Capability Exchange: Improving Access Control Usability in Health IT

Chen Qin, Emily Freebairn, Sean Smith
Department of Computer Science
Dartmouth College

July 18, 2013

Abstract

Clinicians report usability problems in modern health IT systems in part because the strictness of computerization eliminates the layer of informality which previously enabled them to get their jobs done. In this paper, we examine a solution by considering the strictly-enforced medical order as a security *capability*, and then using *capability exchange* to authorize frustrated end-users to re-introduce the necessarily flexibility. We prototype our idea using OpenEMR and Belay, and show how this prototype can address access control usability problems reported by clinicians.

1 Introduction

A common theme among research about large-scale business security systems, including health care systems, is that access control does not hold up under the complexities of real world usage (e.g., [1, 7, 8, 10, 18, 19, 21, 20]). There have been many different attempts to understand where access control fails and how it could be augmented or redesigned in order to address these issues. Grandison and Davis argue that current security models are too broad and do not address the challenges and needs specific to each industry, and provide a set of general guidelines they feel all security systems must address for each particular industry [8]. In particular, they argue that each industry has a *prime directive* that a security model must accommodate in order to limit user frustration and workarounds of the system. For health care, they observe that extraordinary circumstances are commonplace and recommend that any security system must accommodate accordingly, by allowing exceptions and executing checks if a user has no credentials, rather than refusing access or undergoing a lengthy recovery process to retrieve or create credentials.

Alam et al. [1] are more specific and describe particular problem areas in health IT which basic access control does not address, including *dynamic access control*, *delegation of rights* (under certain circumstances, a medical practitioner must be able to delegate their authority to accomplish a particular task), and *break glass policy* (in healthcare emergencies, a medical practitioner must be able to help a patient quickly and efficiently, regardless of whether they ordinarily have an high enough level of authority).

In all these situations, the end-user clinician may face real-world exigencies requiring he or she to get more access than a strict health IT system provides—Figure 1 sketches this problem. In previous work [25], we looked at some potential psychological causes; in this work, we examine a potential technological solution.

This Paper Section 2 presents two specific motivating examples drawn from real clinicians. Section 3 presents the approach we took to the problem. Section 4 discusses our experimental prototype, in OpenEMR and Belay. Section 5 shows, in screenshots, how this prototype can address the motivating examples. Section 6 discusses next steps and concludes.

2 Motivating Examples

We now consider two specific usability problems reported by clinicians. (In concurrent work, we have been preparing a systematic taxonomy [22].) To clarify discussion, we adopt the terms "physician" and "nurse" for the two clinician roles involved.

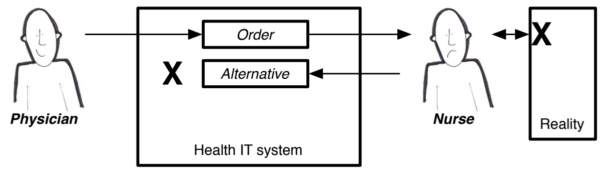


Figure 1: In many of the health IT usability problems we saw, the system embodied an order which the end user was not able to carry out—but the system did not allow the user to perform the natural alternative which had worked in the pre-electronic world.

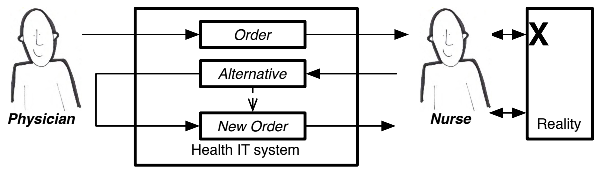


Figure 2: On a high level, our proposed solution enables the end user to exchange (under appropriate authorization) the initial order for an alternative—and remembers a template to offer that option automatically in the future.

Medication Dose Size Koppel et al examined a health IT system which enforced access control for medication by requiring end-user clinicians to scan a barcode on the drug and on the patient, and only permitting the action if the systems' internal records show they matched [13]. Unfortunately, the health IT system did not recognize that two 5mg tablets were equivalent to one 10mg tablet. Consequently, a clinician with an order to give a 10mg dose to patient, but who only had 5mg tablets, either had to deny the patient her medication—or had to operate outside system authorization.

Medication Timeshift Private communication [12] reveals a different medication "access control" problem introduced when a medical enterprise moved to a stricter health IT system. In this case, a physician may often order a medication regimen for some specific number of hours starting at a specific time. (For purposes of this discussion, let's say 5 hours, starting at 5pm.) Unfortunately, the patient may not necessarily be present at the starting time—e.g., the patient may "still be in dialysis" and not show up until an hour later. In the more forgiving older system, the nurse could still give the full regimen, from 6pm to 11pm. However, the new system only permitted the nurse to give the medication until 10pm. Consequently, the nurse either had to deny the patient the tail end of the regimen—or had to operate outside authorization.

3 Our Approach

For such usability challenges, the root cause may arguably lie in the fact that the health IT system did not know "when to look the other way" [6]. However, in these cases, the end-user clinician motivated to circumvent the strict "letter" of the access laws may arguably feel that all relevant stakeholders would permit this circumvention, if asked—the medical equivalent of "that classic bit of Italian clerical casuistry: *if the pope were here, he would understand.*" [2].

