

---[Anonymizing UNIX Systems]---

version 0.9

Author: [van Hauser](#) / *THC*

I. [THE AUDIENCE](#)

II. [GOAL](#)

III. [PREREQUISITES](#)

IV. [USER DATA](#)

1. [Sensitive user data](#)
2. [Protecting /home directories](#)
3. [Traceable user activity](#)
4. [Protecting /var/spool/mail/user files](#)

V. [SYSTEM DATA](#)

1. [Sensitive system data](#)
2. [Traceable system activity](#)
3. [Logging - important and dangerous](#)
4. [Protecting system configs](#)
5. [Computer Memory and sensitive /proc interfaces](#)

VI. [DELETE\(D\) DATA AND SWAP](#)

1. [How to delete files in a secure way](#)
2. [How to wipe free disk space](#)
3. [How to handle swap data](#)
4. [How to handle RAM](#)
5. [Temporary data - it is evil](#)

VII. [NETWORK CONNECTIONS](#)

VIII. [HIDING PRIVACY SETTINGS](#)

1. [Mount is your friend](#)
2. [Removable Medias](#)
3. [???](#)

IX. [EXAMPLE CONFIGURATION AND SCRIPTS](#)

X. [FINAL COMMENTS](#)

1. [Where to get the tools mentioned in this text](#)
2. [Additional thoughts](#)
3. [Greetings \(what would the world be without greets?\)](#)
4. [How to contact me for updates or comments](#)

*** I. THE AUDIENCE**

This text is for any human being out there who wishes to keep their data and doings private from any snooping eye - monitoring network traffic and stealing/accessing the computer including electronic forensics. Hackers, phreakers, criminals, members of democracy parties in totalitarian states, human rights workers, and people with high profiles might be interested in this information. It was especially written for novice hackers so they are not so easily convicted when busted for their early curiosity.

Thanks to Solar Designer, Fyodor, typo, tick, pragmatic, mixer and doc holiday for comments, critics and ideas.

Special thanks to rookie who had the original idea writing this paper but through personal problems couldn't do it himself.

*** II. GOAL**

Our goal is to provide solutions to the following statements:

- (1) The solution should be simple and easy
- (2) All user data should be inaccessible by anyone except their owner
- (3) Nobody should be able to reconstruct what is happening on the system

Maybe you see contradictions ;-)

*** III. PREREQUISITES**

It is important to state the prerequisites for this project:

- The system should be secure. No remote vulnerabilities (and hopefully no local ones either)
- The system administrator(s) must be trusted and willing to set this up
- The operating system to achieve this is a UNIX

Note that the solutions presented do not 100% fit internet servers.

However it's (nearly, bah ;-)) perfect for enduser systems.

For the UNIX part, we show the solutions for Linux because it is the unix most easily for beginners to get their hands on and administrate.

The Linux distribution we use is the SuSE Linux Distribution 6.0

Debian is better but more complicated for beginners. And I dislike redhat for it's missing security.

You should know enough about unix (what is portmap, mount, rc2.d etc.) before trying to understand this text. It's **not** a Linux-Howto!

* IV. USER DATA

*** 1. Sensitive user data

What is sensitive user data? Well *any* data from a user account. This includes:

- utmp/wtmp/lastlog data (login times and duration plus login hosts)
- history files (what commands you typed in your session)
- your emails
- temporary files from applications like mailers, browsers etc.
- applications and their configuration
- your own data (documents, porn pics, confidential data)
- time stamps on your data (when were you accessing/editing which data)
- on multiuser systems: what users CURRENTLY are doing.. this includes process listing, and network connections as well as utmp (which is already covered by another category). -> make proc more restrictive.

We are trying to protect all this data.

Note that utmp/wtmp/lastlog data and mail (mqueue/mail/fax/lpd) is handled in the SYSTEM DATA section.

Note that all user accounts can be seen from /etc/passwd ;-) So maybe you'd like to add some/many fake accounts, together with homedirs and crypted data ...

*** 2. Protecting /home directories

Most important for protecting user data is protecting the users' /home directories.

