Packers vs. Compressors

Authored by Jack Ford,

Advised by Professor Kulkarni

Goals and Motivations

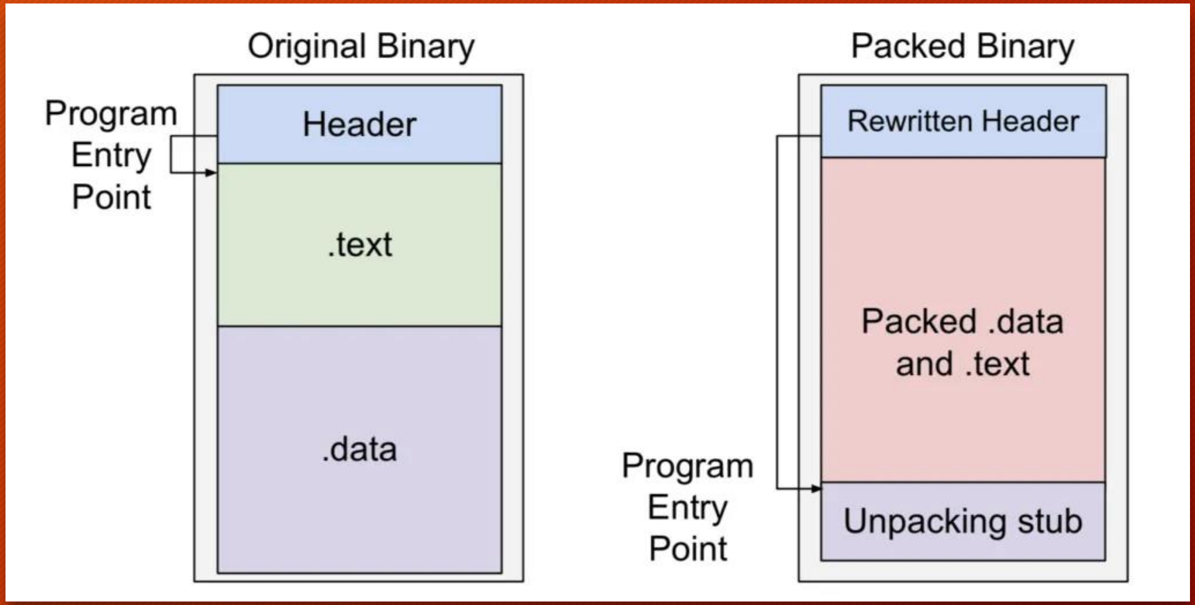
- Obfuscation is a great advantage to malware authors
- Packers are inherently versatile
- STEEL_CORGI
- Compare the algorithms of packers and compression tools

Challenges of packers

	Compression effectiveness	Decompressor size	Decompressor speed
Packers	Binary includes unpacking stub	Must be small – binary includes unpacking stub	Must be simple/fast – unpacking happens at runtime and adds to the program execution time
Compression tools	Decompression program is external to binary	Not much of a concern	Much less concern – decompression is offline, before program execution

