3GPP TSG SA WG3 Security#30 **S3-030553**
6 – 10 October, 2003
Povoa do Varzim, Portugal

Agenda Item:	**7.9 (GBA), 7.18 (presence)**
Source:	**Siemens**
Title:	**Difficulties in using one TLS tunnel to access different servers behind an authentication proxy**
Document for:	**Discussion and decision**

1 Introduction

At the last 3GPP SA 3 meeting #29 it was agreed that " the proxy functionality providing the termination point for TLS should be used." One of the main claimed advantages when using authentication proxies, as described e.g. in Ericsson's S3-030371, is that the complexity of the UE regarding the handling of TLS connection is reduced. In the words of S3-030371 "Only one TLS connection should be used between the UE and the HTTP Proxy even when communicating with several Application Servers."

This contribution shows that the use of only one TLS connection conflicts with standard http behaviour, and that, in general, several TLS connections are required to access application servers with different DNS names, even when a reverse authentication proxy is used. There are several workarounds, but none of them is without problems, as the contribution shows. The only solution which allows the use of only one TLS connection and is compatible with standard http behaviour imposes restrictions on the directory structure of the application servers which may be unwelcome from an operational point of view.

2 Situation

- The user accesses the NAF from his browser via http or https.
- The application servers have their own domain name "server1.example.com"
- Only standard ports are used (http: 80, https: 443). This is the default for browsers, if no explicit port number is given in the URL.

3 Access with http (no TLS):

The user enters the URLs of different servers into his browser, e.g. "http://server1.example1.com/index.html" or "http://server2.example2.com/index.html".

3.1 Case: both DNS names resolve to the same IP address

This leads to the case of "name based virtual hosts". Distinction is made by the "Host" Request header field (required in HTTP/1.1, sent by most browsers in HTTP/1.0 also).
The NAF can do any action it wants to, i.e. serving own pages, proxying to an AS or sending a Redirect 301/302 message to the client (perhaps to redirect to https).
In case the Host Header field is missing, in HTTP/1.0 a normally default host is selected, in HTTP/1.1 the request should be rejected with "400 Bad Request".

3.2 Case: both DNS names resolve to different IP addresses

This leads to "IP based virtual hosts" or to entirely different servers/proxies on the same or on different machines. In case of one machine, it has to have a LAN card responding to multiple IP addresses.
The same actions can be taken as with case 3.1.

4 Access with https (SSL/TLS)

The user enters the URLs of different servers into his browser, e.g. “https://server1.example1.com/index.html” or “https://server2.example2.com/index.html”.

4.1 Case: both DNS names resolve to the same IP address

This leads again to the concept of “name based virtual hosts”, but this is in the best case flawed, if not unworkable with https.
Reason: The TLS handshake and therefore the server authentication takes place before the request header can be sent to the server. Therefore the server does not know which virtual host is the requested one and therefore which server certificate to send to the client. It can only send a “default” certificate. The browser normally takes the CN from the server certificate and compares it to the server name in the requested URL (for a more detailed description see chapter 7). If both coincide (if the user wants the “default” server), everything is okay. Otherwise an error window pops up telling the user about a problem and asking the user if he really wants to complete this action. This cannot be tolerated for everyday users.
Once the user accepts this “error condition”, the TLS connection is created and the HTTP request header is sent. Now the server can determine the requested host (in principle, e.g. Apache working as server and as proxy does it).

4.2 Case: both DNS names resolve to different IP addresses

This case has no problems with https and is similar to case 3.2. Based on the IP address the server or proxy knows which server certificate to send.

4.3 Discussion of https cases

Case 2.2 would be the desirable one as it has no associated problems. But it requires different TLS connections for the different servers, thus making the concept of single authentication for different servers more difficult. By no means would it be possible to transport all data over one TLS connection, as would be favourable for small user equipment.
There are three methods to make case 2.1 more viable, i.e. to work without pop up window in browsers and to have at least a “kind of” server authentication. But all of them have other drawbacks.

4.3.1 Workaround 1: distinct DNS Names

Distinct DNS names are often required, e.g. if wanted for corporate branding / identity or for portability of server names. RFC 2818 (cf. chapter 7) allows multiple dNSName name fields in a certificate, and “a match in any one of the set is considered acceptable”.
Advantages:
All server names are listed explicitly in the certificate; therefore the client can be sure to really connect to one of the authentic servers. A special browser behaviour could be defined based on this feature. The browser would then always try to send all http traffic to the set of DNS names in one certificate over the same TLS connection.
Disadvantages:
From the discussions in mailing lists it seems that this feature is not implemented in todays browsers. Also, to allow one TLS connection to address different servers, the normal operation of browsers has to be changed, as they normally open a distinct TLS connection for each distinct server name. The special browser behaviour mentioned above implies that the certificate issuer and the browser follow conventions regarding the use of certificates, which are not standard http behaviour today.
General:
Nevertheless it could be implemented in special browsers. It has to be checked whether todays widely accepted commercial certificate authorities are willing to provide such certificates. This may be an issue for operators not in control of their own PKI.

