Security Analysis of Authentication Protocols for Next-Generation Mobile and CE Cloud Services

Slawomir Grzonkowski and Peter M. Corcoran
College of Engineering & Informatics,
NUI Galway, Galway, Ireland
slawek.grzonkowski@deri.org peter.corcoran@nuigalway.ie

Thomas Coughlin
President
Coughlin Associates
tom@tomcoughlin.com

***Abstract*—A number of well-known authentication protocols are considered in the context of next-generation mobile and CE network services. The potential weaknesses of current protocols can be overcome using Zero Knowledge Proof (ZKP) techniques to protect user passwords so an alternative ZKP protocol, SeDiCi 2.0, is described. This offers mutual and also two-factor authentication that is considered more secure against various phishing attempts than existing trusted third party protocols. The suitability of such a ZKP protocol for various CE-based cloud computing applications is demonstrated.**

Keywords-authentication protocol; trusted third party; single sign-on; mutual authentication, two-factor authentication, phishing, zero-knowledge proof; mobile & home network services;

I. INTRODUCTION

The next generation of CE systems and services will be dominated by mobile *thin client* devices with significant computational and connectivity capability linked to remote cloud computing infrastructure and service frameworks. These devices will also be capable of powerful multimedia data capture and dynamic content creation in addition to multimedia data consumption. In brief, the consumer will be empowered both as a creator and consumer of digital content and services.

Examples of *thin client* devices include smart-phones and tablets. Such devices now combine the capabilities of a state-of-art digital camera, a digital camcorder and a HD video display. Indeed the only deficiency of such devices is their lack of long-term data storage capabilities. However as wireless network connectivity continues to offer greater speed and higher bandwidths long-term data storage can be readily offered via *cloud computing* services.

As they become increasingly commoditized, the price of such devices will continue to fall with their adoption by consumers becoming more ubiquitous. In parallel we expect to see a new wave of networked data and content services to facilitate the users of such *thin client* devices. These new services will need to incorporate user and service authentication protocols as well as handling sensitive issues such as consumer privacy and the security of personal data.

Given the modular nature of this emerging consumer computing infrastructure one key category of services will be those of user and service *authentication*. Perhaps one of the best known approaches in this area is that of the *trusted third party* (TTP) where a neutral intermediary provides authentication services to both a service provider and a service consumer [1]. In *section II* of this paper we review some of the most recent literature on security issues and the cloud. Several recent authors [16], [19], [20], consider that existing trusted third party (TTP) protocols may provide at least part of the security solution for cloud services. In this paper we summarize number of the better-known TTP authentication services and evaluate the security risks and known weaknesses associated with each in the context of cloud computing services. An improved authentication service using Zero Knowledge Proof (ZKP) techniques [11] is also presented in *section V*.

II. SECURITY IN THE CLOUDS - RELATED WORKS

In 1961, John McCarthy was the first to publicly suggest, in a speech given to celebrate MIT's centennial, that computer time-sharing technology might lead to a future in which computing power and even specific applications could be sold through the utility business model like water or electricity. Then in 1966 all the modern-day characteristics of cloud computing, the comparison to the electricity industry and the use of public, private, government, and community forms, were thoroughly explored by Douglas Parkhill [13]. This idea of a computer or information utility continued to be popular in the late 1960s, but had faded by the mid-1970s as it became clear that the hardware, software and telecommunications technologies of the time were simply not ready. Indeed it is only since 2006 and the appearance of commercial cloud services provided by Amazon and other enterprise vendors that there has been a strong revival in the technical architectures and computing infrastructures which underpin the "Cloud". A useful background to cloud computing is provided by Armbrust et al [18].

