Emergency Alerts as RSS Feeds with Interdomain Authorization

Filippo Gioachin, University of Illinois at Urbana-Champaign, gioachin@uiuc.edu
Ravinder Shankesi, University of Illinois at Urbana-Champaign, rshanke2@uiuc.edu
Michael J. May, University of Pennsylvania, mjmay@seas.upenn.edu
Carl A. Gunter, University of Illinois at Urbana-Champaign, cs.uiuc.edu/cgunter
Wook Shin, University of Illinois at Urbana-Champaign, wookshin@uiuc.edu

***Abstract*—Emergency alert systems typically demand push notification because of the infrequency of such events and the urgency of notifying parties about them. However, push notification systems like email have many limitations, such as susceptibility to SPAM and security vulnerabilities. We explore the idea of basing health alerts on RSS feeds, which are a polling-based notification system. Since emergency alerts may be restricted to parties like doctors or health administrators and may be drawn from diverse administrative domains, RSS for health alerts requires a mechanism for expressing and enforcing inter-domain access policies for feeds. In particular, we explore using Shibboleth, a federated identity system developed for use in universities, and an attribute-based policy language, to provide secure RSS for emergency alerts. We validate the approach by showing how it can be used to deliver CDC PHIN health alerts. Our experimental validation shows that, based on our design, existing server technologies can obtain acceptable throughput even with fairly complex and diverse access policies.**

This article appears in the proceedings of the IARIA International Conference on Internet Monitoring and Protection (ICIMP '07), Santa Clara CA, July 2007.

I. Introduction

Web-based news feeds have become a popular way for people to follow current events. Protocols such as Really Simple Syndication (RSS) provide an easy way to subscribe to one or more feeds and receive updates at suitable intervals. The technique is also effective for personal news such as blogs, which may interest a specialized community. On the Internet today, the majority of such feeds are broadcast to the web generally. It is challenging to restrict distribution so that only a limited collection of subscribers can see content. The problem can be addressed in limited contexts: for instance, a provider might allow publishers to restrict their content to parties who have accounts with the provider and are listed by the publisher in an Access Control List (ACL). Such techniques apply only to the users of a specific provider, and the management of the ACL itself is not scalable.

One area of interest for news notification is emergency alerts, such as ongoing status in events like fires, hurricanes, epidemic outbreaks, and so on. Existing mechanisms can accomplish this well so long as the news is broadcast to the public. However, when there are restrictions on the audience, the mechanisms experience all of the limitations of the current state-of-the-art for restricted access feeds. One particular challenge is that such notifications, although restricted, also need to reach principals in a diverse collection of administrative domains, such as government (at many levels), health care workers and privately-owned first responders (like ambulances and tow trucks). Hence, it is not practical to base a solution on a fixed provider who enables ACLs.

Our aim in this paper is to propose and analyze a scalable approach to inter-domain authorization for news feeds. A scalable approach entails the ability to handle inter-domain access with non-trivial access control policies, an ability to interoperate with existing software components and standards (such as RSS), and acceptable performance. For example, for the first requirement, a server might provide two feeds, one for "classified" data and one for "unclassified" data and let users subscribe to the former only if they have the right credential. However, as the types of access restrictions become more complicated than this, the number of different feeds for different access classes proliferates and the approach becomes unscalable. The second requirement means that there must be limited or no changes in client and server software except to the extent that these are reasonably accommodated with customization techniques like Java Script or PHP. For the types of applications we consider, security is a requirement, and it will have a price, but one aims to limit this price as much as possible. Our specific performance target is to deliver worst-case inter-domain access control with dynamically-determined policies at about 50% of the throughput of Secure Socket Layer (SSL) encrypted feeds with no access control. Caching will eliminate most of this access control overhead in the average case.

Our approach is based on a system we call "Shibboleth RSS" because it uses existing RSS feeds augmented with access control based on Shibboleth, an inter-domain system that uses attributes of principals to determine authorization. Attribute-Based Access Control (ABAC) is a powerful technique that can handle complex policies in an elegant way and integrates well with existing data systems. In Shibboleth RSS, a user approaches the server with a Shibboleth credential, which provides his attributes, and these are used to create a customized feed that provides all items allowed to a user with those attributes. On the server side, the access control policies are described using Boolean formulas, one for each item in the RSS feed.

The main results of the paper are an architecture and implementation for Shibboleth RSS, and a demonstration of how it can be used to implement an interesting emergency alert system where flexible policies can be processed efficiently by the server. The target application uses formats developed for Partner Communication and Alerting (PCA) by the Public Health Information Network (PHIN) initiative of the US Centers for Disease Control and Prevention (CDC). In particular, we provide feeds of eXtensible Markup Language (XML) Common Alerting Protocol (CAP) documents where the CAP alerts include XML access policies in elements reserved for such restrictions. Our application provides an editor for creating alerts in the CAP format. These are kept in a data structure that is converted dynamically with Extensible Stylesheet Language family Transformations (XSLT), based on alert policies and subscriber attributes. Performance results show that a basic platform can provide satisfactory throughput using an implementation based on the PHP hypertext processor.