In some sense, we can look at this in the spirit of Alam's concerns about dynamism, delegation, and break-glass. The end-user dynamically needs more or different permissions from what the system has granted them. In the traditional security view, *delegation* is when user has some rights, and wants to pass them on to a different user—possibly first modifying them by restricting them somehow. In these problematic health IT settings, the user wants to pass rights on to herself or himself, first modifying them to account for the real-world scenario not foreseen by the original granter of the permission. Such end-user extension cannot be blindly allowed always, of course. For example we are aware [22] of a scenario where clinicians would tell the health IT system that a 1L IV bag in fact only contained 0.85L, so that the "bag is nearly empty" alarm would ring earlier—the patient thus received better care from the end-user (no empty bag), but possibly worse care overall if the system is tightly coupled and is recording the patient is receiving 15% less medication than he really is.

Figure 2 sketches our proposed solution: a way for the end-user to do this modified self-delegation to themselves, under appropriate authorization. To carry this out, we look to the technology of *capabilities*. A capability is a security concept that has been around for a while (since 1966 at least [5]), but has also been receiving more recent attention in securing systems. For example, the Capsicum system [26] uses capabilities to provide fine-grained internal protections within UNIX applications; Belay [15] uses capabilities to provide authorization across multiple browsers and servers. Basically, a capability is like a car key—it allows whoever has it to do

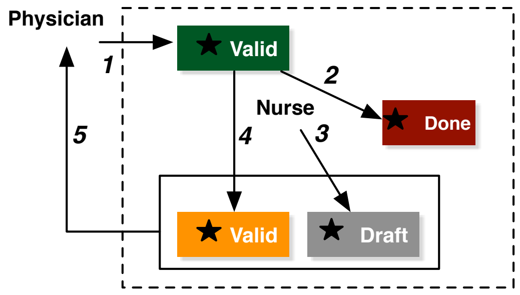


Figure 3: (1) The physician creates a capability and assigns it to the nurse. (2) The nurse can carry out the order, consuming the capability. (3) Alternatively, the nurse can draft one or more proposed capabilities, and (4) propose exchanging the initial capability for these alternatives. (5) This proposal goes back to the physician for approval.

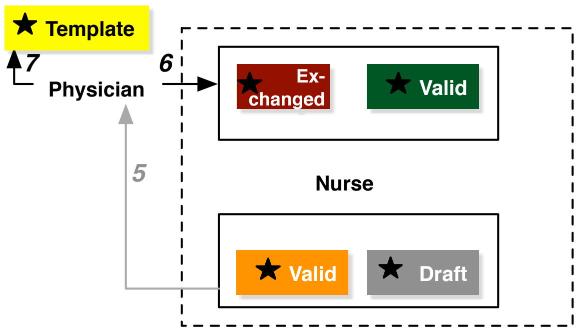


Figure 4: (6) If the physician approves the exchange, the initial capability is revoked and the alternatives become blessed. (7) The system also gives the clinician the option of pre-approving such exchanged in the future.

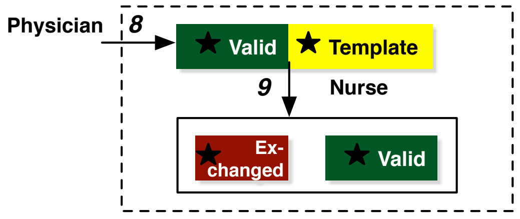


Figure 5: (8) If the physician issues such an order again, she can include templates for pre-approved exchanges. (9) The system gives the nurse the option to automatically carry out that exchange.

particular actions (like driving) to a particular object (like a car). The subject carries the right with them, rather than having the object keep track of authorized subjects.

To allow the necessarily functionality, we designed an approach where health IT system explicitly represents the permission to do the action as a capability—but then also provides a *capability exchange*, permitting a user to trade in some permissions for other ones, initially with explicit approval, but eventually automatically (to reduce problems when the approver might not be available). Figure 3 through Figure 5 sketch the lifecycle we envisioned.

4 Experimental Prototype

To demonstrate the practicality of ideas, we built a prototype (*BelayEMR*) integrating an open-source capability system with an open-source EMR, and then modified them to add capability exchange features for our motivating examples.

