列の形式は「<user>@<domain>」で、「user」は任意の文字で構成され、domain は U ラベルのシーケンスとしてエンコードされた国際化ドメイン名 [RFC5890] である。

PeerConnection API は、この文字列を次のようにチェックする必要がある[MUST]。

1. 文字列の「domain」部分が IdP プロキシのドメイン名と等しい場合、このドメインに対しては IdP が権限を持つため、アサーションは有効である。ドメイン名の比較は、[RFC5890] の 2.3.2.4 で定義されているラベル等価規則を使用して行われる。

2. 文字列の「domain」部分が IdP プロキシのドメイン名と等しくない場合、PeerConnection オブジェクトは、次の両方がない限り、アサーションを拒否する必要がある[MUST]。

    1. IdP ドメインが許容可能なサードパーティの IdP として信頼されている。

    2. ローカルポリシーは、ID 文字列のドメイン部分についてこの IdP ドメインを信頼するように設定される。

ID の「user」部分の「@」または「%」文字は、[RFC3986] の 2.1 節で定義されている「percent-encoding」規則に従ってエスケープしなければならない[MUST]。「@」と「%」以外の文字はパーセントエンコードしてはいけない[MUST NOT]。例えば、「user」が「user@133」で「domain」が「identity.example.com」の場合、結果の文字列は「user%40133@identity.example.com」としてエンコードされる。

エスケープされた「@」文字を含むユーザ ID を表示する場合は注意が必要である。このような文字が表示される前にエスケープされていない場合、実装では、IdP プロキシのドメインと、エスケープされた「@」記号の後に表示される「<user>」部分によって暗示される可能性のあるドメインを区別する必要がある[MUST]。

9. セキュリティに関する考慮事項

RTCWEB のセキュリティ分析の多くは、[RFC8826] または上記の特定の問題の議論に含まれている。繰り返しを避けるために、このセクションでは、(a) この文書で対処されていない残留脅威、および (b) システム内のコンポーネントの 1 つの障害/誤動作によって発生する脅威に焦点を当てる。

9.1. 通信セキュリティ

シグナリングサーバへの通信を保護するために HTTPS が使用されておらず、7 章で使用されている ID メカニズムが使用されていない場合、パス上の攻撃者はハンドシェイク内の DTLS-SRTP フィンガープリントを置き換えることができるため、いずれかのエンドポイントの ID を自身の ID に置き換えることができる。

HTTPS が使用されている場合でも、実装に独立してキーを検証する何らかのメカニズムがない限り、シグナリングサーバは中間者攻撃を行う可能性がある。6.5 節の UI 要件は、モチベーションの高い/セキュリティ意識の高いユーザにこのようなメカニズムを提供するように設計されているが、一般的な使用には適していない。7 章の ID サービスメカニズムは、一般的な使用により適している。ただし、悪意のあるシグナリングサービスは、このような ID アサートをすべて取り除くことができるが、新しい ID アサートを偽造することはできない。利用可能なサードパーティのセキュリティメカニズム (X.509 証明書かサードパーティの IdP か) はすべてサードパーティのセキュリティに依存していることに注意すること。これはもちろん、ユーザの Web サイト自体への接続にも当てはまる。悪意のある IdP に対するセキュリティを保証したいユーザは、ピアのクレデンシャルを直接検証することによってのみ保証できる (例えば、ピアのフィンガープリントをアウトオブバンドで配信される値と照合するなど)。

悪意のあるコンテンツ JavaScript から保護するために、その JavaScript が DTLS キーに直接アクセスしたり、DTLS キーを使用して計算を実行したりすることを許可してはならない[MUST NOT]。例えば、コンテンツ JS がデジタル署名を計算できる場合、コンテンツ JS はブラウザが生成したキーの ID アサーションを取得し、そのアサーションとキーによる署名を使用して、コンテンツ JS によって制御される一時的な Diffie-Hellman (DH) キーで保護された通話を認証することができ、IdP メカニズムによって提供されるセ

キュリティ保証に違反する可能性がある。コンテンツ JS のキーへの直接アクセスを拒否するだけでは十分ではないことに注意すること。WebCrypto API [webcrypto] を使用することを提案する人もいる。また、JS は DTLS エンドポイントに対して有効な操作を実行できないようにする必要がある。最も安全な方法は、単にキーに関連付けられた秘密情報に依存する操作を実行する機能を拒否することである。公開鍵のエクスポートなど、公開情報に依存する操作はもちろん安全である。