The rest of this paper is organized as follows. Section II discusses background information. Section III presents the architecture of our system. A demonstration of the architecture is presented in Section IV. We present our implementation and explain the experiments that we performed on it in Section V. Related work is in Section VI. Section VII concludes.

II. Background

Our architecture uses a combination of RSS and Shibboleth to deliver secured messaging. We validate this with an application based on the CDC PHIN alert system. In this section we provide background on these basic topics.

A. RSS

The RSS family of standards are a scheme for the broadcast of web content. At its simplest, an RSS *feed* is an XML file that summarizes the contents of a web site. A feed contains a list of items which have a title, a short summary, a web link, and a back link to the feed provider. RSS *readers* download feeds from particular sites automatically and display them to users. The addition of creation and expiration times (time to live) to feeds allows readers to ensure that they are always displaying the freshest version of a feed. Feeds can be combined and redistributed using RSS *aggregators* to create summaries of many different web sites.

RSS has been most popularly applied to news sites and blogs, but its strengths would apply equally to the broadcast of any rapidly evolving, article-based content which may be appended or edited. It allows content creators to create automated summaries of their work which save readers the time of going to a site and looking around to see what is new. As a notification engine, RSS is a *polling-based* rather than a *push-based* technology like email.

With the rapid deployment of RSS in browsers, email clients, and stand-alone readers, questions about the security and privacy implications of using it have arisen [27]. From the user side there have not been many security advances perhaps since there have been no documented cases of malware, viruses, or spam being propagated via RSS to date. From the content creator side there are technologies to apply password protection to feeds as well as schemes to combine various authentication and encryption technologies to provide end-to-end privacy. As the technologies mature and gain more widespread adoption in enterprises, security considerations will move to the foreground.

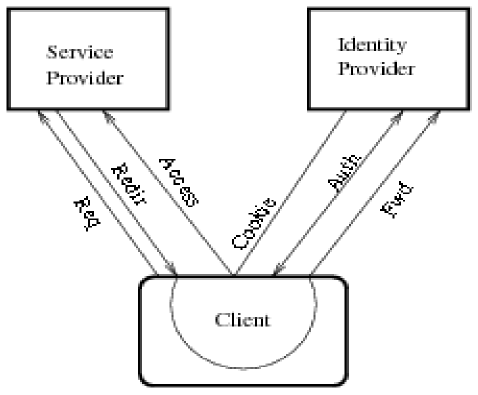

Fig. 1. Shibboleth High-Level Overview

B. Shibboleth

Shibboleth [19] is an inter-domain attribute-based authentication system based on Security Assertion Markup Language (SAML) 1.1 [21]. It enables web-based single sign-on between different entities in a *federation* of organizations that share a trust relationship. In Shibboleth this trust is achieved by the sharing of public-key certificates. Shibboleth allows both authentication and attribute-exchange between two parties in a federation. Users can thereby access resources of multiple parties within a federation without having to maintain an account with each one of them.

Figure 1 gives a high-level overview of the entities in Shibboleth architecture and the steps followed when a user tries to access a resource. The Service Provider (SP) provides an access controlled resource to the users within a federation. The Identity Provider (IdP) vouches for the identity of users and typically maintains user attribute information. Depending on its configuration, the SP can allow access to resources based on user identity or attributes.

Here is a simplified user-interaction scenario as per the figure. A new client visiting a protected web-page at the SP's website (*Req*) is redirected to visit his IdP (*Redir*, *Fwd*) using HTTP redirection. The client authenticates himself with the IdP (*Auth*). After the IdP verifies the client, it sets an authenticated cookie (*Cookie*) that vouches for the user. The cookie may optionally contain relevant user attributes that the SP is allowed to view. The IdP redirects the user back to his original web-page (*Access*). The SP verifies that the cookie is authenticated (since it has the IdP's certificate) and allows the access to the web-page based on the user's identity or attributes.

C. PHIN

The CDC is developing standards to enhance the preparedness of health providers and public health officials across the U.S. The Health Alert Network (HAN) [8], begun in 1998, was funded by CDC to provide a high speed, secure Internet channel for health officials and providers across the various states to communicate and receive health alerts. HAN was later folded into the alerting functional area of PHIN (www.cdc.gov/phin), a larger initiative begun in 2002, which includes four other functional areas for preparedness.

As part of HAN, each state developed its own secured web portal for communication with health care providers and officials. Participating organizations developed, purchased, or received from the CDC client software to enable users to read and send messages. Users send messages to each other and across portals using the Electronic Business using eXtensible Markup Language (ebXML) [32]. The PHIN requirements for performance [11], message formats (PCA) [12], and cross-functional components [10] impose procedural, architectural, and security requirements on the member portals and communication systems. Requirements include message encryption, digital signatures, unique message identification, message expiration, and access control restriction. Alerts are normally sent electronically, but for emergency situations other communication paths (such as telephone, fax, pager, *etc.*) are required as well. Due to the sensitive nature of health alerts, access may be restricted by communicating parties.

CDC recommends using the CAP [4] formats for sending health alerts [12]. CAP is an open XML standard format for alerts and notifications. A CAP alert message contains start time, expiration time, targeted geographical area, and other relevant fields. As a standardized alert format used by various U.S. government departments, CAP enables applications to receive alerts from different sources and process them uniformly.