4.1 OpenEMR

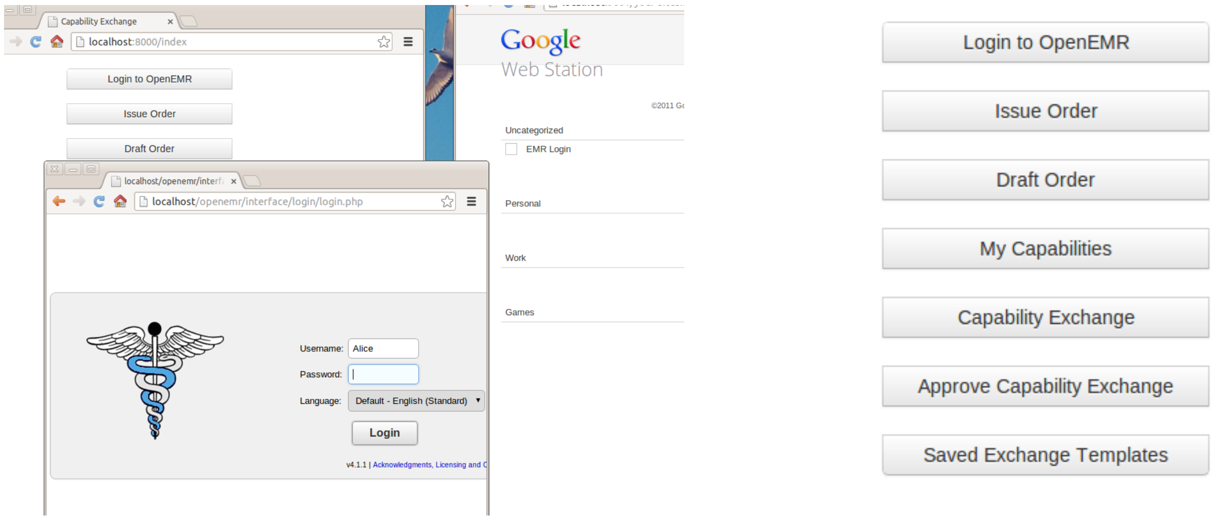
For an open-source EMR, we explored VistA [24] and OpenMRS [17], but ending up settling on OpenEMR [16], due to the ease of installing and modifying it, the realistic look-and-feel (as judged by medical colleagues), and the fact that it actually has a footprint in the US medical system. (E.g., one source estimates that in the US alone, over "30 million patients are being treated at healthcare facilities running OpenEMR" [9].)

4.2 Belay

Google's Belay project [15] is an implementation of the Belay Cloud Access Protocol (BCAP), a capabilities-based security protocol that seeks to support web and cloud based computing contexts for asynchronous web applications.

Belay was inspired by work done in four primary areas, and Belay developers have incorporated these ideas into their work.

Usability vs. security on the web It is a common belief that security and usability are at odds with each other, and that designing for one automatically puts the other at a disadvantage. However, in the last ten years there has been an effort

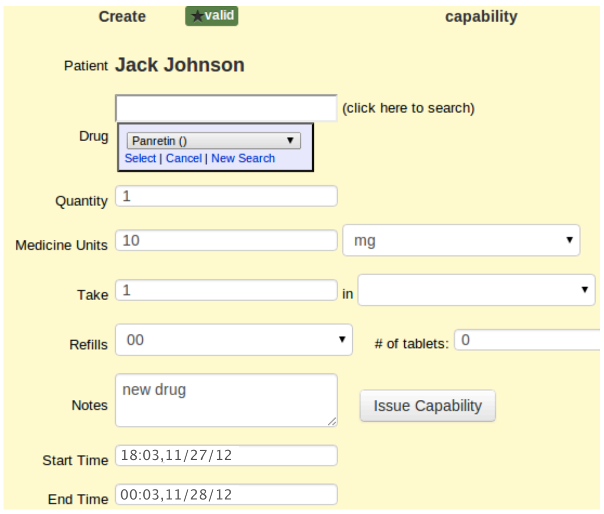


Username: Alice
Password:
Language: Default - English (Standard)
Login
v4.1.1 | Acknowledgments, Licensing and C

Figure 6: (Left) Screenshot of the login window for our Belay/EMR prototype; (right) the menu of capability operations offered

to disprove this notion [11, 28, 27]. Belay continues with this goal in mind for web development, and also includes some of the suggestions made by Ye about cookie management [28] with additional modifications; in fact, Belay replaces cookies entirely with *BCAP* capabilities.

Maintaining session continuity in the interactive web Websites are becoming more and more interactive, and there are a growing number of web apps available to the public. Since the web is both stateless and asynchronous, these web applications



Patient	Jack Johnson
	(click here to search)
Drug	Panretin ()
Quantity	1
Medicine Units	10 mg
Take	1 in
Refills	00 # of tablets: 0
Notes	new drug
Start Time	18:03,11/27/12
End Time	00:03,11/28/12

Figure 7: Physician Alice issues a medication order, in our Belay/EMR prototype

require a means of saving state (or context) either by saving it on the server or, frequently, sending the context back to the client as a cookie. Cookies are often used as a means of both authentication and authorization.

Belay identifies three major problems with cookies:

1. Cookies are passed to the server as part of every request from the client, which makes cross-site request forgery (XSRF) possible.

2. Cookies represent all the state and authority of the user in one token.

3. The state saved by the cookie cannot be used outside the browser.

Both Waterken and the PLT-scheme server are earlier projects that have explored *web-keys* as alternatives to cookie usage [4, 14], inspiring Belay to do the same. A web-key is a large randomly generated string used as a URL. The idea is that this key provides unique and specialized access to a certain resource, so whoever knows this URL can access the resource. If someone loses their key, they can no longer access the resource, and similarly, if someone malicious is able to find the key, they have gained access. However, a web-key is tied to a very specific resource; thus an attacker who is somehow able to gain access to a web-key has both limited authority and resources to work with. A web-key is similar to a cookie in that both are large random numbers that are used by the server to map to specific data. However, a web-key has a few advantages over a cookie,

