Security Issues and Requirements for Internet-Scale Publish-Subscribe Systems

Chenxi Wang*, Antonio Carzaniga‡, David Evans†, Alexander L. Wolf‡
Email: chenxi@cmu.edu, carzanig@colorado.edu, evans@virginia.edu, alw@colorado.edu

Abstract

Publish-subscribe is a communication paradigm that supports dynamic, many-to-many communications in a distributed environment. Content-based pub-sub systems are often implemented on a peer-to-peer infrastructure that enables information dissemination from information producers (publishers) to consumers (subscribers) through a subscription mechanism. In a wide-area pub-sub network, the pub-sub service must handle information dissemination across distinct authoritative domains, heterogeneous platforms and a large, dynamic population of publishers and subscribers. Such an environment raises serious security concerns. In this paper, we investigate the security issues and requirements that arise in an internet-scale content-based pub-sub system. We distinguish among those requirements that can be achieved with current technology and those that require innovative solutions.

1 Introduction

Today's mission-critical systems make extensive use of distributed computing over large, heterogeneous networks. A promising technology in achieving distributed computing is the use of publish-subscribe mechanisms (hereafter refer to as pub-sub). A pub-sub system is a communication infrastructure that enables data access and sharing over disparate systems and among inconsistent data models [7]. Gnutella [16] is an example of a pub-sub system. This paper focuses on content-based publish-subscribe where subscribers register interest to information and the infrastructure routes the information to the subscribers based on the information content and the user subscriptions.

In a wide-area content-based pub-sub network, the underlying pub-sub infrastructure is often implemented as a collection of network servers communicating with each other in a peer-to-peer fashion [8]. In such an environment, the pub-sub service must handle information dissemination across distinct authoritative domains, heterogeneous platforms and a large, dynamic population of publishers and subscribers. Many security concerns exist in such an environment. For example, delivering information to interested (and authorized) parties only is an information privacy concern, and so is the concern of keeping the subscription information private. Additionally, the integrity and availability of the pub-sub mechanism must be ensured.

The current designs of pub-sub systems tend to focus on the performance, scalability and expressiveness issues of the mechanism [8][24]. In this paper, we investigate the basic security issues for pub-sub systems. We will distinguish between those that can be achieved with current technology and those that merit innovative solutions. We will not, however, attempt to design a security model in this paper.

2 Publish-subscribe systems

A pub-sub system is a routing network that delivers datagrams from publishers to interested subscribers. Unlike multicast group communications where group addresses and memberships are statically bound, pub-sub systems use a communication model where the eligibility of group membership is evaluated dynamically. Such a communication system has many potential benefits. For instance, instead of requiring publishers to identify destination addresses for their messages (potentially requiring multiple messages to multiple destinations), a pub-sub network can handle message routing in a way that avoids unnecessary message replications.

In a content-based pub-sub system, the network supports a language that specifies the publication and subscription interface. For example, a subscriber specifies a subscription function defined over the content of datagrams. A publisher publishes a datagram composed from a set of legal vocabularies in the language. When a publisher sends out a datagram, the network attempts to deliver that datagram to every interface whose subscription function returns true when applied to the content of the datagram.

Communication in a pub-sub system is inherently a multi-party, many-to-many interaction. In this paper, we

*: Electrical and Computer Engineering Department, Carnegie Mellon University.
‡: Department of Computer Science, University of Colorado at Boulder.
†: Department of Computer Science, University of Virginia.

