An OAuth-Based Authorization Framework for Access Control in Remote Collaboration Systems

Srikanth Jonnada
SrikanthJonnada@my.unt.edu

Ram Dantu
ram.dantu@unt.edu

Pradhumna Shrestha
pradhumna.shrestha@unt.edu

Ishan Ranasinghe
ishanranasinghearachchilage@my.unt.edu

Logan Widick
loganwidick@my.unt.edu

Department of Computer Science and Engineering, University of North Texas, Denton, TX 76203

***Abstract*— Advanced human computer interaction systems have made it possible for helpers (professionals) to remotely collaborate with workers (individuals seeking assistance from the professionals). For example, a professional can remotely help a worker fix automobiles or electronics, identify how much of what medication to take, or perform household repairs. We have presented a system, Collaborative Appliance for Remote-help (CARE) that allows for such collaborations. Our system allows a skilled professional or other helper to remotely access and control a worker's locally deployed resources over the Internet. These locally deployed resources may include cameras, microphones, speakers, processors, and memory. The remote helper then directs the worker to perform specific tasks to complete the job at hand. Like other Internet of Things (IoT) based systems, CARE is input-constrained. That is, a worker cannot provide input via a touch screen or keyboard.**

The resources are accessed over the Internet. Thus, security and privacy are big concerns. In this paper, we present the authorization and access control framework for the input-constrained CARE system. This framework has been implemented using the OAuth 2.0 Authorization Framework and has been designed to meet the needs of resource owners who have no technical knowledge. We have shown that our proposed framework is very effective and consistent with the access control guidelines set by the National Institute of Standards and Technology (NIST).

Keywords—Access Control, Authorization, Human Computer Interaction, OAuth, Remote Collaboration, Security

I. INTRODUCTION

We depend on many mechanical and electronic devices in our daily lives. These devices may break down, often interfering with our daily routine and quality of life. An expert is usually required to troubleshoot and fix the issues. Unfortunately, due to a lack of skilled workers, consumers often face long wait times and/or expensive service charges.

Video conferencing, online education, and other technologies have made it easier for people to collaborate remotely. These technologies have reduced the time and expense required to complete common objectives. In this paper, we will discuss our proposed novel collaboration system: Collaborative Appliance for Remote-help (CARE) [1].

The rest of the paper is organized as follows: In Section II, we introduce the proposed CARE system, and discuss its importance and applications. We also explain why a security framework is required in such a remote collaboration environment. In Section III, we propose a security framework for access control and authorization in the CARE system. In this way, Section II also provides the context for the proposed framework of Section III. In Section IV, we provide results detailing the efficacy of the proposed security mechanisms. Finally, in Section V we provide our conclusions, and provide a general direction of future work for this research.

II. BACKGROUND ON CARE

The proposed CARE system provides a platform for a helper (the expert) to instruct and support a remote worker (the person seeking help from the expert) to accomplish a task. The system deploys an infrastructure comprised of hardware and software resources that can be controlled over the Internet. These resources help establish common ground [8][10] between the individuals and provide situational awareness [9] to the remote helper. The remote helper monitors and controls the remote infrastructure using real-time audio and video. Figure 1 shows an example of such a deployment. In the next few paragraphs, we will describe the details of this deployed infrastructure.

A high-definition webcam allows the helper to view the worker's environment. The helper can pan and tilt the camera and zoom in and out on objects of interest. In poor lighting conditions, the helper can turn on LED lights to help illuminate the worker's environment. However, the camera may not be able to capture all the visual information required by the helper. To overcome this problem, the components are stationed on a wheeled platform. The wheeled platform and mounted components form an assembly called a CARE device. By remotely controlling the CARE device, the helper can obtain more visual information about the worker's environment and gain better situational awareness [11][12]. To establish a common ground between the remote helper and worker, CARE uses Voice over IP (VoIP) technology for voice communication. The CARE device is equipped with a speaker and a microphone