In the last handful of years, cloud computing has grown from being a promising business concept to one of the fastest growing segments of the IT industry. But as more and more information, applications and services are placed in the cloud, concerns are emerging about how secure an environment the cloud can offer. Security is emerging as one of the primary barriers to the growth of cloud computing. Harauz et al [17] argue the need for broader industry engagement and development of standards for security in cloud computing. The growth of proprietary systems & technologies is seen as a substantial barrier to the adoption of cloud computing. Rimal and Katsaros [19] focus on the architectural aspects of cloud infrastructure providing key guidelines to software architects and applications developers. Regarding security these authors

The work presented in this paper was supported (in part) by the Lion project supported by Science Foundation Ireland under Grant No. RSF0840 and Enterprise Ireland under Grant No. REI 1005. A method and apparatus for authenticating a user is a pending patent.

found that single sign-on models based on SAML protocol (security assertion markup language) and OpenID are positioned to manage authentication issues between users and cloud service providers. Jensen et al [20] review a wide range of technical security issues that arise from the usage of cloud-computing services and conclude that the security capabilities of both Web browsers and Web service frameworks need to improved through closer integration of the two.

Subashini and Kavitha [14] present a review of security issues in service delivery models of cloud computing. They call for an integrated security model and introduce and important element of the security framework for cloud computing - the provision of "Security as a Service" based on the requirements of individual applications. This leads to the concept of a multi-tiered model where individual applications can scale security according to the value of the assets it is handling.

Risenpart et al [15] discuss how cloud services allow users to instantiate virtual machines (VMs) on demand and thus purchase precisely the capacity they require when they require it. However, using the Amazon EC2 service as a case study, these authors show that it is possible to map the internal cloud infrastructure, identify where a particular target VM is likely to reside, and then instantiate new VMs until one is placed co-resident with the target. These authors then show that such a co-resident VM can be used to mount cross-VM side-channel attacks to extract information from a target VM on the same machine. This appears to be a new class of security threat which is unique to cloud computing.

Chow et al [16] characterize many of the concerns regarding the cloud. They argue that current control measures do not adequately address the third-party data storage and processing needs of cloud computing and propose a vision to extend control measures from the enterprise into the cloud through the use of *trusted computing* and applied cryptographic techniques. They propose shifting from the concept of protecting data via the systems and applications that use the data, to that of self-protecting data. This self-protection requires intelligence be embedded in the data itself. Data needs to be self-describing and defending, regardless of its environment. Data needs to be encrypted and packaged with a usage policy. When accessed, data should consult its policy and attempt to re-create a secure environment using virtualization and reveal itself only if the environment is verified as trustworthy.

Finally, Wang et al [21] have proposed the use of a homomorphic token with distributed verification of erasure-coded data to provide error localization and ensure storage correctness of data at the block level. Their scheme offers efficient dynamic operations including data update, delete and append. Their approach provides resilience against Byzantine failure, malicious data modification attack, and even server colluding attacks.

We next review existing TTP (trusted third party) protocols to consider their suitability for providing security solutions for cloud-based CE services.

III. Existing Authentication Services

In this section we present existing solutions that are used as authentication services. They require a trusted arbiter to identify and authenticate the users. Most of them are using a ticket-based approach. The ticket is then delegated and it is acceptable by the entity, requesting authentication.

A. Kerberos

Kerberos [2], [3] is a Third Trusted Party (TTP) authentication protocol. The protocol requires parties to exchange 5 messages and an optional completion response. Kerberos provides an authentication service that acts as a trusted arbitrator. In practice service security is dependent on the client-side implementation. To exchange encrypted messages between parties Kerberos uses DES, a symmetric cryptography protocol. The authentication mechanism is based on the Needham-Schroeder TTP protocol [4] with some enchantments such as timestamps, a ticket-granting service, and a different approach to cross-domain authentication.

