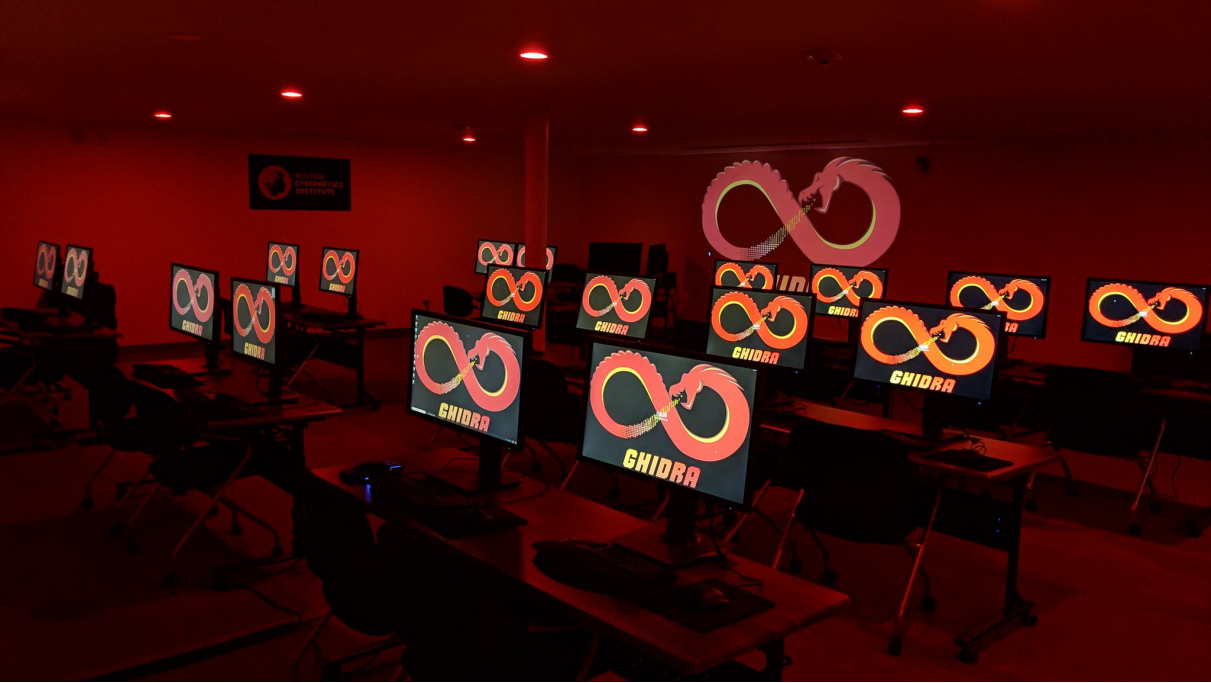
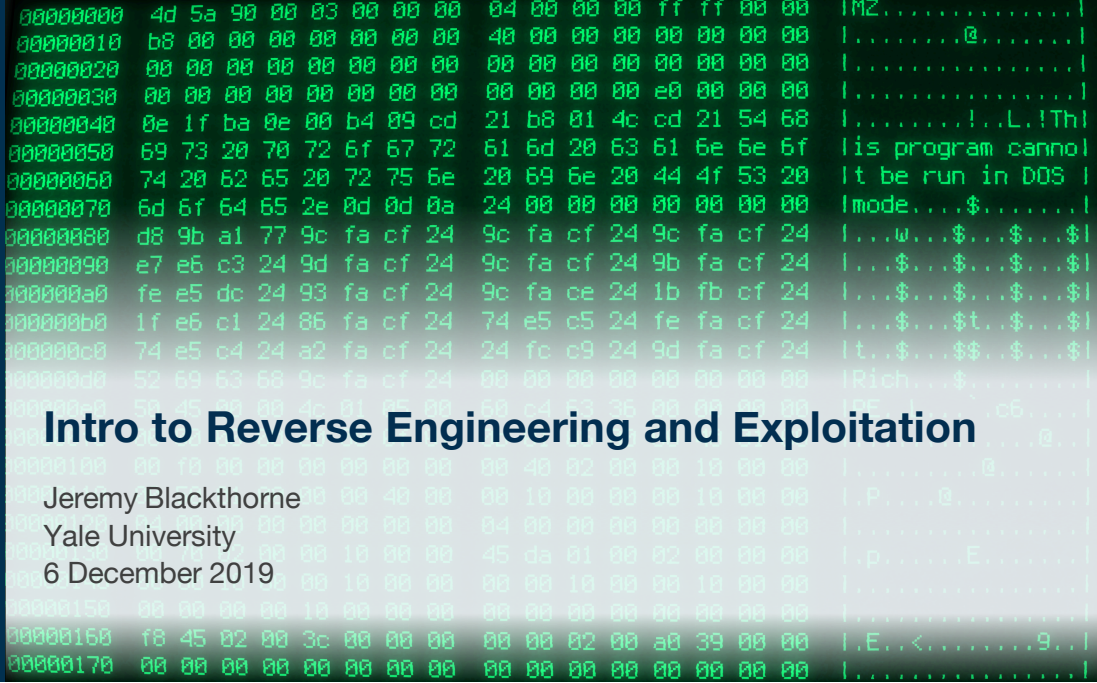

