A blue parallelogram and a light green parallelogram are positioned on the left side of the slide, overlapping each other and the dark background. The blue shape is on the left, and the green shape is to its right, partially overlapping it.

# Files and Processes

Level 0x02



# Quick Overview

- Upcoming Events
- Shell Commands
- Linux

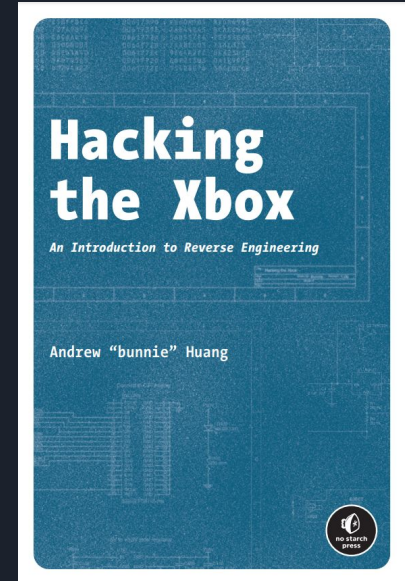
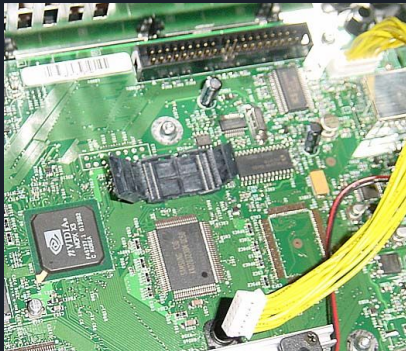


# Hardware Hacking

- Post powder puff
- Group solve Hardware Hacker
  - Arduino / Challenge 3
  - Electronics / Challenge 4

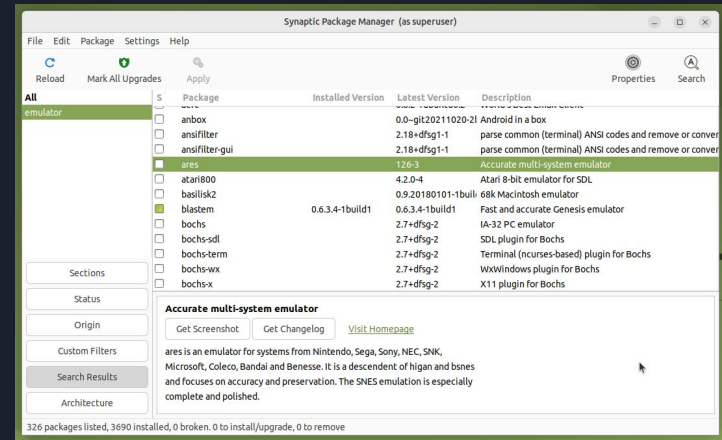
# Andrew “bunnie” Huang

- Hacked the Xbox
  - MIT student, disavowed his project when he published
  - Released his book for free
- Created Chumby
- Writes for Wired, Make magazine
- Wrote book on making hardware in Shenzhen, China



# Packages

Easy to search for and add new software  
(like Steam store, but everything is free)



<code>sudo apt update</code>	Update package lists / versions
<code>sudo apt upgrade</code>	Upgrade installed packages (takes a while)
<code>apt-cache search searchTerm</code>	Searches for packages that match search term
<code>sudo apt install packageName</code>	Installs a new package (and any packages required by that package)
Synaptic Package Manager	GUI for the package manager

# Executable Files

- Linux - permissions bits

- Permission bits for user, then group, then others
- r = read, w = write, x = executable
- `$ ls -l`

```
-rwxrwxr-x 1 mwales mwales 16784 Feb 1 2023 a.out  
-rw-rw-r-- 1 mwales mwales 26 Feb 1 2023 flag.txt  
-rwxrwxr-x 1 mwales mwales 3969 Feb 3 2023 judge.py  
-rwxrwxr-x 1 mwales mwales 330 Feb 2 2023 solution.c
```



- File permissions have 3 groups (from left to right)

- user / Owner
- group
- others

# chmod

- CHange MODe
- Changes the file permissions for a file
- `chmod u+x chal_file`  
`./chal_file`
- Can also assign octal digits  
`chmod 0755 file` ; I can read/write/exec, group and others can read/write
- Change permission for everyone using a for all  
`chmod a+x chal_file`



