Web Application Security Assessment Report On

http://phrms.cloudapp.net

Conducted By:
C-DAC

Report Generation Date:
Thu, Dec 8, 2016, 17:18:28

All Rights Reserved

This is intentionally left blank

CONFIDENTIAL

Security Audit Details

Resource Name: http://phrms.cloudapp.net
Testing URL: http://phrms.cloudapp.net

CONFIDENTIAL

CONFIDENTIAL

Table of Content

CONFIDENTIAL

Summary

IP Address of the Server	: 13.76.135.224
Operating System	: windows server
Application Running	: ASP.NET
Web Server	: Microsoft-IIS/8.0, Kestrel
Port Scanned	: 8082,8092

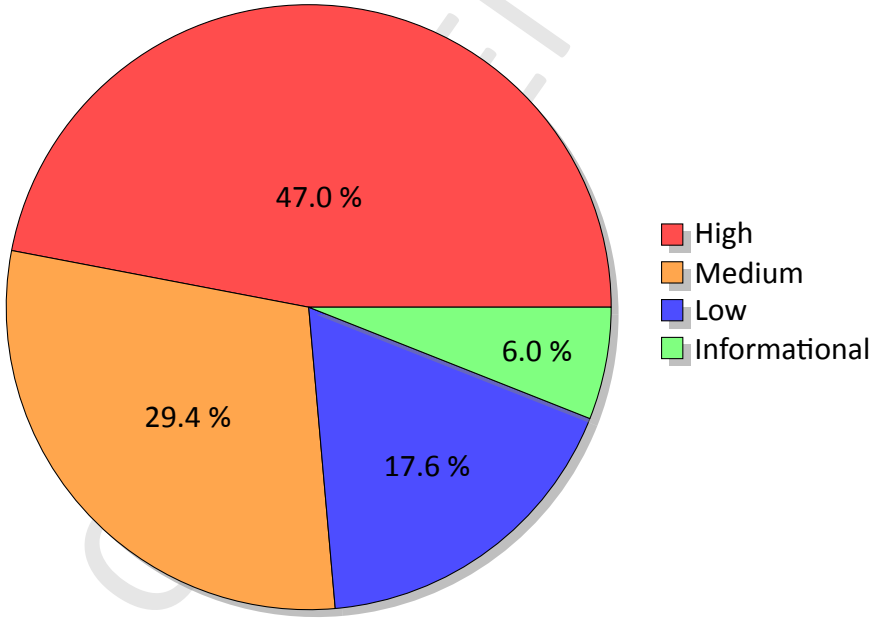
Risk Level

HIGH

One or more High severe vulnerabilities are discovered in the website and thus the overall threat level is rated as High.

Threats Summary

Total Threats Found:17



CONFIDENTIAL

Executive Summary

C-DAC has conducted application security assessment on http://phrms.cloudapp.net application from Mon, Oct 10, 2016 to Mon, Dec 5, 2016. The vulnerabilities/weaknesses observed during the evaluation are given below. The results indicate the status of the application during the evaluation period only.

Overall Threat Level **High**

S.no	Vulnerability Discovered at	Name of the Vulnerability	Recommendation	Risk Level
1	phrms.cloudapp.net:8082	User credentials are sent in clear / easy to decode format	The application should use transport-level encryption (SSL or TLS) to protect all sensitive communications passing between the client and the server	High
2	http://phrms.cloudapp.net:8092/	Host header attack	The web application should use the SERVER_NAME instead of the Host header.	High
3	phrms.cloudapp.net	Possible Brute-force Attack	Implement captcha, Implement Account Lockout, Give static error messages for any wrong input.	High
4	http://phrms.cloudapp.net:8082/HealthCondition	Improper Input Validation	This can be a high risk vulnerability and can be underestimated.	High
5	http://phrms.cloudapp.net:8092/Patient/Index	Cross Site Request Forgery	Developers are encouraged to adopt the Synchronizer Token Pattern (random ”challenge” tokens that are associated with the user’s current session). These challenge tokens are then inserted within the HTML forms and links associated with sensitive server-side operations. CSRF guard code also solves the problem	High
6	http://phrms.cloudapp.net:8092/vendor/peity/test/jquery-1.6.2.min.js	Vulnerable Javascript library	Upgrade to the latest version.	High
7	phrms.cloudapp.net	Microsoft IIS tilde directory enumeration	https://soroush.secproject.com/downloadable/microsoft_i	High

CONFIDENTIAL

8	phrms.cloudapp.net:8092	Session Mismanagement	Follow a secure session management lifecycle which includes proper generation, maintenance and expiration of session tokens.	High
9	phrms.cloudapp.net	Unwanted Methods Enabled	Disable the OPTIONS Method.	Medium
10	phrms.cloudapp.net:8092	Debug method Enabled		Medium
11	phrms.cloudapp.net:8092	Vulnerable Remember Password	Ensure that no credentials are stored in clear text or are easily retrievable in encoded or encrypted forms in cookies.	Medium
12	http://phrms.cloudapp.net:8082/Account/Dashboard	Action Spoofing (Clickjacking)	To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself.	Medium
13	phrms.cloudapp.net:8082	Tracing Error	You can protect your trace.axd file by implementing IIS Basic Authentication or you can add location tag in your web.config file.	Medium
14	http://phrms.cloudapp.net	Unused Ports/Services	You will need to close these or block them from being exposed on the Internet.	Low
15	phrms.cloudapp.net:8092	ASP.NET MVC version disclosure	To remove the X-AspNetMvc-Version header add the following code in Global.asax, in the Application Start event: MvcHandler.DisableMvcResponseHeader = true;	Low
16	phrms.cloudapp.net:8092	Password input field with autocomplete enabled	The password autocomplete should be disabled in sensitive applications.To disable autocomplete, you may use a code similar to:<INPUT TYPE=”password” AUTOCOMPLETE=”off”>	Low
17	phrms.cloudapp.net:8092	Web Server Version Disclosure	Configure your web server to prevent information leakage from the SERVER header of its HTTP response.	Informational

CONFIDENTIAL

Vulnerability in Detail

1. User credentials are sent in clear / easy to decode format — High

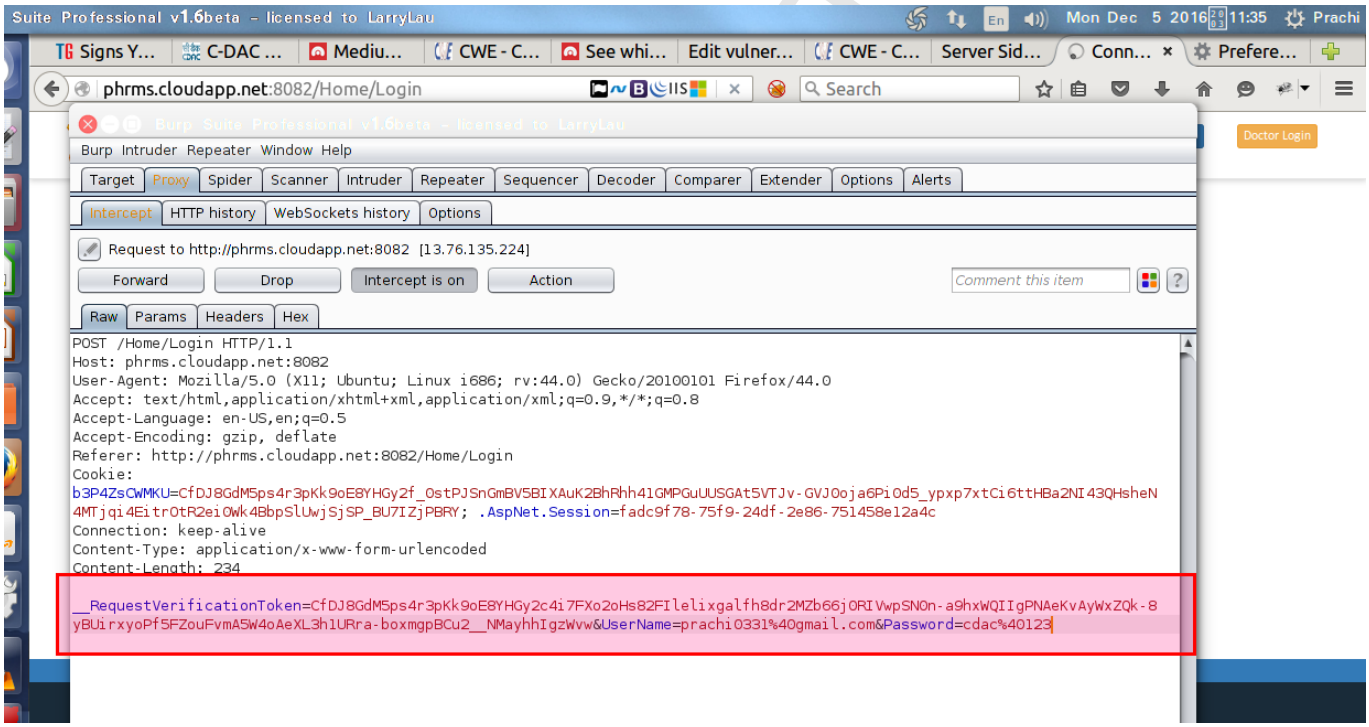
Affected Url : phrms.cloudapp.net:8082

1.1 Description

Credentials submitted over an unencrypted connection are vulnerable to capture by an attacker who is suitably positioned on the network. This includes any malicious party located on the user's own network, within their ISP, within the ISP used by the application, and within the application's hosting infrastructure. Even if switched networks are employed at some of these locations, techniques exist to circumvent this defence and monitor the traffic passing through switches.

1.2 Proof of Concept

1.2.1 Step 1



1.3 Workarounds/Solutions

The application should use transport-level encryption (SSL or TLS) to protect all sensitive communications passing between the client and the server. Communications that should be protected include the login mechanism and related functionality, and any functions where sensitive data can be accessed or privileged actions can be performed. These areas of the application should employ their own session handling mechanism, and the session tokens used should never be transmitted over unencrypted communications. If HTTP cookies are used for transmitting session tokens, then the secure flag should be set to prevent transmission over clear-text HTTP.

Top

2. Host header attack — High

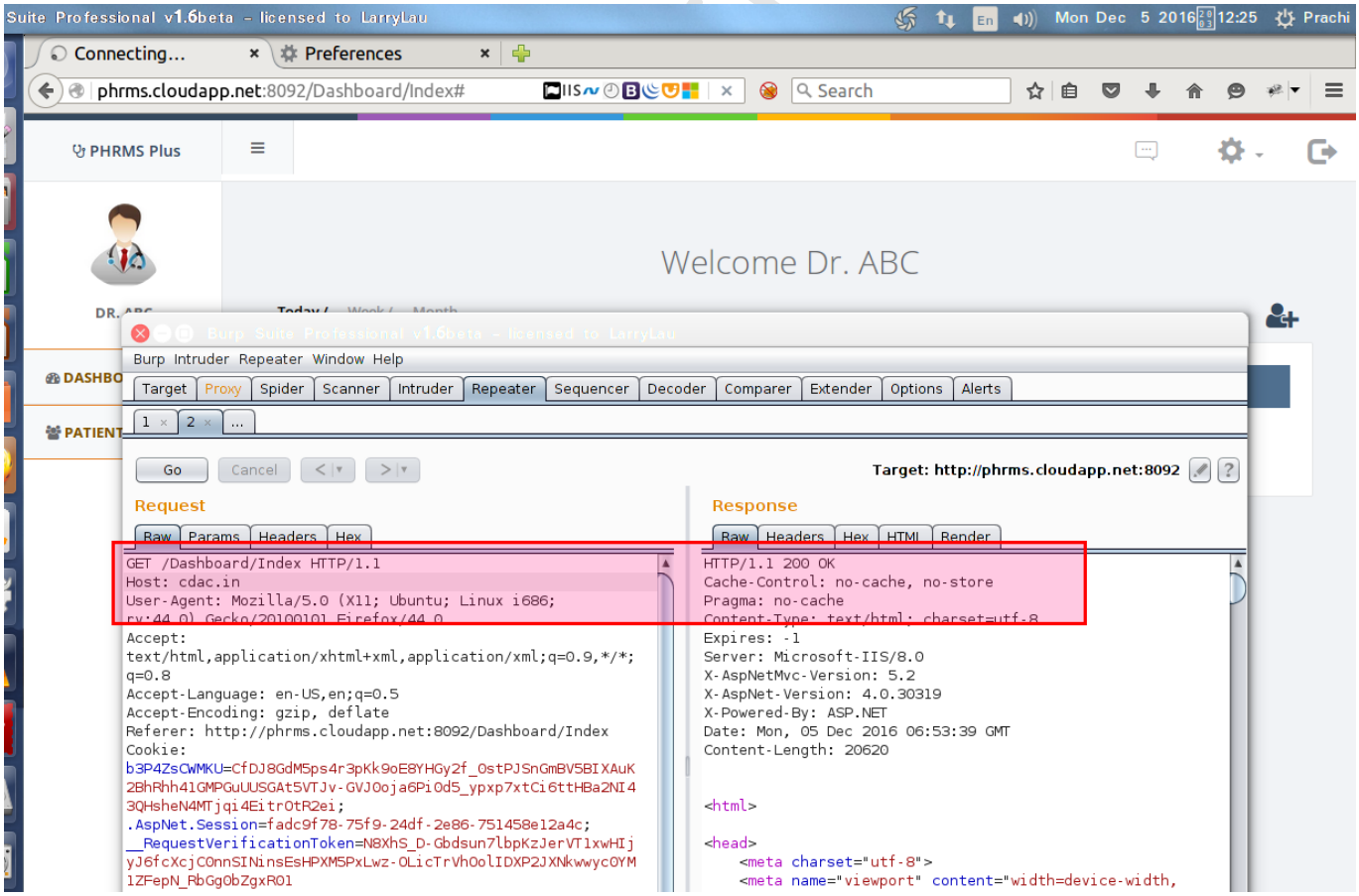
Affected Url : http://phrms.cloudapp.net:8092/

2.1 Description

An attacker can manipulate the Host header as seen by the web application and cause the application to behave in unexpected ways. Developers often resort to the exceedingly untrustworthy HTTP Host header (_SERVER[”HTTP_HOST”] in PHP). Even otherwise-secure applications trust this value enough to write it to the page without HTML-encoding it with code equivalent to: <link href=”http://_SERVER[’HOST’]” (Joomla) ...and append secret keys and tokens to links containing it: <a href=”http://_SERVER[’HOST’]?token=topsecret”> (Django, Gallery, others) ...and even directly import scripts from it: <script src=”http://_SERVER[’HOST’]/misc/jquery.js?v=1.4.4”> (Various)

2.2 Proof of Concept

2.2.1 Step 1



2.3 Workarounds/Solutions

The web application should use the SERVER_NAME instead of the Host header. It should also create a dummy vhost that catches all requests with unrecognized Host headers. This can also be done under Nginx by specifying a non-wildcard SERVER_NAME, and under Apache by using a non-wildcard serverName and turning the UseCanonicalName directive on. Consult references for detailed information.

