International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 11 Issue: 03 | Mar 2024 www.irjet.net p-ISSN: 2395-0072

Improved the authentication technology on Single Sign-On Protocol "OpenIDConnect" to avoid session hijacking attacks

Ibrahim Alahmad [1], Mouhamad Ayman Naal [2], Mahmoud Shaar [3]

[1]First Graduate student (PhD), Department of Computer Engineering, College of Electrical Engineering And electronic, Aleppo University, Syria
[2]Professor in the Department of Computer Engineering, Faculty of Electrical and Electronic Engineering, University of Aleppo, Syria
3Lecturer in the Department of Computer Engineering, Faculty of Electrical and Electronic Engineering, University of Aleppo, Syria

---

Abstract - *Millions of users routinely use Google or Facebook to log in to websites that support the OpenID Connect single sign-on protocol. So the security of this protocol is therefore of critical importance. As revealed in previous studies, systems using OpenID Connect are vulnerable to attack as the users of these systems are typically unaware of these issues and are therefore at risk of attacks that may lead to unauthorized access to user accounts.*

In this paper, a large-scale practical study of the Single Sign-On Protocol "OpenID Connect" was carried out by analyzing the stages of the protocol's work. A group of attacks that were developed in order to penetrate the protocol were also defined, and the session attack was applied to the protocol, and then a new method was proposed to prevent the session attack in the first stage of the protocol by making two-factor authentication method by using the One Time Password to protect session tokens.

Key Words: **Single Sign-On, One Time Password, One Time Pad, OpenID Connect, identity provider**.

1.INTRODUCTION

Single sign-on is a method of access control that requires the user to log in once and allows him to access multiple resources and services after successfully logging in without requiring him to log in again. Thus, the SSO approach allows users to authenticate only once and then enjoy easy access to other applications securely [1]. The OpenID Connect protocol is the latest version of single sign-on protocols. This protocol was released in 2014 and has been supported by many large companies [2].

2. RESEARCH OBJECTIVES

This research aims to provide a comprehensive and detailed analytical study of the OpenID Connect single sign-on protocol and the stages of its work in detail, in addition to studying the malicious attacks that have been developed in order to uncover security vulnerabilities in this protocol. Session attacks will also be applied to the protocol and the results of the attack will be observed. The research has presented a method to protect the protocol. From session attack by using OTP where done.

3. OPENID CONNECT

The OpenID Connect protocol was built based on OAuth to enable user authentication. It also includes a new feature in it, which is the stage of dynamic registration, discovery, and automatic trust achievement between the service provider (RP) and the identity provider (Idp), through which the service provider can automatically discover the identity provider (IDP) responsible for Giving identity. The following figure (1) shows the general working principle of OIDC [3].

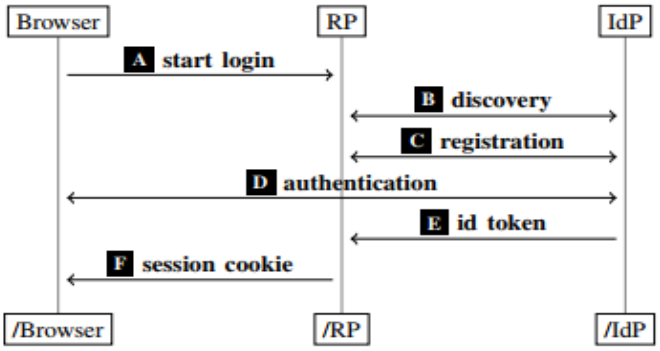


Figure (1) The general principle of OpenID Connect

Where the user begins by requesting the service from the service provider (RP)(A), then the service provider retrieves (obtains) some information (URLs), and this is called the discovery phase (B). Then he registers with the identity provider (IDP) (C), and then the user goes For the identity provider to authenticate itself (D), after which the identity provider issues an Id_token to the service provider, which verifies it to provide the service to the user (E).

The OpenID Connect protocol allows users to authenticate to service providers using existing accounts at the identity provider. OIDC was designed based on the OAuth protocol to enable user authentication. It also includes a new feature in it, which is the stage of dynamic registration, discovery, and

automatic trust establishment between the service provider (SP) and the identity provider (Idp) [4].