# What file types executable on Linux

- Just marking file executable doesn't mean it will work

Just that the kernel won't even try to

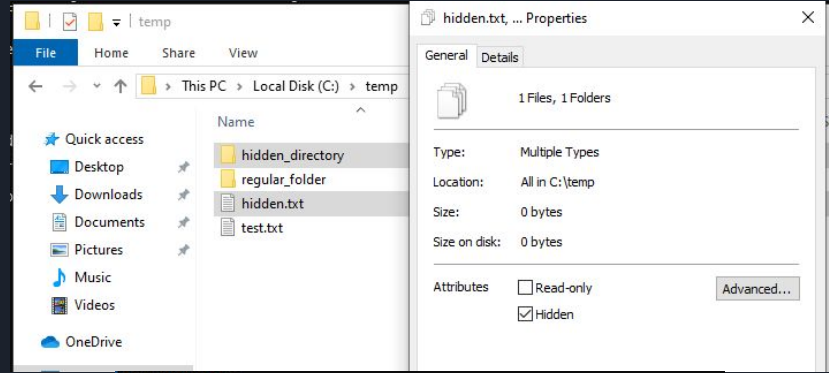
- 2 main types:
  - ELF Binary - compiled / assembly
  - She-bang Line - scripts
- What about Java?

```
mwwales@Metroid: ~  
File Edit View Search Terminal Help  
mwwales@Metroid:~$ hexdump -C ./meme_viewer_v1 | head -n 5  
00000000 7f 45 4c 46 01 01 00 00 00 00 00 00 00 00 00 00 |.ELF.....|  
00000001 03 00 3e 00 00 00 00 00 00 4f 00 00 00 00 00 00 |...>.....0.....|  
00000020 40 00 00 00 00 00 00 00 70 2c 26 00 00 00 00 00 |@.....p,&.....|  
00000030 00 00 00 00 40 00 38 00 0d 00 40 00 22 00 21 00 |....@.8...@.!"!|  
00000040 06 00 00 00 04 00 00 00 40 00 00 00 00 00 00 00 |.....@.....|  
mwwales@Metroid:~$  
mwwales@Metroid:~$ head -n 5 craft_chal.py  
#!/bin/bash  
./encode_chal.py > skibidi_toilet.py  
chmod a+x skibidi_toilet.py  
mwwales@Metroid:~$  
mwwales@Metroid:~$  
mwwales@Metroid:~$ head -n 8 ./skibidi_toilet.py  
#!/usr/bin/env python3  
import base64  
x=""  
a_whole_bun          ch_o  
f_turbulenc          e simp  
vibe_check           pic  
k-me cap re  
dpilled  
mwwales@Metroid:~$
```



# Hidden Files

- Linux hidden files
  - Start filename with a dot / period
  - `ls -a` reveals hidden files
- Typically have user config directories  
In your home folder hidden



```
C:\temp>dir
Volume in drive C has no label.
Volume Serial Number is 961F-BCA3

Directory of C:\temp

09/20/2024 12:38 AM <DIR>      .
09/20/2024 12:38 AM <DIR>      ..
09/20/2024 12:37 AM <DIR>      regular_folder
09/20/2024 12:37 AM              12 test.txt.txt
                        1 File(s)      12 bytes
                        3 Dir(s)  32,542,777,344 bytes free

C:\temp>dir /Ah
Volume in drive C has no label.
Volume Serial Number is 961F-BCA3

Directory of C:\temp

09/20/2024 12:37 AM              0 hidden.txt.txt
09/20/2024 12:38 AM <DIR>      hidden_directory
                        1 File(s)              0 bytes
                        1 Dir(s)  32,542,777,344 bytes free

C:\temp>
```

# Examining Binary Files



- **cat**-ing or viewing binary data in normal text editor usually not fun
- **strings**: show us just the printable sequences of characters
- **file** / **binwalk**: look for magic numbers / bytes in the files
- **hexdump** / **xxd**
- **od**: octal dump 🤔