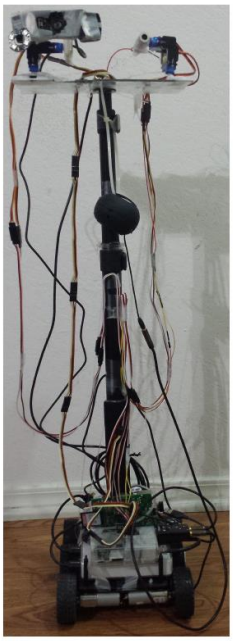

Figure 1: A Sample CARE Device. This device is deployed at the worker's location. A camera provides visuals, and a microphone and speaker provide audio. The device includes a wheeled platform for mobility.

to communicate with the remote helper. In addition to oral communication, CARE uses gestures to exchange information. For example, the CARE device has a laser diode that the remote helper can use to point at objects of interest. The remote gestures allow the helper to provide instruction more easily and precisely. A remote helper can control these resources mounted on the CARE device to obtain any required information about a worker's environment and provide necessary instructions.

There are plentiful protocols that can be used in conjunction with CARE. For example, not every homeowner knows how to do home maintenance and repair tasks. Hiring an expert for on-site inspection and repairs can be expensive even for minor tasks. On average, a homeowner in the United State spent $3,100 on home maintenance in the year 2014, and six times more than that on home improvement [2]. Using CARE, the remote helper can obtain any required information about the worker's environment and provide step-by-step instructions to the house owner.

One of the common home repair protocols that many homeowners must face is fixing a leaky faucet. This involves multiple steps, and unskilled workers may damage the faucets, causing more water to leak. To repair a leaky faucet successfully, the worker must have the required plumbing knowledge. For example, the worker must be able to shut off applicable water valves before starting the repair, react to unexpected events during the repair, and know how to use the tools required for the task. Faucets contain delicate parts which should be handled carefully. In the process of fixing the leaky faucet, the worker should identify the problem, take a sequence of steps for removing the parts of the faucet, identify the parts to be replaced, and replace the necessary parts. Using CARE, the

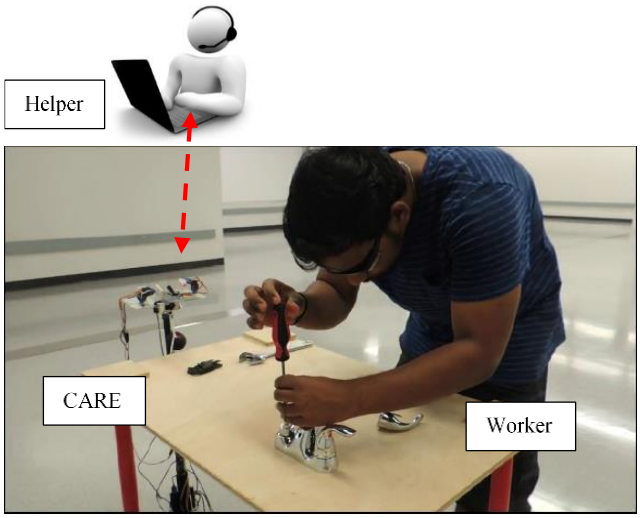

Figure 2: View of the worker's environment. The woker is unscrewing a screw using a screw driver with instructions from the remote helper.

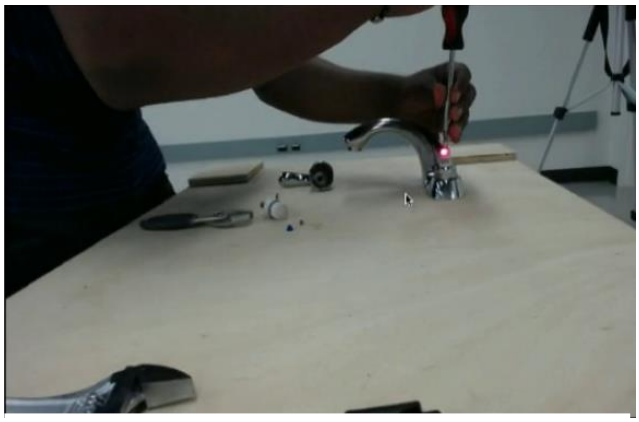

Figure 3: View of the worker's environment through CARE from the remote helper's perspective.

remote helper obtains any required information about the worker's environment, and provides step-by-step instructions to the worker.

