Network Working Group
Request for Comments: 4313
Category: Informational

D. Oran
Cisco Systems, Inc.

December 2005

Requirements for Distributed Control of Automatic Speech Recognition (ASR), Speaker Identification/Speaker Verification (SI/SV), and Text-to-Speech (TTS) Resources

Status of this Memo

Copyright Notice

Abstract

This document outlines the needs and requirements for a protocol to control distributed speech processing of audio streams. By speech processing, this document specifically means automatic speech recognition (ASR), speaker recognition -- which includes both speaker identification (SI) and speaker verification (SV) -- and text-to-speech (TTS). Other IETF protocols, such as SIP and Real Time Streaming Protocol (RTSP), address rendezvous and control for generalized media streams. However, speech processing presents additional requirements that none of the extant IETF protocols address.

Table of Contents

1. Introduction

There are multiple IETF protocols for establishment and termination of media sessions (SIP [6]), low-level media control (Media Gateway Control Protocol (MGCP) [7] and Media Gateway Controller (MEGACO) [8]), and media record and playback (RTSP [9]). This document focuses on requirements for one or more protocols to support the control of network elements that perform Automated Speech Recognition (ASR), speaker identification or verification (SI/SV), and rendering text into audio, also known as Text-to-Speech (TTS). Many multimedia applications can benefit from having automatic speech recognition (ASR) and text-to-speech (TTS) processing available as a distributed, network resource. This requirements document limits its focus to the distributed control of ASR, SI/SV, and TTS servers.

There is a broad range of systems that can benefit from a unified approach to control of TTS, ASR, and SI/SV. These include environments such as Voice over IP (VoIP) gateways to the Public Switched Telephone Network (PSTN), IP telephones, media servers, and wireless mobile devices that obtain speech services via servers on the network.

To date, there are a number of proprietary ASR and TTS APIs, as well as two IETF documents that address this problem [13], [14]. However, there are serious deficiencies to the existing documents. In particular, they mix the semantics of existing protocols yet are close enough to other protocols as to be confusing to the implementer.

This document sets forth requirements for protocols to support distributed speech processing of audio streams. For simplicity, and to remove confusion with existing protocol proposals, this document presents the requirements as being for a "framework" that addresses the distributed control of speech resources. It refers to such a framework as "SPEECHSC", for Speech Services Control.

1.1. Document Conventions

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [3].

2. SPEECHSC Framework

Figure 1 below shows the SPEECHSC framework for speech processing.

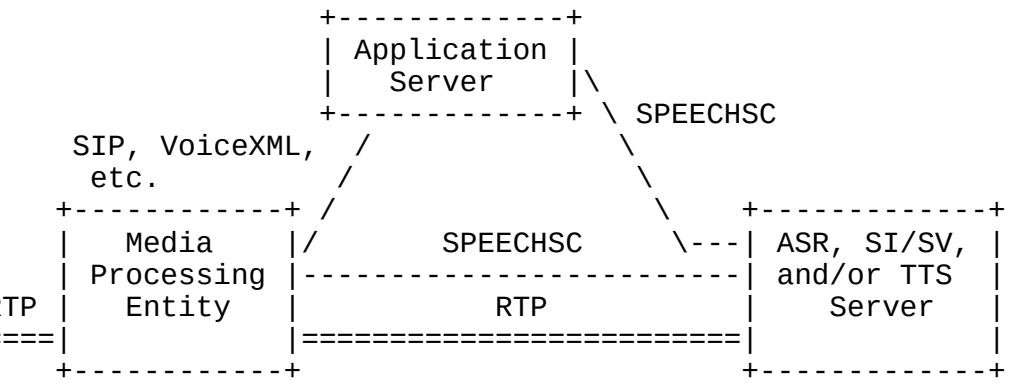

Figure 1: SPEECHSC Framework

The "Media Processing Entity" is a network element that processes media. It may be a pure media handler, or it may also have an associated SIP user agent, VoiceXML browser, or other control entity. The "ASR, SI/SV, and/or TTS Server" is a network element that performs the back-end speech processing. It may generate an RTP stream as output based on text input (TTS) or return recognition results in response to an RTP stream as input (ASR, SI/SV). The "Application Server" is a network element that instructs the Media Processing Entity on what transformations to make to the media stream. Those instructions may be established via a session protocol such as SIP, or provided via a client/server exchange such as VoiceXML. The framework allows either the Media Processing Entity or the Application Server to control the ASR or TTS Server using SPEECHSC as a control protocol, which accounts for the SPEECHSC protocol appearing twice in the diagram.

