I have read and understood the policy on academic integrity as outlined in the syllabus for the Spring 2021 Reverse Engineering and Malware Analysis course and in the LSU Code of Student Conduct. In particular, I understand that copying or providing to other students, in whole or in part, solutions (text *or* source code) to class assignments from *any* source (including work done by former and current CS students, other humans, animals, zombies, materials downloaded from the Internet, etc.) not directly sanctioned by Prof. Richard is **not acceptable**. I understand that all work must be exclusively my own, with the exception of any team projects, for which I am allowed to collaborate with my assigned partner(s).

THERE IS NO FLEXIBILITY IN THIS POLICY. IF YOU CHEAT, YOU FAIL, AND YOUR ACADEMIC CAREER IS LIKELY TO BE PREMATURELY TERMINATED. THE "REASON" YOU CHEATED IS IMMATERIAL. This policy applies equally to students "transmitting" or "receiving" answers.

Print your name: ____

Sign your name: ____

Date: ____

No grades will be assigned to your work until you sign and hand in this agreement.

Once you understand the above, please take the following survey (and be honest—this will be used to "tune" the class):

On a scale of 0 ("what is it?") to 10 ("I'm a black belt in this topic, and could kill you with a single glance…") please rate your experience with:

General familiarity with Linux/Unix:	____________
General familiarity with Windows:	____________
General familiarity with Mac OS X:	____________
Operating systems internals (any of the above):	____________
C:	____________
Python:	____________
Intel assembly language:	____________
Malware / reverse engineering:	____________

LOUISIANA STATE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

Reverse Engineering and Malware Analysis Spring 2021 Syllabus Prof. Golden G. Richard III

Me:

Office	☞	Via Zoom only → https://lsu.zoom.us/j/126108881
Email	☞	*golden@cct.lsu.edu*
Office Hours	☞	Monday, Tuesday, Thursday from 10-11:30am or by appointment via https://lsu.zoom.us/j/126108881

You: A student with college credit in operating systems, a strong interest in computer security, and strong programming skills. Detailed knowledge of C and Intel assembler is ultimately required—you can either "know it already" or pick it up as we go, but you will have to work hard.

Meeting: 9:30a-12:20p every Wednesday via Zoom → https://lsu.zoom.us/j/126108881. **You must have video ON and mute your microphone except to speak. If something isn't clear, speak up or use the chat function in Zoom!**

Textbooks: The book listed below is required. You will also need an Intel assembler language reference. You can either use online resources or pick up a book. The Intel reference manuals, copies of which are included are in the in the initial handouts bundle, can serve as an overpowered assembly language reference. You'll also find the "RE for Beginners" PDF in the initial handouts bundle useful. A tablet will be very helpful for using these manuals, as they are organized as **very** large PDF files.

The IDA Pro Book (Eagle, 2nd Edition, ISBN: 978-1-59327-289-0)

You may also find this book interesting—it provides an introduction to an open-source reverse engineering tool called Ghidra (developed by the NSA):

The Ghidra Book: The Definitive Guide (Eagle and Nance, 1st Edition, ISBN: 978-1718501027).

Grading:

Midterm Examination	☞	35%
Final Examination	☞	35% (comprehensive)
Reverse Engineering Assignments	☞	30%

The following grading scale is used. Grading in college courses is objective—please don't ask me to change your grade on an assignment unless you clearly deserve it and can demonstrate that this is the case.

A+	**97-100**	**A**	**93-96**	**A-**	**90-92**
B+	**87-89**	**B**	**83-86**	**B-**	**80-82**
C+	**77-79**	**C**	**73-76**	**C-**	**70-72**
D+	**67-69**	**D**	**63-66**	**D-**	**60-62**
F	**0-59**				

More Details

BACKGROUND: Reverse engineering involves deep analysis of the code, structure, and functionality of software using both static and dynamic methods. This course provides a solid foundation in reverse engineering, which is crucial to understanding modern malicious software and in crafting potential solutions to recover from and prevent attacks. Reverse engineering is also useful for creating interoperable software, for verifying that software patches function as promised, and for the simple joy of understanding at a deep level how software works.

The reason reverse engineering is hard is because one must typically examine complicated software with access only to the executable. This means all that will be available is an assembly language listing, and all of the documentation, most or all of the variable names, etc. will be unavailable. In most cases, you will have to "help" malware analysis tools do the right thing, including manually differentiating code from data, eliminating anti-analysis techniques, etc. As you might expect, malware authors go out of their way to make analysis even more difficult. Many malware instances will deliberately try to make your job harder by detecting that you're using a debugger, detecting that you're running analysis tools in VMWare, employ encryption to prevent you from analyzing code easily, etc. Sound fun? You're in the right class.