There are three different parties within the OIDC protocol. The relationship between these parties is shown in Figure (2).

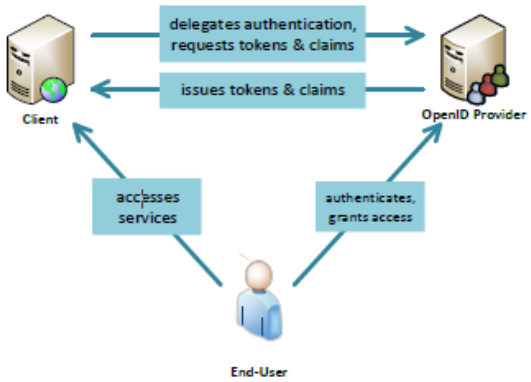


Figure (2) The relationship between the three parties in the OIDC Protocol

Each of these parties assumes a different role as follows:

1. End-User: wants to access the services of the client (service provider) and therefore needs to prove his identity to the client. It also has the possibility to authorize the client to access a specific set of its resources stored in the identity provider (OP).

2. Client: An application that provides certain services that require authentication to the end user. The authentication process is delegated to the identity provider (OP).

3. OpenID Provider: Handles the authentication of the end user and issues an authentication token that contains a specific set of claims that prove the end user's identity. In addition, it can issue a license code in order to allow the client to access end-user resources.

4. STAGES Of THE OPENID CONNECT PROTOCOL

The stages of the protocol are illustrated in Figure (3) [3]. Initially, the user begins the login process by entering the email address, which he sends to the RP service provider in step 1. After that, the service provider uses the discovery service to collect information from the identity provider (IDP), where the service provider uses the WebFinger mechanism to discover information from the identity provider. For this discovery the service provider contacts the server as described in step 2 which responds by sending the address of the identity provider in step 3.

The discovery process continues by the service provider by sending a request to obtain the settings in step 4, and the server responds in step 5 by obtaining all the information from the endpoint addresses that will be used in the stages of the protocol. It also requests and obtains the address of the public key used in encryption, as shown in steps 6 and 7. Here the discovery phase ends and the registration phase begins in steps 8 and 9, where a registration request is sent to the registration endpoint address, and the response is completed by sending secret codes (client_id, client_secret). It is private between the service provider and the identity provider. Here the first stage of the protocol's work ends and then the basic part of the protocol begins: The service provider redirects the user's browser to the server in step 10. The redirection process contains information about the authorization code, the identifier (client_id) of the service provider, and the status value. Then this data is sent by the browser to the identity provider in step 11. The authentication process takes place between User and server in steps 12 and 13. Then it is directed to the service provider in steps 14 and 15, where the status value and the authorization code are matched by the service provider. If they are correct, it contacts the identity provider in step 16 at the endpoint of the codes and sends him the authorization code and secret codes (client_id, client_secret). If they are If correct, the identity provider responds and sends the tokens (access_token, id_token) in step 17 to the service provider, which verifies them and authenticates the user's login process.