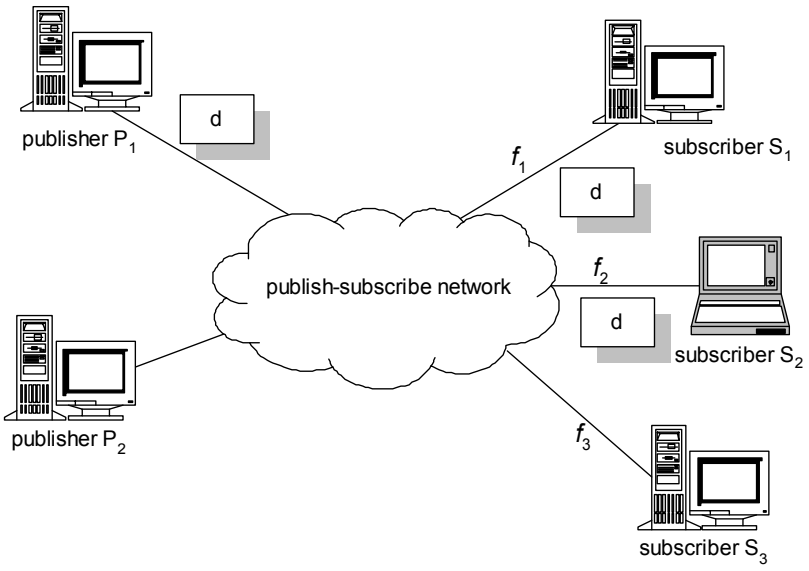

Figure 1: A publish-subscribe system

consider a communication model illustrated by the example in Figure 1. The pub-sub communication protocol can be viewed as follows:

1) Subscribers S_1, S_2, and S_3, send subscriptions f_1, f_2, and f_3 respectively to some hosts in the network.
2) Publisher P_1 sends a datagram d to some entry point of the network, with $f_1(d) \wedge f_2(d) \wedge \neg f_3(d)$; that is, d matches subscriptions f_1 and f_2 but not f_3.
3) The network determines the matching relationships and sends d to S_1 and S_2.

Note that the publisher only sends d into the network once and need not know anything about the subscribers or subscription functions.

We use two example applications throughout this paper. We believe that they are typical of many of the ways that pub-sub will be used. In the later part of this paper, whenever possible we will discuss the security issues within the context of these examples.

Stock quotes dissemination: Consider an application that uses a pub-sub system to disseminate real-time stock quotes. Subscribers specify the stock symbol and a schema (e.g., the frequency of quotes) based on which they will receive quotes.

Human resource resume circulation: Consider an application that allows users (publishers) to post their resumes and sells the resume information to interested human resource offices (subscribers). Subscribers specify key words indicating a particular background to search for relevant resumes.

3 Security requirements

The security requirements for a pub-sub system can be divided into the requirements for a particular application involving publishers and subscribers, and the requirements for the pub-sub infrastructure:

- The *application*, comprising the publishers and subscribers. Publishers and subscribers may not trust each other, and may not trust the pub-sub network.
- The *infrastructure*, consisting of the pub-sub network that provides services to the application. The infrastructure may not trust publishers and subscribers. Components of the infrastructure may not necessarily trust each other.

For example, providing a mechanism that defines who has what access to what information is mostly an application-level concern. It requires a definition of identity, authorization and access control within the pub-sub infrastructure. In the meantime, controlling who is able to change the subscription database maintained by the pub-sub service and restricting channel utilization are infrastructure-level protection issues.

In this section, we explore the various security issues and requirements in the two categories. The general security needs of the application include *confidentiality*, *integrity*, and *availability*, while the security concerns of the infrastructure focus primarily on system *integrity* and *availability*.

Other important issues stem from the fact that the pub-

sub system is a service layer. Invariably, it must deal with applications with varying security needs and requirements. It is therefore inappropriate for the pub-sub system to dictate a global security policy that will be implemented to disseminate information for every application. A major challenge in designing a security architecture for pub-sub is the provision of a flexible security framework allowing diverse policies and mechanisms to be implemented within the same pub-sub infrastructure.

In Section 4, we discuss security issues faced by pub-sub systems that are similar to those present in traditional network security. The content-based routing and dynamic subscription properties of pub-sub systems introduce many new security requirements and challenges. While some of these issues are new, there appear to be several opportunities to adapt known solutions to address these problems. Section 5 discusses specific confidentiality concerns for pub-sub systems. Section 6 considers accountability and billing issues. Section 7 explores denial-of-service vulnerabilities particular to pub-sub systems.

