

# NTLM HTTP Authentication is insecure by design – a new writeup by Amit Klein

*Source:* <http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2005-07/0274.html>

---

*From:* Amit Klein (AKsecurity) ([aksecurity\\_at\\_hotpop.com](mailto:aksecurity_at_hotpop.com))

*Date:* 07/18/05

Date: Mon, 18 Jul 2005 19:40:32 +0200  
To: bugTraq <[bugtraq@securityfocus.com](mailto:bugtraq@securityfocus.com)>

NTLM HTTP Authentication  
(and possibly other connection-oriented  
HTTP authentication and authorization protocols)  
is insecure by design

Or

NTLM Authentication and HTTP proxies  
don't mix

Amit Klein, July 2005

## Introduction

=====

In "Meanwhile on the other side of the webserver" (<http://www.securityfocus.com/archive/1/401866>) I surveyed some possible attacks against a scenario wherein a proxy server is positioned in front of a web server, and that proxy server shares a single TCP connection to the server among several clients. In that write-up, I mentioned several problems related to HTTP Request Smuggling (<http://www.watchfire.com/resources/HTTP-Request-Smuggling.pdf>) and HTTP Response Splitting ([http://www.sanctuminc.com/pdf/WhitePaper\\_HTTPResponse.pdf](http://www.sanctuminc.com/pdf/WhitePaper_HTTPResponse.pdf)). These are attacks that make use of non-RFC HTTP requests (HTTP Request Smuggling) or inject unexpected data (CRLF) through the application into the HTTP response stream (HTTP Response Splitting). In contrast, this write-up discusses a completely different problem, one which is inherent to the situation of a connection-oriented authentication/authorization protocol (e.g. NTLM authentication) used with a proxy server that shares TCP connections among several clients. Exploiting this vulnerability can be performed with 100% RFC compliant HTTP requests, and without attacking the application (i.e. without sending malicious data to the application).

## Theory

=====

In connection oriented security, the authentication is associated with the TCP connection, rather than to the individual HTTP requests it transports. As a result, a proxy server that shares a TCP connection to the server among 2 clients may jeopardize the security of the web application by sending a first request (or a set of requests) with authentication/authorization credentials from the first client, followed by a request with no credentials from the second client, and have the web server associate the privileges of the first request with the second request.

NTLM authentication is an example to such connection-oriented security scheme.

From <http://curl.haxx.se/rfc/ntlm.html#ntlmHttpAuthentication> (lacking official Microsoft specification, this resource is one of the most comprehensive descriptions of NTLM authentication):

This [HTTP NTLM authentication] scheme differs from most "normal" HTTP authentication mechanisms, in that subsequent requests over the authenticated connection are not themselves authenticated; NTLM is connection-oriented, rather than request-oriented. So a second request for "/index.html" would not carry any authentication information, and the server would request none.

This attack is possible because:

1. Proxy servers share the same TCP connection to the server, among several clients. This enables several attacks (on top of the one described here), as discussed in "Meanwhile, on the other side of the web server".
2. Connection-oriented security is an insecure concept because there's no guarantee in the HTTP RFC that a single connection will be used by a single entity. As can be seen, this simply doesn't hold. Note that SSL is not connection-oriented security since each request is encrypted with a secret, shared key, making this protocol implicitly request-oriented.

## Results

=====

I tested this security issue with Microsoft IIS/6.0 (as the web server that requires NTLM authentication – "Integrated Windows Authentication" in Microsoft's IIS GUI terminology) and Sun Microsystems Sun Java System Web Proxy 4 (as the proxy server that shares TCP connections to the same server).

There are some tricky points in making this attack work:

## SecurityFocus Bugtraq: NTLM HTTP Authentication is insecure by design – a new writeup by Amit Klein

1. Microsoft IE 6.0 refuses to conduct NTLM authentication when it is configured to use a forward proxy. Therefore, the setup used was with the Sun Proxy as a reverse proxy.

2. Microsoft IIS/6.0 does not induce the authentication level of a request to the whole connection, if the HTTP request contains the Via header. The Sun Proxy server sends this header by default (is there a way to turn this off?), and so, in order to strip it off, an Apache 2.0.54 reverse proxy server (with ProxyVia Block directive) was introduced between the Sun Proxy server and the IIS server.

After these tweaks, both IE 6.0 and Mozilla 1.4 were used to demonstrate the attack:

In the first step, a browser was used to authenticate to the IIS/6.0 (through the Sun Proxy and the Apache proxy). The authentication was done in NTLM. Since the Apache proxy removed the Via header, the IIS/6.0 induced the authentication credentials on the whole TCP connection.

In the second step, a different client was used to access a restricted resource on the IIS/6.0 through the Sun proxy (and the Apache proxy). The Sun Proxy used the same TCP connection to the Apache as it used for the first request, and likewise, the Apache used the same connection to the IIS/6.0 as it used for the first request, and therefore the credentials of the first request were successfully induced onto the second request, although it arrived from a different client on a different TCP connection (from the client to the Sun Proxy).

### Scope of the attack

=====

\*) Not all proxy servers honor NTLM authentication. Squid, for one, deliberately doesn't support NTLM (<http://www.squid-cache.org/Doc/FAQ/FAQ-11.html#ss11.14>). Indeed, Squid seems to strip off the WWW-Authenticate header if it contains NTLM or Negotiate, thereby effectively disabling NTLM authentication between the client and the web server. But as mentioned above, there are some proxy servers that do support NTLM authentication, such as Sun Proxy 4.