Physical embodiments of the entities can reside in one physical instance per entity, or some combination of entities. For example, a VoiceXML [11] gateway may combine the ASR and TTS functions on the same platform as the Media Processing Entity. Note that VoiceXML gateways themselves are outside the scope of this protocol. Likewise, one can combine the Application Server and Media Processing Entity, as would be the case in an interactive voice response (IVR) platform.

One can also decompose the Media Processing Entity into an entity that controls media endpoints and entities that process media directly. Such would be the case with a decomposed gateway using MGCP or MEGACO. However, this decomposition is again orthogonal to

the scope of SPEECHSC. The following subsections provide a number of example use cases of the SPEECHSC, one each for TTS, ASR, and SI/SV. They are intended to be illustrative only, and not to imply any restriction on the scope of the framework or to limit the decomposition or configuration to that shown in the example.

2.1. TTS Example

This example illustrates a simple usage of SPEECHSC to provide a Text-to-Speech service for playing announcements to a user on a phone with no display for textual error messages. The example scenario is shown below in Figure 2. In the figure, the VoIP gateway acts as both the Media Processing Entity and the Application Server of the SPEECHSC framework in Figure 1.

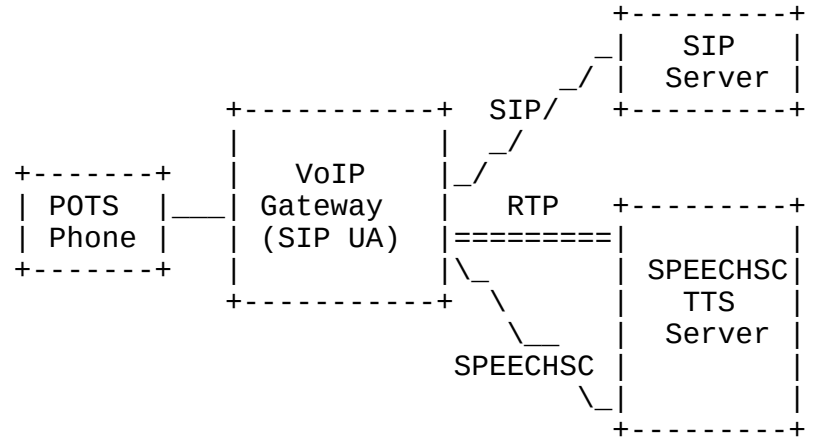

Figure 2: Text-to-Speech Example of SPEECHSC

The Plain Old Telephone Service (POTS) phone on the left attempts to make a phone call. The VoIP gateway, acting as a SIP UA, tries to establish a SIP session to complete the call, but gets an error, such as a SIP "486 Busy Here" response. Without SPEECHSC, the gateway would most likely just output a busy signal to the POTS phone. However, with SPEECHSC access to a TTS server, it can provide a spoken error message. The VoIP gateway therefore constructs a text error string using information from the SIP messages, such as "Your call to 978-555-1212 did not go through because the called party was busy". It then can use SPEECHSC to establish an association with a SPEECHSC server, open an RTP stream between itself and the server, and issue a TTS request for the error message, which will be played to the user on the POTS phone.

2.2. Automatic Speech Recognition Example

This example illustrates a VXML-enabled media processing entity and associated application server using the SPEECHSC framework to supply an ASR-based user interface through an Interactive Voice Response (IVR) system. The example scenario is shown below in Figure 3. The VXML-client corresponds to the "media processing entity", while the IVR application server corresponds to the "application server" of the SPEECHSC framework of Figure 1.

