# Securing web application with Two Factor Authentication

Mahith Madwesh
M.Tech Computer Network
Engineering, BMSCE, Bsavanagudi,
Bengaluru - 19
mahith.scn17@bmsce.ac.in

Dr. Radhika K R
Professor, Department of ISE
BMSCE, Bsavanagudi,
Bengaluru – 19
rkr.ise@bmsce.ac.in

Srilakshmi Bandaru
Principal Engineer, Microchip
Technology,
srilakshmi.bandaru@microchip.com

***Abstract*- Traditional username and password were used to make the web applications secure which does not apply to the current state of internet. In case the attacker gets the username and password, an undoable damage can be done. To address this issue Two factor Authentication has been proposed to improve the security where smart devices are used as the second authentication factor. A web application is an application that keeps running on more than one PC and communicates through a system or a server. Web programmers today are challenged with making large applications and having to make the right choice of which logging to use for better application troubleshooting and to have secure applications .This paper presents an approach for web application security with Two Factor Authentication(2FA) provided by the Identity package and logging using open source or proprietary software. The approach involves three major aspects namely: Two Factor Authentication (2FA) and Authorization (Roles and policy). For this approach, a web application was developed in ASP.NET Core 2.0 and its extensions. The model-View controller (MVC) structure design was utilized in building up the web application, and the parameters for the methodology are cost of usage, program similarity, reaction time and platform compatibility.**

***Index Terms*- Web Application, Identity, Authorization, MVC, ASP.NET Core 2.0**

## I. Introduction

A web application is an application that keeps running on more than one PC and communicates through a system or a server. A web app is accessed with a browser as a client and provides the ability to update and maintain a program without deploying and installing software on computers. Web applications are utilized for web mail, online retail deals, exchange sheets, weblogs, internet banking, among others. One favourable position of a web application is that it very well may be accessed to and utilised by a huge number of individuals simultaneously.

Web Applications run on the internet or intranet and have become a very important part of any business today.

The rising numbers of users of Internet technologies today are capable of handling a large amount of users at a time. However, due to the openness, versatility and multi service of the internet, network security issues have become more and more serious. According to the statistics from China Internet Network Information Centre (CNNIC) [1], in 2015, 42.7% of Internet users encountered network security problems.

User authentication is establish a trusted connection between the user and the server and has become major issue for cyber-security. The single factor authentication with “username + password” has been widely used because it is easy to use and deploy without additional overhead. This has become an issue currently. Key loggers or Trojans installed on client’s computer can track the keys pressed and send them to the attacker without a hint, compromising major security risks. With advent of new concepts in technology, the mouse clicks being clicked on the screen with pixel information can be tracked for which virtual keyboards are also not secure. In 2011, China’s largest website Chine Software Developer Network(CSDN) was hacked and account information of more than 6 million user were leaked and spread quickly via the internet [2]. In May 2016, Google announced to completely cancel the password. Hence single factor authentication is completely unsuitable.

In order to increase the security, Two Factor Authentication (2FA) was proposed. 2FA is method of verifying user’s identity with two different factors: 1) user’s username and password, and 2) something they have, or something they are, i.e., a combination of passwords and physical entities such as smart cards [3], mobiles [4], tokens or fingerprints. Compared with password based two factor based authentication is more secure. The second code is easy to use as the code can be sent to the mobile device the user carries all the time or any other for that matter. For dynamic token, method is a one-time password and provides the high security.

The functionality of websites has increased with latest advancements in Razor pages to include dynamic elements, processing of forms, perform calculations, etc..

ASP.NET Core platform allows the user to make enhanced full stack web applications with both client side development and server side scripting. Server side code is written in C#. Client side pages communicate with server usually using Ajax for faster communication by serializing the data into json format.

In Microchip Technology, worked with Srilakshmi Bandaru, developed a Quality Related Job Responsibilities application for monitoring and maintaining the employee performance used for internal audits. The application was developed and deployed only for internal systems as pilot project to measure and analyze the performance.

The approach of this paper is to provide a complete solution for two factor authentication, cookie based authentication and authorization based on roles and policies to the web application. Logging is the major challenge for web applications are providing security and Data protection. The paper presents a way to secure the application in case the application is to be deployed so the public user can access it. To achieve the above challenges the application is developed by using built-in extensions provided by ASP.NET Core 2.0.

Users register to an online service by signing up, customers trust the service with their passwords – betraying that trust can have serious impact on the business. Password security has improved over time as attacks have become more sophisticated and computing power and storage has become cheaper and more available. The earliest, easiest and quickest method of storing passwords were plain text. Then the concept of hashing was introduced. The password was made to go through a one way function that makes it impossible to get the password again. Hashed passwords are technically impossible to decrypt but not impossible to crack. Two users with the same passwords will have the same hash.

