Web Authentication Approaches & Their Resistance Against MITM Attacks: A Comparative Analysis

Ioannis Gkourtzounis
Department of Computing, The University of Northampton,
Park Campus, Boughton Green Road, NN2 7AL, Northampton, UK
ioannisgk@live.com

***Abstract*—Today more and more websites are sharing resources and services with their users. From Web APIs to Service Oriented Architectures, web applications are exposed globally and security breaches have important negative consequences. Web security has become crucial to e-businesses and proper authentication and authorization lie at the heart of their defenses. In this paper we analyze and compare the most widely used authentication approaches: HTTP Basic Authentication, HTTP Digest Authentication, OAuth 1.0a and OAuth 2.0 frameworks. Finally, we present practical examples of authentication implementations under Man in The Middle attack scenarios and evaluate the security aspects of each approach, the role of the channel of communication and the role of cryptography in web security.**

Keywords*—*Web security, authentication, authorization, Basic Authentication, HTTP Digest, OAuth 1.0, Oauth 2.0, Man in The Middle attack, cryptography, SSL/TLS.

I. Introduction

Today, Facebook, Google and Twitter are just some of the websites that share resources by offering Application Programming Interfaces (APIs) to third parties [1]. Service Oriented Architectures (SOA) [2] and Web APIs helped in the evolution of cloud computing, where businesses provide services to consumers in on-demand basis, and allow developer access to certain functionalities of their services [3].

The web is a complex platform that makes available services and applications, but with many different aspects regarding security considerations [4]. Business applications are accessible from anywhere and this exposure means that security vulnerabilities will inevitably be uncovered and exploited [5]. One of the most worrying aspects of web security today is that deep technical knowledge is not even required to attack a system [6]. As a result, security threats have serious negative consequences to businesses, harming their finance and reputation [7, 8].

In this paper we examine the fundamental ideas of web security and compare the most widely used authentication approaches. We analyze each approach, point out its strengths and weaknesses and then present some practical examples of web authentication implementations under Man in the Middle (MITM) attack scenarios. Our purpose is to evaluate the resistance of those approaches in a comparative analysis. Then, we will present our findings and discuss about possible solutions.

II. Background

In order to make the web more secure we follow some processes, best practises, and implement certain technologies to ensure that web applications work in a reliable and predictable way. This is the concept of web security in general, and Garfinkel et al. [9] point out its three main aspects: we need to secure (i) the web server and its data, (ii) the information that travel across the communication medium and (iii) the clients and their data.

Let us see the main objectives of security. First, individuals should control what information can be collected and by whom. Confidential information should not be available to unauthorized parties. Second, the system must work as expected without unauthorized intervention. Third, the system must be always available to authorized users. Furthermore, establishing identities, verifying trusted users and their actions are also very important [10-12]. Authentication and authorization concepts are at the heart of web security, especially in large distributed systems [13]. We will now examine them in more detail along with their underlying communication protocols.

Generally, authentication is the process of permanently accepting the truth of some claim or message. We can accomplish a high level of authentication if the message is highly confidential [14, 15]. Its main forms are: individual, identity and attribute authentication. So, we can define authentication as the process of establishing a specific level of confidence that (i) an identifier refers to a specific individual, (ii) an identifier refers to an identity or (iii) an attribute applies to a specific individual [16]. Confirming the identity of a web client or user is an example of web authentication.

Authorization is the series of actions to determine what an individual is allowed to do. An appropriate authority decides to allow or deny a request for an authorization decision [16]. It specifies that a user is allowed to exercise a privilege, like accessing resources, based on his identity, that was confirmed during authentication [17, 18]. The authorization rules can be complex and require that we know the identity of the initiator and, depending on the system, the identity of other entities, like the service provider [19]. These rules form the security policy and their goal is to protect sensitive information or resources from unauthorized access. Granting permissions and assigning roles to web clients or users is an example of web authorization.

The web is made up of clients, servers and an underlying

mechanism of exchanging messages or resources. Hypertext Transfer Protocol (HTTP) is a stateless protocol that controls the sequential message exchange between clients and servers [20, 21]. The client requests a resource on the server and the server responds with response headers and an optional response body in HTML [22, 23]. HTTP is easy to use but it is based on plain text messages, thus making user authentication unsecured and vulnerable to attacks [24]. So, HTTPS was developed, providing three security guarantees: server authentication, message integrity, and message confidentiality [25]. HTTPS is based on the Transport Layer Security[1] (TLS) Protocol, a Public Key Infrastructure (PKI) and Certificate Authorities (CAs) that identify web servers [26]. CAs are trusted organizations that sign digital certificates, associating a website's public key with its domain name [26, 27]. Symmetric keys are then used to encrypt data and transmit them through a secure channel [28]. However, as Naylor et al. [29] show, HTTPS adds noticeable performance costs to networks compared to HTTP.

The procedure of how SSL/TLS protocol operates in a client-server TCP connection has the following steps: (i) the client starts the SSL handshake by sending the SSL/TLS version that is is running, the set of algorithms (cipher suite) it can use and the compression methods that it supports, (ii) the server checks the highest SSL/TLS version supported by both and selects a cipher suite and optionally a compression method, (iii) the server sends its certificate to the client and the client checks if it is digitally signed by a trusted CA, by validated the chain of all certificates between the root CA and the server, (iv) both parties compute the keys for the symmetric key encrypted communication, (v) the client sends to the server the Message Authentication Code (MAC), that will be used for authentication (vi) the server verifies the MAC, the handshake ends and now the client and server can communicate securely. Some examples of the symmetric key algorithms that are used in SSL/TLS are AES, RC2, RC4, DES and 3DES.