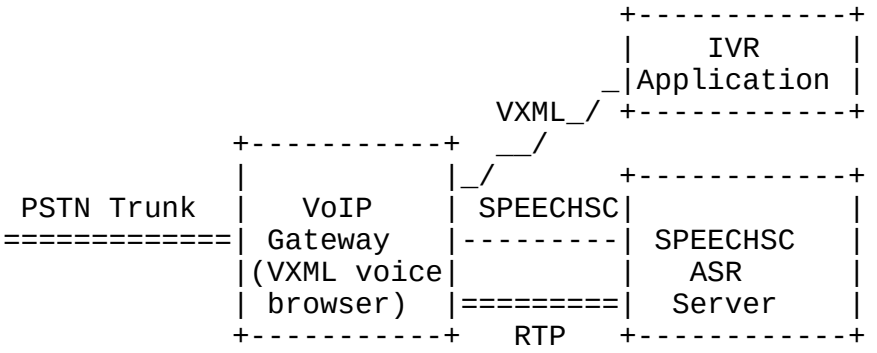

Figure 3: Automatic Speech Recognition Example

In this example, users call into the service in order to obtain stock quotes. The VoIP gateway answers their PSTN call. An IVR application feeds VXML scripts to the gateway to drive the user interaction. The VXML interpreter on the gateway directs the user's media stream to the SPEECHSC ASR server and uses SPEECHSC to control the ASR server.

When, for example, the user speaks the name of a stock in response to an IVR prompt, the SPEECHSC ASR server attempts recognition of the name, and returns the results to the VXML gateway. The VXML gateway, following standard VXML mechanisms, informs the IVR Application of the recognized result. The IVR Application can then do the appropriate information lookup. The answer, of course, can be sent back to the user using text-to-speech. This example does not show this scenario, but it would work analogously to the scenario shown in section Section 2.1.

2.3. Speaker Identification example

This example illustrates using speaker identification to allow voice-actuated login to an IP phone. The example scenario is shown below in Figure 4. In the figure, the IP Phone acts as both the "Media Processing Entity" and the "Application Server" of the SPEECHSC framework in Figure 1.

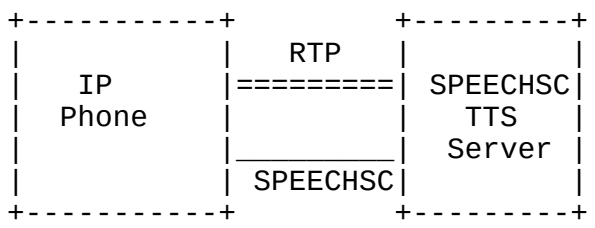

Figure 4: Speaker Identification Example

In this example, a user speaks into a SIP phone in order to get "logged in" to that phone to make and receive phone calls using his identity and preferences. The IP phone uses the SPEECHSC framework to set up an RTP stream between the phone and the SPEECHSC SI/SV server and to request verification. The SV server verifies the user's identity and returns the result, including the necessary login credentials, to the phone via SPEECHSC. The IP Phone may use the identity directly to identify the user in outgoing calls, to fetch the user's preferences from a configuration server, or to request authorization from an Authentication, Authorization, and Accounting (AAA) server, in any combination. Since this example uses SPEECHSC to perform a security-related function, be sure to note the associated material in Section 9.

3. General Requirements

3.1. Reuse Existing Protocols

To the extent feasible, the SPEECHSC framework SHOULD use existing protocols.

3.2. Maintain Existing Protocol Integrity

In meeting the requirement of Section 3.1, the SPEECHSC framework MUST NOT redefine the semantics of an existing protocol. Said differently, we will not break existing protocols or cause backward-compatibility problems.

3.3. Avoid Duplicating Existing Protocols

To the extent feasible, SPEECHSC SHOULD NOT duplicate the functionality of existing protocols. For example, network announcements using SIP [12] and RTSP [9] already define how to request playback of audio. The focus of SPEECHSC is new functionality not addressed by existing protocols or extending existing protocols within the strictures of the requirement in

Section 3.2. Where an existing protocol can be gracefully extended to support SPEECHSC requirements, such extensions are acceptable alternatives for meeting the requirements.

As a corollary to this, the SPEECHSC should not require a separate protocol to perform functions that could be easily added into the SPEECHSC protocol (like redirecting media streams, or discovering capabilities), unless it is similarly easy to embed that protocol directly into the SPEECHSC framework.

3.4. Efficiency

The SPEECHSC framework SHOULD employ protocol elements known to result in efficient operation. Techniques to be considered include:

- Re-use of transport connections across sessions
- Piggybacking of responses on requests in the reverse direction
- Caching of state across requests

3.5. Invocation of Services

The SPEECHSC framework MUST be compliant with the IAB Open Pluggable Edge Services (OPES) [4] framework. The applicability of the SPEECHSC protocol will therefore be specified as occurring between clients and servers at least one of which is operating directly on behalf of the user requesting the service.

