Managing Suck: Kerberos Password Quality at the Universty of Michigan

Thomas L. Kula

Information and Technology Services

University of Michigan

2010 AFS and Kerberos Best Practices Workshop

Passwords should...

- Be easy for the user to remember

Passwords should...

- Be easy for the user to remember
- Be hard for a bad guy to know

Passwords should...

- Be easy for the user to remember
- Be hard for a bad guy to know
- Allow a user to quickly access her resources

Passwords should...

- Be easy for the user to remember
- Be hard for a bad guy to know
- Allow a user to quickly access her resources
- Keep bad guys out forever

These are often mutally exclusive requirements

CPE1704TKS

But users are very possessive of their data

Even data that doesn't matter...

Even data that doesn’t matter...

...but who decides what matters?

I don't want to have to type a novel to read my e-mail, why can't my password be 'password'!

But, of course, you're to blame for letting someone else into their account

But we can give “high-security need” users tokens, right? They’re the only ones we care about, right?

But many users have data that is very sensitive

Dr. Dinwiddle loves chunky peanut butter, and we only give tenure to smooth peanut butter lovers in this department!

Budget cutbacks! Hah! We can't buy paperclips but the Chair gets new carpet in her office!

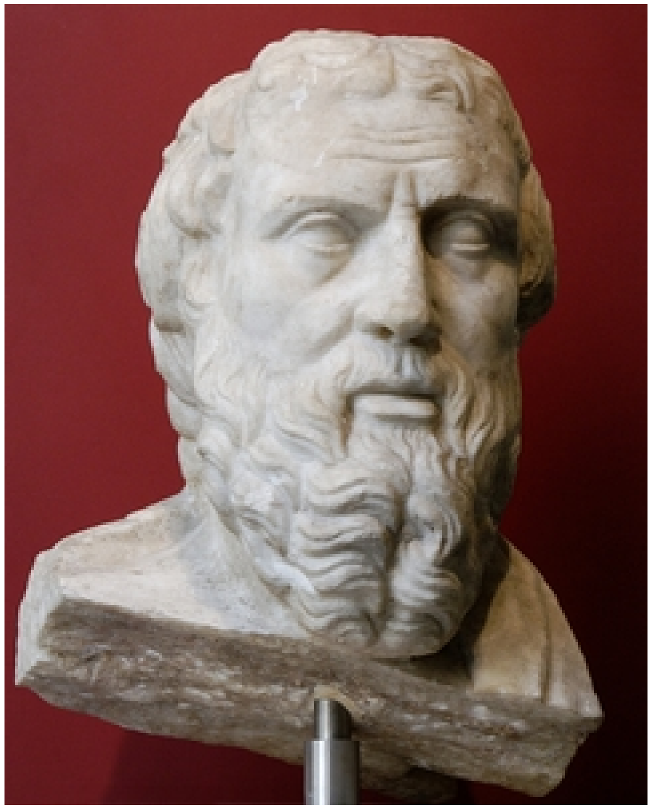
- These examples are university-centric
- But similar concerns exist in any organization of any size
- Or even in any community of any size

In short, the compromise of even “low security need” accounts could still cause problems for any organization

The Michigan Daily
SUMMER EDITION

Detroit Free Press

THE CHRONICLE
of Higher Education



Some History

Some History

- Twice-yearly audits of the kerberos database using *John the Ripper*

- Which can handle afs3 keys (ugh)

Some History

- An amazing number of keys cracked...
- ...in a depressingly short period of time

Some History

- Departments notify users
- Departments had to opt-in to doing anything
- And only recently did some of the larger units opt-in

Some History

- Even after forcing users to change passwords...
- ...many chose weak passwords again
- ...and keep showing up in audits

ITS needed something stronger to try to fix this

Kerberos Password Quality Plugin

- Derived from a framework written for Stanford
- Updated to work with MIT Kerberos 1.6.3
- A locally developed plugin that uses Cracklib

Cracklib

- ...is special

Cracklib

- Essentially a compliment to John the Ripper

Cracklib

- ...is ugly

Cracklib

- Assumes it is part of a one-shot application
- Has very ugly static buffers all over the place
- Leaks file descriptors if you use more than one dictionary
  - But only if the dictionaries have different names....
- Assumes that people's names are in `/etc/passwd`

Rollout, Part 1

- Put all previously cracked passwords in a dictionary
- Add some extra rules to cracklib
  - More Rules = More Better!

