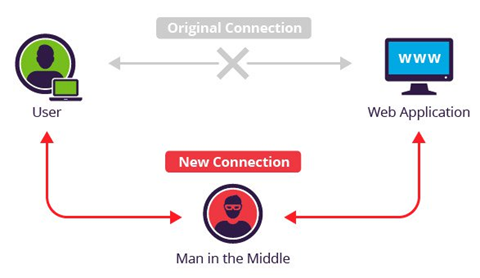
**Man In The Middle Attack Detection**

**Introduction :-**

Internet connections are always vulnerable and can be attacked in multiple ways. One such general type of attack is 'Man-In-The-Middle-Attack'. There is a predefined flow how of internet requests works. But because of WLAN's inherent security vulnerability of broadcasting nature in communication, every data packets sent in a wireless medium is broadcasted where any user in the communication range can capture the packet. The MITM, Attacker gets in between client and server. Now he can access the traffic, view the requests, modify the response from server, steal clients data, etc. In general terms it can be defined as " MITM is an attack where the attacker secretly relays and possibly alters the communication between two parties who believe they are directly communicating with each other".



A man-in-the-middle attack occurs when an attacker reroutes communication between two users through the attacker’s computer without the knowledge of the two communicating users. The attacker can monitor and read the traffic before sending it on to the intended recipient. Each user in the communication unknowingly sends traffic to and receives traffic from the attacker, all while thinking they are communicating only with the intended user. This can happen if an attacker can modify Active Directory Domain Services to add his or her server as a trusted server or modify Domain Name System (DNS) to get clients to connect through the attacker on their way to the server. A man-in-the-middle attack can also occur with media traffic between two clients.

Once man in the middle attacker establishes connection he can,

• **Downgrading**

**• Filtering**

**• Injection**

**• Key Manipulation**

1) **Downgrading:-**

• Parameters exchanged by server and client can be substituted by server and client in the beginning of a connection. Attacker can force the client to initialize a SSH1 connection instead of SSH2.

• Attacker can block key-material exchanged on the port 500 UDP, so end points think that other cannot start the IPSec connection and results in IPSec failure.

2) **Filtering**

• Attacker can change data in packets by recalculating checksum.

• He can change length of payload when transmission of data is in two direction.

• He can establish filters on the activities of the user.

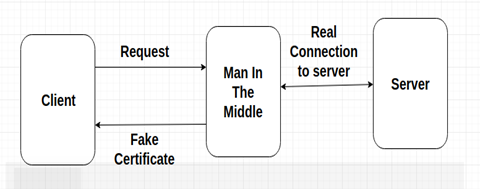
3) **Injection:-**

• The attacker is capable of adding packets in established connection.

• Even when injecting he can modify the sequence numbers and keep connection synchronized.

4) **Key Manipulation:-**

• Attacker can modify the public key exchanged by server and client.



• Man in the middle can use fake certificates by buying certificates from those root certificate provider for few bucks and send that certificate to client. As these certificates are not invalid browser won't generate any security concern.

**Challenges to detect MITM :-**

**• Availability of Multiple Root Certificates :-**

The number of trusted authorities for providing authenticated certificates are increasing rapidly. The modern web browsers come with a large number of preinstalled certificates. In Mozilla there are 66+ trusted root certificates from 24 authorities, and Internet Explorer has 108+ trusted root certificates. But the different amount of certificates indicates that there are differences of opinions about which authorities that are trusted. For the user it means that Internet applications might behave different depending on which browser is used and security warnings might occur. The user must still make the decisions about who to trust, and with the increased amount of certificates available the decision is harder to make.

Previous approaches used to checks that the received certificate from a visited HTTPS website matches with certificate provided by any of the root certificate provider present in the browser. But nowadays with large number of trusted certificated available there are some certificates which an attacker can easily buy. So attacker can replace the server's certificated and instead send his own certificate and browsers won't show any security concerns.

**• Social Aspects :-**

It requires maintenance to provide security through SSL, but lack of maintenance might cause unnecessary security warning. Many certificates used on the Internet have problems that cause security warnings, they might not have a trusted root certificate or might have expired.

When users repeatedly are confronted with “bad” certificates and warnings, it becomes normal for users to accept security warnings. So even if actual MITM occurs doesn’t give it much attention and ignore it making itself vulnerable.