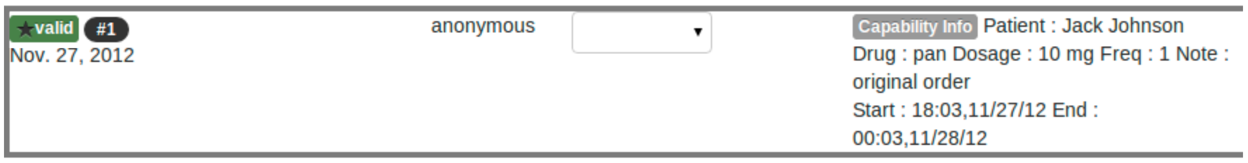

★valid #1
Nov. 27, 2012
anonymous
Capability Info Patient : Jack Johnson
Drug : pan Dosage : 10 mg Freq : 1 Note : original order
Start : 18:03,11/27/12 End : 00:03,11/28/12

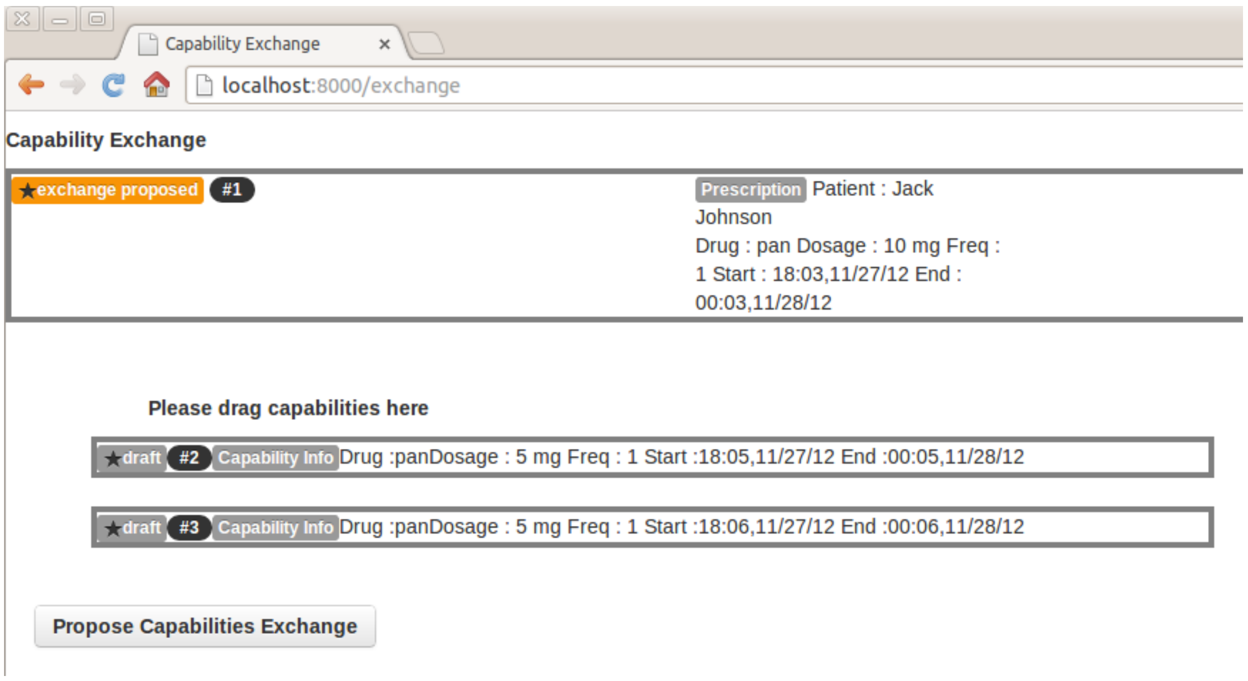
Figure 8: Nurse Bob sees the order capability, in our Belay/EMR prototype

since a web-key has only a portion of the authority of the user, and can only access certain data and do one specific action.

Delegated vs. Federated Authentication Currently, a user has multiple accounts on the internet, with one account for each website that offers personalized service. Each account consists of a username and password that authenticates the user and authorizes that user to do specific things on their account. As more websites have become interactive and want to offer personalized services to each client (thereby requiring an account), the number of accounts a single person has to keep track of has increased, and rather than coming up with a unique password for each account, most people choose the same account name and password. This decreases the security strength of all the user's accounts to the security strength of the website with the weakest security measures.

Some website developers have become aware of this issue, and rather than checking authentication themselves, they delegate authentication to a more well-known website. Federated authentication takes this further and decentralizes delegated authentication, so that any identity provider can authenticate for any relying party. OpenID, an open standard that Google has implemented since 2008 [3], is one such example of federated authentication, and Belay makes use of this as part of the process of simplifying and centralizing security on the web for both common users and web developers.

Cross-App Authorization As web apps have become more popular, developers have begun doing "mashups" of current applications—for instance, a developer might want their web app to use both


Capability Exchange
localhost:8000/exchange

Capability Exchange

★exchange proposed #1
Prescription Patient : Jack Johnson
Drug : pan Dosage : 10 mg Freq : 1 Start : 18:03,11/27/12 End : 00:03,11/28/12

Please drag capabilities here

★draft #2 Capability Info Drug :panDosage : 5 mg Freq : 1 Start :18:05,11/27/12 End :00:05,11/28/12

★draft #3 Capability Info Drug :panDosage : 5 mg Freq : 1 Start :18:06,11/27/12 End :00:06,11/28/12

Figure 9: If Nurse Bob does not have 10mg tablets, he can draft alternative capabilities for two 5mg tablets, and request an exchange.