Each home directory must be encrypted with a strong cypher so that even with full physical access to the system the data can't be obtained. Currently I know of only one software providing a solution to our requirements: CFS - the cryptographic filesystem.

There are also some other crypto solutions available : TCFS, SFS and the loop filesystem with crypt support. They are faster but have got the disadvantage that you'll have to recompile your kernel with patches from these tools. So for the sake of easeness, I stick with CFS here. (Pointers to all tools mentioned in this text can be found at the end)

To enable CFS we must put these six lines in a rc2.d script:

```
portmap
rpc.mountd -P 894      # mountd should bind to port 894
cfsd 895              # cfsd   should bind to port 895
rm -rf /tmp/.tmp
mkdir -p -m 700 /tmp/.tmp
mount -o port=895,intr localhost:/tmp/.tmp /home
```

Additionally we have to put this entry into /etc/exports:

```
/tmp/.tmp      localhost
```

Okay. This starts the sunrpc with the mountdaemon which are necessary for CFS to be started and used.

Now we need to get the following going: if a user logs on, the system has to check if he's already logged in to decide whether to decrypt the users' home directory. This sounds hard but is easy:

the user's /home/user directory doesn't exist (even if it would, because of mount command nine lines above would make it nonexistent), so the user's HOME variable is set to '/' the root directory. Then his login shell is started which looks for it's start scripts. And that's were we put our hooks in.

We create (this example is for bash) the file /.profile with the following contents:

```
cattach /crypt/$USER $USER || exit 0
export HOME=/home/$USER
cd $HOME
if test -f $HOME/.profile; then
    . $HOME/.profile
fi
```

When a user logs on the first time, this script will be executed. The user has to enter the password for his crypted homedir, and after this his correct HOME variable is set and the normal login profile is read and done. If a user doesn't know the passphrase for his crypted homedir, he is logged out.

But how do we remove the decrypted homedir after the user logs out? This script should be clever, because a user could be logged in several times at once, and it should only be removed when the last loginshell exits.

Thank god, this is easy too, we create a /home/user/.bash_logout script:

```
# if the number of user's login shells are > 3 then this is the last.
shells=`ps xu | grep -- "$USER .* S .* -[^\ ]*sh" | wc -l`
test $shells -lt 3 || exit 0
export HOME=/
cd /
cdetach $USER
```

Thats all. From now on, the users' homedirectories are safe.

Note that a user can't login now, start a background job which writes data in his homedirectory and log out because his homedirectory would be removed. The full .bash_logout script I provide in (see two lines below) checks for a \$HOME/.keep file and if present doesn't remove the homedir.

For network logins you should keep in mind that they should not be done via rlogin, telnet, etc. because they send all traffic (including passwords) in plaintext over the network. You should use a tool which encrypts the whole traffic like SSLtelnet or SSH (for SSH you need to set "UseLogin yes" in the /etc/sshd_config file).

You'll find all these scripts with error checking, user creating, stop scripts and config files etc. in section IX. EXAMPLE CONFIGURATION

Note that we started daemons in the section which can be contacted from remote. If you don't want this (because there are no external users who need to mount their crypted user data on their own machine) you should firewall these ports. Look in you manpages ("man ipchains" or "man ipfwadm").

*** 3. Traceable user activity

[Warning, this section shows first how to perform simple electronic forensics]

It is easy to see who logged on the system and what he did by the timestamps. Even if all your data is crypted, by checking the last access time (atime) of your files, someone may check when you logged in last time, for what duration and if you were idleing or doing much stuff.

If the systems doesn't have many users, someone might even tell what you did.

Example: The earliest access time for a crypted file in your homedir can be seen by:

```
ls -altur /crypt/$USER | head -1 # shows the logout file
ls -altu  /crypt/$USER | more    # with some brain you'll find
                                     # the login time
```

then you also have the duration of the session.

By checking the change/modification and access time of those crypted files with their timestamps someone can see how hard you were working, and get more conclusions (e.g. if many files nested in a three levels deep directory where modified this is probably a browser - so you were surfing the net).