Next, we will mention ways that hackers can use to exploit a system. Nisha et al. [30] bring to our attention the most common network attack techniques: eavesdropping, tampering, spoofing, hijacking and capture/replay. Wireless networks are more vulnerable to eavesdropping [31], tampering, and spoofing, where fake data are created to deceive the victim [32]. Session hijacking takes place when an attacker gets a client's session information and uses it to authenticate [33]. In capture/replay, a stream of data is recorded and later sent, in an attempt to create harmful consequences. While all those network attacks are essentially modes of MITM attacks, we need to find ways to defend against those malicious actions.

III. Authentication Approaches

In this section we will examine the most common authentication approaches, the way they are implemented and how they are used, and present their advantages and disadvantages.

[1]Transport Layer Security protocol is the successor of Secure Sockets Layer protocol.

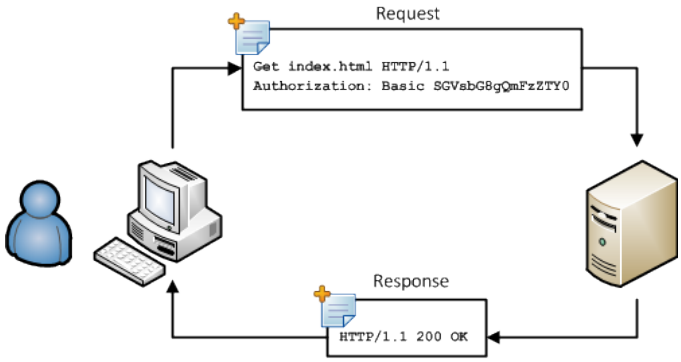

Fig. 1. Basic Authentication process. The server decodes the encoded string, gets the "username:password" string and if the credentials are valid, it gives access to the client [63].

A. HTTP Basic Authentication

In HTTP Basic Authentication [34-36] we identify a user or web client based on his username and password. When a client needs to access a resource, it sends an HTTP request with a header field called "Authorization" and the "username:password" credentials, encoded in Base64[2] plain text format. The server examines the username and password against the ones in the database and if they are confirmed a status code 200 is sent in the response headers. From this moment the client is authenticated. Otherwise, a "401 Authorization Required" code is sent with a header field "WWW-Authenticated" and a name of the realm, a specific resource on the server (Fig. 1).

As the username and password are not encrypted or hashed, we can use Basic Authentication over HTTPS to protect the sensitive data. Basic Authentication provides a simple and fast way for user authentication. It does not require login pages, handshakes, cookies or session identifiers, but it is vulnerable to hijacking and capture/replay attacks, especially over HTTP. Also, allowing the server to store passwords, can not be considered the safest strategy as the server may be compromised and a hacker may be able to decrypt them.

B. HTTP Digest Authentication

HTTP Digest authentication [36-38] is designed to be more secure than Basic Authentication over HTTP. The client sends a request and receives a challenge response with the following information: the realm name, the authentication algorithm name, and a nonce, which is a random and unique string that is sent on every request. When the client receives the challenge, it generates a digest string based on three steps: (i) it creates an MD5 hash[3] of the HTTP method and the path of the request, (ii) it creates MD5 hash of the information from the challenge, (iii) it creates MD5 hash of the username, password and its

[2]Base64 is a binary to text encoding scheme that represents binary data in ASCII string format.

[3]The MD5 algorithm is a popular used hash function that maps the original data and produces a hash value.

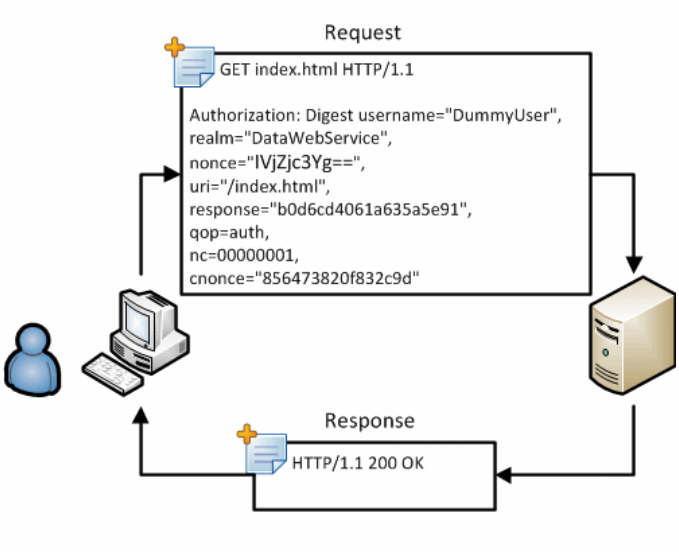

Fig. 2. Digest Authentication process. The server receives the digest string and if the credentials are valid, it gives access to the client [64].