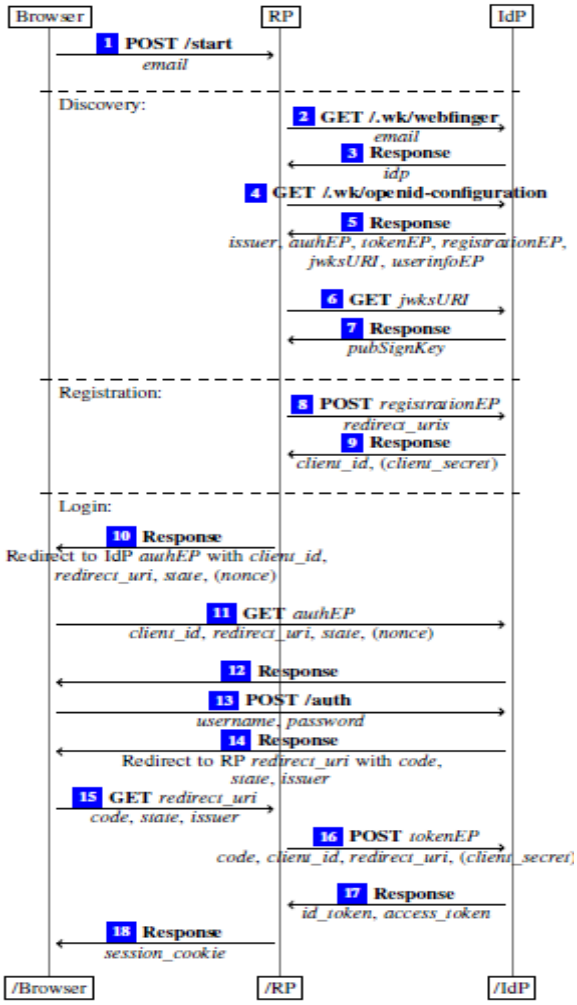


Figure (3): Detailed stages of the OpenID Connect protocol

An id token is a secret token that contains claims about the end user's identity and its JSON Web Token (JWT) data structure. Figure (4) shows an example of an id token [4][5].

Figure (4) is an example of an id token

We note that the id token consists of three parts:

First: The header: contains information including the encryption algorithm used.

Second: The body: It contains the information needed to authenticate the end user and includes:

• End user identity: It consists of two parts:

o Issuer (iss): To find out the identity provider.

o Subject(sub): To know the user's identity.

Both are used to identify the end user.

• The timestamp(iat) and expired(exp) define the time period when the code was produced and expired.

• Nonce: A random string sent by the client during the authentication request used to mitigate attacks.

• Audience(aud): To determine which customers the identity code belongs to.

Third: Signature: It provides a reliable id token.

5. RELATED WORK

Many previous studies have focused on single sign-on protocols by developing techniques to test, analyze, and identify security vulnerabilities in them. Below we review the most important of these studies.

The researchers conducted an analysis of the single sign-on protocol against impersonation attacks, where a technology was developed based on opening a secure communication channel between the service provider and the identity provider, which relies on redirecting the user to the identity provider to prevent the attacker from impersonating the identity of the client from whom the user is requesting a service. The performance was tested and evaluated. The protocol showed a time delay of 650 milliseconds and additional costs in the hard entity, but it provided strong protection and confidentiality in a clear and acceptable manner[6].

A confidential and in-depth analysis of the OpenID Connect single sign-on protocol was conducted and a developed model of OpenID Connect was built, where the secret properties of the protocol were employed to avoid previously discovered attacks and new attacks[3].

The security properties of the GoogleOIDC protocol were studied and examined on a group of clients. A type of attack was applied through which user codes could be obtained and then used to impersonate the user to reach clients. They also presented a set of future advice that should be taken into account in future systems for both providers. Service ''RP and Identity Provider op [7].

By conducting an in-depth analysis of the new feature in the OIDC protocol, which is the dynamic registration and discovery phase, and defining general attack concepts on the OIDC Single Sign-On Protocol, they presented a new type of attack (Malicious Endpoints) that belongs to the class of second-degree attacks, where the attack consists of two stages. This attack exploits the information exchanged between parties to the protocol between the three stages. They were able to break the user's privacy and explain the defects of the new features, and therefore the security of the protocol cannot be guaranteed in its current form, as the attacker in these attacks (Malicious Endpoints Attacks) seeks to steal documents between the client (Client) and the identity provider (OP) and also seeks to steal the authorization code (token_access) that Allows access to user resources[4].

By providing a tool for testing attacks on the OIDC protocol and testing sophisticated attacks (malicious endpoint attacks) in which the identity provider is discovered in the discovery phase, this helped in better understanding the flow of messages within the protocol [8].

The researchers also analyzed the known attacks on the OIDC protocol and classified these attacks into two basic categories:

Single-stage attacks: These are attacks that are applied to one of the stages of the protocol.

Two-stage attacks: These are attacks that rely on more than one stage of the protocol during the attack.

The researchers have presented a tool to improve the confidentiality of the protocol and identify security vulnerabilities within it.

Building the OIDC protocol from several layers allows attackers to intervene between any two of the three stages, which allows for a deeper study of these stages and the possibility of hacking them [9].

By secretly analyzing a group of attacks such as impersonation attacks, phishing attacks, and replay attacks on the OAuth protocol, they found that there are vulnerabilities during user authentication. A method is proposed to securely authenticate a user by verifying the authorization code using an email authentication server through access token distribution. Although the attacker obtains some user information, he cannot use it for authentication [10].

