Useful Hacking Series

Welcome to the Useful Hacking Series, in this series of 20 Episodes our world-renowned penetration tester/international speaker will share with you the top useful tips used during her security audits. The goal in this series is only to be a major supporting tool in everyday administrative tasks, but to also to encourage experimentation in creating your own solutions and having fun with technology using the tools we present. We call the series the "Hacking" for the good guys. Enjoy!

Episode 9: Sniffing and replaying ADFS claims with Fiddler!

Hi! My name is Michael Jankowski-Lorek, I am an expert at CQURE, solutions architect, data scientist and consultant also delivering trainings at CQURE Academy. I will be a guest co-writer in this Useful Hacking Series Episode. Today we will focus on ADFS authentication and how to protect your claims private data and prevent token replay attacks.

In this episode we are going look into the process authentication with ADFS. We will use “Fiddler” - free web debugging proxy tool to analyze network conversation between website to which user is authenticating and its web browser. This is a very useful tool for troubleshooting ADFS authentication problems and we will learn what the attacker using man-in-the-middle (MITM) attack can see and do and **how to prevent token replay attack**.

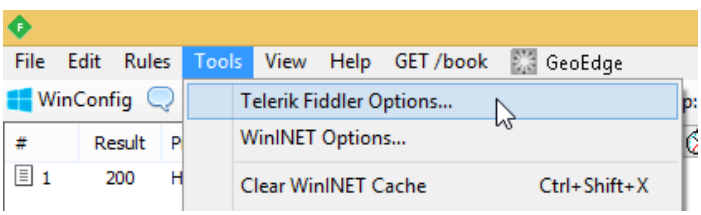
Basics

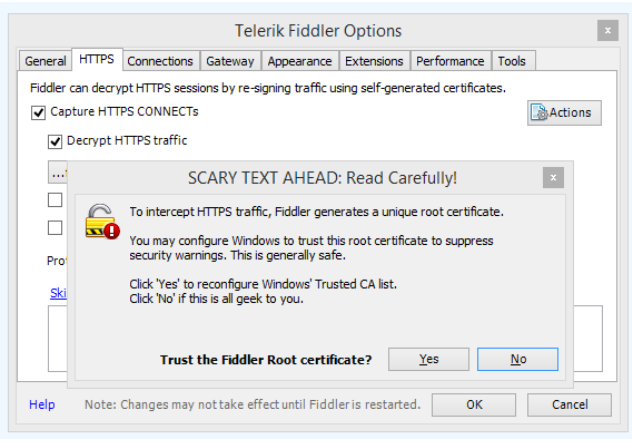
But first let’s start with the basics. What is ADFS and why to use it? Active Directory Federation Services is a standards-based service that allows the secure sharing of identity information between trusted (federated) partners. When a user wants to access resources from one of the federated partners (**RP - resource provider**) they are redirected to their own organization for authentication (**IdP – Identity Provider**) and only claims (signed statements about user) are returned to the resource provider. Main benefits of using ADFS: you never reveal your credentials to third parties, users can experience single sign-on, simplified (centralized) user account management, centralized federated partner management and many more.

