The Strengths of Weaker Identities: Opportunistic Personas[*]

Mark Allman, Christian Kreibich, Vern Paxson, Robin Sommer, Nicholas Weaver
International Computer Science Institute

Abstract

Cryptographic security mechanisms often assume that keys or certificates are strongly tied to a party's identity. This requirement can in practice impose a high bar on making effective use of the cryptographic protections, because securing the coupling between credentials and actual identity can prove to be an arduous process. We frame a more relaxed form of identity, termed *opportunistic personas*, that works by *(i)* generating cryptographic credentials on an as-needed basis, *(ii)* associating credentials not with a user per se but instead as a link to past behavior by the same actor, as a means to inform future interactions, and *(iii)* managing these credentials *implicitly* in an opportunistic fashion. Using three real-world examples, we illustrate the benefits this unorthodox approach to identity management can yield.

1 Introduction

While cryptographic algorithms can provide strong protections in terms of authentication, integrity and privacy, the security mechanisms built from them often assume that any keys or certificates are also strongly tied to a party's identity. For these mechanisms, bootstrapping the identity in the first place, and then soundly managing it in the future, both present significant hurdles for practical use. To avoid these difficulties, some schemes have evolved that use "opportunistic" methods that bypass the need for establishing a solid notion of identity within the confines of the particular scheme. For instance, while IPsec [6] relies on knowing the identities of hosts, "anonymous" IPsec [8] does not require pre-shared keys, but rather generates them on-the-fly.[1]

The general observation is that while a hard-and-fast notion of another party's identity is desirable, we can often make use of other cryptographic properties (e.g., data integrity and on-the-wire privacy) without it. In addition, for some applications it is less important to know who the peer is in a solid way than to know if that peer changed during the course of interacting with it (e.g., to protect against cookie theft or hijacking attacks).

As used in practice, SSH [9] often side-steps the question of airtight identity to offer more secure data transport. When started for the first time, the SSH daemon usually generates a host key for a given machine. The first time a user connects to the given server, they are presented with the server's public key fingerprint for verification. In principle, users should contact the server administrator out-of-band to obtain the fingerprint they should expect. In practice, users generally simply accept the fingerprint—in part because often they are operating in a broader context (such as having just been told by the remote administrator that their account is now ready) that gives them high confidence that the risk at this point of accepting a bogus key is quite low. In subsequent connections, the user's client compares the given server key with their cached copy and warns the user if these do not match. Two key notions we distill from this are *(i)* the opportunistic generation of keys and *(ii)* the rough notion of server identity embodied in how this process works in practice (i.e., that keys are not rigorously vetted by users on the first connection).

In a different domain, Hu and colleagues [5] present a scheme for constructing a PKI to secure BGP routing based on different parties simply assuming that the first public key they receive for a particular autonomous system is likely valid. Their analysis argues that such an approach can yield quite positive deployment incentives with fairly minimal risks regarding credential hijacking going undetected for very long.

In this paper we advocate for generalizing the styles of dealing with identity illustrated by these examples. We term the overall notion "opportunistic personas" to reflect *(i)* the opportunistic generation of cryptographic keys, and *(ii)* the loose notion of identity these keys con-

[*]USENIX Workshop on Hot Topics in Security, August 2007.

[1]Often authentication of identity happens at another layer in the process—such as with a username and password.

vey. While personas do not provide a strong form of identity, we argue they can gain significant benefits in terms of greater ease-of-use, which in some contexts can exceed the utility of attaining stronger forms of identity. Further, we speculate about using opportunistic personas to bootstrap developing solid and general purpose notions of identity (see § 2.3).

In addition to the opportunistic generation of personas, we also argue for the idea that user actions can often be interpreted as an implicit proxy for *management* of personas. Users tend to respond differently to legitimate versus unwanted activity. By observing user reactions, we can then in some cases *infer* the user's trust in particular personas. For example, user reactions could drive the construction of white- and black-lists (see § 2.1).

Certainly, in terms of achieving secure and sound systems, learning personas in an informal fashion is in principle not as desirable as manually exchanging and validating keys. However, in a number of contexts such validation has proven *impractical*: it is either beyond the ability of most users, or at least beyond their "pain threshold" for the perceived benefits [4]. Thus, we believe application and protocol developers will benefit if they broaden their thinking to consider weaker forms of identity may actually provide stronger practical security.

