Stronger Authentication for Password Credential Internet Services

Todd Booth and Karl Andersson
Research@ToddBooth.Com Karl.Andersson@Ltu.Se
http://OrcId.Org/0000-0003-0593-1253 http://OrcId.Org/0000-0003-0244-3561
Division of Computer Science / Luleå University of Technology / 97187 Luleå, Sweden

***Abstract*—Most Web and other on-line service providers ("Internet Services") only support legacy ID (or email) and password (ID/PW) credential authentication. However, there are numerous vulnerabilities concerning ID/PW credentials. Scholars and the industry have proposed several improved security solutions, such as MFA, however most of the Internet Services have refused to adopt these solutions. Mobile phones are much more sensitive to these vulnerabilities (so this paper focuses on mobile phones). Many users take advantage of password managers, to keep track of all their Internet Service profiles. However, the Internet Service profiles found in password managers, are normally kept on the PC or mobile phone's disk, in an encrypted form. Our first contribution is a design guideline, whereby the Internet Service profiles never need to touch the client's disk. Most users would benefit, if they had the ability to use MFA, to login to a legacy Internet Service, which only supports ID/PW credential authentication. Our second contribution is a design guideline, whereby users can choose, for each legacy ID/PW Internet Service, which specific MFA they wish to use. We have also presenting conceptual design guidelines, showing that both of our contributions are minor changes to existing password managers, which can be implemented easily with low overhead.**

Keywords: Mobile Device Management; Security; Credentials; Android; Identification; Authentication; Password; Stormpath

I. Introduction

The acronyms, terms and definitions used in this paper are found below in table I.

Term	Definition
ID/PW	user-id and password credentials
IETF	Internet Engineering Task Force
Information Systems	Web and other Information Servers
JSON	JavaScript Object Notation
MFA	Multi-Factor Authentication
REST	REpresentational State Transfer
RFC	Request for Comments document
W3C	World Wide Web Consortium

TABLE I
Acronym and term definition table

A. Research Problem

Legacy ID/PW credentials are one of the weakest links in our security chain [1]. There are various solutions but on-line servers (Internet Services) often only use ID/PW credentials because most alternatives have been too complex and/or too expensive [2]. Academic research has been criticized as having failed to gain traction because researchers rarely consider real-world constraints [3]. The W3C's Web Authentication Working Group realizes this problem and their charter states, "Overall goals include obviating the use of shared secrets, i.e. passwords, as authentication credentials". It will be many years before most of the Internet Services support the replacement of legacy ID/PW only credentials, which makes this a very relevant security research problem. Our opinion is that there are low cost and simple API solutions, to greatly mitigate Internet Service authentication vulnerabilities, such as [4] and [5]. The main issue is that there is a common misconception that alternatives are too complex and expensive.

B. Contributions

Since mobile devices are more easily stolen, and more vulnerable to Internet Service credential attacks, this conference contribution is focused on mobile devices (even though our research can be generalized to tablets, PCs, and servers). Our contributions are generally, design guidelines and an experiment, showing how clients can use much stronger authentication, for Internet Services. Our specific contributions follow:

1) Design guideline where the Internet Service profiles (with ID/PW) never touch the client's disk and are erased from RAM, after just a few seconds.
2) Design guideline where a client can use MFA, to authenticate to an Internet Service, which only allows ID/PW credential authentication.

We have performed limited experiments concerning our two contributions, and the implementation is quite easy to understand and simple to code.

C. Outline of this Contribution

The rest of this paper is organized as follows. Our solution is developed and presented via cycles, where we incrementally

add functionality, to make this paper easier to understand, which is found in section 2. Our contribution experiment is presented, in section 3. We then show additional related work, in section 4. We wrap it up, with our conclusions and recommended future work, in section 5.

II. Design Cycles

Here we show the detailed design cycles that we went through, in the development of our contributions. We will show how to mitigate authentication vulnerabilities, from simple to complex, showing the next step, which will often introduce new vulnerabilities and issues, which we will then address.

Cycle 1 - Common or Similar Passwords

Users log in to numerous Internet Services, which often only provide for legacy single factor authentication (which are based on only ID/PW credentials). What many users do, is to use the same or similar password, on more than one, and perhaps on numerous Internet Services. They also sometimes use a simple password as the common password, so that it is easy to remember. One major vulnerability is that if the Internet Service password is compromised, the bad guys can use that same or similar password on other popular services, to comprise other accounts controlled by the same user. An advantage of a common password is that the user does not need to store it on the disk.