Figure 2 shows a view of the worker's environment in which he is fixing the faucet. Figure 3 shows the view of the same area from the remote helper's perspective through CARE. It should be noted that the remote helper can control the infrastructure at the work site and hence change the view as per requirements.

We have conducted 33 experiments on the faucet repair protocol using CARE. All subjects could repair a leaky faucet with assistance from a remote helper. Both male and female subjects with different skill levels participated in this experiment. The participants included 25 male subjects and 8 female subjects. At the time of the experiments, all subjects were between 21-30 years old. Only 6 of the male subjects had prior knowledge of faucet repair; all other participants had no prior knowledge regarding this task. The participants took between 404 to 865 seconds to complete the task, depending on prior

knowledge and skill level. According to participant feedback, it was easier to complete the task with the help of the CARE system than with video tutorials.

CARE can be used with many other protocols, including inspecting homes, repairing garage doors, and replacing light switches. CARE can also help with vehicle inspection and repair tasks, such as finding oil leaks, inspecting vehicle exhaust systems, and changing tires. In addition, CARE can help the elderly perform day-to-day tasks, such as finding the right medication to take, preparing the appropriate dose of said medication, or controlling the TV and other devices in the home.

CARE has many decisive advantages over similar robotic applications available today. Unlike most of these solutions, because the remote helper is a human being, CARE can be used to perform new and unscripted tasks in unknown and changing environments. Furthermore, the remote helper can guide the worker according to the worker's skill level.

While CARE does introduce a novel paradigm in remote collaboration, it is nonetheless open to security attacks. In order to assist the worker, the helper needs to remotely control and use the worker's locally deployed hardware and software resources. By accessing these resources, the helper can effectively acquire information about the worker's environment and provide necessary instructions and support. While authorized access of these resources is imperative and hugely beneficial, unauthorized access of the same resources can cause major security concerns. For example, an unauthorized helper could use the CARE device to eavesdrop. The use of the internet for remote support opens further avenues for such attacks. Even an authenticated helper may be able to access resources that are not relevant to the task at hand. This threatens the worker's privacy. Therefore, both authentication and authorization are important for providing security in such systems.

Common access control mechanisms such as Access Control Lists (ACLs) [13], Role-Based Access Control (RBAC) [14], and Attribute-Based Access Control (ABAC) [15] work effectively in centralized environments where users have trusted identities and predefined policies. These limitations prevent the above mechanisms from being used in CARE and other distributed Internet of Things (IoT) environments that require dynamic authorization. Other authors [16][17] have developed different access control mechanisms which also fail to support the dynamic, time-constrained, and resource-constrained access control based on resource owner consent that CARE requires.

In 2012, Hardt developed the OAuth 2.0 Authorization Framework [3]. The OAuth framework is designed to enable a third-party application to obtain limited access to restricted resources on behalf of the resource owner. In a traditional client-server scenario, the client authenticates with the server by using the resource owner's credentials to gain access to a protected resource. If the resource owner wants to give third party applications access to restricted resources, the resource owner needs to share credentials with the third-party, which creates privacy and security issues. Conversely, in the OAuth framework, the third-party does not use the resource owner's credentials to gain access. Instead, the third-party obtains an access token with a specific scope and lifetime. These tokens are issued to third-party clients by an authorization server with the approval of the resource owner. This authorization framework satisfies the requirements of resource authorization for CARE.