9.2. プライバシー

この文書の要件は、次のことを可能にすることを目的としている。

- 位置を明らかにせずに通話に参加するユーザ。
- 通話に応答することに同意する前に、位置情報やプレゼンスステータスさえも明らかにしないようにする潜在的な着信者。

ただし、これらのプライバシー保護は、TURN リレーを使用し、後者の場合は ICE を遅延させるという点でパフォーマンス上のコストがかかる。サイトはこれらのトレードオフをユーザに認識させるべきである[SHOULD]。

ここで提供される保護は、悪意のない通話サービスを前提としていることに注意すること。通話サービスは常にユーザの状態と (Tor のような技術を使わずに) IP アドレスを知っているので、ユーザのプライバシーを意のままに侵害することができる。使用している通話サイトに対してプライバシーを要求するユーザは、Tor などの別のプライバシー強化技術を使用する必要がある。WebRTC/Tor の組み合わせ実装は、Tor を介した信号だけでなく、メディアをルーティングするように手配すべきである[SHOULD]。現在のところ、これにより最適なパフォーマンスが得られない。

さらに、複数の通話にわたって保持される識別子は、特に匿名通話サービスにとって、プライバシーの問題となる可能性がある。そのようなサービスは、通話ごとに別々の DTLS キーを使用し、通話全体で TURN を使用するようにブラウザに指示すべきである[SHOULD]。そうしないと、相手側はリンク可能な情報を学習し、複数の呼び出し間でブラウザを関連付けることができる。さらに、ブラウザは [RFC7022] のプライバシー保護 CNAME 生成モードを実装すべきである[SHOULD]。

9.3. サービス拒否

この文書で説明する同意メカニズムは、攻撃者がクライアントを使用して被害者の同意なしに大量のトラフィックを被害者に送信するサービス拒否 (DoS) 攻撃を軽減することを目的としている。これらのメカニズムは、WebRTC をまったく実装していない被害者を保護するには十分であるが、WebRTC の実装にはさらに注意が必要である。

WebRTC を介して通話を受け入れるコールセンターの場合を考えてみる。攻撃者はコールセンターのフロントエンドをプロキシし、複数のクライアントがコールセンターへの通話を開始できるように手配する。これには多くの場合ユーザの同意が必要であるが、データチャネルは同意を必要としないため、ユーザはそれを直接使用できる。ICE が完了するため、データチャネルをまったくサポートしていない場合、ブラウザは被害者のコールセンターに大量のデータを送信するように誘導される可能性がある。この攻撃を防ぐには、自動化された WebRTC 実装が適切なフロー制御を実装し、不適切な動作をしている (つまり、ICE プローブへの応答を停止する) 呼び出しをトリアージする機能を備えている必要がある。特に、 (攻撃者が制御できない) 妥当な音声やビデオがない場合に、リモートでデータチャネルをスロットルする準備ができている必要がある 。

もう 1 つの関連する攻撃は、信号サービスが音声ストリームとビデオストリームの ICE 候補を交換するこ

とで、他の被害者が音声を含むと予想している (おそらく音声だけを期待しているのだろう!) ビデオをブラウザがシンクに送信することを強制し、過負荷を引き起こす可能性がある。1 つのトランスポートで複数のメディアフローを多重化すると、ICE キープアライブを拒否して 1 つのフローを個別に抑制することが難しくなる。メディアレベル (RTCP) メカニズムを使用するか、実装で応答を完全に拒否して、通話を終了する必要がある。

マグナス・ウェスターランドによって提案された別の攻撃は、攻撃者が次のようにオファーとアンサーを相互接続することである。被害者に発信するように誘導した後、他のユーザのブラウザを制御して誰かに発信させる。そして、彼らのオファーを被害者への明らかなアンサーに変換し、それは大規模な並列分岐のように見える。被害者は依然として ICE の応答に応答し、ブラウザはすべて被害者にメディアを送信しようとする。実装は、限られた数のリモート ufrag に対する ICE バインディング要求にのみ応答することで、この攻撃から身を守ることができる (これが、JS が ufrag とパスワードを制御できないという要件の理由である)。[RFC8834] の 13 章には、RTCP ベースの DoS 攻撃の可能性と対策が数多く記載されている。