Sniffing network conversation

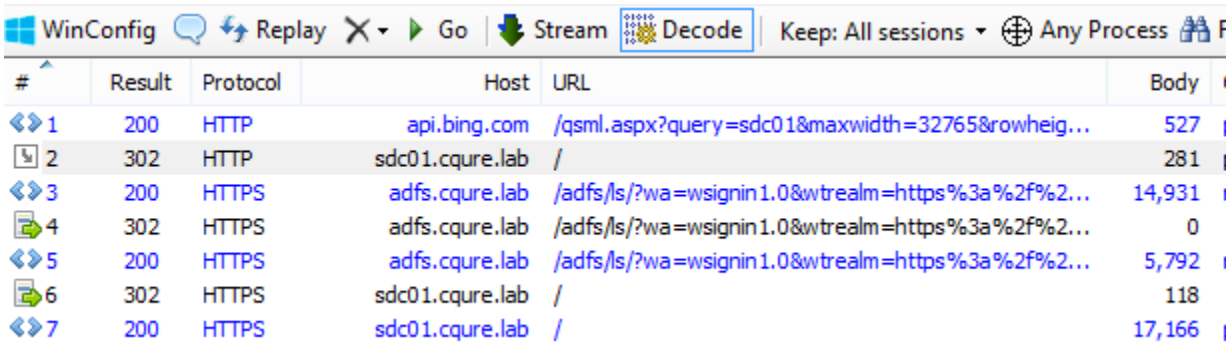
To analyze network traffic during federated authentication process we will use Fiddler which can be downloaded from Telerik’s web site http://www.telerik.com/fiddler. Additionally we will use Fiddler Inspector for Federation Messages to simplify analysis of SAML 2.0 and WS-Federation format messages. To add this to your Fiddler installation just simply download archive from http://identitymodel.codeplex.com/releases/view/52187 and copy content from *\bin\Debug* folder into *\Inspectors* folder in Fiddler installation path.

After installation start your Fiddler and go into **Tools → Tellerik Fiddler Options → HTTP** (tab) and check **Capture** and **Decrypt** HTTPS checkboxes. You will be prompted to install newly generated “Trusted Root Certificate” and from now on Fiddler will act as man-in-the-middle between your web browser and any other server. Screens below present configuration steps.



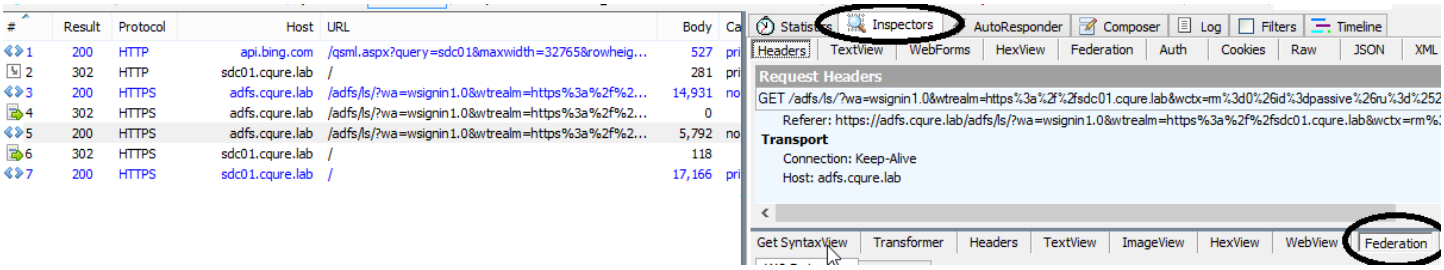


Without closing Fiddler, we'll start web browser and go to website that is federated Resource Provider in our test environment, which is https://sdc01.cqure.lab/ (**#2** in figure below). Resource requires authentication and user is redirected from sdc01.cqure.lab to Identity Provider configured in web.config file (https://adfs.cqure.lab) **#3**. After successful authentication, user's web browsers receives response **#5** with HTML web form that contains token signed by ADFS with all claims issued for RP that was requesting authentication. Web form is automatically posted and sent to sdc01.cqure.lab **#6** where token is verified and authorization is processed by RP based on claims issued by IdP.



#	Result	Protocol	Host	URL	Body
1	200	HTTP	api.bing.com	/qsml.aspx?query=sdc01&maxwidth=32765&rowheig...	527
2	302	HTTP	sdc01.cqure.lab	/	281
3	200	HTTPS	adfs.cqure.lab	/adfs/ls/?wa=wsignin1.0&wtrealm=https%3a%2f%2...	14,931
4	302	HTTPS	adfs.cqure.lab	/adfs/ls/?wa=wsignin1.0&wtrealm=https%3a%2f%2...	0
5	200	HTTPS	adfs.cqure.lab	/adfs/ls/?wa=wsignin1.0&wtrealm=https%3a%2f%2...	5,792
6	302	HTTPS	sdc01.cqure.lab	/	118
7	200	HTTPS	sdc01.cqure.lab	/	17,166

So let's see how the token looks like. We need to select **#5** on list of HTTP/S requests and on the right side of Fiddler chose Inspectors and Federation from lower tabs list.