The OAuth framework has been designed for input-enabled web-based systems in which resource owners can provide input through devices such as touchscreens and keyboards. This framework cannot be utilized as-is for input-constrained IoT devices. The authors of [4] proposed a model based on OAuth for access control of IoT devices, and based their architecture on four key requirements. First, the authorization policies should be application-scoped. Second, the access control system should not depend on the specifics of the client devices. Third, the authorization policies should be easy to administer. Fourth, the authorization operations should be offloaded to external authorization servers to reduce the burden on IoT devices which are constrained by processing capabilities. However, under this model, devices must pre-register under the scope of a service prior to obtaining authorization, effectively preventing the dynamic authorization required for CARE. The authors of [5] propose an architecture that provides a trust relationship between a resource server and clients, and synchronizes this relationship to the authentication server. If a trust relationship between a client and resource server is pre-established, the authorization server provides an access token to the client. However, to apply this architecture, a strong trust relationship between the client, resource server, and resource owner is necessary. This is not possible in most OAuth implementations.

III. The Proposed Authorization Framework

In this section, we describe a new access control and authorization model based on OAuth to meet the requirements of our CARE system. According to our model, the worker is the OAuth resource owner and the helper is the OAuth client. Our model allows workers to grant helpers limited-time access to the resources required to perform a task and control the scope of access granted. In addition, only authorized helpers can receive access tokens which helps protect workers from malicious activity. Our framework is designed to meet the requirements of even technically naïve workers.

The authorization framework for CARE is shown in Figure 4. As marked in the figure, the framework can be divided into the following event flows:

(1). RequestAuthorizationGrant: A resource owner (worker) requests an authorization grant from the authorization server. The resource owner can provide this grant to the client (helper) as proof of the resource owner's permission to access the required resources.

(2). AuthorizationGrant: The authorization server authenticates the resource owner and provides an authorization code.

(3). SubmitAuthorizationGrant: The resource owner provides the authorization grant to the client through a direct communication channel

(4). RequestAccessToken: The client authenticates with the authorization server, provides the authorization grant, and requests an access token. The client provides the scope and time required while requesting an access token.

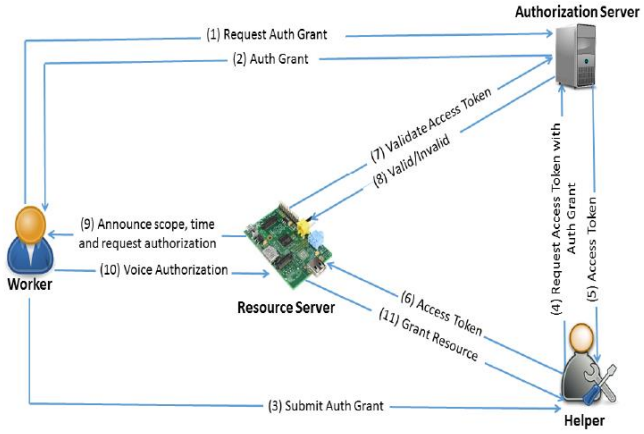

Figure 4: The Proposed Access Control and Authorization Framework for CARE. In this OAuth-inspired framework, the worker is the OAuth resource owner and the helper is the OAuth client.

(5). GetAccessToken: After successfully authenticating the client and validating the authorization grant, the authorization server provides access token to the client.
(6). ProvideAccessToken: The client provides the access token to the resource server to gain access to the requested resources.
(7). ValidateAccessToken: The resource server now validates the access token with the authorization server. The resource server provides client and its credentials along with access token for validation.
(8). Valid or Invalid: The authorization server validates the access token and responds to the resource server.
(9). RequestAuthorization: The resource server announces the requested resources, time, and scope to the resource owner for approval.
(10). VoiceAuthorization: The resource owner verbally approves or declines each access request announced by the resource server. If desired, the resource owner may change the request parameters.
(11). GrantResources: The resource server provides the client access to the approved resources.

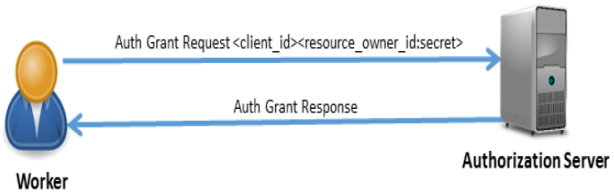

Figure 5: Worker (resource owner) obtaining Authorization Grant from Authorization Server

In the following subsections, we will describe the details of each of the above steps.