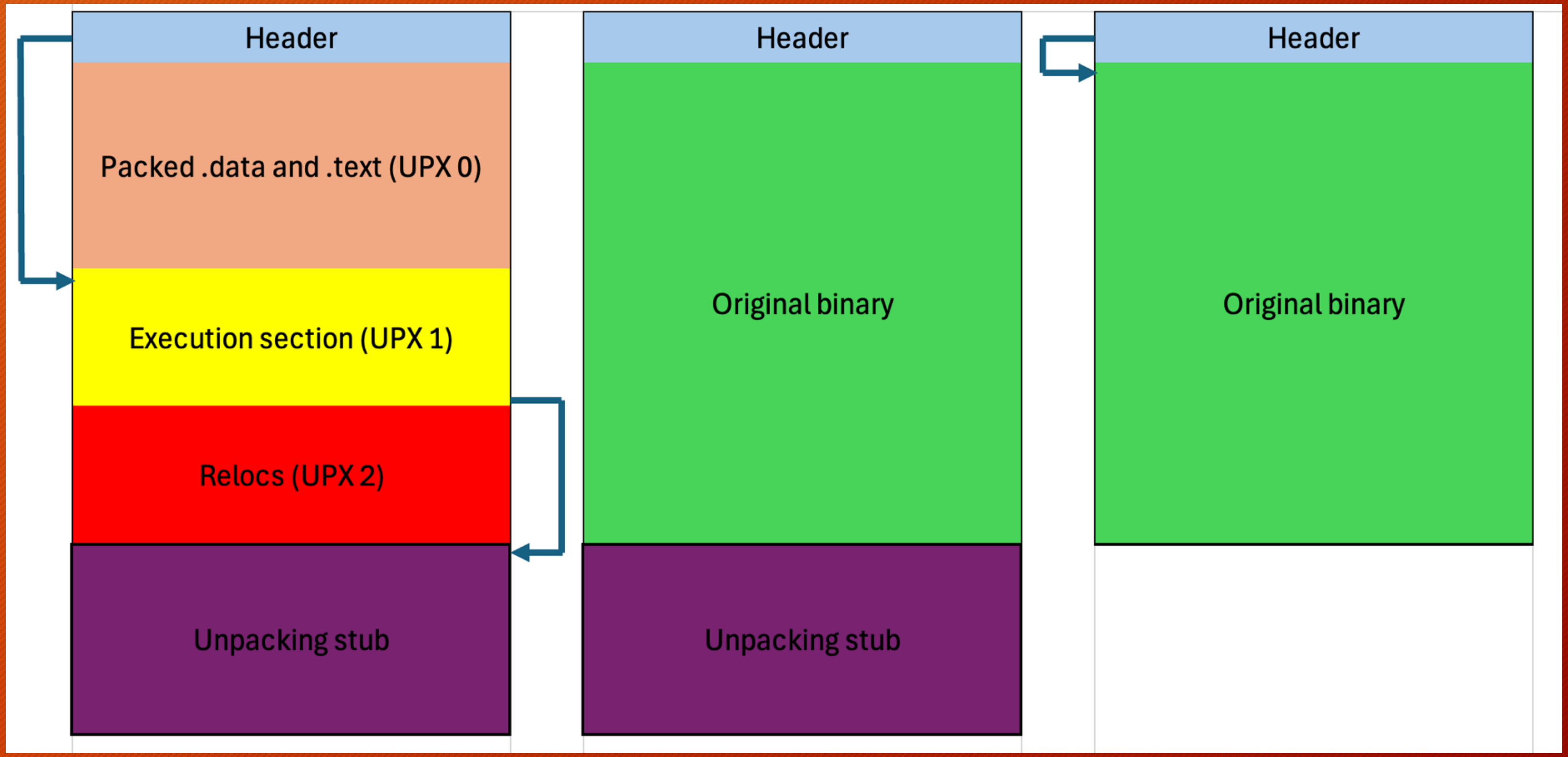
About Packers

- Unique compression technique
- Decrease binary size
- Obfuscation
  - Malware
  - Protecting intellectual property
- Popular packers:
  - UPX
  - Themida
  - ASPack
  - Etc.



