WATSUP: Web Authentication without Sending or Storing User Passwords

Ryan Amos, Gregory Gundersen, Thomas Schaffner

Abstract

ABSTRACT HERE

1 Introduction

Internet users must entrust private information with many different companies, but online security is extremely difficult. Even large and technically advanced companies have lost sensitive information to data breaches and malicious attacks. For example, Yahoo suffered several major breaches in the past few years. First, in 2013 attackers perpetrated the largest recorded data breach when they stole roughly one billion Yahoo accounts. [5]

Then in 2014, a separate attack compromised roughly 500 million Yahoo accounts. [2] Neither of these breaches were discovered until 2016, meaning sensitive user information such as hashed passwords and security questions were stolen years before any user became aware.

These types of breaches are not unique to Yahoo. [3, 4] They exemplify a major issue with internet security: users must entrust their personal information to servers and companies that are not transparent, and sometimes not competent, in their implementation of security protocols. Most companies do not publish their full security practices, and even users who are responsible or knowledgeable about online security have no means to verify that their data is being handled correctly. Furthermore, the companies themselves are often unable to detect breaches promptly. As users register for an increasing number of services, the risks described above grow as well.

In spite of the difficulties and flaws of passwords, such as weak user passwords, password reuse, forgetting passwords, and constant breaches of improperly stored passwords, passwords are still unavoidable for users. Despite push from both users and security experts, alternatives rarely catch on, especially in web authentication [1]. Given that passwords have not been eliminated, we focus on improving the security of user passwords, rather than eliminating them entirely. We propose Web Authentication without Transmitting or Storing User Passwords (WATSUP), a new application-layer protocol that provides users with a transparent, consistent, and easy way to log into a number of different services without trusting any service to properly handle their login credentials. In this paper, we first discuss related work and its drawbacks; next, we propose the WATSUP protocol; third, we discuss a proof-of-concept implementation; and finally, we discuss the advantages and disadvantages of WATSUP and propose future work.

2 Background

2.1 Risks of using passwords

Verifying a user's identity is an important challenge in internet security. Passwords, despite their many drawbacks, are a critical and ubiquitous solution to the problem. Today, anyone who shops, banks, communicates, or socializes online creates multiple user accounts with a number of websites. But password usage has a number of risks, and existing solutions have disadvantages. The WATSUP protocol is designed to eliminate or mitigate as many of the most common risks mentioned below.

RISK: Eavesdropping on plaintext SOLUTION: HTTPS PROBLEM: Not always provided; user ignorance WATSUP: Encrypt the data even on HTTP

The most straightforward risk is that a user's password is captured in transit as plaintext. The most popular solution to eavesdropping is to use HTTP Secure (HTTPS), which tunnels an HTTP connection through a TLS connection. Unfortunately, while HTTPS is supported

by most large companies, it is still not universally used [TK: https://letsencrypt.org/2016/06/22/https-progress-june-2016.html]. In addition, many users may not realize that they should not use websites that do not support HTTPS.

RISK: Server being compromised SOLUTION: Hashing, salting PROBLEM: No guarantee the solution has been implemented WATSUP: Do not trust the server

Perhaps the biggest risk users face is that their passwords are compromised on a companyâĂŹs web server. As discussed in the introduction, these attacks have happened or been discovered as recently as last year, often at staggering scales. The most common methods to mitigate the effects of losing passwords is to salt and hash them. A cryptographic hash function is a one-way function for which it is easy to verify that an input is correct, but hard to extract the input from the output. A company should hash a new password before saving it in a database; to authenticate a user, it hashes the candidate password and compares it to the hashed password on record. âĂIJSaltingâĂİ is adding random information to each password. Both of these methods help prevent rainbow table attacks in which precomputed tables of common password hashed with typical hashing algorithms are used to decode hashed passwords. They also help prevent an attack reusing a compromised password on another website. Unfortunately, many popular websites do not properly handle users data. For example, in its official statement, Yahoo only confirmed that the âĂIJvast majorityâĂİ of passwords had been securely hashed before they were stolen [TK: https://yahoo.tumblr.com/post/150781911849/an-important-message-about-yahoo-user-security], implying some (out of a billion) were not. In the 2012(TK) LinkedIn data breach, LinkedIn failed to salt users passwords and only hashed them with the SHA-1 algorithm, which is known to be insufficiently secure as early as 2005 [TK: Wang et al.: `http://link.springer.com/chapter/10.1007/11535218_2` TK 2 more SHA-1 vulnerability pre 2012 TK linked in data breach].