4 Generic issues

Some security issues in pub-sub systems are not unlike those that appear in other distributed systems that cross administrative domains. In some cases, existing approaches can be adopted to achieve these goals, often with only minor modification. We discuss those cases below.

Authentication. Authentication establishes the identity of the originator of an action. In pub-sub, we consider two flavors of authentication, *end-to-end* and *point-to-point*. End-to-end authentication in this context means that if subscriber *A* receives a message claiming to have originated from publisher *B*, *A* can verify that *B* is indeed the publisher of the message. Point-to-point authentication is concerned only with the immediate end points of a communication: if *A* receives a message from *B*, *A* can verify that *B* is indeed the sender of the message where *A* and *B* can be publishers, subscribers or network servers.

End-to-end authentication can be implemented outside of the pub-sub domain. If a PKI exists independent of the pub-sub network, end-to-end authentication can be accomplished by having publishers sign messages using their private keys. The subscribers can then verify a publisher's identity by verifying the digital signatures attached to the message. The signing and verification operations occur outside of the pub-sub domain, and they can be administered independently.

If the pub-sub infrastructure is trusted, end-to-end authentication can be replaced with point-to-point authentication. Point-to-point authentication is a well-understood practice, and standard techniques should apply here.

There have been instances of pub-sub systems implemented using Opengroup's Distributed Computing Environment (DCE) [18] and its security features [4][21][24]. The potential size of an Internet-scale pub-sub system may give rise to scalability problems for DCE that are not present in smaller-scale systems. The pros and cons of using DCE (and other existing technologies) to outfit a pub-sub system need to be investigated closely before more informed assessment can be made.

Information integrity. The standard means to provide information integrity is by using digital signatures. A digital signature, when signed on the message digest with the sender's private key, provides two pieces of evidence: a) the message content has not been changed since it is signed, and b) the message indeed originated from the sender.

The provision of digital signatures can be largely independent of the pub-sub infrastructure. Consider again using a PKI for publishers and subscribers. Message integrity can be enforced by having the sender digitally sign every outgoing message. The establishment and management of the PKI can be performed independently of the pub-sub layer.

Subscription integrity. In a pub-sub system, the user subscriptions kept by the network form the basis for routing and forwarding — therefore the subscriptions must be protected from unauthorized modifications. This is a traditional access-control issue that can be solved with traditional means providing proper authentication and rights management. For most realistic cases, it is reasonable to assume that subscribers can trust the pub-sub infrastructure to implement subscription functions without malice.

Service integrity. Integrity of the pub-sub service can be put at risk if malicious faults arise at the infrastructure level (e.g., infrastructure hosts are compromised). A malicious server can insert bogus subscriptions and act as a bogus subscriber to neighboring servers. Moreover, it can ignore the routing algorithm entirely and route messages to arbitrary destinations or drop them completely.

Protecting the pub-sub network from malicious intrusions is not unlike protection of other large networks. However, if the infrastructure is compromised, pub-sub systems present new research questions regarding mechanisms that preserve services in the presence of malicious infrastructure servers. This topic is investigated in depth in the following sections.

We note that this is not solely a security architecture issue. For example, one can design the routing algorithm in such a manner that there is never a single route between any pair of publisher and subscriber, and that each message is routed on multiple routes to its destination. At the price of increased resource consumption, this

mechanism ensures a high probability that a message will be delivered to its intended parties despite a small number of malicious servers.

To begin tackling the problem of service integrity in the presence of malicious infrastructure hosts, a comprehensive fault analysis is needed in which the malicious faults are enumerated and consequences examined. We can then begin to understand the extent to which the existing infrastructure may be able to tolerate such malicious faults and subsequently design mechanisms to increase this tolerance.

User anonymity: User anonymity in pub-sub can be achieved with various anonymizing techniques developed for distributed systems [22][25]. It is also worth noting that the pub-sub routing and forwarding mechanism can be used as a lightweight anonymity tool. For example, in the Siena system the publications travel along a shortest path from the publisher to the subscribers [8]. Because of the way the routing mechanism works, a pub-sub server in Siena only knows its immediate predecessor and successor in the path. End-point anonymity is preserved in any path that has more than two hops. It is possible that with some strengthening, the pub-sub routing and forwarding algorithm can be used as a full-fledged anonymity tool without introducing much extra cost.