3.6. Location and Load Balancing

To the extent feasible, the SPEECHSC framework SHOULD exploit existing schemes for supporting service location and load balancing, such as the Service Location Protocol [13] or DNS SRV records [14]. Where such facilities are not deemed adequate, the SPEECHSC framework MAY define additional load balancing techniques.

3.7. Multiple Services

The SPEECHSC framework MUST permit multiple services to operate on a single media stream so that either the same or different servers may be performing speech recognition, speaker identification or verification, etc., in parallel.

3.8. Multiple Media Sessions

The SPEECHSC framework MUST allow a 1:N mapping between session and RTP channels. For example, a single session may include an outbound RTP channel for TTS, an inbound for ASR, and a different inbound for SI/SV (e.g., if processed by different elements on the Media Resource

Element). Note: All of these can be described via SDP, so if SDP is utilized for media channel description, this requirement is met "for free".

3.9. Users with Disabilities

The SPEECHSC framework must have sufficient capabilities to address the critical needs of people with disabilities. In particular, the set of requirements set forth in RFC 3351 [5] MUST be taken into account by the framework. It is also important that implementers of SPEECHSC clients and servers be cognizant that some interaction modalities of SPEECHSC may be inconvenient or simply inappropriate for disabled users. Hearing-impaired individuals may find TTS of limited utility. Speech-impaired users may be unable to make use of ASR or SI/SV capabilities. Therefore, systems employing SPEECHSC MUST provide alternative interaction modes or avoid the use of speech processing entirely.

3.10. Identification of Process That Produced Media or Control Output

The client of a SPEECHSC operation SHOULD be able to ascertain via the SPEECHSC framework what speech process produced the output. For example, an RTP stream containing the spoken output of TTS should be identifiable as TTS output, and the recognized utterance of ASR should be identifiable as having been produced by ASR processing.

4. TTS Requirements

4.1. Requesting Text Playback

The SPEECHSC framework MUST allow a Media Processing Entity or Application Server, using a control protocol, to request the TTS Server to play back text as voice in an RTP stream.

4.2. Text Formats

4.2.1. Plain Text

The SPEECHSC framework MAY assume that all TTS servers are capable of reading plain text. For reading plain text, framework MUST allow the language and voicing to be indicated via session parameters. For finer control over such properties, see [1].

4.2.2. SSML

The SPEECHSC framework MUST support Speech Synthesis Markup Language (SSML)[1] <speak> basics, and SHOULD support other SSML tags. The framework assumes all TTS servers are capable of reading SSML

formatted text. Internationalization of TTS in the SPEECHSC framework, including multi-lingual output within a single utterance, is accomplished via SSML xml:lang tags.

4.2.3. Text in Control Channel

The SPEECHSC framework assumes all TTS servers accept text over the SPEECHSC connection for reading over the RTP connection. The framework assumes the server can accept text either "by value" (embedded in the protocol) or "by reference" (e.g., by de-referencing a Uniform Resource Identifier (URI) embedded in the protocol).

4.2.4. Document Type Indication

A document type specifies the syntax in which the text to be read is encoded. The SPEECHSC framework MUST be capable of explicitly indicating the document type of the text to be processed, as opposed to forcing the server to infer the content by other means.

4.3. Control Channel

The SPEECHSC framework MUST be capable of establishing the control channel between the client and server on a per-session basis, where a session is loosely defined to be associated with a single "call" or "dialog". The protocol SHOULD be capable of maintaining a long-lived control channel for multiple sessions serially, and MAY be capable of shorter time horizons as well, including as short as for the processing of a single utterance.

4.4. Media Origination/Termination by Control Elements

The SPEECHSC framework MUST NOT require the controlling element (application server, media processing entity) to accept or originate media streams. Media streams MAY source & sink from the controlled element (ASR, TTS, etc.).

4.5. Playback Controls

The SPEECHSC framework MUST support "VCR controls" for controlling the playout of streaming media output from SPEECHSC processing, and MUST allow for servers with varying capabilities to accommodate such controls. The protocol SHOULD allow clients to state what controls they wish to use, and for servers to report which ones they honor. These capabilities include:

- The ability to jump in time to the location of a specific marker.
- The ability to jump in time, forwards or backwards, by a specified amount of time. Valid time units MUST include seconds, words, paragraphs, sentences, and markers.
- The ability to increase and decrease playout speed.
- The ability to fast-forward and fast-rewind the audio, where snippets of audio are played as the server moves forwards or backwards in time.
- The ability to pause and resume playout.
- The ability to increase and decrease playout volume.