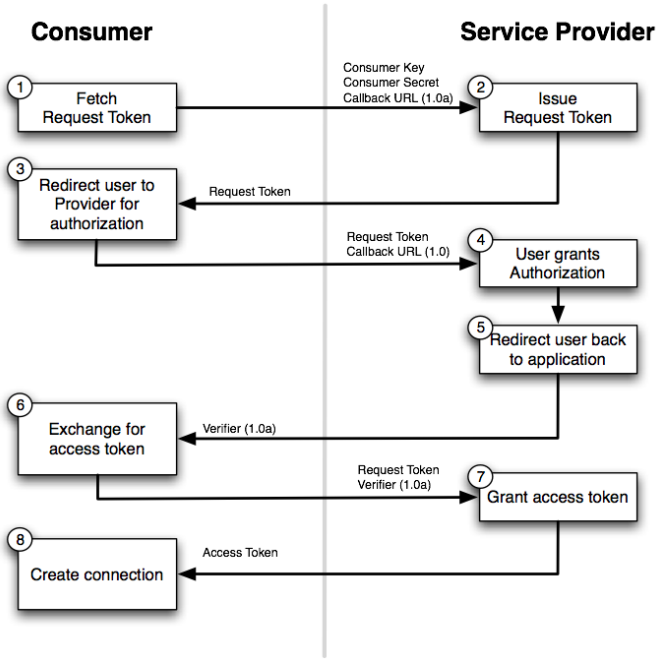

Fig. 3. OAuth 1.0a framework workflow. The service provider verifies the the consumer key and consumer secret, and generates a request token for the consumer. User grants authorization and consumer receives the verifier [65].

own nonce and sequence number. The resulting digest string is sent to the server. Now the server, from the stored MD5 hashes of the username and password, along with the other values, computes the final digest string and if the data is correct then a status code 200 is sent. In case the strings do not match, a ”401 Authorization Required” code is sent back to the client (Fig. 2).

Generating a different server nonce on every request and storing only MD5 hashes of the credentials, raise the security level of the process, especially over HTTPS. Some capture/replay attacks can be prevented if the server nonce includes a timestamp. On the other hand Digest Authentication is hard to implement, it is still insecure over HTTP and it is vulnerable to MITM attacks that can ”downgrade” it to Basic Authentication. The server stores hashed credentials in MD5. Yet, MD5 can be tampered and collisions can easily be created by a attacker [39]. Another disadvantage of Digest Authentication is that it generates more traffic, so it is considered costly.

C. OAuth 1.0a Framework

Open Authorization (OAuth) [40-42] was designed for website authorization only, but it became a popular framework for authentication and authorization in web services and applications. Version 1.0 was vulnerable to a session attack on the request token, so a revision (OAuth 1.0a) included a verification code to the final request.

The basic idea is that an end-user or application (resource owner) owns server resources and clients (consumers) can access those resources on his behalf by making requests to a server (service provider). First a consumer asks for a request token from the service provider with the following parameters: a consumer key, a signature method, a consumer secret (encrypted signature string), a timestamp and a nonce. The consumer key and consumer secret will be verified by the service provider and he will generate a request key and request token. With this information the consumer redirects the user to the authorization page where he can grant access to the consumer. The consumer receives a verifier parameter and with the request token he can now receive an access token from the service provider which will allow him to proceed with further requests. This way, users can authorize third-party access, without exposing their username and password to the network (Fig. 3).

Clients are required to generate new signatures on every request, so if an attacker could access the tokens, he would not be able to change the credentials and that made OAuth 1.0a very secure, even when not running over HTTPS. It was the best way to provide security without an underlying secure channel.

OAuth 1.0a was popular and Twitter, Dropbox and Flickr used it as it was a great way for integrating social networks in a secure and reliable way. The major drawback was that the implementations became very complicated because developers for client applications had to write code supporting different algorithms, like HMAC-SHA1 and RSA-SHA1. HMAC-SHA1 is a key-hashed MAC that uses the SHA1 hash function and a secret key to verify the data integrity and authentication of a message. RSA-SHA1 is a public key cryptographic algorithm for encryption and authentication that uses the SHA1 hash function. OAuth 1.0a also required multiple requests for the authentication and authorization process to be completed. Different kinds of implementations added in the overall complexity, like two-legged and three-legged, for two parties and three parties respectively.

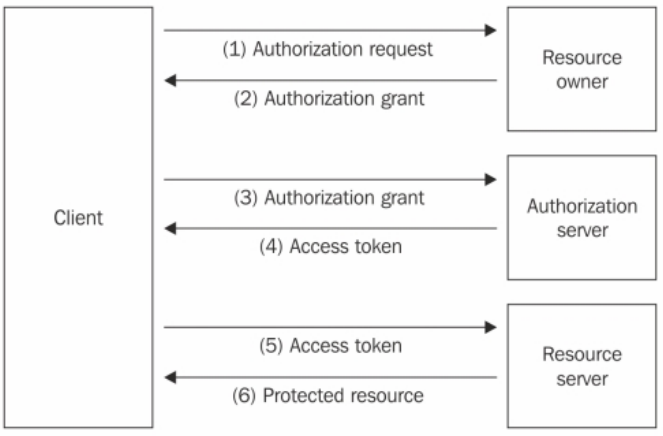

Fig. 4. OAuth 2.0 framework workflow. The client requests authorization from the resource owner and receives a grant. Authorization server checks the grant and sends an access token back to the client.

D. OAuth 2.0 Framework

In OAuth 2.0 [43-46], one of the most popular authentication and authorization frameworks, we have four roles: (i) the resource owner (end-user or application) grants access to server resources, (ii) the resource server hosts the resources and responds to requests, (iii) the client makes the requests to access the resources, and (iv) the authorization server issues access tokens to the client after authenticating the resource owner and obtaining authorization.