A. *Obtaining Grant Authorization*

Figure 5 shows how a resource owner (worker) can get an authorization grant from the authorization sever. The worker first registers with the server and provides an ID and secret to the server. This information is used to authenticate the worker with the server. The worker also provides the helper's client ID. Next, the authorization server authenticates the worker and generates an authorization grant associated with the client ID.

B. *Obtaining Access Token*

Figure 6 shows how the client (helper) obtains an access token from the authorization server. First, the helper

Figure 6: Client (helper) obtaining Access Token from Authorization Server

authenticates using a client ID and secret. The client also provides the authorization code, grant type, scope and time. Next, the authorization server authenticates the client and the authorization grant. In response, the authorization server provides a JavaScript Object Notation (JSON) [6] object that includes the access token, scope, and expiration time. This JSON object also includes a refresh token.

C. *Validating Access Token*

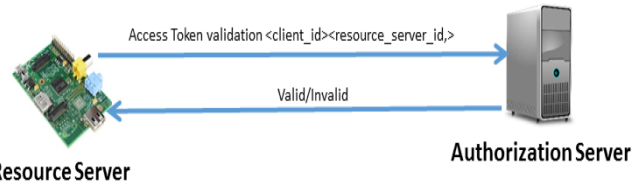

Figure 7: Resource Server Validating Access Token with Authorization Server

The resource server needs to validate the access token it received from the client (helper) with the authorization server. This process is shown in Figure 7. First, the resource server gives its server ID, access token, and the corresponding client ID to the authorization server. Next, the authorization server notifies the resource server if the token is valid or invalid.

D. *Voice Authorization*

After successfully validating the access token, the resource server announces the requested scope and duration to the resource owner (worker). The resource owner provides approval for each resource separately. The resource owner can also propose an alternate duration through voice commands. Within the resource server, this alternate duration overrides the duration from the access token. If the resource server does not get a response from the resource owner within a set period, the resource server times out and automatically declines access to the resources. Figure 8 shows the voice authorization workflow.

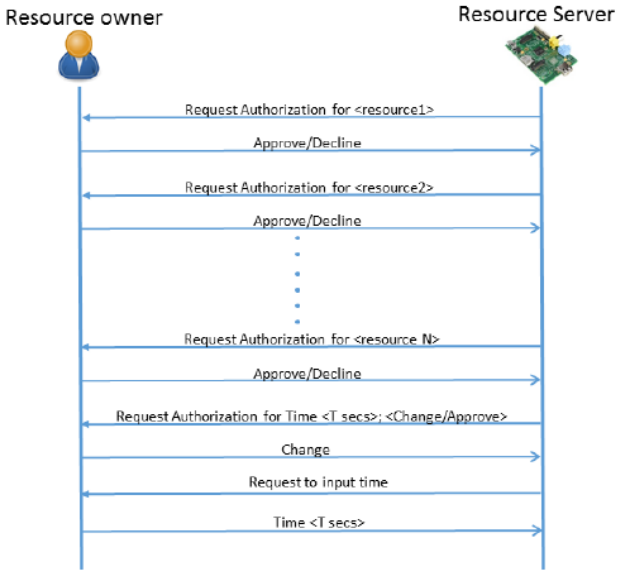

Figure 8: Voice Authorization Protocol between Resource Server and Resource Owner (worker)

E. Renewing Access Token

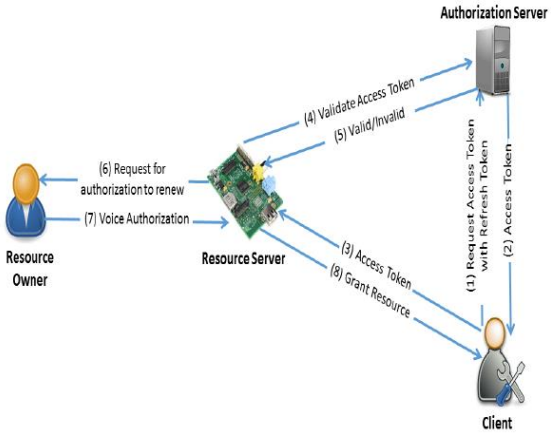

Figure 9: Client (helper) renewing Access Token using Refresh Token. The resource owner (worker) has to approve the renewal.