These controls SHOULD be made easily available to users through the client user interface and through per-user customization capabilities of the client. This is particularly important for hearing-impaired users, who will likely desire settings and control regimes different from those that would be acceptable for non-impaired users.

4.6. Session Parameters

The SPEECHSC framework MUST support the specification of session parameters, such as language, prosody, and voicing.

4.7. Speech Markers

The SPEECHSC framework MUST accommodate speech markers, with capability at least as flexible as that provided in SSML [1]. The framework MUST further provide an efficient mechanism for reporting that a marker has been reached during playout.

5. ASR Requirements

5.1. Requesting Automatic Speech Recognition

The SPEECHSC framework MUST allow a Media Processing Entity or Application Server to request the ASR Server to perform automatic speech recognition on an RTP stream, returning the results over SPEECHSC.

5.2. XML

The SPEECHSC framework assumes that all ASR servers support the VoiceXML speech recognition grammar specification (SRGS) for speech recognition [2].

5.3. Grammar Requirements

5.3.1. Grammar Specification

The SPEECHSC framework assumes all ASR servers are capable of accepting grammar specifications either "by value" (embedded in the protocol) or "by reference" (e.g., by de-referencing a URI embedded in the protocol). The latter MUST allow the indication of a grammar already known to, or otherwise "built in" to, the server. The framework and protocol further SHOULD exploit the ability to store and later retrieve by reference large grammars that were originally supplied by the client.

5.3.2. Explicit Indication of Grammar Format

The SPEECHSC framework protocol MUST be able to explicitly convey the grammar format in which the grammar is encoded and MUST be extensible to allow for conveying new grammar formats as they are defined.

5.3.3. Grammar Sharing

The SPEECHSC framework SHOULD exploit sharing grammars across sessions for servers that are capable of doing so. This supports applications with large grammars for which it is unrealistic to dynamically load. An example is a city-country grammar for a weather service.

5.4. Session Parameters

The SPEECHSC framework MUST accommodate at a minimum all of the protocol parameters currently defined in Media Resource Control Protocol (MRCP) [10] In addition, there SHOULD be a capability to reset parameters within a session.

5.5. Input Capture

The SPEECHSC framework MUST support a method directing the ASR Server to capture the input media stream for later analysis and tuning of the ASR engine.

6. Speaker Identification and Verification Requirements

6.1. Requesting SI/SV

The SPEECHSC framework MUST allow a Media Processing Entity to request the SI/SV Server to perform speaker identification or verification on an RTP stream, returning the results over SPEECHSC.

6.2. Identifiers for SI/SV

The SPEECHSC framework MUST accommodate an identifier for each verification resource and permit control of that resource by ID, because voiceprint format and contents are vendor specific.

6.3. State for Multiple Utterances

The SPEECHSC framework MUST work with SI/SV servers that maintain state to handle multi-utterance verification.

6.4. Input Capture

The SPEECHSC framework MUST support a method for capturing the input media stream for later analysis and tuning of the SI/SV engine. The framework may assume all servers are capable of doing so. In addition, the framework assumes that the captured stream contains enough timestamp context (e.g., the NTP time range from the RTP Control Protocol (RTCP) packets, which corresponds to the RTP timestamps of the captured input) to ascertain after the fact exactly when the verification was requested.

6.5. SI/SV Functional Extensibility

The SPEECHSC framework SHOULD be extensible to additional functions associated with SI/SV, such as prompting, utterance verification, and retraining.

7. Duplexing and Parallel Operation Requirements

One very important requirement for an interactive speech-driven system is that user perception of the quality of the interaction depends strongly on the ability of the user to interrupt a prompt or rendered TTS with speech. Interrupting, or barging, the speech output requires more than energy detection from the user’s direction. Many advanced systems halt the media towards the user by employing the ASR engine to decide if an utterance is likely to be real speech, as opposed to a cough, for example.

7.1. Full Duplex Operation

To achieve low latency between utterance detection and halting of playback, many implementations combine the speaking and ASR functions. The SPEECHSC framework MUST support such full-duplex implementations.

7.2. Multiple Services in Parallel

Good spoken user interfaces typically depend upon the ease with which the user can accomplish his or her task. When making use of speaker identification or verification technologies, user interface improvements often come from the combination of the different technologies: simultaneous identity claim and verification (on the same utterance), simultaneous knowledge and voice verification (using ASR and verification simultaneously). Using ASR and verification on the same utterance is in fact the only way to support rolling or dynamically-generated challenge phrases (e.g., "say 51723"). The SPEECHSC framework MUST support such parallel service implementations.

