Leveraging BIG-IP APM for seamless client NTLM Authentication

Michael Koyfman, 2014-22-07

Many customers express interest to use F5 Access Policy Manager for transparent seamless authentication for their users. There are a couple of leading use cases that drive that desired behavior:

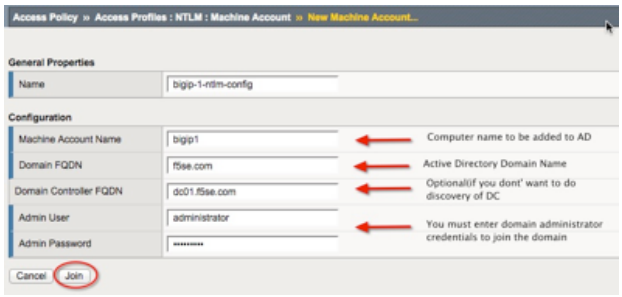
1. Providing silent seamless authentication to Windows-based applications such as Exchange or Sharepoint from the domain-joined machines. The premise is is that if the user is logged in to their domain-joined machine, no matter where they are, they should be able to perform seamless NTLM authentication to their applications such as Sharepoint based on the Windows Integrated Authentication settings.

2. Providing SAML Identity Provider services with APM. When users access SAML-enabled applications, they are asking for SAML assertions. Because APM can act as either native SAML 2.0 IDP or as a proxy to other SAML IDPs such as ADFS, for example, customer desire silent authentication to those IDP services from the domain-joined machines in order to seamlessly enable to SaaS applications such as Office 365, SalesForce.com, Google Apps, etc.

APM can perform three types of 401-based challenge authentication: Basic, NTLM, and Kerberos. Basic always requires user's intervention, but Kerberos and NTLM can enable users to seamlessly authenticate to the APM virtual server and allow it to either securely proxy connection to the backend application such as Sharepoint, leveraging Kerberos Constrained Delegation as the SSO mechanism, or acting as SAML IDP and issuing assertions to the SAML Service Providers based upon user identity extracted during NTLM authentication or Kerberos ticket.

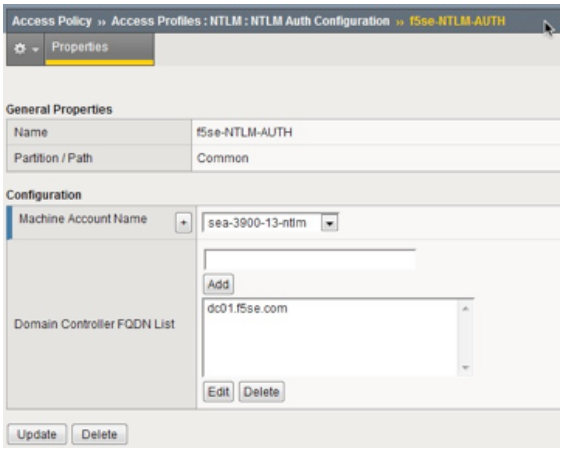
Today, we are going to examine the second use case on how to configure APM to perform client NTLM authentication and use it in the context of sending a SAML assertion to the Office 365 service. It is assumed below that user knows how to configure APM for standard forms-based authentication and also has at least one existing policy(although you can create a new one from the scratch). One of the easiest ways to test this is to deploy the Office 365 configuration using the iApp and the modify configuration to enable NTLM authentication. The steps below assume that you either have a working Office 365 configuration based on the iApp, or you have an equivalent policy that you can modify.

1. First, and foremost, we need to create an NTLM Machine Account object. Under Access Policy, go to Access Profiles->NTLM->Machine Account, and click on Create to join the BIG-IP to the domain and create unique computer object in Active Directory



   Keep in mind that you will need to create a unique account in Active Directory for your BIG-IP. In the example above, the account name is **bigip1**.

2. Create a "NTLM Auth Configuration" using the above machine account name. Under Access Policy, go to Access Profiles->NTLM->NTLM Auth Configuration and click on Create. Give the configuration the name, select the Machine Account Name value based on the object you created in Step 1, and add as many FQDNs for the AD domain controllers in your infrastructure



3. Now we need to create an iRule that will help us handle NTLM authentication to the BIG-IP properly. You need to modify the sec on cline of the RULE_INIT event to match the name of the NTLM Auth configuration you created in step 2. You will also need to replace all instances of

RULE_INIT event to match the name of the NTLM Auth configuration you created in step 2. You will also need to replace all instances of *appname* with a unique identifier. Go to Local Traffic->iRules->iRules List and click on Create. Give the iRule name of “ntlm-auth-iRule” and paste the iRule into the BIG-IP:

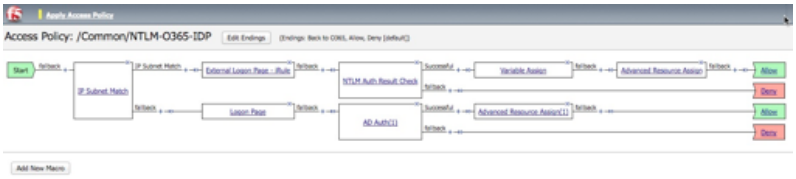
```
when RULE_INIT {
    set static::appname_ntlm_retries 2
    set static::appname_ntlm_config "/Common/appname_ntlm_config"
    set static::appname_access_log_prefix "01490000:7:"
    set static::appname_ntlm_on_demand_prfx "$static::appname_access_log_prefix \[NTLM-ON-DEMAND\]"
}

when ACCESS_SESSION_STARTED {
    ACCESS::session data set "session.ntlm.last.retries" 0
}

when HTTP_REQUEST {
    log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx Request: [HTTP::uri]"
    switch -glob -- [string tolower [HTTP::uri]] {
    "/ntlm/auth" {
        set sid [ACCESS::session sid]
        log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx sid: $sid"
        set referer [HTTP::header value Referer]
        log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx Referer: $referer"
        set x_session_id [ HTTP::header value X-Session-Id ]
        if { [ string length $x_session_id ] != 0 } {
            set sid $x_session_id
       }
        set retries [ACCESS::session data get -sid $sid "session.ntlm.last.retries"]
        log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx retries: $retries"
        set auth_result [ACCESS::session data get -sid $sid "session.ntlm.last.result"]
        log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx auth result: $auth_result"
        if { ($auth_result == 1) || ($retries == $static::appname_ntlm_retries) && ($auth_result != 1) } {
            ECA::disable
            log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx Redirect to: $referer"
            HTTP::redirect "$referer"
       }
        else {
            ECA::enable
            ECA::select select_ntlm:$static::appname_ntlm_config
       }
        unset x_session_id
        unset referer
    }
    default {
        ECA::disable
    }
    }
}

when CLIENT_ACCEPTED {
    set second_pass pass[IP::client_addr][TCP::client_port]

    # Check if this is the first or second time passing through this virtual
    if { [ table lookup $second_pass ] == "1" } {
        set wait_timeout 3000
        set wait_delay 100
        set wait_total 0
        set disable_ssl disablessl[IP::client_addr][TCP::client_port]
```

```
        # Wait for SERVER_CONNECTED event to complete
        while { [ table lookup $disable_ssl ] != 0
             && [ table lookup $disable_ssl ] != 1
             && $wait_total < $wait_timeout } {
            set wait_total [ expr "$wait_total + $wait_delay" ]
            after $wait_delay
        }
        unset wait_delay wait_timeout

        # Check table value set by SERVER_CONNECTED to disable ssl
        set disable_ssl_value [ table lookup $disable_ssl ]
        if { $disable_ssl_value == "1" } {
            set command "SSL::disable"
            eval $command
            unset command
        } elseif { $disable_ssl_value != 0 } {
            log -noname accesscontrol.local1.notice "$static::appname_ntlm_on_demand_prfx Error: SERVER_CONNECTED event not completed after $
        }
        table delete $disable_ssl
        table delete $second_pass
        unset disable_ssl wait_total
    } else {
        # This is the first time through this virtual. Set clientssl flag
        set client_ssl clientssl[IP::client_addr][TCP::client_port]
        if { [ catch { PROFILE::clientssl name } ] } {
            table add $client_ssl "0"
        } else {
            table add $client_ssl "1"
        }
        unset client_ssl
    }
    unset second_pass
}

when SERVER_CONNECTED {
    set client_ssl clientssl[IP::client_addr][TCP::client_port]
    set disable_ssl_value 0

    # Check clientssl flag set from CLIENT_ACCEPTED.
    if { [ table lookup $client_ssl ] == "1" } {
        if { [ catch { PROFILE::serverssl name } ] } {
            # Clientssl is present but serverssl is not. Disable clientssl
            set disable_ssl_value 1
        }
        table delete $client_ssl
    }
    set disable_ssl disablessl[IP::client_addr][TCP::client_port]
    table add $disable_ssl $disable_ssl_value
    unset disable_ssl

    unset client_ssl disable_ssl_value
}

when ECA_REQUEST_ALLOWED {
    log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx NTLM Auth succeed"
    ACCESS::session data set session.ntlm.last.username "[ECA::username]"
```

```
    ACCESS::session data set session.ntlm.last.domainname "[ECA::domainname]"
    ACCESS::session data set session.ntlm.last.machinename "[ECA::client_machine_name]"
    ACCESS::session data set session.ntlm.last.status "[ECA::status]"
    ACCESS::session data set session.ntlm.last.result 1
    ACCESS::disable
    HTTP::header insert X-Session-Id $sid
    log -noname accesscontrol.local1.debug  "$static::appname_ntlm_on_demand_prfx use virtual: [ virtual name ]"

    # Set flag for next CLIENT_ACCEPTED telling it that it is the second pass through virtual
    set second_pass pass[IP::client_addr][TCP::client_port]
    table add $second_pass "1"
    unset second_pass

    # Connect to itself in order to generate HTTP response
    use virtual [ virtual name ]
}

when ECA_REQUEST_DENIED {
    log -noname accesscontrol.local1.debug "$static::appname_ntlm_on_demand_prfx NTLM Auth succeed"
    if { [ACCESS::session data get session.ntlm.last.retries] != $static::appname_ntlm_retries } {
        incr retries
        ACCESS::session data set session.ntlm.last.retries $retries
    }
}
```