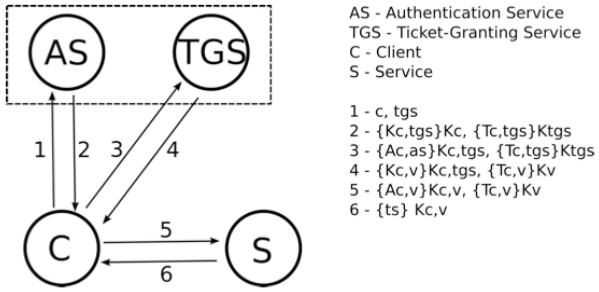

Figure 1: Kerberos authentication protocol

The basic idea of Kerberos is depicted on Figure 1. One flaw with Kerberos is that the replay attack [5] is still feasible. The timestamps that should fix this problem can be reused within the lifetime of the ticket. The lifetimes of tickets can vary between 5 minutes and 8 hours, which is enough for replay or forced delays attacks. Another problem is that the protocol assumes that all the clocks in the network are approximately synchronized. If an attacker has the ability to influence the time on other hosts, then the replay attack can be strengthened. Most network time protocols do not consider security to be a major issue, and this can be a serious weakness. Kerberos is also prone to dictionary password-guessing attacks. Users' passwords are often chosen from a small password space. Hence, an attacker who can record a complete authentication session can attempt to decrypt it with a high probability of success. It was also noted by the authors of [6] that the protocol is also prone an interleaving attack. More recent research [7] demonstrated how to perform an efficient chosen-plaintext attack. The protocol depends on the Authentication Service (AS) availability. Therefore, there is a single point of failure that can paralyze the whole network.

B. OpenID

The OpenID authentication protocol was designed in 2005 [8]. The main objective of OpenID was to provide a Single Sign-On feature for web pages. It provides a decentralized user authentication that supports using the same login credentials at

multiple websites. An illustration of how the protocol works is shown in Figure 2. The future of the protocol and its applicability to ecommerce is, however, unsure due to its phishing vulnerabilities [9].

Figure 2: OpenID authentication protocol

C. OAuth

The OAuth protocol [10] aims at secure authorization on the web. Similarly to OpenID, OAuth provides a Single Sign-On functionality. Its aim is to delegate authorization from webpages to a central authority. The central authority generates tokens for users. These tokens can be then used at webpages as required. The protocol uses nonce values, timestamps and signatures to increase security and to prevent several common attack strategies. The protocol resembles Kerberos in several aspects and thus has comparable advantages and drawbacks.

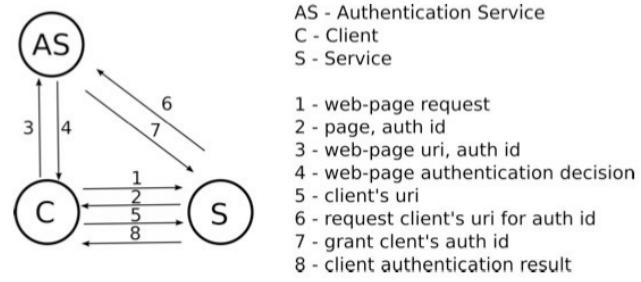

Figure 3: SeDiCi 2.0 authentication protocol

D. Conclusions

All of the above approaches rely on a user's memorable passwords. Thus, they depend upon low entropy secrets that are often prone to brute-force attacks and dictionary attacks on the user's public key or hash of the password. It is also possible that an attacker can simply attempt to login with the given username and try to guess all the possible passwords. Revealing passwords to visited websites increases also the risk of successful phishing attacks. Phishing is a social problem related to authentication, in which an attacker attempts to masquerade a trustworthy entity to obtain customers credentials, such as usernames, passwords, credit card numbers, etc.

IV. A Zero Knowledge Authentication Protocol

SeDiCi 2.0 [12] is a TTP (Third Trusted Party) protocol which is based on Zero-Knowlegde Proof (ZKP) techniques. Its aim is to deliver a better anti-phishing solution by providing mutual authentication without requiring users to reveal their passwords at each visited website. Users are not redirected to other webpages after typing logins as they are in OpenID, and thus, there is no danger of exposing a user's password to malicious parties. The proposed protocol also provides mutual authentication between service consumers and service providers. A user who performs authentication proves to the browser that a given domain is under their control. The domain acts as a Third Trusted Party (TTP) for the verifier. It can contain multiple servers confirmed by the user as trusted.

If the name of a service is present on the trusted list, the user types only the login. This way a very secure process of mutual authentication is achieved. Figure 3 depicts how the protocol works. The parties participating in the protocol are Client (C), Service (S) and Authentication Service (AS).