7.3. Combination of Services

It is optionally of interest that the SPEECHSC framework support more complex remote combination and controls of speech engines:

- Combination in series of engines that may then act on the input or output of ASR, TTS, or Speaker recognition engines. The control MAY then extend beyond such engines to include other audio input and output processing and natural language processing.
- Intermediate exchanges and coordination between engines.
- Remote specification of flows between engines.

These capabilities MAY benefit from service discovery mechanisms (e.g., engines, properties, and states discovery).

8. Additional Considerations (Non-Normative)

The framework assumes that Session Description Protocol (SDP) will be used to describe media sessions and streams. The framework further assumes RTP carriage of media. However, since SDP can be used to describe other media transport schemes (e.g., ATM) these could be used if they provide the necessary elements (e.g., explicit timestamps).

The working group will not be defining distributed speech recognition (DSR) methods, as exemplified by the European Telecommunications Standards Institute (ETSI) Aurora project. The working group will not be recreating functionality available in other protocols, such as SIP or SDP.

TTS looks very much like playing back a file. Extending RTSP looks promising for when one requires VCR controls or markers in the text to be spoken. When one does not require VCR controls, SIP in a framework such as Network Announcements [12] works directly without modification.

ASR has an entirely different set of characteristics. For barge-in support, ASR requires real-time return of intermediate results. Barring the discovery of a good reuse model for an existing protocol, this will most likely become the focus of SPEECHSC.

9. Security Considerations

Protocols relating to speech processing must take security and privacy into account. Many applications of speech technology deal with sensitive information, such as the use of Text-to-Speech to read financial information. Likewise, popular uses for automatic speech recognition include executing financial transactions and shopping.

There are at least three aspects of speech processing security that intersect with the SPEECHSC requirements -- securing the SPEECHSC protocol itself, implementing and deploying the servers that run the protocol, and ensuring that utilization of the technology for providing security functions is appropriate. Each of these aspects in discussed in the following subsections. While some of these considerations are, strictly speaking, out of scope of the protocol itself, they will be carefully considered and accommodated during protocol design, and will be called out as part of the applicability statement accompanying the protocol specification(s). Privacy considerations are discussed as well.

9.1. SPEECHSC Protocol Security

The SPEECHSC protocol MUST in all cases support authentication, authorization, and integrity, and SHOULD support confidentiality. For privacy-sensitive applications, the protocol MUST support confidentiality. We envision that rather than providing protocol-specific security mechanisms in SPEECHSC itself, the resulting protocol will employ security machinery of either a containing protocol or the transport on which it runs. For example, we will consider solutions such as using Transport Layer Security (TLS) for securing the control channel, and Secure Realtime Transport

Protocol (SRTP) for securing the media channel. Third-party dependencies necessitating transitive trust will be minimized or explicitly dealt with through the authentication and authorization aspects of the protocol design.

9.2. Client and Server Implementation and Deployment

Given the possibly sensitive nature of the information carried, SPEECHSC clients and servers need to take steps to ensure confidentiality and integrity of the data and its transformations to and from spoken form. In addition to these general considerations, certain SPEECHSC functions, such as speaker verification and identification, employ voiceprints whose privacy, confidentiality, and integrity must be maintained. Similarly, the requirement to support input capture for analysis and tuning can represent a privacy vulnerability because user utterances are recorded and could be either revealed or replayed inappropriately. Implementers must take care to prevent the exploitation of any centralized voiceprint database and the recorded material from which such voiceprints may be derived. Specific actions that are recommended to minimize these threats include:

- End-to-end authentication, confidentiality, and integrity protection (like TLS) of access to the database to minimize the exposure to external attack.
- Database protection measures such as read/write access control and local login authentication to minimize the exposure to insider threats.
- Copies of the database, especially ones that are maintained at off-site locations, need the same protection as the operational database.

Inappropriate disclosure of this data does not as of the date of this document represent an exploitable threat, but quite possibly might in the future. Specific vulnerabilities that might become feasible are discussed in the next subsection. It is prudent to take measures such as encrypting the voiceprint database and permitting access only through programming interfaces enforcing adequate authorization machinery.