Top

CONFIDENTIAL

3. Possible Bruteforce Attack — High

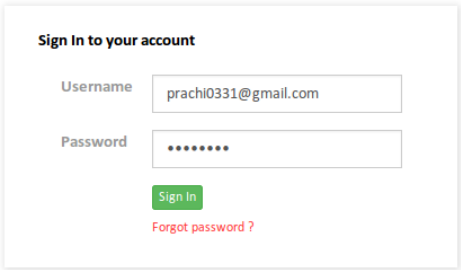
Affected Url	:	phrms.cloudapp.net
CWE	:	CWE-799, CSE-307
OWASP	:	Authentication Testing

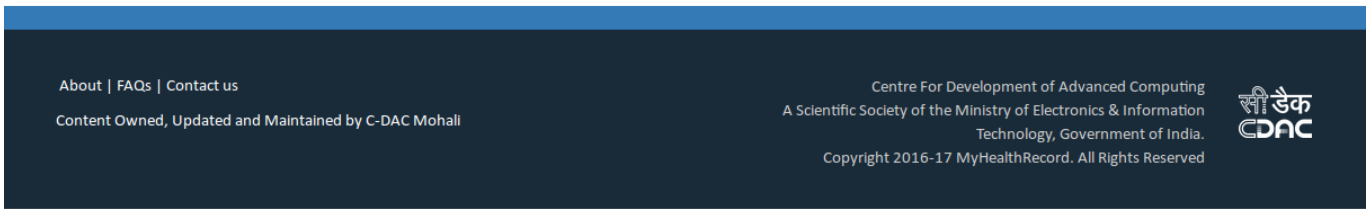
3.1 Description

The application does not properly limit the number or frequency of interactions that it has with an actor, such as the number of incoming requests. This can allow the actor to perform actions more frequently than expected. The actor could be a human or an automated process such as a virus or bot. This could be used to cause a denial of service, compromise program logic (such as limiting humans to a single vote), or other consequences. For example, an authentication routine might not limit the number of times an attacker can guess a password. Or, a web site might conduct a poll but only expect humans to vote a maximum of once a day

3.2 Proof of Concept

3.2.1 Step 1





3.3 Workarounds/Solutions

1. Implement Captcha to deal stop automated programs or bots to submit the form.
2. Implement Account lockout, restrict the number of wrong login hits to the server for the same username and password.
3. Error message should be static, in case of wrong username or wrong password, like ”Invalid username/password”, so that the actor may not get information which value to wrong while login trials.

Top

CONFIDENTIAL

4. Improper Input Validation — High

Affected Url	:	http://phrms.cloudapp.net:8082/HealthCondition
CWE	:	CWE-20
OWASP	:	Data Validation

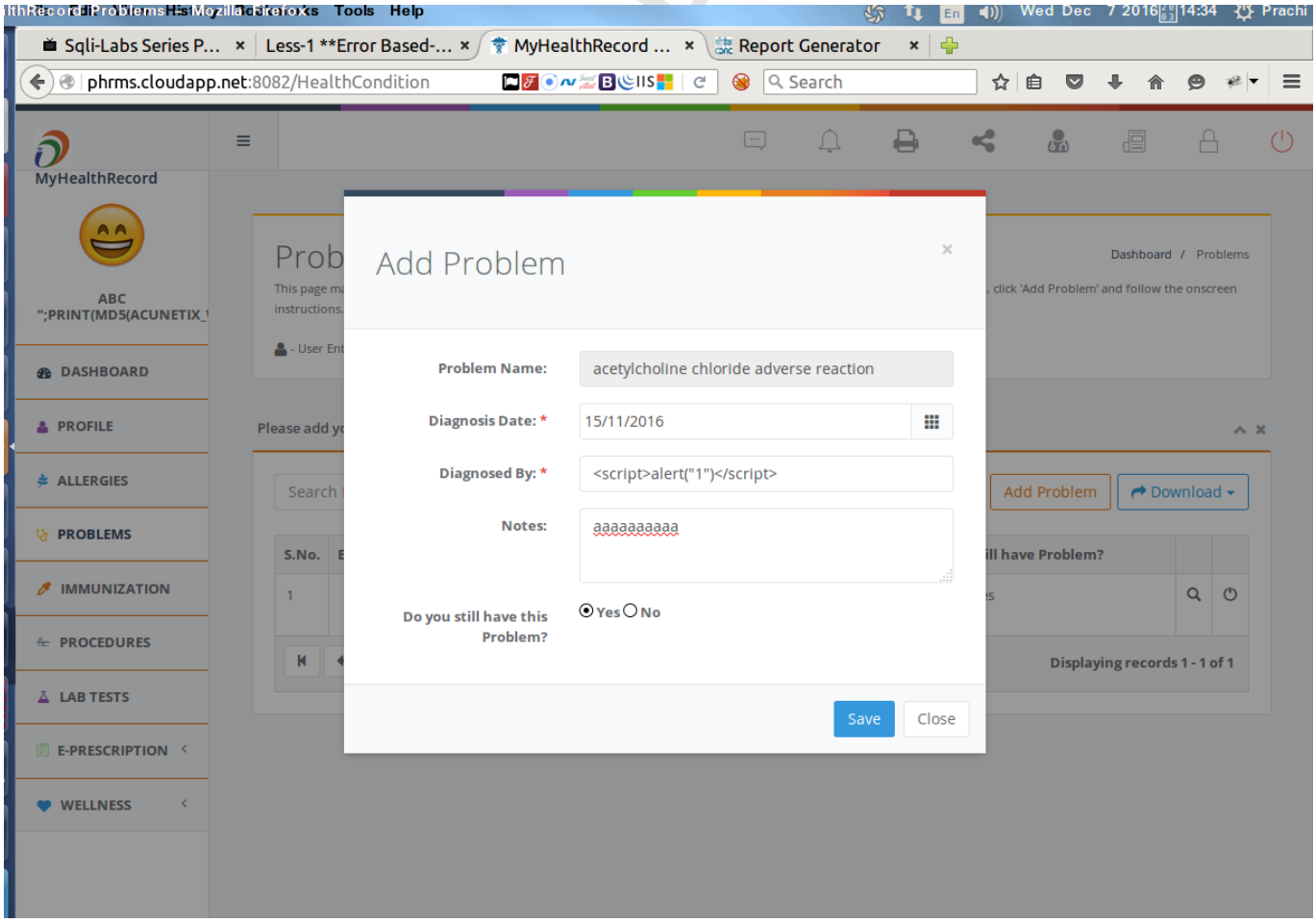
4.1 Description

The most common web application security weakness is the failure to properly validate input coming from the client or from the environment before using it. This weakness leads to almost all of the major vulnerabilities in web applications, such as cross site scripting, SQL injection, interpreter injection, locale/Unicode attacks, file system attacks, and buffer overflows.

4.2 Proof of Concept

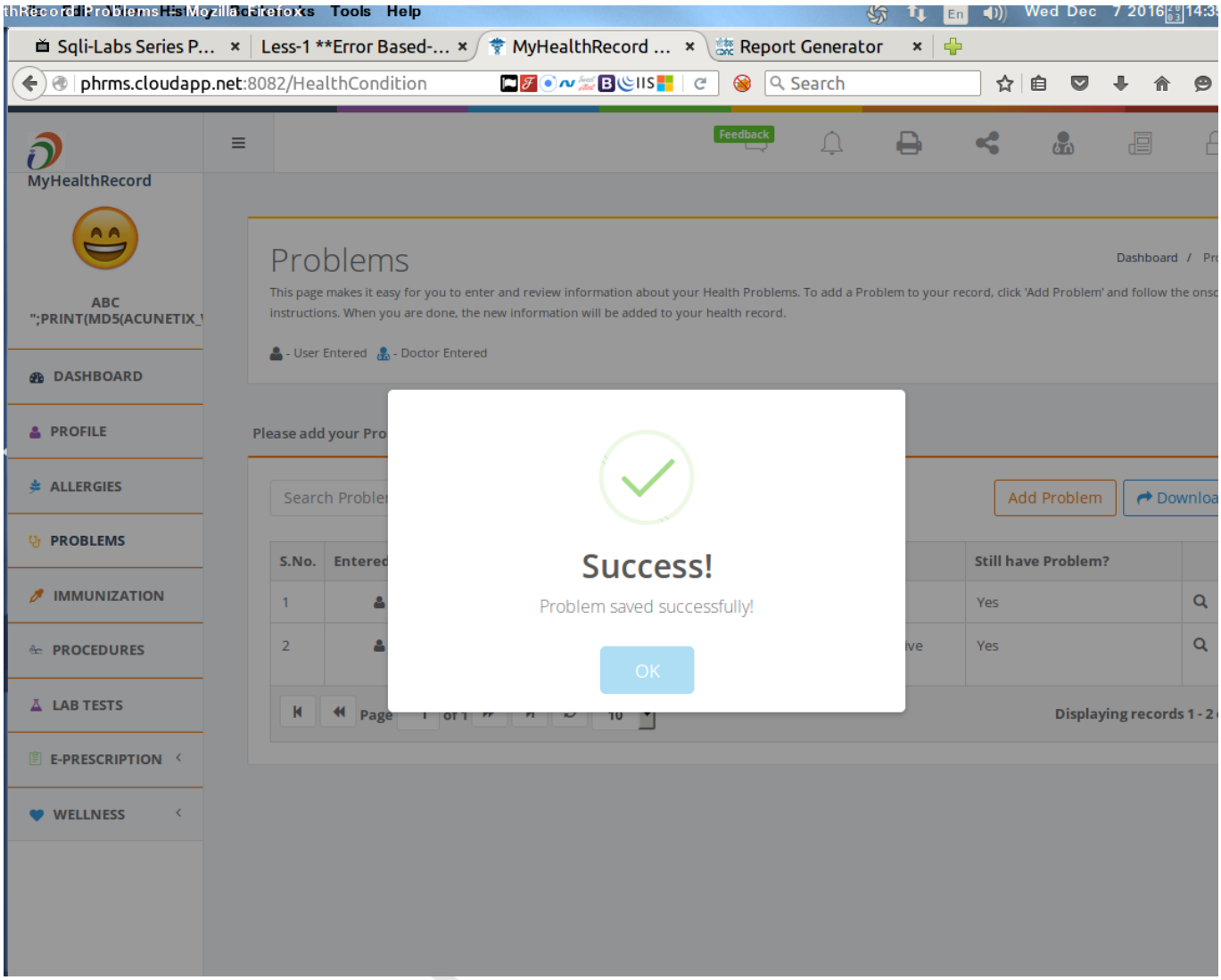
4.2.1 Step 1

After patient login access the above url and try to enter some special character as input.



4.2.2 Step 2

Here it is successfully saved.



4.3 Workarounds/Solutions

This can be a high risk vulnerability and can be underestimated. Mitigating this vulnerability uses a two-fold approach. Ensure all user-controllable data is validated after it is inputted and again before it is outputted to users. Blacklisting is an approach which consists of checking the input data for malicious characters but a more effective approach is whitelisting. Whitelisting consists of only allowing certain characters to be submitted. For example checking if data submitted is alphanumeric and rejecting the request if it is not. You can use an approach like this after data is submitted and then perform a similar approach before data is outputted to the user.

Top

5. Cross Site Request Forgery — High

Affected Url	:	http://phrms.cloudapp.net:8092/Patient/Index
CWE	:	CWE-352
OWASP	:	Session Management

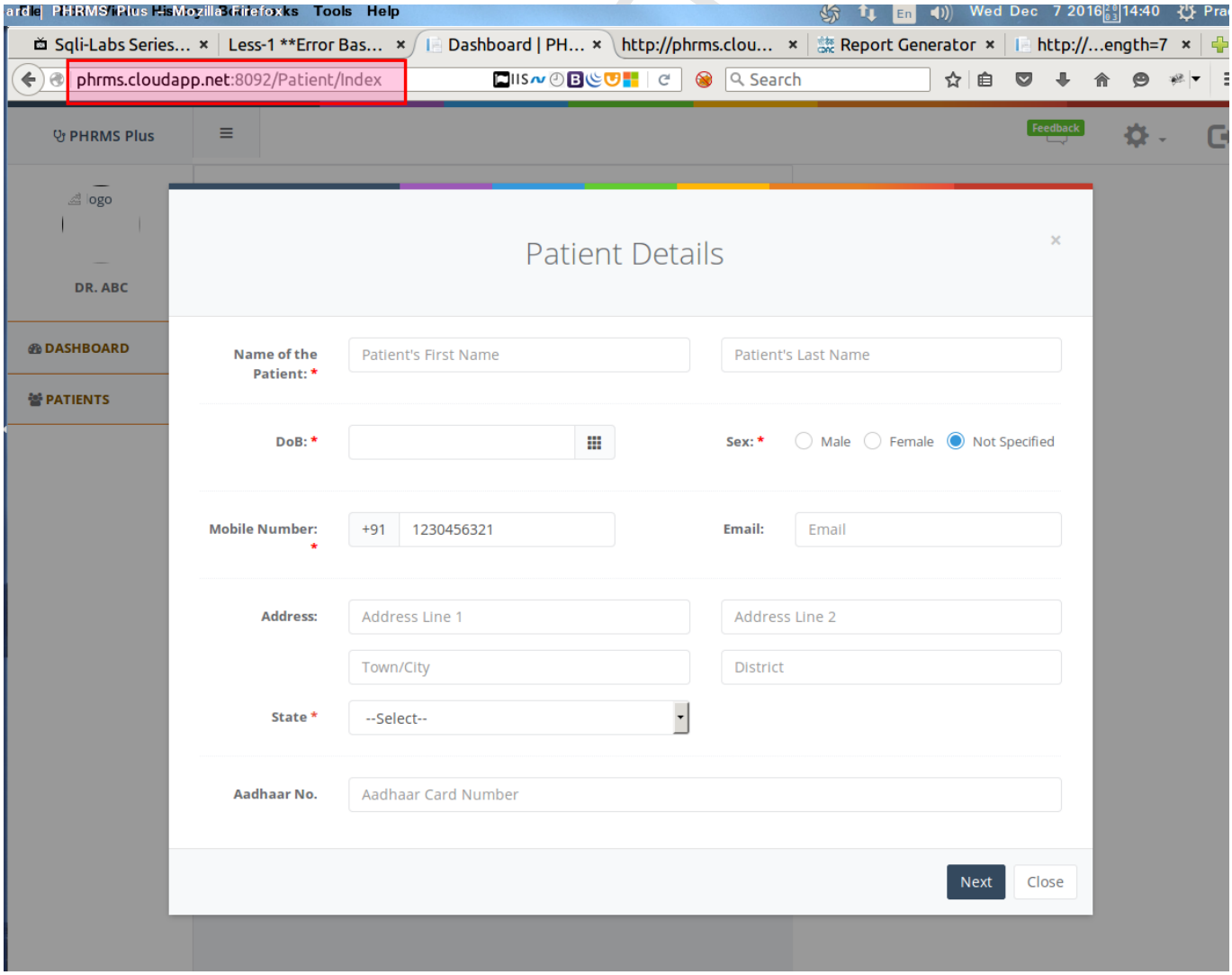
5.1 Description

When a web server is designed to receive a request from a client without any mechanism for verifying that it was intentionally sent, then it might be possible for an attacker to trick a client into making an unintentional request to the web server which will be treated as an authentic request. This can be done via a URL, image load, XMLHttpRequest, etc. and can result in exposure of data or unintended code execution.