Cycle 2 - Password Manager

Mitigation of a single common password is achieved by using very different and better yet, long random ID/PW's, on each Internet Service. However, users can't remember many different and/or random credentials so they need to store these, perhaps in a file on their computer. There are web browsers, plug-ins, and applications which are specifically designed to store all the Internet Services' credentials that a user has created. These are called password managers, many of which were compared by PC Magazine [6]. One example of a password manager, which we will refer to as an example in this paper, is LastPass [7]. So, password managers are a recommended partial solution to the research problem.

LastPass includes many features, including the following: automatic login to Internet Service, analysis to find weak and/or duplicated passwords, the ability to automatically change passwords, and the ability to synchronize the Internet Services profiles between devices and the cloud. LastPass encrypts all the Internet Service profiles. To gain access to the clear-text service profile, the user must first authenticate, to LastPass.

Password managers introduce new vulnerabilities since all passwords are stored on the device's storage. If the bad guys can get the password manager data file(s) and obtain the password manager's global master password, then they have access to all the credentials. This is a very common pattern which we present in this paper, where the mitigation of one vulnerability introduces a new vulnerability (which is often a lower risk). To present our contribution, we will start with the functionality of LastPass, and then describe the required enhancements, to meet our solution requirements.

Cycle 3 - Cloud Based Password Manager

LastPass keeps a copy of the Information Service profiles both locally on the client and in the cloud. There are vulnerabilities, when keeping a local copy of the Internet Service profiles on the local disk.

Our First Main Contribution:

Our first main contribution is the following design guideline. We will eliminate the local disk copy of Internet Service profile attacks, by never writing the Information Service profiles, to the local disk. We will be left with only the Information Service profiles, in the cloud. Our proposed workflow to register and login to an Internet Service follow:

Proposed new *registration workflow*: For registration of the Internet Service, the user registers with the Internet Service as normal, with LastPass. As is done today, LastPass then asks if the user wishes to save the Internet Service profile. If so, LastPass would normally save the profile locally on the disk and sync to the cloud.

Our proposed change, is for LastPass to only save the encrypted profile to the LastPass cloud service. LastPass then erases the Internet Service Profile from RAM. This way, the Internet Service profile never touches the client's disk during registration and the profile is only in RAM for a few seconds. The RAM copy of the password can of course be copied for a few seconds. So even though we have eliminated some attacks and mitigated others, we have not completely eliminated all attacks.

Proposed new *authentication workflow*: For authentication to the Internet Service, the client requests the local LastPass process, to login to a given Information Service. The local LastPass process obtains a limited subset of the Internet Service profile, from the LastPass cloud service (without credentials). Based on the related LastPass service profile, the client is requested to authenticate (in the manner specified in the profile). If approved, the LastPass cloud service returns the entire encrypted Internet Service profile to the local LastPass process. The local LastPass process decrypts the profile and then automatically logs the client in, to that Internet Service. The local LastPass process then erases the Internet Service profile from RAM. This way, the Internet Service profile never touches the client's disk during login and the profile is only in RAM for a few seconds. Thus some vulnerabilities are completely eliminated and others are only mitigated.

Cycle 4 - Unique Credentials and MFA per Internet Service

With the existing LastPass design, the client must authenticate to LastPass, after which, the client has access to all Internet Service profiles. I.E., LastPass is unable to require different (for example MFA), for some profiles, but not others. Note: We

are aware that LastPass allows users to have multiple identities, however each identity can authenticate with the same master credentials (which is not what we propose).

Our Second Main Contribution:

Our second main contribution is the following design guideline. We will also allow the end user, to require different authentications for different Internet Service profiles. Let's take the following example. The user has the following Internet Service profiles, which only allow for legacy ID/PW authentication.