9.3. Use of SPEECHSC for Security Functions

Either speaker identification or verification can be used directly as an authentication technology. Authorization decisions can be coupled with speaker verification in a direct fashion through challenge-response protocols, or indirectly with speaker identification through the use of access control lists or other identity-based authorization mechanisms. When so employed, there are

additional security concerns that need to be addressed through the use of protocol security mechanisms for clients and servers. For example, the ability to manipulate the media stream of a speaker verification request could inappropriately permit or deny access based on impersonation, or simple garbling via noise injection, making it critical to properly secure both the control and data channels, as recommended above. The following issues specific to the use of SI/SV for authentication should be carefully considered:

1. Theft of voiceprints or the recorded samples used to construct them represents a future threat against the use of speaker identification/verification as a biometric authentication technology. A plausible attack vector (not feasible today) is to use the voiceprint information as parametric input to a text-to-speech synthesis system that could mimic the user's voice accurately enough to match the voiceprint. Since it is not very difficult to surreptitiously record reasonably large corpuses of voice samples, the ability to construct voiceprints for input to this attack would render the security of voice-based biometric authentication, even using advanced challenge-response techniques, highly vulnerable. Users of speaker verification for authentication should monitor technological developments in this area closely for such future vulnerabilities (much as users of other authentication technologies should monitor advances in factoring as a way to break asymmetric keying systems).
2. As with other biometric authentication technologies, a downside to the use of speech identification is that revocation is not possible. Once compromised, the biometric information can be used in identification and authentication to other independent systems.
3. Enrollment procedures can be vulnerable to impersonation if not protected both by protocol security mechanisms and some independent proof of identity. (Proof of identity may not be needed in systems that only need to verify continuity of identity since enrollment, as opposed to association with a particular individual.

Further discussion of the use of SI/SV as an authentication technology, and some recommendations concerning advantages and vulnerabilities, can be found in Chapter 5 of [15].

10. Acknowledgements

Eric Burger wrote the original version of these requirements and has continued to contribute actively throughout their development. He is a co-author in all but formal authorship, and is instead acknowledged here as it is preferable that working group co-chairs have non-conflicting roles with respect to the progression of documents.

11. References

11.1. Normative References

[1] Walker, M., Burnett, D., and A. Hunt, "Speech Synthesis Markup Language (SSML) Version 1.0", W3C REC REC-speech-synthesis-20040907, September 2004.

[2] McGlashan, S. and A. Hunt, "Speech Recognition Grammar Specification Version 1.0", W3C REC REC-speech-grammar-20040316, March 2004.

[3] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[4] Floyd, S. and L. Daigle, "IAB Architectural and Policy Considerations for Open Pluggable Edge Services", RFC 3238, January 2002.

[5] Charlton, N., Gasson, M., Gybels, G., Spanner, M., and A. van Wijk, "User Requirements for the Session Initiation Protocol (SIP) in Support of Deaf, Hard of Hearing and Speech-impaired Individuals", RFC 3351, August 2002.

11.2. Informative References

[6] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[7] Andreasen, F. and B. Foster, "Media Gateway Control Protocol (MGCP) Version 1.0", RFC 3435, January 2003.

[8] Groves, C., Pantaleo, M., Ericsson, LM., Anderson, T., and T. Taylor, "Gateway Control Protocol Version 1", RFC 3525, June 2003.

[9] Schulzrinne, H., Rao, A., and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

[10] Shanmugham, S., Monaco, P., and B. Eberman, "MRCP: Media Resource Control Protocol", Work in Progress.

[11] World Wide Web Consortium, "Voice Extensible Markup Language (VoiceXML) Version 2.0", W3C Working Draft , April 2002, <http://www.w3.org/TR/2002/WD-voicexml20-20020424/>.

[12] Burger, E., Ed., Van Dyke, J., and A. Spitzer, "Basic Network Media Services with SIP", RFC 4240, December 2005.

[13] Guttman, E., Perkins, C., Veizades, J., and M. Day, "Service Location Protocol, Version 2", RFC 2608, June 1999.

[14] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[15] Committee on Authentication Technologies and Their Privacy Implications, National Research Council, "Who Goes There?: Authentication Through the Lens of Privacy", Computer Science and Telecommunications Board (CSTB) , 2003, <http://www.nap.edu/catalog/10656.html/ >.

Author's Address

David R. Oran
Cisco Systems, Inc.
7 Ladyslipper Lane
Acton, MA
USA

EMail: oran@cisco.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.