BOSTON CYBERNETICS INSTITUTE
GHIDRA



Intro to Reverse Engineering and Exploitation

Jeremy Blackthorne
Yale University
6 December 2019

Overview

- Reverse Engineering and Exploitation
  - Focus on software
  - Tools
  - Demos
  - Concepts
  - Workshop to follow 2pm – 4pm
- Understanding and Control

Motivation: A New Nature

- You control it or it will control you
- Reversing and exploitation are survival skills

Jeremy Blackthorne

- **Boston Cybernetics Institute**
  - Co-founder and technical staff
  - Cybersecurity research, consulting, and training
  - **“Empowering people to control technology”**
- **Previously at MIT Lincoln Laboratory**
  - Cyber System Assessments Group
  - "...find specific ways to gain and maintain unauthorized control over hardware/software systems.” [1]
- **Education**
  - Bachelor in CS, University of Michigan - Dearborn
  - Master in CS, Rensselaer Polytechnic Institute
  - PhD candidate in CS, Rensselaer Polytechnic Institute
- **United States Marine Corps (2002 – 2006)**
  - 1st Battalion, 7th Marines, 1st Marine Division
  - Three tours in Iraq

Boston Cybernetics Institute (BCI)

- U.S. public benefit corporation founded 2017
- Research, consulting, and training
- Public and private trainings:
  - Reverse-Engineering
  - Malware Analysis
  - Vulnerability Discovery and Exploitation
  - Embedded Systems

