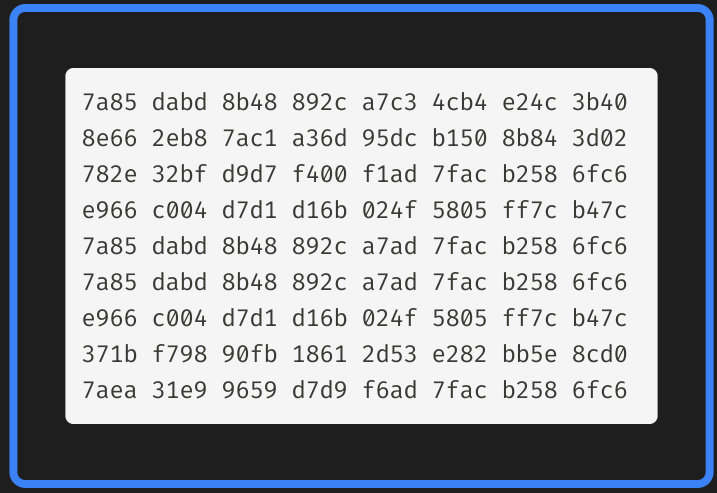
File Formats

Identifying magic bytes, parsing file headers, extracting files from binary.

Start Week 1 →



Welcome to Applied Reverse Engineering

This course will introduce students to the tools and techniques required to analyze the security properties of various systems. Topics covered will include assembly language, executable file formats, operating system internals, and the static/dynamic analysis of compiled binaries. Students will apply these concepts to real-world scenarios like malware analysis and vulnerability analysis with interactive labs, at-home assignments, and a final project.

- **Course Code**: HACS408E
- **Instructors**: Chase Kanipe, Luke Mains
- **Email**: ckanipe@umd.edu, lmains@umd.edu
- **Books**: None required

