Recovering High-Value Secrets with SGX and Social Authentication

Nathan Malkin, Serge Egelman, David Wagner
University of California, Berkeley
{nmalkin, egelman, daw}@cs.berkeley.edu

ABSTRACT

While passwords are frequently forgotten, the methods available today for recovering them suffer from poor security, poor usability, or both. One promising technique is social authentication, in which selected peers vouch for one's identity. However, like many recovery methods, this approach requires that the provider has full access to the underlying data. We propose a novel recovery mechanism that prevents a data breach even if a host system is compromised by relying on hardware features of Intel SGX. Our method for social authentication also includes new defenses against both software-level and social engineering attackers.

1. INTRODUCTION

The widespread availability of encryption means that most computer users today have the capability to secure their confidential data — but only if they don't forget their password. Almost all cryptosystems depend on a secret or private key, and most implementations seek to prevent unauthorized usage by encrypting it with one more key — typically, the user's password.

Unfortunately, passwords, strong and weak alike, are frequently forgotten. This is so commonplace and expected that just about every system makes provisions for this by implementing a recovery mechanism, allowing users an alternate way to prove their identity. For example, a website may send an email to a previously-verified address, a bank may ask for personal information, and an IT desk can verify an employee's ID.

However, all recovery mechanisms suffer from the same limitation: the provider has access to all of the user's data. If data in an account is encrypted, and only the user has the private key, the provider can't restore access. This problem affects many real-world systems, such as full-disk encryption keys, master passwords for password managers, SSH private keys, and PGP secret keys. Setting up recovery for any of these would require sharing the keys with another party, which may then result in the secret being compromised. However, recent advancements in processor technology offer a potential way forward.

Symposium on Usable Privacy and Security (SOUPS) 2016, June 22–24, 2016, Denver, Colorado.

2. SGX: SECURING THE SECRETS

2.1 SGX overview

Intel's Software Guard Extensions (SGX) are a set of instructions and memory changes that add trusted computing functionality to the Intel processor architecture [5]. Chips with this technology started appearing in consumer devices in 2015.

Under the SGX threat model, an attacker can gain operating-system-level privileges on a machine, but will be prevented from reading or tampering with the contents of protected memory, including the code being executed.

An important feature of SGX is remote attestation [3]. The platform is able to prove to anyone that a specific piece of software is executing in the secure enclave. The proof is a cryptographically signed hash of the secure container's contents. The attestation key signing the statement is unique to each processor and can be verified against an endorsement certificate issued by Intel.

2.2 SGX and secure recovery

SGX can address our goals for secure recovery by lifting the requirement that we trust the provider — though it assumes trust in the hardware manufacturer.

Secure code, running in an enclave on a server, can generate a keypair and publicize the public key. A user, on their own machine, can encrypt their secret (such as a PGP private key or a master password) with this key and send it back to the server. Since the keypair was generated in a secure enclave using tamper-proof memory, only the trusted code will be able to decrypt the contents. As part of this process, the client can use attestation to verify that it is interacting with code it trusts running in the enclave.

Now that we've securely delivered a secret to an SGX-enabled system, we need some way of recovering it if the user forgets their password. Consider, as a strawman, a system where we recover a password with another password. This approach can easily be implemented with SGX. The code running in the enclave can verify a password supplied by a user and reveal the secret if the password is correct — and under no other circumstances.

Since our threat model includes server compromise, we would in practice require some additional safeguards for how the password and secret are handled. Rather than supplying the password directly, the client would encrypt it, along with its own public key, using the enclave's public key. After validating the password, the enclave would encrypt the secret it has been storing with the public key provided by the client, thus ensuring that only the user will have access to it.

This system, as described, allows for securely recovering secrets even from an untrusted host. But of course, entering a password to recover another password is hardly a usable solution. What are our alternatives?

3. SOCIAL AUTHENTICATION

3.1 Considering the alternatives

One of the popular approaches to account recovery is the use of security questions (e.g., what was your favorite color in sixth grade?). Since the answers must be produced verbatim, the implementation of this method is nearly identical to the password-based recovery in our strawman strategy. However, research and analysis of existing deployments has shown that security questions exhibit both poor security and poor usability [1, 6]. Users have a hard time remembering their answers, but they are easily guessed by attackers due to having a small search space or being available in public records.

Another approach is authentication with a second factor — something you have — for example by sending a code to the user's email or phone. While more usable, this method has security limitations, as it entails trusting the communication channel as well as the endpoint (e.g., the provider of the email account). While the former can be, under some circumstances, mitigated (such as by requiring TLS connections and terminating them within the enclave), the latter is unavoidable. This is especially a concern because email accounts are relatively easy to compromise (for example, due to frequent password reuse).