Outline

1. Introduction
2. **Reversing**
3. Exploitation
4. Review
5. Path Forward

Forward Engineering

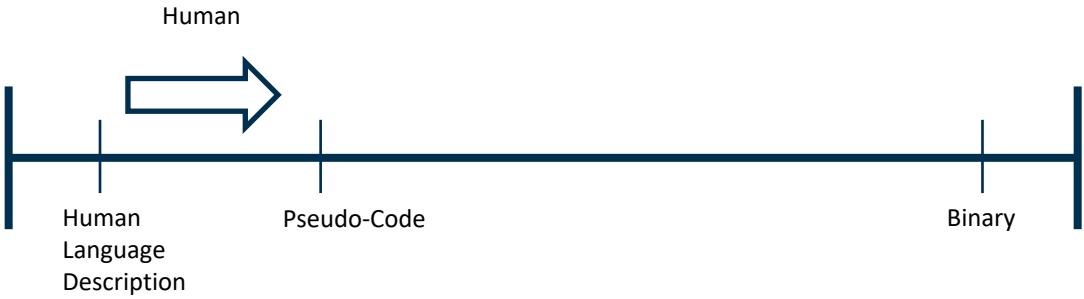
Abstract **Concrete**

Spectrum of Program Abstraction

Forward Engineering

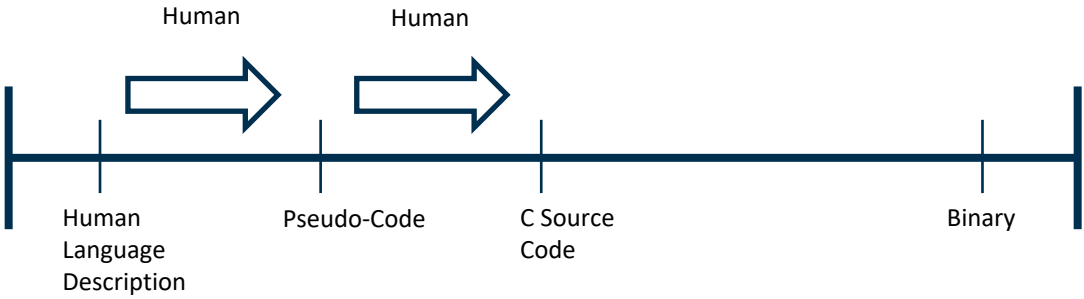
Spectrum of Program Abstraction

Forward Engineering



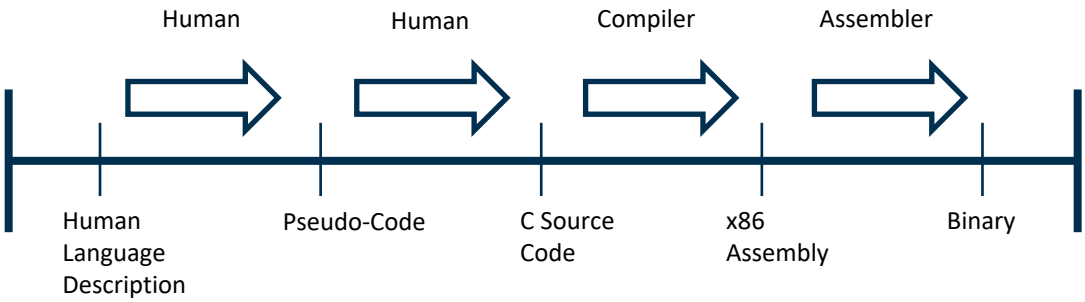
Spectrum of Program Abstraction

Forward Engineering



Spectrum of Program Abstraction

Forward Engineering



Spectrum of Program Abstraction

Reverse Engineering

Abstract **Concrete**

Spectrum of Program Abstraction

Reverse Engineering

Abstract **Concrete**

Spectrum of Program Abstraction

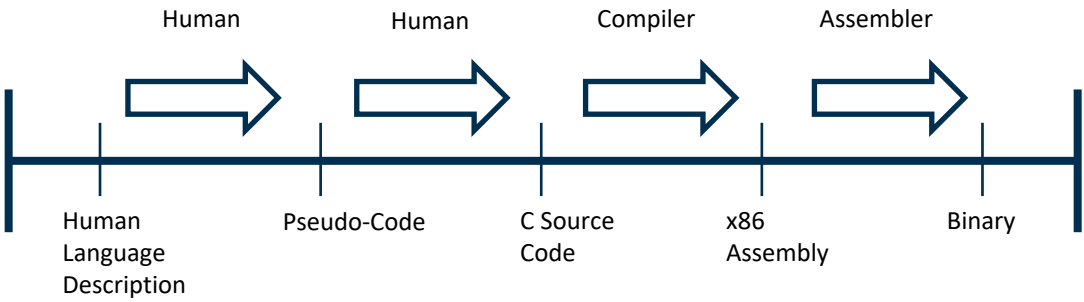
Reverse Engineering

What does this program do?