5 Confidentiality

Pub-sub systems introduce three novel confidentiality issues:

- Can the infrastructure perform content-based routing, without the publishers trusting the infrastructure with the content? (Information confidentiality)
- Can subscribers obtain dynamic, content-based data without revealing their subscription functions to the publishers or infrastructure? (Subscription confidentiality)
- Can publishers control which subscribers may receive particular publications? (Publication confidentiality)

Each of these poses new problems, but there appear to be opportunities to adapt well-known approaches towards satisfactory solutions.

Information confidentiality. When information being published contains sensitive content, publishers and subscribers may wish to keep information secret from the pub-sub infrastructure. This is especially important in a large pub-sub system where information may travel through network segments that are not necessarily trusted. Recall the resume circulation example from Section 2. It is conceivable that suppliers of resumes may wish to keep the resume content private from the routing infrastructure—an untrustworthy infrastructure server may copy every resume that routes through it and then sell them for a profit.

The requirement of confidentiality against the infrastructure is in a fundamental conflict with the pub-sub model. By definition, the pub-sub network routes information based on dynamic evaluations of information content against user subscriptions. Keeping the information private from the routing hosts may hinder such evaluations and hence routing. In particular, routing and forwarding optimizations such as the ones performed by Gryphon [3] and Siena [8] will be impossible to carry out if the infrastructure hosts do not have access to the information content.

Further note that the legitimate receivers of the information (i.e., the subscribers) must be able to read the information content, which may require an out-of-band agreement among the publishers and the subscribers so that the subscribers can recover the content of the publication (such as using a key). Incorporating an out-of-band key distribution or a similar scheme takes away the benefits of the basic pub-sub model—it follows a point-to-point communication model rather than the many-to-many model in a true publish-subscribe system.

A potentially promising technique in providing information confidentiality against the infrastructure is *computing with encrypted data* [1][13]. In general, a function f can be computed with encrypted data (a.k.a: f is encryptable) if there exists two functions E and D such that

- E and D are two polynomial time algorithms
- E maps x to an encrypted instance y
- D maps $f(y)$ to $f(x)$
- Nothing about x is revealed by y except what is implied by the result of $f(y)$

Abadi *et al*. proved that all functions in ZPP[1] are encryptable [1]. In other words, there exist E and D for every polynomial-time boolean function such that the data can be hidden from the function evaluator. However, a protocol between the publishers and subscribers is still needed so that E and D can be agreed upon and computed accordingly. We pointed out earlier that this conflicts with the many-to-many communication model. In addition, these secure computation protocols are often computationally intensive and require a large amount of communication overhead, which could be prohibitively expensive to carry out.

Subscription confidentiality. User subscriptions can reveal sensitive information about the user, in which case the subscriber may wish to keep the subscriptions private. Consider the human resource resume example. An HR person, upon being told that her company is starting a top-

[1] The class ZPP consists of boolean functions that can be computed in polynomial time with zero probability of error.

secret new project, wants to enter a new subscription that allows her to receive resumes with a particular background. Because of the sensitive nature of the project, she may wish to keep her subscription private even from the pub-sub system—after all, the system may turn around and sell this knowledge to her competitors.

More formally, subscription confidentiality against the infrastructure can be viewed as follows:

The subscriber *S* would like the network *N* to compute $f(x)$ without revealing *f* to *N*. Here, *x* is the publication information and *f* is the subscription function.

A closely related topic of interest to subscription confidentiality is *secure circuit evaluation* [2], which has been studied in various models [4][9][27]. Secure circuit evaluation hides the circuit[2] from the circuit evaluator. In theory, if computing with encrypted data can be achieved, hiding the circuit can be implemented as encoding the circuit itself as an input to a universal circuit evaluation function [2]. In practice, however, it is difficult and often impractical to encode the function as an input to a universal circuit; the proposed schemes often involve an expensive protocol.