5.2 Proof of Concept

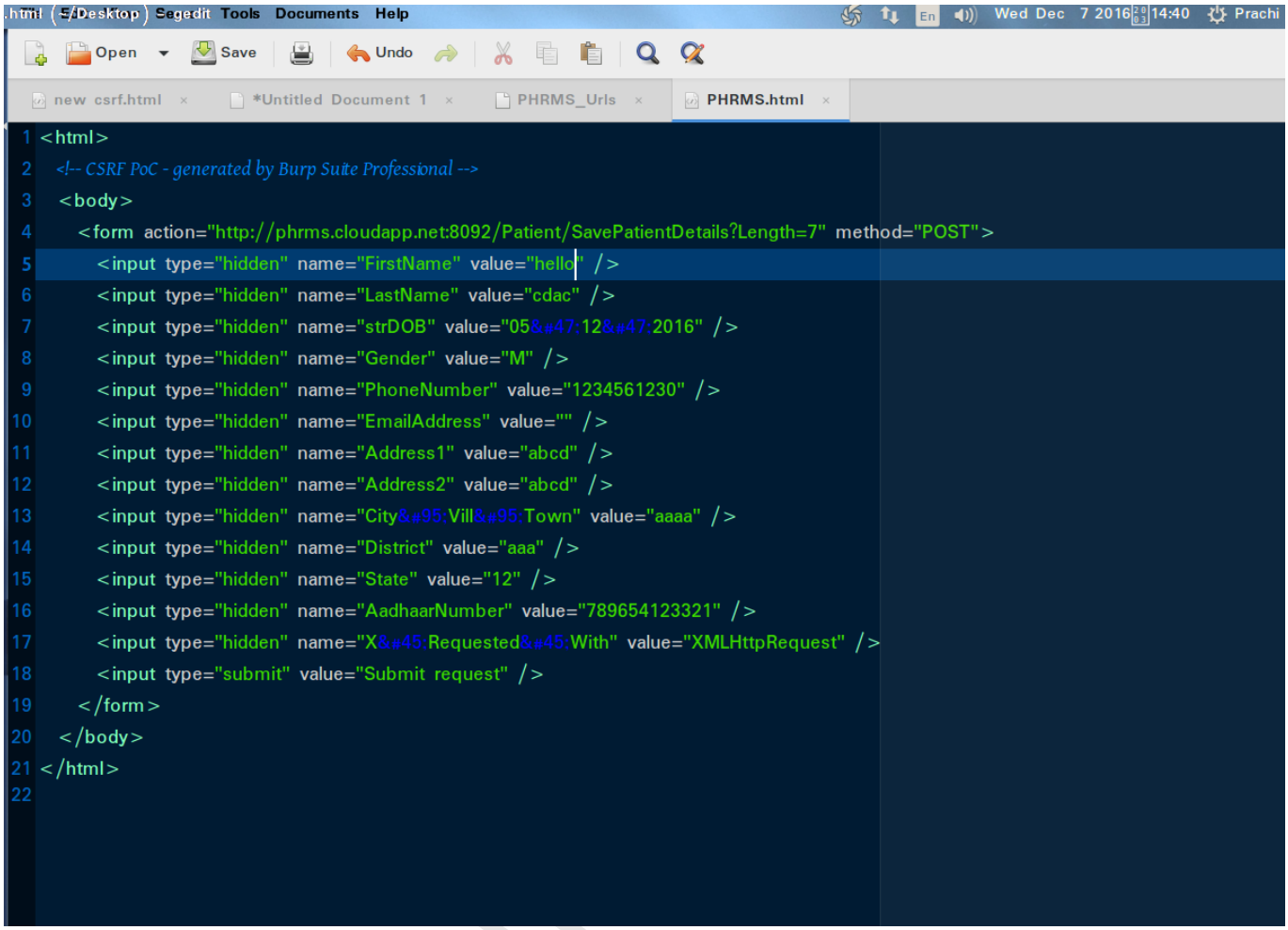
5.2.1 Step 1

Open the above url



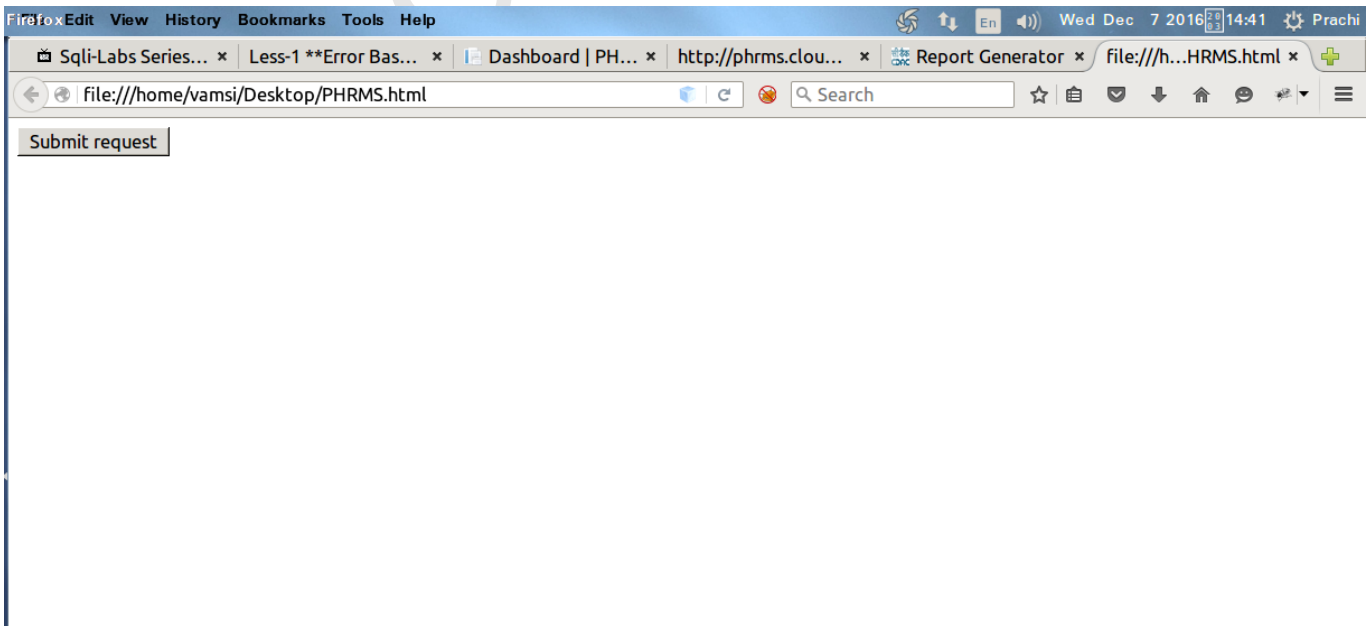
5.2.2 Step 2

Create a .html page for csrf attack



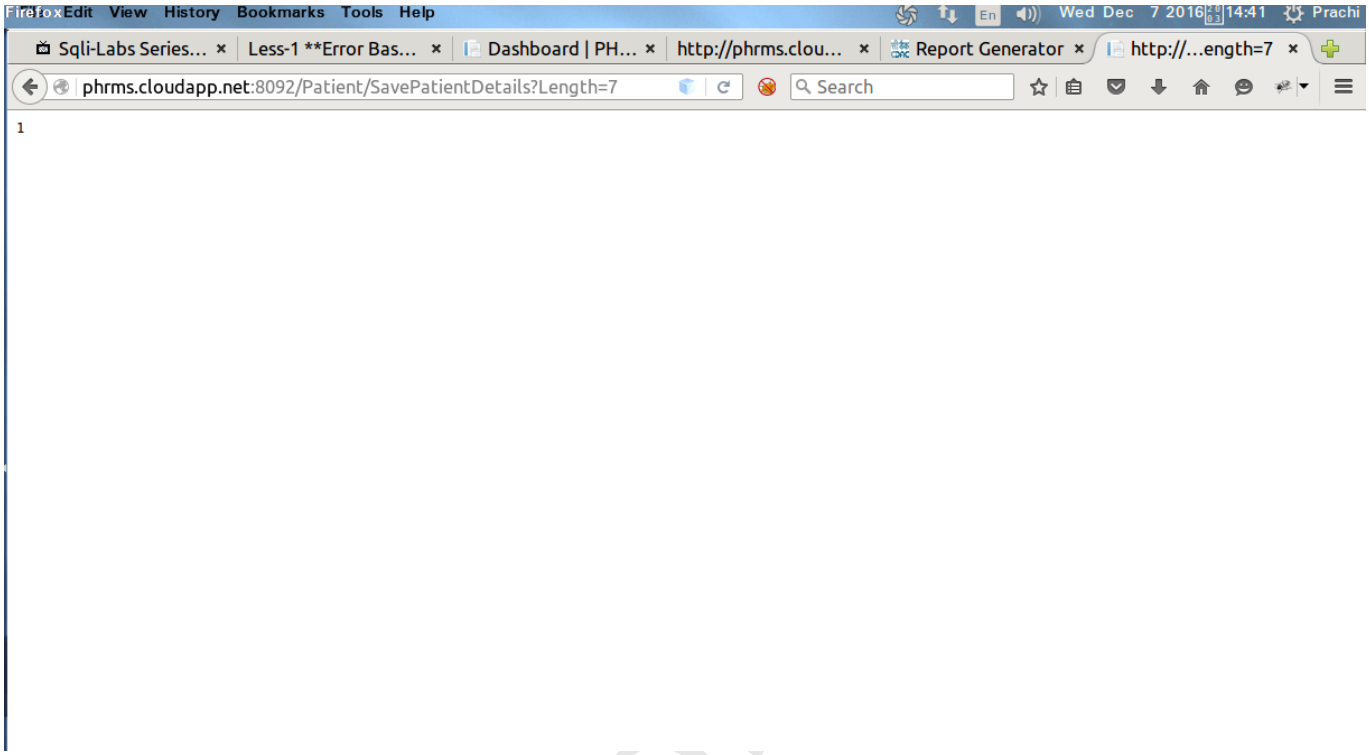
5.2.3 Step 3

Open the same page in the local browser and click on submit button



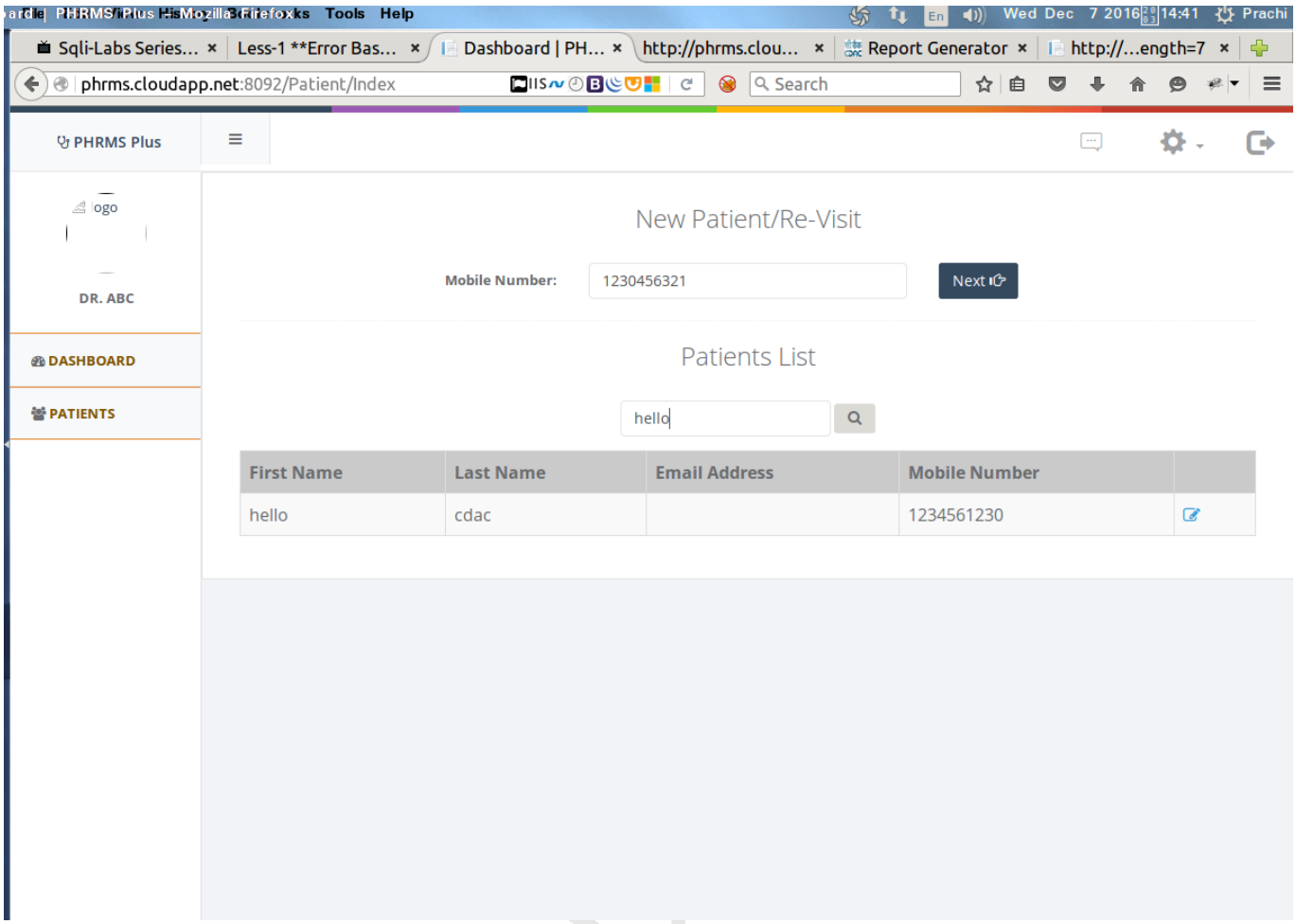
5.2.4 Step 4

The info has been updated.