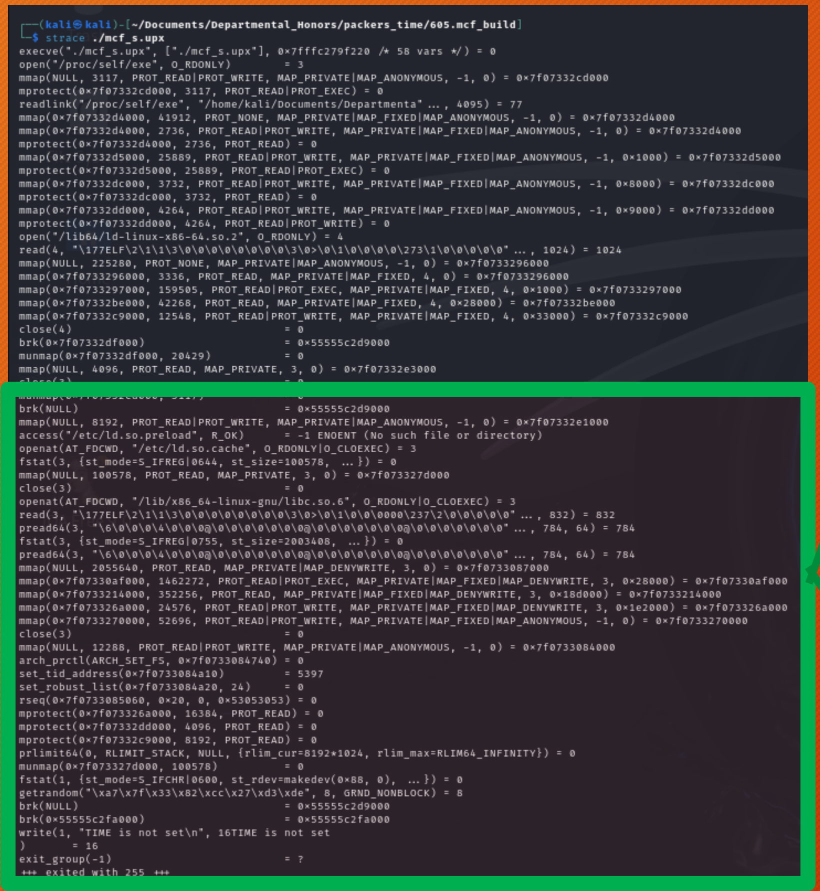
UPX

- 3 Sections
  - Unpacking stub
  - Compressed code
  - Relocatable code
- Code re-writing