Researchers have identified the causes of the cross-site forgery problem in OAuth 2.0 and OIDC and proposed a new approach of mitigations that can be applied by combining the referrer header with the different identity provider URLs used by the service provider in order to prevent CSRF redirection attacks[11].

Researchers have proposed a way to protect OIDC URLs from session hijacking attacks by adding additional innovative parameters during the session such as a special alphanumeric string (SAS) and a private security PIN (SSP).

The results show that adding the above parameters reduces the risk of session hijacking in an OIDC communication environment [12].

By conducting the first formal analysis of the Open Banking Application Programming Interface (FAPI) based on the web infrastructure model, a comprehensive model file was built that includes all participants in the protocol. The security properties were identified, namely authentication, authorization, and session integrity. While trying to prove these properties, they found many security vulnerabilities that enable the attacker to Access protected resources or perform session integrity attacks. Methods have been proposed to mitigate these attacks and thus prove the security of the OpenID FAPI [13].

OAuthGuard is a scanner and protector that works with ISPs using OIDC. The tool was used to scan a group of sites in search of possible security vulnerabilities. Among these sites that had security vulnerabilities, the OAuthGuard tool was able to protect the user's security and privacy for some of them and warn the user that he was using an unsafe application on other sites [14].

A dynamic identity management model based on the OIDC protocol has been proposed. The model consists of three steps: discovering the service provider that indicates the location of the partner organization, registering the authorized party with the OIDC, which allows the organization and its partners to negotiate information to create the federation, and creating a trust federation dynamically. This model allows providing services to hundreds of users from different organizations and thus saving time through dynamic federation during operation[15].

6. SESSION ATTACKS ON OPENID CONNECT

The attacks developed on the OpenID Connect single sign-on protocol, which exploit the mechanism of the protocol through several stages, are among the attacks that take place through several stages, as they exploit the presence of the new stage in the protocol, which is the dynamic registration and discovery stage.

Most web applications need to save some objects between different requests. This way the web application saves certain data over subsequent accesses. The end user accessing the web application is assigned a unique session identifier, which is typically stored in a cookie.

In this section we will describe a new attack that results in authentication failure on the end user. The idea of the attack is to overwrite the elements stored in the session, which leads to security issues.

The basic setup for the attack is as follows:

1. The end user (victim) has an active account on the real client. We assume that the user trusts this client and that the client follows the rules of the OpenID Connect protocol.

2. The end user is registered with the identity provider Honest OP on the domain http://honestop.com. The user trusts this provider, which also follows the rules of the OpenID Connect protocol.

3. To carry out the attack, the attacker must set up his own discovery service that runs on the domain http://Malisious.com. The actions of this discovery service do not deviate from the normal protocol flow.

4. According to the attack model, the attacker does not have any control over the client, the end user, the service provider, or the traffic between these three elements. The attacker is able to send an HTTP request through the end user's browser, for example by embedding an image within the website that causes the browser to issue an automated request when the website is displayed.

The attack description:

We will describe the stages of the protocol attack flow as shown in Figure (5) [16].

Service provider data (Metadata Honest) is stored in the session. The attacker's intention in the first stage is to force the customer to use the underlying service provider Honest OP. For this purpose it creates a malicious link and stores it on the website.

By visiting the website containing this malicious link an HTTP request will be sent to the client through the end user's browser. As a result, the customer begins the discovery phase with the discovery service

http://honestop.com. The client sends a request, determines the endpoint addresses, and stores them in the current session. The end user authentication process takes place in the next stage where the client redirects the user to the token endpoint to authenticate himself and authorize the client. The second phase follows the regular protocol flow where there is no routing to malicious sites or network monitoring between the three parties.

The attacker's intent is to overwrite data stored in the session to force the client to use the attacker's malicious detection service. For this purpose, it sends a second request to the client containing the identity alice@malicious.com. The attacker forces the user's browser to send two HTTP requests. As a result, the client detects the malicious detection service, identifies endpoints, and replaces old session data with new data (Metadata EvilOP). The client then receives the code from the previous stage and sends it with its credentials (client_id, client_secret) to the endpoint of the codes stored in the attacker's session data. As a result, the attacker receives access tokens and is able to access the locked resources in the OP identity provider.