III. Architecture

Our secure notification architecture enables subscribers to securely send messages to and retrieve RSS notification summaries from local HANs. Figure 2 shows its main components. Each client is associated with one or more IdPs (to the left), which maintain public health directories of users and their attributes. Local HANs (to the right) trust certain IdPs to provide user credentials and attributes. A HAN maintains a database of CAP notifications that it publishes as well as their associated message-level access policies. The CAP filter restricts subscriber access to notifications based on policies and provided attributes. A CAP to RSS converter translates notifications into a format suitable for RSS readers.

The arrows in the figure represent the typical execution path for a subscriber request. The client begins by sending a request (P1) to a HAN publisher to view new notifications. Since the client is not yet authenticated, the HAN server sends a redirection to the client's IdP. The client sends authentication tokens to the IdP (P2), which verifies them and retrieves the client's attributes from its public health directory database (P3, P4). The IdP creates a signed cookie with the client's name and

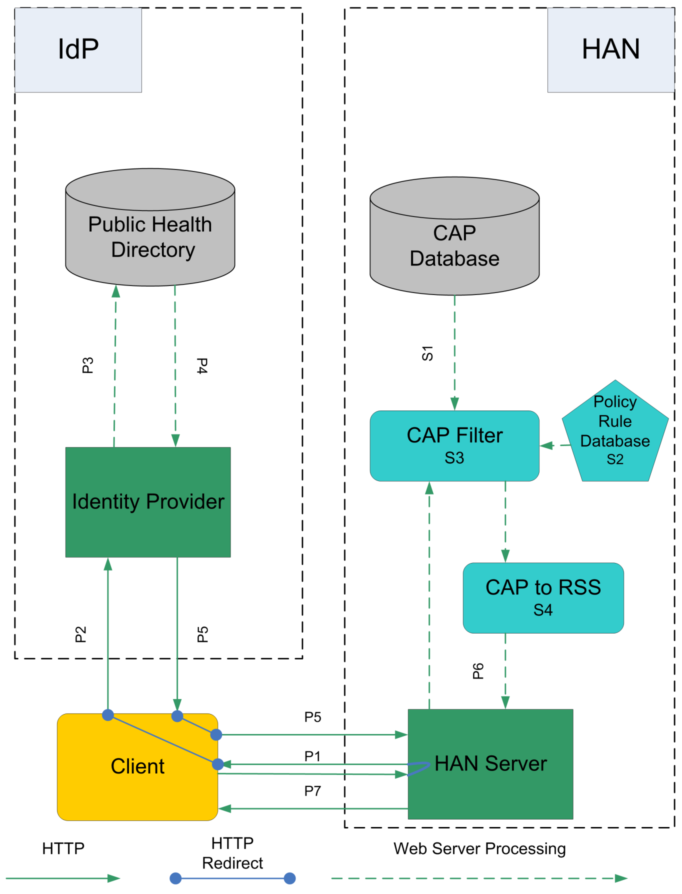

Fig. 2. High-Level Architecture overview of the Shibboleth RSS system.

attributes, and sends it to the client (P5) which then forwards it (P5) to the HAN server. After authenticating the cookie, the HAN server creates an RSS feed for the stored alerts based on the user's signed attributes (P6). Finally, the server returns the feed to the client (P7). Details about the processing within the HAN are given below in III-C.

The HAN service provider authenticates the user's cookie at the first request. It then stores the user's attribute information into a PHP session and stores the session ID as a cookie with the client. This allows the client to retrieve his feed multiple times, without requiring any further re-authentication during that session.

A. Design Considerations

The design of our secure RSS notification system was driven by the following considerations.

First, we needed to allow inter-domain authentication so that health alerts from different states could be viewed by appropriate users. We sought to achieve this without changing the existing framework where users typically only maintain an account with their state health system. To achieve this we used the inter-domain authentication tool Shibboleth.

Second, we needed to allow fine grained access control filtering to the alerts so users only see alerts they are authorized to see. To perform this efficiently, we used a flexible attribute-

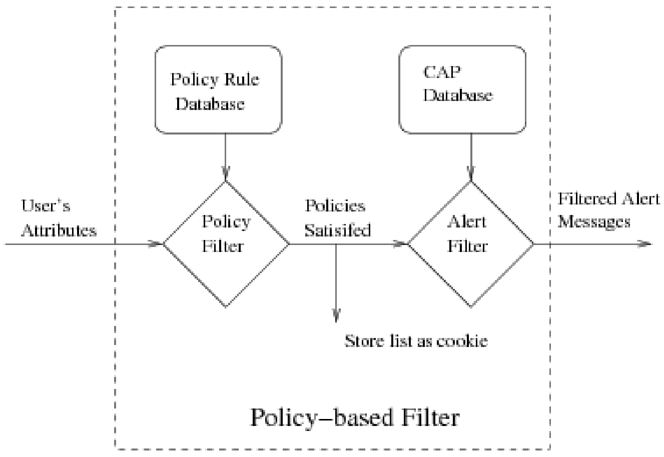

Fig. 3. Filtering of alerts based on user attributes.

based policy language, described later in III-C. The name of the policy is attached to the alert in the specific CAP-defined field.