**Implementation :-**

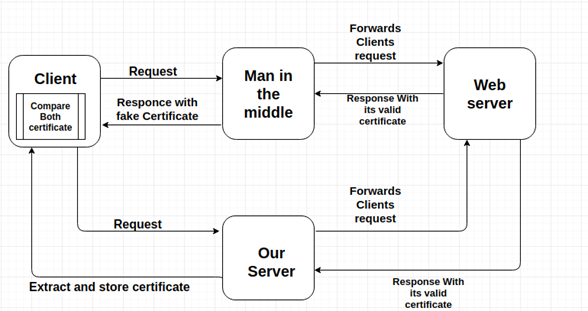
In the MITM detection we check that is there any man in the middle attacker present who is modifying the content.

How our approach works :-

1) When client send request a webpage the man in the middle intercepts the request and forwards to server .

2) The server in response sends the requested webpage to MITM and then Man in the middle forwards the webpage to client along with its own fake certificate.

3) When client initially sends request an additional query is fired in which the similar request is made to our server. Which is done by server side code.



4)Our server sends request to web server. Which as usual sends response along with its genuine certificate. Now our code extracts the certificate from response obtained and stores it on client's device.

5) And finally, the certificate obtained from regular response and certificate obtained from our server is compared. If their fingerprint differ we say that man in the middle is present and active.

**Client-Side Code :-**

<head>

<meta charset=**"UTF-8"**>

<title>**CyberThreatMap.in** </title>

</head>

<body class=**"sea-background"**>

<div class="container">

<div class="row">

<div class="col-sm-8">

<div class="row">

<input class="col-sm-6 form-control" type="text" id="url" value="www.google.com">

<button id="query" type="submit" class="col-sm-2 btn btn-primary">Query</button>

</div>

<div class="row">

<div id="certificate" class="offset-sm-1 col-sm-10 alert alert-success row invisible">