Abstract **Concrete**

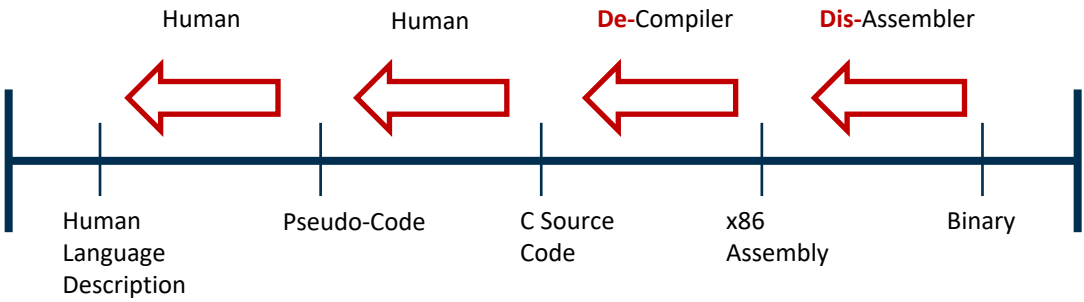
Spectrum of Program Abstraction

Forward Engineering



Spectrum of Program Abstraction

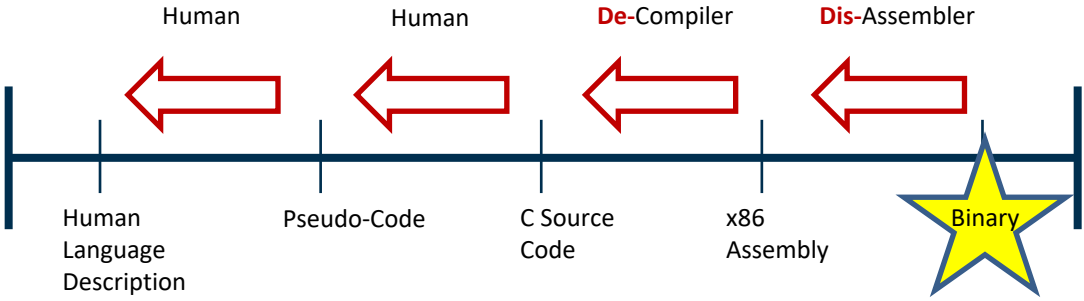
Reverse Engineering



Spectrum of Program Abstraction

Reverse Engineering

What does this program do?



Spectrum of Program Abstraction

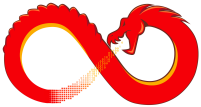
Reverse Engineering

Stop talking

Start showing

Software Reversing Tools

- IDA Pro
  - Free and commercial versions
  - Static and dynamic analysis
- Ghidra
  - Free and open source
  - Static only, dynamic coming soon?
- Binary Ninja
  - Student and commercial versions
  - Static only analysis
- Radare2 (r2)
  - Free and open source
  - Static and dynamic



Recent Ghidra Release

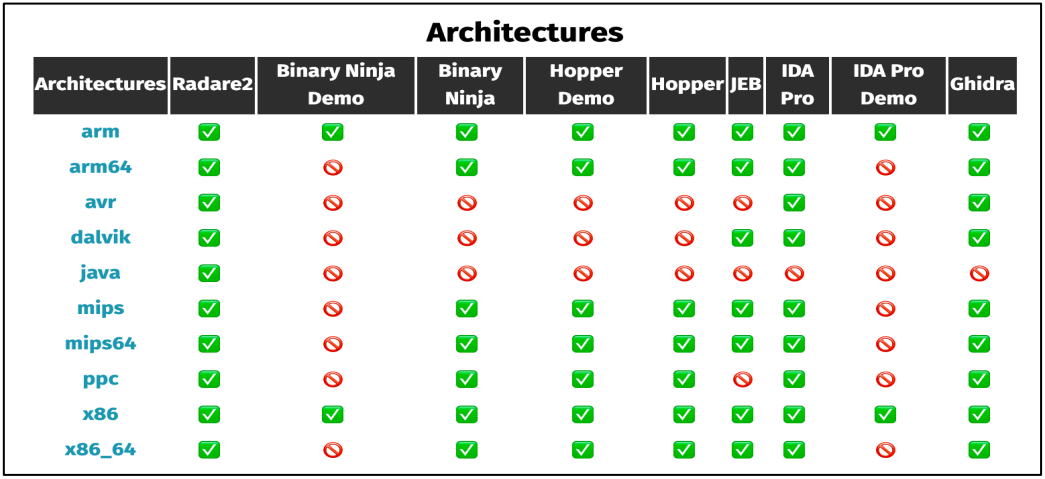
- March 5th, 2019
- RSA Conference
  - **Title:** Come Get Your Free NSA Reverse Engineering Tool!
  - **Author:** Rob Joyce, Senior Advisor, National Security Agency
  - https://www.rsaconference.com/events/us19/agenda/sessions/16608-come-get-your-free-nsa-reverse-engineering-tool