The following steps take place: the client requests authorization directly from the resource owner or via the authorization server, the client receives a grant of a specific type[4]. Next, the client sends the grant to the authorization server, gets authenticated and receives an access token. The client can request access to protected resources from the resource server, by sending the access token with his request (Fig. 4).

In case of an application acting as the client, there are two main flows that it can get the access token to access server resources. In implicit grant flow the user is redirected to the service provider for authentication. After authentication an access token is sent to the application, but in authorization code grant flow, the application gets a code, which he sends back to the service provider along with a secondary secret code. If the data is valid, the application receives the access token and it is now ready to access the resources by sending the access token as a parameter in the subsequent requests.

OAuth 2.0 introduced some very important changes trying to neutralize the negative characteristics of OAuth 1.0a. The clients get a single token, the signature is not encrypted but the use over HTTPS became compulsory. Leaving the encryption of the requests to SSL/TLS was a sensible thing to do and it improved the implementation. The token now expires after a specific time period and needs to be refreshed. This made OAuth 2.0 more secure. Refresh tokens are issued by the authorization server so the client can obtain a new access token when one is expired. OAuth 2.0 is considered expensive in computational resources because generating and validating signatures require a lot of effort. Still, due to the security it provides, some of the biggest companies have embraced it, like Facebook, Google and GitHub.

[4]The grant types in OAuth 2.0 are: authorization code, implicit, resource owner username and password, and client credentials.

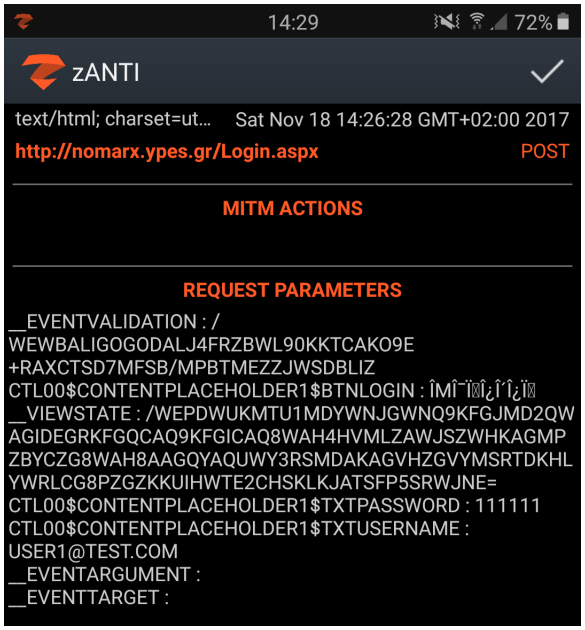

Fig. 5. MITM attack on a target in the same wireless network using zANTI. We can see the username (user1@test.com) and password (111111) entered by the user on the POST request parameters.

IV. Comparative Analysis

For our comparative analysis, we take into account the characteristics of each authentication approach and we also examine their resistance in MITM attacks. The criteria for the comparison are how easy it is for an attacker to access the user credentials in plain text, known vulnerabilities and the results of our attacks. The overall security of each authentication implementation against MITM attacks over HTTP and over an encrypted channel of communication, will be evaluated. We will also point out which approaches are easy to implement and which are slow and generate a lot of traffic.

In the following examples we examine the consequences of MITM attacks to various authentication implementations. The characteristics [47-50] of this kind of attacks are: (i) the attacker intercepts the communication between the client and the server, (ii) he impersonates the server so the client is connected to his device directly, and (iii) the attacker can capture the client's credentials while the communication appears normal. A hacker can launch the MITM attack by ARP cache poisoning[5], by DNS spoofing or session hijacking [51]. These examples will demonstrate how authentication approaches behave against MITM attacks and whether the user credentials will be exposed or remain hidden.

[5]ARP (Address Resolution Protocol) poisoning is basically IP and MAC address forging, combined with cache update policy exploits.

#1 <--- 11-18 13:53:35

```
POST /evsharing-platform/oauth/token?
grant_type=password&username=evsharingUser&password=evsharingPass HTTP/1.1
Accept: application/json
Authorization: Basic ZXZzaGFyaW5nQ2xpZW50OmV2c2hhcmluZ1Bhc3M=
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0.1; SM-N930F Build/MMB29K)
Host: 178.62.121.237:8080
Connection: Keep-Alive
Accept-Encoding: gzip
```

#2 ---> 11-18 13:53:35

```
HTTP/1.1 200
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Frame-Options: DENY
Content-Type: application/json;charset=UTF-8
Transfer-Encoding: chunked
Date: Sat, 18 Nov 2017 11:53:35 GMT

b0
{"access_token":"d2326004-af3a-43f7-813b-5db452576b11","token_type":"bear
er","refresh_token":"affd0c14-6ed4-417d-9203-ebb6c591a98c","expires_in":
119,"scope":"read write trust"}
0
```

#3 <--- 11-18 13:53:36

```
POST /evsharing-platform/api/login/?access_token=d2326004-
af3a-43f7-813b-5db452576b11 HTTP/1.1
Accept: application/json
Content-Type: application/json;charset=UTF-8
Content-Length: 67
User-Agent: Dalvik/2.1.0 (Linux; U; Android 6.0.1; SM-N930F Build/MMB29K)
Host: 178.62.121.237:8080
Connection: Keep-Alive
Accept-Encoding: gzip

{"username":"user1@test.com","password":"5054Dh45Jh6PKlzarV5KBw=="}
```

Fig. 6. MITM attack on a RESTful web service with Packet Capture. The resource owner and client credentials are visible on the first POST request, on the response we can see the tokens and on the second POST request we can also see the user's credentials.

