TWiki > DICE Web > FinalProjectReport168 (02 Nov 2016, TobyBlake)

Final project report for "Password strength checks" (168)

Project Description

The April 2010 Development Meeting suggested a project which would "look at checking the strength of existing passwords on the KDC and/or checking password strength when changed on the KDC (at the moment strength checks are only made on DICE clients)." The Operational Meeting of 22nd June 2011 subsequently returned to the subject of password security (in the context of Indicident Management), and how we can best avoid our users having weak passwords. Mention was made of a PAM module which captures passwords entered by users and flags up passwords considered weak; it was agreed that investigation of that module should be added to this project. The project should ideally allow for the possibility of using several different types of checks, operating in and/or combinations. Whether the kadmind hooks allow for this remains to be seen, though a meta-module might do the job if it's not there natively.

Project page

Project updates

Work Done

This project picked up on investigation work done previously by George Ross. At this point the project was stalled until our KDCs had been upgraded to a version of kerberos with the appropriate pluggable interface.

Initially, we looked at the existing policies we had in place on our systems and in what contexts they were applied (i.e. in the different ways a user could set/change their password). We found that ... "We have four different places in which password policies are applied. They are all different from each other, both in terms of the restrictions they apply and the rules they follow (e.g. what constitutes a given character class)." Following this, we determined it was important that there be only one policy applied (by us) to password changes and that this had to be applied at the server side. We also decided that it was important that our password policy be flexible in terms of the restrictions applied, i.e. the longer the password, the fewer class restrictions.

MIT kerberos has a pluggable interface for password quality checking and Russ Allbery has produced a module (to our knowledge, this is the only such publicly available module).

We investigated using this module. It fitted what we wanted to do, but the character class restrictions weren't quite what we wanted - it offered flexibility based on password length, but only in terms of what specific character classes were included (e.g. for a password of 8-12 characters, that it must have lower case, upper case and digits). We wanted to be able to specify that a password of a given length include a certain number of different character

classes, without specifying what classes they were.

We patched the krb5-strength code to provide the above functionality. The patch has been passed upstream (but there has been no new release of the module since March 2014).

We also patched the krb5-strength module to provide more accurate information to users on the reasons for passwords being rejected. This was necessary because the messages returned made assumptions that the kadmin built-in password policies are being used exclusively. We patched kadmin to provide errors linking to our online password policy. We felt this was an acceptable trade-off between providing useful information to users and the unpleasantness of hard-coding local policy information in software. This is only needed on the KDC master.

We originally planned on using the cracklib dictionary check as part of our password policy. However, following concerns raised by the author of the krb5-strength module about the code quality of cracklib, we decided against this. Instead, we used the sqlite database checking provided by krb5-strength - this rejects a password if it is is within "edit distance one of any word in the dictionary, meaning that the database word can be formed from the password by deleting, adding, or changing a single character".

The wordlist we use for this dicionary check is a combination of the one distributed in RedHat's cracklib-2.9.6 RPM with the free openwall wordlist. In total, this contains nearly 5.8 million words, English and other languages.

The above fits into our wider security policy, in that we expect any brute-force dictionary guessing attack to be primarily thwarted by our running of fail2ban on ssh servers and KDCs.

To reproduce some text from the project's update page:

It is worth stating what we consider the threats to any individual user's passwords to be and how to protect against these:

1. Brute-force dictionary attack against our systems
    a. from outside the University
    b. from within the University
2. Password re-use (i.e. DICE password used elsewhere) and password compromised on another system
3. Password compromised externally (poor password management, phishing, etc.)

Use of the krb5-strength module with a good wordlist protects against 1. Our use of fail2ban protects us further against 1a. (this currently only applies to attacks from outside the University networks - we should perhaps consider extending this to all IPs from inside the University, to protect against 1b.)

2. and 3. can only really be guarded against through user education and awareness. Education is important to enable users to, e.g. spot phishing attempts. If a password has been compromised, it is important that we spot this as soon as possible - e.g. through our monthly mailing to each user detailing their successfull authentications. More details on this

and other, related, considerations here.

In addition to the above, all sysadmins receive a weekly email with details of authentications (successful and unsuccessful) against their principal. We also gather and process statistics on all attempted authentications against our KDCs on a weekly basis.

Our password policy is published here:

http://computing.help.inf.ed.ac.uk/password-policy

Corresponding system blog article:

http://blog.inf.ed.ac.uk/systems/2016/09/27/changes-to-dice-password-policy/

The new password policy was in place in time for this year's new student intake.

Effort Spent

The project took 174 hours (gdmr: 40, toby: 134) in total (~25 days)

Comments

As always, having a test realm (TEST.INF.ED.AC.UK) was invaluable.

Future Considerations

We should consider performing an offline brute-force attack against our existing KDC passwords to identify any weak ones.

We should ensure we introduce fail2ban (or alternatives) for all authentication points into our systems.

We should consider extending our fail2ban protection for EdLAN addresses as well as external ones.

The above two points, in particular, require cross-unit coordination. See Fail2BanMeeting20160818.

We should ensure any IP-level protection works for IPv6 as well as IPv4.

We should consider ways in which we can utilise information in the KDC (and KDC logs) to spot unusual or suspicious activity (although this is, of course, difficult to define). Project #351 may be useful in this respect.

-- TobyBlake - 01 Nov 2016

Topic revision: r2 - 02 Nov 2016 - 13:56:16 - TobyBlake