V. SeDiCi 2.0 Protocol

A prerequisite for the authentication process is that Client is able to communicate with both Consumer Service and Authentication Service. In the case of web-based applications this means that the *same origin* policy of most web-browsers has to be circumvented. To make the prototype work, we prepared a plugin-based implementation that allowed our application to bypass this built-in browser policy.

A. Protocol Participants

- Client (C) - this is a user who requests Service and is offered mutual authentication.
- Authentication Service (AS) - this is an authentication provider. It can be hosted by the user, if the user has a domain and a server. It can be also a company server storing multiple Clients.
- Service (S) - a URL that Client intends to visit. In this protocol, *Service* wants an identity proof of the Client that is delivered using Authentication Service.

B. Steps

1. Client creates an account at Authentication Service
    a. Client's application generates public key from the provided password
    b. Client's login and public key are sent to Authentication Service
2. Client visits Service
3. Client wants to create an account at the new Service
4. Client enters the login and press the register button
    a. The Client's application verifies the URI and records it in the Client's profile
    b. If the Verification was successful, the Client's login is sent to Service
    c. Service records the Client's login
5. Client visits Service again and attempts to authenticate
    a. Service sends *Auth ID* as a response to the Client's request
    b. Client's application sends the Service's URI and *Auth ID* to Authentication Service
    c. Authentication Service verifies the URI
    d. If the verification is successful, *Auth ID* is exposed to the given URI

e. Client sends the login to Service
f. Service verifies Client's login by checking if *Auth ID* is exposed
g. If the verification is successful, Client is authenticated

Step 1 is described in Section C. Steps 2-4 are presented in Sections D and E. Step 5 is described in detail in Section F.

C. Authenticate Service Registration

The account registration procedure for Client at Authentication Service is based on the ZKP proposition described in [11]. The difference is that the procedure is performed by a Client-side plugin instead of JavaScript. During the procedure, the plugin generates public and private key pair. The public key is sent to the server together with the desired username. If the username is not taken, the account is created. Otherwise, Client is asked for picking another username.

D. Service Registration

Registration at a custom website differs in comparison with traditional approaches. Instead of revealing an email address while a user creates a new account at a website, the user reveals only his login and sends it to Service. The plugin captures this event and sends the destination address to Authentication Service. If this address is not present on the list of known phishers, Authentication Service approves the website and inserts it to the Client's profile. Then, the login is revealed to the website. The website stores this login and other data that is specific for it. Additionally, there are also generic Clinet's properties that can be accessible through REST-based API and/or visiting user's login.

E. Client's profile

A typical Client's profile is stored in rdf and it is composed of three parts. The first one describes the user and his login, e.g., *alice.drmlab.org*. The login can be typed in a web-browser to retrieve its machine understandable version. Our prototype implementation also allows manipulation on these data using REST-based API.

The user's profile includes also descriptions of the login, public key, and a list of websites that the user knows. The second part provides more information about the websites that the user knows. Finally, the third part stores additional information about the user's public key.

F. Client authentication

During Client's authentication, three sub-authentication processes are executed. We detail them below.

1) Client to Service

This process is performed each time a user wants to login to a webpage and it is described by the following steps:

1. **Webpage request** - a client requests a web page that requires authentication procedure to be performed
2. **Auth ID** - the Service offering the requested webpage responds to the Client with the requested content. The content requires, however, authentication; thus, it's source contains *Auth ID*, which is a unique number generated for this authentication session by the Service
3. **Webpage uri, auth id** - The Authentication Service verify if the received *webpage URI* is known by the user. If the verification is positive, the Authentication Service inserts the received *Auth ID* at a place specific for the Client and for the *Webpage URI*
4. **Webpage authentication decision** - If the verification process is positive, the Authentication Service continues authentication, otherwise the authentication procedure is interrupted
5. **Client's uri** – If the webpage authentication is successful, the Client provides the login, which is formed as an URI, and sends it to the Service
6. **Request client's uri for auth id** - The Service communicates with the URI represented by the Client's login to verify if the Client is able to control the URI expressed by the login. During this process Auth ID acts as an one-time password that is proven in a challenge response way. The communication begins with a random number send to the server that is the *Service's* challenge
7. **Grant client's auth id** - Depending on the challenge protocol, the Authentication Service responds to the Service to verify if the Client posses the domain
8. **Client authentication result** - If the Client's login is correct, the user is given the access to the webpage requested in Step 1