同意に関する一方的な混乱に基づく攻撃は、シグナリングメッセージの主要な部分が署名されていない限り、サードパーティの ID メカニズムに直面しても可能であることに注意すること。一方、メッセージ全体に署名すると、通話アプリケーションの機能が厳しく制限されるため、ここでは難しいトレードオフがある。

9.4. IdP 認証メカニズム

このメカニズムは、IdP 上のセキュリティと、前述のセキュリティ不変条件を正しく適用する PeerConnection に依存している。高レベルでは、IdP は、アサーションで識別されたユーザがそのアサーションとの関連付けを希望していることを証明している。したがって、任意のサードパーティがユーザに関連付けられたアサーションを取得したり、RP が受け入れるようなアサーションを生成したりすることはできないはずである。

9.4.1. PeerConnection の源泉 (origin) チェック

基本的に、IdP プロキシはブラウザによってロードされる HTML と JS の一部にすぎないため、Web 攻撃者はブラウザで生成されたものではなく、独自の IFRAME を作成し、IdP プロキシ HTML/JS をロードし、独自のキーに対して署名を要求することを止めることはできない。ただし、そのプロキシは、IdP の発信元ではなく、攻撃者の発信元にある。ブラウザ自体だけが、(a) IdP の源泉 (origin) にあり、(b) 正しい API サーフェスを公開するコンテキストをインスタンス化できる。したがって、送信側の IdP プロキシは、アサーションを発行する前に、IdP の源泉 (origin) で実行されていることを確認する必要がある[MUST]。

このチェックでは、ブラウザ (またはユーザの認証データにアクセスできる他のエンティティ) が要求を証明し、したがってフィンガープリントを証明することのみがアサートされることに注意すること。これは、ブラウザが関連付けられた秘密鍵にアクセスできることを示すものではなく、したがって、攻撃者は自分の ID を別のパーティのキーイングマテリアルに添付できるため、アリスから発信された通話は攻撃者から発信されたように見える。この形式の攻撃に対する防御については、[RFC8844] を参照。

9.4.2. IdP Well-Known URI

7.5 節で説明されているように、IdP プロキシ HTML/JS ランディングページは、IdP のドメイン名に基づく Well-Known URI にある。この要件により、IdP (例えば、Facebook のウォールで) に一部のリソースを書き込むことができる攻撃者が、IdP になりすますことができなくなる。

9.4.3. IdP 生成 ID とホスティングサイトのプライバシー

IdP のアサーションの構造によっては、通話サイトが IdP の観点からユーザの ID を学習する場合がある。多くの場合、ユーザは IdP 経由でサイトに認証されているため、これは問題ではない。

例えば、ユーザが Facebook Connect でログインした後、Facebook の ID で通話を認証している場合などである。ただし、それ以外の場合は、ユーザがまだサイトに身元を明かしていない可能性がある。一般的に、IdP は、ユーザが自分の ID をサイトに公開することを望んでいることを検証するか (例：通常の IdP 権限ダイアログを使用)、または、ID 情報が既知の RP (例えば、ソーシャルグラフの隣接関係) にのみ利用可能であり、通話サイトには利用できないように手配する必要がある[SHOULD]。アサーション要求の"domain"フィールドを使用して、ユーザが自分の ID を通話サイトに開示することに同意したことを確認できる。これは PeerConnection によって提供されるため、正しいと信頼できるからである。

9.4.4. サードパーティ IdP のセキュリティ

前述のように、各サードパーティの IdP は新しいユニバーサルトラストポイントを表すため、これらの IdP の数はかなり制限される必要がある。ほとんどの IdP は、Facebook などの非限定 ID を発行するものであっても、権威ある IdP としてリキャストできる (例：123456@facebook.com)。ただし、このような場合、ユーザインタフェースへの影響は完全には望ましくない。中間的なアプローチの 1 つは、大規模な権限のある IdP 用の特別な (ユーザが設定可能な) UI を持つことで、これにより、ユーザは通話が Facebook や Google などによって認証されていることをすぐに把握できる。

