Zeropasswords

Patented

What's wrong with SAAS login?

Nikos Leoutsarakos

Tiny bio

Nikos has a Physics background and a M.Sc. in Computer science from McGill University in Montreal, Canada, where he lives with his wife and two children. He has been publishing and developing systems in the areas of cryptography, wireless/mobility and digital signatures since 1994.

Nikos can be contacted at NIKOS@ZEROPASSWORDS.COM

Prelude

You do not have to be a data scientist to read this paper. If you use a browser to visit your favorite Websites, meet your social media friends online, and check your email daily, this paper is for you.

Let's start by asking: "what happens when you use a third party to help you login to a Website?"

Companies listed above are part of a group of companies that promote a SAAS user authentication service to Websites for a fee (see table, column E[1]). SAAS login is implemented using a smartphone based PKI challenge-response protocol and Trust Tokens. The service consists of three software applications (see Table). The first one resides at the Website and executes the activities of column C. The second one resides on the service company's cloud (Authentication Cloud) and executes the activities of column E. The third one is a mobile app on subscriber's smartphone (column G).

While on the surface SAAS login seems to work, a closer analysis reveals a plethora of issues that rank from inconvenient to grounds for a law suit. I personally found 22 problems with SAAS login.

1. **Website[2] does not authenticate its users.** A third party does. Authentication clouds (companies of column E) aspire to become sentinels to Websites and authenticate users on their behalf. "All a Website has to do is outsource the login process to us for a fee", they advocate.
2. **Website must "obey" TrustTokens.** Should a positive TrustToken arrive in Step 5, Website has no choice in Step 6 but to login the username of Step 1 and expose his private content and sensitive information on the browser. Website relies exclusively on

[1] Processes employed by companies of column E may vary from one another but they all execute the steps described in the table above

[2] Website is indicative of online service, online game, remote hardware, etc., any computing system that controls access to it (not open to public)

the validity and trustworthiness of received TrustTokens. In cases of compromised TrustTokens, is a Website liable?

3. **Steps 3-5 are out of the control of Websites.** In other words, Website is agnostic to user authentication process(es) employed by companies in column E. In case of wrongful logins, is a Website liable?

	A.	B.	C.	D.	E.	F.	G.
	User (browser)	**Network** (Man-In-The-Middle)	**Website**	**Network** (Man-In-The-Middle)	**User Authentication Cloud**	**Network** (Man-In-The-Middle)	**User** (phone)
1.	User claims an identity	→ Send "username"					
2.			Translate Username to UserID	→ Send "UserID"			
3.					Choose a secret S (random or not) send it to UserID's phone and keep it	→ Send S	
4.						Send S' ←	Sign S with private key S' = Ppr(S)
5.				Send TrustToken (e.g. OAuth 2.0) ←	Retrieve UserID's public key and use it to verify Ppub(S') = S (?)		
6.	View Webpages	Send Webpages ←	Received TrustToken is proof that "username" was authenticated				

4. **Website must trust authenticator.** A malicious or compromised company of column E can get a user to sign "anything" in Step 3. User better have a different private key on his phone to sign contracts and approve money transfers! In cases of misused user private-keys, is a Website liable?
5. **Step 5 is a "single point of failure" for Websites.** What if TrustTokens stop coming? How will companies in column E compensate their customer-Websites for down time? They won't. They will ask Websites to have a second (backup) way to login. A dual (or multi) user authentication system will quickly get out of sync; suffer from

security vulnerabilities; and introduce user authentication conflicts which can be exploited by hackers.

6. **Users must register twice.** Users registered at a Website must also register with clouds of column E. This inherent stipulation causes a logistics nightmare for users, and a logistics and cost nightmare for Websites. Users must download all mobile apps of present and future companies-clouds of column E to have the freedom to login to any Website, and Websites must deploy, pay and be ready to interact with all present and future companies-clouds of column E if they do not want to disappoint users.
7. **Databases of Step 2 need be maintained by the Website.** As users register onto or unregister from a Website an automated process must be introduced to add or remove user entries from the databases of Step 2 kept at the Website (one database per company-cloud of column E). In case of wrongful logins, or inability to login, due to the fact that databases of Step 2 are out of sync with the actual database of registered users, is a Website liable?
8. **Websites cannot maintain login-transaction logs.** Outsourcing login to a cloud prevents Websites from interacting with their registered users directly. Consequently, it would be futile for Websites to keep track of login transactions which are based exclusively on TrustTokens. Such logs would not be admissible in court of law because they do not contain proofs of user identities, nor do they contain proofs of users' intent to login. In cases of unlawful logins Websites cannot defend themselves in court of law.
9. **User authentication clouds cannot maintain login transaction logs.** Clouds of column E serve many Websites and keeping track of who logged in, to which Website, and when, even if it is done discreetly, would be a direct violation of users' and Websites' privacy.
10. **Users have no proof.** Users have no record, receipt, or other type of proof in their phones, which would legally bind them and their smartphones to specific login transactions to specific Websites. In case of unlawful logins users cannot defend themselves in court of law.
11. **The authentication cloud can be bypassed.** A Man-In-The-Middle can replace the "TrustToken" in Step 5 or 6 to his benefit. The end result will be MIM controlling the decisions in Step 6 totally undetected. In other words, the user authentication Steps 3-5 executed by the authentication cloud become irrelevant.