A. MITM Attack on a login page over HTTP

Our first example is a client, connected in a wireless network, who accesses a login page over HTTP. The security of the login page and the authentication process is implemented in the application level. For our attack we used zANTI from Zimperium [52], a free tool used for professional penetration testing. After research, we found three websites that used HTTP for the login process: NOMARX.YPES.GR, ACCI.GR and GSRT.GR. We selected our target, enabled MITM attack, and from that time the client was connected to the internet via zANTI, our man in the middle. On all three websites the usernames and passwords were easy to extract as they were in the headers in plain text format (Fig. 5).

B. MITM Attack on a RESTful web service over HTTP

For our second example we implemented a RESTful web service [53, 54] and added authentication and authorization security per OAuth 2.0 specifications, with the exception of using it over HTTP and not HTTPS. We followed this approach to see the strength of the authentication protocol independently from the channel of communication. For this attack we used a free Android tool on Google Play called Packet Capture [55]. When Packet Capture is installed on an Android device, it acts as a MITM, intercepting all communication. We created, installed and run an application that uses our RESTful web service client and started capturing packets. We logged in to the web service from our application and saw that Packet Capture had captured all the requests and the information were visible in plain text, such as usernames, passwords and tokens (Fig. 6).

<--- TEXT

```
POST /ws/eBayISAPI.dll?co_partnerId=2&siteid=0&UsingSSL=1 HTTP/1.1
Host: signin.ebay.com
Connection: keep-alive
Content-Length: 3143
Cache-Control: max-age=0
Origin: https://signin.ebay.com
Upgrade-Insecure-Requests: 1
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Linux; Android 6.0.1; SM-N930F Build/MMB29K)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/62.0.3202.84 Mobile Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/
apng,*/*;q=0.8
Referer: https://signin.ebay.com/ws/eBayISAPI.dll?

                                (omitted text)

d4e1c8c19ea6ad861e300442d467&siteid=0&co_partnerId=2&ru=https://
m.ebay.com&pp=&pa1=&pa2=&pa3=&i1=-1&pageType=-1&rtmData=&usid=cf7f2b2f15f0a
b117152f562fff1333b&rqid=cf7f2b2f15f0ab1171568926ff6d29da&afbpmName=sess1&k
gct=&userid_otp=user1@test.com&otp=&keepMeSignInOption3=1&userid=user1@test.co
m&pass=111111&sgnBt=Sign in&keepMeSignInOption2=1&keepMeSignInOption=1&htmid=sl
%3D%7Cslnew%3D%7Cht5%3DAQAAAV
%252BjOdksAAUxNWZjZjdmMmIyZi5hYjExNzE1LjJmNTYyLmZmZjEzMzNjOhxqr64htrLZ
%252BH4sOzFnzjGjOXO*%7Cht5new%3Dtrue&kdata=%1E%1F%E5%1E1511014743861%1E1%1F
%E5%1E1511014743870%1E0%1F%E5%1E1511014744217%1E1%1F%E5%1E1511014744220%1E0%1F
%E5%1E1511014744565%1E1%1F%E5%1E1511014744567%1E0%1F%E5%1E1511014744928%1E1%1F
%E5%1E1511014744940%1E0%1F%E5%1E1511014745304%1E1%1F%E5%1E1511014745310%1E0%1F
%E5%1E1511014745664%1E1%1F%E5%1E1511014745670%1E0%1F
```

Fig. 7. MITM attack on eBay authentication service over HTTPS with Packet Capture. We can see the user id (user1@test.com) and password (111111) in plain text, in the last section of the POST request.

C. MITM Attack on a Device Communicating over HTTPS

The last example was about examining HTTPS security, so we tested the login procedures of eBay and Amazon. We used Packet Capture to install a custom certificate on the Android device. It acted as a MITM but this time the communication was encrypted. So, Packet Capture tool sets up a CA to sign a custom certificate that associates a server with its public key. Client's applications and browsers can not verify invalid certificates and display a warning to the user [47]. If the user accepts, the forged certificate is considered trusted and the attacker can intercept the communication [56, 57]. We enabled the tool and then tried to login to eBay and Amazon. We found that all information were decrypted by the tool and the username and password were visible in the requests (Fig. 7).

CRITERIA/APPROACHES	BASIC	DIGEST	OAUTH 1.0a	OAUTH 2.0
Credentials visibility	In plain text format	Hashed with nonce & other parameters	Encrypted with other parameters	In plain text format
Attacker effort for MITM over HTTP	Requires little effort	More effort, due to hashed credentials	Requires a lot of effort	More effort, due to SSL/TLS
Vulnerable to MITM over HTTP	Credentials can be exposed	Credentials can be exposed	Hard to expose credentials	Credentials can be exposed
Attacker effort for MITM over HTTPS	Requires user to accept custom CA	Requires user to accept custom CA	Requires user to accept custom CA	Requires user to accept custom CA
Vulnerable to MITM over HTTPS	Credentials can be exposed	Credentials can be exposed	Hard to expose credentials	Credentials can be exposed
Security assessment over HTTP	Not secure, credentials are visible	More secure than Basic but vulnerable	Very secure, new signatures on requests	Not secure, credentials are visible
Security assessment over HTTPS	Secure, but depends only on SSL/TLS	Secure with hashed values & SSL/TLS	Very secure, credentials are not exposed	Very secure, but depends on SSLS/TLS
Implementation difficulty	Simple and easy	Harder to implement	Very hard, many implementations	Simple implementation with tokens
Fast or slow response	Fast	Not fast, requires many calculations	Slow, due to multiple requests	Fast, but requires many calculations
Traffic generation	Low traffic	Medium traffic	High traffic, due to multiple requests	High traffic, due to multiple requests
Mainly used for	Authentication	Authentication	Authentication & Authorization	Authentication & Authorization
OVERALL ASSESSMENT	Simple, fast, secure only over HTTPS	More secure than Basic over HTTP	Hard to implement, but very secure	Simple, very secure only over HTTPS