Course Survey

https://forms.gle/ncP8va5yojNijqzf6

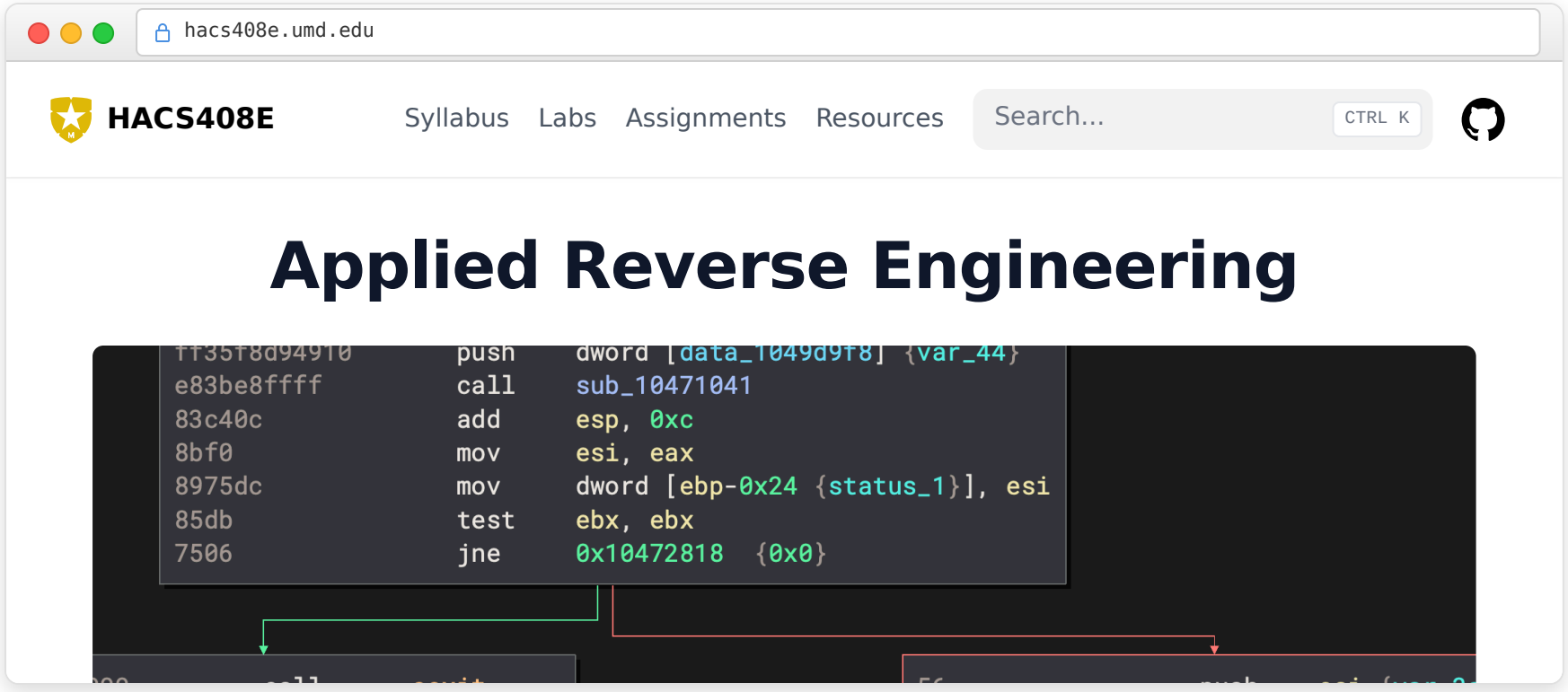
Assignments

This is an *applied* course. The majority of time in-class will be spent on labs and the largest portion of your grade will come from homeworks that test your mastery of the lab material.

Assignment	Weight
Homework	40%
Labs	10%
Quizzes	10%
Team Presentation	20%
Final	20%

Course Website

The syllabus, assignments, and labs, and other resources can be found on the course website. Submissions for labs and homeworks will be on Canvas.



Policies

Authoritative course policies are in the syllabus, but I'd like to highlight a few here.

- Collaboration is allowed on the labs and homeworks, but you should create your own writeups
- Use of AI is allowed except during the quizzes.

What is Reverse Engineering?

Reverse engineering is the process of analyzing an existing system to understand it's inner workings. This course will be primarily focused on *software* reversing with applications to cybersecurity.

Reasons:

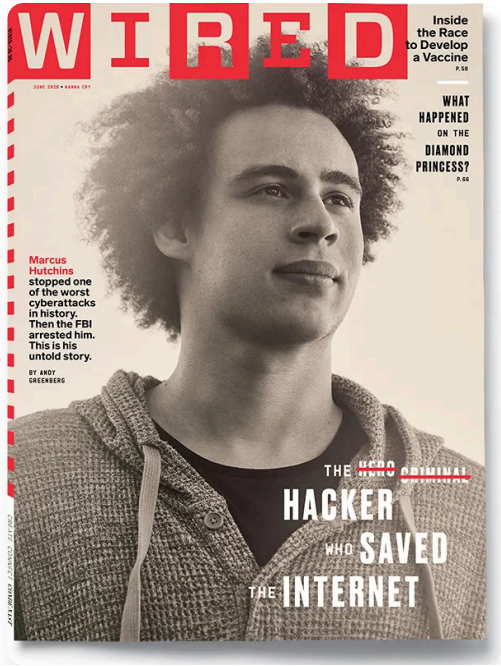
- Malware Analysis
- Vulnerability Analysis and Exploitation
- Interoperability
- ~~Piracy~~
- ~~Intellectual Property Theft~~

In The News: Malware Analysis

In May of 2017, the WannaCry ransomware had infected an estimated 300,000 computers worldwide, encrypting user files and demanding a ransom payment.

British researcher Marcus Hutchins:

- Reverse engineered the worm's artifacts
- Spotted a hard-coded domain that functioned as a kill switch when registered
- Registered the domain, halting the spread
- Arrested for writing malware in his free time

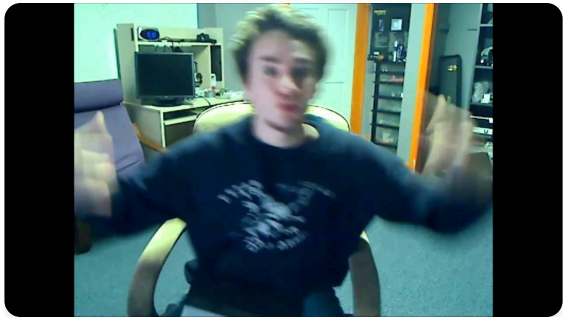


In The News: Interoperability

Console developer go to great lengths to prevent users from running custom software, primarily to prevent piracy.

In December of 2009, George Hotz:

- Began a multi-year project to exploit the Playstation 3
- Found exploits that granted him hypervisor-level code execution
- Published the exploits and a private key
- Sony applied for a temporary restraining order



In The News: Exploitation

Google's Project Zero has some of the best public writeups on exploitation in the wild. One example is their writeup on ForcedEntry, a zero-click iPhone exploit that triggers an integer overflow in Apple's Core Graphics library. The exploit is part of the Pegasus spyware developed by the NSO Group, an Israeli security firm.

