Security Soup

Jeremy Allison

Samba Team.

Now with instant Simple - plus wholesome protocols!
What are we trying to do with security protocols?

- Security protocols are not an end in and of themselves.
  - We just want to add them to existing useful protocols.
- The idea is to add privacy (encryption), non-repudiation (signing) on top of client/server and peer to peer protocols.
  - This needs to be done in such a way as to interface transparently with existing authentication mechanisms (everyone wants single-sign on).
- Ideally this is done transparently to existing protocols so client/server writers get the benefits without having to understand the details.
Why is Samba concerned with this?

- Samba, and the Windows directory services it provides are where the rubber meets the road for many “standard” security protocols.
  - Plus a few extra surprises Microsoft cooked up on its own (they're the stones in the soup :-().
- “Standard” security protocols are SASL, SPNEGO, Kerberos5, GSSAPI.
- Microsoft “stones” are LM, NTLM, NTLMv2, LMv2, NTLMSSP, SMB Signing, DCE/RPC security and SChannel.
- Cryptographic standards such as MD4, MD5-HMAC, DES and RC4 are also in the broth.
All of these protocols in the SMB/CIFS world are designed to integrate with symmetric key encryption, not public key.

- No SSL or certificates to deal with, thank goodness :-)!
- Designed for “password” style authentication mechanisms (cryptographic hashes).
- Client and server both have knowledge of the “secret password” used to create the secure communications protocols.
Many of the “standard” protocols are described as “simple”.

- They even have this in the name!
- SPNEGO = Simple and Protected GSS-API Negotiation mechanism.
- SASL = Simple Authentication and Security Layer.

These protocols are not simple :-). They are designed to be generic so any authentication mechanism can be slotted in below this level.

- This leads to unbelievable complexity in trying to figure this stuff out.....
An example of “simple”...

- Security Blob: 60820B6206062B0601050502A0820B56...
  - GSS-API
    - OID: 1.3.6.1.5.5.2 (SPNEGO (Simple Protected Negotiation))
  - SPNEGO
    - negTokenInit
      - mechType
        - OID: 1.2.840.48018.1.2.2 (MS KRB5 (Microsoft Kerberos 5))
        - OID: 1.3.6.1.4.1.311.2.2.10 (NTLMSSP (Microsoft NTLM Security Support Provider))
      - mechToken
        - krb5_blob: 60820B2B06092A864886F71201020201...
          - OID: 1.2.840.113554.1.2.2 (KRB5 (Kerberos 5))
          - krb5_tok_id: KRB5_AP_REQ (0x0001)
    - Kerberos
      - Version: 5
      - MSG Type: AP-REQ
      - APOptions: 0020000000
  - Ticket
  - Encrypted Data: Authenticator
Basic Authentication Protocols – LM and NTLM (challenge response)

Client

Password Hash + Challenge

Response

Hello from client!

Challenge (BLOB)

Request to logon: Username + Response (BLOB)

Server

Password Hash + Challenge

Response

Logon successful or error message.
More complex Authentication Protocols – NTLMv2 and LMv2

Request to logon (client workstation identifier)

Challenges (BLOB)

Response + Client blob + (optional data)

Start of communication or error message.

Password Hash + Challenge + Server name + Username + Client blob

Response

Password Hash + Challenge + Server name + Username + Client blob

Response
Selecting NTLM or NTLMv2?

- This isn't intuitive. Extended security can be negotiated, but NTLM or NTLMv2 must be selected by hand (registry on Windows, smb.conf in Samba).

Extended Security Negotiated?
(Usually means high NT4 service pack or above).

- False
  - LM/NTLM or NTLMv2

- True
  - LM/NTLM or NTLMv2
  - Kerberos 5

Manually select
Kerberos 5 Authentication – Part 1

TGT and Server Ticket

Please send me a ticket-granting-ticket for user “foo” (with proof I know foo's password)

Ok, here is a TGT for user “foo” (encrypted with foo's password)

Please send me a service ticket for server “BAR”. Here's my TGT as proof I can ask this

Ok, here is a ticket for server “BAR”, with an authenicator encrypted in BAR's password

Kerberos KDC
Contains master passwords for all users and servers (services).
Kerberos 5 Authentication – Part 2
Using the Server ticket

Request to logon to server “BAR” as user “foo”.
Here is the kerberos server ticket.

Logon successful or error.

- More secure than the Microsoft proprietary NTLM and NTLMv2 protocols.
- Depends on synchronised clocks. If clocks not synchronised Windows clients silently drop back to NTLM/NTLMv2!
How do we decide whether to use NTLM or Kerberos?