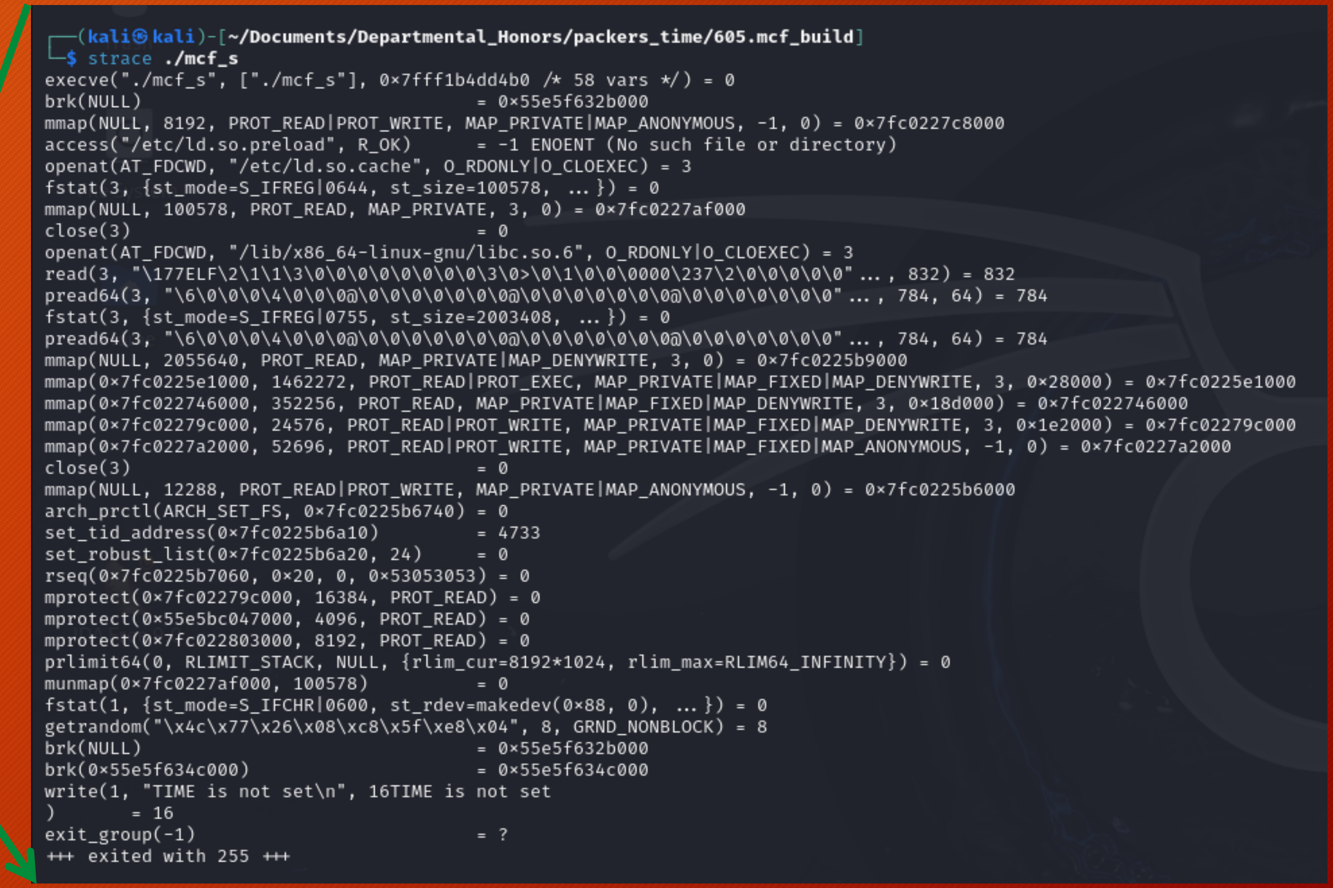


Step 1 Step 2 Step 3

Strace



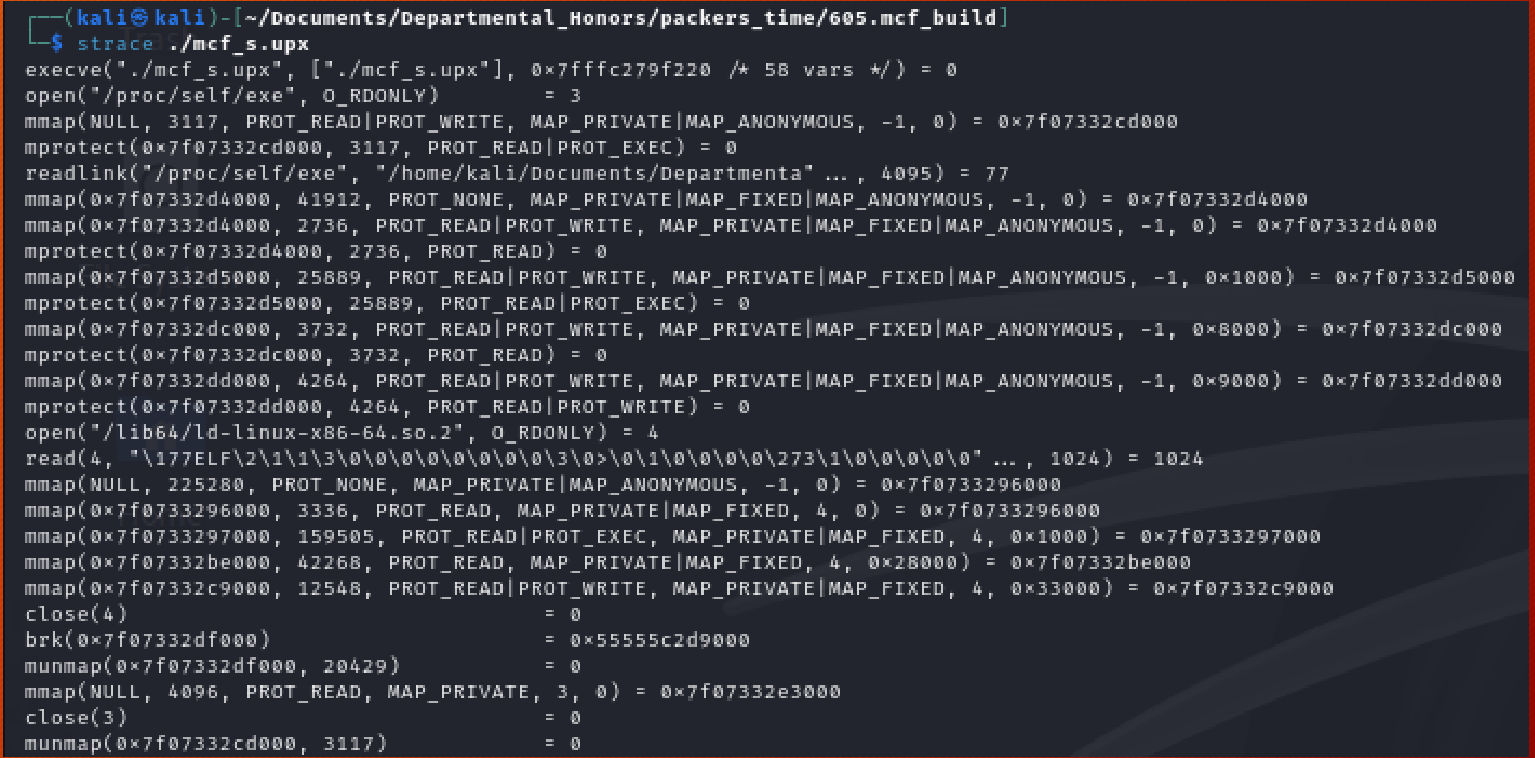
```
┌──(kali㉿kali)-[~/Documents/Departmental_Honors/packers_time/605.mcf_build]
└─$ strace ./mcf_s.upx
execve("./mcf_s.upx", ["./mcf_s.upx"], 0×7fffc279f220 /* 58 vars */) = 0
open("/proc/self/exe", O_RDONLY)        = 3
mmap(NULL, 3117, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0×7f07332cd000
mprotect(0×7f07332cd000, 3117, PROT_READ|PROT_EXEC) = 0
readlink("/proc/self/exe", "/home/kali/Documents/Departmenta" ..., 4095) = 77
mmap(0×7f07332d4000, 41912, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0×7f07332d4000
mmap(0×7f07332d4000, 2736, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0×7f07332d4000
mprotect(0×7f07332d4000, 2736, PROT_READ) = 0
mmap(0×7f07332d5000, 25889, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0×1000) = 0×7f07332d5000
mprotect(0×7f07332d5000, 25889, PROT_READ|PROT_EXEC) = 0
mmap(0×7f07332dc000, 3732, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0×8000) = 0×7f07332dc000
mprotect(0×7f07332dc000, 3732, PROT_READ) = 0
mmap(0×7f07332dd000, 4264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0×9000) = 0×7f07332dd000
mprotect(0×7f07332dd000, 4264, PROT_READ|PROT_WRITE) = 0
open("/lib64/ld-linux-x86-64.so.2", O_RDONLY) = 4
read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\273\1\0\0\0\0\0" ..., 1024) = 1024
mmap(NULL, 225280, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0×7f0733296000
mmap(0×7f0733296000, 3336, PROT_READ, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0×7f0733296000
mmap(0×7f0733297000, 159505, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0×1000) = 0×7f0733297000
mmap(0×7f07332be000, 42268, PROT_READ, MAP_PRIVATE|MAP_FIXED, 4, 0×28000) = 0×7f07332be000
mmap(0×7f07332c9000, 12548, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 4, 0×33000) = 0×7f07332c9000
close(4)                                = 0
brk(0×7f07332df000)                     = 0×55555c2d9000
munmap(0×7f07332df000, 20429)           = 0
mmap(NULL, 4096, PROT_READ, MAP_PRIVATE, 3, 0) = 0×7f07332e3000
close(3)                                = 0
munmap(0×7f07332cd000, 3117)            = 0
brk(NULL)                               = 0×55555c2d9000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0×7f07332e1000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=100578, ...}) = 0
mmap(NULL, 100578, PROT_READ, MAP_PRIVATE, 3, 0) = 0×7f073327d000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000\237\2\0\0\0\0\0" ..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0" ..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2003408, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0" ..., 784, 64) = 784
mmap(NULL, 2055640, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0×7f0733087000
mmap(0×7f07330af000, 1462272, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0×28000) = 0×7f07330af000
mmap(0×7f0733214000, 352256, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0×18d000) = 0×7f0733214000
mmap(0×7f073326a000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0×1e2000) = 0×7f073326a000
mmap(0×7f0733270000, 52696, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0×7f0733270000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0×7f0733084000