5.2.5 Step 5

CONFID



5.3 Workarounds/Solutions

1:CSRF is currently difficult to detect reliably using automated techniques. This is because each application has its own implicit security policy that dictates which requests can be influenced by an outsider and automatically performed on behalf of a user, versus which requests require strong confidence that the user intends to make the request.
2:Generate a unique token for each entity
3:Use CAPTCHA when possible
4:Identify dangerous operations and always demand confirmation of dangerous actions
5:Use protection frameworks like OWASP CSRF Guard, PHP CSRF Guard, Net CSRF Guard

Top

6. Vulnerable Javascript library — High

Affected Url : http://phrms.cloudapp.net:8092/vendor/peity/test/jquery-1.6.2.min.js

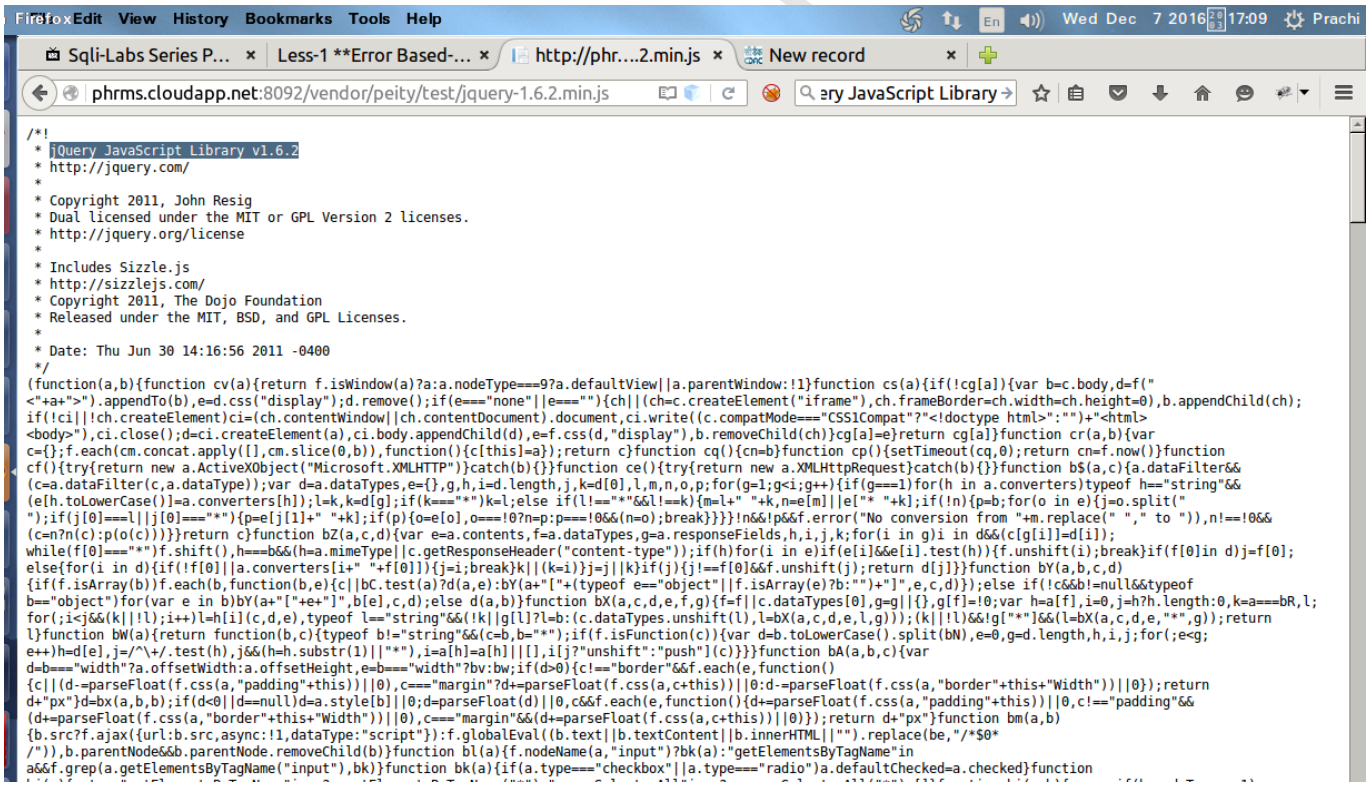
OWASP : Configuration

6.1 Description

You are using a vulnerable Javascript library. One or more vulnerabilities were reported for this version of the Javascript library. Consult Attack details and Web References for more information about the affected library and the vulnerabilities that were reported.

6.2 Proof of Concept

6.2.1 Step 1



6.3 Workarounds/Solutions

Upgrade to the latest version.

Top

7. Microsoft IIS tilde directory enumeration — High

Affected Url : phrms.cloudapp.net

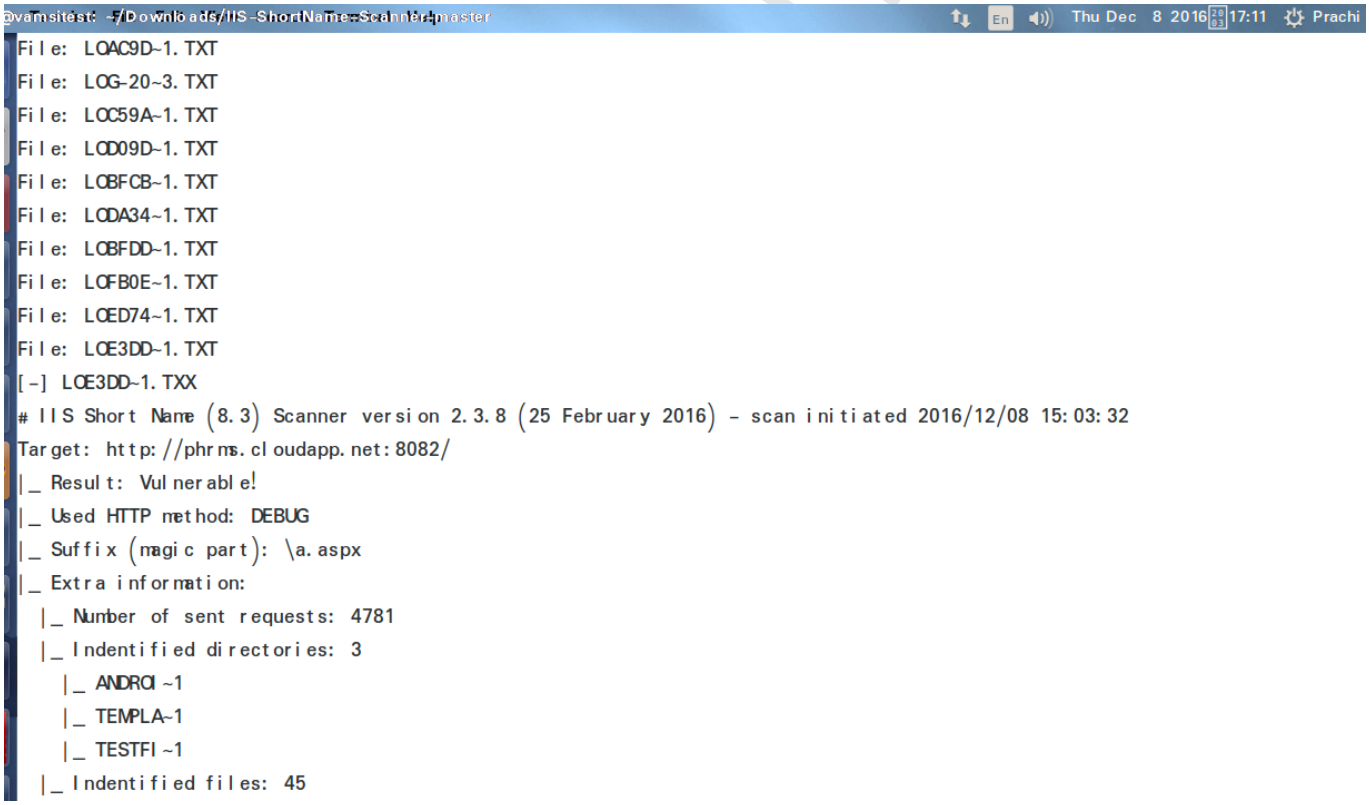
CWE : CWE-20

7.1 Description

It is possible to detect short names of files and directories which have an 8.3 file naming scheme equivalent in Windows by using some vectors in several versions of Microsoft IIS. For instance, it is possible to detect all short-names of ".aspx" files as they have 4 letters in their extensions. This can be a major issue especially for the .Net websites which are vulnerable to direct URL access as an attacker can find important files and folders that they are not normally visible.

7.2 Proof of Concept

7.2.1 Step 1



7.3 Workarounds/Solutions

Consult the "Prevention Technique(s)" section from Soroush Dalili's paper on this subject. A link to this paper is listed in the recommendation section below.

Top

8. Session Mismanagement High

Affected Url	:	phrms.cloudapp.net:8092
CWE	:	CWE-384,CWE-613
OWASP	:	SESSION MANAGEMENT

8.1 Description

Application functions related to authentication and session management are often not implemented correctly,allowing attackers to compromise passwords, keys, session tokens, or exploit other implementation flaws to assume other users? identities.Such flaws may allow some or even all accounts to be attacked. Once successful, the attacker can do anything the victim could do.Privileged accounts are frequently targeted.

8.1 Case 1

8.1.1 Session Replay Attack

8.1.2 Description

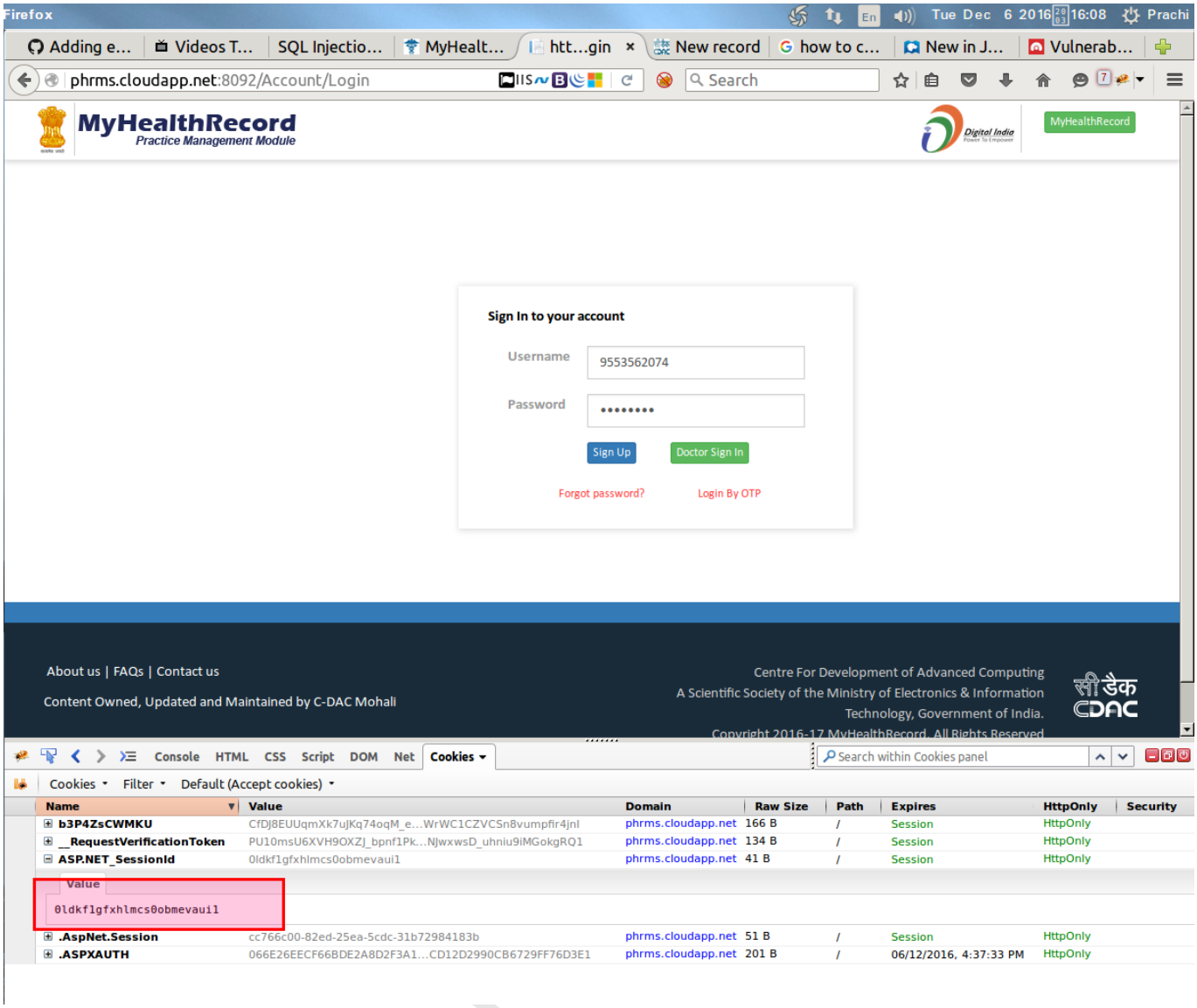
This attack targets the reuse of valid session ID to spoof the target system in order to gain privileges. The attacker tries to reuse a stolen session ID used previously during a transaction to perform spoofing and session hijacking. Another name for this type of attack is Reusing of Session id.

