

A decorative graphic on the left side of the slide consisting of two overlapping parallelograms. The front one is blue and the back one is light green. They are positioned diagonally, with the blue one partially covering the green one.

# Level 0x04

Diffs and Patches



# Topics

- Events
- Diffs / Patches
- ROM Hacks

# Orlando B-Sides

- [Sunshine CTF / HackUCF](#)
  - 10AM Friday 9/27 - 10AM Sunday
  - Past years have been all skill levels
- [B-Sides Conference](#)
  - Free for students
  - 9AM - 6PM Saturday 9/28
  - See past year videos on youtube. Search "B-Sides Orlando" on youtube
  - 3 talk tracks
  - Villages (lockpicking, rf, career, soldering)



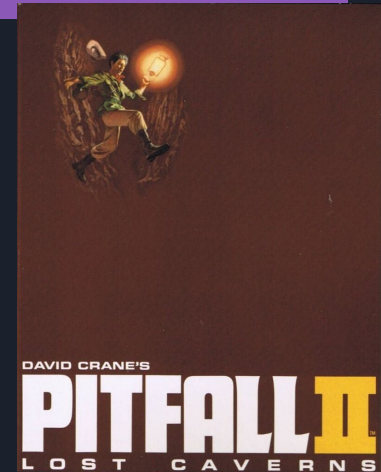
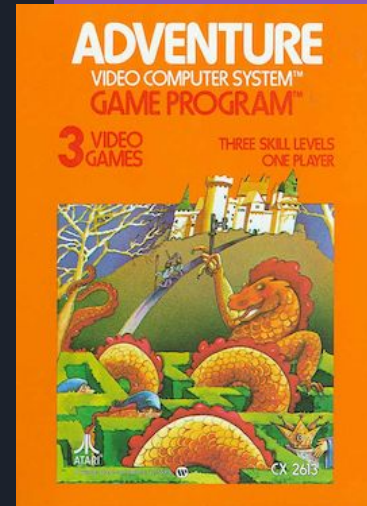
A medium shot of Warren Robinett, a man with grey hair, wearing a yellow and black plaid button-down shirt. He is looking slightly off-camera with a thoughtful expression. The background is a plain, light-colored wall.

**WARREN ROBINETT**  
CREATOR, ADVENTURE

# Easter Eggs

## Warren Robinett

- “First” was in Adventure
  - Not actually the first, but this one popularized the term “Easter Egg”
- Bring a secret dot to a certain room
- Prints programmers name
  - Big no-no at Atari!!
  - Why we have Activision





# Diff

```
mwailes@Metroid:~/scratch/patch_test$ cat hw.c
#include <stdio.h>

int main(int argc, char** argv)
{
    printf("Hello World\n");
    return 0;
}
```

`diff` creates a little snippet of the code change between 2 versions of a file

Can email the diff to maintainer

```
mwailes@Metroid:~/scratch/patch_test$ cat hw_rev2.c
#include <stdio.h>

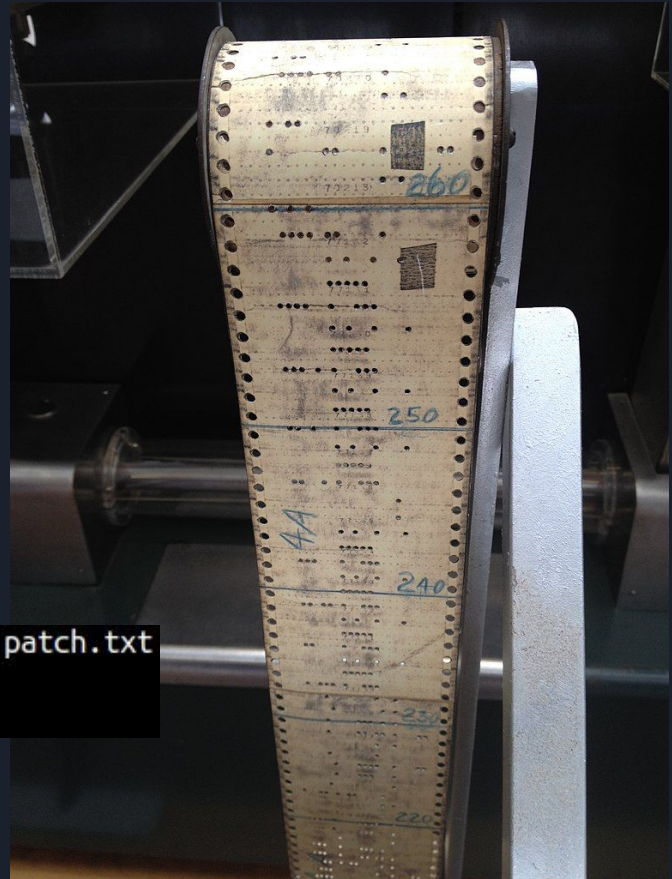
int main(int argc, char** argv)
{
    printf("Hello World\n");
    printf("Goodbye Cruel World\n");
    return 0;
}
```

```
mwailes@Metroid:~/scratch/patch_test$ diff -u hw.c hw_rev2.c
--- hw.c      2024-04-25 01:19:40.909594695 -0400
+++ hw_rev2.c 2024-04-25 01:20:15.006166220 -0400
@@ -3,6 +3,7 @@
 int main(int argc, char** argv)
 {
     printf("Hello World\n");
+    printf("Goodbye Cruel World\n");
     return 0;
 }
```