Third, for practical reasons, the system should be built on top of existing software and provide acceptable request throughput. Requiring users to install new software would severely limit the architecture's usefulness, so we designed a web browser and RSS-based user interface. In particular, this entails placing the policy and filtering logic on the servers.

B. Identity Providers and HAN Publishers

For the IdP's public health directory, we maintain the records of users' roles (*e.g.* Doctor, Nurse), organizational level (*e.g.* city, regional, national) as well as twenty-three other attributes. For large scale implementations, the CDC recommends a Public-Health Directory Schema (PHINDir) [9] that organizations may standardize upon, but for our work we only implemented a subset of them. Since policies depend on attributes, it is essential that all IdPs offer a certain minimal set of common attributes. We implemented the IdP's public health directory using the Fedora directory server.

The HAN publisher maintains a database of CAP alerts. Since they are in a standard format, it is easy to add other CAP alerts from different sources. We introduced a web based GUI management interface to allow administrators to add new alerts. The publisher server interfaces with users via a set of PHP scripts.

C. Policies

Access control policies make decisions based on user attributes. For instance, doctors may be allowed to see a preliminary public health alert before it is confirmed and the general public is informed. To enforce this, a naive approach would be to annotate each message with its own policy, but such an approach would quickly become unreasonably and unnecessarily complex. Instead, we use a two-stage scheme for evaluating policies, as shown in Figure 3. First, the site administrator writes a set of default policies that the publisher can enforce. Those policies are stored in a Policy Rule Database. Then, message writers can associate their messages with any of the stored policies. There are two performance benefits to using this approach. First, we only need to evaluate each policy at most once per request. Second, we can store the list of policies that the user satisfies in the user's session information. This means that we only need to evaluate the user's policies at most once per session. This significantly improves performance by allowing users to refresh their feeds without re-evaluating all the policies.

Policies are stored in an XML file which allows for arbitrary boolean logic expressions on attributes. Available combinators include equality ($=, \neq$) and value comparisons ($<, >, \geq, \leq$). The example below gives a sample policy snippet which allows access only if the user is a federal or local public health official.

```
<restriction>
 <name>Agent</name>
 <description>Agent descr.</description>
 <function>
  <and>
   <eq value="PUBLIC_HEALTH">Role</eq>
   <or>
    <eq value="STATE">Organization</eq>
    <eq value="FEDERAL">Organization</eq>
   </or>
  </and>
 </function>
</restriction>
```

The restriction tag begins the policy snippet. Element `name` is the unique identifier of the policy, as is used in the CAP *restrictions* field. `Function` is the boolean expression evaluated with the user's attributes. In this example, Role and Organization are the attributes of the user, and PUBLIC_HEALTH, STATE and FEDERAL are some of the possible values of these attributes.

Returning to Figure 2, when the HAN server receives a request accompanied by a signed attributes cookie, it loads the alerts from the CAP database and parses them into an XML tree (S1). It then either loads and evaluates the policies for the requesting user (S2) or retrieves them from the user's stored session state. If the user does not provide an authenticated attribute cookie, the filter created only allows nil policies (*i.e.* public notifications). The resulting filter is applied to the alerts to generate the set visible to the user (S3). Finally, the CAP structure is transformed to RSS (S4). The translation is performed using an XSL [2] transformation (XSLT) [5], invoked directly from the PHP script.

The final RSS output is bound to another XSL file to provide an HTML-compliant format that is displayable by browsers. The HTML format includes an additional, optional set of Javascript scripts to provide searching and other capabilities. There is room for additional optimizations as well, such as caching the RSS feeds for common policies. This will greatly improve efficiency if the CAP database does not change frequently. We consider other optimizations in Section V.

IV. Demonstration

We developed a pair of scenarios to demonstrate how our Shibboleth RSS system can be used for communication between individuals who are in different health care domains. We have recorded a movie demonstrating the two scenarios using our current implementation. The movie is available at seclab.uiuc.edu/securerss.

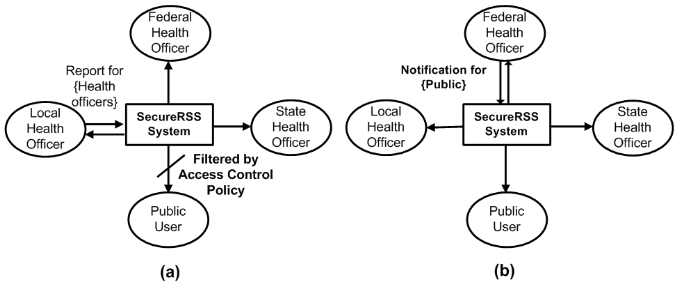

Fig. 4. Demo Scenarios: (a) a report message that is only available to health officers and (b) a notification message that is available to public.

