APM Session Invalidation Using ASM

Colin Walker, 2011-17-10

Introduction:

Whenever customers expose their internal resources on the Web using VPNs or SSL VPNS there is still some concern over what type of traffic comes through the connection. In order to assist with these concerns we can provide a combined SSLVPN solution with added Application Security using the APM and ASM modules. The guidance for configuring these two modules can be found at the link below:

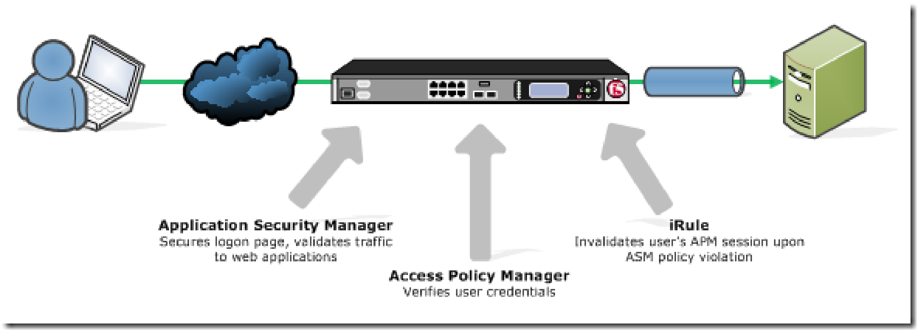
http://support.f5.com/kb/en-us/products/big-ip_ltm/manuals/product/big_ip_mod_interop_10_1_0.html?sr=16032121

However, once the customer has configured both APM and ASM there is still a residual concern, *what about the APM session?*

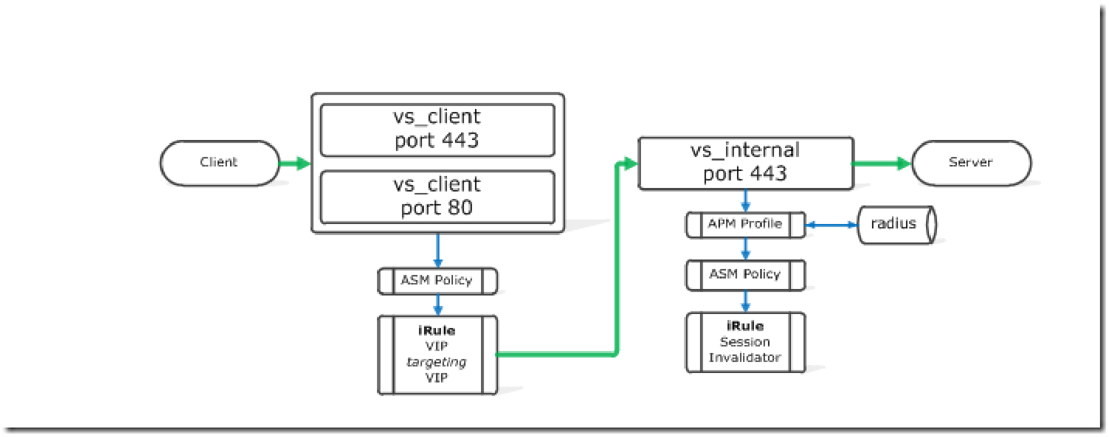
The following guide provides a means to configure an environment with both APM and ASM modules *by using an iRule to track the APM session and end it if an ASM violation occurs.*

With that in mind, let’s get started.

Overview:


Application Security Manager
Secures logon page, validates traffic to web applications
Access Policy Manager
Verifies user credentials
iRule
Invalidates user's APM session upon ASM policy violation

We will cover a scenario where the customer will have a Logon Page generated by APM and secured by ASM. This logon page will then authenticate against a backend system and allow access to the desired web application; the application will also be protected by ASM.