This insight will now make it possible to check what commands were run:

Let's say the login time as 22 hours ago, so you run:

```
find / -type f -atime 0 -ls # shows the accessed files
find / -type f -mtime 0 -ls # shows the modified files
```

(this can be done with directories too)

Now check the output for the correct timeframe and analyze what you found. e.g. the telnet client was accessed. So it's probable, the user used it to connect to another system. I think you can imagine now what is possible.

To protect against this is also very easy:

Create the file /usr/local/bin/touch_them and make it executable with the following contents:

```
find /crypt /tmp /etc /var/spool 2> /dev/null | xargs -n 250 touch
```

Then put the following line into /etc/crontab:

```
50 * * * *      root    /usr/local/bin/touch_them
```

finally you change the 4th row of all lines in /etc/fstab which have the keyword "ext2" in their third (the filesystem type) row:

```
defaults          (or anything else)
```

should become

```
defaults,noatime (the old value is kept, and noatime is appended)
```

example:

```
/dev/hda1 / ext2 defaults 1 1
```

becomes

```
/dev/hda1 / ext2 defaults,noatime 1 1
```

What did we achieve? The crontab entry with the small script updates the atime, mtime and ctime to the current time every hour of special directories - especially those which may hold user data.

The mount options we changed now prevent the update of the atime. However, this needs a current 2.2.x kernel - it isn't implemented on the 2.0 kernel tree!

*** 4. Protecting /var/spool/* files

/var/spool/mail :

Now it gets tricky. How can we protect the new mail for a user from spying eyes? It can't be sent directly to a user's homedir like qmail would do because it's crypted. The easiest solution is to use pgp to encrypt your outgoing emails and tell all your friends that they should also encrypt all emails to you.

However, this is not satisfying. An attacker can still see who sent the user the email. The only possibility to hide this is using anonymous remailer. This is not a great solution, so this is an open point (see section [X.2](#): Additional thoughts)

/var/spool/{mqueue|fax|lpd} :

Well, all you can do is try to flush the queues when shutting down.

After that you have to decide if you delete the remaining files in a secure way or leave it where it is. Or program a special script which does something with the data (like taring the data and encrypting it with pgp, doing the reverse when the system is rebooted)

You can also create a whole crypted /var partition, but that would require someone at the console while booting the system - every time.

* V. SYSTEM DATA

*** 1. Sensitive system data

What is sensitive system data? **Anything** which gives conclusion on incoming and outgoing data, configuration files, logs, reboots and shutdowns.

This includes:

- utmp/wtmp/lastlog data (boot, reboot, shutdown times + user times)
- ppp dialup script
- sendmail and tcp wrapper configurations
- proxy cache data (e.g. squid web/ftp proxy)
- syslog messages
- /var/spool/* data {mqueue|fax|lpd|mail}
- temporary files from daemons
- time stamps on data (when were what data accessed/edited)

How to prevent time stamp forensica, see section [IV.3](#)

How to protect /var/spool/* data, see section [IV.4](#) for an incomplete solution.

*** 2. Traceable system activity

(prevent of time stamp forensic is handled in section [IV.3](#)) To trace system activity, you can easily check temporary files of daemons and applications. Some of them write to /tmp, root applications usually (should) write to /var/run. We handle this together with section [V.3](#): Logging. All you have to do is this, and only *once* :

```
cd /var
mv run log
ln -s log/run run
```

this moves the /var/run directory to /var/log/run and sets a symlink in it's former place so that applications still find their files.

*** 3. Logging - important and dangerous

Logging is important to trace problems like misconfigurations.

Logging is dangerous because an attacker can see important data in the logfiles, like the user's login and logout time, if they executed "su" or other commands etc.

We try to find a balance between this.

Our solution: Write all log data to one special directory.

This directory is a RAM disk so the data is lost after a system shutdown. Ensure that syslogd [/etc/syslog.conf] and daemons (e.g. httpd [apache]) only write to our special logging directory or a system console. /var/log should be used as our special logging directory.