2) Client to Authentication Service

This authentication procedure is performed each time the plugin is run. Client authenticates to Authentication Service in a Zero-Knowledge Poof way based on isomorphic graphs [11]. To authenticate, Client enters the login and the password to the Client-side plugin. The browser generates a public and private key pair from the entered credentials. Since the public part is shared with Authentication Service, the user is able to prove identity in a ZKP way.

3) Service to Authentication Service

After receiving the Client's login, the Service is able to connect the corresponding Authentication Service. To prove the Client's identity, the Service generates another random number *num* and sends it to the Authentication Service. Having the *Auth ID* and the corresponding address *www.drmlab.org*, Authentication Service responds with the result of a hash of a combination of *num* and *Auth ID*:

HASH(num : Auth ID)

Since Service knows both Auth ID and *num*, the resulting hash is sufficient to determine if the Authentication Service knows *Auth ID*. Using this approach Service is able to verify if Client indeed can manipulate information at the server that is determined from the login. Thus, Client confirms the identity.

VI. SECURITY PROPERTIES

The presented protocol requires users URIs or other identifiers that can be used to locate their bearers. The advantage of URI is that it contains a user name and a corresponding authentication service address; and also URI is globally unique. The described protocol requires users to have control over the domain that they use for authentication. Since this is something they have, it can be considered as a second authentication factor. For example, other authors considered a bookmark as a second authentication factor [23].

In the proposed protocol the user never types his password at the visited websites. A user login is the only information that is revealed. However, the login alone is not useful for malicious services if they obtain it. Thus, in SeDiCi 2.0 the user is only responsible for providing identification and the responsibility of authentication is in fact moved to third trusted parties. This way we address the problem of low entropy user passwords and also the problem of phishing, in which for example malicious websites are convincing users to reveal their passwords.

The SeDiCi 2.0 protocol does not require any physical tokens. However, when it is deployed on the user browser, it requires a plugin that overwrites the same origin policies that are a standard security mechanism for websites. Only this way a web-browser can communicate with external services.

VII. CONCLUSIONS

We presented a novel authentication protocol that is suitable for cloud-based services. In comparison with existing solutions such as Kerberos, this protocol does not require physical tokens and it is not prone to replay attacks; also, there is no shared password between the user and the authentication service: this part is based on a password-based zero-knowledge proof based protocol that we developed earlier [11], [22].

In addition to OpenID, our protocol addresses the problems of phishing and mutual authentication. It also supports the functionality of single-sign on.

In a prototype that we developed, the users were required to have control over the domains that they were using for authentication. Thus, we our solution applies two-factor authentication that is considered more secure than traditional approaches [23].

We claim that the proposed protocol can provide a suitable solution for CE-based cloud authentication services.

REFERENCES

[1] T. Coughlin and M. Alvarex, Angels in our Midst: Associative Metadata in Cloud Storage, CE Society Newsletter, Spring 2011.

[2] B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. In IEEE Communications, 32(9):33-38, September 1994.

[3] B. C. Neuman, T. Yu, S. Hartman and K. Raeburn. The Kerberos Network Authentication System (RFC4120), July 2005.

[4] R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. Commun. ACM, vol. 21, no. 12, pages 993–999, 1978.

[5] S. M. Bellovin and M. Merritt. Limitations of the Kerberos authentication system. SIGCOMM Comput. Commun. Rev., vol. 20, no. 5, pages 119–132, 1990.

[6] T. Wen-Guey and H. Chi-Ming. Inter-protocol interleaving attacks on some authentication and key distribution protocols. Information Processing Letters, vol. 69, no. 6, pages 297 – 302, 1999.