Initial Reaction

THE SKY IS FALLING!

Initial Reaction

- In retrospect, the additional rules were a blindingly bad idea
- In retrospect, having previously-cracked passwords in a crack-lib dictionary was a blindingly bad idea

Initial Reaction

- Did I mention that **`kadmind`** really hates running out of file descriptors?

Finding Assumptions in Weird Places

- Part of our legacy provisioning system would generate random passwords for users

- Passwords which didn't always pass the new plugin's checks...

- ...and would really confuse users who told it to generate a password for them and got an error message saying the password they supplied wasn't good enough

Rollout, Part 2

- Previously broken passwords move to simple lookup list

- Extra rules removed

- Single dictionary

More Reaction

- Why did *this* password fail?

- Why *doesn't* this password fail?

Why did *this* password fail?

- It’s trivial for a human to look at a password and decide, “Yeah, that looks good”
- It’s incredibly hard to algorithmically decide “Yeah, that looks good”

Why *doesn't* this password fail?

- The best you will get with this strategy is a giant list of rules

Why *doesn’t* this password fail?

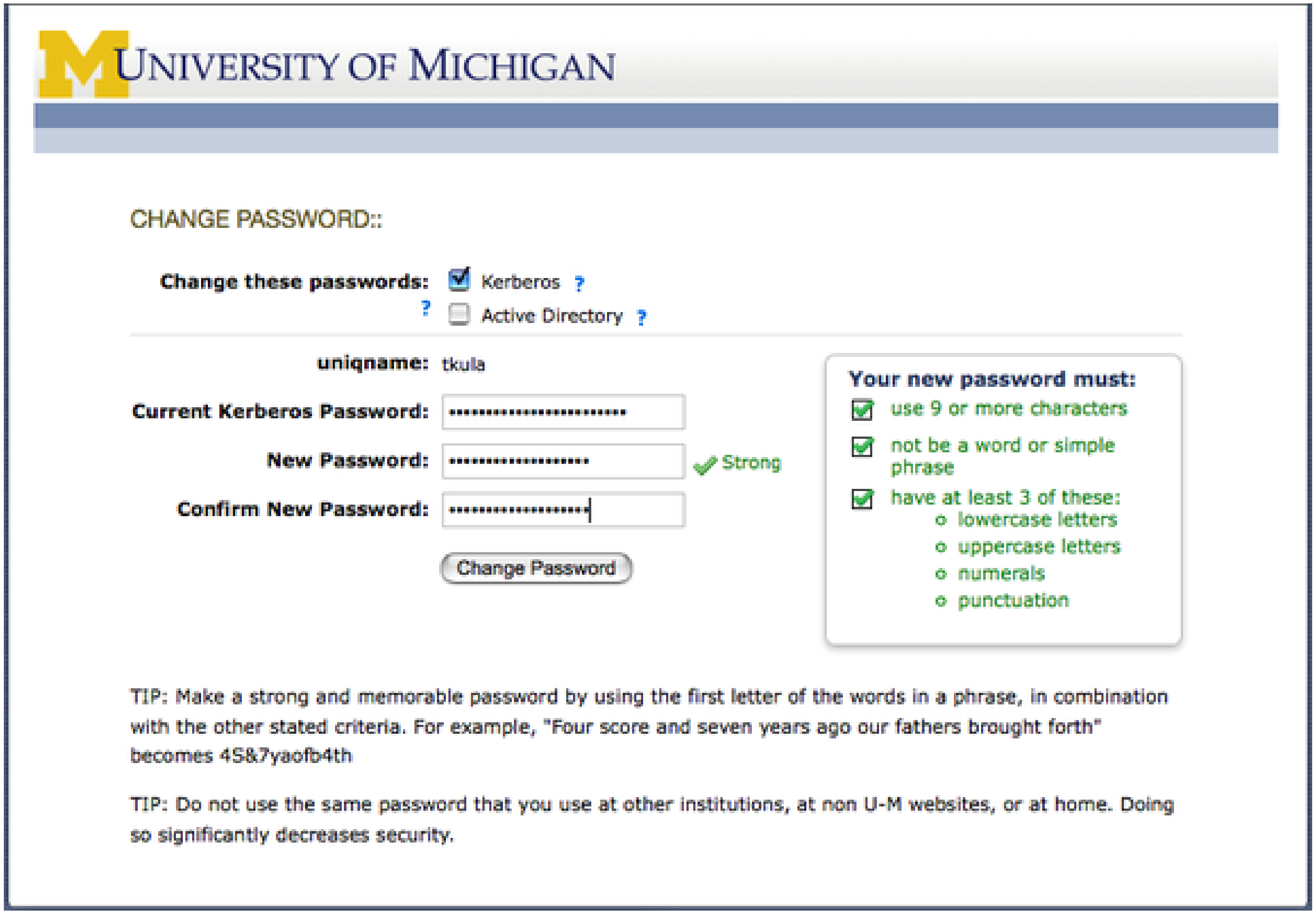
- The best you will get with this strategy is a giant list of rules
- Users are incredibly good at inventing new ways of making bad passwords