Now we put the following commands into /sbin/init.d/boot.local:

```
umask 027
mke2fs -m0 /dev/ram0 1> /dev/null 2>&1
rm -rf /var/log/* 2> /dev/null
mount -t ext2 /dev/ram0 /var/log
chmod 751 /var/log
cd /var/log
mkdir -m 775 run
chgrp uucp run
for i in `grep /var/log /etc/syslog.conf|grep -v '^#' | \
awk '{print $2}'|sed 's/^-//'\`
do > $i ; done
umask 007 # 002 might be used too.
for i in run/utmp wtmp lastlog
do > $i ; chgrp tty $i ; done
cd /
kill -HUP `pidof syslogd` 2> /dev/null
```

After your next reboot it behaves like described above.

Some of you will not like the idea of having no logs after a reboot. This way you can't trace an intruder or guess from your logs what crashed the machine. Either you can tar the files and pgp before the shutdown is complete (but the data would be lost if a crash occurs), or you might also use ssyslog or syslog-ng, special syslogs with crypting capabilities, and write the data you really want to keep to (just an example) /var/slog.

You can also create a whole crypted /var partition, but that would require someone at the console while booting the system - every time.

***** 4. Protecting system configs**

This is tricky. It is easy to achieve but for a price. If we create an account with uid which has his homedir in /home and is hence protected by our CFS configuration, you need to be at the console at every reboot. This isn't practical for server systems that need to be administrated and rebooted remotely. This solution is only good for end-user pcs.

Just create an account with the uid 0 (e.g. with the login name "admin"). You can use the create_user script from section IX.

Put all your sensitive configuration files you want to protect into this directory (ppp dialup scripts, sendmail.cf configs, squid configs with their cache directory set to a subdir of "admin" etc.)

Now create a small shellsript which starts these daemons with a command line option to use the config files in your "admin" homedir.

Your system is then secure from extracting the sensitive information from the config files. But for a price. You have to log in after each reboot as user "admin", enter your CFS passphrase and start the script.

***** 5. Computer Memory and sensitive /proc interfaces**

For a real multiuser system on which the administrator want additionally ensure the privacy of the user online, he has to hide the user process information, a user would normally see when issuing a "who" or "ps" command. To protect the user's process information, you can use Solar Designer's secure-linux kernel patch. To protect the utmp/wtmp/lastlog we ensure that these files are only readable by root and group tty, hence a normal user can't access this data. (This is done in the boot.local example script)

Now one problem is left. Even with normal RAM a well funded organisation can get the contents after the system is powered off. With the modern SDRAM it's even worse, where the data stays on the RAM permanently until new data is written. For this, I introduced a small tool for the secure_delete package 2.1, called "smem" which tries to clean the memory. This one should be called on shutdown. It is done in the example in section [VI.4](#)

*** VI. DELETE(D) DATA AND SWAP**

***** 1. How to delete files in a secure way<**

When a file is deleted, only the inode data is freed, the contents of the data is NOT wiped and can be gathered with tools like "dd" or the tool manipulate_data from THC.

Peter Gutmann wrote a paper with the name "Secure Deletion of Data from Magnetic and Solid-State Memory" presented 1996 at the 6th Usenix Security Symposium. This is the best civilian paper on how to wipe data in a way that it is hard for even electronic microscopes to regain the data.

There are four tools out there which uses the techniques described there, two called "wipe", one called "srm" from THC's secure_delete package and "shred" which is part of the new fileutil package from GNU.

Ours is still the best from it's design, features and security, and it has also all important and advanced commandline options and speed you need.

To use one of these tools for deletion just set an alias in /etc/profile:

```
alias rm=srm      # or wipe or shred
```

or even better, move /bin/rm to /bin/rm.orig and copy the secure delete program to /bin/rm. This ensures, that all data which is deleted via rm is securely wiped.

