Unix malwares
Myth or reality ?

Agenda

1. A bit of history on Unix malwares
2. Why should anyone want to target Unix platforms and users ?
3. A proof of concept “malware framework”
4. Back to the future

A bit of history

- Who was targeted ?
  - Not users but servers: Morris worm, Lion, Ramen
- Why ?
  - Unix servers had more bandwidth and resources, were easier to find and attack
- Nevertheless, users have been threatened:
  - BitchX, trojaned fragroute/dsniff/irssi … and a few months ago infected gnome screensavers packages
  - Breaking news : unreal ircd had its code backdoored on November 2009, it was only discovered in June 2010 ;)

Why ?

- Unix users are more likely to be suspicious, aren't they ?
- There are very few desktops running Linux, BSD, Solaris (did you just say Mac OS X ?)
- But...
  - Who reads source code today ?
  - Did you recently have a look at any tarball and/or package installation scripts or do you treat packaging systems as black boxes ?

Why ?

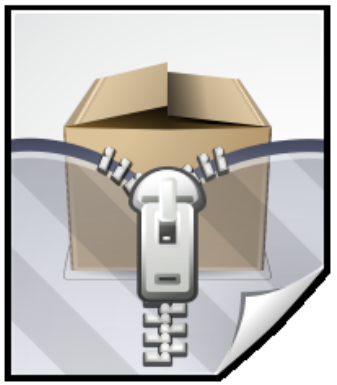
- Most unix “home users” are geeks who have a premium Internet connection and sometimes powerfull boxes.
- User are (too ?) confident:
  - Why would a free software dveloper put some kind of backdoor in his code ?
- Unix users tend to connect to other boxes using SSH:
  - Using keys and/or capturing passwords, succesfull spread is easier.

A proof of concept framework

- Infection vector chosen: packaging systems
  - Packages installation requires root privileges (aptitude/yum/pkgadd/make/whatever)
  - Different ways of doing things: volatile or persistent
  - No security checks done, user is only warned when installing non-PGP signed packages
  - MD5 and/or SHA checksums mismatch doesn't prevent package installation

A proof of concept framework

- Malicious actions
  - Modifying installed packages files
  - Overwrite files from not installed yet packages
  - Trigger malicious code on install/uninstall



Let the fun begin ! :)

A proof of concept framework

- How to choose the package to infect ?
  - Check for software popularity :
    - Freshmeat.net has such graphs
    - Check popcon.debian.org
    - Opensuse has a similar project : popcorn
  - Check for dependency
    - apt-cache dotty apache2 | dot -T png | display
    - debtree
    - rpmgraph

The classical tarball

- Most projects use the autoconf/automake toolchain.
  - Malware can be embedded in the “configure” script or even Makefile.
- The configure script is run with standard user privileges
- “make install” is not :-)
- Remember the fragroute/dsniff/irssi trojaned tarballs

Debian packages

- Many many ways:
  - Targets: preinst and postinst scripts
  - But also prerm and postrm :-)
  - update-rc.d to deploy malicious init scripts
- The pre* and post* scripts are kept inside /var/lib/dpkg/info:
  - Need to cover tracks to prevent the admin from discovering things

Debian packages

- Packages are signed using GPG

- APT tools warn users about packages downloaded from untrusted repositories

  WARNING: untrusted versions of the following packages will be installed!

  Untrusted packages could compromise your system's security.
  You should only proceed with the installation if you are certain that
  this is what you want to do.

  untrusted_package_name

  Do you want to ignore this warning and proceed anyway?
  To continue, enter "Yes"; to abort, enter "No":

- Hopefully there's a way to 'circumvent' this protection :D Google used the trick in its chrome debian package