Another related subject of interest is *Private Information Retrieval* (PIR) [11][12]. PIR mechanisms allow a user to retrieve records from a database, and in the meantime, hide what she retrieves from the database. Studies on PIR schemes showed that PIR is at least as hard as Oblivious Transfer [12], which implies the existence of one-way functions. A close examination of subscription confidentiality suggests that close relations exist between PIR and subscription confidentiality. For example, one can easily construct a PIR mechanism using a black box that implements subscription confidentiality simply by inputting every database record as a publication into the black box. Conversely, a simple case of a subscription function that matches a finite number of pre-defined strings can be reduced to PIR with publications modeled as databases and subscriptions as PIR queries.

The construction of PIR from subscription confidentiality suggests that the latter cannot be achieved using weak computational primitives—it is at least as hard as PIR schemes. A more general reduction from subscription confidentiality to PIR can be the starting point of constructing realistic confidentiality mechanisms to hide user subscriptions from the pub-sub infrastructure. However, all PIR-based schemes move some filtering operations from the database to the user, which implies more communication overhead as well as user-side computation load. Therefore challenges regarding performance and efficiency will still remain even if a general reduction can be constructed.

It is worth noting that the combination of information and subscription confidentiality against the infrastructure achieves a fairly strong level of user privacy—the most other people can deduce is that you are associated with this particular pub-sub system, however, they will not be able to find out how you are using the service (e.g., publishing what or subscribing to what). This can be a viable alternative to straightforward user anonymity.

Publication confidentiality. In many pub-sub applications, publishers do not know and perhaps do not care to know the identity of the subscribers who receive their information. In those applications, there is no need for subscription control — anybody can subscribe to anything. In other applications, however, it is important that publications be kept secret from ones who are not legitimate subscribers (the concept of legitimacy should be application specific). Consider the stock quote example. For billing purposes, quotes should be sent to paying customers only. Therefore they must be kept confidential from other users.

Publication confidentiality can be handled independent of the pub-sub infrastructure. For example, the publisher can distribute a group key to the subscribers using some out-of-band channel and encrypt the information content with the key. This ensures that only the subscribers with the right key can read the message. The drawback of this scheme is obvious: setting up a group key a priori, in essence, transforms the communication model into a traditional multicast model, and therefore minimizes the benefits of publish and subscribe.

Alternatively, publishers can trust the infrastructure to maintain publication confidentiality. For example, instead of registering with the stock quote provider, a user can register with the pub-sub system for the stock quote service. Publishers enter the quotes into the system without knowing who will be receiving them. It is then up to the pub-sub system to ensure that only registered users receive the appropriate quotes.

A potential solution here is to let the application choose an appropriate mechanism. That is, applications that do not care about publication confidentiality should not have to pay the cost associated with exerting confidentiality control. Meanwhile, for applications that desire publication confidentiality, the pub-sub layer must provide a) an interface for the application to specify a control policy, and b) a mechanism that supports such policies. Designing such a flexible security framework is no trivial undertaking—the interface must be expressive and easy to use, and the system must be prepared to carry out a whole spectrum of mechanisms desired by the applications. There is also the question of whether to implement an inexpensive policy as the default case—the default case can always be overwritten, but the specifics of a default policy must be carefully laid out so that it does not detract from the system flexibility.

[2] Circuit here means a function that can be represented as a binary circuit.

6 Accountability

In commercial pub-sub applications, publishers may want to charge subscribers for the information they provide. The nature of the pub-sub system, however, means there may be no direct relationship between a publisher and subscriber. Further, a publisher has no way of knowing which subscribers receive (and should be charged for) particular datagrams.

