ABSTRACT
Far from a polished academic paper on a defined problem space, I am hoping that the following will stimulate some discussion on the security, usability, and privacy concerns associated with Personal Health Records, given my experience with the HealthVault platform over the past three years. Goal #1: get the smart minds in the room to think about the future of sharing health data, Goal #1a: don’t get fired.

1. INTRODUCTION
The HealthVault platform launched as a beta in October 2007, and has since run as a means for users to organize and analyze their health data. The following is a quick list of security challenges faced throughout the development and support of the HealthVault ecosystem, and a glimpse of things to come in the field of PHR security.

2. YOU’RE DOING WHAT?!
One of the first questions that consumers and security researchers alike are quick to ask is, “why would I possibly want to store my sensitive health information on the Internet?” Rather than create a position paper extolling the virtues of data availability and connected health scenarios, taking a deeper look into the core question provides perspective on the desired functionality and security requirements of a Personal Health Record system. What are the perceived and actual risks that users assume when storing health data in the cloud? What are the corresponding risks faced by the organization responsible for developing and supporting the infrastructure? Lastly, what threats and mitigations are unique to this problem space, and what does the future hold in store for the field?

3. AUTHENTICATION
One of the first tasks associated with building a PHR system is creating/selecting a strong user and application authentication mechanism that will identify the actors accessing your health information. Currently, Microsoft HealthVault and other popular PHR providers are relying upon existing third parties for user authentication. In particular, HealthVault supports multiple authenticating services in the form of Windows Live ID and a selected subset of OpenID providers that meet certain security requirements. This flexibility provides redundancy in situations of critical data availability, as well as alternate authentication options for users that might desire stronger-than-default account safeguards [1]. Relying on third-party authentication systems allows engineering teams to focus on delivering core scenarios while leaving the overhead associated with user management (password resets, brute-force protection, etc.) to a dedicated team of experts in that field.

Application authentication is often handled differently than user authentication, as the bar can and should be set higher for partners looking to access the information stored within a PHR. Public-key cryptography provides the groundwork for application authentication in the HealthVault ecosystem, but the details are best left to reference in the MSDN Developer Center [2].

4. GRANULAR AUTHORIZATION
Another crucial (and challenging) component of a PHR is the authorization model that governs how data flows in, out, and within the system. Empowering users with the fine-grained authorization options necessary to scope data access to the most appropriate levels can also confuse them to the point that insecure
decisions become default behavior (the “just click yes” response that can become second nature to users).

Personal Health Records are fundamentally different than Electronic Medical Records, in that the user is (typically) full owner of the data and of the trust decisions made with that data. This quality raises several questions on the role and purpose of PHRs: If the user/patient is given complete control over their data, what role should it play in making medical decisions? Could purposely-omitted data pose a risk to user safety? How do you provide data integrity in a system where users and applications can create and modify a variety of data?

5. TRUST BOUNDARIES GALORE
You might recall some of the issues that arose when native applications were marking sections as Shared, so that multiple processes could write to and read from the same physical address range in memory [3]. End result: any malicious application running in the same session could inject corrupted data into the shared section, often allowing Application A to exploit security vulnerabilities in Application B. Well, take that same idea (multiple applications writing to a shared memory space), and apply it to a single platform for reading and writing health data. Data written in one context is read in another, and developer caution must be taken to prevent Cross-Site Scripting, SQL Injection, and other data-interpreted-as-code threats [4]. Throw in cross-user sharing, collusion threats between applications, along with the aforementioned authorization requirements, and the associated problems can yield interesting defensive designs.

6. POLICIES, POLICING
In creating an ecosystem comprised of partner applications, the platform must introduce and enforce a number of “rules of engagement” – the goals of which are to maintain security and integrity of the overall system. A combination of policy (partner agreements, legal terms) and policing (periodic automated reviews, baseline software version requirements) should be employed to protect user data. Any open-application ecosystem must realize that brand reputation, and ultimately user trust, depends on the security of not just the platform, but of all applications that make up the system.

7. BUT WAIT, THERE’S MORE!
The above were only a quick and small selection of the set of security challenges associated with developing and running a PHR system. Given the format of the HealthSec conference, I’m hoping to spend additional time on the topics that are only mentioned here: DoS and data availability, the role of cryptography, and the role and responsibilities of a PHR provider.

8. REFERENCES