8.1.3 Proof of Concept

8.1.3.1 Step 1

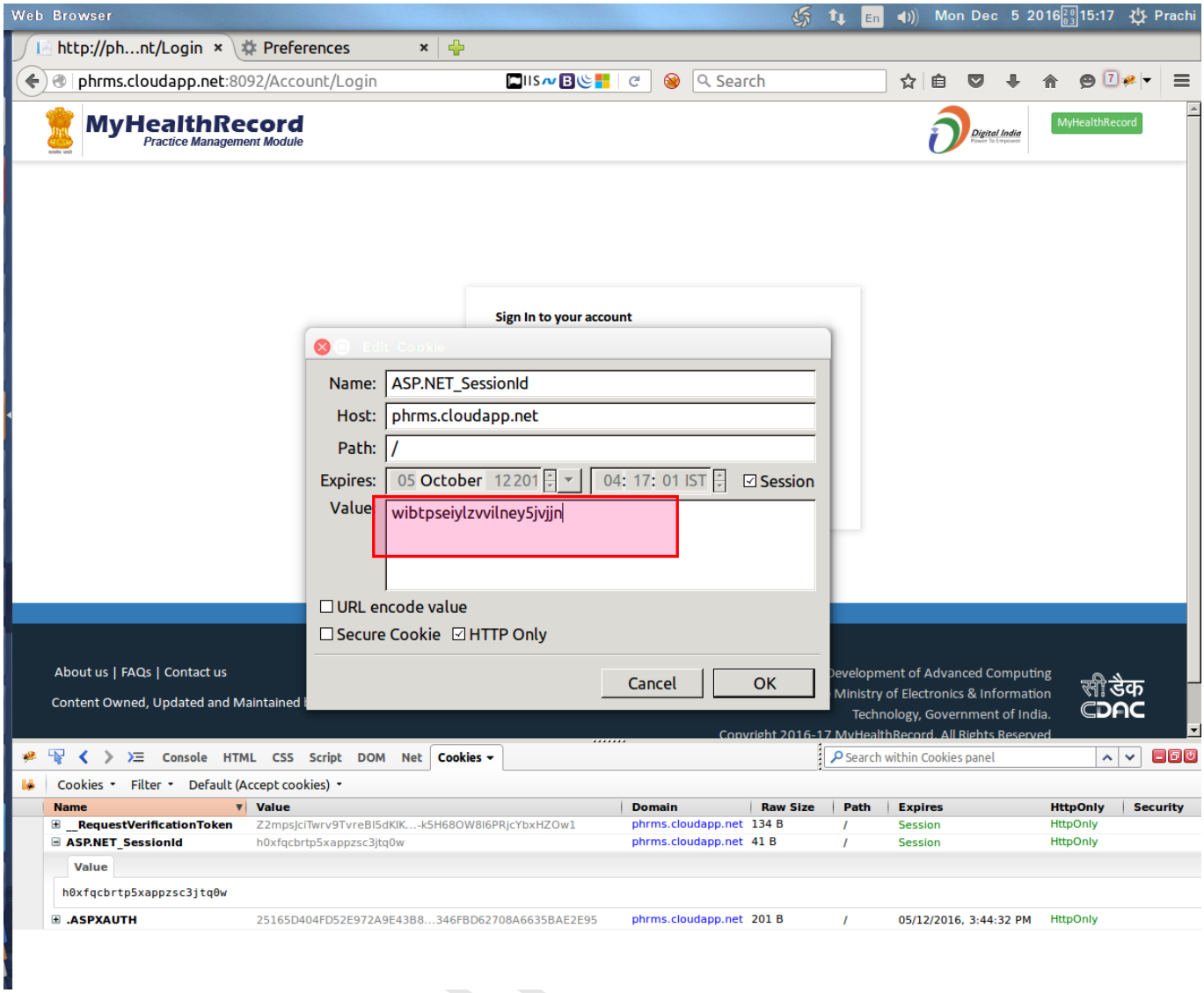
Accessed this url phrms.cloudapp.net:8092 , web server issued session id to client and observed session i d

CONFIDENTIAL



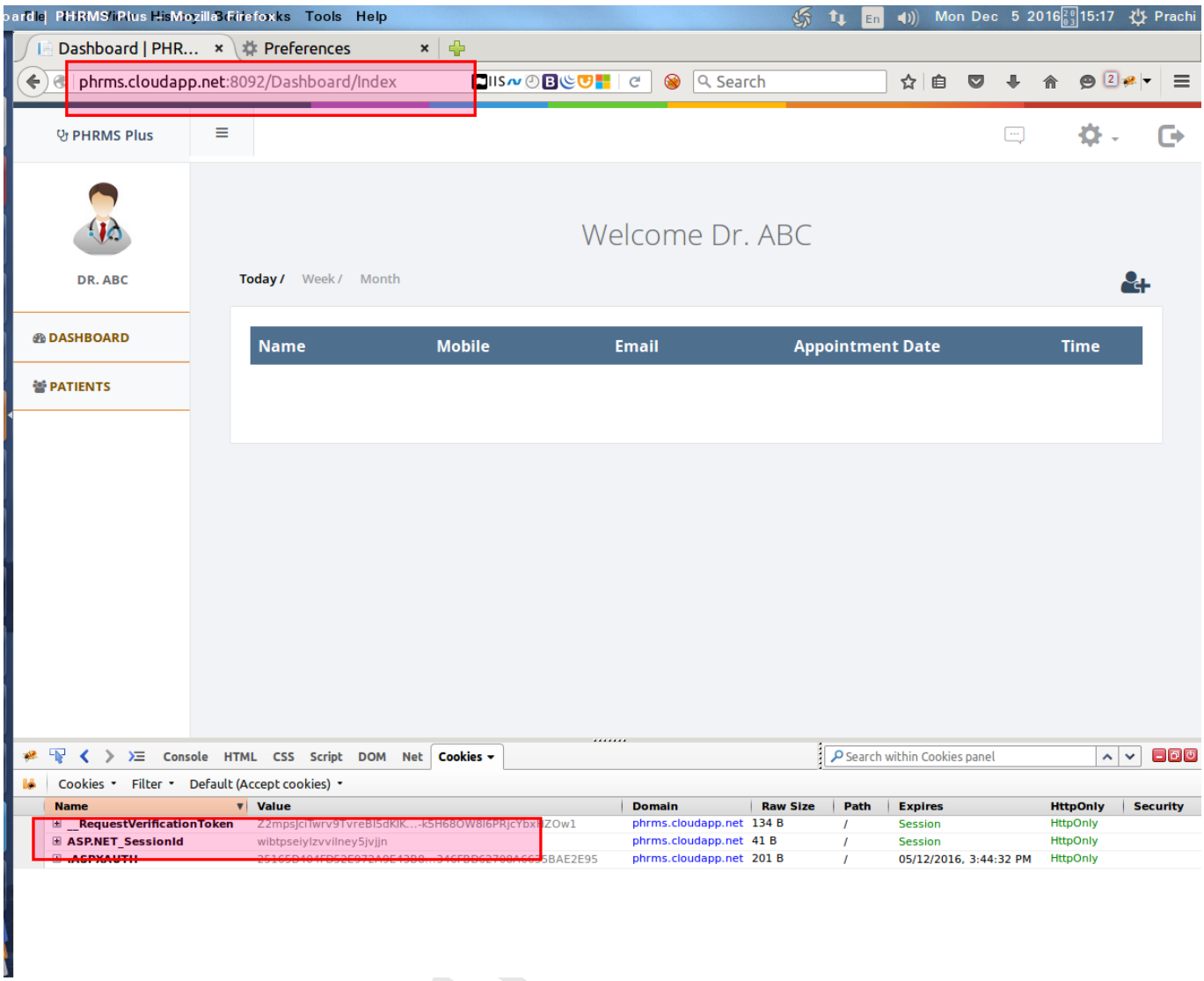
8.1.3.2 Step 2

In this step using cookie editor we modified session id with used session id which is used previously.



8.1.3.3 Step 3

Now you can see we logged in successfully with used session id web server accepted used session id and responding well



8.1.4 Workarounds/Solutions

Always invalidate a session ID after the user logout. Setup a session time out for the session IDs. Protect the communication between the client and server. For instance it is best practice to use SSL to mitigate man in the middle attack. Do not code send session ID with GET method, otherwise the session ID will be copied to the URL. In general avoid writing session IDs in the URLs. URLs can get logged in log files, which are vulnerable to an attacker. Encrypt the session data associated with the session ID.

Top

8.2 Case 2

8.2.1 Session Hijacking

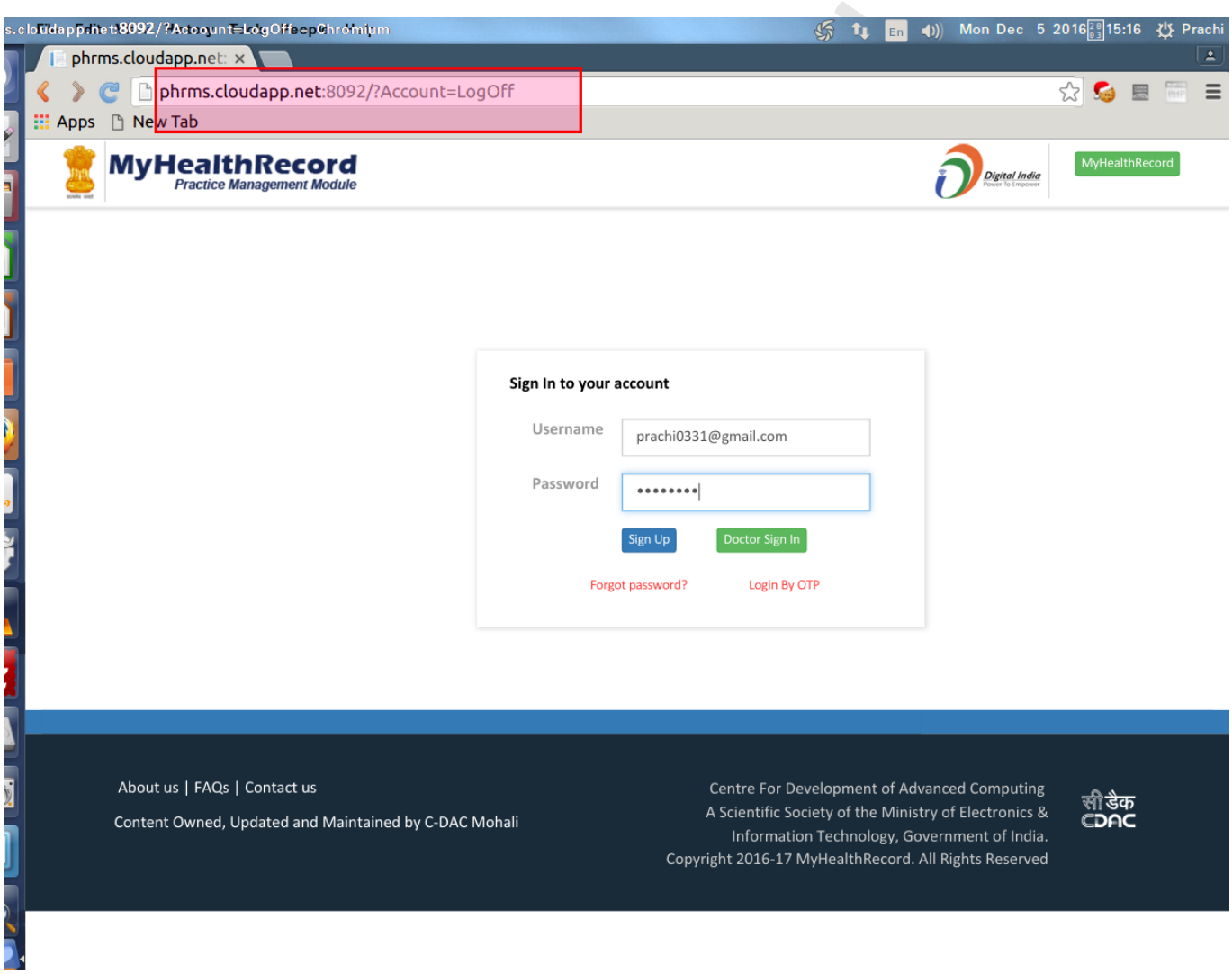
8.2.2 Description

session hijacking, sometimes also known as cookie hijacking is the exploitation of a valid computer session.A cookie must contain some amount of hard-to-guess data.The harder it is to forge a valid cookie,the harder is to break into legitimate user's session. If an attacker can guess the cookie used in an active session of a legitimate user,he/she will be able to fully impersonate that user.