Fig. 8. Comparative analysis table. Four authentication approaches assessed against MITM attacks and other criteria.

D. Findings

We compared four authentication approaches, tested their implementations against MITM attack scenarios and have some important findings to present. First, Basic, Digest and OAuth 2.0 approaches should be used only over an encrypted channel of communication. Second, the only secure approach over HTTP is considered the OAuth 1.0a framework. Third, the implementation difficulty plays an important role in the broader adoption of an authentication approach by companies and developers, and OAuth 1.0a is very complex and hard to implement. And that is the reason why most companies chose OAuth 2.0 for securing their web services. In summary, Basic Authentication is simple, fast and secure only over HTTPS, Digest Authentication is more secure over HTTP but still has vulnerabilities, OAuth 1.0a is hard to implement but very secure, and OAuth 2.0 is simple and very secure over HTTPS (Fig. 8).

Furthermore, launching a successful MITM attack on Basic Authentication requires little effort from the attacker. In the case of Digest Authentication more effort is needed due to hashed passwords, while in OAuth 1.0a it is very hard to expose the credentials and in OAuth 2.0 the SSL/TLS encryption has to be bypassed by making the user accept and install a custom CA. If the user does not accept the custom CA, the data are safe on the encrypted SSL/TLS channel.

From the above, we can conclude how important cryptography is, in authentication and authorization procedures. Even if we apply the best and most secure approach, all data are visible in plain text messages over HTTP and this gives an attacker the opportunity to analyze them and figure out ways to bypass the current security. Using an SSL/TLS protocol means that the client sends data to a verified server and that all messages between the client and server are authenticated and encrypted. Communicating over an encrypted channel adds another line of defense against malicious actions. Only when the user accepts a custom CA and installs it, this line of defense can be compromised.

Cryptography algorithms like AES, RC2, RC4, DES and 3DES are used to ensure information confidentiality. As we can see, cryptography is a very important asset in web security, especially where sensitive information travel across the Internet.

V. Conclusions

Basic Authentication is easy to implement but it is unsafe, especially if used over HTTP. Digest Authentication is more secure by including hashing and timestamps but still vulnerable to attacks. OAuth 1.0a was designed to offer great security even when communicating over an unencrypted channel. It gained significant popularity but its major drawback was the complexity of the implementation. OAuth 2.0 was simpler, leaving the encryption to SSL/TLS, and added expiring tokens for greater security. Making its implementation simpler, it attracted big companies and most of them use OAuth 2.0 for authentication and authorization today.

We saw that intercepting communication over HTTP, even if using OAuth 2.0, is very easy with the use of free tools. We also found that if a user accepts a CA warning, an attacker can decrypt his HTTPS requests and extract all information, usernames and passwords. Even the strongest authentication approach over HTTPS can be defeated by a well orchestrated MITM attack and a novice user decision. However, we should always bulletproof the sensitive data travelling between clients and servers by using only encrypted channels of communication. Cryptography is essential to web security and ensures information confidentiality.

Many countermeasures have been proposed for improving HTTPS security, like HTTPAS (HTTP Active Secure framework) [48] that maximizes the use of available trusted CAs, HTTPSLock [58] by not letting users accept invalid certificates, and HSTS (HTTP Strict Transport Security) [59, 60] that forces HTTPS communication only. The extension of TLS based on Quantum Cryptography, has also been proposed since 2010, that allows key exchange with absolute security [61]. Alternatively, we can abstract security critical code and describe it in Security Policy Definition Language (SPDL) specification documents [62]. As Garfinkel et al. [9] emphasize, cryptography is the fundamental technology to encrypt and protect our data and we should improve it constantly.

REFERENCES

[1] M. Maleshkova, C. Pedrinaci and J. Domingue, "Investigating Web APIs on the World Wide Web", *2010 Eighth IEEE European Conference on Web Services*, 2010.

[2] V. Henrich, E. Hinrichs, M. Hinrichs, and T. Zastrow, "Service-Oriented Architectures: From Desktop Tools to Web Services and Web Applications", *Multilinguality and Interoperability in Language Processing with Emphasis on Romanian*, pp.978-973, 2010.

[3] N. Sultan, "Cloud computing for education: A new dawn?", *International Journal of Information Management*, vol. 30, no. 2, pp. 109-116, 2010.

[4] D. Akhawe, A. Barth, P. Lam, J. Mitchell and D. Song, "Towards a Formal Foundation of Web Security", *2010 23rd IEEE Computer Security Foundations Symposium*, 2010.