Google maps and a weather forecasting app from NOAA. However, it has been noted (e.g., [23, 4]) that access control lists cannot ensure secure behavior in interactions between more than two parties. BCAP capabilities offer a more secure alternative for such interactions [15].

4.3 Integration

From a high-level view, what we did is extend OpenEMR to to use Belay authentication, to package Belay capabilities as the vehicle to convey authorization—and then to add the ability for end-users to request *capability exchanges* when the reality of their usage situation does not match the authorization that was given them.

For our work, we used OpenEMR 4.1.0, and the August 2012 trunk branch of Google Belay. (The Autumn 2012 branch introduced a bug we identified and file.) In OpenEMR, we changed its ACL rule setting to populate our test user roles (e.g., physician Alice, nurse Bob, etc).

We changed the login page to allow users to log in with a Belay account. As our motivating examples dealt with drugs, we changed the OpenEMR prescription page to allow the physician role to create and assign capabilities when they give a medication order, and the nurse role to view and act on these capabilities.

In Belay, we added logic to pull out database records from OpenEMR. We added logic to build capability models for our BelayEMR application and integrate with with entities and models in OpenEMR settings. We also added logic to handle capability invocations, retrieval, and meta data lookup (that is, inquiring what this capability is about before retrieve it).

For both the nurse and the physician roles, we added features to support representative cycles of capability exchange. We changed the prescription page to allow the nurse role to create a capability exchange request, and added logic to allow the nurse to create *draft* capabilities (not valid) for exchange request purpose. We added an exchange request view page for for the physician to review capability exchange requests, and then approve or reject requests.

We also added "allow similar requests" in the physician's capability exchange request page, to serve as templates for future capability exchange scenarios (potentially even pre-approved—thus avoiding problems should the approver not be available when the exchange is required), and logic to replace exchange requests capabilities for its substitution draft capabilities. Our initial hypothesis here is that adding templates for the basic scenarios (e.g. "10mg" can be exchanged for two "5mg," for any patient) that users encounter would suffice; however, if too many variations emerge in practice, then we would need to examine ways to create more general templates (each covering more cases) and perhaps to limit template creation/storage to the most common cases.

4.4 Implementation

Figure 15 sketches our basic software architecture.

In Belay, a capability is a URL pointing to a capability agent in or outside of a website. In our case, it lives together with the OpenEMR site Alice and Bob use. The capability agent stores the actions each capability URL can perform During invocation time, an http request comes from a client browser through BCAP, passing parameters to the capability agent. Eventually, after an action gets called, the browser gets a response. These actions and method signature are defined when a new capability URL is created at the capability agent.

In a capability exchange scenario (say, for medication dose sizes), when user Bob find a capability issued by Alice too constraining, he prepares drafts capabilities in OpenEMR. He calls `cap_draft` POST with these new drafts and sends them to the capability agent, which responds with two `cap_medication_draft` capabilities. Bob sends the "exchange request" to Alice through OpenEMR. Alice checks the `cap_medication_draft` with HEAD method to view what these capabilities are about, and then approves them and revokes the initial capability via the agent. The exchange is recorded in database; if Alice opts to create a template, this template is also stored in the database.

We built BelayEMR upon the Python Django framework. The total codebase (excluding Google Belay trunk code, OpenEMR code, and the Django framework) consists about 32K lines of code—although about 28K of that are browser script libraries to support BCAP communication and user interaction.

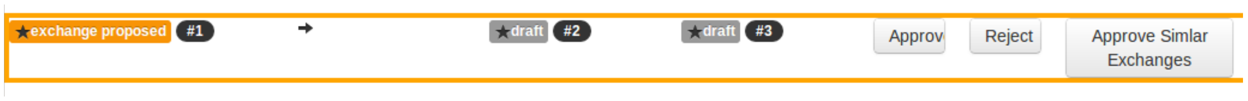
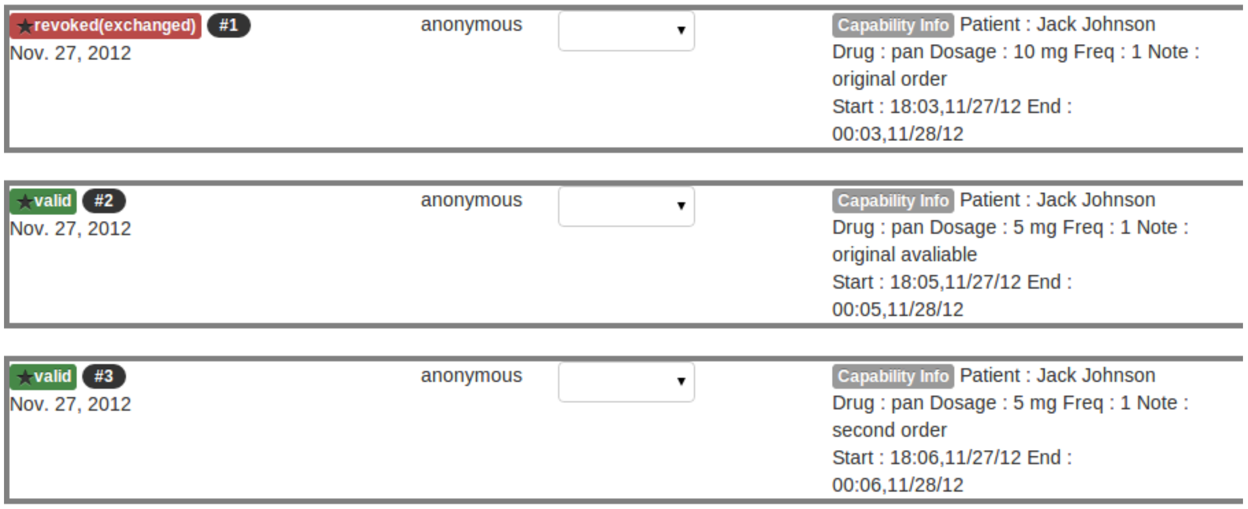