\*) Not all proxy servers share TCP connection to the server. Many do, some don't (e.g. Apache 2.0 mod\_proxy).

\*) If IE is to be tricked, then it mustn't be configured with a forward proxy server. That means that the attack is effective for IE (only) with transparent proxy servers (such as ones used by many ISPs), and reverse proxy servers (as demonstrated above). The Mozilla browser has no such inhibitions, and therefore, a Mozilla shop (e.g. some universities and open source organizations) may be

more vulnerable.

\*) The web server (IIS/6.0) must receive a Via-less request. The Microsoft implementation assumes that the Via header is always sent by a proxy server, and this is indeed mandated by the HTTP/1.1 RFC 2616 (<http://www.ietf.org/rfc/rfc2616.txt>), section 14.45:

The Via general-header field MUST be used by gateways and proxies to indicate the intermediate protocols and recipients between the user agent and the server on requests [...]

However, it seems that not all servers adhere to this standard. For example, Apache 2.0.54 mod\_proxy does not generate a Via header by default (see the ProxyVia directive –

[http://httpd.apache.org/docs-2.0/mod/mod\\_proxy.html#proxyvia](http://httpd.apache.org/docs-2.0/mod/mod_proxy.html#proxyvia), yet the default httpd.conf file contains a commented-out "ProxyVia On" directive, so it's possible that many Apache proxy deployments do send the Via header). That isn't to say that Apache 2.0.54 mod\_proxy facilitates this attack – as mentioned above, it does not, because it does not share the connection to the server among several clients. Anyway, there are many "anonymous" proxy servers in the Internet, which deliberately do not send the Via header, ironically with the intention to increase the privacy of their users. And there are many other devices and configurations that may remove the Via header if it exists (in the above example, I introduced the Apache proxy server to do just that).

\*) Last but not least – NTLM authentication should be used, and over HTTP (not over HTTPS). This is the default configuration of Microsoft Outlook Web Access 2000/2003.

#### Recommendations

=====

\*) Proxy vendors – do not to share TCP connections to the server among several clients. Yes, it improves performance, but it's also insecure and enables/aids 3 different attacks (the one described here, HTTP Request Smuggling and HTTP Response Splitting). Also, comply to the RFC and send the HTTP Via request header by default (Apache Group – please take note).

\*) Designers of protocols past, present and future – do not rely on TCP connection being used by a single logical entity. As a special case, NTLM should be withdrawn or redesigned (OK, this won't happen...). Also, do not rely on the Via header (or any other header) to indicate that the client is a proxy server. Design the protocol such that it will be indifferent to whether the client is a proxy server or a browser.

\*) Site owners – abandon NTLM authentication in favor of other authentication/authorization options (e.g. HTTP digest authentication – see RFC 2617 – <http://www.ietf.org/rfc/rfc2617.txt>).

Alternatively, use NTLM over HTTPS (SSL) to avoid this vulnerability, but make sure that the SSL is terminated on the web server, not some SSL accelerator (which may in itself facilitate the attack, e.g. if it shares a TCP connection to the server among several clients).

Another alternative is to configure the web server not to use persistent HTTP connections for resources that are protected by NTLM authentication.

\*) Proxy owners – in order to protect your clients and your clients' privacy: do not turn off generating the "Via" HTTP request header by the proxy server. True, it indicates that the request comes from a proxy server, but in the case of NTLM authentication, it increases the likelihood of the client not to be subject to the attack described here. If possible, turn off TCP connection sharing in your proxy server. If none of this is possible, consider actively disrupting NTLM authentication, in order to force your clients to use other (hopefully more secure) authentication methods.

A note about detection/prevention

=====

Since the attacker's request is practically identical to the request sent by the authenticated user, it's quite a problem for an external product (such as IDS/IPS/WAF) to detect this attack.

Of course, if the IDS/IPS/WAF is between the web-server and the proxy, it stands very little chance to detect that something's wrong, since the attacker's request is practically identical to the valid user's requests. However, it can block the attack simply by (gracefully, if possible) closing the TCP connection after a successful response (i.e. not 401) for a request containing NTLM authentication.

If the proxy server is on site, and the IDS/IPS/WAF is in front of it, then protection becomes harder – the IDS/IPS/WAF would have to replace the NTLM authentication of the server with its own, and practically replicate the logic from the web-server to itself, in order to ensure that a request without credentials is made only to a resource which is public.

It's also not too trivial to automatically scan for this kind of vulnerability. A scanner would have to be positioned in front of the proxy server (which may be away from the site), and would have to simulate the attack using two TCP connections.

A note about basic authentication in IIS/5.0

=====

If memory serves, and peculiarly enough, awhile ago Ronen Heled, Chaim Linhart and me bumped into an implementation quirk of IIS/5.0 wherein HTTP basic authentication seems to be also connection

## SecurityFocus Bugtraq: NTLM HTTP Authentication is insecure by design – a new writeup by Amit Klein

oriented, that is, if the TCP connection had already transmitted an HTTP request with valid Authorization header, the credentials are used for the next requests (on this TCP connection) even if these do not contain the Authorization header. Here too, the presence of the Via HTTP request header turns off the connection-orientedness. Again – this is something we noted awhile ago as a byproduct of a research in a different direction, and since I have no solid evidence, I am reluctant to point at it as a vulnerability. If someone can verify this on IIS/5.0 (I didn't manage to replicate it on IIS/6.0), please step forward...