The remainder of this paper progresses as follows. In § 2 we outline several examples where use of opportunistic personas could mitigate real-world problems. § 3 offers some guidelines on using opportunistic personas. Finally, we conclude in § 4.

2 Examples Of Employing Personas

In this section we present several examples of applications for which the notion of opportunistic personas could enhance security.

2.1 Reducing False Positives in Email Filters

Mail filters are used to sift out a large variety of unwanted email—everything from malware to spam to phishing attacks. While some filters can be quite accurate, the process inevitably flags legitimate messages as junk. Even high accuracy ratios will inevitably falsely finger legitimate emails. Also, as the overall rate of incoming email increases, the interval between wrongly catching legitimate emails decreases. Depending on the user's setup, this flagging can cause anything from a simple indication in the mail reader to the message being put in a side ("junk") folder to the outright deletion of the message. For some, a false positive can be much more costly than the stream of unsolicited mail itself. One way to help combat the problem of false positives is by developing positive reputations of trusted senders whose messages can then be safely whitelisted. However, such approaches usually founder on the fact that (i) email addresses are easy to spoof[2], (ii) developing cryptographic identities is difficult (per the discussion in the previous section), and (iii) the inconvenience to the user of then populating their whitelist with the identities of the different legitimate senders. We propose an approach based on the notion of opportunistic personas that can address these concerns. Our goal is not to offer better filtering of unwanted email, but rather to prevent legitimate mail from getting caught in the maze of traps already deployed.

To use the idea of opportunistic personas, a user's mailer would generate a public/private key pair[3] when first configured. This key pair would then be used to consistently sign the user's messages. The public half of the key would be included in the messages themselves. A crucial simplifying aspect of our scheme is that disseminating the sender's key and using it to sign a message are both done *without* user intervention. There is no attempt to explicitly associate a key with a user, and thus having a user's mailer sign each of their messages with the key does not provide a deep degree of authentication. Rather, it serves to indicate that whoever sent a message is the same person as who sent previous messages signed with that same key.

Each email recipient, in turn, *automatically* develops their own view of the reputation associated with a sender's particular public key *based on whether the recipient has found that, in the past, mail signed by that key is legitimate*. This "reputation" is deduced *implicitly* and in a quite simple fashion: when a user takes some action that indicates an incoming mail is legitimate (e.g., replying to it), then their mail reader adds the sender's public key to a whitelist.[4] Before applying filtering to signed incoming email, the reader (or MTA upstream; see below) first checks whether the signature is valid and corresponds to a public key in the whitelist. If so, then the reader can bypass its normal filtering, eliminating the chance of a false positive.[5] Essentially, the scheme allows automatic construction of reliable whitelists.

The approach sketched above offers a great degree of incremental deployability: senders using a modified mailer would have their mail consistently signed without

[2]For instance, if source addresses are to be believed, then one of the authors of this paper regularly sends spam to another author.

[3]Generation could proceed without a passphrase, or take the user's existing POP3/IMAP password as the passphrase.

[4]While we believe that most users would not want to be bothered, the interface to this whitelisting of personas could in principle be exposed to direct user command, as well.

[5]The procedure could also work such that all messages are first checked to see if they are unwanted and only those that are unwanted are run through the signature verifier. Our goal in this paper is to not dwell on the particulars, but to concentrate on the high-level concepts.

needing the user to take any explicit steps, and receivers would benefit from having senders whose mail they find useful automatically whitelisted in the future.

One key aspect that requires development is how to share these whitelists, as filtering is often done by a mail server and not by the user's own machine. Therefore, the observations from the user will have to be conveyed to the server in order to short-circuit its spam filtering. This could be done cheaply by having the mailer fire off a periodic automated email containing the whitelisted keys to some general address within the organization that automatically installs such lists in the site's mail processing framework.

Of course, users' lists of whitelisted keys could also be shared with each other. However, there are two problems with doing so. First, sharing keyrings among actual users likely means that the users would have to be involved, which is at odds of with the entire scheme of doing things opportunistically without involving the user. Automated schemes for sharing whitelists may be possible, but may also allow for attackers of the system to slip their keys into the whitelists. Second, sharing whitelists is not likely to gain users much protection from false positives. Since most filters only have a very small fraction of false positives, it seems likely that a key will be judged as legitimate and whitelisted fairly readily by each user. On the other hand, an area for experimentation would be an organization coalescing its users whitelists together when processing email on a central server. In this case, users are not bothered by the process and may derive some marginal benefit.