# Patch

- Apply diff / patch file to update your copy of the code
- “Patch” word comes from “patching” punched holes in paper tape programs

```
mwales@Metroid:~/scratch/patch_test$ diff -u hw.c hw_rev2.c > patch.txt
mwales@Metroid:~/scratch/patch_test$ patch hw.c patch.txt
patching file hw.c
```



# Graphical Diff Tools

- Kompare, meld, Beyond Compare, and Github/Gitlab have graphical diff

Photo gallery will now sort photos alphabetically

master

mwales committed on Apr 9, 2019

1 parent e7bf952 commit diaf7f6

Showing 1 changed file with 2 additions and 1 deletion.

Whitespace Ignore whitespace Split Unified

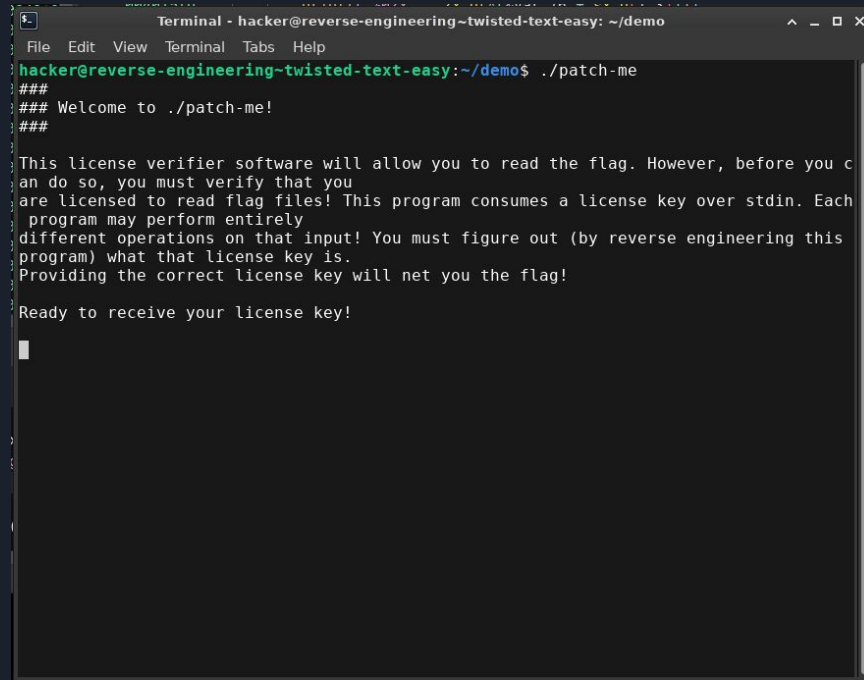
photoGallery/photo\_gallery.py

```
@@ -65,6 +65,7 @@ def isImageFile(filename):
65
66 # Generate the new filename for all of the files and create thumbnails
67 filenameMap = {}
68 + filelist = sorted(filelist)
69 for singleFile in filelist:
70     if os.path.isfile(singleFile) and isImageFile(singleFile):
71         hashedName = processFile(singleFile)
72
73 @@ -83,7 +84,7 @@ def isImageFile(filename):
83 html.write('<html>\n')
84 html.write("&<body>\n\n")
85
86 - for orig in filenameMap:
87     hashname = filenameMap[orig]
88     os.rename(orig, hashname)
89     html.write('  <a href="' + hashname + '">\n')
90
91 + for orig in sorted(filenameMap.keys()):
92     hashname = filenameMap[orig]
93     os.rename(orig, hashname)
94     html.write('  <a href="' + hashname + '">\n')
```