1) Their financial account. Let's suppose that they have a minimum of 10,000 USD in this account.
2) Their favorite Web based news and weather services
3) A USA bank (many USA banks only require legacy ID/PW authentication, which may be surprising in other countries) Let's suppose that they only have a maximum of 100 USD in this account
4) Their employer's Web based email service

Even through the above Internet Services only support ID/PW authentication, let's assume that the user wishes to use the following authentication, prior to the user being able to retrieve the ID/PW credentials, from the LastPass cloud service:

1) Their financial account (minimum of 10,000) USD - user wishes to use MFA via the Google Authenticator app on their personal phone
2) Their favorite ten Web based news and weather services - user wishes to use Social login authentication via FaceBook
3) The USA bank (maximum of 100 USD) - user wishes to use normal ID/PW authentication (via the LastPass master password)
4) Their employer's Web based email service - user wishes to use MFA via an SMS sent to their work phone

With LastPass and other password managers, this is simply not possible to do. With our design guideline contribution, the above is very simple to do. Our proposed new workflow follows:

During the initial financial service registration (or during modification), LastPass simply needs to add an option, where the user can specify the required client to LastPass authentication, to retrieve the credentials.

For example, when the user registers or changes their financial account profile, the user should be able to state that they wish to use MFA via the Google Authenticator app. Then, during attempted login, when the user tries to retrieve the financial account ID/PW from the LastPass cloud service, the service simply requires MFA via the Google Authenticator app. Only upon Google Authenticator app success, does the cloud service send the encrypted ID/PW credentials to the client, which then decrypts and performs an automatic login. As stated before, after login, the client then immediately erases the ID/PW credentials from RAM.

Likewise, when the user registers their ten Web based news and weather services, the user should be able to specify that the user must provide the FaceBook social login credentials. The other Internet Service required authentication works in a very similar way.

Now, even if an attacker completely compromises the LastPass client application, when they try to access the financial account, they will not be provided with the ID/PW credentials unless they approve, via the Google Authenticator app. If they try to modify the financial account permissions, they will also need to approve, via the Google Authenticator smart-phone app.

With the above, if there is a malicious software on the client device, which tries to access or change the financial profile information, the real user will receive a request to approve the access, via the Google Authenticator app., which the user should reject. Of course, our contribution will not stop a brute force attack, directly against the Internet Services, but at least we have mitigated some other attacks, in a very simple low cost way.

III. Experiment

We focused on a low cost and a very easy to implement solution, which has strong real-world applicability. Our efforts were to develop the implementation, concerning just the additional features that a password manager, such as LastPass, would require. LastPass already has most of the required client side and cloud based service functionality.

We created the additional required client and server code, via two different client/server authentication APIs, Stormpath [8] and Auth0 [4]. However, the APIs were designed for an Internet Service, which would have many users and our requirement is totally different. Our requirement is that a given cloud service instance is for clients, each of which will have credentials for many Internet Services. Nevertheless, the API were still quite helpful for our experiments.

By using the Stormpath API [8], we can then use any of the following, to enhance legacy ID/PW only credential Internet Services: OAuth token, social login integration, two-factor authentication via SMS, Google Authenticator, AD/LDAP integration, and SAML support. By using the Auth0 [4], we can use many of the same authentication methods and Apple TouchID.

We will now discuss our experiment with Stormpath and Auth0. The components involved, when a user wishes to access a given service (for example a bank Service X) are the following: 1) The client wishing access to Service X. 2) The Stormpath or Auth0 service. 3) The destination Service X.

Stormpath and Auth0 have many API frameworks, which all work in a similar manner. We are experienced with C, C++ and C#, and we decided to implement our experimental server side code in C# (where the server is also actually a client towards the service). It was very easy to understand the Stormpath and Auth0 C# .NET APIs. In under an hour, we did the following: Our code, which used the Stormpath and Auth0 services, would be executed on the LastPass Cloud Service. On the end user's client, no additional new code is required, since we can use existing Web browsers, authenticator apps (such as Google Authenticator)., SMS messages, etc.