4.3.2 Workaround 2: wildcard certificates

Make all server DNS names to sub-domain names of one parent domain, e.g. “server1.my-domain.com” and “server2.my-domain.com”. Then get a wildcard certificate for “*.my-domain.com”. This works only for sub-domains of the same level, e.g. “server3.subdomain.my-domain.com” would not be acceptable.

Advantages:

Most browsers don’t complain about incompatible certificates. At least the parent domain is authenticated in this server authentication.

Disadvantages:

Citation from Apache-modssl mailing list just this month: <http://marc.theaimsgroup.com/?l=apache-modssl&m=106242827506368&w=4>

```
Yes, you can use wildcard certificates. It is possible to use them on the same IP
address and port and it works (this is from memory of what those who use this
method have written).

However:

1. CAs have got wise to wildcard certificates and charge a couple of limbs for the
privilege of using them.
2. There's no guarantee that IE will support it and Microsoft may well break sup-
port for it again.

If you are doing this on a "private" network, probably neither of the above will
affect you.
```

Further comments on the problem may be found in these mailing lists, name based virtual hosting with https comes up again and again. If I remember it right, only Thawte gave such certificates in the past, not Verisign (to be checked).

General:

Anyhow, this is no exact server authentication, as any server with this name convention will be accepted by the browser, e.g. also “fake.my-domain.com”.

4.3.3 Workaround 3: one DNS server name

Make all servers behind the authentication proxy look like one server. This means, all of them have the same DNS server name, but have a difference only later on in the URL. One solution might be to assign to every top-level directory one application server, e.g. “server.my-domain.com/serv1/index.html” and “server.my-domain.com/serv2/index.html”. The Authentication proxy can now call different remote servers depending on the information from the URL. (There might be other information from the URL to distinguish between application servers, depending on the proxying functions of the web server used as proxy).

Advantages:

CN in server certificate and DNS name of server will always coincide. All traffic to the different servers may go via the same TLS connection (important for small user equipment).

Disadvantages:

Even more than in twist 1 the names of the server are restricted, they must even be equal.

The name space and the link organisation inside the application servers must follow strict rules (perhaps critical with web content management systems, which are not widely configurable).

To give an example: Assume a distinction of servers by top-level directory. There are many web pages calling pictures with absolute links via <img src=”/images/blank.gif”> instead of relative links with <img src=”../../images/blank.gif”> in a page with URL “/dir/subdir/index.html”. This page is from the outside for the browser “https://server.my-domain.com /serv1/dir/subdir/index.html”. In fetching the image the browser would give the URL “https://server.my-domain.com/images/blank.gif”, as it does not know that the root directory of the web server does not coincide with the “virtual root” of the external URL.

This example is not exhaustive, other solutions (and other pitfalls) may be possible. It shall only show that this solution is not without problems. At least it hinders the independent administration of the application servers.

5 Access with http and protocol upgrading to TLS (RFC 2817)

The user enters the URLs of different servers into his browser, e.g. “http://server1.example1.com/index.html” or “http://server2.example2.com/index.html”. The server is configured to accept TLS connections only, i.e. it responds with “426 Upgrade Required“. The client has to react to this response depending on the kind of the connection. For further details not covered in this memo please refer to RFC 2817.
All further remarks about this case are tentative, as on one hand this RFC is quite old (May 2000), on the other hand it seems that it is not employed at the moment. E.g. for the Apache server experimental support only may be available in version 2.1, which has not even an estimated release date yet.

5.1 Connection without Proxy

On receipt of “426 Upgrade Required“ the client sends a new request with

```
OPTIONS * HTTP/1.1
Host: server.example.com
Upgrade: TLS/1.0
Connection: Upgrade
```

The server answers with

```
HTTP/1.1 101 Switching Protocols
Upgrade: TLS/1.0, HTTP/1.1
Connection: Upgrade
```

Now the TLS handshake starts inside the existing TCP/IP connection.

5.2 Connection with Proxy(ies)

Additionally all proxies in between have to support the http CONNECT method. This means that proxies must be able to switch for the existing TCP/IP connections from ordinary proxy operation (with interpretation of http headers) to tunnel operation (transparent pass-through) not only just after establishment (as for ordinary https), but also later on.
On receipt of “426 Upgrade Required“ the client sends a new request with

```
CONNECT server.example.com:80 HTTP/1.1
Host: server.example.com:80
```

The first and all further proxies in the chain have to react to this request and to take further action as described in RFC 2817. If this includes a reestablishment of the TCP/IP connection(s) is not stated and beyond my knowledge.
After the establishment of the tunnel the client has to follow procedures in chapter 5.1.