If you can't install THC's secure_delete package or any other (for any reason) you can also set the wipe flag from the ext2 filesystem on files you wish to wipe before rm'ing them. It's nearly the same, but it's NOT a secure wipe like mentioned above. It's set by:

```
chattr +s filename(s)
```

[Note that it is **still** possible for a well funded organisation to get your data. Don't rely on this! See section [VI.4](#) !]

*** 2. How to wipe free disk space

Most times applications like the editor in your mail program write a temporary file. And you don't know about it - you weren't even asked :(Because they don't wipe the data in a secure way, an attacker can get all your private emails just because you didn't know. That's bad.

The solution: You use a wiper program which cleans all unused data from the disk partitions. The only one available is the one from THC's secure_delete package. You could put "sfill" (that is what it is called) in you crontab so it is run regulary but this might create problems when at this moment this space is needed by an important application. At least when the system shuts down, sfill should be called.

Put this in the "stop" part of a late rc2.d script:

```
sfill -llf /tmp 2> /dev/null
sfill -llf /var/spool 2> /dev/null
```

Note that it is a good idea to generate a new parition for /tmp itself, and putting a symlink from /usr/tmp and /var/tmp to /tmp. This way it is easier to control and wipe.

Again, if you can't install the secure_delete package for any reason, you can also use this solution (slower and not as secure):

```
dd if=/dev/zero of=/tmp/cleanup
sync
rm /tmp/cleanup
```

*** 3. How to handle swap data

Securely wiping files and free disk space - well what's left? Today, harddisk MB's are cheaper than RAM, thats why swap space is used to expand the available RAM. This is in reality a file or partition on your harddisk. And can have your sensitive data in it.

Again there is only one tool which helps you out here, "sswap" from THC's secure_delete package ;-)

Put this line after the "swapoff" line in /sbin/init.d/halt:

```
sswap -l /dev/XXXX # the device for your swap, check /etc/fstab
```

*** 4. How to handle RAM

In section [V.5](#) I wrote about sensitive information in your RAM, the fast memory of your computer system. It can hold very sensitive information like the email you wrote before pgp'ing it, passwords, anything.

To ensure, that the memory is cleaned, use the smem utility.

It should be called like this in the stop part of a late rc2.d script (as already mentioned above), after the wiping the file of /tmp etc. and then wiping the free memory:

```
smem -ll
```

*** 5. Temporary data - it is evil

After you have secured/anonymized/privatized your system so far everything's ready - or did you forget something?

Remember what we told you in section [VI.1](#), that temporary data is written somewhere and sometimes you don't know. If you are unlucky, all we've done here was useless. We have to ensure that there's no temporary data left on the devices and that it can't be recovered either. We already dealt with /var/log, /var/run and sent email (/var/spool/...), and we wipe all free disk space from our temporary disk locations. Now we must wipe also the temporary data.

Put this line in the stop part of a late rc2.d script (before sfill from [VI.3](#)):

```
( cd /tmp ; ls -A | xargs -n 250 srm -r ; )
```

Also a \$USER/tmp directory should be created for all users under the CFS /home protection and a TMPDIR variable set to this directory.

See section IX. for all these scripts ...

* VII. NETWORK CONNECTIONS

This is a very specialized area of this document. I write here a few ways how someone can protect some of their data being transferred on the internet.

The basic prerequisites are as following: You've got an external POP3 and SMTP (mail relay) where you get and send your email. When your go on irc, you also don't like your real hostname being printed on the channels.

Your external mail server should be in another country, because if maybe some official agencies think you're doing something illegal (and I'm sure you won't) it's harder to get a search warrant. It's also harder because companies or individuals that try to get your data would need to invest more time, work and money to get it.

You can tunnel your SMTP and POP3 via ssh to the external mail server. For POP3 this is easy, but for SMTP this is a bit harder.

Just as an example, irc traffic can be tunneled through this as well, but dcc stuff won't work (one way doesn't work, the other would reveal your ip address to the sender and the data is not encrypted on any part of the internet)

Note that you can also use redirectors and proxies to accomplish further redirecting for other protocols (www, irc, ftp proxies etc.)