RISK: Phishing SOLUTION: Hash password on hostname PROBLEM: Rainbow table attacks WATSUP: Hash the hostname and username

Another common means of compromising a userâĂŹs account is called âĂIJphishing.âĂİ In phishing, a user is asked to submit their username and password to a malicious website that poses as a legitimate one, typically through email request to reset a password from a domain posing as the legitimate service. One solution to this problem is to hash the password with the websiteâĂŹs hostname before sending it. If this is done consistently, phishing is prevented because the malicious website will have a different hostname. This allows for improved portability, since no secret data is stored, and no trusted third party is required [TK: http://www.cs.utexas.edu/ bwaters/publications/papers/www2005.pdf]. However this method is still vulnerable to weak passwords and poor server salting. For example, an attacker who has compromised a database of passwords that has been salted with the hostname and then hashed can still create a rainbow table for that website. WATSUPâĂŹs solution to this problem is to salt the base password with the hostname with the username. This prevents a rainbow table attack since the salt is unique for each user.

RISK: Replay attack SOLUTION: One-time passwords PROBLEM: Not typically implemented

Another common attack vector is called a âĂIJreplay attack.âĂİ In a replay attack, an adversary intercepts the user's authentication credentials and simply re-transmits them in a subsequent login to masquerade as the original user. To prevent this, each communication can use a nonce, which is a number that can only ever be used once. Typically, they are generated by a web server and sent to the user after an initial verification step. For example, Google provides dual authentication using nonces; but this service is only provided when the user logs in from an unknown device. WATSUPâĂŹs goal is to use OTPs for every log in without additional overhead for the user.

RISK: Password reuse SOLUTION: Use strong but unique passwords for different accounts PROBLEM: Cognitive overhead SOLUTION: Mitigate the effects of password reuse

Finally, passwords are insecure because remembering many strong passwords is hard, and users have a tendency to reuse passwords. Many systems have been designed to lessen the cognitive load for users. For example, password managers are popular tools to generate and save unique user passwords, reducing password reuse with-

out increasing a user's cognitive load. However, these password managers are frequently insecure [TK: https://www.usenix.org/system/files/conference/usenixsecurity14/sec14-paper-li-zhiwei.pdf]. Additionally, they often rely on a trusted, non-transparent third-party or are not portable. Importantly, a password manager does not prevent phishing or replay attacks. WATSUP combines the ease of use of a password manager without storing the userâĂŹs password, by providing a single, transparent version of source code, and still prevents phishing and replay attacks. Related work One solution to many of the above problems is Password Multiplier, which derives a site-specific password from a base password and the siteâĂŹs hostname. This uses SHA-1 as a key derivation function, and the hostname as a salt. Key derivation functions are functions that take the userâĂŹs base password and produce multiple distinct high entropy keys by using a distinct salt. If one derived key is compromised, the base password and all other keys remain secure. [TK http://www.cs.utexas.edu/ bwaters/publications/papers/www2005.pdf] This does not protect against replay attacks on the same site. If everyone were to use this and the server failed to adequately salt and hash the derived keys, then everyone would be using the same salt. In this scenario, a rainbow table attack on a per-hostname basis would be reasonable, particularly against large databases.

Another similar solution is a client-server web authentication protocol known as Secure, Quick, Reliable, Login (SQRL). It performs web authentication on a "something you have" model. SQRL uses a randomly generated master password to provide strong, derived logins on a per-site basis, almost completely eliminating user passwords and making phishing difficult. On a login request, a link or QR code is provided for input to the SQRL app, which may be on a different device. The SQRL app then completes the login. As a side effect, this also protects against phishing [TK: http://sqrl.pl] However, SQRL has a few issues. First, it requires its own protocol, and does not run over HTTP. This is a significant barrier to deployment. Second, it requires the rendering of QR codes. Finally, the code is implemented in assembly. While this was done so that the code is as clear as possible, e.g. the compiler cannot eliminate security-critical code, it makes incremental deployment difficult. In most respects, we argue that SQRL is simply more complicated than WATSUP without additional benefit. Implications We argue that any solution to the web authentication problem must resolve the following issues:

The solution should be portable. It should not require any data be stored or transmitted before it can be used on a new device. This means using passwords. Users should not have to trust servers. Users should be protected against replay and reuse attacks, even without HTTPS. The solution should allow for phased deployment by not interfering with standard login The solution should be as simple as possible for servers to implement, so that any developer can safely add it to their server, without risking user security

3 Design

4 Evaluation

5 Related Work

6 Conclusions

References

[1] P. C. v. O. Cormac Herley. A research agenda acknowledging the persistence of passwords. In *IEEE Security & Privacy Magazine*, Piscataway, NJ, Feb. 2011.

[2] S. Fiegerman. Yahoo says 500 million accounts stolen, 2016. URL http://money.cnn.com/2016/09/22/technology/yahoo-data-breach/. [Online; accessed 2016-01-12].

[3] S. Fiegerman. Plain text offenders, 2016. URL http://plaintextoffenders.com/. [Online; accessed 2016-01-12].

[4] T. Hunt. have i been pwned?, 2016. URL https://haveibeenpwned.com/. [Online; accessed 2016-01-12].

[5] S. Thielman. Yahoo hack: 1bn accounts compromised by biggest data breach in history, 2016. URL https://www.theguardian.com/technology/2016/dec/14/

yahoo-hack-security-of-one-billion-accounts-breached.
[Online; accessed 2016-01-12].