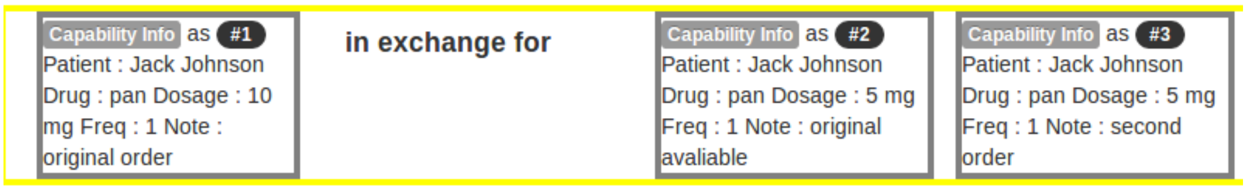
Figure 10: Physician Alice has the option of approving the capability exchange Nurse Bob proposed.


★revoked(exchanged) #1
Nov. 27, 2012
anonymous
Capability Info Patient : Jack Johnson
Drug : pan Dosage : 10 mg Freq : 1 Note : original order
Start : 18:03,11/27/12 End : 00:03,11/28/12

★valid #2
Nov. 27, 2012
anonymous
Capability Info Patient : Jack Johnson
Drug : pan Dosage : 5 mg Freq : 1 Note : original avaliable
Start : 18:05,11/27/12 End : 00:05,11/28/12

★valid #3
Nov. 27, 2012
anonymous
Capability Info Patient : Jack Johnson
Drug : pan Dosage : 5 mg Freq : 1 Note : second order
Start : 18:06,11/27/12 End : 00:06,11/28/12

Figure 11: If physician Alice approves the exchange, then nurse Bob's initial capability (which he could not execute) has been revoked and replaced with his alternatives, which he can.

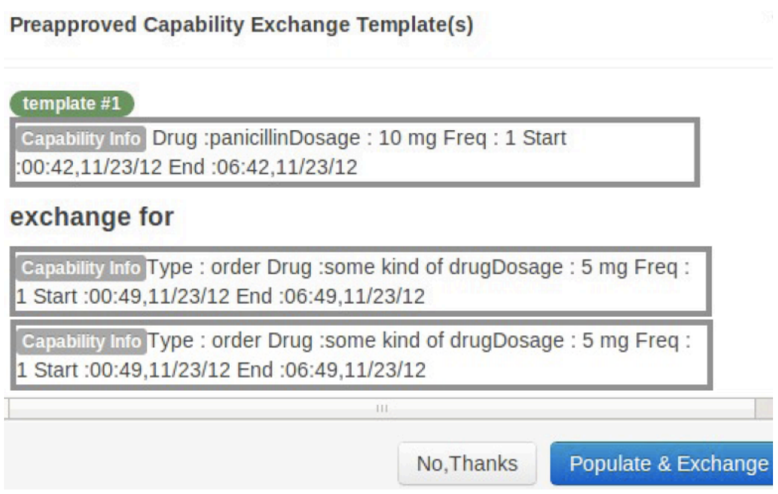

Capability Info as #1
Patient : Jack Johnson
Drug : pan Dosage : 10 mg Freq : 1 Note : original order

in exchange for

Capability Info as #2
Patient : Jack Johnson
Drug : pan Dosage : 5 mg Freq : 1 Note : original avaliable

Capability Info as #3
Patient : Jack Johnson
Drug : pan Dosage : 5 mg Freq : 1 Note : second order

Figure 12: When the system prompts physician Alice to approve or reject the capability exchange that nurse Bob proposed, it also gives her the option to create a template to pre-approve similar exchanges in the future. (See also the right-most button in Figure 10.)