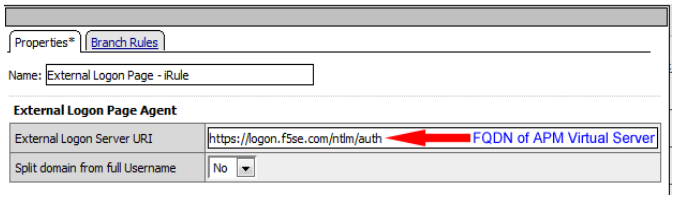
After creating this iRule, assign it to the APM Virtual Server.

4. Now you can create or modify your existing policy as below. Let's examine how the policy depicted below is structured. The assumption is that the policy is going to be used to authenticate both internal and external users. If the users are coming in from the internal corporate network, we want to steer them straight to the NTLM authentication, if not, we want to use forms-based login for to authenticate them. I've started the policy with IP Subnet Match action to steer clients from certain networks to the NTLM authentication. One the desired source networks are matched, we move on to an External Login Page object that will send user back to the APM virtual and request NTLM authentication.

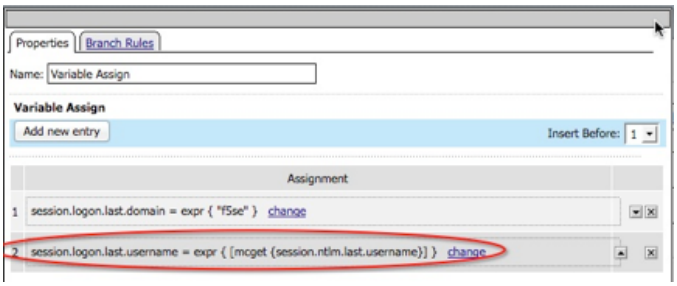


Let's examine how the policy depicted above is structured. The assumption is that the policy is going to be used to authenticate both internal and external users. If the users are coming in from the internal corporate network, we want to steer them straight to the NTLM authentication, if not, we want to use forms-based login for to authenticate them. I've started the policy with IP Subnet Match action to steer clients from certain networks to the NTLM authentication.

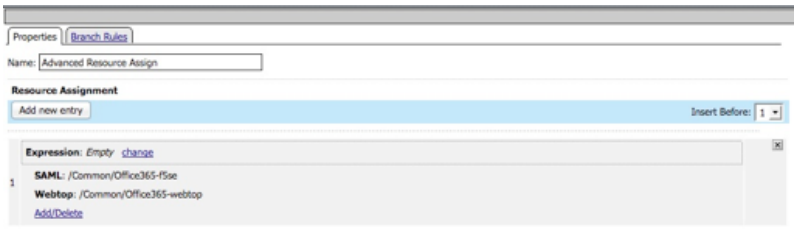
Once the desired source networks are matched, we create an External Login Page object that will send user back to the APM virtual and request NTLM authentication.



After sending the user to the “external login page”, which in fact is just a request to the same virtual server that is handled by the iRule that enables NTLM authentication between the client and BIG-IP, we need to check the status of the NTLM authentication, so we add the “NTLM Auth Result Check” action to see if the NTLM authentication was successful. If so, we need to populate the username session variable to enable APM to use it in session reporting/tracking, SAML assertion, SSO, etc.