9.4.4.1. 紛らわしい文字

ID 文字列には幅広い文字が許可されているため、攻撃者が他の ID と混同しやすい ID を作成する可能性がある (このトピックの詳細については、[RFC6943] を参照)。これは、このタイプの識別子空間 (例：メールアドレス) に関する問題である。これらの識別子の作成では、スクリプトの混在や同様の紛らわしい文字を避けるべきである。これらの識別子をユーザに提示する場合は、スクリプトの混合使用のケース ([RFC5890] の 4.4 節を参照) を強調表示することを検討する必要がある。その他のベストプラクティスはまだ開発中である。

9.4.5. Web セキュリティ機能の相互作用

以下で説明するように、多くのオプションの Web セキュリティ機能は、このメカニズムに問題を引き起こす可能性がある。

9.4.5.1. ポップアップブロック

ポップアップブロックが使用されている場合、IdP プロキシはポップアップウィンドウ、ダイアログ、またはその他の形式のユーザ操作を生成できない。これにより、ユーザの操作を回避するために IdP プロキシが使用されなくなる。「LOGINNEEDED」メッセージを使用すると、IdP プロキシはユーザログインの必要性を通話サイトに通知し、IdP プロキシ自体からの直接のユーザ操作に頼らずに、この要件を満たすために必要な情報を提供できる。

9.4.5.2. サードパーティのクッキー

一部のブラウザでは、プライバシー上の理由からサードパーティの Cookie (トップレベルページ以外の源泉 (origin) に関連付けられた Cookie) をブロックすることができる。Cookie を使用してログインを永続化する IdP は、サードパーティの Cookie ブロッキングによって破棄される。一つの選択肢は、これを制限として受け入れることである。もう 1 つは、PeerConnection オブジェクトで IdP プロキシのサードパーティ Cookie

ブロッキングを無効にすることである。

10. IANA に関する考慮事項

この仕様では、[RFC4566] の 8.2.4 項の手順に従って "identity" SDP 属性を定義している。

登録に必要な情報は、[RFC8827] の原文参照のこと。

11. 参考資料

11.1. 標準参照

[FIPS186] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", NIST PUB 186-4, DOI 10.6028/NIST.FIPS.186-4, July 2013, <https://doi.org/10.6028/NIST.FIPS.186-4>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <https://www.rfc-editor.org/info/rfc2818>.

[RFC3264] Rosenberg, J. and H. Schulzrinne, "An Offer/Answer Model with Session Description Protocol (SDP)", RFC 3264, DOI 10.17487/RFC3264, June 2002, <https://www.rfc-editor.org/info/rfc3264>.

[RFC3711] Baugher, M., McGrew, D., Naslund, M., Carrara, E., and K. Norrman, "The Secure Real-time Transport Protocol (SRTP)", RFC 3711, DOI 10.17487/RFC3711, March 2004, <https://www.rfc-editor.org/info/rfc3711>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, DOI 10.17487/RFC4566, July 2006, <https://www.rfc-editor.org/info/rfc4566>.

[RFC4568] Andreasen, F., Baugher, M., and D. Wing, "Session Description Protocol (SDP) Security Descriptions for Media Streams", RFC 4568, DOI 10.17487/RFC4568, July 2006, <https://www.rfc-editor.org/info/rfc4568>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5763] Fischl, J., Tschofenig, H., and E. Rescorla, "Framework for Establishing a Secure Real-time Transport Protocol (SRTP) Security Context Using Datagram Transport Layer Security (DTLS)", RFC 5763, DOI 10.17487/RFC5763, May 2010, <https://www.rfc-editor.org/info/rfc5763>.

[RFC5764] McGrew, D. and E. Rescorla, "Datagram Transport Layer Security (DTLS) Extension to Establish Keys for the Secure Real-time Transport Protocol (SRTP)", RFC 5764, DOI 10.17487/RFC5764, May 2010, <https://www.rfc-editor.org/info/rfc5764>.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/info/rfc6454>.

[RFC7022] Begen, A., Perkins, C., Wing, D., and E. Rescorla, "Guidelines for Choosing RTP Control Protocol

(RTCP) Canonical Names (CNAMEs)", RFC 7022, DOI 10.17487/RFC7022, September 2013, <https://www.rfc-editor.org/info/rfc7022>.

[RFC7675] Perumal, M., Wing, D., Ravindranath, R., Reddy, T., and M. Thomson, "Session Traversal Utilities for NAT (STUN) Usage for Consent Freshness", RFC 7675, DOI 10.17487/RFC7675, October 2015, <https://www.rfc-editor.org/info/rfc7675>.