Ghidra History?

```
user@gunmetal ~/D/ghidra-Ghidra_9.0.2_build> grep -r 199[0-9] * --include "*.java"
Ghidra/Framework/SoftwareModeling/src/main/java/ghidra/program/model/address/AddressOutOfBoundsException.java: * @version 1999-0
Ghidra/Framework/SoftwareModeling/src/main/java/ghidra/program/model/scalar/ScalarOverflowException.java: * @version 1999-03-31
Ghidra/Framework/SoftwareModeling/src/main/java/ghidra/program/model/scalar/ScalarFormat.java: * @version 1999/02/04
Ghidra/Framework/SoftwareModeling/src/main/java/ghidra/program/model/mem/MemoryAccessException.java: * @version 1999-03-31
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
).getConstraintValueString());
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
Ghidra/Framework/Docking/src/test/java/docking/widgets/table/constrainteditor/DateRangeConstraintEditorTest.java:
Ghidra/Framework/Generic/src/main/java/generic/hash/SimpleCRC32.java:          0,1996959894,-301047508,-1727442502,124634137,
Ghidra/Framework/Generic/src/main/java/generic/hash/SimpleCRC32.java:          -522852066,-1747789432,162941995,2125561021,-407
Ghidra/Framework/Generic/src/main/java/generic/hash/SimpleCRC32.java:          795835527,1483230225,-1050600021,-1234817731,199
Ghidra/Framework/Generic/src/main/java/generic/hash/SimpleCRC32.java:          -1950435094,-54949764,1658658271,366619977,-1932
Ghidra/Framework/Generic/src/main/java/ghidra/util/exception/NotYetImplementedException.java: * @version 1999/02/05
Ghidra/Features/MicrosoftCodeAnalyzer/src/main/java/ghidra/app/cmd/data/exceptionhandling/EHFunctionInfoModel.java:     public s
Ghidra/Features/MicrosoftCodeAnalyzer/src/main/java/ghidra/app/cmd/data/exceptionhandling/EHFunctionInfoModel.java:     public s
Ghidra/Features/MicrosoftCodeAnalyzer/src/main/java/ghidra/app/cmd/data/exceptionhandling/EHFunctionInfoModel.java:     public s
Ghidra/Features/Base/src/test.slow/java/ghidra/app/merge/listing/FunctionMergerThunk2Test.java: private final static String OVER
Ghidra/Features/Base/src/test.slow/java/ghidra/app/merge/listing/FunctionMergerThunkTest.java:  private final static String OVER
Ghidra/Features/Base/src/test.slow/java/ghidra/app/merge/listing/FunctionMergerThunkTest.java:          chooseVariousOptions("01
Ghidra/Features/Base/src/test.slow/java/ghidra/app/merge/listing/FunctionMergerThunkTest.java:          chooseVariousOptions("01
Ghidra/Features/Base/src/main/java/ghidra/app/util/bin/format/pe/debug/DebugCodeViewConstants.java:     /**Newer CV info, define
```

- @0xAlexei first showed me a version of this

Detailed Tool Comparison



"Unfair comparison between r2, IDA Pro, and Hopper." -https://rada.re/r/cmp.html

Publicly Reported Bugs

MITRE's Common Vulnerabilities and Exposures (CVE's):
https://cve.mitre.org/

Reverse Engineering Goals

Understanding

- Who made this?
- What does this do?
- Where did this get sent from?
- When was this made?
- Why does this exist? (intent)
- How does this work?

Reverse Engineering Applications (1/2)

Malware Analysis

- W.32 Stuxnet Dossier [2]
- NotPetya Technical Analysis [3]
- Gauss: Abnormal Distribution [4]

Reverse Engineering Applications (2/2)