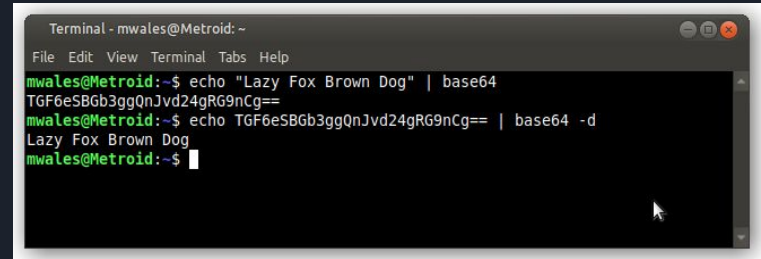
```
mwales@Metroid:~/scratch/hidden_test$ hexdump -C test_file.txt
00000000  4e 6f 74 20 6d 75 63 68  20 74 6f 20 73 65 20  |Not much to see |
00000010  69 6e 20 68 65 72 65 2e  2e 2e 0a                |in here....|
0000001b

mwales@Metroid:~/scratch/hidden_test$ hexdump test_file.txt
00000000  6f4e 2074 756d 6863 7420 206f 6573 2065
00000010  6e69 6820 7265 2e65 2e2e 000a
0000001b

mwales@Metroid:~/scratch/hidden_test$ xxd test_file.txt
00000000: 4e6f 7420 6d75 6368 2074 6f20 7365 6520  Not much to see
00000010: 696e 2068 6572 652e 2e2e 0a                in here....
mwales@Metroid:~/scratch/hidden_test$ od test_file.txt
00000000 067516 020164 072555 064143 072040 020157 062563 020145
00000020 067151 064040 071145 027145 027056 000012
00000033
mwales@Metroid:~/scratch/hidden_test$
```

# base64

- A byte represents 0 - 255.
- Printable ASCII is 32 - 127 (94 characters)
- base64 uses 64 characters (6 bits)
  - A-Z a-z
  - 0-9
  - +/
  - = (padding, it's always at the end)
- You can use base64 to encode binary data into printable text (3 binary bytes -> 4 base64 characters)
- Uses
  - Email attachments
  - Dumping a binary file out via serial shell
  - Simple obfuscation
- `base64 -d` decodes base64 back into binary/ASCII

A terminal window titled "Terminal - mwales@Metroid: ~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The terminal shows the following commands and output:

```
mwales@Metroid:~$ echo "Lazy Fox Brown Dog" | base64
TGF6eSBGb3ggQnJvd24gRG9nKg==
mwales@Metroid:~$ echo TGF6eSBGb3ggQnJvd24gRG9nKg== | base64 -d
Lazy Fox Brown Dog
mwales@Metroid:~$
```

# rot13

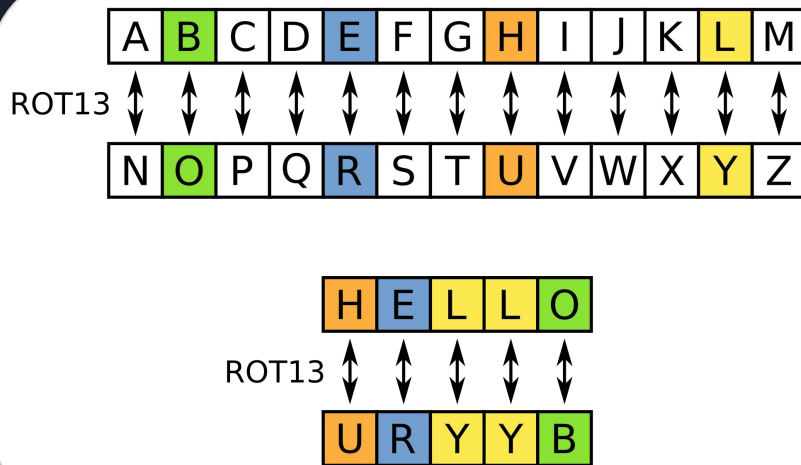
- Simple substitution cipher
- Caesar cipher (invented in Rome)
- Rotate alphabet 3 characters

guvf vf qhzo

this is dumb

irk ones gnat

vex barf tang





# Linux processes

- Can run programs in the background with ampersand (&):  
`gedit filename.txt &`
- List processes running:  
`ps aux`  
`top`  
`htop`
- Kill a running process:  
`kill pid`  
`killall chrome`
- Exit a process that is running:  
`Control-c`
- Built-in program/service to run programs periodically all the time: `cron` / `crontab`



# Attributions

- Benjamin D. Esham for ROT13 picture
- Bunnie Huang info and pics from wikipedia and his website:  
<https://www.bunniestudios.com/>