[7] T. Yu, S. Hartman and K. Raeburn. The Perils of Unauthenticated Encryption: Kerberos Version 4. In Proceedings of the Network and Distributed System Security Symposium. The Internet Society, 2004.

[8] D. Recordon and D. Reed. OpenID 2.0: a platform for user-centric identity management. In DIM '06: Proceedings of the second ACM workshop on Digital identity management, pages 11–16, New York, NY, USA, 2006. ACM Press.

[9] B. Adida. The Browser as a Secure Platform for Loosely Coupled Private-Data Mashups. In W2SP 2007, Proceedings of the First Workshop on Web 2.0 Security and Privacy, Oakland, CA, USA, 2007.

[10] E. Hammer-Lahav, Ed. *The OAuth 1.0 Protocol*., http://tools.ietf.org/html/draft-hammer-oauth-10

[11] S. Grzonkowski, W. Zaremba, M. Zaremba and B. McDaniel. Extending Web Applications with a Lightweight Zero Knowledge Proof Authentication. In CSTST '08: Proceedings of the 5th international conference on Soft computing as transdisciplinary science and technology

[12] S. Grzonkowski. SeDiCi: An Authentication Service Taking Advantage of Zero-Knowledge Proofs. In Financial Cryptography. Springer, 2010

[13] D F Parkhill, *The Challenge of the Computer Utility*, Addison-Wesley Publishing Company, (1966)

[14] S. Subashini, V. Kavitha, "A survey on security issues in service delivery models of cloud computing" in *Journal of Network and Computer Applications*, Vol. 34, No. 1. (15 January 2011), pp. 1-11.

[15] T., Ristenpart, E., Tromer, H., Shacham, and S., Savage. Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds. In *Proceedings of the 16th ACM conference on Computer and communications security* (CCS '09). ACM, New York, NY, USA, 199-212, 2009.

[16] R., Chow, P., Golle, M. Jakobsson, E., Shi, J., Staddon, R., Masuoka, and J. Molina. *Controlling data in the cloud: outsourcing computation without outsourcing control.* In Proceedings of the 2009 ACM workshop on Cloud computing security (CCSW '09). ACM, New York, NY, USA, 85-90, 2009.

[17] L.M., Kaufman; , *Data Security in the World of Cloud Computing*, Security & Privacy, IEEE , vol.7, no.4, pp.61-64, July-Aug. 2009

[18] M., Armbrust, A., Fox, R., Griffith, A.D., Joseph, R., Katz, A., Konwinski, G., Lee, D., Patterson, A., Rabkin, I.,. Stoica, and M., Zaharia. 2010. A view of cloud computing. *Commun. ACM* 53, 4 (April 2010), 50-58.

[19] B.P., Rimal, A., Jukan, D., Katsaros and Y. Goeleven, *Architectural Requirements for Cloud Computing Systems: An Enterprise Cloud Approach*, Journal of Grid Computing, Volume 9, Number 1, (March 2011) pp.3-26.

[20] J., Schwenk, N. Gruschka, L. Iacono; , *On Technical Security Issues in Cloud Computing*, Cloud Computing, 2009. CLOUD '09. IEEE International Conference on , vol., no., pp.109-116, 21-25 Sept. 2009

[21] Cong Wang; Qian Wang; Kui Ren; Wenjing Lou; , *Ensuring data storage security in Cloud Computing*, Quality of Service, 2009. IWQoS. 17th International Workshop on , vol., no., pp.1-9, 13-15 July 2009

[22] S. Grzonkowski, P.M. Corcoran. *A secure and efficient micropayment solution for online gaming.* In Proceedings of the International IEEE Consumer Electronics Society's Games Innovations Conference 2009 (ICE-GIC 09), 2009.

[23] B. Adida. *Beamauth: two-factor web authentication with a bookmark.* In CCS'07: Proceedings of the 14th ACM conference on Computer and communications security, pages 48-57, New York, NY, USA, 2007. ACM.