Privacy

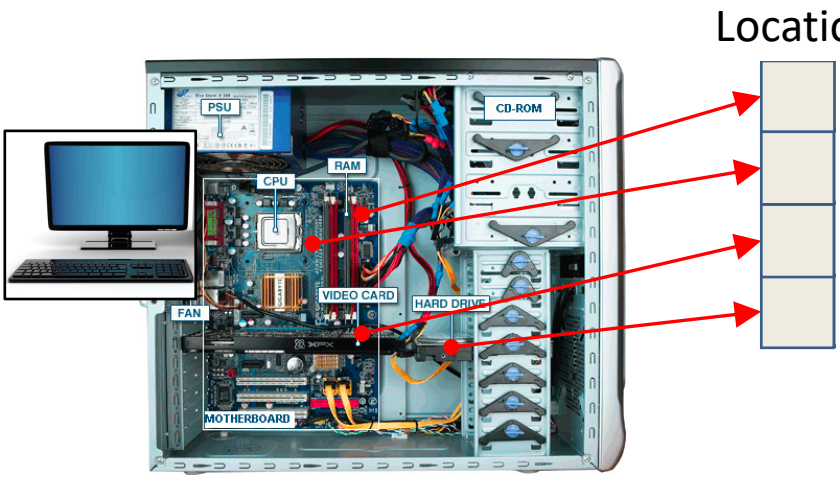
- Ring Video Doorbell – “Police officers who download videos from homeowners’ Ring doorbell cameras can keep them forever and share them with whomever they’d like without providing evidence of a crime,…” [5]

Compatibility, extension, maintenance

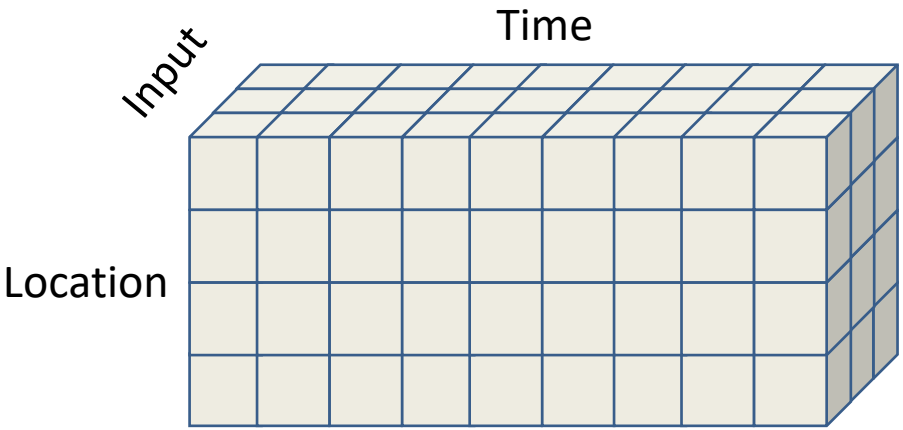
- EFF Reverse Engineering FAQ [6]
- Why American Farmers are Hacking Their Tractors with Ukrainian Firmware [7]
- SimCity Offline Mode Achieved By Hacker [8]