[5] J. Fonseca and M. Vieira, "Mapping software faults with web security vulnerabilities", *2008 IEEE International Conference on Dependable Systems and Networks With FTCS and DCC (DSN)*, 2008.

[6] M. Andrews, "Guest Editor's Introduction: The State of Web Security", *IEEE Security Privacy Magazine*, vol. 4, no. 4, pp. 14-15, 2006.

[7] H. Matbouli and Q. Gao, "An overview on web security threats and impact to e-commerce success", *2012 International Conference on Information Technology and e-Services*, 2012.

[8] L. Qian, J. Wan, L. Chen and X. Chen, "Complete Web Security Testing Methods and Recommendations", *2013 International Conference on Computer Sciences and Applications*, 2013.

[9] S. Garfinkel and G. Spafford, *Web security, privacy, and commerce*. Beijing [etc.]: O'Reilly, 2002.

[10] W. Stallings, *Data and computer communications*. Upper Saddle River, NJ: Pearson Prentice Hall, 2007.

[11] D. Geer, "Taking steps to secure web services", *Computer*, vol. 36, no. 10, pp. 14-16, 2003.

[12] K. Knorr and S. Rhrig, "Security of Electronic Business Applications: Structure and Quantification", *Electronic Commerce and Web Technologies*, pp. 25-37, 2000.

[13] A.H.M. Emam, "Additional authentication and authorization using registered email-ID for cloud computing", *International Journal of Soft Computing and Engineering*, vol. 3, no. 2, pp.110-113, 2013.

[14] G. Bella and S. Bistarelli, "Information Assurance for security protocols", *Computers Security*, vol. 24, no. 4, pp. 322-333, 2005.

[15] G. Bella, "What is correctness of security protocols?", *J. UCS*, vol. 14, no. 12, pp.2083-2106, 2008.

[16] S. Kent and L. Millett, *Who goes there? Authentication Through the Lens of Privacy*. Washington, D.C.: National Academies Press, 2003.

[17] B. Thuraisingham, C. Clifton, A. Gupta, E. Bertino and E. Ferrari, "Directions for Web and e-commerce applications security", *Proceedings Tenth IEEE International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE)*, 2001.

[18] D. Inclezan and M. Sobolewski, "Security Policy Management in Federated Computing Environments", 2007.

[19] B. Hartman, D. Flinn, K. Beznosov and S. Kawamoto, *Mastering Web services security*. Indianapolis, Ind.: Wiley Technology Pub., 2003.

[20] P. Davern, N. Nashid, A. Zahran and C. Sreenan, "HTTP Acceleration over High Latency Links", *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, 2011.

[21] P. Davern, N. Nashid, C. Sreenan and A. Zahran, "HTTPEP: a HTTP Performance Enhancing Proxy for Satellite Systems", *International Journal of Next-Generation Computing*, vol. 2, 2011.

[22] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. and Berners-Lee, "Hypertext transfer protocol–HTTP/1.1", No. RFC 2616, 1999.

[23] A. Jawad, "The Fundamentals of HTTP/2", 2016.

[24] K. Lee, H. Yeuk, S. Kim and K. Yim, "Security Assessment on User Authentication by an HttpSendRequest Hooking in an HTTP Client", *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, 2013.

[25] T. Dierks and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, 2008.

[26] Z. Durumeric, J. Kasten, M. Bailey and J. Halderman, "Analysis of the HTTPS certificate ecosystem", *Proceedings of the 2013 conference on Internet measurement conference - IMC '13*, 2013.

[27] J. Amann, M. Vallentin and R. Sommer, "Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service", 2012.

[28] A. Ranjan, V. Kumar and M. Hussain, "Security analysis of TLS authentication", *2014 International Conference on Contemporary Computing and Informatics (IC3I)*, 2014.

[29] D. Naylor, A. Finamore, I. Leontiadis, Y. Grunenberger, M. Mellia, M. Munaf, K. Papagiannaki and P. Steenkiste, "The Cost of the "S" in HTTPS", *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies - CoNEXT '14*, 2014.

[30] V. Nisha, L. Aliyar and A. Ali, "An overview of cryptographic solutions to web security", *2010 IEEE International Conference on Computational Intelligence and Computing Research*, 2010.

[31] Y. Zou and G. Wang, "Intercept Behavior Analysis of Industrial Wireless Sensor Networks in the Presence of Eavesdropping Attack", *IEEE Transactions on Industrial Informatics*, vol. 12, no. 2, pp. 780-787, 2016.

[32] E.W. Felten, D. Balfanz, D. Dean and D.S. Wallach, "Web spoofing: An internet con game", *Software World*, vol. 28, no. 2, pp.6-8, 1997.

[33] N. Nikiforakis, W. Meert, Y. Younan, M. Johns and W. Joosen, "SessionShield: Lightweight Protection against Session Hijacking", *ESSoS*, 11, pp.87-100, 2011.

[34] J. Reschke, "The'Basic'HTTP Authentication Scheme", No. RFC 7617, 2015.

[35] D. Peng, C. Li and H. Huo, "An extended UsernameToken-based approach for REST-style Web Service Security Authentication", *2009 2nd IEEE International Conference on Computer Science and Information Technology*, 2009.

[36] P. Sturgeon, *Build APIs You Won't Hate*. [S.l.]: Leanpub, 2015.

[37] F. Fiedler, "HTTP Authentication: Basic and Digest Access Authentication RFC 2617 Obsoletes RFC 2069", 2015.