1) **Restricted Report Scenario:** A local health officer reports a suspected bird flu case to the Shibboleth RSS system and asks other health officers to watch for similar cases. The report needs to be restricted to authorized officers since the reported case has not been confirmed. The local officer describes the designated readers using an access control policy as described in III-C, so unauthenticated users cannot read the report unless they have authenticated to and received signed attributes from a trusted IdP. Figure 4(a) illustrates this scenario. The demonstration of the scenario is performed in the following steps:
   a) First, a local health officer starts his web browser and points it to the Shibboleth RSS page.
   b) When the officer presses the login button, an authentication window pops up. The authentication window was provided by the officer's IdP server.
   c) The officer enters his ID and password, logs in the system, and sees new menus that are only available to authenticated persons.
   d) He reports a case of suspected bird flu and picks an access policy from a list of previously defined policies.
   e) In the second part of this scenario, a federal health officer starts a browser and points it to the Shibboleth RSS page.
   f) Before she logs in, she cannot see the suspected bird flu case.
   g) Once she logs in using her IdP, she is able to read the sensitive alert.
2) **Public Report Scenario:** A CDC officer confirms that the suspected bird flu case is real, so a public outbreak notice is made available. Users can read the public notification without authenticating to the system. This is illustrated in Figure 4(b). The demonstration of the scenario is performed in the following steps:
   a) First, a federal health officer logs into the Shibboleth RSS system using attributes which are passed from her IdP.
   b) She has found that a suspected bird flu case was real, so she writes a notification to give a public alarm.
   c) After writing the notification, she selects a policy which allows anyone to access it.
   d) Next, a user accesses the CDC health alert system without logging in and updates his RSS feeds.
   e) The user can then see several public notifications including the bird flu warning.

These scenarios do not represent at all the potential complexity of the policies that may be used in the RSS feed. However, they do illustrate the basic idea that the feeds are customized to user attributes based on Shibboleth credentials. We explore the issues with complex policies in the next section with respect to their impact on performance.

V. Experimental Validation

For the validation of our system, we considered only the components related to the generation and retrieval of the RSS feed. For this, we focused on the server throughput. We did not consider the cost associated with the user's login since it is an operation performed sporadically. In this section we analyze the performance of the various operations performed by the server, followed by the throughput obtained with a few optimizations. All of the tests were performed with SSL encryption enabled since it is a fundamental component for the point-to-point security. We conclude the section with considerations of the effects of the SSL encryption on the system.

In order to test our implementation, we set up an Apache web server running on a Linux SuSE 10.1 machine with all the packages for PHP, and its data structures manipulation software associated with XML and XSLT. The server machine has a 3 GHz Pentium-4 single core processor, 1 GB of RAM, and a 100-BaseT ethernet interconnect. While other server class machines are typically more powerful, our test server has a reasonable ratio between bandwidth capacity and processor power. Our architecture can support any numbers of IdPs, but for our tests we deployed only one since it does not represent a performance bottleneck. We implemented and deployed the IdP on a Linux machine running Fedora Core 5 with Shibboleth's open-source IdP version 1.3c. We used Apache http daemon with Apache Tomcat as the Java servlet container for running the IdP.

We constructed two databases for storing the CAP information, one of size 128 kB and the other of size 512 kB. The smaller one contains 54 infos clustered into 15 alerts while the big one has four times as many. Similarly, we built two policy files, one with ten rules and the other with fifty rules. From these we created four possible scenarios by combining them depending on whether there are small or large number of CAPs and few or many policies. Each CAP database uses a random set of rules from the set available for the given scenario. We consider the small database and policy file a more realistic scenario in real applications than the others. We propose the others as a comparison with extreme situations. In particular, we presume data from the databases would be moved periodically to permanent storage and the database would contain only the most recent alerts. As for the policies, since the user posting alerts will have to search through the available policies to find the most suitable one, a larger number of policies will make it harder for the poster to select the proper one.

For each test, we used ten machines acting as clients. Each machine was set up to issue 21 downloads at a time (one for each of the 20 authenticated users and one for a single non-authenticated user) and repeat this operation after all 21 downloads finished. We found that using this many clients pushed the server to its limit of bandwidth or, almost always, processor usage. Each test was run for a total time of one and three minutes, and the effective processor/bandwidth usage was monitored throughout the execution.

For each scenario, we divided the work performed by the server in its various tasks and took a performance measurement for each incrementally. The results are shown in Figure 5. The

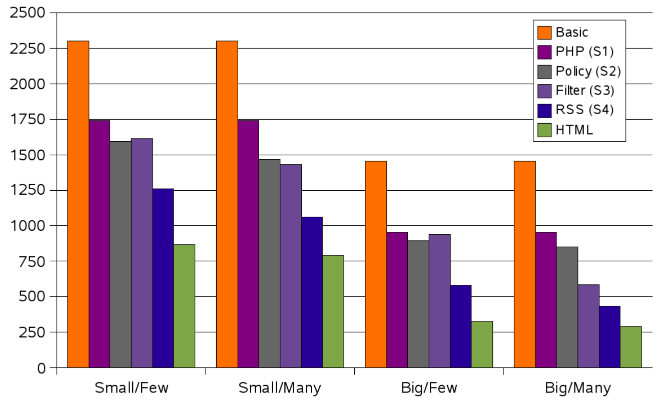

Fig. 5. Throughput of the server in pages served per minute split according into tasks.

first bar represents a download of a file of the same size of the CAP input without processing by our script. The remaining bars represent the rate at which further steps could be accomplished. For example, the second (purple) bar represents average throughput doing only the SSL and PHP operations, whereas the third represents average throughput doing the SSL, PHP, and policy steps, and so on. The final (green) bar represents throughput of our server when it performs all steps (SSL, S1-4, and HTML formatting). Since the measurements are averages, they do not strictly decrease in all cases. Overall the measurements in Figure 5 represent a worst-case scenario where there are no optimizations at all.