To increase the multifaceted nature of the hashed passwords, salts are utilized. These are irregular strings of information (created and put away at whatever point the secret key is changed) which are put in plain-message close by the hash, recovered on login and afterwards used to produce the secret phrase hash (HASH(salt + secret word)). By utilizing salts, an assailant needs to go through the whole procedure once per-client, instead of once for the whole client base. A short four-character alphanumeric salt multiplies the size of the rainbow table required to crack the passwords by $36^4$ (1,679,616). At GoSquared, we use 32-byte salts, so the multiplier is $16^{64}$ ($1.158\times10^{77}$). In reality, be that as it may, the multiplier is just the quantity of one of a kind salts in your framework (number of clients) as once your database has been undermined the attacker can see which salts are being used.

This method has been recommended for a long time, and it's still the go-to system for new applications. A strong hashing algorithm (such as scrypt, bcrypt or PBKDF2) along with a secure salt is usually enough to prevent attackers from cracking passwords (unless they're using common passwords) [4].

When a user registers in the application, based on the mail being used, the user will be categorized as User, Manager or Admin. The password will be stored in the database with Hashing implemented in JavaScript. Hashing is provided by default in ASP.NET Core Identity. Presently Identity by default provides PBKDF2 algorithm. The data protection code base includes a package Cryptography.KeyDerivation which contains cryptographic key derivation functions. This package is a standalone component and has no dependencies on the rest of the data protection system.

Submitting the form to the database is done asynchronously which avoids performance bottlenecks and enhance overall application responsiveness of the application. Form validation is also added which prevents user from submitting same form multiple times.

The implications are, the traditional single-factor based authentication faces a great thread to security if the password is leaked. The user will have no control of the security which can be very serious. In addition the proposed two factor authentication which prevents password guessing attacks, can be deployed to the applications which are used in internal networks can be deployed to the public.

This paper presents an approach for Two Factor Authentication (2FA), Cookie based authentication and Authorization based on roles and policies using the Identity of ASP.NET Core 2.0

## II. RELATED WORK

### *A. Hardware Token*

Hardware tokens are usually small, portable devices which have little or no user interface. Hardware tokens are used when a host computer has to be connected to the internal networks or for secure authentication over the internet.[ht_2008]. Initially one time passwords were sent to the user in a primitive way, either delivered directly to the person, or through credit card or emailed. Once all the one time passwords were over, the user had to get a new card with TAN-code for two factor authentication. The first type requires a connection to the computer's USB-port. The second type runs free from the Internet and public telephone networks. Thus, contactless OTP tokens are protected from any malicious software and the one-time passwords cannot be intercepted. Lately, there appeared the USB-tokens, which, although inserted into the connector, activate only at the touch of a button (eg, Yubikey). Even if there will be a virus on the computer, the latter will fail to infect this token and use it to intercept one-time passwords [5]. Time-based hardware tokens follow the standard of 2FA.The SecurID is stored as the second factor in the dongle. However, this mechanism requires the interaction between the user and the hardware token. Besides, the server needs to provide each

client with a separate hardware token, which makes the deployment of token expensive. It also requires each site having its own authentication devices and needs users to carry them.
The modern tokens use different algorithms of the one-time password generation: by event (HOTP), by time (TOTP), "challenge-response" (OCRA). These devices can provide strong authentication for the most important data exchange areas. Therefore, such 2FA mechanism is only used in e-government, online banking and other similar fields [6].

*B. Short-range Wireless Communication*
Short-range wireless communication implements the second factor authentication with Bluetooth, NFC or WiFi, which is able to reduce the interaction between the user and mobile phone. Bluetooth is the most widely used technique. The browser and the phone generate a challenge-response pair via Bluetooth to compute the distance between them. However, the Bluetooth API is no longer supported by current mainstream browsers. NFC is a new short-range wireless communication method for smart devices. However, the mainstream browsers do not provide APIs to support NFC devices, and most of personal computers are not integrated with NFC. Moreover, NFC requires the user to hold his mobile phone to complete the two-factor authentication. Therefore, the interaction between the user and the mobile phone is complicated. WiFi communication requires that the computer located in the same network as the mobile phone. The computer needs to use additional software to generate an access node so that the user's mobile phone can be connected to the network where the computer is located. Such method requires that the mobile application continuously monitors the login request sent by the browser, which would degrade the performance of smart devices.

*C. Short-range Ultrasonic*
The browser and mobile phone can communicate with each other by the short-range ultrasound which can be recognizable to the computer and not audible to the human ear. Due to the limited performance of the phone, this communication method can only produce highly directional near-field ultrahigh frequency sound waves [7], and the signal attenuation is quite fast. When this technique is deployed for authentication, mobile phones cannot use external devices such as earphones, and such high-frequency sound wave will have a health impact on children and animals [8].

*D. Location Information*
With the Global Position System (GPS) information, the server can detect whether the computer and phone are located in the same environment. GPS sensors have been deployed in mainstream smartphones but not in most of computers. Many APIs provided by the browser can obtain geolocation information to complete the login [9]. However, the geolocation information is not accurate. For example, when the device is located in a VPN network or an enterprise's large management network, the geolocation is not the real location of the device and can be obtained by attackers, which would result in an authentication failure.