If the access token or approved access time expires, the client (helper) can use the refresh token (provided with the initial access token) to renew the access token. The information flow is shown in Figure 9. The rest of the renewal process remains the same as the process required to obtain the initial access token. However, the resource owner does not have to give a new authorization grant to the client (Steps 1, 2 and 3).

F. Resource Owner Revoking Access Token

If the resource owner (worker) suspects that a client (helper) is performing an unacceptable activity, the resource owner can block the client by revoking the access token and refresh token. The information flow is shown in Figure 10.

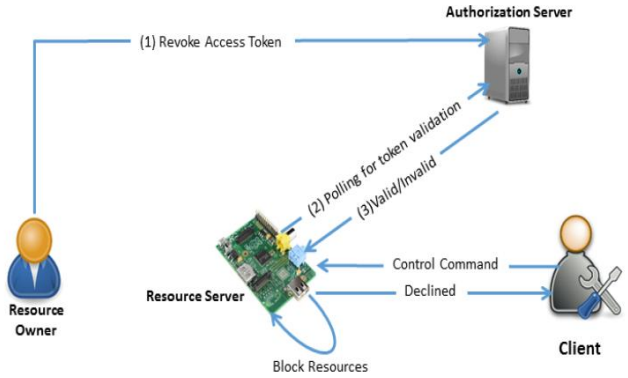

Figure 10: Resource Owner (Worker) Revoking Access Token. After the token is revoked, the client (helper) cannot continue accessing the resources.

After access is approved, the resource server keeps asking the authorization server for the status of the access token. When a resource server asks for the status of a revoked access token, the authorization server will issue a negative response. After receiving this negative response, the resource server immediately blocks the client.

G. Resource Server Revoking Access Token

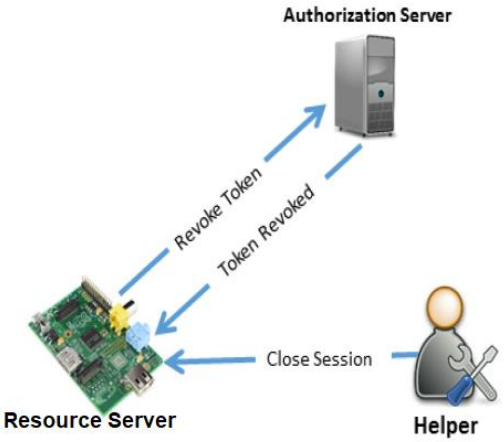

Figure 11: Access Token is Revoked When the Helper (Resource Owner) Ends The Session

As shown in Figure 11, when the access time approved by the resource owner (worker) expires or the client ends the session, the resource server asks the authorization server to revoke the access token.

IV. RESULTS

In 2012, the National Institute of Standards and Technology (NIST) published guidelines for evaluating access control systems [7]. This document discusses quality metrics to verify the administration, enforcement, performance, and support properties of access control mechanisms. The proposed authorization framework complies with the NIST metrics summarized in Table I.