Reverse engineering was used by multiple parties:

- NSO Group reversed iPhone internals to find the exploit
- Security researchers reversed the exploit to patch the vulnerability

Exploit development continues to be a high demand skill. By the end of this class you'll be able to understand how these chains work.

Ethics

- Do not attempt to use what you learn in this class to commit illegal acts.
- The techniques taught in this course can be used for multiple purposes
- Use them in a way that complies with U.S. law and university policy

Topics for this course

We don’t have time to cover every aspect of software reverse engineering. These will be our focus.

- Identify and characterize arbitrary files
- Analyze source code and compiled binaries
- Some coverage of Linux and Windows internals
- Network protocols
- Malware Analysis
- Vulnerability Analysis and Exploitation
- Modern Applications and Languages

Jobs that use Reverse Engineering

Since this is an *applied* course, we're focused on skills that are immediately applicable in the workforce. Quiz questions will be inspired by questions I've gotten in interviews.

- Security Researcher
- Malware Analyst
- Forensic Analyst
- Incident Response
- Vulnerability Analyst
- Security Operations Center Analyst
- Computer Network Operations Developer
- **Software Engineer**
- ...

Lab Setup

File Identification

Scenario: You're given a file of an unknown type. How can you identify it to begin your analysis?

- Double click it and see what happens
  - Might not work for all files
  - Could be malicious
- Look for the extension extension (`.exe` , `.pdf` , `.`)
  - Might not have one
  - Malware authors might alter it to make a malicious file appear legitimate
- Thankfully, most file types have a publically documented "magic bytes"
  - These are a signature (usually 4-8 bytes) that identify different file types
  - At the beginning of the file

Identifying Magic Bytes

You can dump the hex of a file using the `xxd` utility.

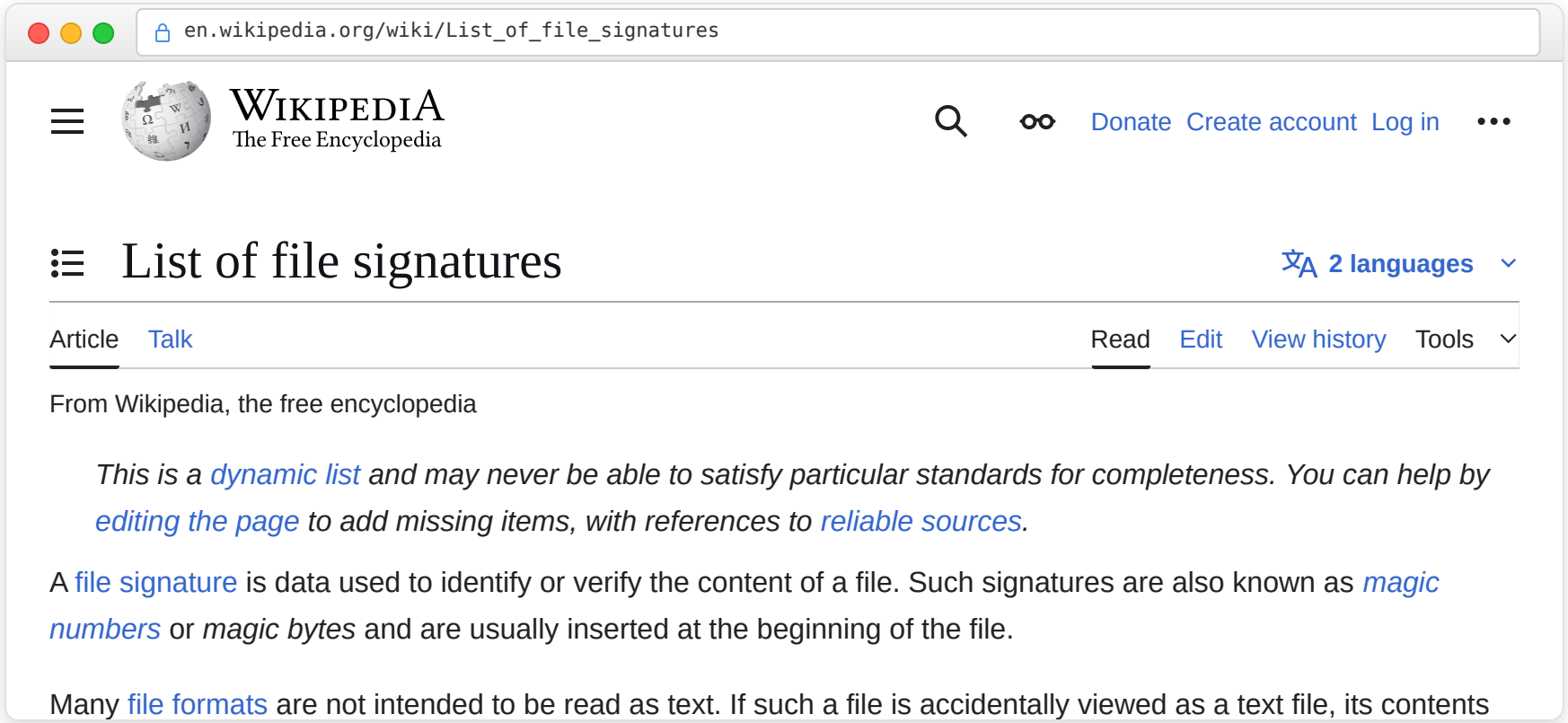
```
chase@Chases-MacBook-Pro ~ % xxd /bin/ls | head
00000000: cafe babe 0000 0002 0100 0007 0000 0003  ................
00000010: 0000 4000 0000 bbf0 0000 000e 0100 000c  ..@.............
00000020: 8000 0002 0001 0000 0001 5c00 0000 000e  ..........\.....
00000030: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000040: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000050: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000060: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000070: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000080: 0000 0000 0000 0000 0000 0000 0000 0000  ................
00000090: 0000 0000 0000 0000 0000 0000 0000 0000  ................
```