arch_prctl(ARCH_SET_FS, 0×7f0733084740) = 0
set_tid_address(0×7f0733084a10)         = 5397
set_robust_list(0×7f0733084a20, 24)     = 0
rseq(0×7f0733085060, 0×20, 0, 0×53053053) = 0
mprotect(0×7f073326a000, 16384, PROT_READ) = 0
mprotect(0×7f07332dd000, 4096, PROT_READ) = 0
mprotect(0×7f07332c9000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0×7f073327d000, 100578)          = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0×88, 0), ...}) = 0
getrandom("\xa7\x7f\x33\x82\xcc\x27\xd3\xde", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0×55555c2d9000
brk(0×55555c2fa000)                     = 0×55555c2fa000
write(1, "TIME is not set\n", 16TIME is not set
)       = 16
exit_group(-1)                          = ?
+++ exited with 255 +++
```



```
┌──(kali㉿kali)-[~/Documents/Departmental_Honors/packers_time/605.mcf_build]
└─$ strace ./mcf_s
execve("./mcf_s", ["./mcf_s"], 0×7fff1b4dd4b0 /* 58 vars */) = 0
brk(NULL)                               = 0×55e5f632b000
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0×7fc0227c8000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or directory)
openat(AT_FDCWD, "/etc/ld.so.cache", O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=100578, ...}) = 0
mmap(NULL, 100578, PROT_READ, MAP_PRIVATE, 3, 0) = 0×7fc0227af000
close(3)                                = 0
openat(AT_FDCWD, "/lib/x86_64-linux-gnu/libc.so.6", O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0000\237\2\0\0\0\0\0" ..., 832) = 832
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0" ..., 784, 64) = 784
fstat(3, {st_mode=S_IFREG|0755, st_size=2003408, ...}) = 0
pread64(3, "\6\0\0\0\4\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0@\0\0\0\0\0\0\0" ..., 784, 64) = 784
mmap(NULL, 2055640, PROT_READ, MAP_PRIVATE|MAP_DENYWRITE, 3, 0) = 0×7fc0225b9000
mmap(0×7fc0225e1000, 1462272, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0×28000) = 0×7fc0225e1000
mmap(0×7fc022746000, 352256, PROT_READ, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0×18d000) = 0×7fc022746000
mmap(0×7fc02279c000, 24576, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_DENYWRITE, 3, 0×1e2000) = 0×7fc02279c000
mmap(0×7fc0227a2000, 52696, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0×7fc0227a2000
close(3)                                = 0
mmap(NULL, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0×7fc0225b6000
arch_prctl(ARCH_SET_FS, 0×7fc0225b6740) = 0
set_tid_address(0×7fc0225b6a10)         = 4733
set_robust_list(0×7fc0225b6a20, 24)     = 0
rseq(0×7fc0225b7060, 0×20, 0, 0×53053053) = 0
mprotect(0×7fc02279c000, 16384, PROT_READ) = 0
mprotect(0×55e5bc047000, 4096, PROT_READ) = 0
mprotect(0×7fc022803000, 8192, PROT_READ) = 0
prlimit64(0, RLIMIT_STACK, NULL, {rlim_cur=8192*1024, rlim_max=RLIM64_INFINITY}) = 0
munmap(0×7fc0227af000, 100578)          = 0
fstat(1, {st_mode=S_IFCHR|0600, st_rdev=makedev(0×88, 0), ...}) = 0
getrandom("\x4c\x77\x26\x08\xc8\x5f\xe8\x04", 8, GRND_NONBLOCK) = 8
brk(NULL)                               = 0×55e5f632b000
brk(0×55e5f634c000)                     = 0×55e5f634c000
write(1, "TIME is not set\n", 16TIME is not set
)       = 16
exit_group(-1)                          = ?
+++ exited with 255 +++
```