ASSIGNMENTS: There will be a number of laboratory assignments in this course. There are dedicated machines in the teaching lab for your use, which you can access remotely. You should consider the due date for each assignment to be a hard deadline. When the due date arrives, turn in what you have—I do give partial credit, but…

NO LATE SUBMISSIONS WILL BE ACCEPTED. ANY SOLUTION SUBMITTED AFTER THE DUE DATE WILL RECEIVE A GRADE OF ZERO.

Submission procedures will be discussed in class.

TESTS: The final examination is comprehensive with an emphasis on material after the midterm. Any missed test will receive a grade of zero unless arrangements are made with me. Both the midterm and final are closed book, closed notes. **The tests will be**

a mix of basic concepts and questions related to your laboratory assignments. Take careful notes when you working on these and study these notes before the examinations. Be prepared to explain techniques you used in the laboratory.

CHEATING: All submitted work must be exclusively your own. Cheating is:

- Copying, in whole or in part, the solutions of former students, current students, or any other living being, alive or dead. "Copying" includes transmission through email, port knocking, the Web, smoke signals, ESP, steganography, or any other means.
- Obtaining solutions from the Internet or other any archival source.
- Looking at a solution is cheating. If you see something that looks like a solution to a class assignment, avert your eyes and *run away as fast as you can.*

As an LSU student, you are obligated to abide by the LSU Code of Student Conduct, which can be found here: http://students.lsu.edu/saa/students/code. Familiarize yourself with the code of conduct, but put simply, **do your own work**. Of course discussing assignments at a high level for clarification, discussing problems concerning the computing equipment, and studying in groups for examinations is not cheating, but when you submit something, it has to be your own work or the work of your assigned team.

STUDENTS WITH DISABILITIES:

LSU is committed to helping students with disabilities. For more information, please see http://students.lsu.edu/disability and make me aware of any issues so we can resolve them.

CLASS MATERIALS: via MOODLE (*http://moodle3.lsu.edu*)

SLIDES: Lecture slides are available via MOODLE. Please try to view the slides online as much as possible and avoid printing them! The remaining trees will love you.

LEARNING OUTCOMES: Upon completion of this course, students will:

- ✓ Have a firm understanding of the legal and ethical issues surrounding reverse engineering efforts
- ✓ Understand the primary motivations for reverse engineering
- ✓ Be familiar with the basic architectures of both historical and modern malware
- ✓ Have a deep understanding of static and dynamic analysis techniques commonly employed to reverse engineer and understand malware
- ✓ Be able to independently reverse engineer malware samples using state-of-the-art tools

Overview of Topics

The following is a tentative list of topics we'll cover in the course. My approach is to introduce new material as needed, interspersed with real reverse assignments (both on your own and collaboratively, as a class). This is an "immersive" approach to learning reverse engineering, which means that the order in which we discuss the topics is flexible.

- Introduction to reverse engineering
- Why is reverse engineering useful?
    - Interoperability
    - Security auditing
    - DRM
    - Analysis of malware
- Legal Issues
    - Under what circumstances is reverse engineering legal?
    - When isn't reverse engineering legal?
- Overview of important foundational knowledge
    - Important assembler languages
    - Popular programming languages for malware development
    - Operating systems internals
    - Hardware
- Overview of modern malicious software
    - Viruses
    - Worms
    - Trojans
    - Ransomware
    - Botnets
- Historical malware
    - MS-DOS malware case studies
- Polymorphic and metamorphic malware
    - Malware detection
    - Worm fingerprinting / signature generation
    - Behavioral approaches for detection of malware
    - Hardware agents for system integrity checking
- Static and dynamic reverse engineering techniques
    - System monitoring tools
    - System call, filesystem, and registry tracing
    - Debuggers
    - Disassemblers

    - Decompilers
- Toolchains and high-level languages
    - Compilation toolchains and impacts on reverse engineering
    - Representation of compiled high-level language structures in assembler
    - Virtual machines for interpreted high-level languages
- Worm case study
    - Infection vectors
    - Target selection
    - Propagation
- Executable file formats
    - Portable Executable (PE) format
    - ELF
    - Mach-O
- Advanced code obfuscation techniques
    - Control flow obfuscation
    - Opaque predicates
    - Arithmetic obfuscation
- Encrypted and packed executables
    - Identifying packed executables
    - Packing strategies
    - Unpacking
- Encrypted malware case studies
    - Infection vector and propagation
    - Obfuscation strategies
    - Unpacking
    - Effects
- Anti-debugging techniques
    - Debugger detection
    - Strategies for blocking debugger access
- Anti-VM techniques
    - Types of virtualization
    - Detection of virtualization
    - Exploiting runtime differences in physical vs. virtualized hardware
- Memory forensics and reverse engineering
    - Memory acquisition
    - Malware detection using memory forensics
    - Dumping process memory for analysis