Meeting 104

// Reverse Engineering Basics //

If You’re New!

- Join our Slack: cyberatuc.slack.com
- **SIGN IN!** *(Slackbot will post the link in slack)*
- Feel free to get involved with one of our committees:

  Content Finance Public Affairs Outreach Recruitment Lab

Cyber@UC

Announcements / Upcoming Events

- We are currently 5th place in NSA Codebreaker
- 10/30: Rockwell Automation Visit / Demo
- 11/6 - Club Elections

Rockwell Automation Guest Speaker

Wednesday, October 30th - Rhodes 850D Patrick Feeley - Senior Embedded Software Engineer

- An overview of Industrial Control Systems (ICS) and how they relate to cybersecurity
- History of ICS Security (Stuxnet, Ukrainian Power Grid Attacks, Trisys, etc.)
- Challenges in ICS security attacks
- Potentially a DEMO of how security issues in ICS can be exploited to cause physical damage

Bring Resumes and Questions!

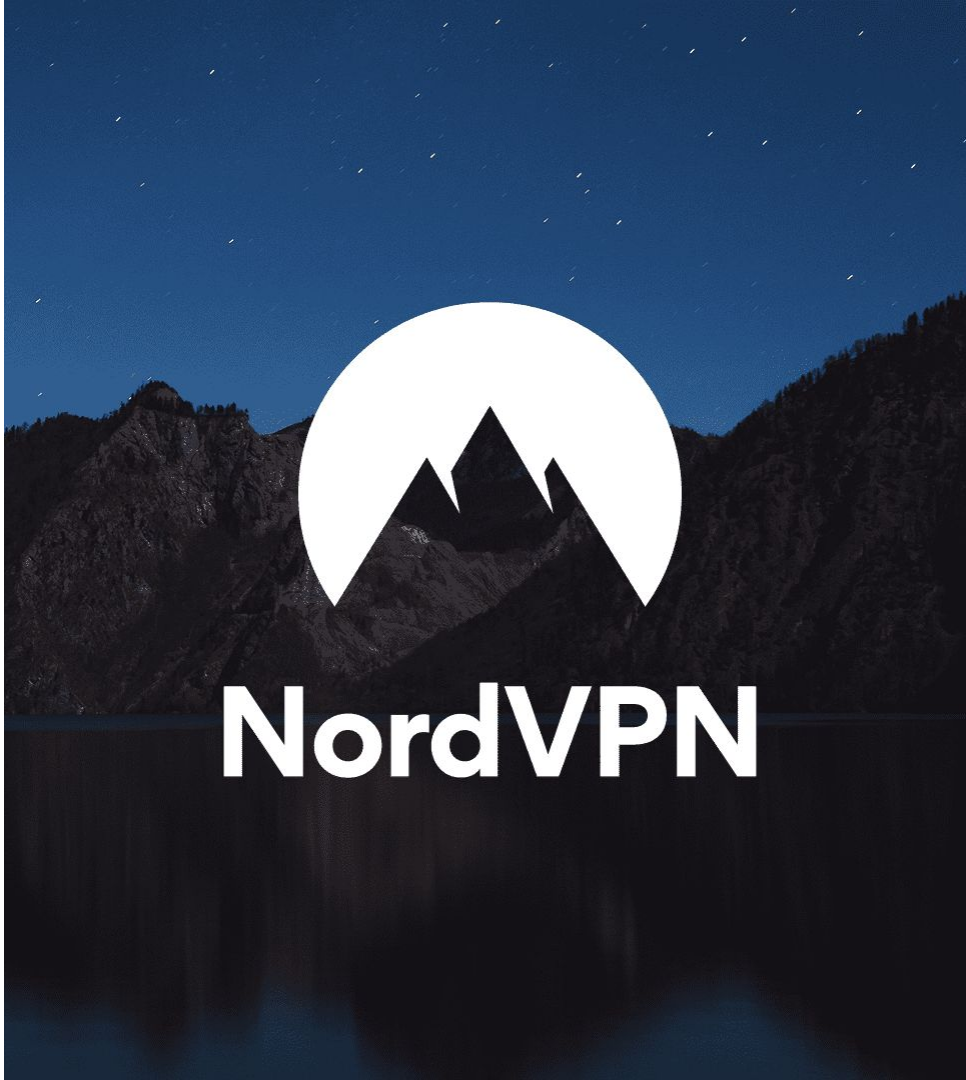
Reach out to **@Michael Sengelmann** on **cyberatuc.slack.com** if you have any questions

Cyber@UC + Rockwell Automation

Weekly News

NordVPN Breach

- Insecure remote management system account
- Attacker could spin up their own server
- 1 of 3000 servers
- Would need the client to ignore the expired key



https://nordvpn.com/blog/official-response-datacenter-breach

Reverse Engineering Basics

Agenda

- I wasn’t here last week!
- What is REeee
- Why REeee
- REeee Tools
- Binary Compilation Process
- CTF Challenge from Battelle
    - Goats walkthrough w/ ghidra

I wasn’t here last week!

- Shame on you
- Jason Armstrong from the NSA came and gave us an incredible talk on the history of Encryption and even brought an original enigma machine for us to play with

What is Reverse Engineering?

- Process of analyzing software to figure out how it works, how it was written, and more
- Typically done with a combination of debuggers, disassemblers, and decompilers
- Static analysis
  - Inspect the program without running it
  - View code, draw conclusions
- Dynamic analysis
  - Inspect the program as it runs

Why Reverse Engineer Things?