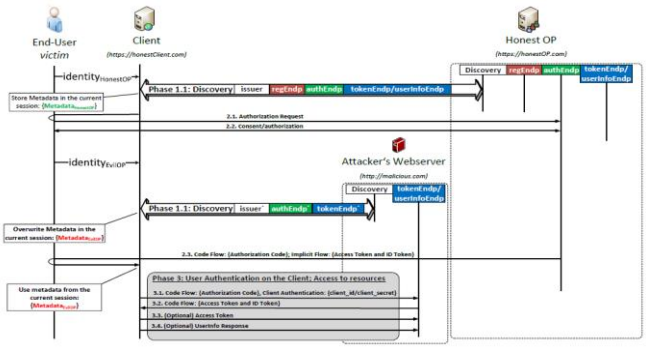


Figure (5) Stages of a session attack

7.APPLYING THE SESSION ATTACK

Attack assumptions:

1.User credentials are managed in one place.

2. User credentials are stored in the identity provider.

3.The attacker is redirected to the urls without any action from the user and this is called open redirection.

The attack was implemented on the localhost website using the Asp.net core framework.

SQL Server database was used.

A website was created that includes a login using OpenID Connect.

Figure (6) shows the user accounts registered on the server:

2	0b75826f-5dd2-4207-9ab4-cd0ab2078da6	user5@gmail.com	USER5@GMAIL.COM
3	220623db-563b-4da3-9b5e-9fb13e46b2f8	user2@gmail.com	USER2@GMAIL.COM
4	30851fb9-1900-4cdb-9046-783898a914f1	myuser@gmail.com	MYUSER@GMAIL.COM

Figure (6) User accounts on the server

Hash encryption technology protects passwords on the server, and the user's pages are navigated after successful login using explicit e-mail. This is the method used in the protocol in general, where e-mail is the only explicit and secure information in the database query process after the transition. To the user's personal page.

First, the protocol tokens obtained through a mix-up attack are used to obtain information related to the user, such as his token ID, which is shown in Figure (7).

Name	Value	Domain	Path
.AspNetCore....	CfDJ8B112XR wJAtCiLpXPde chon0NR9Bk_ P3Qp28ziQIz3 nmeCDsqB8Z9 FuWs_wXEZvy xONp- Hbcx31omh2S F7- jRf1KcH44zEw AWRPtYItZXD CkhEAwfPAGS	localhost	/

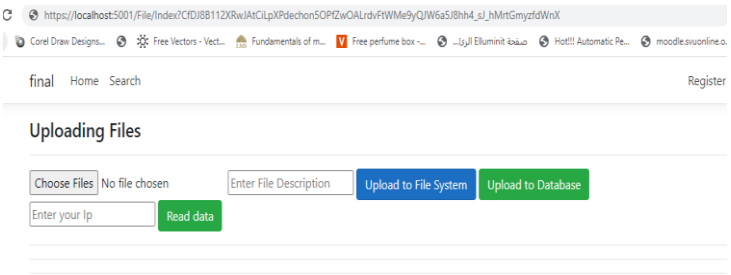
Figure (7) id token values after applying the attack

The url is then obtained from the tokens and the token stored in the value generated by the attack application is used to form the user's access token.

Then a request is made to create a session from the “service provider” server.

```
<input name="__RequestVerificationToken" type="hidden"
    value="CfDJ8B112XRwJAtCiLpXPdechon5OPfZwOALrdvFtWMe9yQJW6a5J8hh4_sJ_hMrtGmyzfdWnX_cBWyOTYc8v
    FBBqdiEF1uexGNff0fhbVps_RHDidJ0u-HDGb5gdf8UIeqAheWzURve_8aZJO2VllzoM" /></form>
```

The Access token is used to access user data after forming the full URL for the protocol



Here user data was accessed without logging in.

The results of the attack and its impact on the protocol are limited to obtaining the user’s access token that the attacker

will use in order to access the protocol, as he will pass the protocol's protection and access the user's data, which means losing the user's data and account, and this means achieving large losses for users when working with the protocol, and this is what makes us need to use Techniques that prevent the attacker from forming an access token URL so that the session is protected by adding codes that change for each session and are uniform between the parties to the protocol.