Program State

State Locations

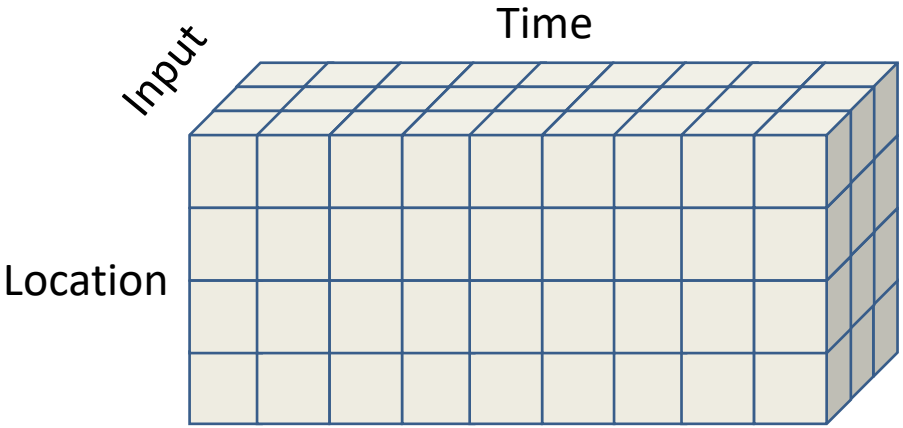


Program State Space



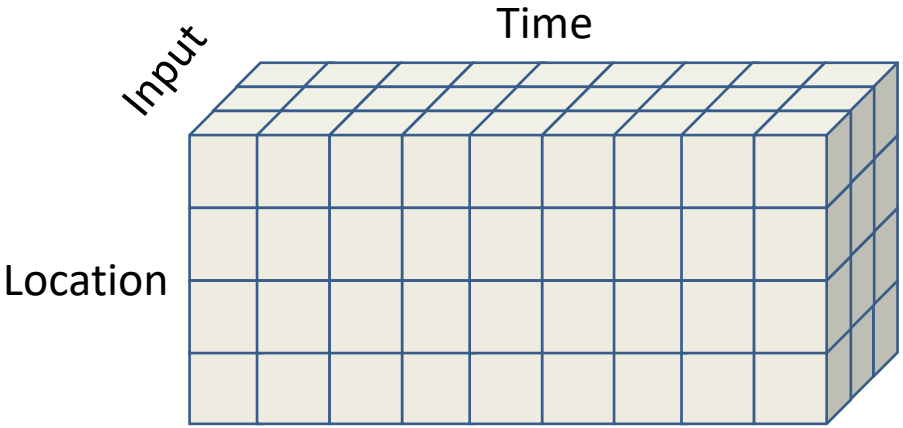
Capabilities: (Read, Write, Execute)

Program State Space - Read



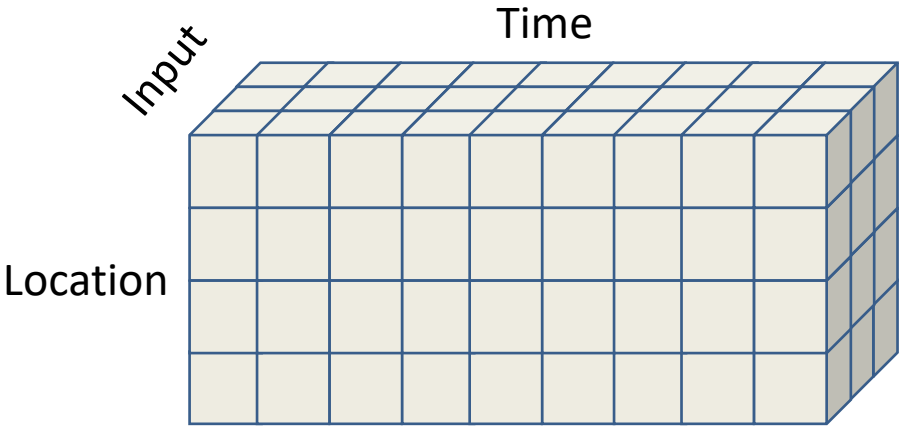
Capabilities: (**Read**, Write, Execute)

Program State Space - Write



Capabilities: (Read, **Write**, Execute)

Program State Space - Execute



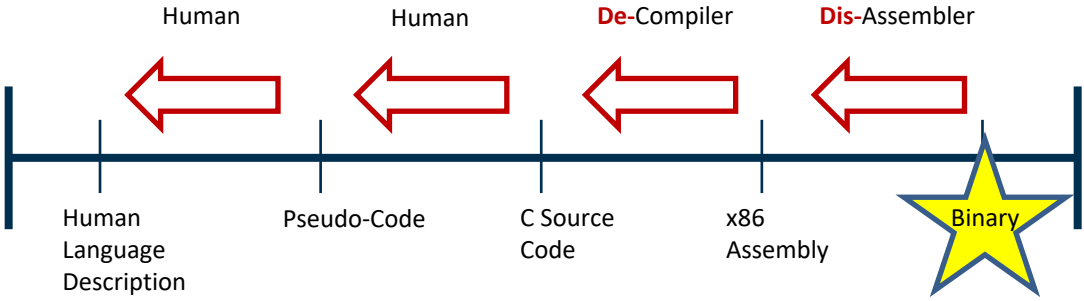
Capabilities: (Read, Write, **Execute**)