Extended Security

SPNEGO

GSS API
- Kerberos

NTLMSSP
- NTLM/NTLMv2
Simple and Protected GSS-API Negotiation Mechanism

- SPNEGO (rfc2478) is a framing around GSS-API.
  - Uses the same on-the-wire protocol description (ASN.1) as GSS-API, Kerberos 5 and LDAP.
  - Designed to allow lower level selection of a security protocol (or authentication mechanism).
  - Used in SMB/CIFS to select either NTLMSSP or Kerberos.
  - Initial message contains a list of client supported protocols with optional data blob for the preferred protocol (the first one).
  - Windows (and Samba) clients appear to be the only users of SPNEGO to select any other protocol than Kerberos 5 (NTLMSSP).
GSS-API – Needless complexity?

- GSS-API (rfc2743) is actually an API that describes how applications can negotiate a security protocol.
  - I have only ever seen this used with kerberos 5 (rfc1964 covers how the GSS-API framing of kerberos 5 is marshalled on the wire).

- What this essentially means is that we have a stack of three protocols (SPNEGO/GSS-API/Kerberos 5) where one would have done (with an out of band mechanism).

- I'm not aware of any other multi-protocol use of SPNEGO or GSS-API.
SASL (rfc2222) is another method of selecting between different underlying security protocols (called “mechanisms” in SASL).

- SASL mechanisms are represented by text strings (actually simple :-).

- Each client/server protocol using SASL is responsible for passing the SASL blobs in a way specific to that protocol (many internet text-based protocols encode the blobs in base64).

- SASL represents a framework for naming different security protocols and describing their use in a client/server protocol.

- SASL has more than one protocol defined (unlike SPNEGO :-).
• SASL is used within SMB/CIFS for LDAP lookups to Active directory servers.

SASL can also wrap SPNEGO and GSS-API of course.....
Beyond Authentication – Session
Keys

- After authenticating the users long term password should not be used for any further cryptographic keys.
  - Basic security principles, don't expose passwords more than you have to.
- A random session key can be generated as part of the authentication protocols.
  - Kerberos is rather better in that the KDC generates a random session key for further use and adds it to the ticket.
  - For NTLMSSP the session key can be one derived from the hashes in the authentication protocol (bad) or the client can propose a session key and ship it to the server encrypted by the derived session key (better maybe ?).
SMB signing is used to prevent TCP session hijacking of a CIFS/SMB data stream.

- It uses the previously described session key to “sign” each SMB/CIFS message, using an incrementing sequence number synchronised between client and server.

- The MD5 signature replaces the sequence number in the packet sent on the wire.
SMB Signing Problems

- The above sounds simple enough to implement from the description (like most Microsoft “documentation”).
- It doesn't take into account asynchronous messages from server to client like oplock breaks.
- It also doesn't document how to cope with partial message fragments (SMBtrans calls).
  - Turns out the sequence number must remain constant over each part of the streamed message.
- Getting this right is non-trivial. Microsoft currently is battling a “data corruption when SMB signing is turned on” bug reported into the press by a disgruntled Windows software vendor.
• Windows RPC protocol (DCE) can be transported over “raw” TCP or layered over SMB packets (via end-points called “named pipes” such as `\PIPE\SPOOLSS`).

• DCE/RPC can negotiate security on a connection and has protocol specifications for adding packet “signing” (like SMB signing) and packet “sealing” (bulk encryption).

• Microsoft don't use the published DCE security mechanism but instead used NTLMSSP and now Kerberos 5.

• An unauthenticated (guest) SMB connection can then bootstrap itself into an authenticated pipe connection by using DCE/RPC security negotiation.
DCE/RPC Encapsulation in SMB

- SMB Packet (may be signed)
  - SMB Header
  - DCE Payload (or partial fragment)
    - DCE Header
    - DCE Data
    - DCE Verifier

- DCE header is unencrypted
- DCE data may be encrypted
- If exists contains signature of DCE packet.
- SChannel is used to secure Windows Domain Controller to Domain controller communications.
  - Could be used as a generic Microsoft DCE/RPC encryption mechanism.
- Uses signing and sealing with an incrementing sequence number as part of the signing check.
- Probably only Samba needs to worry about this protocol, I doubt it will bleed out into the generic security protocol community.
NTLMSSP is widely used by Microsoft clients as a SASL negotiated mechanism.

- Web browsers, mail clients and others will use this if available.

- Kerberos 5 support is useful but not yet enough to provide secure single sign-on in all circumstances.

- Open Source/Free Software projects need to be able to cope with NTLMSSP protocols.

- Samba can help here, with the ntlm_auth helper used by squid and other projects.

- Unfortunately this means setting up Samba correctly :-).
Questions and Comments?

Email: jra@samba.org

Opening Windows to a Wider World