Out-of-band solutions. The most obvious solution to accountability is for publishers to bill subscribers by selling keys that decrypt selected data, which is similar to the publication confidentiality discussed earlier. Again consider the stock quote example. A subscriber could pay the supplier of stock quotes a monthly fee for the relevant keys. Publishers would send quotes into the pub-sub network encrypted with the appropriate key so that only subscribers who had paid for that information would be able to decrypt it. This ensures only paying subscribers will be able to view the information, but sacrifices many of the advantages of a pub-sub network. It requires subscribers to reveal their identity and interests to publishers, and demands a direct publisher-subscriber relationship. Further, it eliminates the possibility of per-data payment schemes and dynamic subscriptions.

Infrastructure-based accountability. If both subscribers and publishers trust the pub-sub infrastructure to account fairly, the pub-sub infrastructure can bill subscribers according to the amount of information they receive and pay publishers according to the information they provide without there being any direct relationship between publishers and subscribers. In the stock quote application, the infrastructure would keep track of who received what quotes at what frequency and who publishes them. Periodically the system would bill the subscribers and send a portion of the payment to the appropriate publishers.

This type of account management can be carried out at the entry points of the system where the network interacts with the publishers and the subscribers. An important issue here is for the system to demonstrate that its accounting is conducted in a fair way (see the discussion of auditability below.)

In addition to user account management, accountability can also extends to pub-sub servers when the message routing network spans distinct domains that include possibly competing and mutually suspecting organizations. In these cases, auditing needs to take place at the interfacing points of the various subsystems.

The various issues with accountability in pub-sub remain much the same as those in other distributed systems. The techniques in these other systems, such as the market-based pricing account management scheme in [20], can be adopted here (see [6][20]).

Auditability. If the pub-sub infrastructure is used to bill subscribers and reward publishers, publishers may wish to audit the accounting to verify that they are being paid fairly for the subscriptions they satisfy. Complete auditability is impossible if we wish to provide subscription confidentiality. Auditing based on statistical sampling at trusted points within the pub-sub network could provide a satisfactory solution, however. A publisher could examine logs from points in the network and compare the number of datagrams present to the fees collected from the pub-sub infrastructure. This would give the publisher a confident lower bound on the number of subscriptions satisfied. The pub-sub infrastructure could still cheat when a single datagram satisfies many subscriptions. However, if the log points were located throughout the network, it would be difficult for the pub-sub infrastructure to conduct any large-scale cheating without being detected. There is a natural tension between auditability (which improves as log points are moved closer to subscribers) and subscriber confidentiality (which is compromised by logging close to subscribers).

Another potential scheme for auditing is the use of verifiable secure computation [15]. If the accounting operation is viewed as a function and the logs are an input to the function, verifiable secure computation techniques allow the result of the accounting to be verified without disclosing the inputs (logs) to the function.

7 Availability

As in other communication systems, denial-of-service attacks remain as a significant risk for pub-sub systems. In addition to the standard infrastructure attacks to which all distributed applications are vulnerable (as discussed in Section 4), pub-sub systems open up some new classes of attacks. In particular, malicious publications and subscriptions can be used to overload the system.

Denial-of-service attacks are impossible to prevent in the general case. However, certain measures can be taken to minimize the probability of a wide spread denial-of-service attack. We discuss three different measures here. From the simplest to the most sophisticated, each scheme results in a certain level of publication bandwidth control.

Limited publication. This is a straightforward measure that simply limits the size of each publication [25]. Variations of this scheme may limit the frequency of publication or both size and frequency.

CPU-cycle-based payment scheme. This is a more sophisticated scheme that combines payment with publication control. *Hash Cash* is one such scheme [17]. The basic idea behind Hash Cash and other CPU cycle based payment schemes is that it requires the publisher to perform some complex computational tasks (e.g. finding collisions in the hash function) before publishing. The more complex the task, the more control the system has over the publication process.

Customized publication control. The primary drawback of the previous two schemes is that they are both one-size-fits-all solutions and therefore lack the flexibility of differentiating among different publications. A customized publication control does just that; it allows subscribers to specify which publisher is allowed to publish information. Bogus publications can be weeded out at the system entry points. Consider the scenario in which a subscriber wishes to accept notifications from publishers who know a certain secret. This criterion can be expressed as part of the subscription. In addition to the normal subscription function, the subscriber specifies a challenge clause, which the servers use to challenge the sender of the publications. More specifically, consider two functions, f and g, with the following two properties:

- f and g are hard to invert,
- $g(f(x), f(y)) = f(g(x,y))$

The publish and subscribe protocol behaves as follows: When subscriber A initiates a challenged subscription, A establishes a filter that includes g as the challenge function.