Preapproved Capability Exchange Template(s)

template #1

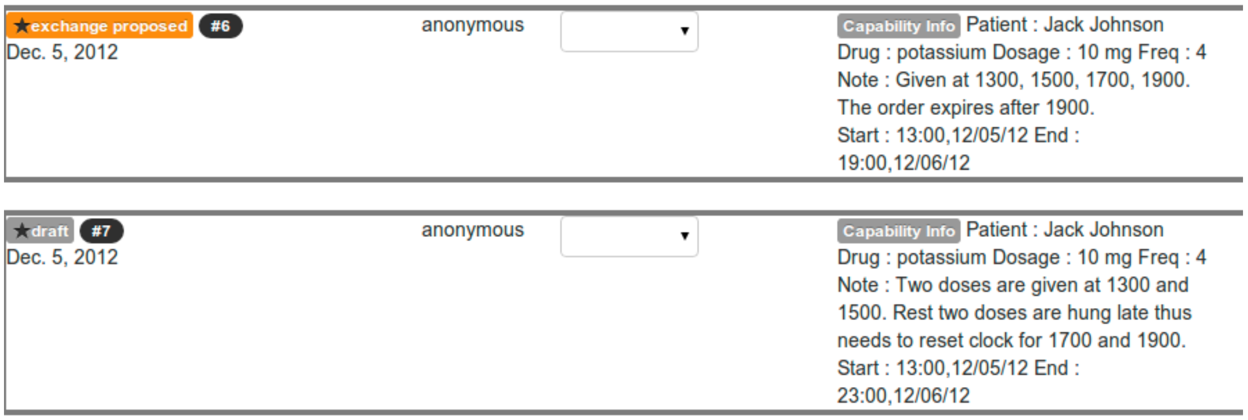
Capability Info Drug :panicillinDosage : 10 mg Freq : 1 Start :00:42,11/23/12 End :06:42,11/23/12

exchange for

Capability Info Type : order Drug :some kind of drugDosage : 5 mg Freq : 1 Start :00:49,11/23/12 End :06:49,11/23/12

Capability Info Type : order Drug :some kind of drugDosage : 5 mg Freq : 1 Start :00:49,11/23/12 End :06:49,11/23/12

Figure 13: If nurse Bob receives a similar order to one he previously needed to exchange and a pre-approved template exists, the system will permit him to directly carry out this exchange (e.g., obviating the actions in Figure 9 an Figure 10 and going straight to Figure 11).



★exchange proposed #6 Dec. 5, 2012	anonymous		Capability Info Patient : Jack Johnson Drug : potassium Dosage : 10 mg Freq : 4 Note : Given at 1300, 1500, 1700, 1900. The order expires after 1900. Start : 13:00,12/05/12 End : 19:00,12/06/12
★draft #7 Dec. 5, 2012	anonymous		Capability Info Patient : Jack Johnson Drug : potassium Dosage : 10 mg Freq : 4 Note : Two doses are given at 1300 and 1500. Rest two doses are hung late thus needs to reset clock for 1700 and 1900. Start : 13:00,12/05/12 End : 23:00,12/06/12

Figure 14: If nurse Bob needs to timeshift the medication order (e.g., because the patient arrived late due to dialysis), he can propose to exchange the impossible capability for a possible one.

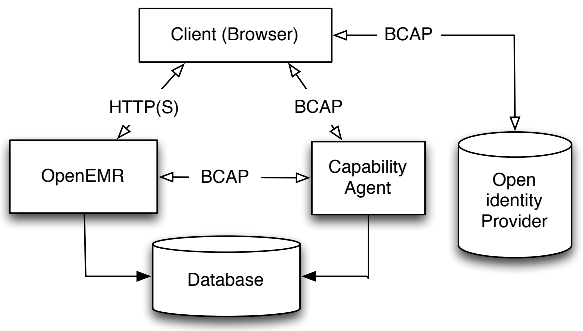


Figure 15: Architecture of BelayEMR

5 Addressing the Examples

We built our prototype to explicitly demonstrate how our notion of capability exchange might help with the problem scenarios in Section 2. Initially, users log in to OpenEMR via Belay and see an additional suite of capability actions for the medication suite (Figure 6).

For the 10mg/5mg case of Section 2, physician Alice now issues a capability when she creates her 10mg order (Figure 7). Nurse Bob sees this capability (Figure 8), but if he doesn't have 10mg pills, the system lets him draft the alternative approach he sees but Alice did not envision and request an exchange (Figure 9). If Alice approves this request (Figure 10), then Bob now has the capabilities he needs (Figure 11). To streamline future interaction, Alice can also instruct the system to pre-approve future requests that are similar (Figure 12, Figure 13).

(Indeed, Koppel et al [13] suggested the problem might be addressed by "new policies for medication administration of different medication forms"; what we are doing is creating a technological path to realize that suggestion.)

BelayEMR can similarly address the timeshift problem of Section 2: nurse Bob drafts a proposed alternative capability embodying the timeshift he knows about but Alice did not envision (Figure 14).

6 Conclusions and Future Work

Many researchers, including our own lab, have lamented the mismatch between the rigidity of strict access control in health IT and the end-user flexibility this domain appears to require. In this project, we have explored a potential solution, to allow end-user flexibility while also preserving security; we have also demonstrated evidence of its potential effectiveness by prototyping it in real-world open-source capability and EMR codebases.

Next steps here include further exploration of the engineering feasibility: that is, more fully incorporating the capability exchange framework throughout OpenEMR and dealing with resulting challenges (such as expressiveness of templates and scalability of keeping track of them). We also need to explore the usability and usefulness: will clinicians be able to easily understand and use these features, and will the features indeed reduce the usability trouble that appears to challenge this space? Naturally, a larger installation would also require a more thorough security analysis—as well as an exploration of the other advantages Belay can bring to the OpenEMR framework.