<table id="certificate-stats"></table>

</div>

</div>

</div>

</div>

<script>

***const*** urlPrefix **=** **https://tlrtr0pirg.execute-api.us-east1.amazonaws.com/prod/Cyberthreatmap**"**;**

$**.**ajax**({**url**:** url**,**

success**:** ***function*(**response**){**

***const*** plaintext **=** response.toString**(**CryptoJS.enc.Utf8**);**

***const*** crt **=** JSON.parse**(**plaintext**);**

***if*** **(!**crt.success**)** **{**

$**(**"#certificate"**)**

**.**removeClass**(**"invisible"**)**

**.**removeClass**(**"alert-success"**)**

**.**addClass**(**"alert-warning"**);**

$**(**"#certificate-stats"**).**text**(**crt.message**);**

**}** ***else*** **{**

***const*** certTxt **=** "<tr><td>Subject:</td><td>" **+** crt.subject.CN **+** "\n</td></tr>"

**+** "<tr><td>Alternative names:</td><td>" **+** crt.subjectaltname **+** "\n</td></tr>"

**+** "<tr><td>Fingerprint:</td><td>" **+** crt.fingerprint **+** "\n</td></tr>"

**+** "<tr><td>CA (O):</td><td>" **+** crt.issuer.O **+** "\n</td></tr>"

**+** "<tr><td>CA (OU):</td><td>" **+** crt.issuer.OU **+** "\n</td></tr>"

**+** "<tr><td>CA (C):</td><td>" **+** crt.issuer.C **+** "\n</td></tr>"

**+** "<tr><td>CA (CN):</td><td>" **+** crt.issuer.CN **+** "\n</td></tr>"

**+** "<tr><td>CA Issuer:</td><td>" **+** crt.infoAccess**[**"CA Issuers - URI"**][**0**]** **+** "\n</td></tr>"

**+** "<tr><td>OCSP:</td><td>" **+** crt.infoAccess**[**"OCSP - URI"**][**0**]** **+** "\n</td></tr>"

**+** "<tr><td>Valid from:</td><td>" **+** crt.valid\_from **+** "\n</td></tr>"

**+** "<tr><td>Valid to:</td><td>" **+** crt.valid\_to **+** "\n</td></tr>"

**+** "<tr><td>Serial:</td><td>" **+** crt.serialNumber **+** "\n</td></tr>"**;**

$**(**"#certificate"**)**

**.**removeClass**(**"invisible"**)**

**.**removeClass**(**"alert-warning"**)**

**.**addClass**(**"alert-success"**);**

$**(**"#certificate-stats"**).**html**(**certTxt**);**

**}**

**},**

error**:** ***function*(**result**)** **{**

$**(**"#certificate"**)**

**.**removeClass**(**"invisible"**)**

**.**removeClass**(**"alert-success"**)**

**.**addClass**(**"alert-warning"**);**

$**(**"#certificate-stats"**).**text**(**"Unable to connect to snuck.me remote."**);**

**}**

**});**

**});**

**});**

</body>

</html>

**Explanation of code:-**

* The home page would consist of a input section where user can check for MITM vulnerability for any website. Our application would collect the genuine certificate of specified website and and compare it with regular certificate.
* The head section defines the universal encoding and title of the web page.
* Next the body part consist of mainly input section and an invisible table which will be later filled with details.
* In input section user can enter the website name and the whole process will be performed with the details of certificate of that website. That input section is written in bootstrap.
* The jQuery [$.ajax()](http://api.jquery.com/jquery.ajax/) function is used to perform an asynchronous HTTP request. The url parameter is a string containing the URL you want to reach with the Ajax call, while success is an object literal containing the configuration for the Ajax request. Here success refers that A function to be called if the request succeeds.
* The response from api will be send to function from response variable. And later converted to string format and stored in plaintext variable.

Eg.

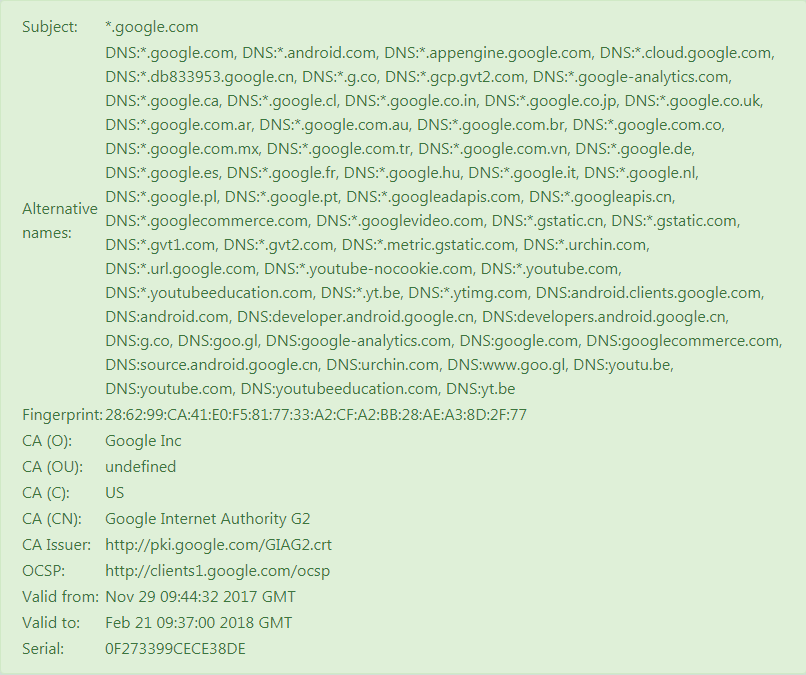
Following is the example of google.com

Response:

"{\"subject\":{\"C\":\"US\",\"ST\":\"California\",\"L\":\"Mountain View\",\"O\":\"Google Inc\",\"CN\":\"\*.google.com\"},\"issuer\":{\"C\":\"US\",\"O\":\"Google Inc\",\"CN\":\"Google Internet Authority G2\"},\"subjectaltname\":\"DNS:\*.google.com, DNS:\*.android.com, DNS:\*.appengine.google.com, DNS:\*.cloud.google.com, DNS:\*.db833953.google.cn, DNS:\*.g.co, DNS:\*.gcp.gvt2.com, DNS:\*.google-analytics.com, DNS:\*.google.ca, DNS:\*.google.cl, DNS:\*.google.co.in, DNS:\*.google.co.jp, DNS:\*.google.co.uk, DNS:\*.google.com.ar, DNS:\*.google.com.au, DNS:\*.google.com.br, DNS:\*.google.com.co, DNS:\*.google.com.mx, DNS:\*.google.com.tr, DNS:\*.google.com.vn, DNS:\*.google.de, DNS:\*.google.es, DNS:\*.google.fr, DNS:\*.google.hu, DNS:\*.google.it, DNS:\*.google.nl, DNS:\*.google.pl, DNS:\*.google.pt, DNS:\*.googleadapis.com, DNS:\*.googleapis.cn, DNS:\*.googlecommerce.com, DNS:\*.googlevideo.com, DNS:\*.gstatic.cn, DNS:\*.gstatic.com, DNS:\*.gvt1.com, DNS:\*.gvt2.com, DNS:\*.metric.gstatic.com, DNS:\*.urchin.com, DNS:\*.url.google.com, DNS:\*.youtube-nocookie.com, DNS:\*.youtube.com, DNS:\*.youtubeeducation.com, DNS:\*.yt.be, DNS:\*.ytimg.com, DNS:android.clients.google.com, DNS:android.com, DNS:developer.android.google.cn, DNS:developers.android.google.cn, DNS:g.co, DNS:goo.gl, DNS:google-analytics.com, DNS:google.com, DNS:googlecommerce.com, DNS:source.android.google.cn, DNS:urchin.com, DNS:www.goo.gl, DNS:youtu.be, DNS:youtube.com, DNS:youtubeeducation.com, DNS:yt.be\",\"infoAccess\":{\"CA Issuers - URI\":[\"http://pki.google.com/GIAG2.crt\"],\"OCSP - URI\":[\"http://clients1.google.com/ocsp\"]},\"valid\_from\":\"Nov 29 09:44:32 2017 GMT\",\"valid\_to\":\"Feb 21 09:37:00 2018 GMT\",\"fingerprint\":\"28:62:99:CA:41:E0:F5:81:77:33:A2:CF:A2:BB:28:AE:A3:8D:2F:77\",\"ext\_key\_usage\":[\"1.3.6.1.5.5.7.3.1\"],\"serialNumber\":\"0F273399CECE38DE\",\"raw\":{\"type\":\"Buffer\",\"data\":[48,130,7,131,48,130,6,107,160,3,2,1,2,2,8,15,39,51,153,206,206,56,222,48,13,6,9,42,134,72,134,247,13,1,1,11,5,0,48,73,49,11,48,9,6,3,85,4,6,19,2,85,83,49,19,48,17,6,3,85,4,10,19,10,71,111,111,103,108,101,32,73,110,99,49,37,48,35,6,3,85,4,3,19,28,71,111,111,103,108,101,32,73,110,116,101,114,110,101,116,32,65,117,116,104,111,114,105,116,121,32,71,50,48,30,23,13,49,55,49,49,50,57,48,57,52,52,51,50,90,23,13,49,56,48,50,50,49,48,57,51,55,48,48,90,48,102,49,11,48,9,6,3,85,4,6,19,2,85,83,49,19,48,17,6,3,85,4,8,12,10,67,97,108,105,102,111,114,110,105,97,49,22,48,20,6,3,85,4,7,12,13,77,111,117,110,116,97,105,110,32,86,105,101,119,49,19,48,17,6,3,85,4,10,12,10,71,111,111,103,108,101,32,73,110,99,49,21,48,19,6,3,85,4,3,12,12,42,46,103,111,111,103,108,101,46,99,111,109,48,89,48,19,6,7,42,134,72,206,61,2,1,6,8,42,134,72,206,61,3,1,7,3,66,0,4,183,160,46,178,8,124,133,192,10,47,104,106,219,178,16,205,128,171,221,94,180,95,95,221,121,42,83,229,141,115,10,185,82,234,71,203,49,255,197,242,164,121,31,171,199,20,88,133,251,73,26,88,255,251,99,250,109,91,136,222,125,172,168,181,163,130,5,27,48,130,5,23,48,19,6,3,85,29,37,4,12,48,10,6,8,43,6,1,5,5,7,3,1,48,14,6,3,85,29,15,1,1,255,4,4,3,2,7,128,48,130,3,225,6,3,85,29,17,4,130,3,216,48,130,3,212,130,12,42,46,103,111,111,103,108,101,46,99,111,109,130,13,42,46,97,110,100,114,111,105,100,46,99,111,109,130,22,42,46,97,112,112,101,110,103,105,110,101,46,103,111,111,103,108,101,46,99,111,109,130,18,42,46,99,108,111,117,100,46,103,111,111,103,108,101,46,99,111,109,130,20,42,46,100,98,56,51,51,57,53,51,46,103,111,111,103,108,101,46,99,110,130,6,42,46,103,46,99,111,130,14,42,46,103,99,112,46,103,118,116,50,46,99,111,109,130,22,42,46,103,111,111,103,108,101,45,97,110,97,108,121,116,105,99,115,46,99,111,109,130,11,42,46,103,111,111,103,108,101,46,99,97,130,11,42,46,103,111,111,103,108,101,46,99,108,130,14,42,46,103,111,111,103,108,101,46,99,111,46,105,110,130,14,42,46,103,111,111,103,108,101,46,99,111,46,106,112,130,14,42,46,103,111,111,103,108,101,46,99,111,46,117,107,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,97,114,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,97,117,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,98,114,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