*E. Software Token*
At the point when the PDAs just showed up, they were costly and just a couple could manage the cost of such device. However, after some time, the expense of the gadgets, just as the levies for versatile interchanges diminished fundamentally in this way expanding the quantity of the phones proprietors. The transitory passwords and TAN-codes were sent by means of SMS. SMS verification is as yet a genuinely regular methods for the OTP passwords conveyance. Google 2-Step authentication [10] is a ordinary programming token instrument dependent on client's telephone and confirmation code. The product verification code is sent by short message administration or an application running on the cell phone. Such component needs the client to validate the validation code on the cell phone to the login interface of the program. Sound-Proof [11] uses the sound of the environment as an authentication factor, but it does not consider the legality authentication of the mobile phone and suffers security issues in a silent environment.

The implications are, 2FA has gained much attention in academia and industry. However, due to the inconvenient interaction, it is inevitable to change the user experience and bring the security issues for authentication. This paper proposes a change in the 2FA which in which the user has to enter the second factor authentication code being shown in the application. The user primarily has to enter the 16 digit code in the settings into the app to configure the app. After setting up the backup password, the authenticator app can be used everytime a user wants to sign in. Sliding sessions is used to maintain the session validity of the user. Server side application development is rapidly rising, the developer are facing challenges in making the application secure with Authentication and authorization which are compatible with different browsers. While some are compatible, others are not.

## III. PROPOSED SYSTEM

The traditional single-factor authentication is based on username and password, which is easy to deploy but vulnerable to dictionary attacks, snooping and brute force attacks. Although using different passwords on different websites can improve the security, it brings a lot of trouble to users in terms of memory. In the critical need to plan a progressively secure validation system, two-factor confirmation (2FA) turns out, joining the secret phrase with the elements, for example, credit card, cell phone, token or unique finger impression. In any case, as visiting various destinations regularly require various tokens, when you have to visit numerous locales in the meantime, conveying a not insignificant rundown of tokens will be extremely troublesome. Therefore, compared with the single factor

authentication, two-factor authentication will bring more inconvenience to users when using different physical entities as authentication factors. Besides, the interactive between human and entities may allow an attacker to obtain the second factor through fraud.

The Two factor authentication proposed in this paper will be much easier to use as the One-time password will be present as an application in user's mobile phone. No extra hardware device for authentication needs to be used. The authentication is completely transparent to the user. This also addresses the security concerns of the single factor authentication.

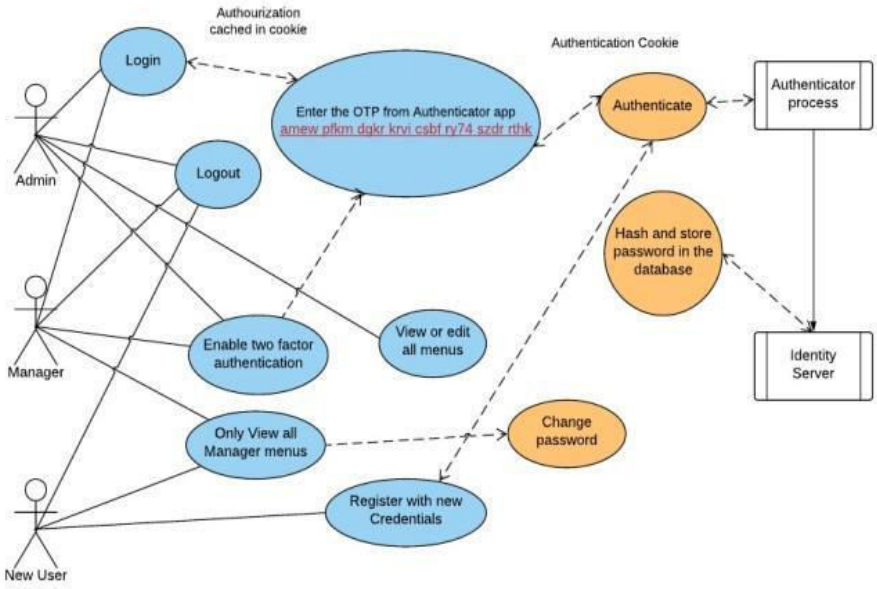



Fig. 1 Use case Diagram

*A. Design of Two Factor Authentication*

The proposed system is composed of registration module, login module, Two factor authentication module, database access module. The use case diagram is shown in Figure 1.

1) Login/Registration Module: This module is responsible for the authentication and authorization of the user when registering or logging to the application.

2) Two factor authentication module: The user will be able to enable the two factor authentication from account settings. User has to enter the 16 digit unique code enable two factor authentication.

3) Database access Module: This module is responsible for querying or inserting data in the database during user login or registration.

This paper proposes a transparent two factor authentication scheme. The function of browser is to implement login and registration module. The server side is used for authentication and authorization. The mobile application configured with the application is used for second factor authentication. The whole authentication flow is shown in Figure 2.