Strace (cont.)

- System calls made by the unpacking stub
- Opens itself
- Many memory mappings
- Ends with a memory unmapping



```
┌──(kali㉿kali)-[~/Documents/Departmental_Honors/packers_time/605.mcf_build]
└─$ strace ./mcf_s.upx
execve("./mcf_s.upx", ["./mcf_s.upx"], 0x7fffc279f220 /* 58 vars */) = 0
open("/proc/self/exe", O_RDONLY)        = 3
mmap(NULL, 3117, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f07332cd000
mprotect(0x7f07332cd000, 3117, PROT_READ|PROT_EXEC) = 0
readlink("/proc/self/exe", "/home/kali/Documents/Departmenta"..., 4095) = 77
mmap(0x7f07332d4000, 41912, PROT_NONE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f07332d4000
mmap(0x7f07332d4000, 2736, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0) = 0x7f07332d4000
mprotect(0x7f07332d4000, 2736, PROT_READ) = 0
mmap(0x7f07332d5000, 25889, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0x1000) = 0x7f07332d5000
mprotect(0x7f07332d5000, 25889, PROT_READ|PROT_EXEC) = 0
mmap(0x7f07332dc000, 3732, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0x8000) = 0x7f07332dc000
mprotect(0x7f07332dc000, 3732, PROT_READ) = 0
mmap(0x7f07332dd000, 4264, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0x9000) = 0x7f07332dd000
mprotect(0x7f07332dd000, 4264, PROT_READ|PROT_WRITE) = 0
open("/lib64/ld-linux-x86-64.so.2", O_RDONLY) = 4
read(4, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0>\0\1\0\0\0\0\273\1\0\0\0\0\0"..., 1024) = 1024
mmap(NULL, 225280, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f0733296000
mmap(0x7f0733296000, 3336, PROT_READ, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0x7f0733296000
mmap(0x7f0733297000, 159505, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0x1000) = 0x7f0733297000
mmap(0x7f07332be000, 42268, PROT_READ, MAP_PRIVATE|MAP_FIXED, 4, 0x28000) = 0x7f07332be000
mmap(0x7f07332c9000, 12548, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED, 4, 0x33000) = 0x7f07332c9000
close(4)                                = 0
brk(0x7f07332df000)                     = 0x55555c2d9000
munmap(0x7f07332df000, 20429)           = 0
mmap(NULL, 4096, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f07332e3000
close(3)                                = 0
munmap(0x7f07332cd000, 3117)            = 0
```