Thats all. All mail traffic (and as you can see below, irc traffic too) is being crypted between you and your mail/proxy server.

sendmail.cf (important parts):

```
DSsmtp:[127.0.0.1]
DjTHE_DOMAIN_NAME_OF_YOUR_EMAIL
DMTHE_DOMAIN_NAME_OF_YOUR_EMAIL
- Msmtpl,          P=[IPC], F=mDFMuX, S=11/31, R=21, E=\r\n, L=990,
+ Msmtpl,          P=[IPC], F=mDFMuXk, S=11/31, R=21, E=\r\n, L=990,
(add the "k" switch to the smtp option config line)
```

~user/.fetchmailrc:

```
poll localhost protocol POP3:
    user USER_REMOTE with pass PASSWORD_REMOTE is USER_LOCAL here
    mda "/usr/sbin/sendmail -oem USER_LOCAL"
```

(enter the corresponding USER_* and PASSWORD in here)

The ssh commandline which tunnels the traffic for POP3, SMTP and irc:

```
ssh -a -f -x -L 110:localhost:110 -L 6667:irc.server.com:6667 -L \
25:localhost:25 your_mail_server.com
```

That's all. I won't tell you more. Use your brain ;-)

* VIII. HIDING PRIVACY SETTINGS

*** 1. Mount is your friend

Take a look at the following commands:

```
# ls -l /home
total 3
drwxr-x---  1 root    root      1024 Mar 28 14:53 admin
drwxr-x---  1 vh      thc        1024 Mar 28 16:22 vh
drwxr-x---  1 user    users     1024 Mar 28 11:22 user
# mount -t ext2 /dev/hda11 /home      # or a ramdisk, doesn't matter
# ls -l /home
total 0
# : whoops, where are the homedirs ?
# umount /home
# ls -al /home
total 3
drwxr-x---  1 root    root      1024 Mar 28 14:53 admin
drwxr-x---  1 vh      thc        1024 Mar 28 16:22 vh
drwxr-x---  1 user    users     1024 Mar 28 11:22 user
# : ah, yeah there they are again ...
```

This is a nice feature to hide your crypted data and binaries. Just put your files into e.g. /usr/local/bin and /usr/local/crypt and mount a decoy filesystem over /usr/local. If you then have got a process started in your boot scripts which opens a file on the decoy filesystem, the filesystem can't be unmounted until the process is killed. This way, it's much harder for someone to detect your data!

*** 2. Removable Medias

An even better possibility is: put all your sensitive data on a removable media. Put your media in, mount it, it run the startscript from it to activate all the privacy stuff. This way you made it one step harder for someone to get to know whats going on.

*** 3. ???

Any other ideas? Think about it! (and maybe send me your ideas ;-)

* IX. EXAMPLE CONFIGURATION AND SCRIPTS

[Click here](#) to download the anonymous-unix-0.9.tar.gz tools!

* X. FINAL COMMENTS

*** 1. Where to get the tools mentioned in this text

- Crypto Filesystems

CFS (Cryptographic File System) <http://www.replay.com>

TCFS (Transparent CFS) <ftp://mikonos.dia.unisa.it/pub/tcfs/>

SFS (Stegano File System) <http://www.linux-security.org/sfs>

Crypto Loopback Filesystem

<ftp://ftp.csua.berkeley.edu/pub/cypherpunks/filesystems/linux/>

- Tools

THC's secure_delete package <http://www.thc.org>

secure-linux kernel patch <http://www.false.com/security>

syslog-ng <http://www.balabit.hu/products/syslog-ng.htm>

ssyslog <http://www.core-sdi.com/ssyslog>

- The example Linux Distribution

SuSE Linux Distribution <http://www.suse.com>

*** 2. Additional thoughts

The following problems are still present:

- If an attacker can gain access to the system without rebooting and in time before data is wiped, unmounted, etc. these countermeasures are worthless.

- If a really well funded organisation is trying to decrypt your data via brute force/dictionary or good electronic microscopes and technical staff with excellent knowhow, your wiping won't help you very much.