Using these figures we identified bottlenecks of the system and developed a few optimizations:

- We stored to the file system a version of the unfiltered database after being transformed to an RSS feed and used it to perform the filtering, thereby avoiding the conversion stage for each request. (The original CAP database is still maintained for other uses.)
- We evaluated the policies for a user once at login and stored them in the session allocated to the user. Since the user is given a cookie with a reference number to this session ID, this is secure as long as the session data is stored on the server.
- We combined the filtering into a single operation, by searching all the policies not satisfied at the same time. This contrasts with the previous version which independently deleted each policy that was not satisfied.

We did not add the most effective optimization, caching the results of policy processing, because we wanted to stress the system in a worst case filtering scenario where virtually every request requires complete processing.

After these modifications, we re-ran our tests and obtained the results plotted in Figure 6. As explained in section III-C,

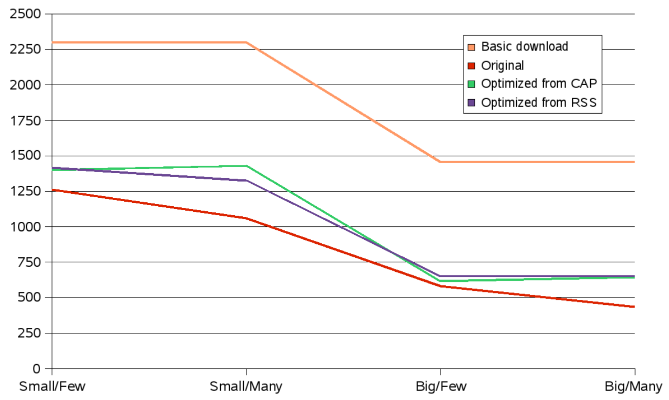

Fig. 6. Throughput of our RSS feed, with different optimizations, compared to a simple download of a file encrypted with SSL.

the XSL transformation is performed in two stages. However, the natural output for the system is an RSS feed and the client is responsible to prepare it for visualization in the most appropriate way. In the optimized version we therefore leave the final conversion to the client and consider the RSS feed as the final output.

In Figure 6, there are the two curves from Figure 5 ("Basic" and "RSS (S4)") plotted against two more for the optimized versions. These two optimized versions differ in the implementation of the first optimization described above, namely the storing of the unfiltered database after conversion to RSS. In particular, the line "Optimized from RSS" implements includes this optimization and the other does not. We reported both of these since neither completely outperforms the other and the outcome depends on the scenario. The reason for this can be seen considering that while the "RSS" optimization avoids the final conversion, it presents a more complex structure to the alert filter, which consumes more time for filtering.

From the graph, we can see a throughput of our implementation varies from 45% of the unfiltered throughput in the worst case, to more than 60% for smaller files. Moreover, we can see that while there is an impact on performance from the size of the input, as expected, there is no impact from the number of policies available to the user.

Finally, we evaluated the impact of SSL encryption in comparison with the impact of our filtering scheme. The results are shown in Figure 7. The "Basic plain" curve is limited by the bandwidth of the system, while the others are mostly limited by the CPU processing power. Our scheme is the only source of bottleneck when the CAP database is large, but for smaller inputs SSL negotiation and encryption is the predominant cost. We do not represent it in the graph, but we found that for files even smaller in size, the SSL overhead limits the throughput to less than 2,500 downloads per second.

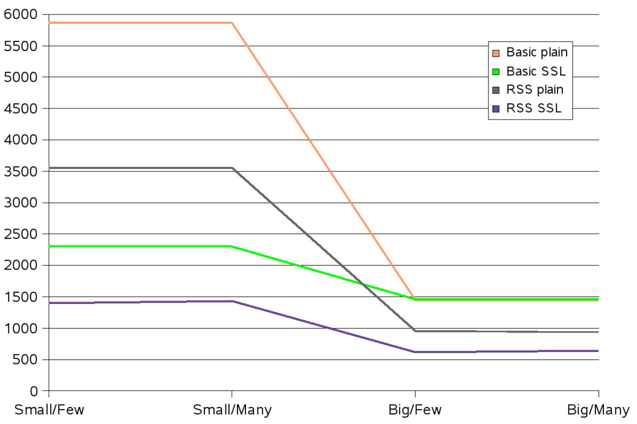

Fig. 7. Performance impact of SSL encryption vs. our multistep RSS feed creation.

We can conclude that any online system that needs to be secured through SSL encryption (such as web email) will have a limited number of connections allowed per second, even though each file downloaded is very small. This does not provide a comprehensive comparison with all the other existing HAN implementations, but it does provide insight on the limitations that systems that require strong security face.