Compression Algorithms

- DEFLATE
  - Lossless data compression file format used by ZIP to pack files
  - LZ77
    - Lossless sliding window compression method
  - Huffman Coding
    - Bit-code representation of symbols
    - Most frequent symbols receive smallest bit-code representation
  - Complex combination of the two

Packing Algorithms

- UCL
    - Algorithm used by UPX to pack executables
    - Evolved from LZ77 and other variants of the Lempel-Ziv algorithm
    - LZO = Lempel-Ziv-Oberhumer
        - Named after one of the creators of UPX
    - Open-source re-implementation of some NRV (not really vanished) compression algorithms
    - Shares similarities to LZO
    - Block compressor
    - Improvements on Lempel-Ziv:
        - Greedy parsing
        - Fixed-size blocks
        - No memory for decompression
        - Decompression fits within less than 200 bytes

Research Questions

- How effective are packers in reducing binary size?
- What is the size of the UPX unpacking stub?
- What is the size comparison of unpacked and packed binaries?
- How effective are packers in reducing binary size, compared to compression tools?
- Is unpacking faster compared to decompression?

Methodology

- Size
    - Compressed/packed multiple binaries
    - Evaluated the resulting sizes
- Speed
    - Primarily unpacking/decompression time(s)
    - return 0 as the first line in main
        - Environment variable check
    - hyperfine - a tool for measuring the average execution time for multiple runs of a program