Nevertheless, we offer this is a first step.

Acknowledgments and Availability

This work is supported in part by the US National Science Foundation's Trustworthy Computing award #0910842 and is part of the Trustworthy Information Systems for Healthcare project sponsored by Dartmouth College's Institute for Security, Technology, and Society; the work was also supported in part by Google. However, views and conclusions are the authors alone. We are grateful to the Google Belay team and the anonymous reviewers for their helpful comments.

Contact `chen-qin@cs.dartmouth.edu` for information on code availability.

References

[1] M. Alam, M. Hafner, M. Memom, and P. Hung. Modeling and Enforcing Advanced Access Control Policies in Healthcare Systems with SECTET. In Holger Giese, editor, *Models in Software Engineering*. Springer-Verlag, 2008.

[2] J. Allen, Jr. At the Vatican, Exceptions Make the Rule. *The New York Times*, September 2005.

[3] W. Chun. Using Federated Authentication via OpenID in Google App Engine. `https://developers.google.com/appengine/articles/openid`.

[4] T. Close. Waterken Server. `http://waterken.sourceforge.net`.

[5] J.B. Dennis and E.C. Van Horn. Programing Semantics for Multiprogrammed Computations. *Communications of the ACM*, 9(3):143–155, March 1966.

[6] E. Felten. Too Stupid to Look the Other Way. *Freedom to Tinker*, October 2002.

[7] A. Ferreira, R. Cruz-Correia, L. Antunes, and D. Chadwick. Access Control: How Can it Improve Patients' Healthcare? *Studies In Health Technology And Informatics*, 127:65–76, 2007.

[8] T. Grandison and J. Davis. The Impact of Industry Constraints on Model-Driven Data Disclosure Controls. In *Proceedings of the 1st International Workshop on Model-Based Trustworthy Health Information Systems (MOTHIS)*, 2007.

[9] P. Groen. OpenEMR continue to grow in popularity and use. *OpenHealthNews*, December 2012.

[10] C. Gunter, D. Liebovitz, and B. Malin. Experience-Based Access Management: A Life-Cycle Framework for Identity and Access Management Systems. *IEEE Security and Privacy*, 9(5):48–55, 2011.

[11] A. Karp, M. Stiegler, and T. Close. Not One Click for Security. Technical report, HP Laboratories, March 2009.

[12] R. Koppel, S.W. Smith, and students. Private communication with clinician interviewees, 2008–2012.

[13] R. Koppel, T. Wetterneck, J.L. Telles, and B.-T. Karsh. Workarounds to Barcode Medication Administration Systems: their Occurrences, Causes, and Threats to Patient Safety. *Journal of the American Medical Informatics Association*, 15(4), 2008.

[14] S. Krishnamurthi, P. Hopkins, J. Mccarthy, P. Graunke, G. Pettyjohn, and M. Felleisen. Implementation and Use of the PLT Scheme Web Server. *Higher-Order and Symbolic Computation*, 20(4):431–460, December 2007.

[15] M. Lentczner. Belay Research. `https://sites.google.com/site/belayresearchproject/`.

[16] OpenEMR Project. `http://www.open-emr.org`.

[17] OpenMRS. `http://openmrs.org`.

[18] J. Saleem, A. Russ, A. Neddo, P. Blades, B. Doebbeling, and B. Foresman. Paper Persistence, Workarounds, and Communication Breakdowns in Computerized Consultation Management. *International Journal of Medical Informatics*, 80(7):466–479, 2011.

[19] S. Sinclair. *Access Control in and for Real-World Organizations*. PhD thesis, Dartmouth College, 2013.

[20] S. Sinclair and S. W. Smith. What's Wrong with Access Control in the Real World? *IEEE Security and Privacy*, 8(4):74–77, 2010.

[21] S. Sinclair and S.W. Smith. Access Control Realities As Observed in a Clinical Medical Setting. Technical Report Computer Science Technical Report TR2012-714, Dartmouth College, April 2012.

[22] S.W. Smith and R. Koppel. Healthcare Information Technology's Relativity Problems: A Typology of How Patients' Physical Reality, Clincians' Mental Models, and Healthcare Information Technology Differ. *Journal of the American Medical Informatics Association*, 2013.

[23] M. Stiegler. Rich Sharing for the Web. Technical report, HP Laboratories, July 2009.

[24] United States Department of Veterans Affairs. VistA–eHealth. `http://www.ehealth.va.gov/VistA.asp`.

[25] Y. Wang, S.W. Smith, and A. Gettinger. Access Control Hygiene and the Empathy Gap in Medical IT. In *USENIX HealthSec*, 2012.

[26] R.N.M. Watson, J. Anderson, B. Laurie, and K. Kennaway. Capsicum: practical capabilities for UNIX. In *Proceedings of the 19th USENIX Security Symposium*, 2010.

[27] K.-P. Ye. User Interaction Design for Secure Systems. In *Proceedings of the International Conference on Information and Communications Security*, pages 278–290. Springer-Verlag, 2002.

[28] K.-P. Ye. Aligning Security and Usability. *IEEE Security and Privacy*, 2(5):48–55, 2004.