# Patching

- We can also use Binary Ninja to patch binaries
  - We don't have source code, just machine language instructions
- Why?
  - Cracks
  - Change behavior
  - Defeat anti-debugging checks

A terminal window titled "Terminal - hacker@reverse-engineering-twisted-text-easy: ~/demo". The prompt is "hacker@reverse-engineering-twisted-text-easy:~/demo\$". The user has entered the command "./patch-me". The program outputs "### Welcome to ./patch-me!", followed by a detailed license agreement: "This license verifier software will allow you to read the flag. However, before you can do so, you must verify that you are licensed to read flag files! This program consumes a license key over stdin. Each program may perform entirely different operations on that input! You must figure out (by reverse engineering this program) what that license key is. Providing the correct license key will net you the flag!". It then prompts "Ready to receive your license key!".

```
Terminal - hacker@reverse-engineering-twisted-text-easy: ~/demo
File Edit View Terminal Tabs Help
hacker@reverse-engineering-twisted-text-easy:~/demo$ ./patch-me
###
### Welcome to ./patch-me!
###

This license verifier software will allow you to read the flag. However, before you c
an do so, you must verify that you
are licensed to read flag files! This program consumes a license key over stdin. Each
program may perform entirely
different operations on that input! You must figure out (by reverse engineering this
program) what that license key is.
Providing the correct license key will net you the flag!

Ready to receive your license key!
```

# Demo

- Example is a future pwn.college challenge
- Program has a key check
  - User enters secret key
  - User data gets transformed
  - If user data correct, secret flag printed
- Challenge intended for reverse engineering

```
Terminal - hacker@reverse-engineering-twisted-text-easy: ~/demo
File Edit View Terminal Tabs Help
hacker@reverse-engineering-twisted-text-easy:~/demo$ ./patch-me
###
### Welcome to ./patch-me!
###

This license verifier software will allow you to read the flag. However, before you can do so, you must verify that you are licensed to read flag files! This program consumes a license key over stdin. Each program may perform entirely different operations on that input! You must figure out (by reverse engineering this program) what that license key is. Providing the correct license key will net you the flag!

Ready to receive your license key!

let me in
Initial input:

    6c 65 74 20 6d

This challenge is now mangling your input using the `reverse` mangler.

This mangled your input, resulting in:

    6d 20 74 65 6c

The mangling is done! The resulting bytes will be used for the final comparison.

Final result of mangling input:

    6d 20 74 65 6c

Expected result:

    74 64 6d 62 75

Checking the received license key!

Wrong! No flag for you!
```

# Find critical conditional

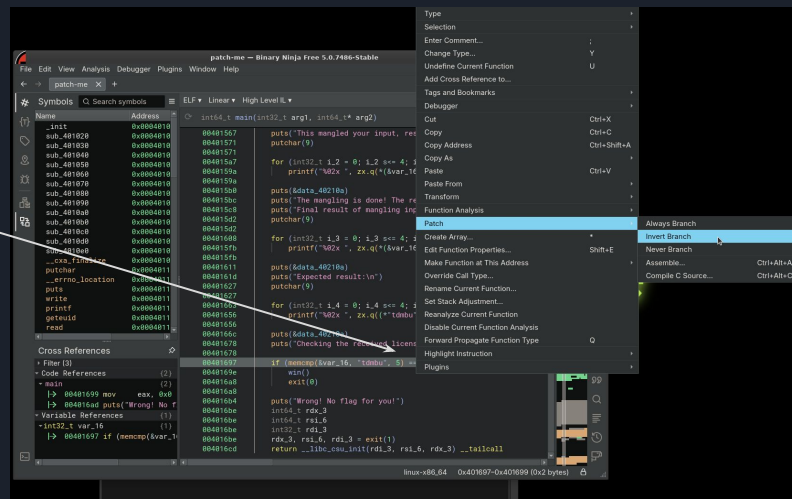
- Find where user input passes / fails
- `memcmp` compares two regions of memory
  - Returns 0 if they are the same
  - Returns non-zero if they are different