Results

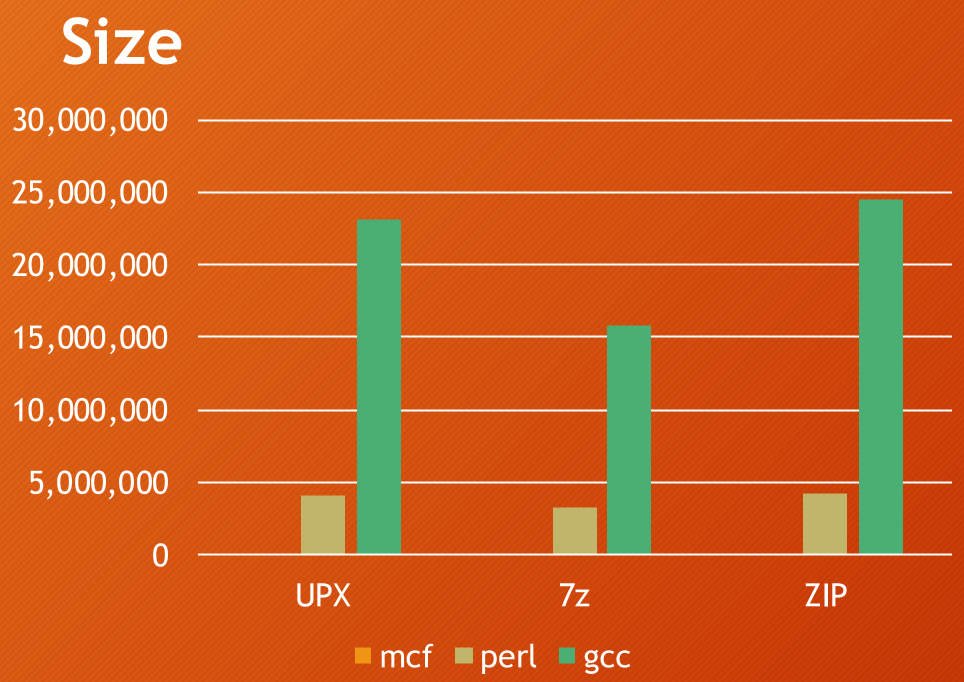
Size

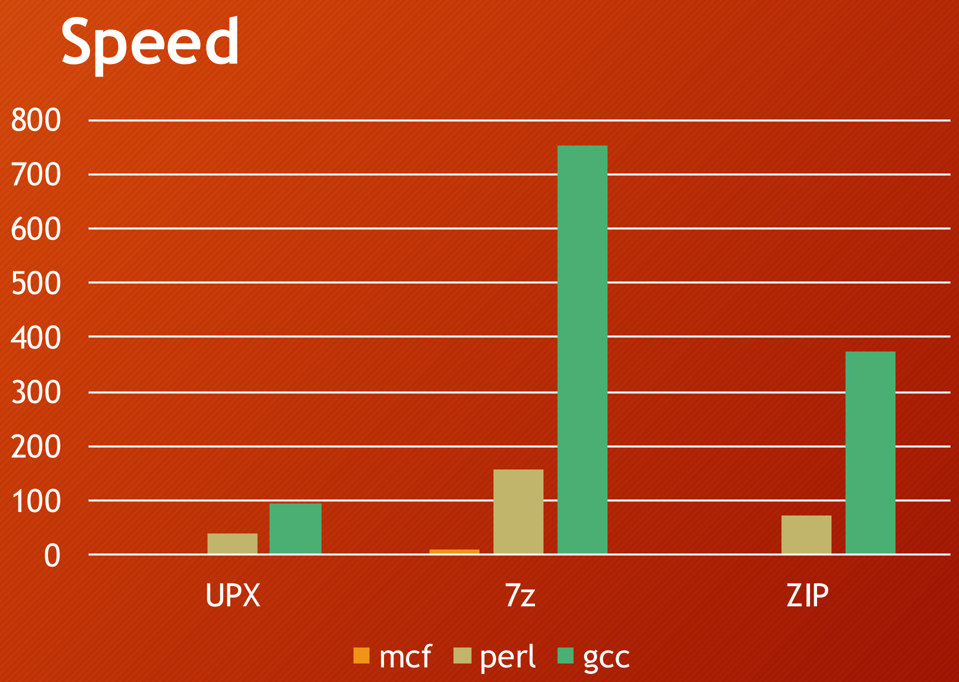
	UPX	7z	ZIP
hello	15,952		
mcf	65,024	51,652	62,876
perl	4,123,040	3,222,591	4,233,593
gcc	23,062,492	15,844,883	24,450,841

Speed

	UPX	7z	ZIP
hello	660.4 μs		
mcf	1.9 ms	8.0 ms	2.7 ms
perl	38.1 ms	157.8 ms	72.1 ms
gcc	94.8 ms	751.8 ms	374.2 ms

Graphs


Size
30,000,000
25,000,000
20,000,000
15,000,000
10,000,000
5,000,000
0
UPX
7z
ZIP
mcf
perl
gcc


Speed
800
700
600
500
400
300
200
100
0
UPX
7z
ZIP
mcf
perl
gcc

Conclusion

- UPX is best at decompression speed on large binaries
  - Sacrifices size for speed
  - Size is comparable to ZIP, but 7z beats both
  - Uses similar algorithm to ZIP
  - Algorithms provide the efficiency
  - Simplistic and refined method of unpacking

Limitations and Future Work

- Analysis of Windows binaries
- Improvements to UPX
  - Less syscalls
  - More efficient memory management
- Analysis and comparison of more packers
  - Different packing goals
  - Mostly Windows tools
  - Payment required
  - Not open-source

Related Work

- https://dl.acm.org/doi/full/10.1145/3530810
  - Malware perspective
  - Different packing techniques
- https://ieeexplore.ieee.org/document/10538977
  - Unpacking process
  - Distinguish between different packers
- https://www.usenix.org/system/files/usenixsecurity23-cheng-binlin.pdf
  - Control flow