Service Virtualization with iRules

Lori MacVittie, 2007-23-05

Service virtualization is a term that began appearing soon after the introduction of XML focused appliances into SOA land. It refers to the capability of an intermediary [proxy] to present a different URI endpoint to the client than is actually used by the service. Service virtualization is often used as a base form of security in virtualized service environments.

Example:

Internal: http://elmer.example.com/service/service1

External: http://elmer.example.com/85a1

This concept is not new; it's basically a form of URL rewriting and has long been available on application delivery controllers like BIG-IP. Proxy servers and XML intermediaries support this as a "checkbox feature", and it's useful for a number of reasons, which we'll get to shortly.

It should be no surprise that BIG-IP supports service virtualization as though it is second nature. That's probably because it is. There are a number of ways to implement service virtualization using iRules; this example uses externalization of a URI mapping to increase benefits beyond simple security.

The Concept:

Basically what we're doing is exchanging the SOAP endpoint for an obscure endpoint. By replacing the endpoint URI with an obscure URI, we're limiting invocation of services at that endpoint to only clients who know the obscure URI. This means forceful browsing type attacks can't be used, nor can attackers glean information about the actual service implementation platform from the URI, many of which follow specific patterns based on the vendor implementation.

The Implementation

1. Configure a data group

```
class service_endpoints {
  "/service/service1 85a1"
  "/service/service2 34xd"
  "/service/service3 44y7"
  "/service/service4 ag65"
  "/service/service5 997t"
}
```

2. When BIG-IP receives a request, it needs to translate the virtual URI to the real endpoint.

```
when HTTP_REQUEST {
   set newuri [findclass [string tolower [HTTP::uri]] $::service_endpoints " "]
   if { not ($newuri eq "" } {
      HTTP::uri $newuri
      pool myServicePool
  }
}
```

This technique works for *any* URI you need to rewrite. The use of a data group isn't a requirement, though without it you lose the benefits of loose coupling that comes from externalizing the data.

The biggest difference between service virtualization and pure URL rewriting is that it's likely that the client otained the endpoint when it requested a WSDL, which is not the obscure URI you want. To remedy this, you can intercept requests to the WSDL and replace the endpoint with the virtualized URI from the data group ***service_endpoints*** or you can modify the WSDL being served.

It is also possible to provide WSDL aggregation using iRules that presents the appropriate obscure endpoint, but that's a post for another day.

The Benefits

By mediating direct access to the endpoint you achieve a couple of benefits:

1. Agility

> The loose coupling between the endpoints and protected URIs allows easy migration to newer versions without requiring modification to clients. Because the the client accesses the service via an obscure URI, the actual service endpoint can be modified at any time without presenting problems to the client. This makes the process of migration or changes on the back-end transparent to the client.

2. Prevents information leakage

> URI endpoints for Web services platforms are often constructed in such a way as to make it obvious on which platform the service is running. For example, it's well known that almost every .NET hosted web service uses the extension **.asmx.** By obscuring this information using a virtualized URI, that knowledge is not easily obtainable by attackers.

Imbibing: Coffee

Technorati tags: application delivery, F5, BIG-IP, MacVittie, SOA, iRules