Debian packages

- Google's trick:
  - Postinst script silently installs a new trusted GPG key

```
# Install the repository signing key (see also:
# http://www.google.com/linuxrepositories/aboutkey.html)
install_key() {
  APT_KEY="`which apt-key 2> /dev/null`"
  if [ -x "$APT_KEY" ]; then
    "$APT_KEY" add - >/dev/null 2>&1 <<KEYDATA
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.2.2 (GNU/Linux)

mQGiBEXwb0YRBADQva2NLpYXxgjNkbuP0LnPoEXruGmvi3XMIxjEUFuGNCP4Rj/a
```

  - Install Chrome, get a free trusted key you haven't heard about !

```
$ sudo apt-key list

mQGiBEXwb0YRBADQva2Npub   1024D/7FAC5991 2007-03-08
uid                  Google, Inc. Linux Package Signing Key <linux-packages-keymaster@google.com>
sub   2048g/C07CB649 2007-03-08
```

RPM packages

- Malware friendly
  - Targets: pre, post, preun and postun
  - But also %verifyscript : use rpm verification command to trigger infection
  - What !? %triggerin an %triggerun: especially useful to prevent HIDS from detecting a malware

BSD

- Softwares are divived into two categories:
  - Base system: the only way to add malware is to compromise an official source repository
  - Ports: there are mostly official mirrors :(hopefully, one can still provide precompiled packages ;)

Mac OS X

- Three different Unix packaging techniques:
  - Fink: debian based
  - Darwin ports: BSD-like
  - Pkgsrc: netbsd-like

Solaris, HP-UX, AIX

- Solaris:
  - packaging system is close to the one used in BSD systems
- HP-UX:
  - less and less used, but SWA seems vulnerable
- AIX:
  - lpp: has preinst, postinst and postun style scripts
  - RPM packages can be installed, even if it is considered as third party

A PoC framework

- The malware injection lightweight framework (MILF) aims at
  - Injecting malicious shellscripts into packages
  - Adding GPG key for unofficials repositories
  - Circumventing host based IDS in the near future ;)

A PoC framework

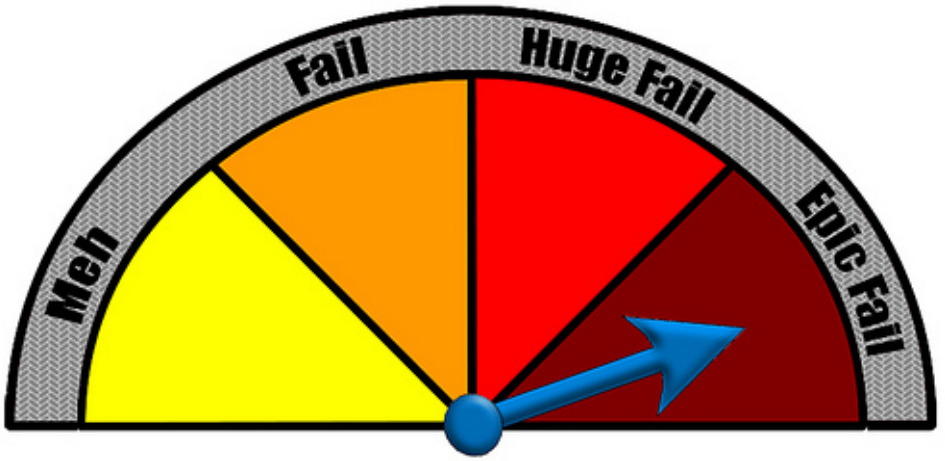
- MILF currently infects the following packaging systems:
  - Debian format
  - RedHat Package Management (WIP)
  - Classical tarballs
- HIDS evasion should be done through:
  - The use of encoded scripts
  - The flaws of softwares such as SSI/rpmshield/whatever

A PoC framework

- A short demo !

Back to the future

- What if packaging systems coders have had security in mind from the beginning ?
- Is there a need for antimalware on Unix ?
  - NEED MOAR PWN2OWN :)))



Questions ?


WARNING
IF THE HELP DESK THINKS YOUR QUESTION IS STUPID, WE WILL SET YOU ON FIRE