12. **The authentication cloud can be compromised.** A Man-In-The-Middle can replace the "UserID" in Step 2 with his; grab a copy of S in Step 3; and replace S' in Step 4 with S signed with his private key. The end result will be MIM controlling the decisions in Step 5 totally undetected.
13. **Website's database or process can be compromised** in Step 2 so that one or more usernames translate to a malicious UserID. The end result will be UserID-imposter and his phone approving other usernames' requests to login, and thus gaining access to their private content and sensitive information.
14. **Cloud database or process can be compromised** in Step 3 so that communication is established with a malicious phone instead of UserIDs' smartphones. The end result will be an imposter and his phone approving login requests of other users, and thus gaining access to their private content and sensitive information.
15. **Call forwarding.** If the database in Step 3 contains phone numbers used to establish communication with phones, there is no need to compromise it. An imposter need only take a copy of the database and then take control by simply call-forwarding users' phone numbers to his. The end result will be imposter and his phone approving login requests of other users, and thus gaining access to their private content and sensitive information.
16. **The Achilles heel.** Verification in Step 5 compares S from Step 3 with S resulting from decrypting S' with public key in Step 5. An imposter can compromise the Boolean outcome of this comparison, and hence the generation of TrustTokens, to his benefit.
17. **Worldwide user authentication authority.** Companies of column E promote a SAAS model of user authentication. Each company strives to become a central worldwide user authentication authority. In practice, these companies have designed, implemented and deployed architectures where "central" means cloud and "worldwide authority" means authenticate the users of the world.
18. **Outsourcing requires trust.** Websites will need to trust one or more clouds of column E to authenticate users on its behalf. It is likely that for non-technical reasons a Chinese, Russian, or German Website for example, may not want an American company-cloud to have such power over its users.
19. **Single point of failure.** SAAS model of user authentication invites professional hackers because it pays. Worldwide centralized login will create new worthwhile targets

and attacks will be diverted from Websites to authentication clouds where hackers will attempt to compromise verification processes, network packets and the new enormous central databases. Except this time, instead of affecting the registered users of one Website hackers can cause problems to thousands of Websites and hundreds of millions of users.

20. **No favoritism.** The software application (or applications, one per cloud) that resides at the Website and executes Step 2 must be able to communicate securely with all present and future user authentication clouds. Websites have no choice but to constantly update this application if they want to provide to their users the freedom to choose which authentication cloud they want to be authenticated by. However this creates a logistics problem for Websites with users switching from cloud to cloud, new clouds appearing, old clouds disappearing, and each cloud stipulating its own communication requirements due to lack of standards. In addition, authentication clouds suffer from the flip side of this logistics problem. They need to make sure that millions of Websites have their latest software application installed and running.
21. **User nightmare.** The mobile application (or applications, one per cloud) which is running on user smartphones and executes Step 4 must be able to communicate securely with present and future user authentication clouds. Users have no choice but to constantly update their mobile application if they want to login to any Website on the Web. However, this creates a logistics problem for users who cannot be authenticated unless they go through the registration process of new clouds as they appear, or re-register with existing clouds if Websites demand it.
22. **Recall username.** In Step 1 user claims an identity. In practice, the user is asked to type in a username, or a code that he is known by at the Website. Consequently, username fatigue, i.e. the memory taxing task of recalling and entering the correct username at the correct Website remains user's responsibility.

Conclusion

Majority of the issues above are caused by the SAAS architecture of user authentication. In other words, outsourcing user authentication to a cloud is the problem. The very idea of outsourcing the most sensitive part of a Website, or of an online service, to a third

party is shown here to be the root of most objections listed above. Proliferation of SAAS login with PKI will face resistance from Websites and from the public, more for political and logistics reasons (see 17 to 22 above) and less because of its technology shortcomings and the way it works (see 2 to 16 above).

It is unavoidable that Websites which decided to outsource their login process will soon come to harsh realizations when unwanted or unlawful access transactions occur and digital and physical valuables are lost or stolen. Who is to be blamed then? Users will blame the Websites, the Websites will blame the authentication clouds and the clouds will blame Websites and/or users. Without access transaction logs and proofs of user approvals/disapprovals, it will be as hard, if not harder than it's been with passwords, to resolve disputes. At least with passwords all matters were between a Website and its registered users.

In a separate document we analyze a non-SAAS PKI login process, i.e. a process that does not involve a third party.