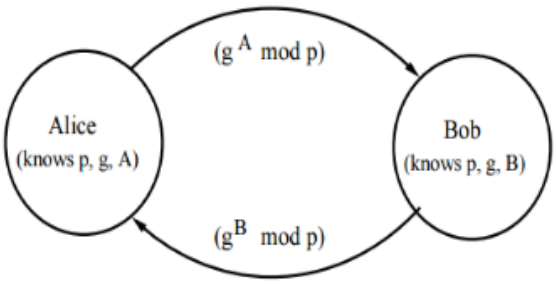
8. PROPOSE OPENID CONNECT ARCHITECTURE

This section presents and discusses the proposed OpenID Connect Architecture. The proposed approach has three components: DIFFIE-HELLMAN KEY EXCHANGE, ONE TIME PASSWORD, ONE TIME PAD . These components are further elaborated.

8.1 DIFFIE-HELLMAN (DH) KEY EXCHANGE ALGORITHM

Diffie-Hellman key exchange is a method of securely exchanging cryptographic keys over a public channel that allows two parties to create a shared secret key over an unsecured communication channel and then use this key to encrypt subsequent communications using a symmetric key.

Diffie-Hellman key exchange works by allowing two parties (Alice, Bob) to agree on a shared secret key over an insecure channel without any other party being able to intercept the key or learn anything about it. Keys are exchanged according to the following steps shown in Figure (8)[17].



Figure(8) Steps for key exchange in Diffie-Hellman

Alice and Bob agree on two large prime numbers p and g and an algorithm for public key swapping.

Alice chooses a secret integer a and then sends Bob the A=g^amod p

Bob chooses a secret integer b and then sends to Alice B=g^bmod p

Alice calculates s=B^amod p and Bob calculates s=A^bmod p

Alice and Bob then have shared secret keys that can be used to create a secure communication channel.

The security of Diffie-Hellman key exchange is based on the fact that it is computationally infeasible for an attacker to determine the shared secret keys from the public values of p, g, A, and B. This allows Alice, Bob to exchange the key securely, even over an insecure channel.

8.2 ONE TIME PASSWORD (OTP) ALGORITHM

A one-time password is an automatically generated, one-time string of numbers and letters that authenticates a user for a single transaction or login session.

A one-time password is more secure than a static password, especially a user-generated password that can be weak or reused across multiple accounts. A one-time password may replace login information for authentication, plus it can be used as an additional layer of security for an account. Or similar online entity after the password. If the password is exposed, it cannot be reused and the next login by the user will require a completely different authentication code because it changes with each login. One-time password has several forms [18]:

1- Time Synchronization: Depending on the time comparison between the dynamic token and the dynamic password, the time synchronization token usually generates a new password every 60 seconds, which meets the strict requirements for authentication.

2- Event Synchronization: The server and token use the specified sequence of events and the same initial value to calculate the password using the HASH algorithm.

3- Challenge/Respond: The server generates a challenge number (chg) and sends it to the token, which both calculates the password using the challenge number and some pre-arranged parameters.

8.3 ONE TIME PAD (OTP) ALGORITHM

OTP is an unbreakable encryption system if used correctly. A randomly generated private key or pad is used only once to encrypt a message of the same length as the ciphertext which is then decrypted by the receiver using a matching one-time pad and key. Each bit or character of the plaintext (unencrypted) is encrypted by a combination of standard arithmetic with a bit or character of a secret random key (or pad) of the same length as the plaintext, to produce the ciphertext. Messages encrypted with random-based keys have the advantage that there is no way in theory to break the code by analyzing a string of messages. Each cipher is unique and unrelated to the next, making it impossible to detect a pattern[19].

The single-pane encryption scheme is simple yet powerful with absolute security unmatched by modern encryption algorithms. There are a set of rules that must be followed so that the one-time pad encryption cannot be broken, which are:

• The key must be as long as the encrypted data or message.

• The key is generated randomly.

• Each key is used only once and the sender and receiver must destroy their keys after use.

• There should be only two copies of the key: one for the sender and one for the receiver.

9.Improving authentication technology in OIDC:

Figure (9) shows a sample flow of the authentication protocol proposed in OpenID Connect:

By the RP service provider:

• RP provides OP with the email explicitly because it is not confidential data.