We note that attackers cannot in general impersonate legitimate senders, since they lack their private keys, and cannot benefit from minting their own public/private key pairs, since recipients will not find this mail worthwhile and thus their mailers will not learn to whitelist the associated key. If an attacker does obtain a private key (say as part of a more general compromise of a host), the private key will get them little leverage, because the number of users who will have whitelisted this key will be small. Furthermore, if a recipient receives a piece of mail from a whitelisted sending key, their mailer can observe the user filing the message in a "junk" folder and remove the associated key from the whitelist.[6]

Another tempting thought is to use opportunistic personas to encrypt (rather than just sign) email. The problem with this idea is that the opportunistic keys are generated on a per-mailer basis. Therefore, if a user reads their mail on both a desktop machine and a laptop they will send mail signed with two keys. Therefore, when someone wishes to encrypt an email they will have to encrypt the email to both keys. While this is not difficult once all the keys are known, the problem is that a sender does not have any good way to know when it possesses all the keys for a recipient. Using encryption when not all of the recipient's keys are known will lead to the user not being able to read the message in some mail clients they use.

This last bit suggests that we may be able to use protocol extensions to help with on-the-fly use of cryptography. For instance, we could think about extending IMAP [2] such that the IMAP server can hold a user's keys and a single key can be used wherever a user reads their mail. While this is more complicated than the simple scheme sketched above, it could be done in a way that remains transparent to the users.

2.2 Web Transactions

Our next example application of opportunistic personas involves web browsing. We look towards *helping* solve the issues surrounding phishing, where users are tricked into logging into a fake site that appears to be legitimate. The user's credentials are logged and the attacker replays these to masquerade as the user and login to the legitimate site. In this paper we will use the example of a thief accessing a user's bank account (even though alternate forms also exist). Both users and banks have an interest in thwarting this form of attack and therefore we consider using the notion of opportunistic personas in both directions of web transactions.

On the server side, a bank's web site could set its security posture regarding the legitimacy of a request based on the origin of the request *with respect to previous requests*. For example, if some user always logged in from some given IP address, an attempt to log in from a different IP address could raise a flag leading to the bank taking a more skeptical posture—say, by querying the user, in addition to username and password, with additional "security questions" (e.g., the tired-but-common queries regarding birth place, dog's name, ending balance on last statement, etc.) before granting access; or by increasing monitoring of that account's transactions.

Such an implementation could be approximated by a server cookie the client includes in its request, or by using the client's IP address. Bank of America's SiteKey [1] is an example of a cookie-based approach. However, using cookies or IP addresses only loosely couples the origin of the request with a user (e.g., IP addresses can be ephemeral for a given computer and cookies replicated and replayed). Our notion of opportunistic personas can provide a tighter coupling. The user's web browser would first generate a key pair and then use the private key to sign all of its requests. It would also include the public key with the requests, and the web server

[6]The mailer could even flag this change in the behavior associated with the key, such that the recipient might in turn inform the sender that their key appears to have been stolen.

would then record the public key included with the user's initial account creation. The heightened security posture could then be instituted when a username/password comes signed with a different key (or no signature at all). Since the browser does not have to share the secret key with anyone and only the secret key can be used to sign requests that will correctly correlate with the username/password this use of opportunistic personas provides a tighter origin hint than IP addresses or cookies.

On the other side of things, web servers could provide clients with a notion of their identity by including public keys and signatures with their transmission. SSL already provides this functionality. However rather than just verifying that a server's certificate is signed by a known authority, the clients would also generate a factual history of web sites they have visited. This would allow a web browser to warn the user when submitting credentials to a site they had not previously visited, similar to the approach taken by SSH.

An additional use of opportunistic web server personas could be in the creation of user passwords. Ross and colleagues [7] detail a scheme called PwdHash that makes custom and secure passwords for users for each web site they visit. The general idea is that when registering for some web service the browser creates a password by hashing the user-entered password and the DNS name of the server. Therefore, if users use the same passwords at multiple sites they will in fact end up different for each site since the DNS name of each site is a component of the password. Further, a phisher will not be able to coax the right password from a user because even if the user enters their password locally the browser will turn it into a hash that will be wrong because the phisher has a different DNS name than the legitimate service.