TABLE I: EVALUATION OF METRICS DISCUSSED IN NIST GUIDELINES FOR ACCESS CONTROL (AC) SYSTEMS [7]*

Metric	*Evaluation*
Ease of privilege assignments	
How many steps are required for assigning a privilege?	This AC system involves 11 steps for granting resources to a user, shown in Figure 4
How many steps are required for removing a privilege?	This AC system involves three steps for a worker to revoke access to resources, shown in Figure 10 .
Flexibilities of conguration into existing systems	Access Control is enforced by an application on the CARE device (client) that interacts with Authorization Server (server) through HTTP. This implements a client/server communication protocol.
Horizontal scope (across platforms and applications) of control	AC system can authorize multiple users for a single host and multiple users for multiple hosts via a network.
Vertical scope (between application, DBMS, and OS) of control	The scope of access control includes applications, files, hardware resources and network devices.
Least privilege principle support	This AC system enforces least privilege principle. Final approval of authorization for resources is associated with a human through voice authorization, shown in Figure 4. This AC system also enforces least time principle.
Safety (connements and constraints)	This AC system prevents unauthorized access to resources and relaying permissions to unauthorized users.
Operational/Situational awareness	This AC system is capable of operations/situational awareness control as the final decision of access is controlled by a human through voice authorization, shown shown in Figures 4 and 8 .
Granularity of control	This AC system allows configuring the granularity of controlled objects. Not just the objects, access to their features are also controlled in this system.

*Table continued in the next column as Table II

The results in this section were obtained using a Raspberry Pi board with a 900MHz quad-core ARM Cortex-A7 CPU and 1GB RAM.

A. Other Metrics

We also measured the following:

(1) Resource Utilization

TABLE II: CONITNUATION OF TABLE I

Metric	*Evaluation*
Response Time	
Does the response time of granting an access request met the organization's requirement?	It takes 584 milliseconds from a helper submitting an authorization grant to get an access token, a resource server validating the access token. Rest of the workow is dependent on the pace at which the worker provides the authorization grant to the helper and he provides voice authorization.
Integrated with authentication function	This AC system can be integrated with identity providers for authenticating users.
OS compatibility	This AC system is independent of Operating System of the device.
User interfaces and API	This AC system provides a GUI for AC policy management and authoring

- CPU utilization 4.15% on average
- Memory Utilization 9.6%

(2) Number of interactions handled by each network elements (to analyze the load on each element):

- The resource server handles four HTTP interactions during the complete authorization.
- The authorization server handles six HTTP interactions during the complete authorization.
- The client application handles four HTTP interactions during the complete authorization.

V. CONCLUSION AND FUTURE WORK

Advanced technologies have made it possible to not only remotely manage and control electronic and mechanical devices, but also enable effective remote interaction between humans. We proposed a system named Collaborative Appliance for Remote-help (CARE) that allows remotely located helpers (experts) to assist workers (people seeking help from the experts) in various tasks in the workers' daily lives. For example, the CARE system can help workers fix cars and perform home repairs. This system is expected to address the shortage of skilled professionals in these and other areas. The CARE system will also decrease the costs of having professional services at the job site. A worker deploys a CARE device at the job site, which can then be remotely controlled by an expert to help complete a task. The CARE system is input-constrained and can be used by individuals with little or no technical knowledge.

The remote helper is required to access and control the worker's locally deployed resources using the Internet, so security and privacy are significant concerns. Access control and authorization are critical. Furthermore, even an authenticated helper should not be able to access local resources that are not relevant to the job.

In this paper, we used the OAuth 2.0 Authorization Framework to implement the access control and authorization features of CARE. Using OAuth, we provided access to the resources using time-limited access tokens. These tokens were implemented using a resource server and an authorization server. The framework can be summarized as follows.

1. The resource owner (worker) registers the client at the authorization server as a consent to provide access to the resources.
2. The helper requests access tokens from the authorization server.
3. The helper provides these tokens to the resource server.
4. The resource server validates the token with the authorization servers.
5. The resource server provides access to the helper once token validation is complete.
6. The resource server denies access to the helper as soon as the access token expires or is revoked.

We have found this framework to be very effective in implementing access control and authorization methodologies as demonstrated by our system's compliance with the NIST guidelines. We recommend using similar security design principles for access control of remote resources in other advanced HCI systems as well.