3.2 Introducing social authentication

One authentication method that has received relatively little attention is social authentication, the notion of your identity being verified by people you know. Though this idea is fundamentally embedded in many human systems, it was first introduced to the realm of authentication in 2006 by Brainard et al. [2]. Under this approach, users who can't access the system seek out a previously designated "helper," who verifies their identity (by face-to-face conversation or a phone call) and vouches for them by providing a specially issued "vouch-code" obtained from the server. The asker can then present the code to the server to obtain access.

Subsequently, Schechter et al. extended this approach to apply to email accounts and conducted the first usability study authentication [7]. They found that most people were able to recover their accounts successfully and that the system repelled a high percentage of attacks that used forged email addresses. (Insider attacks, however, were more successful.) Since then, social authentication has seen wide-scale deployment as an account recovery feature on Facebook, where it is known as "trusted contacts" [4].

3.3 Applying social authentication

By combining social authentication with the SGX technology described above, we can achieve a system with desirable security and usability properties. We propose some additional enhancements for this technique.

Authenticating helpers

In addition to authenticating helpers by email, we can identify and authenticate them by implementing OAuth for social networks and other popular providers (e.g., Facebook, Google, Twitter, and Yahoo). In addition to a more streamlined user experience — helpers don't need to switch windows and wait for an email — this improves security, as communication between the secure enclave and the API will be done over TLS (which will not necessarily work with all email providers).

Requiring multiple helpers

To reduce the risk from a helper being compromised, or a provider acting maliciously, we suggest requiring multiple helpers for the recovery process. A user-defined threshold would determine the minimum number of vouches for access.

Preventing social engineering

While warning and instructions were found by Schechter et al. to be relatively effective, we propose strengthening the defenses against social engineering attacks by requiring the asker and helper to communicate in a manner mediated by the recovery server. One very effective option is to set up a video call, but this requires the two parties to be online and available at the same time. Instead, we propose that the asker records a video requesting assistance, which is then played to the helpers. To prevent replay attacks, the asker is required to say a unique code for each recovery attempt.

4. RESEARCH QUESTIONS

Our research agenda includes implementing the system described above and conducting user tests to address questions such as the following:

- Is this system usable? Will users be willing to tolerate the delays inherent in social authentication?
- Will people trust the security of this system, or will they prefer more familiar recovery techniques? Is user education required and, if so, how can it be done most effectively?
- What is the minimum number of helpers required for this system to be effective?
- Who are the helpers? Do they actually need to be people who know the asker, or can we ask strangers to compare the recovery message with another prerecorded video?
- How vulnerable is this method to insider attacks? What other social engineering attacks are possible?
- If our threat model includes server compromise, how can we ensure that SGX delivers the promised confidentiality and integrity properties? (e.g., avoiding replay and side-channel attacks)

5. REFERENCES

[1] J. Bonneau, E. Bursztein, I. Caron, R. Jackson, and M. Williamson. Secrets, lies, and account recovery: Lessons from the use of personal knowledge questions at google. In *WWW'15 - Proceedings of the 22nd international conference on World Wide Web*, 2015.

[2] J. Brainard, A. Juels, M. Yung, R. L. Rivest, and M. Szydlo. Fourth factor authentication: Somebody you know. In *In Proc. of CCS*, 2006.

[3] V. Costan and S. Devadas. Intel sgx explained. Cryptology ePrint Archive, Report 2016/086, 2016.

[4] Facebook, Inc. Introducing trusted contacts. `https://www.facebook.com/notes/facebook-security/introducing-trusted-contacts/10151362774980766`, 2013.

[5] F. McKeen, I. Alexandrovich, A. Berenzon, C. V. Rozas, H. Shafi, V. Shanbhogue, and U. R. Savagaonkar. Innovative instructions and software model for isolated execution. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy*, HASP '13, pages 10:1–10:1, New York, NY, USA, 2013. ACM.

[6] S. Schechter, A. J. B. Brush, and S. Egelman. It's no secret. measuring the security and reliability of authentication via secret questions. In *In Proceedings of IEEE Symposium on Security and Privacy*, pages 375–390, 2009.

[7] S. Schechter, S. Egelman, and R. W. Reeder. It's not what you know, but who you know: A social approach to last-resort authentication. In *In CHI '09: Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems.* ACM, 2009.