Consider instead the case where the user's effective password is constructed from one that they entered plus the web server's public key as provided to the client during the registration process. When the user subsequently needs to provide a password, it can be generated from the given key ID and the user-entered password (assuming the request from the server is properly signed with the given key ID). This is similar to PwdHash, except we factor out reliance on the DNS, which yields considerable flexibility. For instance, merging banks could continue signing their messages with appropriate keys such that browsers generate the correct passwords even though the banks no longer actively us their old DNS names. In other words, the web server adopts a cryptographic persona that only depends on a key pair it generates—not on the overall information delivery process.

If web servers would sign their content on a per-message level (rather than at the transport-layer as SSL does), browsers (users) could verify that they receive legitimate content even if they retrieve the content from a cache (either local, or a distributed cache like Coral [3]), or from a content distribution network (CDN). Sites could, e.g., serve signed index pages and then offload the remainder of the page transmission to a CDN or proxy. As long as all objects are signed with the same key, the browser (user) can trust that all objects came from the same origin. This again fosters the notion that securing data itself is of prime importance, given the myriad ways data can be transmitted.

2.3 Persona Promotion

In the previous two cases we have used opportunistic personas to determine a rough sense of identity for a particular purpose. In this section we go a step further—into the realm of linking a user's actual identity with a given opportunistic persona, which can then be used for a variety of purposes beyond what opportunistic personas allow. We feel that while the idea of *persona promotion* is speculative, it potentially represents a significant advance in that it provides for *some* concreteness in terms of cryptographic identity, without requiring the use of one of the more heavyweight schemes which non-technical users can hardly cope with [4]).

We start with two examples of applications that revolve around *personal contacts*: audio/video conferencing (à la Skype) and calendaring. As above, these applications can opportunistically generate keys and use them to sign the content they generate (A/V streams and meeting invitations/RSVPs, respectively). Since both of these applications involve personal contact it may be possible to promote the personas to actual identities. This could be done explicitly, with the calendar tool asking the user whether the people who RSVPed were in fact in the meeting or with the A/V tool asking the user to indicate if they know the person with whom they are conversing. More implicitly, we could also use the signal of carrying on a conversation for a certain length of time to indicate that the other party is well-known and we can then assume the identity in the "buddy list" is accurate.[7] In both cases the indication would trigger the user to sign the peer's key to indicate trust in the given identity.

The developed notion of identity could be be exported to share trust across applications. It would be fairly straightforward for the calendar or A/V tool to simply export keys and new signatures to a personal or even a public key server. Key servers are already prevalent and this would in turn make the keys broadly available. Other applications could then also interface with these key servers to learn about promoted personas.

[7] Of course, if such techniques were well-known there could be obvious social engineering tricks to try to coax people to stay on the line for the requisite amount of time.

A vulnerability of the calendar application is an attacker intercepting a meeting invitation, forging a response based on their own key and also inserting the meeting on the intended recipient's calendar such that they attend the meeting and have their supposed key validated by the other attendees. In this case, the attacker will have a key that has been signed as belonging to some third party. This problem only manifests itself before a user has verified an invitee, which means the window of opportunity does not make this a likely avenue for attack. A danger for A/V applications is that computers may have multiple users who share an account which is under one person's name, such as when a family computer uses only Mom's Skype identifier. In this case, the persona will not precisely map to an individual. If the A/V client is explicitly interacting with the user this may be reasonably easy to handle, but the implicit notion of observing a conversation will perhaps associate the wrong identity with the key.[8]

One logistical problem with the overall notion of persona promotion is that personal contact applications may not derive much benefit from this operation. Hence, there may be little incentive for application developers to help establish a hard-and-fast notion of identity in a general fashion. Two paths we envision could help with this problem. First, if a personal contact tool provides a "plugin" architecture then it is possible that independent developers could add the functionality sketched in this section without relying on the application developer. A second possibility is that of a tool that is part of a larger suite of tools offered by some particular vendor (e.g., iCal and iChat within OSX). In this case, the developer may have incentive to add this feature to a personal contact tool to benefit other tools (e.g., a mail reader) in their suite.

A second problem is that the people someone sets up meetings with or talks to likely represent only a fraction of the people one interacts with on a regular basis. This seemingly imposes a limit on the usefulness of the approach. However, we note that the people one meets with and talks to are likely to represent the set of people with whom it is the most important to securely communicate (e.g., by encrypting a confidential memo or sharing an internal budget). In addition, as with PGP, a web of trust can be formed such that one can gain confidence in identities even in the absence of personal contact.