Note: Magic bytes aren't infallible. `0×cafebabe` is used to identify both Mach-O binaries and Java classes.

```
chase@Chases-MacBook-Pro ~ % file /bin/ls
/bin/ls: Mach-O universal binary with 2 architectures
```

List of Magic Bytes

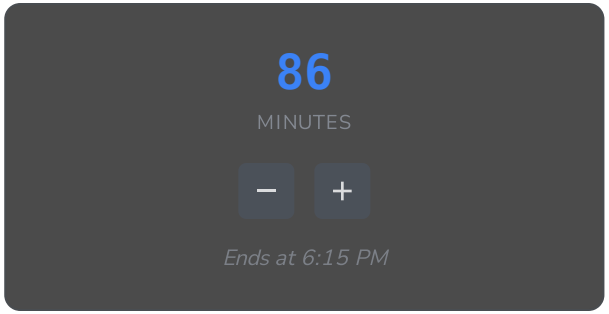
Various internet sources have lists of magic bytes you can search for.



Lab 1

File identification.

https://hacs408e.umd.edu/schedule/week-01/lab-1/



File Carving

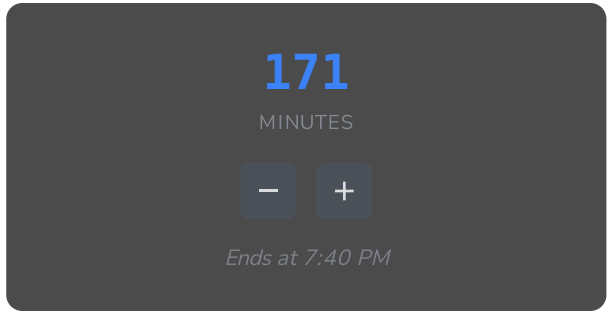
Sometimes a single binary blob will have multiple files embedded in it. We can use `dd` to carve those files from the blob.

```
dd if=example.bin of=output.bin skip=SKIP_BYTES bs=1 count=SIZE
```

```
00000000: AA BB CC DD EE FF 00 11 22 33 44 55 66 77 88 99  ........"3DUfw..
00000010: 00 11 22 33 44 55 66 77 88 99 AA BB CC DD EE FF  .."3DUfw........
00000020: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00001000: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00001010: FF EE DD CC BB AA 99 88 77 66 55 44 33 22 11 00  ........wfUD3"..
00001020: 89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  .PNG......IHDR
00002000: 00 00 02 80 00 00 01 E0 08 06 00 00 00 75 71 3C  .............uq<
00002010: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00002020: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00003000: 4C 00 00 00 04 67 41 4D 41 00 00 B1 8F 0B FC 61  L....gAMA......a
00003010: FF D8 FF E0 00 10 4A 46 49 46 00 01 02 01 00 60  ......JFIF.....`
00003020: 00 60 00 00 FF DB 00 43 00 08 06 06 07 06 05 08  .`.....C........
00004000: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00004010: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00004020: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
00005000: 50 4B 03 04 14 00 06 00 08 00 00 00 21 00 B3 AC  PK..........!...
00005010: 8D 4E 00 00 00 00 00 00 00 00 00 00 08 00 1C 00  .N..............
00005020: 07 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12  ................
```

Lab 2

https://hacs408e.umd.edu/schedule/week-01/lab-2/



Homework

We'll announce the first homework soon. It will be due in 2 weeks.