

Re: Each HTTP object being requested twice (401 then 200 responses)

Re: Each HTTP object being requested twice (401 then 200 responses)

Source:

<http://www.derkeiler.com/Newsgroups/microsoft.public.inetsrv.iis.security/2008-03/msg00006.html>

- *From:* benny.hauk@xxxxxxxxxx
 - *Date:* Tue, 4 Mar 2008 10:41:17 -0800 (PST)
-

On Mar 4, 8:56 am, David Wang <w3.4...@xxxxxxxxxx> wrote:

On Mar 3, 9:33 am, benny.h...@xxxxxxxxxx wrote:

On Mar 3, 11:09 am, David Wang <w3.4...@xxxxxxxxxx> wrote:

On Mar 3, 7:42 am, benny.h...@xxxxxxxxxx wrote:

I'm running a .NET 1.1 app on IIS6 with keep-alives and Session state enabled. Directory Security is set to only allow "Integrated Windows Authentication" and the web.config authentication setting is "windows". The web pages load fine and the client is correctly passing the current user's credentials behind the scenes as they should... almost.

When I run Fiddler on my IE7 browser I see that every object on the page is requested twice. The first one fails with a 401-Not Authorized because the request was made anonymously. When the browser

Re: Each HTTP object being requested twice (401 then 200 responses)

requests the same object a second time it uses kerberos; "Negotiate" with a HUGE token, which in my experience indicates that SSPI choose Kerberos and not NTLM, which is good since that's a requirement for our configuration (we have to delegate the credentials all the way through an n-tiered architecture).

The problem is that every single object is authenticating essentially from scratch so each page and each object within a page is requested twice and each second request has a HUGE payload in the header because of the kerberos overhead. The second request is always succeeding so user thinks everything is fine (overall performance of server stinks though and I think this is why – it's processing between 1–2 requests/sec).

As a contrast to that, when I goto the Advanced tab in my IE7 browser and uncheck "Enable Integrated Windows Authentication" (essentially forcing my browser to use NTLM instead of Kerberos) then the page and its objects are requested more correctly (the first page is rejected and tries against passing NTLM token then all other subsequent objects pass a session ID cookie and therefore succeed the first time. I would call this the fix/workaround only, like I said, my customer's requirements dictate that Kerberos be used instead of NTLM.

Anyone know how to make Kerberos behave more like NTLM in this way: IE requests page anonymously and gets a 401 response, then makes same

Re: Each HTTP object being requested twice (401 then 200 responses)

Re: Each HTTP object being requested twice (401 then 200 responses)

request authenticating with Kerberos and gets a 200 response. THEN, every other subsequent request tied to that session successfully gets objects back the first time with kerberos credentials being passed so they are only requested once and don't reauthenticate every single request (which creates overhead in communicating with the domain controller, etc, etc, etc).

One thing I noticed is that when I force NTLM the Cookie is of this form:

Cookie:
ASPSESSIONIDCCADBTCD=DKHBJACBEAIGBBJAKAJBKD

And when Kerberos is used the Cookie is of this form:

Cookie:
ASP.NET_SessionId=yuwpra55mlvefl45srf5mbed

Ring any bells?
Benny
benny period hawk – gmail

<http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Librar...>

Due to the nature of Kerberos, you will *never* be able to avoid that huge payload on every request. You can minimize the size of that payload by minimizing the group membership of the user involved, but

Re: Each HTTP object being requested twice (401 then 200 responses)

Re: Each HTTP object being requested twice (401 then 200 responses)

the payload (i.e. ticket) has to be present on every request because that is the authentication proof. Alternatively, NTLM uses the TCP connection as the authentication proof. Inherent in all designs, each mechanism of maintaining authentication proof has its benefits and drawbacks.

Now, that ticket is supposed to be cached by the client and automatically sent on the requests without re-authenticating, so you have to look at the client as to why it is reauthenticating.

FYI: You cannot make Kerberos behave more like NTLM. I recommend you read and understand how existing authentication protocols work because when you do, you will see the problems with what you request. Please explain how you plan to maintain an authenticated Kerberos "Session" that is secured against spoofs, replay attacks, and DoS.