5.3 Discussion

Basically this solution behaves similar to the plain http access described in chapter 3. In particular, name based virtual hosting is possible, as the server receives the request header in plaintext before the establishment of the TLS tunnel. Still the server certificate will be examined by the browser; therefore it depends on how the browser implements the rules given in chapter 7. Access over the same TLS connection would probably require a change of the way the browser handles TLS and would not allow a cross-check of server certificate and particular server name (as in chapter 4.1).
If this solution is viable with proxies is beyond my knowledge, as there is just an email thread on the same Apache modssl list stating <http://marc.theaimsgroup.com/?l=apache-modssl&m=106338732106075&w=4>:

```
List:     apache-modssl
Subject:  Re: Are "client requested update" supported?
```

```
From:     Eric Rescorla <ekr () rtfm ! com>
Date:     2003-09-12 17:28:28

"Adrien Felon" <adrien@nexgen-software.fr> writes:
> Whatever, I am now wondering how strong is the pressure to migrate from the
> classical "https" scheme to the "client requested upgrade" (as I see these
> as somehow alternatives, as the client explicitely request "https"..). As
> far as I understand, RFC 2817 (May 2000.) clearly states that things like
> HTTPS should be deprecated. So I wonder what the "market" says... You say
> there is no client: am I really going to write the first one that supports
> this? As apache starts to support it, I guess there might be some other
> people looking fot it also.
There's no pressure at all, and for good reason. RFC 2817 is badly broken.

To take merely one example, it has a terrible interaction with proxies.

-Ekr
```

On the other hand time might bring a better support for this RFC. At first glance there shouldn't be a security flaw, as the server can always disconnect if the client does not properly respond to the “426 Upgrade Required“ response.

6 Conclusion

This memo addresses two problems, i.e. “name based virtual hosts over TLS” and “using a single TLS connection to access different hosts”.

6.1 Name based Virtual Hosts

Correct handling of name based virtual hosts is possible only in cases of chapter 3 (no TLS) and 5 (protocol upgrading to TLS). In the cases of chapter 4 (TLS without protocol upgrading) the server has no possibility to determine the host name before the TLS handshake takes place and therefore cannot send the appropriate server certificate. Workarounds have either multiple/wildcard server names in the certificate (support in browsers doubtful) or restrict the solution to one server name only.

6.2 One TLS Connection for many servers

Using one TLS connection to different servers contradicts the notion of “server certificates” for authentication, which is postulated by SSL/TLS and built into the browsers. Thus any twisting of this notion leads to problems, and a case-by-case trade-off evaluation has to be conducted.
Basically all cases of chapter 4 and 5 would allow one connection, but only case **Error! Reference source not found.** would not require a change of the common behaviour of browsers (which must be configurable, as this particular behaviour is only applicable to specific NAFs, not to general Internet access). None of these cases allows the authentication of a particular application server behind the NAF by means of TLS (but that might not be necessary).

7 Appendix: Server Identity

The following gives an excerpt from RFC 2818 (chapter 3.1) with respect to checks on server identity.
It has to be checked, if all methods described in this RFC are supported by browsers and/or certificate providers.

```
In general, HTTP/TLS requests are generated by dereferencing a URI.  As a
consequence, the hostname for the server is known to the client. If the
hostname is available, the client MUST check it against the server's iden-
tity as presented in the server's Certificate message, in order to prevent
man-in-the-middle attacks.

If the client has external information as to the expected identity of the
server, the hostname check MAY be omitted. (For instance, a client may be
connecting to a machine whose address and hostname are dynamic but the cli-
ent knows the certificate that the server will present.) In such cases, it
is important to narrow the scope of acceptable certificates as much as pos-
sible in order to prevent man in the middle attacks.  In special cases, it
may be appropriate for the client to simply ignore the server's identity,
but it must be understood that this leaves the connection open to active
attack.

If a subjectAltName extension of type dNSName is present, that MUST be used
as the identity. Otherwise, the (most specific) Common Name field in the
Subject field of the certificate MUST be used. Although the use of the Com-
mon Name is existing practice, it is deprecated and Certification Authori-
ties are encouraged to use the dNSName instead.

Matching is performed using the matching rules specified by [RFC2459].  If
more than one identity of a given type is present in the certificate (e.g.,
more than one dNSName name, a match in any one of the set is considered ac-
ceptable.) Names may contain the wildcard character * which is considered
to match any single domain name component or component fragment. E.g.,
*.a.com matches foo.a.com but not bar.foo.a.com. f*.com matches foo.com but
not bar.com.

In some cases, the URI is specified as an IP address rather than a host-
name. In this case, the iPAddress subjectAltName must be present in the
certificate and must exactly match the IP in the URI.

If the hostname does not match the identity in the certificate, user ori-
ented clients MUST either notify the user (clients MAY give the user the
opportunity to continue with the connection in any case) or terminate the
connection with a bad certificate error. Automated clients MUST log the er-
ror to an appropriate audit log (if available) and SHOULD terminate the
connection (with a bad certificate error). Automated clients MAY provide a
configuration setting that disables this check, but MUST provide a setting
which enables it.

Note that in many cases the URI itself comes from an untrusted source. The
above-described check provides no protection against attacks where this
source is compromised. For example, if the URI was obtained by clicking on
an HTML page which was itself obtained without using HTTP/TLS, a man in the
middle could have replaced the URI.  In order to prevent this form of at-
tack, users should carefully examine the certificate presented by the
server to determine if it meets their expectations.
```