3 Discussion

From the examples in the previous section we can distill a set of guidelines for using opportunistic personas.

First, consideration of the use of personas versus highly vetted cryptographic identities for various applications must always be framed in terms of trade-offs. The point of personas is to extract strength (often the benefit of greater ease-of-use, and hence practical deployability) by giving up strength elsewhere (the weaker tie to an actual identity). Thus, such uses should explicitly identify the benefits to gain, where a key facet concerns an understanding of the scope within which an opportunistic persona is relevant.

Second, not only does employment of personas weaken ties to identities, but doing so also potentially opens up new avenues of attacks. To this end, one needs to assess the implications of compromise of a given persona key: How does a particular use of an opportunistic identity change the security equation in such circumstances? Even if keys remain protected, one must give thought to possibly increased opportunities for man-in-the-middle attacks (since the two endpoints of a conversation may have less confidence that they can soundly share a session key) or social engineering (attackers leveraging implicit decision-making regarding what trust to place in personas in order to trick users into taking actions that will falsely reflect trust; essentially, a form of "mimicry" attack).

Finally, one needs to examine what additional work might be placed on users, versus the user-interaction simplifications that occur when employing opportunistic personas. This includes an assessment of what sort of user behaviors to leverage for implicit management of persona trust; opportunities for introducing ties between multiple personas; and the degree to which users can fruitfully develop correct intuitions and expectations regarding how personas behave, analogous to how users understand the properties of personas in alternate domains (such as "caller ID").

4 Summary

We have sketched *opportunistic personas*—i.e., broadly using a rough notion of cryptographic identity that can be generated as-needed. While a persona is not as airtight as a traditional identity in cryptographic systems, we show several examples whereby personas mitigate problems even if they do not solve the entire problem. In addition, while the underlying notion has been used before (e.g., in SSH), we have attempted to distill general concepts and show how personas can offer power beyond securing data transport. We have aimed to illustrate the potential power of the scheme sufficiently for developers to consider the trade-offs of its possible application to their protocol and application design efforts. We hope that the community will join in with further exploring such ideas.

[8] On the other hand, if the persona is known to be a "group persona" and is promoted to a "group key," the implicit scheme may still be fine.

5 Acknowledgments

We thank Ethan Blanton for discussions involving these issues. This work was supported in part by the National Science Foundation under grants ITR/ANI-0205519, NSF-0433702 and CNS-0626539, for which we are grateful. This work was also supported by a fellowship within the Postdoc-Programme of the German Academic Exchange Service (DAAD). Parts of this work were produced under the auspices of the Institute for Information Infrastructure Protection (I3P) research program. The I3P is managed by Dartmouth College, and supported under Award number 2003-TK-TX-0003 from the U.S. Department of Homeland Security, Science and Technology Directorate. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security, the Science and Technology Directorate, the National Science Foundation, the I3P, or Dartmouth College.

References

[1] SiteKey at Bank of America. `http://www.bankofamerica.com/privacy/sitekey/`.

[2] M. Crispin. Internet Message Access Protocol - Version 4rev1, March 2003. RFC 3501.

[3] Michael Freedman, Eric Freudenthal, and David Mazieres. Democratizing Content Publication with Coral. In *Proc. USENIX/ACM Symposium on Networked Systems Design and Implementation*, March 2004.

[4] Peter Gutmann. Plug-and-Play PKI: A PKI Your Mother Can Use. In *Proc. USENIX Security Symposium*, 2003.

[5] Yih-Chun Hu, David McGrew, Adrian Perrig, Brian Weis, and Dan Wendlandt. (R)Evolutionary Bootstrapping of a Global PKI for Securing BGP. In *Proc. ACM HotNets*, 2006.

[6] S. Kent and K. Seo. Security Architecture for the Internet Protocol, December 2005. RFC 4301.

[7] Blake Ross, Collin Jackson, Nicholas Miyake, Dan Boneh, and John C. Mitchell. Stronger Password Authentication Using Browser Extensions. In *Proc. USENIX Security Symposium*, 2005.

[8] J. Touch, D. Black, and Y. Wang. Problem and Applicability Statement for Better Than Nothing Security (BTNS), February 2007. Intenet-Draft draft-ietf-btns-prob-and-applic-05.txt (work in progress).

[9] T. Ylonen and C. Lonvick. The Secure Shell (SSH) Protocol Architecture, January 2006. RFC 4251.