,99,111,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,109,120,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,116,114,130,15,42,46,103,111,111,103,108,101,46,99,111,109,46,118,110,130,11,42,46,103,111,111,103,108,101,46,100,101,130,11,42,46,103,111,111,103,108,101,46,101,115,130,11,42,46,103,111,111,103,108,101,46,102,114,130,11,42,46,103,111,111,103,108,101,46,104,117,130,11,42,46,103,111,111,103,108,101,46,105,116,130,11,42,46,103,111,111,103,108,101,46,110,108,130,11,42,46,103,111,111,103,108,101,46,112,108,130,11,42,46,103,111,111,103,108,101,46,112,116,130,18,42,46,103,111,111,103,108,101,97,100,97,112,105,115,46,99,111,109,130,15,42,46,103,111,111,103,108,101,97,112,105,115,46,99,110,130,20,42,46,103,111,111,103,108,101,99,111,109,109,101,114,99,101,46,99,111,109,130,17,42,46,103,111,111,103,108,101,118,105,100,101,111,46,99,111,109,130,12,42,46,103,115,116,97,116,105,99,46,99,110,130,13,42,46,103,115,116,97,116,105,99,46,99,111,109,130,10,42,46,103,118,116,49,46,99,111,109,130,10,42,46,103,118,116,50,46,99,111,109,130,20,42,46,109,101,116,114,105,99,46,103,115,116,97,116,105,99,46,99,111,109,130,12,42,46,117,114,99,104,105,110,46,99,111,109,130,16,42,46,117,114,108,46,103,111,111,103,108,101,46,99,111,109,130,22,42,46,121,111,117,116,117,98,101,45,110,111,99,111,111,107,105,101,46,99,111,109,130,13,42,46,121,111,117,116,117,98,101,46,99,111,109,130,22,42,46,121,111,117,116,117,98,101,101,100,117,99,97,116,105,111,110,46,99,111,109,130,7,42,46,121,116,46,98,101,130,11,42,46,121,116,105,109,103,46,99,111,109,130,26,97,110,100,114,111,105,100,46,99,108,105,101,110,116,115,46,103,111,111,103,108,101,46,99,111,109,130,11,97,110,100,114,111,105,100,46,99,111,109,130,27,100,101,118,101,108,111,112,101,114,46,97,110,100,114,111,105,100,46,103,111,111,103,108,101,46,99,110,130,28,100,101,118,101,108,111,112,101,114,115,46,97,110,100,114,111,105,100,46,103,111,111,103,108,101,46,99,110,130,4,103,46,99,111,130,6,103,111,111,46,103,108,130,20,103,111,111,103,108,101,45,97,110,97,108,121,116,105,99,115,46,99,111,109,130,10,103,111,111,103,108,101,46,99,111,109,130,18,103,111,111,103,108,101,99,111,109,109,101,114,99,101,46,99,111,109,130,24,115,111,117,114,99,101,46,97,110,100,114,111,105,100,46,103,111,111,103,108,101,46,99,110,130,10,117,114,99,104,105,110,46,99,111,109,130,10,119,119,119,46,103,111,111,46,103,108,130,8,121,111,117,116,117,46,98,101,130,11,121,111,117,116,117,98,101,46,99,111,109,130,20,121,111,117,116,117,98,101,101,100,117,99,97,116,105,111,110,46,99,111,109,130,5,121,116,46,98,101,48,104,6,8,43,6,1,5,5,7,1,1,4,92,48,90,48,43,6,8,43,6,1,5,5,7,48,2,134,31,104,116,116,112,58,47,47,112,107,105,46,103,111,111,103,108,101,46,99,111,109,47,71,73,65,71,50,46,99,114,116,48,43,6,8,43,6,1,5,5,7,48,1,134,31,104,116,116,112,58,47,47,99,108,105,101,110,116,115,49,46,103,111,111,103,108,101,46,99,111,109,47,111,99,115,112,48,29,6,3,85,29,14,4,22,4,20,74,239,215,240,121,66,224,199,204,102,13,22,17,222,53,231,28,153,10,87,48,12,6,3,85,29,19,1,1,255,4,2,48,0,48,31,6,3,85,29,35,4,24,48,22,128,20,74,221,6,22,27,188,246,104,181,118,245,129,182,187,98,26,186,90,129,47,48,33,6,3,85,29,32,4,26,48,24,48,12,6,10,43,6,1,4,1,214,121,2,5,1,48,8,6,6,103,129,12,1,2,2,48,48,6,3,85,29,31,4,41,48,39,48,37,160,35,160,33,134,31,104,116,116,112,58,47,47,112,107,105,46,103,111,111,15,66,181,250,54,181,97,182,145,136,172,97,60,128,136,102,186,42,213,72,3,234,241,240,35,151,135,136,52,237,32,244,70,55,43,150,92,74,162,147,206,182,54,227,76,58,151,11,105,99,104,87,221,3,41,244,6,247,95,187,21,35,39,47,23,33,208,211,95,127,70,73,143,46,138,133,5,250,23,93,238,146,143,140,131,236,84,115,104,132,9,105,79,84,89,30,140,43,226,165,113,187,137,232,70,199,236,145,245,156,192,98,98,45,224,194,168,248,49,194,59,152,25,100,129,18,246,22,109,32,119,244,16,174,85,141,36,15,14,163,62,6,16,24,203,60,253,188,38,91,224,198,212,131,99,222,237,225,247,238,103,84,112,77,170,153,116,127,204,205,133,162,132,93,24,83,140,233,117,248,216,167,252,9,116,68,128,63,188,150,216,54,228,34,145,141,53,121,202,117,128,199,246]},\"success\":true,\"message\":\"Found certificate for google.com\"}"