The screenshot shows the Binary Ninja interface with a disassembly of a C program. The program prompts for a license key and checks it using `memcmp`. A red arrow points from the list item "memcmp compares two regions of memory" to the `memcmp` call in the disassembly.

```
patch-me - Binary Ninja Free 5.0.7486-Stable
File Edit View Analysis Debugger Plugins Window Help
← → patch-me X +
* Symbols Q Search symbols ELF Linear High Level IL
Name Address
_init 0x0004010
sub_401020 0x0004010
sub_401030 0x0004010
sub_401040 0x0004010
sub_401050 0x0004010
sub_401060 0x0004010
sub_401070 0x0004010
sub_401080 0x0004010
sub_401090 0x0004010
sub_4010a0 0x0004010
sub_4010b0 0x0004010
sub_4010c0 0x0004010
sub_4010d0 0x0004010
sub_4010e0 0x0004010
__cxa_finalize 0x0004010
putchar 0x0004011
__errno_location 0x0004011
puts 0x0004011
write 0x0004011
printf 0x0004011
getuid 0x0004011
read 0x0004011
Cross References
Filter (3)
Code References (2)
main (2)
↳ 00401699 mov eax, 0x0
↳ 004016ad puts("Wrong! No f
Variable References (1)
int32_t var_16 (1)
↳ 00401697 if (memcmp(&var_1
int64_t main(int32_t arg1, int64_t* arg2)
00401567 puts("This mangled your input, result1.")
00401571 putchar(9)
00401571
004015a7 for (int32_t i_2 = 0; i_2 <= 4; i_2 += 1)
004015b0 00 6b fb ff ff 48 8d 3d .H...H.=
004015b8 c4 0d 00 00 e8 5f fb ff .....
004015c8 ff 48 8d 3d 19 0e 00 00 .H=....
004015d8 e8 53 fb ff ff bf 09 00 .S.....
004015e0 00 00 e8 29 fb ff ff c7 .....
004015d8 45 e8 00 00 00 00 eb 24 E.....$
004015e0 8b 45 e8 48 98 0f b6 44 .E.H...d
004015e8 05 f2 0f b6 c8 89 c6 48 .....H
004015f0
00401611 puts(&data_40210a)
0040161d puts("Expected result:\n")
00401627 putchar(9)
00401627
00401663 for (int32_t i_4 = 0; i_4 <= 4; i_4 += 1)
00401656 printf("%02x ", zx.q(("tdmbu")[sx.q(i_4)]), "tdmbu")
00401656
0040166c puts(&data_40210a)
00401678 puts("Checking the received license ke...")
00401678
00401697 if (memcmp(&var_16, "tdmbu", 5) == 0)
0040169e win()
004016a8 exit(0)
004016a8
004016b4 puts("Wrong! No flag for you!")
004016be int64_t rdx_3
004016be int64_t rsi_6
004016be int32_t rdi_3
004016be rdx_3, rsi_6, rdi_3 = exit(1)
004016cd return __libc_csu_init(rdi_3, rsi_6, rdx_3) __tailcall
linux-x86_64 0x401697-0x401699 (0x2 bytes)
```

# Patch with Binary Ninja

- Right-click on conditional
  - The “==” in demo
- Select Patch...
- Select Invert Branch
- Save file!
  - “Save the file contents only”