- preprocessing step: All legitimate publishers know a secret function f, which is distributed in an out-of-band method.
- subscriber → network: {subscription, g}
- publisher → network: $\{x, f(x)\}$, x is a random number chosen by the publisher.
- network → publisher: $\{y, g(x, y)\}$, y is a random number chosen by the network and g is the challenge function in the subscription.
- publisher → network: $\{f(y), f(g(x,y))\}$
- the network server can now compute $g(f(x), f(y))$ and compare that with $f(g(x,y))$. The server allows publication only if the two match.

In the last step, the network server verifies that the publication indeed should be forwarded to the subscribers by engaging in a challenge-and-response protocol with the publisher. With this scheme, bogus notifications will not generate unnecessary message traffic within the network. Note that this scheme is very similar to a public-key authentication mechanism. The only difference is that we might be able to find two functions f and g that are more efficient than the public key operations. A more detailed description of such an authentication mechanism can be found in [26].

We note that subscribers can specify any arbitrary function in place of the challenge-and-response example. Also note that such a function can be easily incorporated as an extension to the subscription semantics—basically as an extension to the language to allow the specification of an extra clause. The more complex the function, the more expressive the subscription language needs to be, which will further constrain the types of optimization the network is able to perform in terms of routing and forwarding. The tradeoff between the power of publication control versus amenability for optimization must be examined if such mechanism is to be adopted.

8 Summary

This paper presents a general discussion of the security issues and requirements in an Internet-scale pub-sub system. The dynamic content-based routing mechanism in such networks poses both opportunities and challenges for security. The nature of pub-sub systems present several apparent security paradoxes, among them routing based on content while keeping the content confidential, routing based on subscriptions without revealing the subscription functions, accounting based on subscriptions satisfied without revealing the subscription functions. We have not presented full solutions to any of these problems, but have suggested approaches based on new applications of well-known mechanisms that may lead to solutions.

We intend this paper to be an initial roadmap of establishing a comprehensive security architecture for large-scale pub-sub systems. These systems raise numerous interesting and important security issues for further research.

9 Acknowledgements

The authors thank John Knight for discussions that helped shaping some of the ideas here.

The work of Chenxi Wang was completed when she was a research scientist at the University of Virginia. Her work was supported in part by the Defense Advanced Research Agency, under the agreement number F30602-96-1-0314.

The work of David Evans was supported by in part by the National Science Foundation and NASA under agreement numbers NSF CCR-0092945 and NASA NRC 99-LaRC-4.

The work of A. Carzaniga and A.L. Wolf was supported in part by the Defense Advanced Research Projects Agency, Air Force Research Laboratory, Space and Naval Warfare System Center, and Army Research Office under agreement numbers F30602-01-1-0503, F30602-00-2-0608, N66001-00-1-8945, and DAAD19-01-1-0484.

The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Defense Advanced Research

Projects Agency, Air Force Research Laboratory, Space and Naval Warfare System Center, Army Research Office, National Science Foundation, NASA or the U.S. Government.

10 References

[1]. M. Abadi, J. Feigenbaum, and J. Kilian. "On Hiding Information from an Oracle". In the proceedings of the Ninth Annual ACM Conference on Theory of Computing. May, 1987. New York, pages 195-203.

[2]. M. Abadi and J. Feigenbaum. "Secure Circuit Evaluation". Journal of Cryptology, Vol 2. No. 1: pages 1-12. 1990.

[3]. M. Aguilera, R. Strom, D. Sturman, M. Astley, and T. Chandra. "Matching Events in a Content-based Subscription System". In the conference proceedings of the Principles of Distributed Computing, 1999.

[4]. M. Ben-Or, S. Goldwasser, and A. Wigderson. "Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Compuation," Proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pages 1-10.