//David<http://w3-4u.blogspot.com><http://blogs.msdn.com/David.Wang>
//– Hide quoted text –

– Show quoted text –

This is encouraging (I think). What I hear you saying is that to use Kerberos, there's no way around every single page being requested twice (once with a 401 response and the second time with a 200 response after passing the Kerberos token). I really thought once a session was authenticated that a simple session token would suffice (then if you wanted to protect against spoofs, replay attacks, etc you would need to also incorporate ssl certificates to encrypt the data). If you can confirm that I'm totally off base here and what we are seeing with the double requests is "by design" then I'll relay that on to the customer and move on (poor performance or not).

Re: Each HTTP object being requested twice (401 then 200 responses)

Re: Each HTTP object being requested twice (401 then 200 responses)

I also hear you saying that the kerberos tokens being passed aren't really being "regenerated" for each request but that they are essentially cached for the duration of the session. If that's the case then I'm not sure this is the cause of the poor performance we are seeing in the first place (I sort of assumed that when every page was "forgetting" that the session requires "NTLM,Negotiate" that it was reauthenticating the user from scratch.

I'll do some reading of the link you sent.

Thanks again David.– Hide quoted text –

– Show quoted text –

Let me clarify what I am saying...

Under Kerberos, you have to send that large kerberos token around in the authentication header of every request that is "authenticated" because that token is the proof of identity.

Since that large token in request header has performance and other implications, the signal for clients to start sending the token for authentication is the initial 401 response with WWW-Authenticate: Negotiate header. So, the *first* request for resource that requires Kerberos authentication requires two requests (first 401, then 200), but all subsequent requests, assuming client sends the Kerberos authentication token, should only take one request. It's just like Basic Authentication at this point, except the Kerberos token is much more versatile and secure.

Regarding your statement "I really thought once a session was authenticated that a simple session token would suffice" — if that is your view/requirement of authentication, then all you need is Basic over SSL.

What Kerberos brings to the table is Delegation. It has the per-request behavior of Basic with superior security over NTLM, and it supports Delegation.

Regarding your statement that SSL encryption protects against spoofs, replay attacks, etc — that is not the complete way to look at things. Consider the following:

Re: Each HTTP object being requested twice (401 then 200 responses)

Re: Each HTTP object being requested twice (401 then 200 responses)

1. If you use Basic over SSL, the endpoints are still vulnerable to spoof, replay attacks, etc because SSL only covers transport between endpoints and Basic Auth is still vulnerable at the Endpoints.
2. If you use Kerberos, you do not need SSL or anything to secure the endpoints and you are still protected (within the time limit of the token's freshness window) the entire from generation of the token by the user to the validation of the token by the service

Thus, SSL does not guarantee security, and lack of SSL does not mean insecurity.

Kerberos tokens should not be regenerated for every request. However, that may not be the case in your situation — you have not given any data to support nor deny the claim. You have to monitor the network traffic to determine what is going on.

I think something else is going on in your situation that is related to AuthPersistence

//David <http://w3-4u.blogspot.com> <http://blogs.msdn.com/David.Wang>
/-- Hide quoted text --

-- Show quoted text --

Well, I think I've found what I was looking for here:

<http://support.microsoft.com/kb/917557>

The new registry entry "EnableKerbAuthPersist" seems to accomplish the same thing for Kerberos as setting AuthPersistSingleRequest=False in the metabase does for NTLM (persist the token used).

I won't be able to restart the web services until after hours but this sounds like it addresses removing most of the 401/200 response pairs exactly.

(I ran across the mention of the "EnableKerbAuthPersist" in a posting of yours from last May – so thanks are in order!)

The size of the kerberos tokens is amazing (interesting how it's related to how many groups that account is a member of) but I don't think network throughput is a large concern at the moment.

Benny

ref: <http://www.webservertalk.com/archive122-2007-5-1895997.html>

Re: Each HTTP object being requested twice (401 then 200 responses)