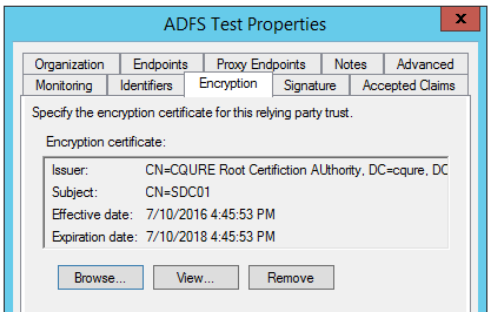
Listing below presents part of token sent in XML format containing **RequestSecurityTokenResponse**. Beside information about where and when this token was created we can find all claims issued during authentication process.

```
    -<saml:AudienceRestrictionCondition>
       <saml:Audience>https://sdc01.cqure.lab/</saml:Audience>
     </saml:AudienceRestrictionCondition>
   </saml:Conditions>
  -<saml:AttributeStatement>
    -<saml:Subject>
      -<saml:SubjectConfirmation>
          <saml:ConfirmationMethod>urn:oasis:names:tc:SAML:1.0:cm:bearer</saml:ConfirmationMethod>
       </saml:SubjectConfirmation>
     </saml:Subject>
    -<saml:Attribute
        AttributeName="name"AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims">
       <saml:AttributeValue>Administrator</saml:AttributeValue>
     </saml:Attribute>
    -<saml:Attribute
        AttributeName="upn"AttributeNamespace="http://schemas.xmlsoap.org/ws/2005/05/identity/claims">
       <saml:AttributeValue>Administrator@cqure.lab</saml:AttributeValue>
     </saml:Attribute>
    -<saml:Attribute AttributeName="telephonenumber"AttributeNamespace="http://cqure.lab/schemas/claims">
       <saml:AttributeValue>1234512345</saml:AttributeValue>
```

What is sent to RP is controlled by IdP administrators and is determined during configuration of federation with other parties. Sometimes RP might require to send claims containing sensitive information which we do not want to share with anyone else besides RP. In this example you can see private phone number. Of course all messages are sent through SSL tunnel but in case of MITM attack between RP and client web browser, hacker could receive private information.

Encrypting claims

To be sure that no one except RP can read claims sent with authentication token we need to encrypt their content. First, we need to have certificate installed on IIS hosting website. Content of the token will be encrypted with public key of that certificate so we need to publish it in RP's metadata or copy it to our ADFS server and chose it in **AD FS management console**.



Go to **AD FS console → Trust relationships → Relaying Party Trust → Your RP Name** (properties) **→ Encryption** (tab) and browse for public key of certificate (figure above).

Second step is to add information about certificate to the web.config file. Add **serviceCertificate** inside <microsoft.identityModel> <service> section.

Remember that Windows user account which application pool is using must have access to private key of certificate to decrypt claims, otherwise we will receive an error. We can grant this privileges using MMC console for computer's certificates. Locate certificate, then choose **All Tasks → Manage Private Key** and grant read permission to *IIS Apppool\<NameOfAppPool>* user.