* JSON.parse(plantext) function converts the result to proper format which consist only of relevant information to user..

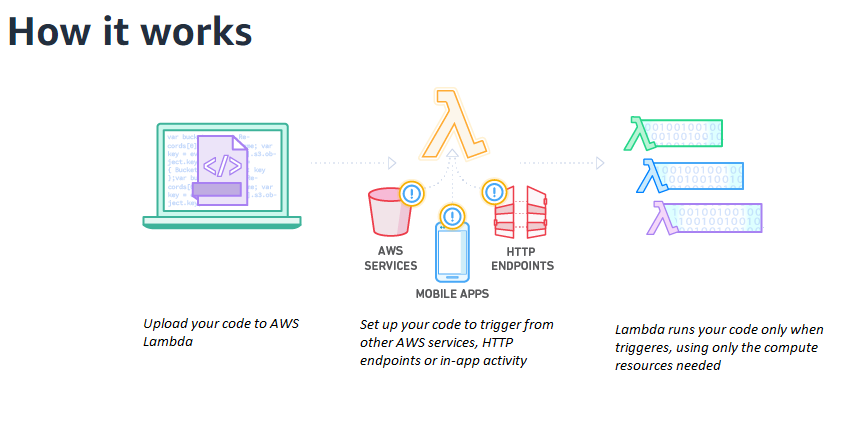
Eg.



* If program does not show any error the formatted output is displayed as above. Or if any error occurs it shown with some styles changes.
* For the backend api has been created with amazon web services. The lambda tool of aws is utilized in this project.