# Demo

- Now we win!!!
- We didn't really, has to be run as root
- You can't patch this challenge

```
Terminal - hacker@reverse-engineering-twisted-text-easy: ~/demo
File Edit View Terminal Tabs Help
hacker@reverse-engineering-twisted-text-easy:~/demo$ ./patched
###
### Welcome to ./patched!
###

This license verifier software will allow you to read the flag. However, before you can do so, you must verify that you are licensed to read flag files! This program consumes a license key over stdin. Each program may perform entirely different operations on that input! You must figure out (by reverse engineering this program) what that license key is. Providing the correct license key will net you the flag!

Ready to receive your license key!

letmein
Initial input:

6c 65 74 6d 65

This challenge is now mangling your input using the 'reverse' mangler.
This mangled your input, resulting in:

65 6d 74 65 6c

The mangling is done! The resulting bytes will be used for the final comparison.
Final result of mangling input:

65 6d 74 65 6c

Expected result:

74 64 6d 62 75

Checking the received license key!

You win! Here is your flag:

ERROR: Failed to open the flag -- Permission denied!
Your effective user id is not 0!
You must directly run the suid binary in order to have the correct permissions!
```

# Rom Hacks

- Community of people that “patch” old video games
  - Language Translations
  - Difficulty Fixes
  - Gameplay Improvements
  - Complete Overhauls

The Legend of Zelda - Linear Version  
21 March 2025 - Update by Vermin

ROM Hacks - NES



[f Share](#) [t Tweet](#)

This mod is for people who felt the classic legend of zelda too cryptic or annoyed at the thought of using a guide.

More linear than before, this romhack aims to kill the need for a guide, turning the game into a casual experience for new players, reducing the number of enemies and making it easier to discover secrets. Parts of the original map are moved, while maintaining most of the path leading to a dungeon (graveyard at level 6, lost forest at level 7, etc.).

Changes:

- Visible hints for bombable walls and burnable bushes
- Dungeon Automap with item location
- Bonus heart containers
- You can enter all dungeons in order without missing one
- Items in shops cost has been lowered
- Lowered the number of enemies on the first levels



ROMhacking.net

[Home](#) [Forum](#) [Sections](#) [Community](#) [Submissions](#) [Changes](#) [Search](#) [Help](#)

News

Category

Platform

None Selected

Nintendo Entertainment System

Game

Year

None Selected

Search Title

GO

(31 to 40) of 51 Results

Pentris v2  
18 January 2025 - Update by zohassadar

ROM Hacks - NES

[f Share](#) [t Tweet](#)

Pentris includes 18 shapes each consisting of 5 blocks that must be pieced together on a 14x22 playfield with completed rows being cleared. It's like Tetris, but with more frustration. Granular control over the Delayed AutoShift (DAS), Auto Repeat Rate (ARR), and Charge (ARE) settings are now provided in the main patch (formerly a separate Anydas patch).

Version 2.0 adds the following features:

- Menu on level select screen
- Same Piece Sets
- Marathon mode
- Transition mode
- Optional Tetriminos
- Anydas part of main release
- 0-Arr
- Fixed bug that caused overlapping spawn tiles to temporarily disappear

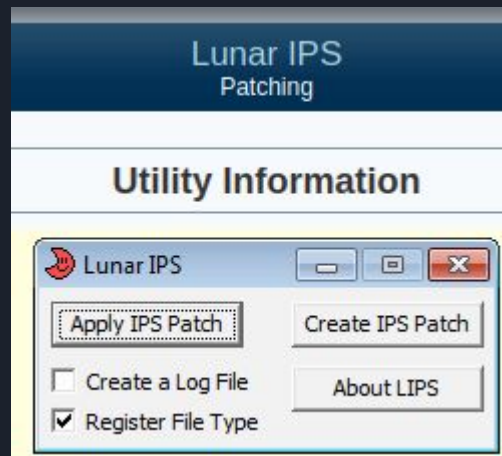
See the [Github page](#) for full details.



[Relevant Link - Project Page](#) [\[ COMMENTS \]](#)

# Lunar IPS

- Patching system for binaries
  - IPS file
- Lunar IPS popular patching tool
- Download IPS file for patch
- “Acquire” proper ROM file



## ROM / ISO Information:

- Database match: Legend of Zelda, The (USA)
- Database: No-Intro: Nintendo Entertainment System (v. 20210216-231042)
- File SHA-1: DAB79C84934F9AA5DB4E7DAD390E5D0C12443FA2
- File CRC32: D7AE93DF
- ROM SHA-1: A12D74C73A0481599A5D832361D168F4737BB CF6
- ROM CRC32: 3FE272FB



# Links

•  
•