Logical Flow:

> 1. Client connects to the vs_client VIP
> 2. ASM policy applied and VTV redirects to vs_internal
> 3. Client lands on APM logon page and enter credentials
> 4. Credentials are verified against RADIUS server
> 5. Client session is secured by second ASM Policy
> 6. Client request LB'd to server

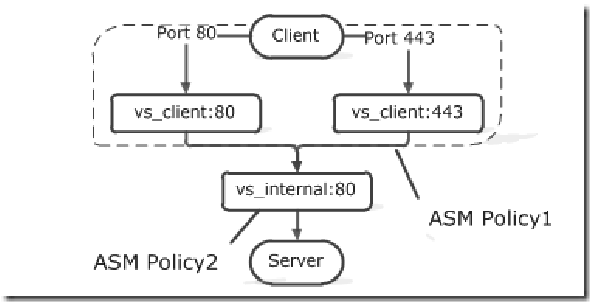
With Auto Last Hop and SNAT Automap the flow will follow the same path out.

ASM Policies:

In order to secure the Logon Page generated by APM we will create two ASM policies. If the two policies are the same the system will create two entries for each violation, this could complicate the troubleshooting and logging process.

> *First:* Policy built to secure Logon Page
>
> *Second:* Policy built to secure the customer application

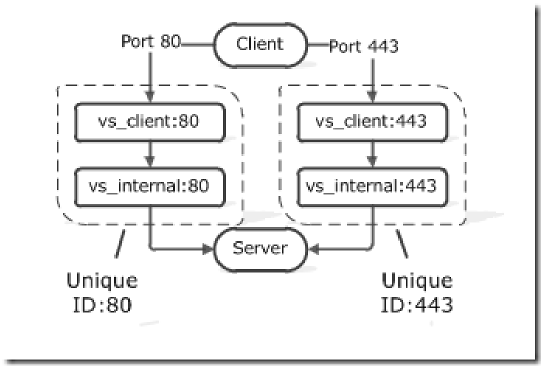
Virtuals:



The actual user deployment may vary however for our example we needed to cover both HTTPS and HTTP objects on the webpage. As such we created two client facing virtuals as shown above.

In order to maintain a single session id per user we have the two client facing virtuals pass traffic through vs_internal. This is done by creating a virtual on port 80 and redirects to 443. Once the traffic is on port 443 it will go through the flow described in the Flow section.

The second option available for deploying mixed environments is creating a client facing virtual and server facing virtual pair. This option is not viable when attempting to invalidate sessions with the iRule as we are unable to invalidate both sessions based on a single violation. A diagram of this configuration is below for clarification.



iRule:

The following iRule provides a means to invalidate sessions based on the rules configured in the ASM module

```
when ACCESS_ACL_ALLOWED {
    set mrhsession [HTTP::cookie value "LastMRH_Session"]

    if { [table lookup $mrhsession] == "violation" } {
        set user_logon [ACCESS::session data get "session.logon.last.username"]
        set sessionid [ACCESS::session data get "session.user.sessionid"]

        log local0.warn "ASM VIOLATION - Session: $sessionid, User: $user_logon"
        ACCESS::session remove
        table delete $mrhsession
    }
}

when ASM_REQUEST_VIOLATION {
    set mrhsession [HTTP::cookie value "LastMRH_Session"]

    if { $mrhsession != ""} {
        table set $mrhsession "violation"
        log local0.warn "ASM VIOLATION - MRHSession: $mrhsession"
    }
}
```

The iRule uses tables to maintain variables between scopes. By using a table the Events are able to share variables and act according to the value of those variables.

With the iRule above, whenever a violation occurs "violation" is inserted into the table matching the APM session id. If the particular session that triggers the next "ACCESS_ACL_ALLOWED" matches that session id and the value "violation" is present the session is removed.

Gotchas:

Keep in mind that this iRule heavily relies on the ASM policy actually blocking the violations. For example, without the iRule blocking a SQL insertion, the page would still display the information requested. If the ASM policy had caught and blocked this request it would not display the information and on subsequent requests the user would find that their session is no longer valid.

However, if the ASM policies are not properly configured before putting this iRule in place it will not work as intended.

Many thanks go to Josh Mendoza for the fine work on writing up this solution, and passing it along to be published. If anyone out there has any questions or comments, feel free to leave them here and we'll pass them along.