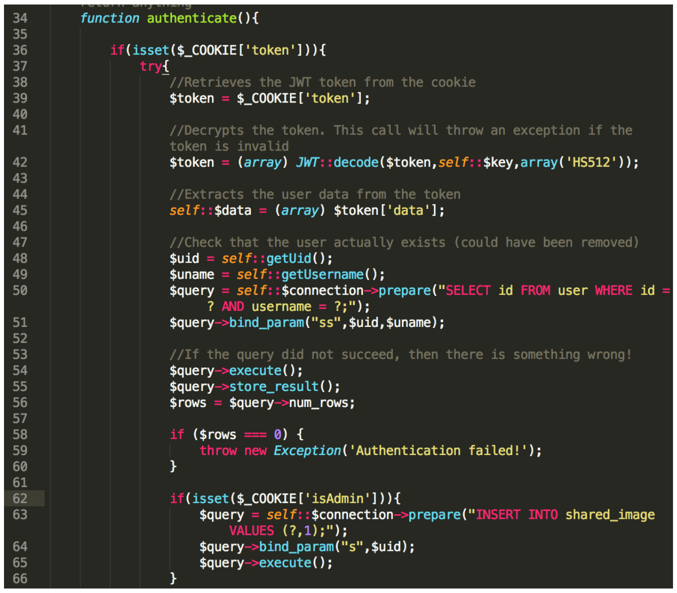
1 Backdoor Findings - Team E

1.1 Cookie Session Fixation

The session fixation attack exploits that one person may fixate (find or set) another person's session identifier (i.e. admin)[1]. The exploit was found in the cookie used for authentication on the server in the **authenticate()** method inside the **ssas.php** file. Firstly, a new user was registered. Then, the cookie session value for the new user was retrieved using the "EditThisCookie" plugin in Google Chrome (similar to tamper data or firebug in Firefox).

Based on this information, one can now spoof the admin cookie by adding a new cookie using "EditThisCookie" and change the cookie session value to be the same as the newly registered user from before - with the only exception being the cookie name that should be "isAdmin" as displayed in the source code on **line 62** below. That specific cookie name is required in order to actually authenticate as the admin with his cookie. As shown on the code below, the cookie session value is never validated so the only requirement is to use the same cookie name as the admin along with a token value from another user:



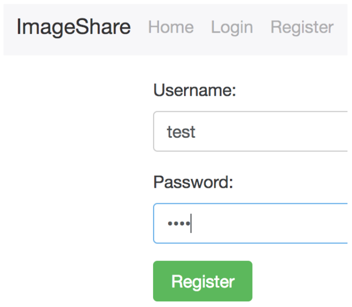
```
34      function authenticate(){
35
36          if(isset($_COOKIE['token'])){
37              try{
38                  //Retrieves the JWT token from the cookie
39                  $token = $_COOKIE['token'];
40
41                  //Decrypts the token. This call will throw an exception if the
                    token is invalid
42                  $token = (array) JWT::decode($token,self::$key,array('HS512'));
43
44                  //Extracts the user data from the token
45                  self::$data = (array) $token['data'];
46
47                  //Check that the user actually exists (could have been removed)
48                  $uid = self::getUid();
49                  $uname = self::getUsername();
50                  $query = self::$connection->prepare("SELECT id FROM user WHERE id =
                        ? AND username = ?;");
51                  $query->bind_param("ss",$uid,$uname);
52
53                  //If the query did not succeed, then there is something wrong!
54                  $query->execute();
55                  $query->store_result();
56                  $rows = $query->num_rows;
57
58                  if ($rows === 0) {
59                      throw new Exception('Authentication failed!');
60                  }
61
62                  if(isset($_COOKIE['isAdmin'])){
63                      $query = self::$connection->prepare("INSERT INTO shared_image
                            VALUES (?,1);");
64                      $query->bind_param("s",$uid);
65                      $query->execute();
66                  }
```

In practice, the steps used to get access to the admin account are shown on the next page.

[1]`https://en.wikipedia.org/wiki/Session_fixation` seen 24/4/17

Reproduction Steps:

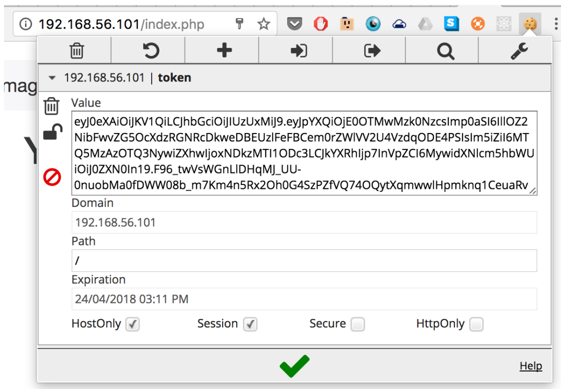
1) Register a new user:



Username: test

Password: ••••

2) Copy the cookie session value revealed in the "EditThisCookie" plugin:



192.168.56.101 | **token**

Value

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE0OTMwMzk0NzcsImp0aSI6IllOZ2
NibFwvZG5OcXdzRGNRcDkweDBEUzlFeFBCem0rZWlVV2U4VzdqODE4PSIsIm5iZiI6MT
Q5MzAzOTQ3NywiZXhwIjoxNDkzMTI1ODc3LCJkYXRhIjp7InVpZCI6MywidXNlcm5hbWU
iOiJ0ZXN0In19.F96_twVsWGnLlDHqMJ_UU-
0nuobMa0fDWW08b_m7Km4n5Rx2Oh0G4SzPZfVQ74OQytXqmwwlHpmknq1CeuaRv
```

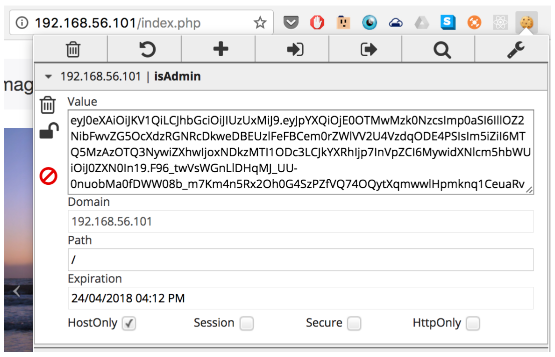
Domain: 192.168.56.101

Path: /

Expiration: 24/04/2018 03:11 PM

HostOnly ✓ Session ✓ Secure ☐ HttpOnly ☐

3) Create a new cookie using the same cookie session value, "isAdmin" as the cookie name and a valid expiration date:



192.168.56.101 | **isAdmin**

Value

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJpYXQiOjE0OTMwMzk0NzcsImp0aSI6IllOZ2
NibFwvZG5OcXdzRGNRcDkweDBEUzlFeFBCem0rZWlVV2U4VzdqODE4PSIsIm5iZiI6MT
Q5MzAzOTQ3NywiZXhwIjoxNDkzMTI1ODc3LCJkYXRhIjp7InVpZCI6MywidXNlcm5hbWU
iOiJ0ZXN0In19.F96_twVsWGnLlDHqMJ_UU-
0nuobMa0fDWW08b_m7Km4n5Rx2Oh0G4SzPZfVQ74OQytXqmwwlHpmknq1CeuaRv
```

Domain: 192.168.56.101

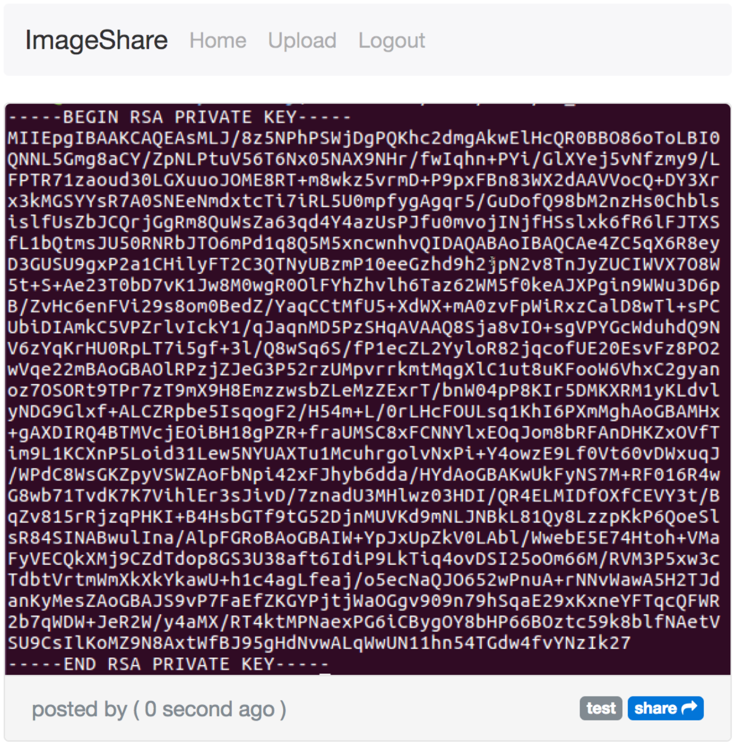
Path: /

Expiration: 24/04/2018 04:12 PM

HostOnly ✓ Session ☐ Secure ☐ HttpOnly ☐

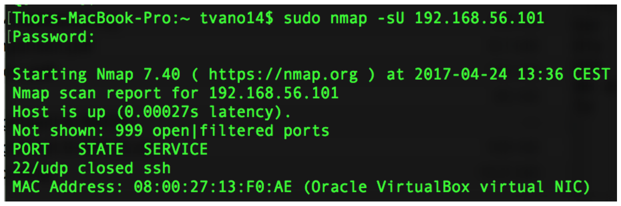
As a result, one can now login with the spoofed admin cookie session that is only validated against the cookie name, "isAdmin", and expiration data but not the actual cookie contents.