8.2.3 Proof of Concept

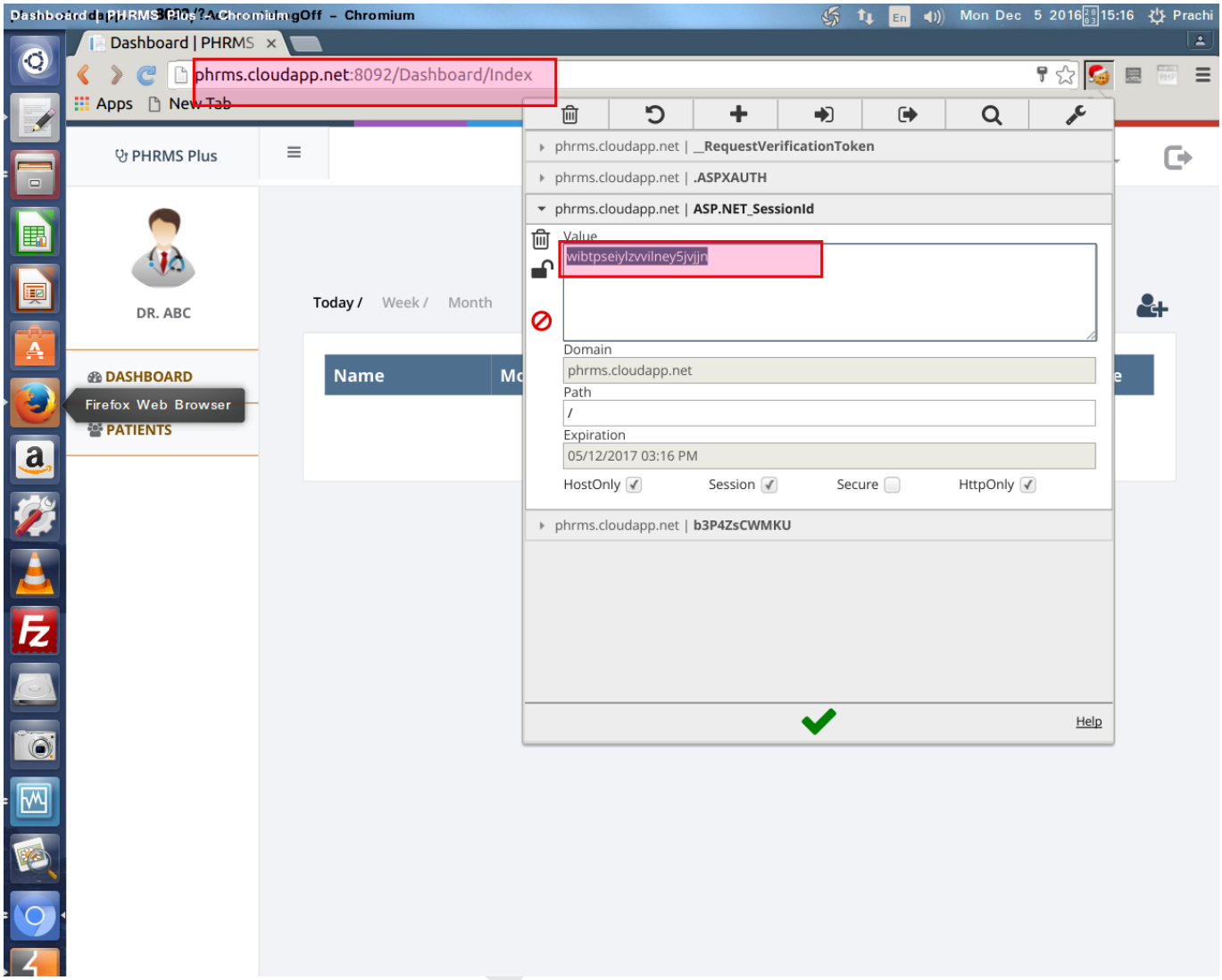
8.2.3.1 Step 1

Try to login from the above page



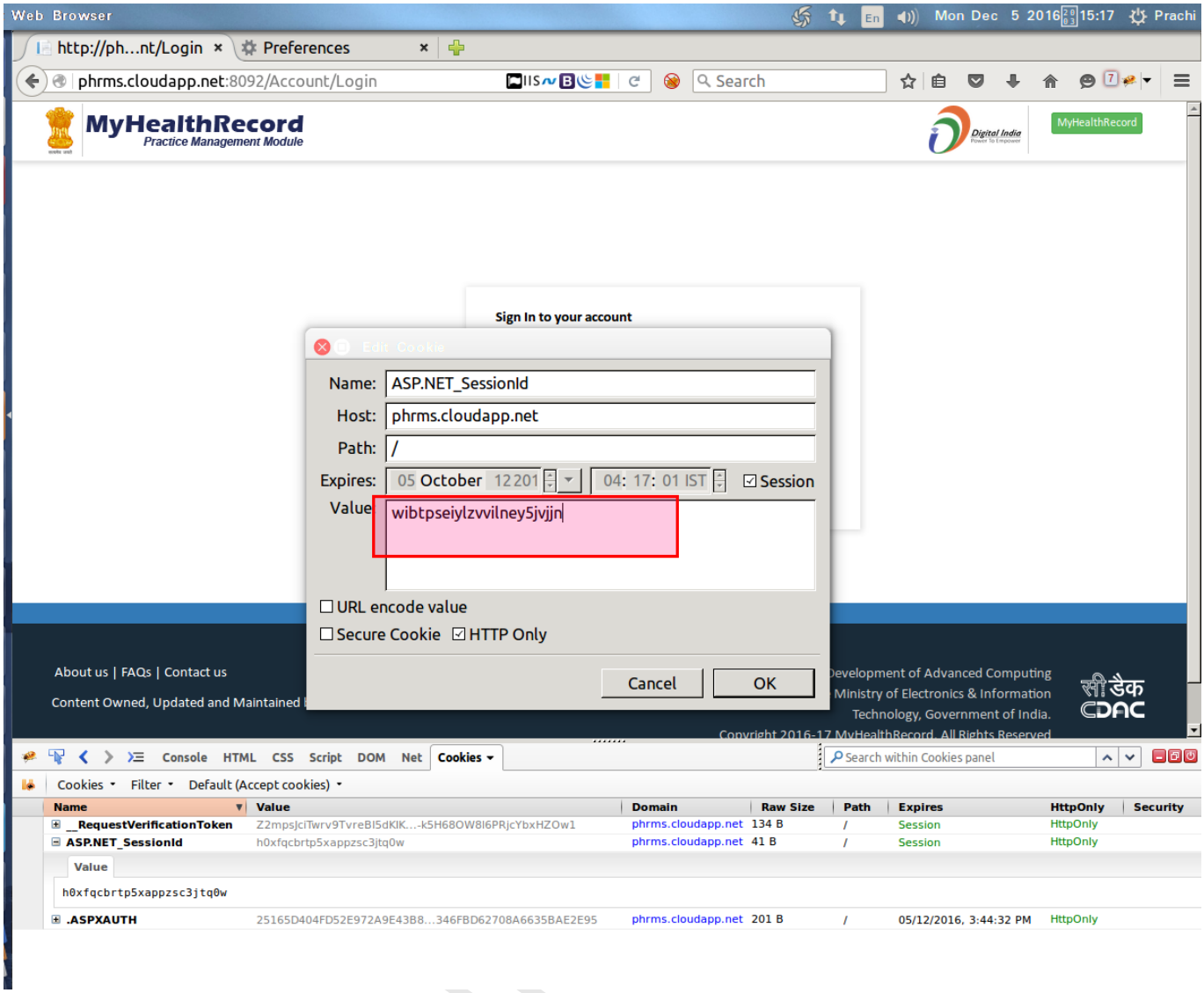
8.2.3.2 Step 2

After login copied the internal url as well as the session id value

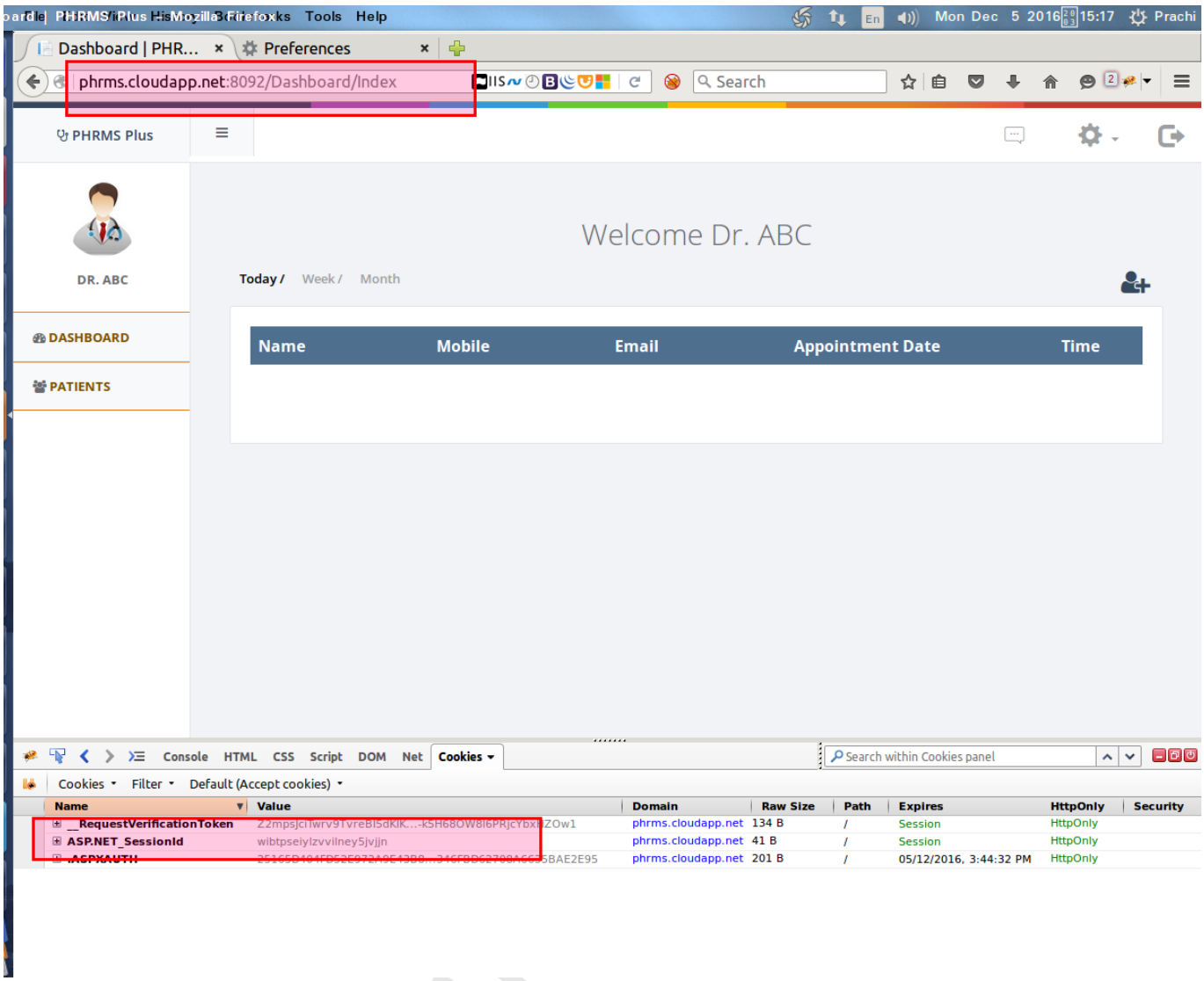


8.2.3.3 Step 3

Access the url http://phrms.cloudapp.net:8092Now paste the same url and the session id value in the diff erent browser and successfully logged in without passing the credentials


Web Browser
Mon Dec 5 2016 15:17 Prachi
http://ph...nt/Login
Preferences
phrms.cloudapp.net:8092/Account/Login
Search
MyHealthRecord
Practice Management Module
Digital India
MyHealthRecord
Sign In to your account
Edit Cookie
Name: ASP.NET_SessionId
Host: phrms.cloudapp.net
Path: /
Expires: 05 October 12201 04: 17: 01 IST Session
Value wibtpseiylzvvilney5jvjjn
URL encode value
Secure Cookie HTTP Only
Cancel
OK
About us | FAQs | Contact us
Content Owned, Updated and Maintained
Development of Advanced Computing
Ministry of Electronics & Information
Technology, Government of India.
Copyright 2016-17 MyHealthRecord. All Rights Reserved
Console HTML CSS Script DOM Net Cookies
Search within Cookies panel
Cookies Filter Default (Accept cookies)
Name Value Domain Raw Size Path Expires HttpOnly Security
__RequestVerificationToken Z2mpsJciTwrv9TvreBI5dKIK...-k5H68OW8I6PRjcYbxHZOw1 phrms.cloudapp.net 134 B / Session HttpOnly
ASP.NET_SessionId h0xfqcbrtp5xappzsc3jtq0w phrms.cloudapp.net 41 B / Session HttpOnly
Value
h0xfqcbrtp5xappzsc3jtq0w
.ASPXAUTH 25165D404FD52E972A9E43B8...346FBD62708A6635BAE2E95 phrms.cloudapp.net 201 B / 05/12/2016, 3:44:32 PM HttpOnly

8.2.3.4 Step 4



8.2.4 Other affected URL

8.2.4.1 http://phrms.cloudapp.net:8092/Dashboard/Index

8.2.5 Workarounds/Solutions

Apply Secure and HttpOnly flags.
Use of a long random number or string as the session key. This reduces the risk that an attacker could simply guess a valid session key through trial and error or brute force attacks.
use HTTPS to protect the session ID during transmission.
Set the Domain and Path parameters for the cookie correctly.

Top

9. Unwanted Methods Enabled — Medium

Affected Url	:	phrms.cloudapp.net
CWE	:	CWE-749
OWASP	:	Configuration Management

9.1 Description

HTTP OPTIONS method is enabled on this web server. The OPTIONS method is used by the client to find out what are the HTTP methods and other options supported by a web server.

9.2 Proof of Concept

9.2.1 Step 1

9.3 Workarounds/Solutions

Disable the OPTIONS Method.

