Windows Protocol Analysis: MSCHAP & Friends

Gros, Charles-Henri
Haley, David
Lisanke, Bob
Schaff, Clovis

Initial Project Goals

Our first idea was to study Microsoft file-sharing protocols, such as SMB. We especially wanted to look at how secure these protocols were; notably authentication and the following communication.
After about ten man-hours of research, we discovered that the area was extremely vast, with at least 6 different dialects and no clear, formal specifications. For example, the latest incarnation of file-sharing is the Microsoft Common Internet File Sharing Protocol (CIFS) but there is no formal specification for this: even the informal specifications are just "drafts" and "proposals".

Phase One: Project Narrowing

During our research, we came across an interesting article by Bruce Schneier and "Mudge" that studied the cryptography used in Microsoft point to point tunneling protocols; notably, the LanMan hash used in MS-CHAP1.
This article can be found at: http://www.schneier.com/paper-pptp.html
We found this very interesting and it gave us a good starting point for something concrete. We started looking for an RFC for MS-CHAP1 and we were very happy to find RFC2433. That led to RFC1994, which describes CHAP, and RFC2759, which describes MS-CHAP2.

What is CHAP?

CHAP stands for Challenge Handshake Authentication Protocol. It is used in point-to-point user authentication and follows the "Password Paradigm". That is, the user submits a password (one way or another) and the server verifies this. CHAP itself does not specify encryption/hashing algorithms and simply describes the steps to be taken.
MS-CHAP1 is Microsoft's extension of CHAP for their purposes. It chooses a hashing algorithm (LanMan and MD4 for the NT hash), and has some extra features such as extensive failure message types.
MS-CHAP1 had some major security flaws (as described by Schneier/Mudge's article) and therefore Microsoft designed MS-CHAP2, which implements more robust cryptography as well as server authentication.

Phase Two: Murφ Modeling

We chose Murφ as a tool to model this protocol, which turned out to work quite well. It took some time to get around the syntax (especially the rule priorities!), as well as how to correctly model lots of session numbers, challenges, etc.

Our first Murφ model was over-simplistic and was designed to show the attacks Schneier described. We were not happy with such a "stupid" intruder, so we redesigned the code under a much more robust and generic model. The hardest part was representing who knows what, and who can generate which hashes based on what knowledge.
Our intruder implements a rather different attitude in that it does not remember whole messages. Rather, as it sees messages go by, it adds components to its knowledge, and is then capable of rebuilding messages based on which components it has.

Attacks Found

We modeled/verified the attacks Schneier outlines, and also found a man-in-the-middle attack.

We also saw a new attack for MS-CHAP1, in which the intruder can send a FAILURE_EXPIRED message to a client, thus prompting a ChangePassword packet – which contains the LanMan hash, which is vulnerable to brute-force/dictionary attacks. Thus, we used the knowledge from Schneier's article (that a LanMan hash can be easily broken) to show that the protocol allows this hash to be sent. In theory the protocol does not normally send the LM hash, but only does when a ChangePassword packet is sent.

Conclusions

From slides:

- Hard to sort through morass of informal specifications
- MSCHAP2 seems to fix MSCHAP1 problems, but allows for version roll-back attacks
- Murφ seems adequate for this protocol
- However, the found attacks are obvious enough after having formalized the RFCs
- MSCHAPv2: better crypto, but still only as secure as password
- Backwards compatibility removes much of the point of an upgrade – both for MSCHAPv1 (LanMan hash) and MSCHAPv2 (compatibility with v1)
- MSCHAPv1 mistake (poor hash) should have been avoided
    - Improper, insufficient cryptanalysis
- Big problem with MSCHAPv1 is not the fault of the protocol itself
- MSCHAPv2: more robust crypto, but protocol is still flawed