This reveals the content of the encrypted Public Private Encryption key used for SSH



```
-----BEGIN RSA PRIVATE KEY-----
MIIEpgIBAAKCAQEAsMLJ/8z5NPhPSWjDgPQKhc2dmgAkwElHcQR0BBO86oToLBI0
QNNL5Gmg8aCY/ZpNLPtuV56T6Nx05NAX9NHr/fwIqhn+PYi/GlXYej5vNfzmy9/L
FPTR71zaoud30LGXuuoJOME8RT+m8wkz5vrmD+P9pxFBn83WX2dAAVVocQ+DY3Xr
x3kMGSYYsR7A0SNEeNmdxtcTi7iRL5U0mpfygAgqr5/GuDofQ98bM2nzHs0Chbls
islfUsZbJCQrjGgRm8QuWsZa63qd4Y4azUsPJfu0mvojINjfHSslxk6fR6lFJTXS
fL1bQtmsJU50RNRbJTO6mPd1q8Q5M5xncwnhvQIDAQABAoIBAQCAe4ZC5qX6R8ey
D3GUSU9gxP2a1CHilyFT2C3QTNyUBzmP10eeGzhd9h2jpN2v8TnJyZUCIWVX7O8W
5t+S+Ae23T0bD7vK1Jw8M0wgR0OlFYhZhvlh6Taz62WM5f0keAJXPgin9WWu3D6p
B/ZvHc6enFVi29s8om0BedZ/YaqCCtMfU5+XdWX+mA0zvFpWiRxzCalD8wTl+sPC
UbiDIAmkC5VPZrlvIckY1/qJaqnMD5PzSHqAVAAQ8Sja8vIO+sgVPYGcWduhdQ9N
V6zYqKrHU0RpLT7i5gf+3l/Q8wSq6S/fP1ecZL2YyloR82jqcofUE20EsvFz8PO2
wVqe22mBAoGBAOlRPzjZJeG3P52rzUMpvrrkmtMqgXlC1ut8uKFooW6VhxC2gyan
oz7OSORt9TPr7zT9mX9H8EmzzwsbZLeMzZExrT/bnW04pP8KIr5DMKXRM1yKLdvl
yNDG9Glxf+ALCZRpbe5IsqogF2/H54m+L/0rLHcFOULsq1KhI6PXmMghAoGBAMHx
+gAXDIRQ4BTMVcjEOiBH18gPZR+fraUMSC8xFCNNYlxEOqJom8bRFAnDHKZxOVfT
im9L1KCXnP5Loid31Lew5NYUAXTu1McuhrgolvNxPi+Y4owzE9Lf0Vt60vDWxuqJ
/WPdC8WsGKZpyVSWZAoFbNpi42xFJhyb6dda/HYdAoGBAKwUkFyNS7M+RF016R4w
G8wb71TvdK7K7VihlEr3sJivD/7znadU3MHlwz03HDI/QR4ELMIDfOXfCEVY3t/B
qZv815rRjzqPHKI+B4HsbGTf9tG52DjnMUVKd9mNLJNBkL81Qy8LzzpKkP6QoeSl
sR84SINABwulIna/AlpFGRoBAoGBAIW+YpJxUpZkV0LAbl/WwebE5E74Htoh+VMa
FyVECQkXMj9CZdTdop8GS3U38aft6IdiP9LkTiq4ovDSI25oOm66M/RVM3P5xw3c
TdbtVrtmWmXkXkYkawU+h1c4agLfeaj/o5ecNaQJO652wPnuA+rNNvWawA5H2TJd
anKyMesZAoGBAJS9vP7FaEfZKGYPjtjWaOGgv909n79hSqaE29xKxneYFTqcQFWR
2b7qWDW+JeR2W/y4aMX/RT4ktMPNaexPG6iCBygOY8bHP66BOztc59k8blfNAetV
SU9CsIlKoMZ9N8AxtWfBJ95gHdNvwALqWwUN11hn54TGdw4fvYNzIk27
-----END RSA PRIVATE KEY-----
```

Further, the Nmap securty scanner reveals the following SSH service based on a UDP service scan:



```
Thors-MacBook-Pro:~ tvano14$ sudo nmap -sU 192.168.56.101
Password:

Starting Nmap 7.40 ( https://nmap.org ) at 2017-04-24 13:36 CEST
Nmap scan report for 192.168.56.101
Host is up (0.00027s latency).
Not shown: 999 open|filtered ports
PORT   STATE  SERVICE
22/udp closed ssh
MAC Address: 08:00:27:13:F0:AE (Oracle VirtualBox virtual NIC)
```

As a backdoor, to get root access on the virtual machine, the encrypted key on the admin account image may be used to retrieve the public private encryption key used to access the otherwise closed SSH service. The hardest part for the adversary is to manually write down the encrypted key to actually decrypt it. This security measure increased the resources (i.e. time) spent by the adversary to break into the system dramatically. When written down, the encrypted SSH key may be decrypted using an online key RSA decryptor. As a result, the open SSH service may now be used to log into the remote machine and execute commands as root.