Concept Map: Malware Analysis & Reverse Engineering

Nodes and labeled relationships:

- Base64 encoding —weak form of→ Obfuscation
- Cryptor —is based on→ Obfuscation
- Packer —is based on→ Obfuscation
- Scylla —useful tool against→ Packer
- VirusTotal —uses→ OSINT
- Compiler —translates programming language into→ Assembly
- Assembly —which translates to→ Machine code
- Assembly —a program into→ Dissasembly
- kernel-mode drivers —also used to debug→ WinDBG
- GDB —can→ Dissasembly
- GDB —you can setup→ Breakpoints
- x64dbg —used to→ Debug
- WinDBG —used to→ Debug
- GDB —used to→ Debug
- CreateRemoteThread() —is run after→ DLLMain()
- Process Injection —uses→ CreateRemoteThread()
- Process Injection —uses→ LoadLibraryA()
- .DLL —injects malicious→ LoadLibraryA()
- LoadLibraryA() —inject malicious→ .SO
- Process Injection —involves creating→ Thread
- Ransomware —often uses→ Symmetric Encryption
- Ransomware —often uses→ Asymmetric Encryption
- Ransomware —uses to encrypt→ file handles
- Hashes —used to find→ Malware
- Ransomware —often employed by→ Malware
- Loader —can start→ Process Injection
- IDA —has a cloud→ Decompiler
- IDA —is popular tool→ Dynamic Analysis
- Static Analysis —used in→ YARA Rules
- Keylogger —type of→ Malware
- Dropper —type of→ Malware
- Worm —type of→ Malware
- Ransomware —type of→ Malware
- Windows Defender —detects→ Malware
- Malware —type of→ Botnet
- Malware —tries to achieve→ Persistence
- Botnet —utilizes→ C2
- RAT —type of→ Malware
- Rootkit —type of→ Malware
- Virus —type of→ Malware
- Device I/O —can interact with→ Rootkit
- PatchGuard —protects against→ Rootkit
- NjRAT —is→ RAT
- NjRAT —uses→ RCE
- Dynamic Analysis —used to analyze→ Malware
- Static Analysis —used to analyze→ Malware
- FLOSS —helpful tool→ Static Analysis
- FLOSS —extracts→ Strings
- Ghidra —is popular tool→ Dynamic Analysis
- Ghidra —developed by→ NSA
- OpSec —used in→ Reverse Engineering
- Reverse Engineering —involves→ Dynamic Analysis
- Reverse Engineering —useful to practice→ Crackme
- Strings —to solve, identify→ Source
- Crackme —to understand the→ Control Flow
- Control Flow —use a→ Control Flow graph
- Source —and→ Sink
- Horacio Gonzalez —enjoys learning→ Reverse Engineering
- Horacio Gonzalez —likes chapter about→ Control Flow Integrity
- Sandbox —has tools like→ Wireshark
- Sandbox —has tools like→ Fakenet
- Sandbox —has tools like→ Regshot
- Sandbox —is created using→ Hypervisor
- Sandbox —should have→ Network Isolation
- Wireshark —used to find→ Indicators of compromise
- Fakenet —used to find→ Indicators of compromise
- Fakenet —uses→ sockets
- Regshot —used to find→ Indicators of compromise
- Hypervisor —used to create→ Virtual Machine
- PEExplorer —to analyze→ .EXE
- .EXE —you will find→ "MZ"
- Regshot —takes→ Snapshot
- Snapshot —of the→ Windows Registry
- Control Flow Hijacking —attack against→ Control Flow Integrity
- DEP —protects→ Control Flow Integrity
- Stack Canaries —protects→ Control Flow Integrity
- Control Flow Hijacking —example attack→ Buffer overflow
- Nop Sled —commonly used in→ Buffer overflow
- pwntools —used to perform→ Buffer overflow
- ASLR —defense system against→ Buffer overflow
- DEP —can get around using→ Use-After-Free
- Stack Canaries —can get around using→ Format String Vulnerability
- Buffer overflow —changes→ Return Instruction
- ROP —overwrites→ Return Instruction
- Return Instruction —becomes a malicious→ Shellcode
- ROP —defeats→ DEP
- Stack Canaries —defeats→ ROP
- DEP —causes→ Segmentation fault
- PEB —with the list→ VAD
- _EPROCESS —compares objects from→ PEB
- Pool Tag Scanning —goes beyond using→ _EPROCESS
- Volatitily —can find→ _EPROCESS
- Memory Forensics —is popular tool→ Volatitily
- Memory Forensics —needs dump of→ RAM
- _EPROCESS —malware hides from using→ DKOM
- 32bit —architecture→ x86
- 64bit —architecture→ ARM
- x86 —is type of CPU
- ARM —is type of CPU
- CPU —has a virtual→ Virtual Machine
- CPU —has→ Registers
- CPU —uses→ Process
- Registers —is one of them→ RIP
- Registers —is one of them→ RBP
- Registers —is one of them→ RSP
- Kernel Space —interact with→ Syscall Calling Conventions
- User Space —interact with→ Stack frames
- User Space —create→ Stack frames
- C Calling Conventions —create→ Stack frames
- Syscall Calling Conventions —creates new→ Process
- C Calling Conventions —creates new→ Process
- Process —is assigned with→ Virtual Address Space
- Procmon —monitors→ Process

Identify the 3 most intellectually interesting things you learned in this course. Why did you choose these 3?

1. Exploring Reverse Engineering with Ghidra and IDA: Understanding how programs and malware works and dissecting its inner workings felt like being a detective, piecing together clues and unraveling mysteries.
2. Learning Python Scripting: I did know how to program in python, so then having the opportunity to learn this language was very cool.
3. Learning how to perform buffer overflow attack: Getting to learn with memory manipulation and exploiting these vulnerabilities gave me a deeper understanding of cybersecurity risks and how to defend against them, making me better equipped to protect systems from malicious attacks.

Identify the 3 most professionally useful things you learned in this course. Why did you choose these 3?

1. Learning how to use IDA, because IDA is very expensive, and the fact that we had the opportunity to learn how to use it without paying a single penny is very nice.
2. To familiarize ourselves with Cybersecurity Tools and Techniques, such as Wireshark, Procmon, Fakenet, and sandbox environments enhances our ability to detect, analyze, and respond to security incidents.
3. Understanding Malware Behavior and Analysis Techniques, analyzing malware samples helps in developing defense strategies, creating signatures for detection systems, and understanding new threats in the cybersecurity world.