1) We created server side Stormpath and Auth0 Cloud Service applications (located at the providers)
2) Created our own server side Cloud Service (located in the Internet)
3) From our Cloud Service, we dynamically connected to the Cloud Service
4) From our Cloud Service, we retrieved the required application information (for example, how to connect to the cloud service, for our specific accounts)
5) As part of the registration work-flow, we created one of the user's Web based news profiles, where the user wished to use a Social Login integration. Again, the Web based service only allows for ID/PW authentication, but before the user can retrieve this information from our cloud service, the user must login, via a Social login integration. As an example, we chose that the user must login to Facebook, to obtain their ID/PW credentials (for the news site).
6) As part of the login work-flow, on our Cloud Service, we also retrieved the user account information and profile
7) Stormpath allows users to store 10 MB of profile information, per client, so this can be used to save the credential requirements, per Internet Service
8) Auth0 had an interesting feature named "WebTask", which allows extensibility in Auth0's multi-tenant platform. It is simply to run code on private WebTask containers via simple web requests. This allows, for example, password managers to define their own code to run on Auth0's multi-tenant platform, in a fairly secure manner. This would allow the password manager to define MTA authentication in ways, other than Auth0. For example, the user must perform MFA via the Google Authenticator application and their smart-phone GPS must be within 1 kilometer of their normal work location.

Classic id/pw credential registration: The classic password credential registration protocol, is shown in figure 1.

With our solution, after successful registration, the client encrypts the site information (including ID/PW), and then stores that information into the Stormpath Cloud Service (with our additional code). As part of the site information, the client has full control to configure the required authentication information (to retrieve the information from the Cloud Service).

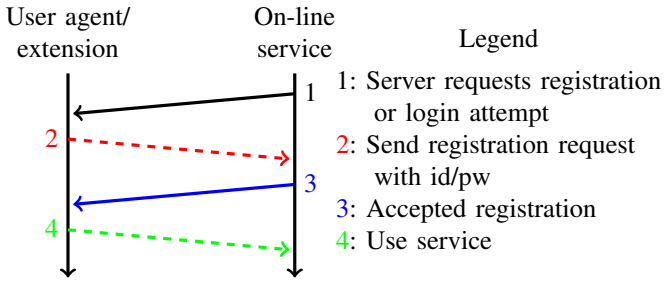

Fig. 1. Classic userid/password registration

For example, to access the client's financial site information, the client can configure the Cloud Service to require the use of the Google authentication application to retrieve the site profile information (including ID/PW).

Now, when the client wishes to login to the financial site, they first request the site information (including ID/PW) from the Stormpath Cloud Service. The Service detects that MFA with the Google authentication application is required. So, the Service pushes a request to the client's Google authentication application. The client then clicks on their smart-phone Google authenticator app. to approve, after which the service sends the site profile information to the client. The client decrypts the information and then sends the information to the financial site. Then the client erases the site information from RAM. Again, the site information never touches the client's hard disk.

IV. Related Work

We first present the related work and for some of these we provide our specific comments. Our comments are always preceded by "Comments:"

A. Related work - summary and our response

In [2], Lindemann concludes that after 40 years of IT innovations, passwords are still the most widely used authentication method, which are inherently insecure. Lindemann then explains how secure hardware in conjunction with a generic protocol can help overcoming today's authentication challenges and how this protocol can be used as a solid basis for federation. Comments: Lindemann's solution requires changes on the on-line service while our solution does not.

In [9], Zissis, et. al. describes the massive migration to cloud which requires cloud based authentication methods. They evaluate cloud security vulnerabilities by identifying the requirement of unique security requirements. They go on to present a viable solution that eliminates these potential threats, via the use of a Trusted Third Party, whereby the third party auditor (TPA) will verify the integrity of data state changes. Comments: Our solution is a viable improved cloud based authentication solution, which can be used by all cloud based solutions.

In [10], Soares, et. al. state that the emergence of mobile computing also makes authentication a priority, and has been reinforcing the need to build stronger and more resilient mechanisms; and simultaneously providing the means to develop new authentication mechanisms, namely Multi-Factor Authentication (MFA) schemes. They go on to state that the degradation of the security of password-based mechanisms, combined with the increasing number of perils on the Internet, is rendering one-factor authentication outdated. Comments: Our solution provides a higher-level framework, which can be used with all types of authentication approaches.

In [11], Castiglione, et. al. provides an innovative blind signature-based approach to multi service authentication. This allows several attacks to be mitigated, related to users' privacy, while at the same time, allowing users to have much more control over numerous customized digital identities. They state that users should be free to choose their own identity provider per specific security and privacy requirements/policies as well as to their personal trust relationship with the involved organization. Comments: Our paper continues their work, via providing a framework, which can be used to implement their solution (via small incremental changes to cloud based password managers).