Worse, people have different ideas of what a good password is

`2emPeech.` *is too much like the phrase* `To Impeach`

How do you explain to users what they have to do to get a good password?

- With some hundred or so crackib rules and 1.7 million words in a dictionary, you can only give a guideline
- My explanation for how the plugin works, *for support staff*, goes on for two pages.


UNIVERSITY OF MICHIGAN

CHANGE PASSWORD::

Change these passwords:	[✓] Kerberos ?
?	[ ] Active Directory ?
uniqname:	tkula
Current Kerberos Password:	••••••••••••••••••••••••
New Password:	•••••••••••••••••••• ✓ Strong
Confirm New Password:	••••••••••••••••••••

Change Password

Your new password must:

- ✓ use 9 or more characters
- ✓ not be a word or simple phrase
- ✓ have at least 3 of these:
  - lowercase letters
  - uppercase letters
  - numerals
  - punctuation

TIP: Make a strong and memorable password by using the first letter of the words in a phrase, in combination with the other stated criteria. For example, "Four score and seven years ago our fathers brought forth" becomes 4S&7yaofb4th

TIP: Do not use the same password that you use at other institutions, at non U-M websites, or at home. Doing so significantly decreases security.

We have a password quality plugin…

…we're done, right?

Future Thoughts

- We audit passwords using John the Ripper
- We ensure password quality using Cracklib
- Which are essentially compilments of each other

Doc, it hurts when I use that password!

Then don't use that password!

Does our combination of JtR/Cracklib catch all the threats we care about?

Have we even quantified what all the threats we care about *are*?

Most importantly, are we looking at what new threats are, and reacting to those new threats?

A case for forcing password changes every N months

...but not for the reason you're thinking

How good is a really strong quality assurance system if there are passwords from 1993 that have never gone through it?

And, how do you get rid of ancient key types?

We will have principals with only single-DES keys

- Until we force people to change passwords

We will have principals with only single-DES keys

- Until we force people to change passwords

- ... or until the heat death of the Universe

Forcing Password Changes

- I don't think forcing users to change passwords every 90 days adds much

- But making them change it at least every two years gives you a decent timeline for rolling out new encryption types and phasing out old ones

Thanks and Questions