**\*\*\*What is Amazon Lambda**

AWS Lambda is a compute service that lets you run code without provisioning or managing servers. AWS Lambda executes your code only when needed and scales automatically, from a few requests per day to thousands per second. You pay only for the compute time you consume - there is no charge when your code is not running. With AWS Lambda, you can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs your code on a high-availability compute infrastructure and performs all of the administration of the compute resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging. All you need to do is supply your code in one of the languages that AWS Lambda supports (currently Node.js, Java, C# and Python).



You can use AWS Lambda to run your code in response to events, such as changes to data in an Amazon S3 bucket or an Amazon DynamoDB table; to run your code in response to HTTP requests using Amazon API Gateway; or invoke your code using API calls made using AWS SDKs. With these capabilities, you can use Lambda to easily build data processing triggers for AWS services like Amazon S3 and Amazon DynamoDB process streaming data stored in Kinesis, or create your own back end that operates at AWS scale, performance, and security.

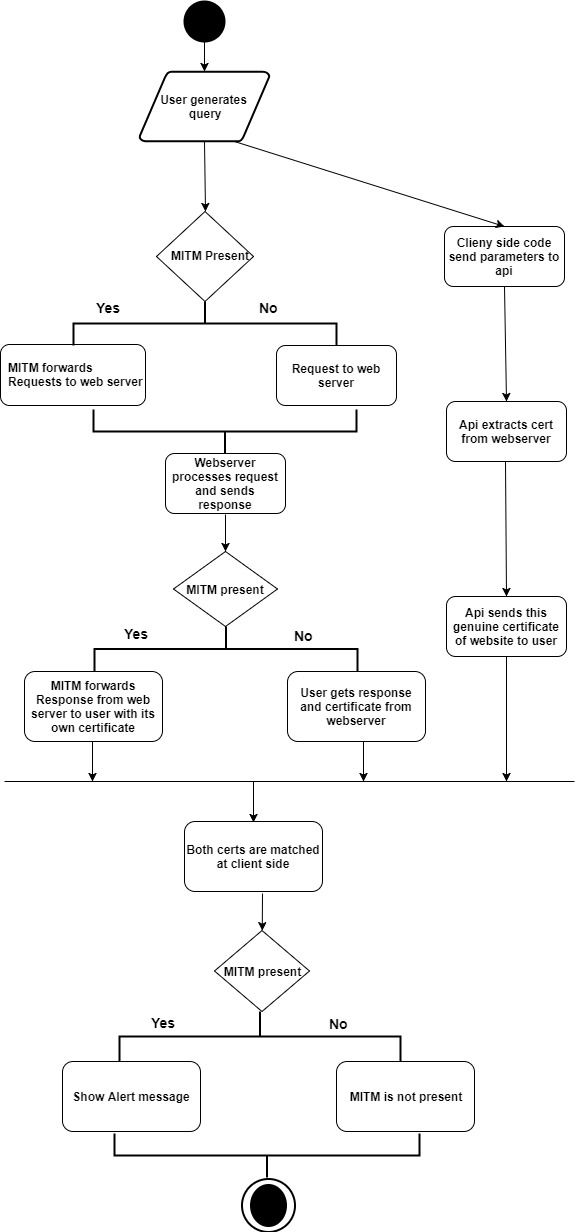
You can also build [serverless](https://aws.amazon.com/serverless" \t "_blank) applications composed of functions that are triggered by events and automatically deploy them using AWS CodePipeline and AWS CodeBuild. For more information, see [Deploying Lambda-based Applications](https://docs.aws.amazon.com/lambda/latest/dg/deploying-lambda-apps.html).

For more information about the AWS Lambda execution environment, see [Lambda Execution Environment and Available Libraries](https://docs.aws.amazon.com/lambda/latest/dg/current-supported-versions.html). For information about how AWS Lambda determines compute resources required to execute your code, see [Configuring Lambda Functions](https://docs.aws.amazon.com/lambda/latest/dg/resource-model.html).

AWS Lambda is an ideal compute platform for many application scenarios, provided that you can write your application code in languages supported by AWS Lambda (that is, Node.js, Java, C# and Python), and run within the AWS Lambda standard runtime environment and resources provided by Lambda.