Top

10. Debug method Enabled — Medium

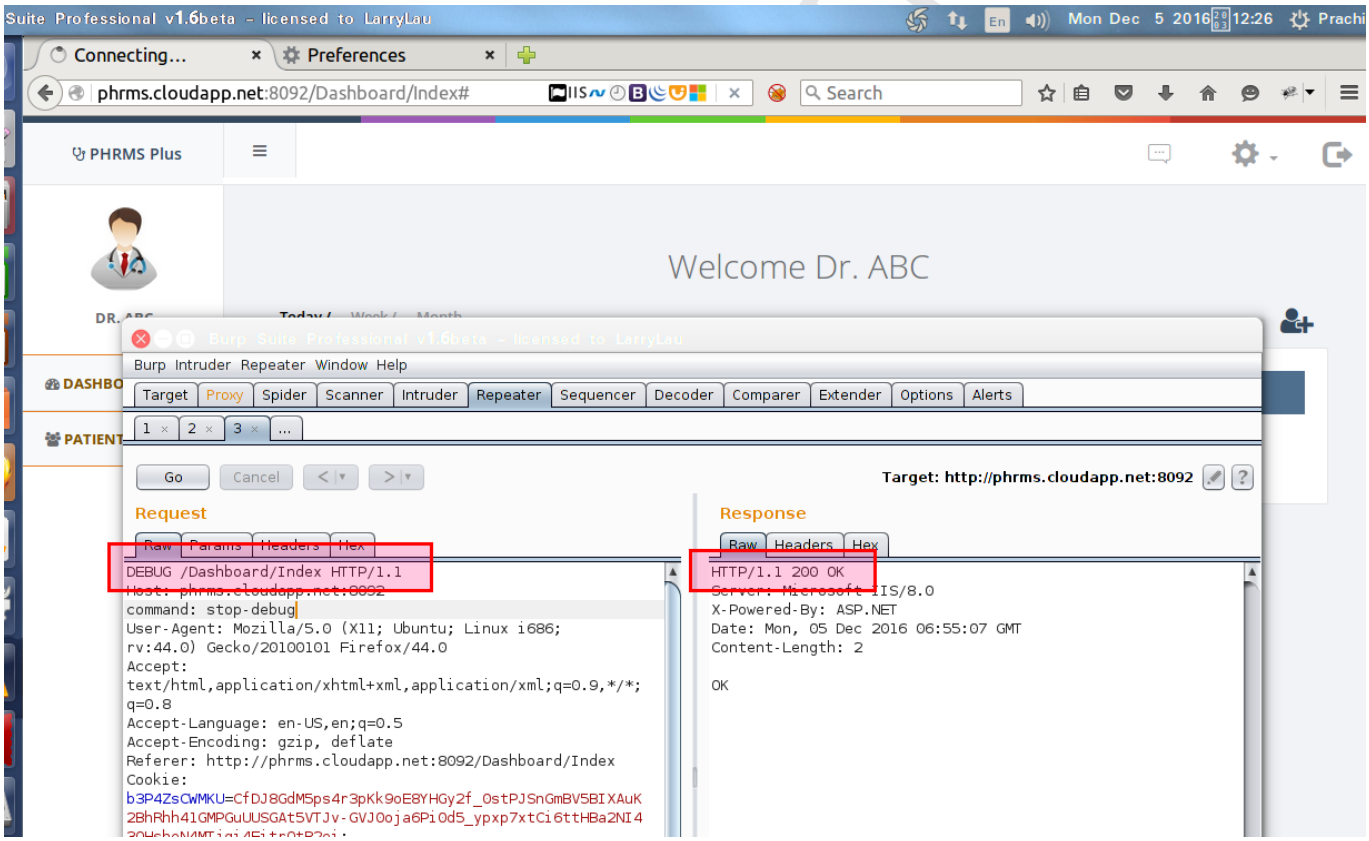
Affected Url : phrms.cloudapp.net:8092

10.1 Description

ASP.NET debugging is enabled on this application. It is recommended to disable debug mode before deploying a production application. By default, debugging is disabled, and although debugging is frequently enabled to troubleshoot a problem, it is also frequently not disabled again after the problem is resolved.

10.2 Proof of Concept

10.2.1 Step 1



10.3 Workarounds/Solutions

Debug method should be turned off in production environment.

Top

11. Vulnerable Remember Password — Medium

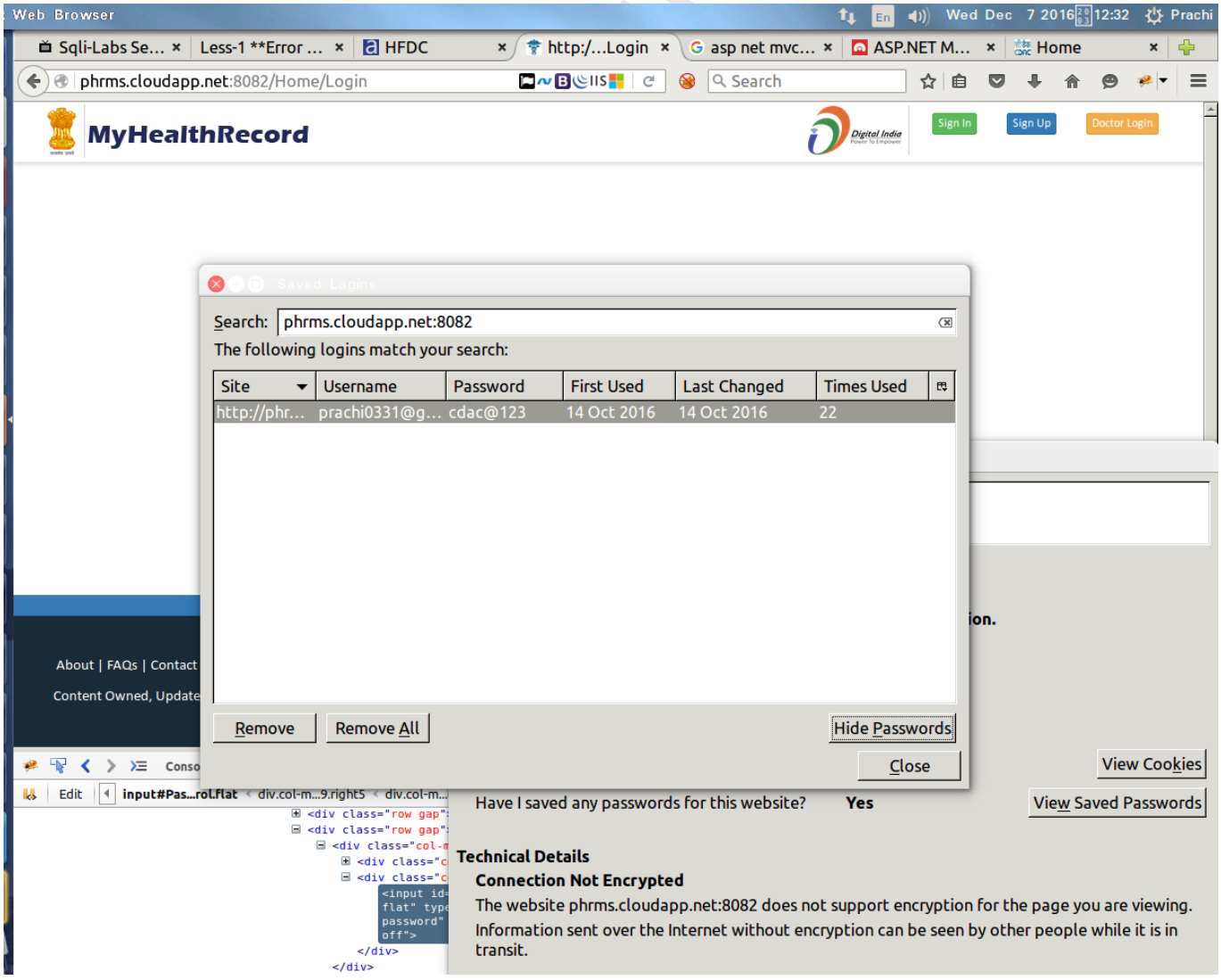
Affected Url : phrms.cloudapp.net:8092

11.1 Description

Some websites will offer custom ”remember me” functionality to allow users to persist log ins on a specific client system.If an attacker can gain access to the victim’s browser (e.g. through a Cross Site Scripting attack, or through a shared computer), then they can retrieve the stored passwords. It is not uncommon for browsers to store these passwords in an easily retrievable manner, but even if the browser were to store the passwords encrypted and only retrievable through the use of a master password, an attacker could retrieve the password by visiting the target web application’s authentication form, entering the victim’s username, and letting the browser to enter the password.

11.2 Proof of Concept

11.2.1 Step 1



11.3 Workarounds/Solutions

Ensure that no credentials are stored in clear text or are easily retrievable in encoded or encrypted forms in cookies.

Top

CONFIDENTIAL

12. Action Spoofing (Clickjacking) Medium

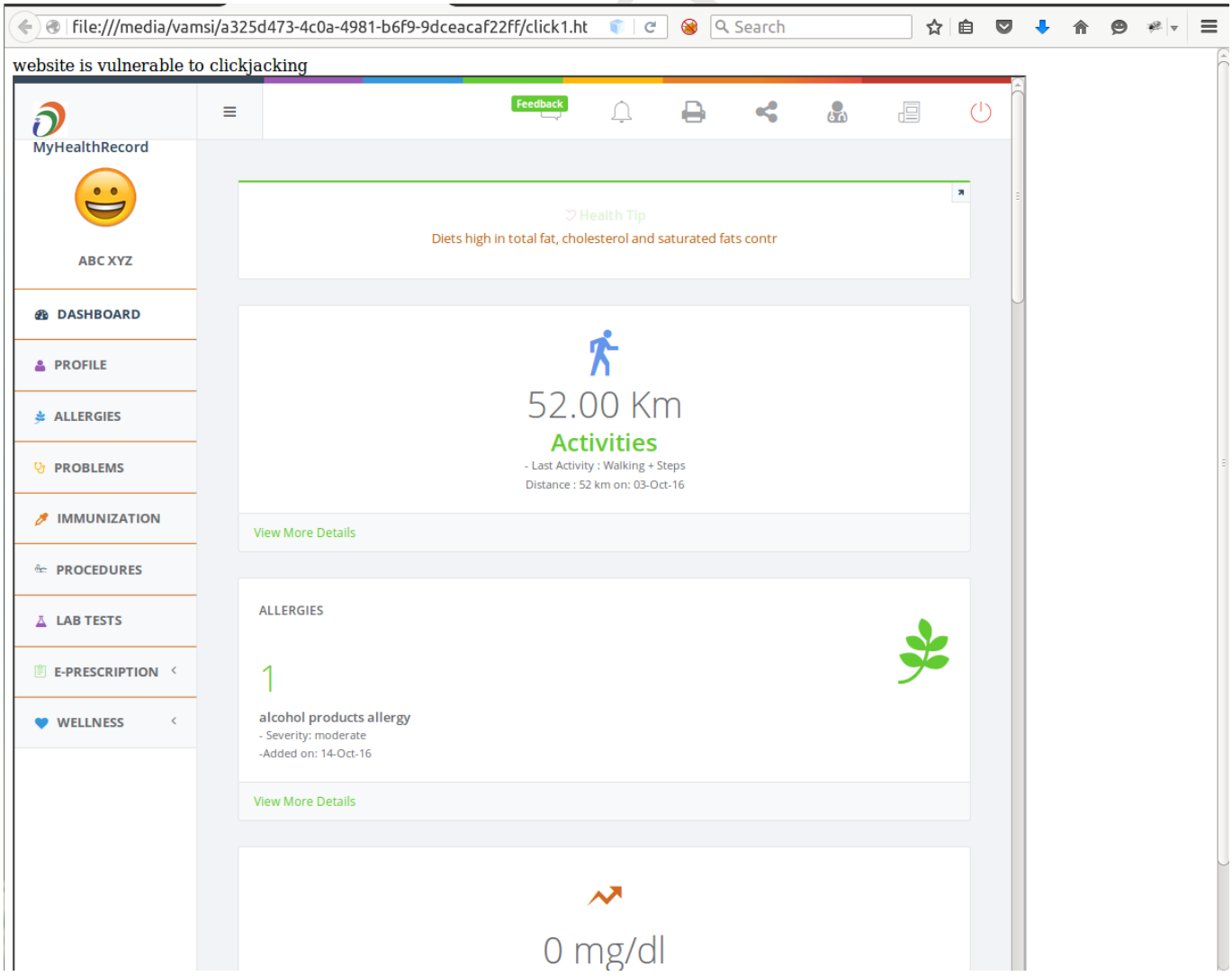
Affected Url	:	http://phrms.cloudapp.net:8082/Account/Dashboard
CWE	:	CWE-693
OWASP	:	Configuration Management

12.1 Description

In a clickjacking attack the victim is tricked into unknowingly initiating some action in one system while interacting with the UI from seemingly completely different system. While being logged in to some target system, the victim visits the attacker's malicious site which displays a UI that the victim wishes to interact with. In reality, the clickjacked page has a transparent layer above the visible UI with action controls that the attacker wishes the victim to execute.

12.2 Proof of Concept

12.2.1 Step 1



12.3 Workarounds/Solutions

You should review the application functions that are accessible from with in the response and determine whether they can be used by application users to perform any sensitive actions within the application. If so, then a framing attack targeting this response may result in unauthorized actions. To effectively prevent framing attacks, the application should return a response header with the name X-Frame-Options and the value DENY to prevent framing altogether, or the value SAMEORIGIN to allow framing only by pages on the same origin as the response itself.

Top

CONFIDENTIAL

13. Tracing Error — Medium

Affected Url : phrms.cloudapp.net:8082

13.1 Description

ASP.NET tracing enables you to view diagnostic information about a single request for an ASP.NET page. ASP.NET tracing enables you to follow a page's execution path, display diagnostic information at run time, and debug your application. ASP.NET tracing can be integrated with system-level tracing to provide multiple levels of tracing output in distributed and multi-tier applications.

13.2 Proof of Concept

13.2.1 Step 1

```
root@vamsitest:~# nc 13.76.135.224 8082
TRACE / HTTP/1.0

HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Server: Kestrel
X-Powered-By: ASP.NET
Date: Wed, 07 Dec 2016 11:43:55 GMT
Connection: close
```

13.3 Workarounds/Solutions

You can protect your trace.axd file by implementing IIS Basic Authentication or you can add location tag in your web.config file.

Top

14. Unused Ports/Services — Low

Affected Url : http://phrms.cloudapp.net

14.1 Description

Open ports allow hackers to:
1.Configure the service to distribute content: Unused services tend to be left with default configurations, which are not always secure, or may be using default passwords.
2.Exploit old versions of unused software: Unused services tend to be forgotten, which means that they not get updated. Old versions of software tend to be full of known vulnerabilities.
3.Gain better information on your network: Some services give an attacker easy access to certain information, at the very least, they can have a very good guess on the operating system that the server is running, which is already a good head start.