[38] R. Shekh-Yusef, D. Ahrens and S. Bremer, "HTTP Digest Access Authentication", No. RFC 7616, 2015.

[39] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions", *Lecture Notes in Computer Science*, pp. 19-35, 2005.

[40] E. Hammer-Lahav, "The OAuth 1.0 protocol", No. RFC 5849, 2010.

[41] E. Chen, Y. Pei, S. Chen, Y. Tian, R. Kotcher and P. Tague, "OAuth Demystified for Mobile Application Developers", *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS '14*, 2014.

[42] A. Niruntasukrat, C. Issariyapat, P. Pongpaibool, K. Meesublak, P. Aiumsupucgul and A. Panya, "Authorization mechanism for MQTT-based Internet of Things", *2016 IEEE International Conference on Communications Workshops (ICC)*, 2016.

[43] D. Hardt, "The OAuth 2.0 Authorization Framework", No. RFC 6749, 2012.

[44] T. Lodderstedt, M. McGloin and P. Hunt, "OAuth 2.0 Threat Model and Security Considerations", No. RFC 6819, 2013.

[45] M. Darwish and A. Ouda, "Evaluation of an OAuth 2.0 protocol implementation for web server applications", *2015 International Conference and Workshop on Computing and Communication (IEMCON)*, 2015.

[46] D. Aas, "Authentication and Authorization for Native Mobile Applications using OAuth 2.0", 2013.

[47] X. Brustoloni and J. Brustoloni, "Hardening Web browsers against man-in-the-middle and eavesdropping attacks", *Proceedings of the 14th international conference on World Wide Web - WWW '05*, 2005.

[48] P. Zhou and X. Gu, "HTTPAS: active authentication against HTTPS man-in-the-middle attacks", *IET Communications*, vol. 10, no. 17, pp. 2308-2314, 2016.

[49] R. Oppliger, R. Hauser and D. Basin, "SSL/TLS session-aware user authentication Or how to effectively thwart the man-in-the-middle", *Computer Communications*, vol. 29, no. 12, pp. 2238-2246, 2006.

[50] S. Stricot-Tarboton, S. Chaisiri and R. Ko, "Taxonomy of Man-in-the-Middle Attacks on HTTPS", *2016 IEEE Trustcom/BigDataSE/ISPA*, 2016.

[51] S. Kaka, V. Sastry and R. Maiti, "On the MitM vulnerability in mobile banking applications for android devices", *2016 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, 2016.

[52] "zANTI - Mobile Security Risk Assessment — Zimperium", *Zimperium.com*, 2017. [Online]. Available: https://www.zimperium.com/zanti-mobile-penetration-testing. [Accessed: 18- Nov- 2017].

[53] S. Prez, F. Durao, S. Meli, P. Dolog and O. Daz, "RESTful, Resource-Oriented Architectures: A Model-Driven Approach", *Lecture Notes in Computer Science*, pp. 282-294, 2011.

[54] L. Richardson and S. Ruby, *RESTful web services*. Beijing: O'Reilly, 2007.

[55] "Packet Capture - Android Apps on Google Play", *Play.google.com*, 2017. [Online]. Available: https://play.google.com/store/apps/details?id=app.greyshirts.sslcapturehl=en. [Accessed: 18- Nov- 2017].

[56] F. Callegati, W. Cerroni and M. Ramilli, "Man-in-the-Middle Attack to the HTTPS Protocol", *IEEE Security Privacy Magazine*, vol. 7, no. 1, pp. 78-81, 2009.

[57] K. Benton and T. Bross, "Timing Analysis of SSL/TLS Man in the Middle Attacks", *Computing Research Repository*, 2013.

[58] A. Fung and K. Cheung, "HTTPSLock: Enforcing HTTPS in Unmodified Browsers with Cached Javascript", *2010 Fourth International Conference on Network and System Security*, 2010.

[59] J. Hodges, C. Jackson and A. Barth, "HTTP Strict Transport Security (HSTS)", No. RFC 6797, 2012.

[60] M. Kranch and J. Bonneau, "Upgrading HTTPS in mid-air: An Empirical Study of Strict Transport Security and Key Pinning", *Proceedings 2015 Network and Distributed System Security Symposium*, 2015.

[61] M. Elboukhari, M. Azizi and A. Azizi, "Improving TLS Security By Quantum Cryptography", *International Journal of Network Security Its Applications*, vol. 2, no. 3, pp. 87-100, 2010.

[62] D. Scott and R. Sharp, "Abstracting application-level web security", *Proceedings of the eleventh international conference on World Wide Web - WWW '02*, 2002.

[63] P. Kalkman, "Basic Authentication on a WCF REST Service - CodeProject", *Codeproject.com*, 2018. [Online]. Available: https://www.codeproject.com/Articles/149738/Basic-Authentication-on-a-WCF-REST-Service. [Accessed: 08- Jan- 2018].

[64] P. Kalkman, "Digest Authentication on a WCF REST Service - CodeProject", *Codeproject.com*, 2018. [Online]. Available: https://www.codeproject.com/Articles/162726/Digest-Authentication-on-a-WCF-REST-Service. [Accessed: 08- Jan- 2018].

[65] "Spring Social Reference", *Docs.spring.io*, 2018. [Online]. Available: https://docs.spring.io/spring-social/docs/1.1.0.RELEASE/reference/htmlsingle/section$_oauth2ServiceProviders$ $08 - Jan - 2018$].