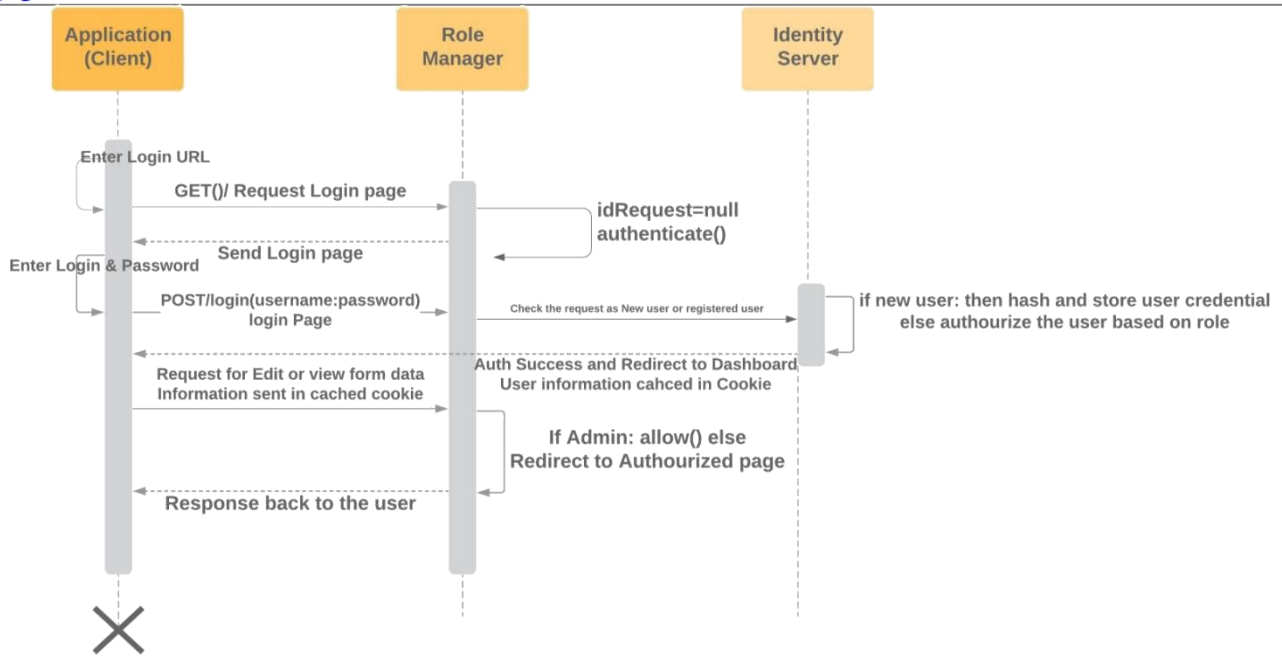



Fig.2 Sequence flow of authentication

1) The browser sends the username and password to the server.
2) The server verifies if the username and password are legal. If legal, the page moves to the second factor page where the one time password from the application is to be entered.
3) The browser checks with the server if the code is correct. If correct, the user is allowed to login.
4) Depending on the user logged in and based on role different resources are made accessible.

*B. Password Security*
When the user registers with a new userID and password. The identity in ASP.NET Core uses the PBKDF2 algorithm to store the password in the database in the hashed form.

Password based Key Derivation Function 2 or PBKDF2 to put it plainly, is an encryption system, which fundamentally utilizes a secret word and controls it to produce a strong key which could be utilized for encryption and accordingly decryption . This article essentially centers around the .Net classes accessible to empower this and the utilization of AES encryption Algorithm to work with PBKDF2.

To generate the PBKDF2 password you need the following:

*1. Pseudorandom function*
.Net give the Rfc2898DerviceBytes class which is Pseudorandom work generator dependent on MCSHA1. The Rfc2898DerviceBytes takes your salt esteem, the secret key and an emphasis number as contributions, to create your PBKDF2 key.

*2. Salt*
Salt can be any arbitrary information. One arbitrary number generator which is crypto based and can come convenient in creating Salt is the RNGCryptoServiceProvider class.

*3. Iterations*
Iterations is the number of times the function repeats to generate the key. The higher the number the more secure the key is. But this brings some performance implications and hence you have to identify a number (preferably in thousands) which ensures maximum security with minimal performance cost for your application. It is also preferable to keep this as a random number rather than multiples of 1000 or 100. This is on the grounds that amid unscrambling to reproduce the secret key based encryption key a similar number of emphasess must be utilized. Consequently the more arbitrary the number is, the more troublesome it would be to recognize it and thus increasingly secure the information is.