14.2 Proof of Concept

14.2.1 Step 1

```
root@ramsitest:~# nmap 13.76.135.224

Starting Nmap 6.47 ( http://nmap.org ) at 2016-12-05 12:29 IST
Nmap scan report for 13.76.135.224
Host is up (0.014s latency).
Not shown: 981 filtered ports
PORT     STATE  SERVICE
21/tcp   open   ftp
25/tcp   open   smtp
53/tcp   open   domain
80/tcp   open   http
89/tcp   open   su-mit-tg
110/tcp  open   pop3
113/tcp  closed ident
119/tcp  open   nntp
135/tcp  open   msrpc
143/tcp  open   imap
443/tcp  open   https
1433/tcp open   ms-sql-s
8008/tcp open   http
8010/tcp open   xmpp
8081/tcp open   blackice-icecap
8082/tcp open   blackice-alerts
8084/tcp open   unknown
8085/tcp open   unknown
8087/tcp open   simplifymedia

Nmap done: 1 IP address (1 host up) scanned in 52.43 seconds
```

14.3 Workarounds/Solutions

You will need to close these or block them from being exposed on the Internet.

Top

CONFIDENTIAL

15. ASP.NET MVC version disclosure Low

Affected Url : phrms.cloudapp.net:8092

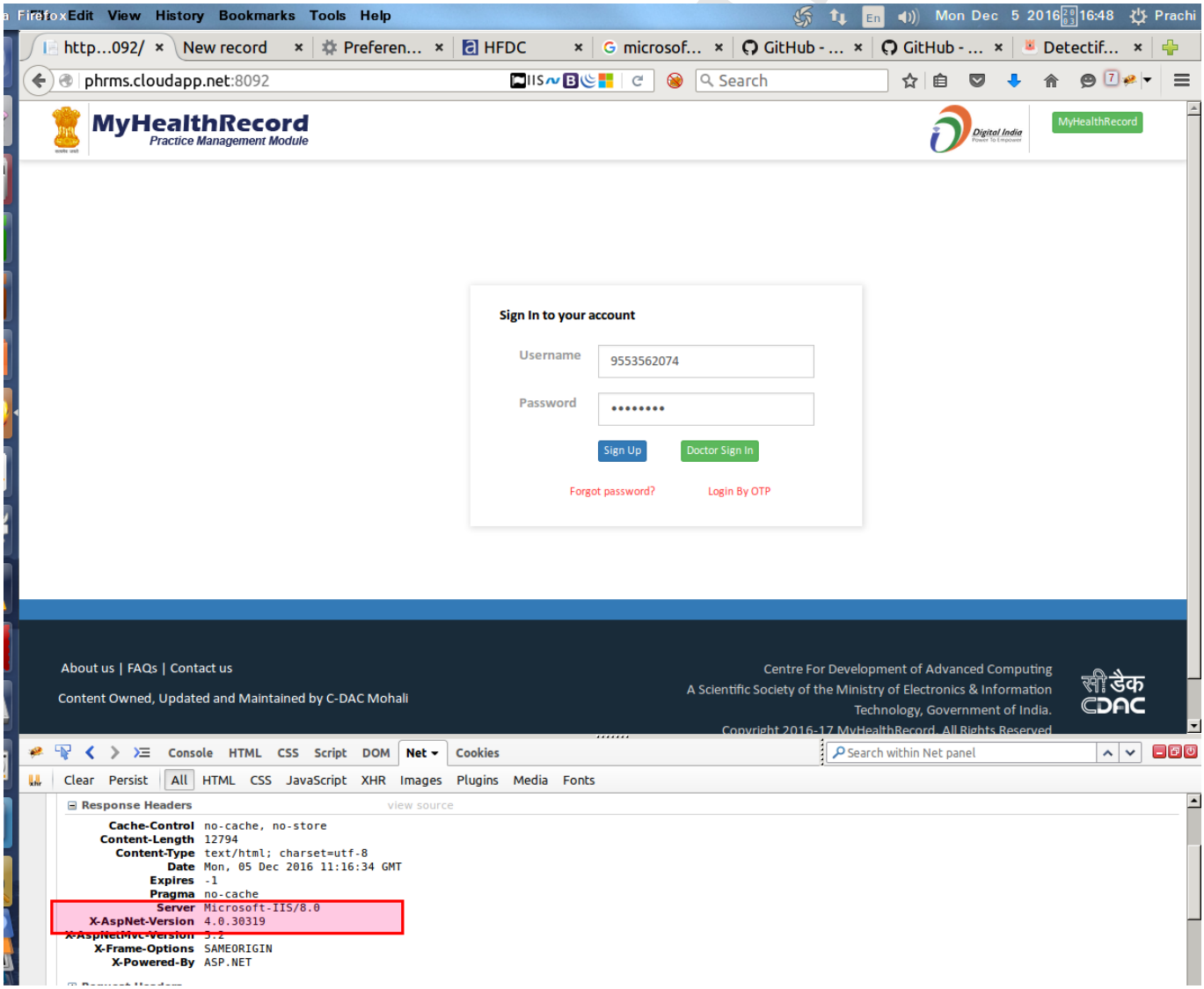
CWE : CWE-200

15.1 Description

The HTTP responses returned by this web application include a header named X-AspNetMvc-Version. The value of this header disclose the version of ASP.NET MVC in use. It is not necessary for production sites and should be disabled.

15.2 Proof of Concept

15.2.1 Step 1



15.3 Workarounds/Solutions

To remove the X-AspNetMvc-Version header add the following code in Global.asax, in the Application Start event:
MvcHandler.DisableMvcResponseHeader = true;

Top

CONFIDENTIAL

16. Password input field with autocomplete enabled — Low

Affected Url : phrms.cloudapp.net:8092

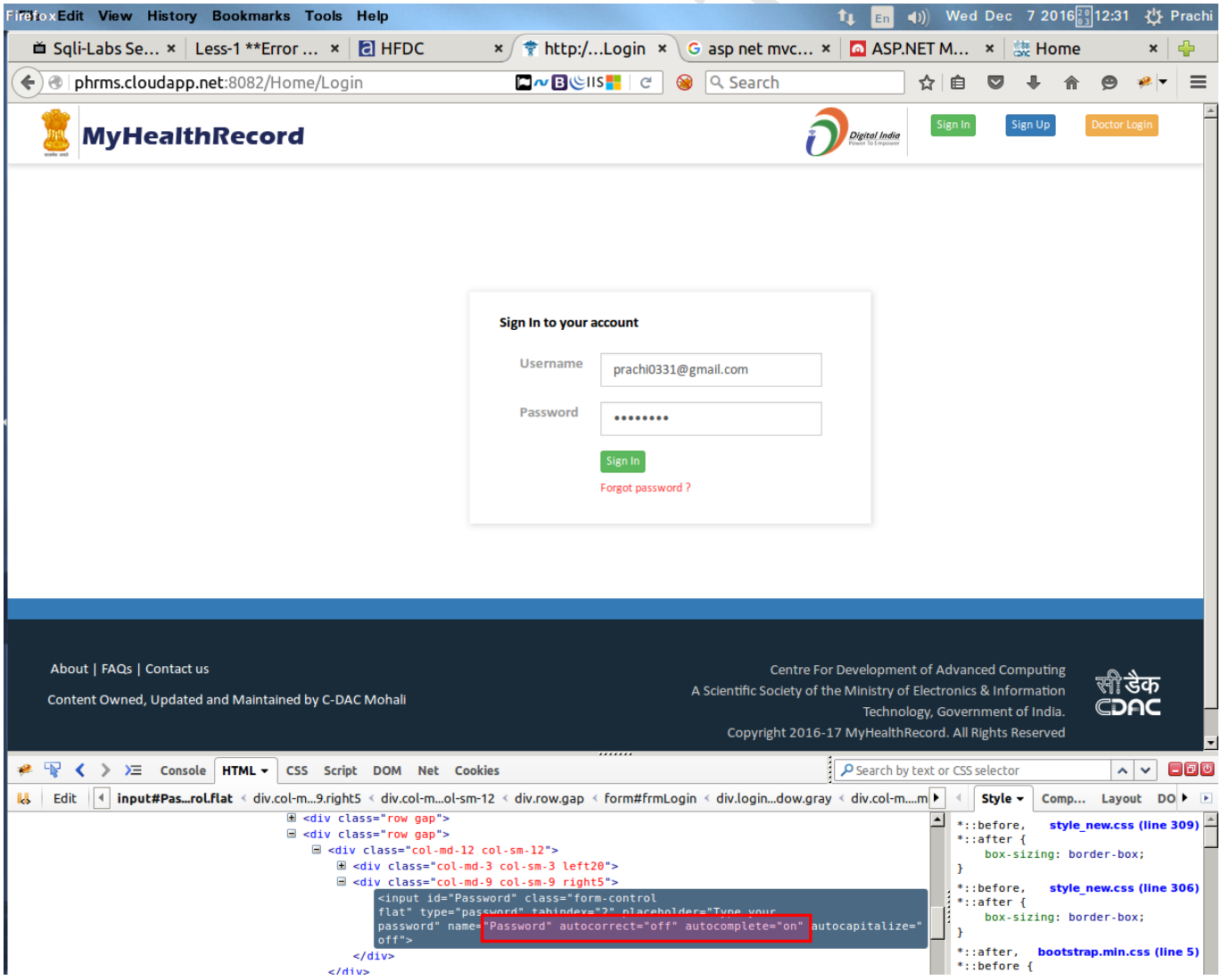
OWASP : BYPASSING AUTHENTICATION

16.1 Description

When a new name and password is entered in a form and the form is submitted, the browser asks if the password should be saved. Thereafter when the form is displayed, the name and password are filled in automatically or are completed as the name is entered. An attacker with local access could obtain the clear text password from the browser cache.

16.2 Proof of Concept

16.2.1 Step 1



16.3 Workarounds/Solutions

The password autocomplete should be disabled in sensitive applications. To disable autocomplete, you may use a code similar to: <INPUT TYPE=”password” AUTOCOMPLETE=”off”>

Top

CONFIDENTIAL

17. Web Server Version Disclosure — Informational

Affected Url : phrms.cloudapp.net:8092

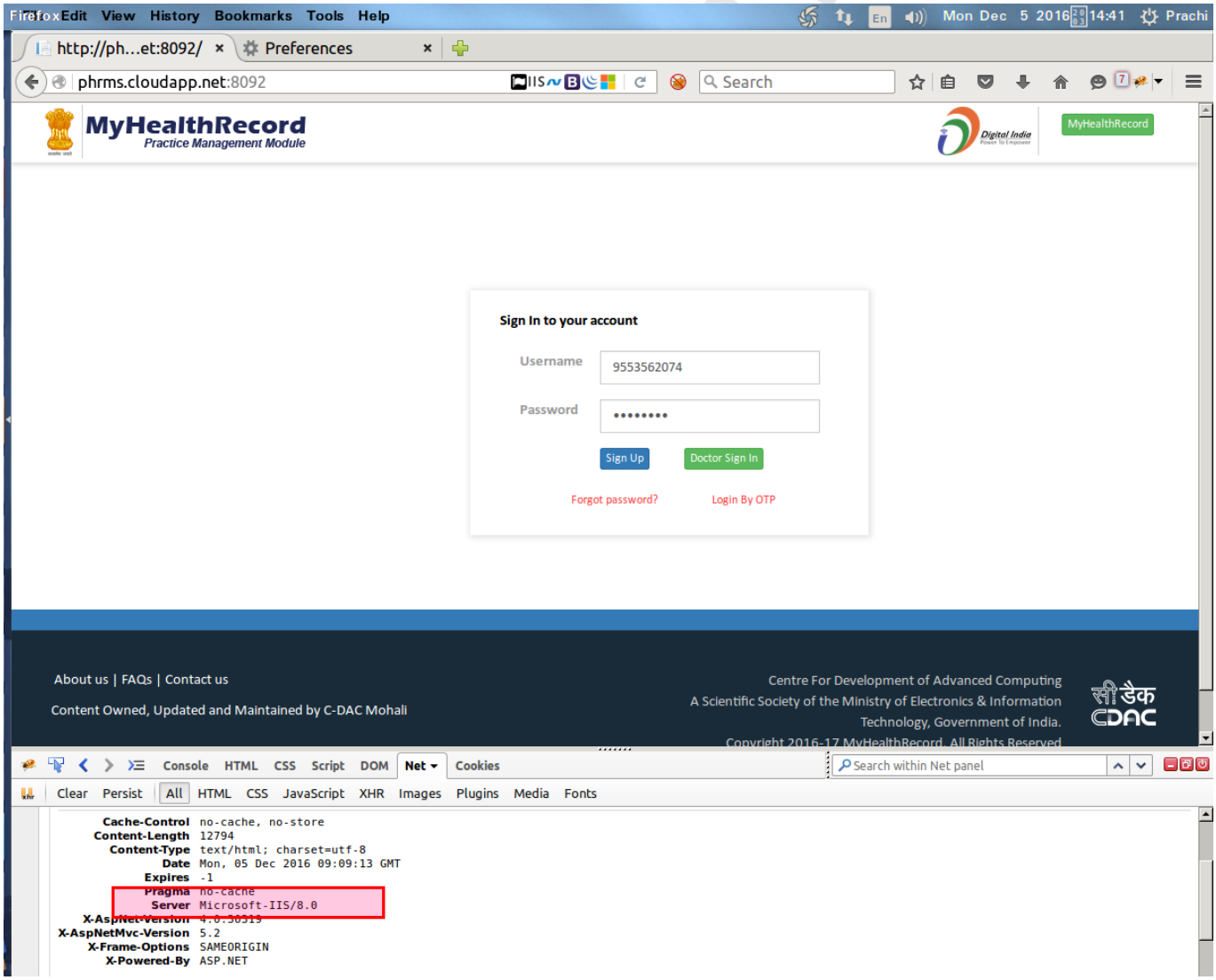
CWE : CWE-205

17.1 Description

This information can help an attacker gain a greater understanding of the systems in use and potentially develop further attacks targeted at the specific version of web server.

17.2 Proof of Concept

17.2.1 Step 1



17.3 Workarounds/Solutions

An attacker might use the disclosed information to harvest specific security vulnerabilities for the version identified.Configure your web server to prevent information leakage from the SERVER header of its HTTP response.

Top

CONFIDENTIAL

End of Report

CONFIDENTIAL