• Encrypted data, including the hash password and one-time password codes, is exchanged between the two parties.

• OP and RP contain two identical copies of the One Time Pad algorithm, and data is not exchanged between them, but email is used as the encryption key.

From the OP identity provider:

• Receives email from RP and encrypted data from one time password.

• Provides the RP with access to resources.

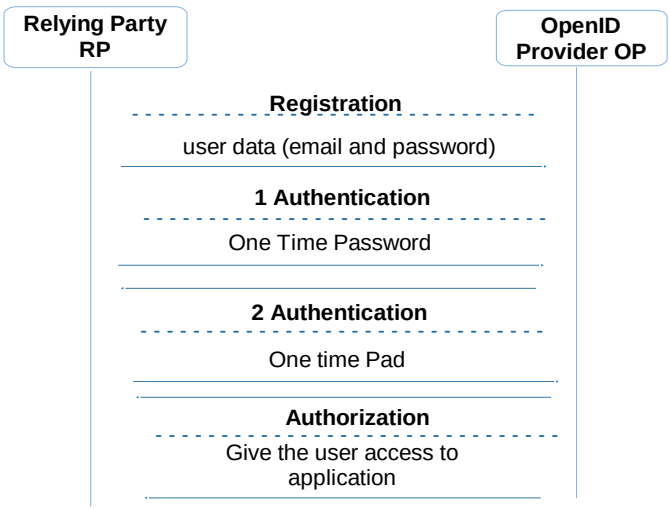


Figure (9) The flow of the proposed authentication protocol in OpenID Connect

10. RESULTS

The proposed method added new symbols to the session that prevented the attacker from forming the access url. On the other hand, the new method caused a time delay in the operation of the protocol due to the time added to form and authenticate the symbols. Figure (10) shows the results of the time delay that occurred after adding the proposed method on the first two sites. It is a ready-made MVC model provided by Microsoft and runs on its ASP.NET Core platform. The second is a file management system that we created using the same platform.

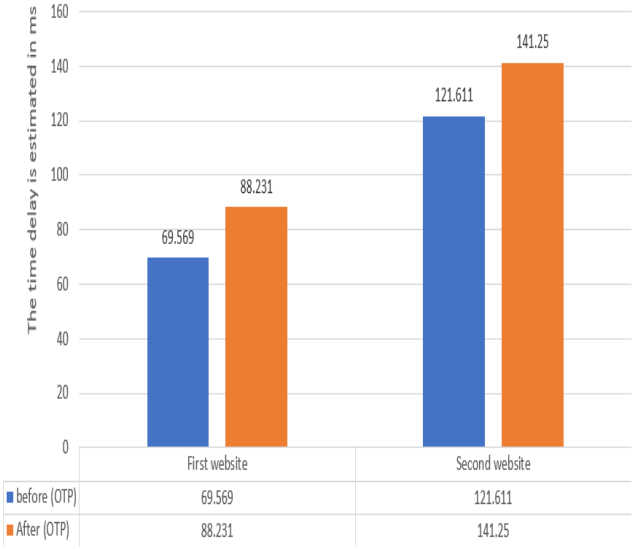


Figure (10) Time delay results before and after applying the OTP technique

The proposed method for the authentication process enables the user to protect the user's access token from session attacks by generating a new access token every time a session is created.

Using the proposed method to improve the performance of the protocol, make the value of the session expiration period meaningless, which is considered one of the codes extracted from malicious point attacks and which the attacker uses to know the validity of the session in order to access the access token.

The proposed method increases the time complexity of the protocol due to the addition of time to the one time pad algorithm, which creates a new access token for the user in each session.

The resulting time delay has an almost negligible impact on the added protection value. The reason is due to the simplicity of the one time pad algorithm from a mathematical standpoint, in addition to the fact that the protocol is intended to work on servers with high hardware.

11. CONCLUSIONS

The research presented a new method to prevent session attacks by adding codes that prevent the attacker from creating an access token URL for the victim by generating permanently variable and encrypted codes for each session. The proposed method increased the protection of the sites, but it caused a time delay. We will work during the next stage to reduce the delay. Time delay by forming additional symbols that are derived from the current session and the previous session, which means reducing the time delay.