Below code explains the use of Rfc2898DeriveBytes with AES encryption.

```
var encryptor = Aes.create();
```



```
var pdb = new Rfc2898DeriveBytes(passphrase, salt,
iterations); encryptor.Key = pdb.GetBytes(32); encryptor.IV=
pdb.GetBytes(16);
var memoryStream = new MemoryStream();
var cryptoStream = new CryptoStream(memoryStream,
                    encryptor.CreateEncryptor(), CryptoStreamMode.Write);
var dataBytes = new Encoding.Unicode.GetBytes("String to be encrypted");
cryptoStream.Write(dataBytes, 0, dataBytes.Length); cryptoStream.Close();


var encryptedText = Convert.ToBase64String(memoryStream.ToArray());
```

PBKDF2 is particularly helpful when there is a need to decrypt and have no mechanism to store vector or password.

*C. Security Analysis*

1) Guessing attacks: The security of T2FA stems from the inability of attackers to guess the second factor of victim's application at the time of the attack. It is difficult for attackers to guess the code by the victims phone since the 16 digit code is unique each time it is generated. Therefore, T2FA is immune to guessing attacks.
2) Phishing attacks: In single factor authentication the victim might receive a link to login. But the website may not simulate second factor authentication , meaning thieves can learn the password but cannot access data.
3) Man-in-the-Middle-Attacks: Hacker tries to insert themselves between server and client and tries to access sensitive data. Since there is not transmission of One-time password (OTP), such attacks are futile. When HTTPS is enabled, it guarantees that third party cannot access the communication.

## IV. EXPERIMENTAL RESULTS

*A. Test Environment*

The project is implemented on Windows 10 local environment with ASP.NET Core 2.0 platform and .NET framework 4.6.

Web application performance can be measured as following:

1) Response Time: Time difference between users send request and system response meaning, the time difference between HTTPRequest send and HTTPResponse received by the client.
2) Throughput: Total number of requests handled by a server per second. E.g. 1000 transaction can handle per second.
3) Resource utilization: Resource utilization is calculated based upon server and Network resources. Resources that consume during request processing are:
    1) CPU
    2) Memory
    3) Disk I/O
    4) Network I/O
4) Workload: How many users can the server handle with loading the application. There can be two types of user load simultaneous users or concurrent users:
    - Simultaneous users: Have active connections to the same website
    - Concurrent user: Hit the site exactly at the same moment

Performance testing is the testing, which is performed, to ascertain how the components of a system are performing, given a particular situation. Load test is a container of performance test and it run performance test under certain load till it reach their threshold limit. So in Load test same functionality (Admin user) will be run by multiple concurrent users.

Load pattern setting allows us two different options to select from:

- Constant load: Specify same number of user constantly hit your application for specified time. But at the beginning of the load test, it is not reasonable and realistic demand for any site. So use it carefully.
- Step load: can specify a user load that increases with time up to a defined maximum number. It is a good option to check, what maximum user load capacity the application can handle on a particular server configuration. As load increases, resources on the server will eventually run out.

Test Mix Model has multiple workflows where we can define based on the test cases. There are four model types that you can provide.

1. **Based on the total number of tests**: In this case, you can select which Web Performance Test will run more when a virtual user starts hitting your application. After load test execution, it matches number of test run with

assigned test distribution. You will see this below under result section. This describe maximum iterations will occur for the test cases for web application that is used.

2. **Based on the number of virtual users:** It give % period of virtual client that will run specific experiment. For example In the event that there are 1000 dynamic client on your site, at that point 600 clients are utilizing Search Item usefulness. It implies persistently 60% of all out client burden will continue utilizing Search Item module without keeping track that what number of cycles has been finished for Search Item work process.

3. **Based on user pace:** it provide frequency for particular test that will run how much times against per user per hour.

4. **Based on sequential order:** Here you can specify the sequence for test cases that need to execute in some order. During load test same order will follow in through multiple loop until load test complete.

The following resources are considered for the load testing purposes:

- **.NET CLR resources**: It provide all resources consumption against load test like how much bytes from heap consumed by your application. On right hand side under Key Indicator graph, you can see .NET CLR memory from heap consumption is increasing during load. Here you can take decision if there is any memory leakage or not.

- **IIS (ASP.NET/ASP.Net Application)**: It provide resource consumption at IIS level like Request/Sec, Session Active, Request Rejected (due to security or any reason), Request Queued, Output Cache Hit Ratio (how much pages processed directly from cache), Cache Hit Ratio (how many times application cache used), etc.

- **Memory:** It provide how much memory consumed during load test like % Committed Byte In Use, Available Mbytes (amount of physical memory, in MB, available to running processes).

- **Network Interface**: It provide network resources consumption like Bytes sent/sec, Bytes received/sec, and Current Bandwidth. Here you can take decision if bandwidth consumption more, then try to make request/repose size optimize by taking the following action for the same.
  - Usage of Ajax calls
  - Usage of JSON serialization instead of XML serialization.
  - Usage of client side cache, if same static data need to process from server multiple times.

- **Process**: It provide what are processes that used by your application and what are their consumption like "% processor time", Handle Count, Thread Count taken by sqlsever, devenv (visual studio).