[RFC7918] Langley, A., Modadugu, N., and B. Moeller, "Transport Layer Security (TLS) False Start", RFC 7918, DOI 10.17487/RFC7918, August 2016, <https://www.rfc-editor.org/info/rfc7918>.

[RFC8122] Lennox, J. and C. Holmberg, "Connection-Oriented Media Transport over the Transport Layer Security (TLS) Protocol in the Session Description Protocol (SDP)", RFC 8122, DOI 10.17487/RFC8122, March 2017, <https://www.rfc-editor.org/info/rfc8122>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/info/rfc8259>.

[RFC8261] Tuexen, M., Stewart, R., Jesup, R., and S. Loreto, "Datagram Transport Layer Security (DTLS) Encapsulation of SCTP Packets", RFC 8261, DOI 10.17487/RFC8261, November 2017, <https://www.rfc-editor.org/info/rfc8261>.

[RFC8445] Keranen, A., Holmberg, C., and J. Rosenberg, "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal", RFC 8445, DOI 10.17487/RFC8445, July 2018, <https://www.rfc-editor.org/info/rfc8445>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

[RFC8825] Alvestrand, H., "Overview: Real-Time Protocols for Browser-Based Applications", RFC 8825, DOI 10.17487/RFC8825, January 2021, <https://www.rfc-editor.org/info/rfc8825>.

[RFC8826] Rescorla, E., "Security Considerations for WebRTC", RFC 8826, DOI 10.17487/RFC8826, January 2021, <https://www.rfc-editor.org/info/rfc8826>.

[RFC8829] Uberti, J., Jennings, C., and E. Rescorla, Ed., "JavaScript Session Establishment Protocol (JSEP)", RFC 8829, DOI 10.17487/RFC8829, January 2021, <https://www.rfc-editor.org/info/rfc8829>.

[RFC8834] Perkins, C., Westerlund, M., and J. Ott, "Media Transport and Use of RTP in WebRTC", RFC 8834, DOI 10.17487/RFC8834, January 2021, <https://www.rfc-editor.org/info/rfc8834>.

[RFC8844] Thomson, M. and E. Rescorla, "Unknown Key-Share Attacks on Uses of TLS with the Session Description Protocol (SDP)", RFC 8844, DOI 10.17487/RFC8844, January 2021, <https://www.rfc-editor.org/info/rfc8844>.

[webcrypto] Watson, M., "Web Cryptography API", W3C Recommendation, 26 January 2017, <https://www.w3.org/TR/2017/REC-WebCryptoAPI-20170126/>.

[webrtc-api] Jennings, C., Boström, H., and J-I. Bruaroey, "WebRTC 1.0: Real-time Communication Between Browsers", W3C Proposed Recommendation, <https://www.w3.org/TR/webrtc/>.

11.2. 参考文献

[fetch] van Kesteren, A., "Fetch", <https://fetch.spec.whatwg.org/>.

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E.

Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

[RFC5705] Rescorla, E., "Keying Material Exporters for Transport Layer Security (TLS)", RFC 5705, DOI 10.17487/RFC5705, March 2010, <https://www.rfc-editor.org/info/rfc5705>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <https://www.rfc-editor.org/info/rfc6120>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <https://www.rfc-editor.org/info/rfc6265>.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, DOI 10.17487/RFC6455, December 2011, <https://www.rfc-editor.org/info/rfc6455>.

[RFC6943] Thaler, D., Ed., "Issues in Identifier Comparison for Security Purposes", RFC 6943, DOI 10.17487/RFC6943, May 2013, <https://www.rfc-editor.org/info/rfc6943>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/info/rfc7617>.

[RFC8224] Peterson, J., Jennings, C., Rescorla, E., and C. Wendt, "Authenticated Identity Management in the Session Initiation Protocol (SIP)", RFC 8224, DOI 10.17487/RFC8224, February 2018, <https://www.rfc-editor.org/info/rfc8224>.

[RFC8828] Uberti, J. and G. Shieh, "WebRTC IP Address Handling Requirements", RFC 8828, DOI 10.17487/RFC8828, January 2021, <https://www.rfc-editor.org/info/rfc8828>.

[TLS-DTLS13] Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-39, 2 November 2020, <https://tools.ietf.org/html/draft-ietf-tls-dtls13-39>.