Beyond the Getting Started exercise, you can explore the various use cases, each of which is provided with a tutorial that walks you through an example scenario. Depending on your application needs (for example, whether you want event driven Lambda function invocation or on-demand invocation), you can follow specific tutorials that meet your specific needs. For more information, see Use Cases.

**FlowChart :-**



**Flow chart Explanation:-**

Initially user enters the website name in our system. Which in turn fires two queries, one query sends website name to our api as a parameter. One query is sent to web server as a request, if MITM is present it forwards the request to web server.

Now our api, with the parameter available extracts the certificate from the webserver and sends to user in form of text.

The user gets response directly form webserver with genuine certificate if MITM is not present. But if MITM is present he forwards response from webserver to client with its own fake certificate.

Now the client has both certificate, one from webserver and one from our api. The client side code compares fingerprint of both the certificate. If they differ indicates presence of Man in The Middle and an alert is sent to client otherwise network is clear.

**Api Code Explanation:-**

* Initially a lambda function is created with nodejs (version strictly 6.4) configuration and the following code is written. Before beginning With coding part npm is installed on machine. And npm is initialized then other required files are installed as per requirement.

**npm init**

* As shown in first line, it require 'get-ssl-certificate', which is install from terminal. From command

**npm install get-ssl-certificate – – save**

* When lambda function is created a *handler* is specified, A function in code that AWS Lambda can invoke when the service execute the code. Following is the general syntax when creating handler function in nodejs.

exports.*myHandler* = function(event, context, callback)

{

....

}

* event – AWS Lambda uses this parameter to pass in event data to the handler.
* context – AWS Lambda uses this parameter to provide your handler the runtime information of the Lambda function that is executing.
* callback – You can use the optional callback to return information to the caller, otherwise return value is null.

**Note:-**

The callback is supported only in the Node.js runtimes v6.10 and v4.3. If you are using runtime v0.10.42, you need to use the context methods (done, succeed, and fail) to properly terminate the Lambda function.

***myHandler*** – This is the name of the function AWS Lambda invokes. You export this so it is visible to AWS Lambda. Suppose you save this code as helloworld.js. Then, helloworld.myHandler is the handler.

* Here event contains the website name. Which is saved in remote Url.
* The sslCertificate.get() function finds the details of the certificate like Fingerprint, CA, CA Issuer, Validity, Fingerprint, etc.
* If the certificate's data is extracted successfully, then it is send to client side in text format. So that even if the MITM is present he cannot modify the data.

**var** sslCertificate **=** require**(**'get-ssl-certificate'**);**

exports**.**handler **=** **(event,** context**,** callback**)** **=>** **{**

**const** remoteUrl**=event** **;**

sslCertificate**.**get**(**remoteUrl**)**

**.**then**(function(**certificate**)** **{**

certificate**.**success **=** **true;**

certificate**.**message **=** `Found certificate for ${remoteUrl}`**;**

**const** plaintext **=** JSON**.**stringify**(**certificate**);**

callback**(null,** plaintext**);**

**}).catch(function(**reason**){**

**const** plaintext **=** JSON**.**stringify**({**

success**:** **false,**

message**:** `Unable to find certificate for ${remoteUrl}`

**});**

callback**(null,** plaintext**);**

**});**

**};**

Once the certificate from both routes is obtained they are compared by their fingerprint. If found same MITM is not present or else if certificates are not found same then the Man in the middle is present.

**Conclusion:-**

The man in the middle attack is a serious issue, in which client server network is infiltrated by third party. This third party can perform his malicious activities on client, which could possibly result into great loss of user. This paper has provided a way to detect man in the middle.

In this work we have shown a method for detecting MITM attacks. This work does not follow the orthodox method of detecting the MITM by the certificate of the response obtained and checking the authenticity of it. The test results show that our method can detect MITM attacks.

**References:-**

**1)**https://en.wikipedia.org/wiki/Man-in-the-middle\_attack

2)https://www.incapsula.com/web-application-security/man-in-the-middle-mitm.html

3)http://citeseerx.ist.psu.edu/viewdoc/downloaddoi=10.1.1.159.8579&rep=rep1&type=pdf

4)https://docs.aws.amazon.com/lambda/latest/dg/welcome.html

5)https://docs.aws.amazon.com/lambda/latest/dg/lambda-introduction-function.html

6)https://dl.acm.org/citation.cfm?id=2557579