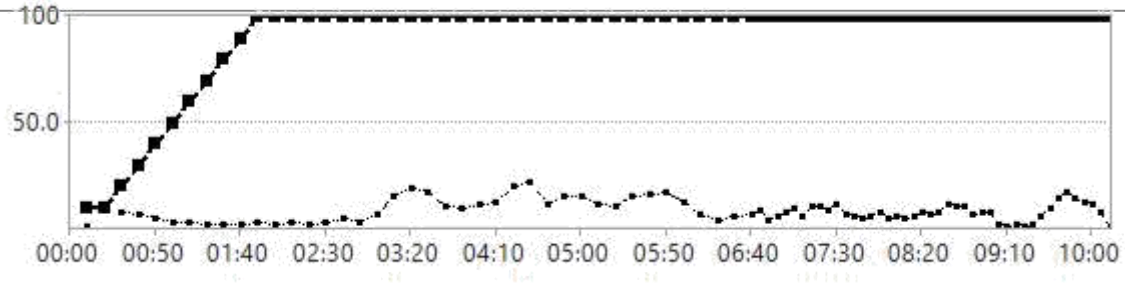
- **SQL Server resources**: It provide all consumption at SQL server level like Full scan/sec, Index search/sec, Transactions/sec, Lock Timeout/sec, Number of Deadlock/sec. Here you can see what issues occur at heavy load on database side (e.g. deadlock, timeout, table scan).

In Graph 1, horizontal axis represents time duration with fraction of 10 seconds. The run load test is only for 10 minutes:

**Key Indicator graph**: selected the following resources from Counter set

- **User Load:** It increases every 10 sec, starts from 10 and reach up to 180 (in 3 min).
- **Transaction/sec:** Transactions/sec increase with respect to user load.
- **Bytes in all Heap:** It increase with user load. But if it sharply rise then there can be memory leakage problem like resources does not release properly after consumption.

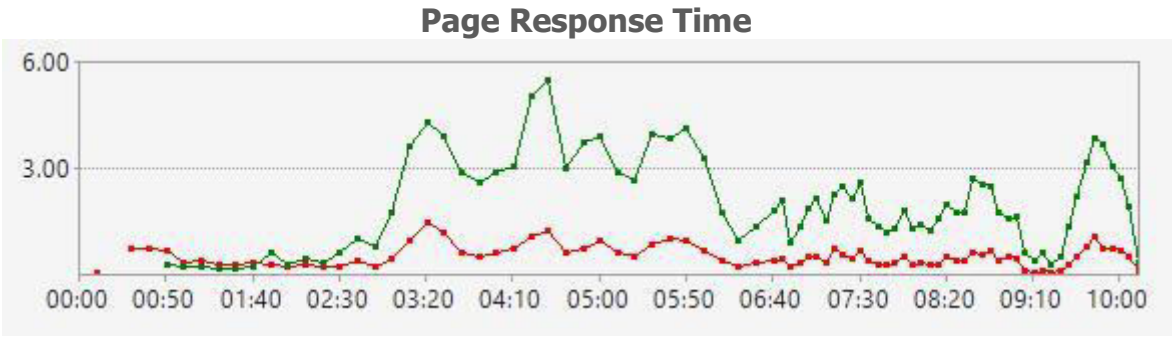
## Key Indicators




| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg. |
|---|---|---|---|---|---|---|---|---|
| User Load | _Total | LoadTest:Scenario | MAHITH | | 100 | 10 | 200 | 175 |
| Avg. Page Time | _Total | LoadTest:Page | MAHITH | | 10 | 0.11 | 2.22 | 0.95 |

Fig.3 Key Indicators Graph

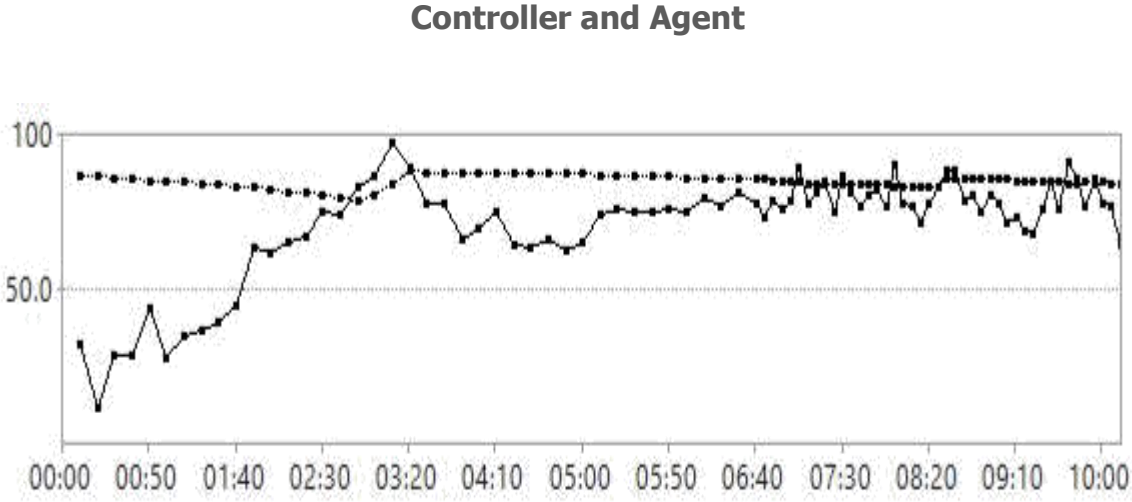
**Page Response Time:** Click pages from *Counter* ⇒ *Scenario* ⇒ *Web Performance Test* ⇒ *Pages Page Name*, against which you want to check performance across load test. On graph, you can see how it performs when load increase. As load increase, its performance go down. There are multiple properties that you can check against any page like Avg Page Time, Pages/Sec, etc.




| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg. |
|---|---|---|---|---|---|---|---|---|
| Avg. Page Time | Login{GET} | LoadTest:Page | MAHITH | | 6 | 0.068 | 1.49 | 0.58 |
| Avg. Page Time | Login{POST} | LoadTest:Page | MAHITH | | 6 | 0.18 | 5.55 | 2.32 |

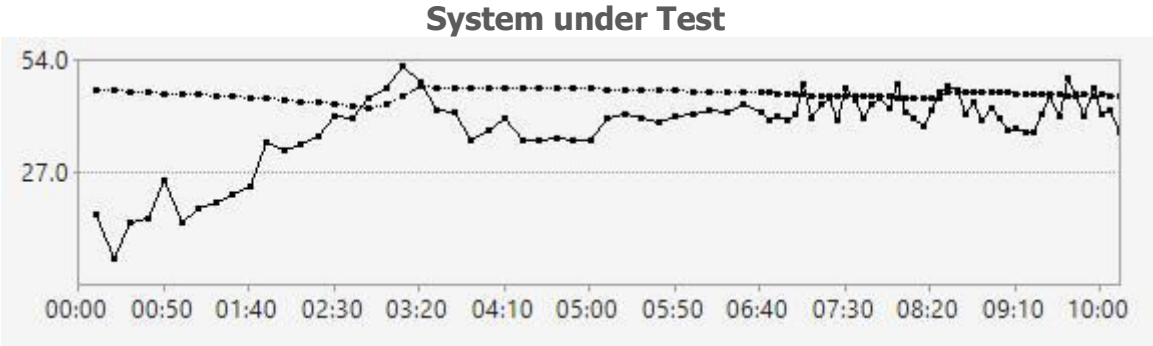
Fig.4 Page Response Time Graph

**Controller and Agent:** System resources like % CPU utilization, Memory utilization, Network I/O bytes sent and received per second, .NET CLR threads can be checked here. Here CPU % usage exceed threshold value 80%.



| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg. |
|---|---|---|---|---|---|---|---|---|
| % Processor Time | 0 | Processor | MAHITH | | 100 | 12.4 | 98.8 | 69.9 |
| Available MBytes | - | Memory | MAHITH | | 10,000 | 8,014 | 8,934 | 8,599 |

Fig.5 Controller and Agents Graph




| Counter | Instance | Category | Computer | Color | Range | Min | Max | Avg. |
|---|---|---|---|---|---|---|---|---|
| % Processor Time | _Total | Processor | MAHITH | | 100 | 11.8 | 98.8 | 69.5 |
| Available MBytes | - | Memory | MAHITH | | 10,000 | 8,014 | 8,934 | 8,599 |

Fig.6 System under Test Graph

## IV. CONCLUSION

This paper proposes the concept of transparent two-factor authentication and then proposes the transparent two-factor mechanism. The first authentication factor is still the traditional username and password. The second authentication factor is the mobile phone, which includes a unique authentication code for the user which generated every 20 sec. When the user logs in, the browser sends a login request. Both of them use their respective methods to generate request, and adjust the time-stamp to synchronize the time. The application compares the two factors to determine whether the user information is corrects and then determines whether the login is valid. The second factor authentication is completely transparent to the user, avoiding the tedious interaction between the user and the device. Therefore, the proposed T2FA exhibits the anti-fraud ability with good application prospect. The effectiveness of our proposed T2FA is further proved with detailed experiments.

## REFERENCES

[1] “Statistical Report on Internet Development in China”, 2016. [Online]. Available: https://cnnic.com.cn/IDR/ReportDownloads/201604/ P020160419390562421055.pdf.

[2] “Chinese Internet Suffers The Most Serious User Data Leak In History”, [Online]. Available: https://blogs.forcepoint.com/security - labs/chinese-internet-suffers-most-serious-us.

[3] S. Kumari and M. K. Khan, “More secure smart card-based remote user password authentication scheme with user anonymity,” *Secur. Commun Networks*, vol. 7, no. 11, pp. 2039-2053, Nov. 2014.

[4] Z. Siddiqui, A. H. Abdullah, M. K. Khan, and A. S. Alghamdi, “Smart Environment as a Service: Three Factor Cloud Based User Authentication for Telecare Medical Information System,” *J. Med. Syst.*, vol. 38, no. 1, p. 9997, Jan. 2014.