The presented access control and authentication framework in this paper has been modified from the originally proposed model in [3]. In our framework, after the client presents an access token to the resource server, the resource server asks the resource owner to verbally approve the request as shown in Figure 4. This is unique to CARE which requires real-time interaction between the resource owner and the remote helper. Other modifications made to the standard OAuth 2.0 protocol are as follows:

- The resource owner (worker) requests an authorization grant from the authorization server before the client (helper) can request an access token.
- The resource server verifies the access token with the authorization server before resources can be released to the client.

As future work, we intend to present a formal security analysis of the proposed model since it differs substantially from OAuth.

We also plan to publish a novel methodology for analysis and evaluation of the complexity of the collaboration protocols for CARE, and further applications of CARE in modern IoT and Cyber-Physical systems.

ACKNOWLEDGMENTS

This research was partially funded by the National Science Foundation (NSF) grants CNS1229700 and IIS1545599.

REFERENCES

[1] S. Jonnada, Analysis And Performance Of A Cyber Human System And Protocols For Geographically Separated Collaborators, Ph.D. Dissertation, University of North Texas, August 2017.

[2] HomeAdvisor, True cost guide report for home maintanance services, https://www.homeadvisor.com/r/wp-content/uploads/2015/04/2015-cost-report.pdf, (Accessed on 01/19/2018)

[3] D. Hardt, The OAuth 2.0 Authorization Framework, October 2012.

[4] F. Fernandez, A. Alonso, L. Marco, J. Salvachua, A Model To Enable Application-Scoped Access Control As A Service For IoT Using OAuth 2.0, 20th Conference on Innovations in Clouds, Internet and Networks (ICIN), 2017, pp. 322-324.

[5] M. Noureddine, R. Bashroush, A Provisioning Model Towards OAuth 2.0 Performance Optimization, 10th International Conference on Cybernetic Intelligent Systems (CIS), 2011, pp. 76-80.

[6] T. Bray, The JavaScript Object Notation (JSON) Data Interchange Format, December 2017

[7] V. C. Hu, K. Scarfone, Guidelines for Access Control System Evaluation Metrics, National Institute of Standards and Technology, NISTIR 7874, September 2012.

[8] Herbert H Clark, Susan E Brennan, et al., Grounding in communication, Perspectives on socially shared cognition 13 (1991), no. 1991, 127-149.

[9] Mica R Endsley, Toward a theory of situation awareness in dynamic systems, Human factors 37 (1995), no. 1, 32-64.

[10] David R Traum, A computational theory of grounding in natural language conversation.,Tech. report, ROCHESTER UNIV NY DEPT OF COMPUTER SCIENCE, 1994.

[11] Robert E Kraut, Susan R Fussell, and Jane Siegel, Visual information as a conversational resource in collaborative physical tasks, Human-computer interaction 18 (2003), no. 1,13-49

[12] Darren Gergle, Robert E Kraut, and Susan R Fussell, Using visual information for grounding and awareness in collaborative tasks, Human{Computer Interaction 28 (2013), no. 1,1-39.

[13] Hakki C Cankaya, Access control lists, Encyclopedia of Cryptography and Security,Springer, 2011, pp. 9-12.

[14] Ravi S Sandhu, Edward J Coyne, Hal L Feinstein, and Charles E Youman, Role-based access control models, Computer 29 (1996), no. 2, 38-47.

[15] Vincent C Hu, D Richard Kuhn, and David F Ferraiolo, Attribute-based access control,Computer 48 (2015), no. 2, 85-88.

[16] Ludwig Seitz, Goran Selander, and Christian Gehrmann, Authorization framework for the internet-of-things, World of Wireless, Mobile and Multimedia Networks (WoWMoM),2013 IEEE 14th International Symposium and Workshops on a, IEEE, 2013, pp. 1-6.

[17] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi, A capability-based security approach to manage access control in the internet of things, Mathematical and Computer Modelling 58 (2013), no. 5, 1189-1205.

[18] Rui Zhang, Yanchao Zhang, and Kui Ren, Distributed privacy-preserving access control in sensor networks, IEEE Transactions on Parallel and Distributed Systems 23 (2012), no. 8, 1427-1438