[5]. B. Blakley. "CORBA security". Addison-wesley. August 1999.

[6]. S. Brands. "Untraceable off-line cash in wallets with observers", In Advances in Cryptology: Proceedings of Crypto'93, pages 302-318. Springer-Verlag (LNCS773), 1993.

[7]. A. Carzaniga, D. Rosenblum and A. Wolf. "Achieving Scalability and Expressiveness in an Internet-Scale Event notification service". In Proceedings of the 2000 ACM Conference of PODC 2000. Portland Oregon. Pages 219 –227. 2000.

[8]. A. Carzaniga, D. Rosenblum and A. Wolf. "Design and Evaluation of a Wide-Area Event Notification Service". ACM Transactions on Computer Systems, 19(3):332-383, Aug 2001.

[9]. S. Chapin, C. Wang, W. Wulf, and A. Grimshaw. "A New Security Model for Distributed Meta Systems". In Future Generations of Computer Science. October 1998.

[10]. D. Chaum, C. Crepeau, and I. Damgard. "Multiparty Unconditionally Secure Protocols," In the proceedings of the 20th Annual ACM Symposium on Theory of Computing, 1988, pages 11-19.

[11]. B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. "Private Information Retrieval". In the proceedings of the 36th IEEE Conference on the Foundations of Computer Science (FOCS). Pages 41-50. October 1995.

[12]. G. Di Crescenzo, T. Malkin, and R. Ostrovsky. "Single-database private information retrieval implies oblivious transfer". In Advances in Cryptology—EUROCRYPT'00.

[13]. J. Feigenbaum. "Encrypting problem instances, or…, Can you take advantage of someone without having to trust him?" In the proceedings of Crypto' 85, Springer-Verlag, 1986, pages 477-488.

[14]. M. Franklin and M. Yung. "Secure and Efficient Off-line Digital Money". In the proceedings of the Annual International Colloquium on Automata, Langaues and Programming. 1993.

[15]. R. Gennaro, S. Micali, "Verifiable Secret Sharing as Secure Computation", Eurocrypt '95, LNCS 921, Springer-Verlag, Berlin 1995, pages 168-182.

[16]. Gnutella. http://www.gnutellanews.com.

[17]. Hash Cash. http://www.cypherspace.org/-~adam/hashcash.

[18]. Opengroup. Distributed Computing Environment. http://www.opengroup.org/dce.

[19]. D. Rosenblum and A. Wolf. A design framework for Internet-scale event observation and notification. In Proceedings of the Sixth European Software Engineering Conference. LNCS, number 1301, pages 344-360. Springer-Verlag, 1997.

[20]. J. Sairamesh, D. Ferguson, and Y. Yemini, "An Approach to Pricing, Optimal Allocation, and Quality of Service Provisioning in High-Speed Networks", In the Proceedings of INFOCOM'95.

[21]. Bill Segall and David Arnold. "Elvin has left the building:
A publish/subscribe notification service with quenching". In the Proceedings of AUUG '97, Brisbane, Australia, September 1997.

[22]. P.F. Syverson, D.M. Goldschlag, and M.G. Reed. "Anonymous connections and onion routing". In Proceedings of the 1997 IEEE Symposium on Security and Privacy, May 1997.

[23]. Transarc, IBM white paper. June 1998. http://www.transarc.ibm.com/library

[24]. Talarian middleware whitepaper. http://www.talarian.com

[25]. M. Waldman, A. Rubin, L. Cranor. "Publius, A robust, tamper-evident, censorship-resistant web publishing system". In the proceedings of the 9th USENIX Security Symposium. August, 2000.

[26]. W. Wulf, A. Yasinsac, K. Oliver, and R. Peri. "Remote Authentication without prior shared knowledge". In the proceedings of the Network and Distributed Systems Security Conference, February 1994. San Diego, California.

[27]. A. Yao. "How to generate and exchange secrets," Proceedings of the 27th Annual IEEE Symposium on Foundations of Computer Science, 1986, pages 162-167.