In addition to the experiments on the machine mentioned above, we also ran them on a server set up as SuSE 10.2 virtual machine running under VMware with two processors. The results are not reported here because they are similar to those presented above with the exception of a scaling factor due to the different processing power. This suggests that our conclusions are robust with respect to a meaningful range of server capabilities.

VI. Related Work

Even though RSS was designed for the syndication and broadcast of content, as noted in the background section, recent applications have brought up the need for feeds to have security and privacy protections [25][27][29]. At a simple level, some blog sites such as Google's Blogger allow content creators to password their pages and RSS summaries, but since standard http authentication [7][13] does not encrypt the password in transmission, it suffers from security weaknesses. Stronger protection schemes have been implemented using http authorization over SSL [28][15] and feed encryption [17] using Blowfish [30].

The notion of applying access policies to feeds merges existing work on inter-domain authentication and authorization enforcement with new distribution technology. Enterprise software for protecting feeds is available (*e.g.* Newsgator [23]) for distribution of feeds on intranets. Web based document storage and retrieval systems with federated authentication have been introduced as well (*e.g.* WSEmail [20]). Our introduction of inter-domain attribute-based authentication using Shibboleth [19] is an extension of previous work to adapt to public health alerts, a application domain which already relies on a state-by-state distributed authentication and authorization framework.

Attribute based encryption [26], access control [16][33], and messaging [3] have been explored in recent work. Attribute-based systems have an advantage over access control list and role based systems in that they enable permissions to be granted based on properties of the requestor, removing the need to manage and update lists of access rights and roles. We use attribute based policies akin to those of Goyal, *et al.* [16] to compose generic policies that can base decisions on distributed attribute servers.

Our work to streamline public health communications using RSS feeds builds on the recent developments in electronic public health infrastructure. The infrastructure has been built piecemeal across different countries with different systems for cities, regions, and states [1]. One vein of work to improve the infrastructure has been to analyze the implementation requirements for networks of different sizes and purposes [24][18][14][6][22], however, as we have noted, the fragility and problems that come from non-uniform systems argue strongly for unified solutions [31]. In that direction, there has been work in the creation of specialized information and alert portals, for example Zeng, *et al.*'s West Nile Virus and Botulism portal [34]. While disease information portals can help with the correlation of data from disparate sources, they are not as well suited for the active messaging and security that the PHIN PCA architecture has sought to solve though its messaging standards. Our standards-based messaging solution with its strong security properties and proven performance and scalability offers a superior solution to unifying health alerts and communications.

VII. Conclusion

Our architecture for secure RSS based emergency notifications shows the flexibility and utility of attribute-based inter-domain authorization systems. Our Shibboleth-based approach relies on authenticated cookies and federated identity and service providers to allow the creation and enforcement of message level policy restrictions. We use RSS to summarize the contents of rapidly changing notification databases and distribute personalized summaries to requestors. Our application of SSL and Shibboleth is a novel approach to satisfy the PHIN PCA security requirements that enables the easy composition of message policies and removes cooperation barriers between disparate local identity providers.

In terms of performance, even though RSS is a polling based update mechanism which would seem to be performance challenged, with off-the-shelf tools and some simple optimizations we can achieve reasonable server throughput. Importantly, we show that our biggest performance penalty comes from using SSL tunnels, not the evaluation of policies. Since the notifications server is required by law to encrypt its messages, this means that the application of policies can be done at reasonable cost relative to the (required) cost of encryption.

As part of future work, we will consider user interface issues such as optimizations for assigning types and policies to messages. Our web interface can be extended with Shibboleth authenticated web services to allow for rich client interactions in creating messages. Browser and RSS reader Shibboleth and SSL extensions will enable more automated feed reading as well.

Acknowledgements

We appreciated the contributions of Vijay Nori on client software in the early steps in this project. We also appreciated help from Rakesh Bobba and Himanshu Khurana. This work was partially supported by: NSF CNS05-5170 CNS05-09268 CNS05-24695, ONR N00014-04-1-0562 N00014-02-1-0715, and a grant from the MacArthur Foundation.

References

[1] Carla AbouZahr and Ties Boerma. Health information systems: the foundations of public health. *Bulletin of the World Health Organization*, 83(8):578–583, August 2005. Ref. No 04-014951.

[2] Anders Berglund. Extensible stylesheet language (XSL). W3c recommendation, 5 Dec 2006. www.w3.org/TR/xsl11.

[3] Rakesh Bobba, Omid Fatemieh, Fariba Khan, Carl A. Gunter, and Himanshu Khurana. Using attribute-based access control to enable attribute-based messaging. In *Annual Computer Security Applications Conference (ACSAC '06)*, Miami Beach, FL, Dec 2006. Applied Computer Security Associates (ACSA).

[4] Art Botterell. Common alerting protocol, v. 1.0. OASIS standard, OASIS, 2004.

[5] James Clark. XSL transformations (XSLT). W3c recommendation, 16 Nov 1999. www.w3.org/TR/xslt.

[6] Andrew S. Doniger, Daniel Labowitz, Stephen Mershon, and Ivan J. Gotham. Design and implementation of a local health alert network. *Journal of Public Health Management and Practice*, 7(5):64–75, Sep 2001.