- The solution for /var/spool/mail and /var/spool/mqueue etc. is far away from being perfect. Remember this. Ideas welcome.

- The configuration of your system daemons can only be secured if you are present at the console after a reboot. That's the price.

- It is not very hard to detect the privacy stuff done. This might bring you in trouble in countries like China or Iran. Removable medias might help, or try a crypto filesystem with stegano support.

Secure your system against unauthorized (from your point of view) access and use strong passwords.

*** 3. Greetings (what would the world be without greets?)

What would the world be without love and greetings? ;-)

Greets to individuals (in alphabetic order):

Doc Holiday, Froody, Fyodor, plasmoid, pragmatic, rookie, Solar Designer, Tick, Wilkins.

Greets to groups:

ADM, THC (of course ;-)) and arF

Greets to channel members:

#bluebox, #hack, #hax, #!adm and #ccc

*** 4. How to contact me for updates or comments

Please send me any further ideas you've got to make this documentation better! Did I wrote bad bad english in some part? Could I rephrase parts to make it easier to understand? What is wrong? What's missing? [van Hauser / THC - \[The Hacker's Choice\]](#)

THC's Webpage -> <http://www.thc.org>
(or <http://thc.pimmel.com> or <http://www.thc.org>)

```
Type Bits/KeyID      Date      User ID
pub  2048/CDD6A571  1998/04/27  van Hauser / THC <vh@reptile.rug.ac.be>
```

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: 2.6.3i
```

```
mQENAzVE0A4AAAEIAOzKPhKBDFDyeTvMKQ1xx6781tEdIYgrkrsUeL6VoJ8H8CIU
SeXDUCvU3JlMKITD6nPMFJ/DT0iKHgnHUZGdCQEk/blYHUYOciGlDPGsg3WeTX7L
XLlM4DwqDvPz5QUQ+U+VHuNOUzgxfcjhHsjJj2qorVZ/T5x4k3U960CMJ11eOVNC
meD/+c6a2FfLzJG0sJ/kIZ9HUKY/dvXDInOJaalQclmYjKvfcPsSzas4ddiXiDyc
QcKX+HAXIdmT7bjq5+JS6yspnBvIZC55tB7ci2axTjwpkdzJBZIkCoBlWsDXNwyq
s70Lo3H9dcaNt4ubz5OMVIvJHFMCEtIGS83WpXEABRG0J3ZhbiBIYXVzZXIgLjBU
SEMgPHZoQHJlchRpbGUucnVnLmFjLmJlPokAlQMFEDVE0D7Kb9wCOxiMfQEBvPAD
/3UCDgJs1CNg/zpLhRuUBLYsZ1kimb9cbB/ufLlI4lYM5WMYw+YfGN0p02oY4pVn
CQN6ca50sqeXHWfn7LxBT3lXEPcckd+vb9LPPCzuDPS/zYnOkUXgUQdPo69B04d1
C9C1YXcZjplYso2q3NYnuc0lu7WVD0qT52snNUdkd19ciQEVAwUQNUTQDhLSBkvN
1qVxAQGRTwgA05OmurXHVBvFcvDaBRMhX6pKbTiVKh8HdJa8IdvuqHOCYFZ2L+xZ
PAQy2WCqeakvss9Xn9I28/PQZ+6TmqWUmG0qgxe5MwkaXWxsZKwRsQ8hH+bcppsZ
2/Q3BxSfPege4PPwFWSajnymsnmhdVvvrvt69grzJDM+iMK0WR33+RvtgjUj+i22X
lpt5hLHufDatQzukMu4R84M1tbGnUCNF0wICrU4U503yCA4DT/1eMoDXI0BQXmM/
Ygk9b02Icy+lw1WPodrWmg4TJhdIgxuYlNLIu6TyqDYxJA/c525cBbdqwoE+YvUI
o7CN/bjN0bKglY/BMTHEK3mpRLLWxVMRYw==
```

```
=MdZX
-----END PGP PUBLIC KEY BLOCK-----
```