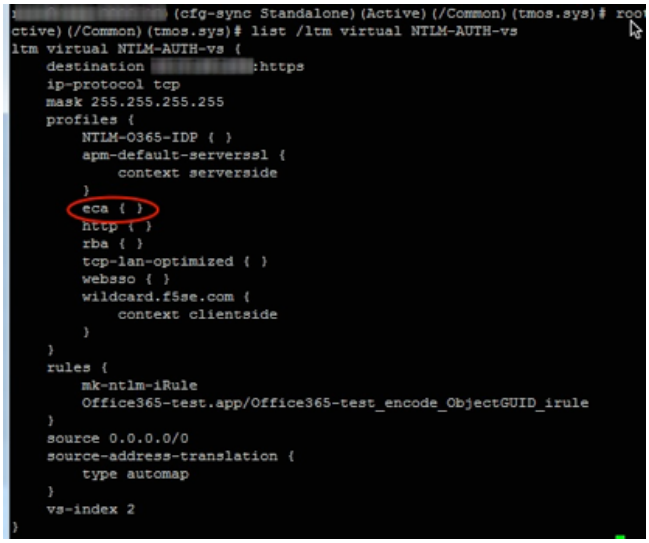


Now you can assign necessary resources to the user session. In this example, we are assigning APM to act as the IDP to Office 365.



After you finished creating or modifying the Access Policy, make sure it is assigned to the APM virtual.

5. Now we need to associate a ECA profile with the Virtual Server in order to enable NTLM functionality. This assignment needs to be performed via the command line. Establish an SSH connection in the box and enter TMSH and type the following commands, substituting the name of your virtual server for the highlighted portion
    a. /sys
    b. modify /ltm virtual NTLM-AUTH-vs profiles add { eca }
    c. save config
    d. list /ltm virtual NTLM-AUTH-vs
    e. Note the ‘eca’ profile associated with the virtual server



6. Next, we need to modify how the virtual server handles preservation of the original source port of the connection. This can be done either from the BIG-IP Administrative interface, or from the command line. Both examples are shown below.

A. Command Line Interface
    a. Using the same SSH session as established in Step 5, type the following commands substituting the name of your virtual server for the

highlighted portion:

i. modify /ltm virtual **NTLM-AUTH-vs** source-port preserve-strict

ii. save /sys config

B. BIG-IP Administrative Interface

a. From the main menu, go to *Local Traffic* > *Virtual Servers* > *Virtual Server List*.

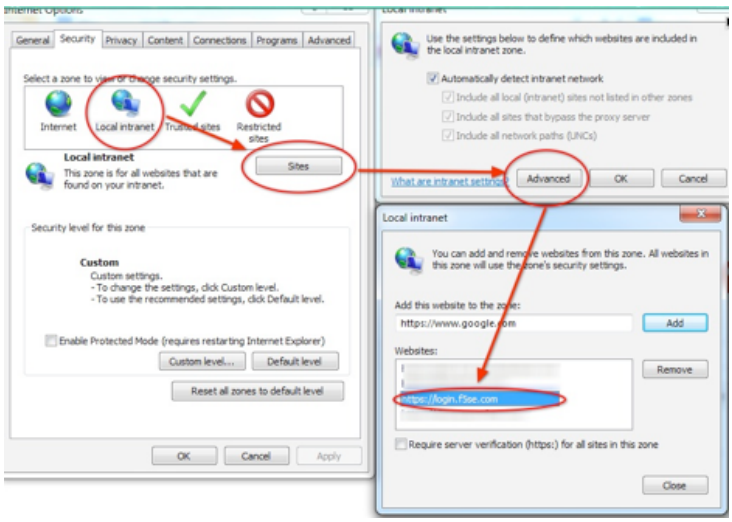
b. Click on the APM virtual server.

c. Under **Configuration**, select *Advanced*.

d. For **Source Port**, select *Preserve Strict*.

e. Click **Update**."

7. Last, but not least, you need ensure that the machine you're using to achieve the silent sign-on has the APM Virtual FQDN added to its Local Intranet zone as per the picture below.



Voila! You should be all set. Point your browser on the machine to the FQDN of the APM Virtual Server where you assigned the new policy and iRule, and you should be silently authenticated. If you are interested in performing SSO to applications such as Sharepoint, you will need to setup Kerberos SSO in order to perform single sign-on to the Sharepoint based on the NTLM authentication.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc. Corporate Headquarters info@f5.com	F5 Networks Asia-Pacific apacinfo@f5.com	F5 Networks Ltd. Europe/Middle-East/Africa emeainfo@f5.com	F5 Networks Japan K.K. f5j-info@f5.com