[7] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. *RFC 2068: Hypertext Transfer Protocol – HTTP/1.1*. Internet Society, Jan 1997.

[8] Centers for Disease Control and Prevention. Health alert network. www2.cdc.gov/han, 18 Jan 2002.

[9] Centers for Disease Control and Prevention. *Public Health Directory Schema*, 1.1 edition, 23 Jun 2003.

[10] Centers for Disease Control and Prevention. *PHIN Preparedness: Cross Functional Components*, 1.0 edition, 26 April 2005.

[11] Centers for Disease Control and Prevention. *PHIN Preparedness: Key Performance Measures*, 1.0 edition, 26 April 2005.

[12] Centers for Disease Control and Prevention. *PHIN Preparedness: Partner Communications And Alerting Functional Requirements*, 1.0 edition, 26 April 2005.

[13] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. *RFC 2617: HTTP Authentication: Basic and Digest Access Authentication*. Internet Society, June 1999.

[14] Ann Fruhling, Anthony Sambol, Steven Hinrichs, and Gert-Jan de-Vreede. Designing an emergency response system for electronic laboratory diagnostics consultation. In *Proceedings of the 39th Hawaii International Conference on System Sciences*. IEEE, IEEE Press, 2006.

[15] Steven Garrity. Private RSS feeds: Support for security in aggregators. labs.silverorange.com/archives/2003/july/privaterss, 9 July 2003.

[16] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 89–98, New York, NY, USA, 2006. ACM Press.

[17] Joe Gregorio. Secure RSS syndication. *XML.com – XML From the Inside Out*, 13 July 2005. www.xml.com/pub/a/2005/07/13/secure-rss.html.

[18] Lawrence P. Hanrahan, Henry A. Anderson, Brian Busby, Marni Bekkedal, Thomas Sieger, Laura Stephenson, Lynda Knobeloch, Mark Werner, Pamela Imm, and Joseph Olson. Wisconsin's environmental public health tracking network: information systems design for childhood cancer surveillance. *Environmental Health Perspectives*, 112(14):1434–9, Oct 2004.

[19] Internet2. Shibboleth project – internet2 middleware. shibboleth.internet2.edu/, 2007.

[20] Kevin D. Lux, Michael J. May, Nayan L. Bhattad, and Carl A. Gunter. WSEmail: Secure internet messaging based on web services. In *IEEE International Conference on Web Services*, Orlando, FL, 2005. IEEE.

[21] Eve Maler, Prateek Mishra, and Rob Philpott. Assertions and protocol for the OASIS security assertion markup language (saml) v1.1. OASIS standard, OASIS, 2003. oasis-sstc-saml-core-1.1.

[22] Clement J. McDonald, J. Marc Overhage, Michael Barnes, Gunther Schadow, Lonnie Blevins, Paul R. Dexter, Burke Mamlin, and INPC Management Committee. The indiana network for patient care: A working local health information infrastructure. *Health Affairs*, 24(5):1214–1220, Sep/Oct 2005. Project HOPE-The People-to-People Health Foundation, Inc.

[23] Newsgator. Newsgator enterprise server. www.newsgator.com/enterprise.aspx, Accessed Feb 2007.

[24] Fay Cobb Payton and Patricia Flatley Brennan. How a community health information network is really used. *Commun. ACM*, 42(12):85–89, 1999.

[25] Mark Pilgrim. How to consume rss safely. diveintomark.org/archives/2003/06/12/how_to_consume_rss_safely, 12 June 2003.

[26] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 99–112, New York, NY, USA, 2006. ACM Press.

[27] Jim Rapoza. RSS' security deadline. *EWeek*, 15 Sep 2006. www.eweek.com/article2/0,1759,2016515,00.asp.

[28] Greg Reinacker. RSS "security". www.rassoc.com/gregr/weblog/archive.aspx?post=775, 8 Sep 2005.

[29] Paul F. Roberts. Secure RSS courts enterprise adoption. *EWeek*, 2 Sep 2005. www.eweek.com/article2/0,1895,1855477,00.asp.

[30] Bruce Schneier. The Blowfish encryption algorithm. www.schneier.com/blowfish.html, Accessed Feb 2007.

[31] Nancy L. Snee and Kathleen A. McCormick. The case for integrating public health informatics networks: An overview of the elements required for an integrated enterprise information infrastructure for protecting public health. *IEEE Engineering In Medicine And Biology Magazine*, 23(1):81–88, January/February 2004.

[32] ebXML Technical Architecture Project Team. *ebXML Technical Architecture Specification v.1.0.4*. OASIS, 16 Feb 2001. ebxml.org.

[33] Lingyu Wang, Duminda Wijesekera, and Sushil Jajodia. A logic-based framework for attribute based access control. In *FMSE '04: Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, pages 45–55, New York, NY, USA, 2004. ACM Press.

[34] Daniel Zeng, Hsinchun Chen, Chunju Tseng, Catherine Larson, Millincent Eidson, Ivan Gotham, Cecil Lynch, and Michael Ascher. *Intelligence and Security Informatics*, chapter West Nile Virus and Botulism Portal: A Case Study in Infectious Disease Informatics, pages 28–41. Springer, 2004.