The goal: Keep the corporate users/devices off of the guest network.

Below I will describe three options that may help. All have some trade-offs but in many cases should work for you. For guest networks you have two basic authentication methods. Registered users that are required to enter a Username and a Password. Un-Registered users where the default prompts for the clients email address. A third method doesn’t depend on the guest authentication type. I will attempt to describe a configuration solution for each of these authentication cases.

The first method.

If your guest network requires a Username and Password and you are using the Aruba Mobility Controllers Internal Database for guest authentication than keeping corporate users off your guest network is fairly trivial. This method assumes you are using some backend server for your corporate clients such as a Radius. In this method we ADD the backend server to the Server Group list and configure it in position 1 and enable fail-through for the group of servers.

aaa server-group "guest-server-group"

allow-fail-through

auth-server my-corp-server

auth-server Internal

set role condition role value-of

!

To make this work properly a few assumptions are made. The first is we are not expecting the attribute ‘role’ from the ‘my-corp-server’. Second, we are expecting the default role in the Captive Portal Profile to be a dead-end or an isolation role for the corporate users as shown below. The behavior here is that a real registered guest would fail in the corporate server and because of that failure would fall through to the next server in the list and here they would pass when checked against the Internal server. The Internal server would pass over the role assigned in the Internal Database and would override the dead-end role assigned as the default role in the Captive Portal Profile.

aaa authentication captive-portal "my-guest-portal"

default-role dead-end

!

Now the trade-offs with this method. Each guest authentication would be seen by the backend server, in this case the Radius. Corporate users that get their hands on true guest credentials will still be able to access the guest network.

The second method.

This method requires the use of Machine Authentication. You may or may not want to enforce machine authentication but it is the trick that makes this method work. Machine Authentication implies that the backend server is setup to support Machine Authentication as well as the client device sending the machine ID. This only works for domain machines so that rules out any Apple and Unix devices since they do not support Machine Authentication along with user authentication.

The configuration for this is very straight forward. The behavior is not obvious however so I will try to describe the behavior.

First thing we need to do is to extent the Machine Authentication Cache timer to one week or longer. The default is 24 hours and experience says this isn’t long enough for this purpose. Next we need to decide if we really want to enforce a separate role for the machines or just let them on. Build those roles accordingly and I suggest a read of the Aruba OS User Guide for clarification on the different roles in use here. Build as required for your situation.

aaa authentication dot1x "my-dot1x-profile"

machine-authentication enable

machine-authentication machine-default-role "my-corp-machine-role"

machine-authentication cache-timeout 168

machine-authentication user-default-role "my-corp-user-role"

!

Now on the AAA Profile for the guests set it up for MAC Authentication. Authenticate against the internal database with a success role of dead-end or deny. Configure the Policies accordingly as well.

aaa profile "guest-aaa"

initial-role "guest-logon"

authentication-mac "MAC-Auth"

mac-default-role "guest-deny"

mac-server-group "guest-server-group"

!

When a domain device connects to the secure network their MAC address is put into the local database upon a success message from the backend server and kept there for the duration of the cache timer. You can actually look at the Internal Database to see devices that have been learned by using the command ‘show local-userdb’.

When a domain machine that has its MAC Address stored in the internal database connects to the guest network it will pass MAC Authentication and the resulting success puts that client into the dead-end role thereby preventing it from getting an IP and the portal page.

One of the advantages of using this method is that domain devices are automatically learned and placed into the internal database. Beware that devices can and will age out of the cache unless refreshed by a full machine authentication cycle. A more permanent solution is to collect the MAC Addresses of all the known devices and then populate the internal database ahead of time. Many customer feel that is too my work and this automatic method is good enough.

Some known weaknesses to this approach. It doesn’t work on non-domain machines like Apple products. Another downside of this approach is that it relies on learning what a valid machine is. That requires the device go through a full authentication cycle on the wireless so that it can be learned. Any device that is not known either through age-out or not seen at all will be able to access the guest network. Take an example of a device that is docked for several weeks at a time and then goes to wireless and the guest network first. That device would not be known in the database and hence gain access to the guest network.

However the expected behavior is a trained one. You should expect known/learned clients to ‘test’ the waters and connect to the guest network. They will fail more often than pass and eventually be trained to not bother anymore with it since it will fail for them more often than it will succeed.

The third method.

This method involves some knowledge about your corporate users and the services they attempt to reach on the corporate network in the background without them even knowing it. Such as the Domain Controller, file servers, exchange servers, and other similar common devices. To implement this method you simple write one or more policies in the guest logon role that look for this common type of traffic and if seen blacklist that client for some period of time.

The first step is to build a net destination containing the hosts that may be hit by the corporate user.

netdestination corp-background-traps

host 10.20.30.13

host 10.30.40.14

host 10.40.50.15

!

ip access-list session guest-logon

any alias corp-background-traps any deny blacklist

<your other policies here>

!

You could tighten up the above with specific service ports that SMB or NETBIOS might use but I thought I would just keep it fairly generic for this example.