[5] https://engineering.gosquared.com/evolution-of-password-security

[6] https://www.protectimus.com/blog/the-evolution-of-two-step-authentication-means/

[7] D. A. Russell, J. P. Titlow, and Y.-J. Bemmen, “Acoustic monopoles, dipoles, and quadrupoles: An experiment revisited,” Am. J. Phys., vol. 67, no. 8, pp. 660-664, Aug. 1999.

[8] A. Rodrłguez Valiente, A. Trinidad, J. R. Garcła Berrocal, C. Grriz, and R. Ramłrez Camacho, “Extended high-frequency (9C20 kHz) audiometry reference thresholds in 645 healthy subjects,” Int. J. Audiol., vol. 53, no. 8, pp. 531-545, Aug. 2014.

[9] MOZZILLA, “Location-Aware Browsing.” [Online]. Available: https://www.mozilla.org/en-US/firefox/geolocation.

[10] G. INC, “Google 2-Step Verification.” [Online]. Available: https://www.google.com/landing/2step/.

[11] N. Karapanos, C. Marforio, C. Soriente, S. e Capkun, and S. Capkun, "Sound-proof: usable two-factor authentication based on ambient sound," in Proc. 24th USENIX Conf. Secur. Symp. (SEC'15), pp. 483-498, 2015.

[12] Wei-hua FENG, Ya-Li. LIU, "Design and implementation of uniform authentication system on cookie [J]", *Computer Engineering and Design*, vol. 023, pp. 4971-4975, 2010.

[13] *Wikipedia. Microsoft Account [EB/OL].*

[14] Csharp corner web site: https://docs.microsoft.com/en-us/dotnet/framework/network- programming/

[15] Microsoft Docs: https://docs.microsoft.com/en-gb/dotnet/standard/security/

[16] Microsoft Security Docs: https://docs.microsoft.com/en-gb/dotnet/standard/security/role-based-security

[17] Csharp corner web site: https://www.c-sharpcorner.com/article/data-logging-in-asp-net-core-using-serilog/

[18] Csharp corner web site: https://www.c-sharpcorner.com/article/getting-started-with-asp-net-core-2-0-identity-and-role-management/

[19] Csharp corner web site: https://www.c-sharpcorner.com/article/getting-started-with-asp-net-core-2-0-identity-and-role-management/

[20] StackExchange: https://stackify.com/nlog-vs-log4net-vs-serilog/

[21] J. Kuuskeri and T. Mikkonen, "Partitioning Web Applications between the Server and the Client," *J. Web Eng.*, vol. 9, no. 3, 2010, pp. 207–226.

[22] J. Offutt, "Quality Attributes of Web Software Applications," *IEEE Software*, vol. 19, no. 2, 2002, pp. 25–32.

[23] M. Hevery and A. Abrons, "Declarative Web- Applications without Server: Demonstration of How a Fully Functional Web-Application can be Built in an Hour with only HTML, CSS & <angular/> JavaScript Library," *Proc. Object-Oriented Programming,*

[24] G. Toffetti et al., "State-of-the-Art and Trends in the Systematic Development of Rich Internet Applications," *J. Web Eng.*, vol. 10, no. 1, 2011, pp. 70–86.

[25] C. Kroiss, N. Koch, and A. Knapp, "UWE4JSF: A Model-Driven Generation Approach for Web Applications," *Proc. 9th Int'l Conf. Web Eng.* (ICWE 09), LNCS 5648, Springer, 2009, pp. 493–496.

[26] J. Offutt, "Quality Attributes of Web Software Applications," *IEEE Software*, vol. 19, no. 2, 2002, pp. 25–32.

[27] Leff, A., & Rayfield, J.T. (2001), *Web-Application Development Using the Model-View-Controller Design Pattern*, Paper presented at the Fifth IEEE International Enterprise Distributed Object Computing Conference, 118-127.

[28] Swales D., Sewry D. and Terzoli A (2003), *A Performance Comparison of Web Development Technologies to Distribute Multimedia across an Intranet*, Retrieved fromhttp://www.satnac.org.za/proceedings/2003/bband/bband2/703 -Swales.pdf

[29] Feeman, A. (2012), *Pro ASP.NET MVC 4*, Apress, 47-51

[30] Web Application Security Statistics, "http://projects.webappsec.org/w/page/13246989/WebApplication SecurityStatistics."

[31] A. Jesudoss and N.P. Subramaniam, "A Taxonomy of Authentication Techniques for Web Services", International Journal of Engineering Research and Technology, Vol. 3 2014, pp. 271-275.

[32] Mudassar Raza, Muhammad Iqbal, Muhammad Sharif and Waqas Haider, "A Survey of Password Attacks and Comparative Analysis on Methods for Secure Authentication", World Applied Sciences Journal, vol. 19, pp. 439-444, Jan. 2012.