REFERENCES

[1] S,Suoranta, K,Manzoor, A,Tontti, J,Ruuskanen and T,Aura, "Logout in single sign-on systems: Problems and Solutions", journal of information security and applications, 2014, pp. 61–77.

[2] N. Naik and P. Jenkins , "An Analysis of Open Standard Identity Protocols in Cloud Computing Security Paradigm" , IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, 2016, pp. 428–431.

[3] Fett,D, Küsters,R and Schmitz,G, "The Web SSO Standard OpenIDConnect: In-DepthFormal Security Analysis and Security Guidelines", IEEE 30th Computer Security Foundations Symposium (CSF), 2017, pp. 189-202.

[4] Mainka,C, Mladenov,V and Schwenk,J. " On the security of modern Single Sign-On Protocols – Second-Order Vulnerabilities in OpenID Connect",2016.

[5] Fett,D, Küsters,R and Schmitz,G, "A Comprehensive Formal Security Analysis of OAuth 2.0", In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 1204–1215.

[6] Yinzhi Cao, Yan Shoshitaishvili and Kevin Borgolte, "Protecting Web-based Single Sign-on ProtocolsagainstRelying Party ImpersonationAttacksthrough a Dedicated Bi-directionalAuthenticated Secure Channel" , Research in Attacks, Intrusions and Defenses, 2015, pp 276–298.

[7]] Li,W and Mitchell,C." Analysing the Security of Google'si mplementation of OpenIDConnect", International Conference on Detection of Intrusions and Malware, and Vulnerability Assessmen, 2016, pp. 357–376 .

[8] Mainka,C, Mladenov,V and Schwenk,J." Do not trust me: Using malicious IdPs for analyzing and attacking Single Sign-On", IEEE European Symposium on Security and Privacy (EuroS&P), 2016, pp. 321-336 .

[9] Mainka,C, Mladenov,V, Schwenk,J and Wich,T, " SoK: Single Sign-On Security – An Evaluation of OpenID Connect", IEEE European Symposium on Security and Privacy (EuroS&P) , 2017, pp. 251-266 .

[10] C,Chae, K,Kim and H,Cho , "A study on secure user authentication and authorization in OAuth Protocol", Springer Cluster Computing volume 22, 2017, pp. 1991–1999.

[11] W. Li, C.J. Mitchell, and T. Chen, "Mitigating CSRF attacks on OAuth 2.0 and OpenID Connect", http://arxiv.org/abs/1801.07983.

[12] M,Bilal, M,Asif and A,Bashir, "Assessment of Secure OpenID-Based DAAA Protocol for Avoiding Session Hijacking in Web Applications", Security and Communication Networks Volume 2018, Article ID 6315039, 10 pages, https://doi.org/10.1155/2018/6315039.

[13] D. Fett, P. Hosseyni, and R. Kusters , "An Extensive Formal Security Analysis of the OpenID Financial-grade API", IEEE Symposium on Security and Privacy (SP), 2019, pp. 1054–1072.

[14] W. Li, C.J. Mitchell, and T. Chen, "OAuthGuard: Protecting User Security and Privacy with OAuth 2.0 and OpenIDConnect",2019,https://www.researchgate.net/publication/330672797.

[15] A,Alsadeh, N,Yatim, Y,Hassouneh, "A Dynamic Federated Identity Management Using OpenID Connect", Future Internet 2022,14, 339. https://doi.org/10.3390/fi14110339.

[16] Mladenov,V and Mainka,C , "OpenID Connect Security Considerations",https://www.nds.ruhr-uni-bochum.de/media/ei/veroeffentlichungen/2017/01/13/OIDCSecurity_1.pdf, as of September 30, 2020.

[17] Nan Li, "Research on Diffie-Hellman Key Exchange Protocol", Information Engineering Teaching and research section, 2010.

[18] M, Abousteif, A,F,Tammam and A,Wahdan," A Novel Approach For Generating One-Time Password With Secure Distribution", Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), 2020.

[19] E,M,Manucom, B,Gerado and R,Medina, "Analysis of Key Randomness in Improved One-Time Pad Cryptography", IEEE 13th International Conference on Anti-counterfeiting, Security, and Identification (ASID), 2019.