In [12], Castiglione, et. al. show that hierarchical access may represent a limitation in many real-world cases. They then go on to propose a novel model that generalizes the conventional hierarchical access control paradigm, by extending it to certain additional sets of qualified users. Finally, they then apply their newly created model and propose two constructions for hierarchical key assignment schemes, which are provably secure with respect to key indistinguishability. Comments: Our solution can use their proposed solution, between the client and our Password Management Service.

In [13], we (Tesfay, Andersson and Booth) described a cloud and reputation based security model for Android phones. Comment: Our new solution can implement that cloud and reputation based security model, in our proposed Password Management Service.

V. Conclusion

We have proposed two novel contributions, which only requires a minimal of new functionality to existing password manager applications. Our specific contributions follow (repeated):

1) Design guideline where the Internet Service profiles (with ID/PW) never touch the client's disk and are erased from RAM, after just a few seconds.
2) Design guideline where a client can use MFA, to authenticate to a legal Internet Service, which only allows ID/PW credential authentication.

We have created conceptual design guidelines, showing how easy it is to add much of the newly required functionality. Our novel solution eliminates some vulnerabilities and mitigates others.

The validity of this work was a proof of concept, for a very specific use case. Future work is to generalize this, so that the same concepts are applicable and verified, for a much wider set of use cases.

We can also add more functionality, as to what are the user's requested conditions for authentication. For example, on a Internet Service basis, the user can require that the request must come from a pre-defined block of IP addresses (ex: home IP address or work IP network). Another example is that the user can require that their own smart-phone must be in a special location (work or home via GPS). We plan to contact a password manager vendor and propose that they implement our contributions.

References

[1] D. Ferbrache, "Passwords are broken – the future shape of biometrics," *ScienceDirect, Elsevier*, vol. 2016, no. 3, pp. 5–7, Mar. 2016.

[2] R. Lindemann, "Not built on sand - How modern authentication complements federation," in *Series of the Gesellschaft fur Informatik (GI)*, ser. Lecture Notes in Informatics (LNI), Proceedings - Series of the Gesellschaft fur Informatik (GI), vol. P-223. Gesellschaft fur Informatik (GI), 2013, pp. 164–168.

[3] J. Bonneau, C. Herley, P. Van Oorschot, and F. Stajano, "The quest to replace passwords a framework for comparative evaluation of web authentication schemes," in *33rd IEEE Symposium on Security and Privacy*, 2012, pp. 553–567.

[4] Auth0, ""Identity infrastructure, built for developers", https://auth0.com, [Online: Accessed on March 26th, 2016]."

[5] Stormpath, ""User Identity API For Developers", https://stormpath.com/, [Online: Access on March 26th, 2016]."

[6] PCMag, ""The Best Password Managers for 2016",http://www.pcmag.com/article2/0,2817,2407168,00.asp%7d, [Online: Accessed on April 3rd, 2016]."

[7] "LastPass | Password Manager, Auto Form Filler, Random Password Generator & Secure Digital Wallet App." [Online]. Available: https://www.lastpass.com/

[8] Stormpath, "Identity & User Management API | Stormpath."

[9] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Generation Computer Systems*, vol. 28, no. 3, pp. 583–592, 2012.

[10] L. Soares, D. Fernandes, M. Freire, and P. Inacio, "Secure user authentication in cloud computing management interfaces," in *2013 IEEE 32nd International Performance Computing and Communications Conference, IPCCC 2013*. San Diego, CA, USA: IEEE, 2013.

[11] A. Castiglione, F. Palmieri, C.-L. Chen, and Y.-C. Chang, "A blind signature-based approach for cross-domain authentication in the cloud environment," *International Journal of Data Warehousing and Mining*, vol. 12, no. 1, pp. 34–48, 2016.

[12] A. Castiglione, S. De, B. Masucci, F. Palmieri, J. Li, and X. Huang, "Hierarchical and shared access control," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 850–865, 2016.

[13] W. Tesfay, T. Booth, and K. Andersson, "Reputation based security model for android applications," in *TrustCom-2012 - 11th IEEE Int. Conference on Ubiquitous Computing and Communications, IUCC-2012*. Liverpool; United Kingdom: IEEE Computer Society, 2012, pp. 896–901.