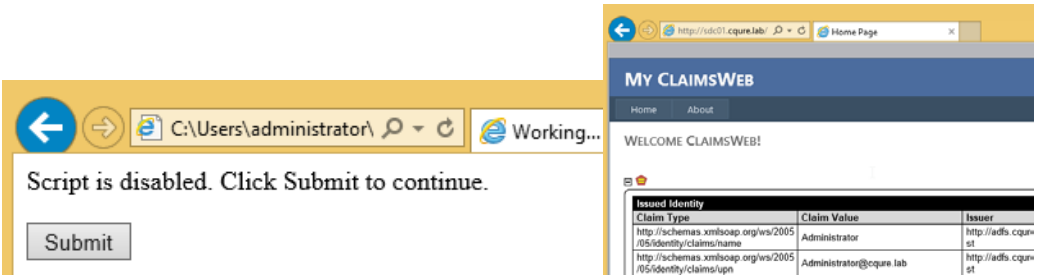
Now when we are sniffing token we see only encrypted data unless we have access to private key.

```
      </KeyInfo>
    -<xenc:CipherData>
        <xenc:CipherValue>4S1EvoUoMi1/Jmh4D6u6bTCoOFOfw/FcyoCEJiyqiim+S61CRi5snTZVB
      </xenc:CipherData>
    </xenc:EncryptedData>
  </t:RequestedSecurityToken>
  <t:TokenType>urn:oasis:names:tc:SAML:1.0:assertion</t:TokenType>
  <t:RequestType>http://schemas.xmlsoap.org/ws/2005/02/trust/Issue</t:RequestType>
  <t:KeyType>http://schemas.xmlsoap.org/ws/2005/05/identity/NoProofKey</t:KeyType>
</t:RequestSecurityTokenResponse>
```

Preventing token replay attack

As mentioned before token created by ADFS is sent to client's web browser in HTML Web Form which is than posted to RP website. By default this token can be used to authenticate at RP again if it's captured by MIMT attack. And now let's test this. Again we go to line **#5** in Fiddler but this time we choose TextView on lower tabs list and click View in Notepad in the bottom right corner. This will open new notepad instance with HTML code. We save this file as **replayToken.html**.

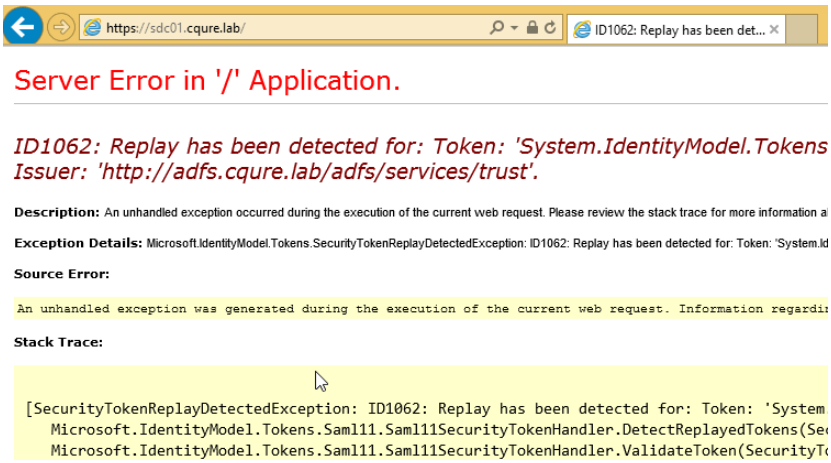
We need to remember to sign out from website and restart web browser so we are sure that we are no longer authenticated. Next we try to access the website again and it should redirect us to ADFS login page but this time instead of providing our credentials we simply open previously saved **replayToken.html**. Depending on settings of our web browser we will see HTML Web Form and button to send it or we will be automatically redirected to the website.



This attack will work until the token expiration time. To prevent this kind of attack we need to enable **Token Replay Detection** in our application.

To do this open web.config file and add **tokenReplayDetection** entry inside <microsoft.identityModel> <service> section. We can specify how long and how big should be the history of used tokens.

After saving changes we can test the replay token attack again. Remember that the first time after the change token will still work but the second attempt will generate an error **SecurityTokenReplayDetectedException** that should be handled by the application code.



I hope that this short article will help you to understand basics of ADFS authentication process and that you've learned how to use Fiddler to sniff tokens. This sill is very important for troubleshooting. Also now you know how to protect your claims from an unauthorized access and protect from token replay attack in your web application.

Contact us at info@cqure.us if you want to learn more about ADFS or setting your own test environment for this article.

Stay CQURE!

Michael Jankowski-Lorek (CQURE Academy)