- Figure out how things work
- Change how things work by extending them
- Find vulnerabilities

- - - Reverse engineering is used to:

- Make exploits
- Hack video games
- Win CTF's (like CodeBreaker)

Reverse Engineering Tools

Binary Tools (ELF / PE / MachO)	Android / Java Tools
GHIDRA diStorm3 IDA edb-debugger OllyDbg Valgrind YARA Strings R2 / Cutter Binary Ninja	**GHIDRA** apktool dex2jar jad javasnoop jd-gui smali



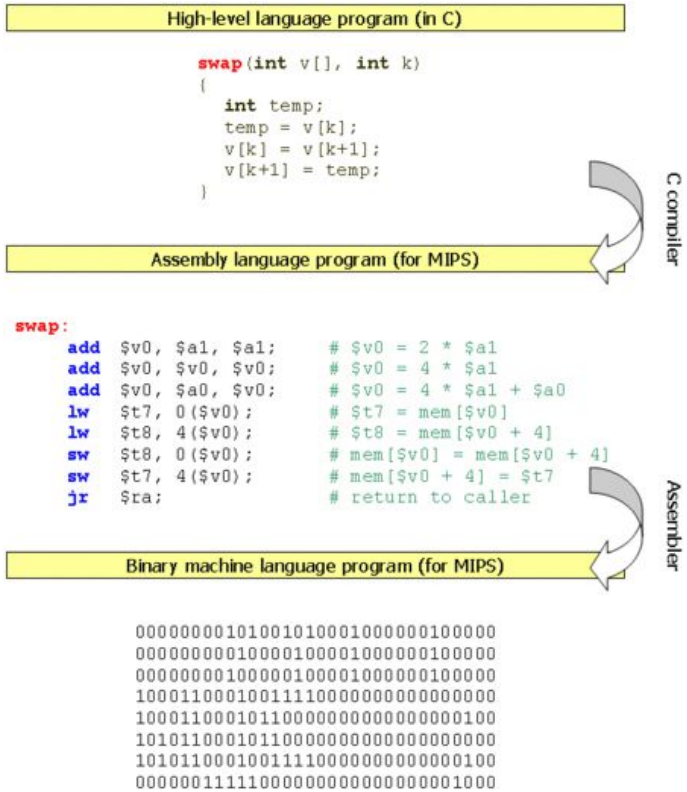



Cyber@UC


GHIDRA

Compilation Process - Executables

- Source code is written in language of choice (here in C)
- Code compiles to assembly / interpreter code
- Native Code (C++, C, Rust) continue being compiled to the actual numbers that the processor runs
- On some scripting languages, compilation may not be done at all as the script is interpreted by the language binary

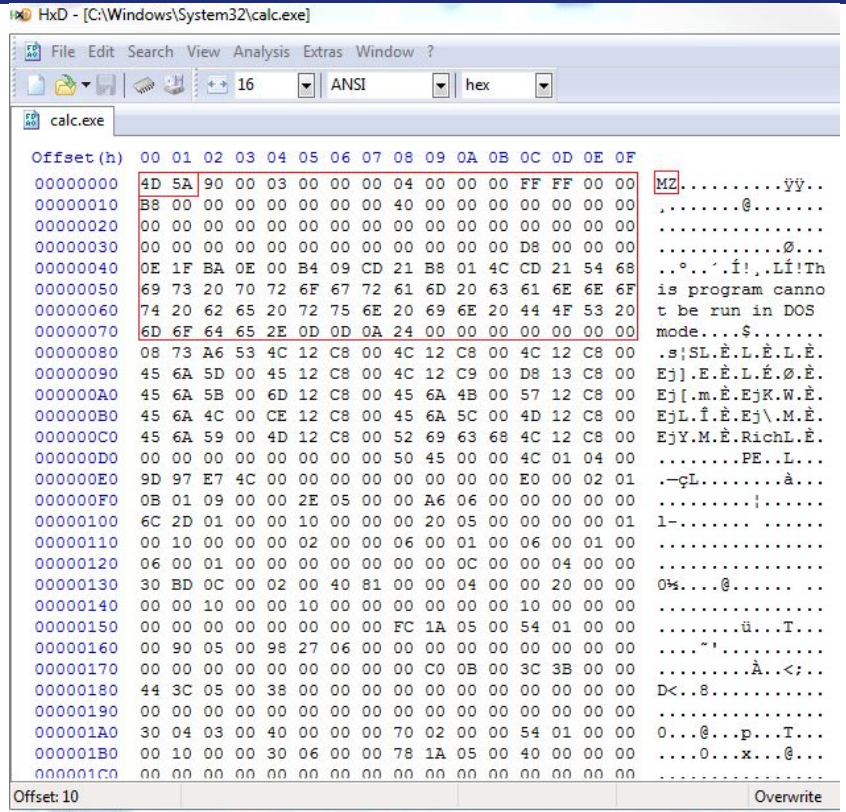


Compilation Process - Dependencies

- Static Linking
  - Dependencies are included in your output binary (.LIB/.O)
  - Pros: much more portable, single output program
  - Con: larger output binary size
- Dynamic Linking
  - Dependencies are looked up by the OS when the program is run (.DLL/.SO)
  - Pro: smaller individual binary size, multiple programs can share deps
  - Cons: Dependencies might not be on machine, more files to track
- Run-Time Linking
  - Like dynamic linking except the program finds the dependencies it wants to load manually and then pulls them into memory
  - Pro: Reverse engineers take slightly longer to see your dependencies
  - Con: Almost exclusively used by malware, so if you're drying to make a video game hard to hack you'll probably be blocked by antivirus from even installing

Compilation Process - Strings

- Typically the strings in your application get shoved into a special section of the binary in ASCII or UTF-16 format
- All modern Windows program include “This program cannot be run in DOS mode” right at the beginning as well
- We can scan a binary for plain text strings in a matter of nanoseconds



Cyber@UC


BATTELLE