Outline

1. Introduction
2. Reversing
3. **Exploitation**
4. Review
5. Path Forward

Reverse Engineering

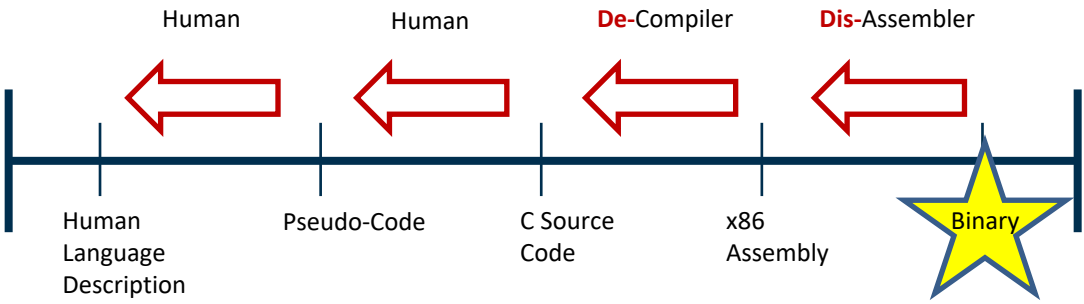
What does this program do?



Spectrum of Program Abstraction

Reverse Engineering

What *can I make this* program to do?



Spectrum of Program Abstraction

EXPLOITATION

The goal is control

Input

- STDIN
- File
- Network socket
- Inter-process communication
- Shared memory
- Cosmic Rays [9]

Vulnerability Types aka Bug Classes

MITRE's Common Weakness Enumeration (CWE):
https://cwe.mitre.org/data/index.html

Primitive Types - RWX

1) Read
    1) Address
    2) Direction
    3) Magnitude
2) Write
    1) Address
    2) Direction
    3) Magnitude
    4) Value
3) Execute
    1) Address
    2) Direction
    3) Magnitude

1) Read
    1) Partial Read
    2) Arbitrary Read
        - “Read-what-where”
2) Write
    1) Partial write
    2) Arbitrary write
        - “Write-what-where”
3) Execute
    1) Partial execution
    2) Arbitrary Execution
        - ”arbitrary code execution”
        - ”pwned”
        - “pop a shell”
        - “remote code execution” (RCE)

Compose Primitives

(Arbitrary Write) + (Execute-what-where) = Arbitrary Execution

Payload

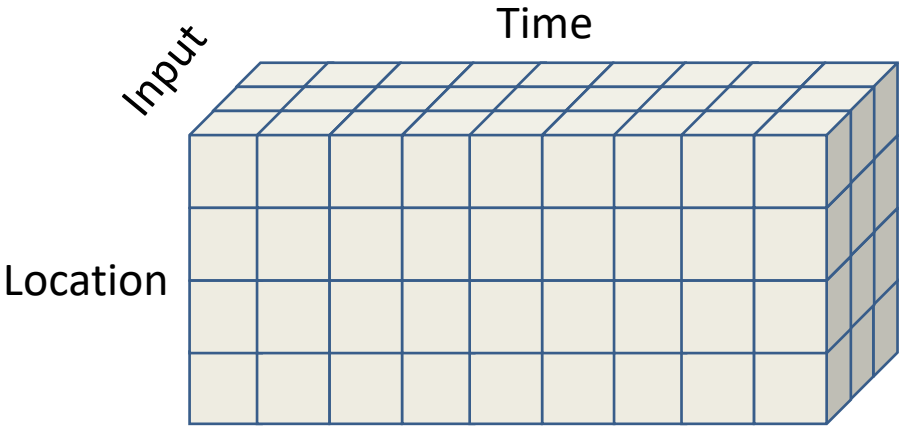
You can execute whatever you want, so now what?

- Privilege escalation
- Pivot
- Exfiltrate data
- Cryptolock and extort
- Sabotage

Exploitation Process: IVPCP

- **I**nput identification
- **V**ulnerability discovery
- **P**rimitive abstraction
- **C**omposition of primitives
- **P**ayload selection

Program State Space with Privilege



Capabilities: (Read, ~~Write~~, Execute)

Publicly Reported Bugs

MITRE's Common Vulnerabilities and Exposures (CVE's):
https://cve.mitre.org/

Exploitation Language

- Memory corruption
- Buffer Overflow
- Vulnerability
- Feature, not a bug
- Exploitation
- Weird Machines: https://www.cs.dartmouth.edu/~sergey/wm/

Exploitation

“He may be able to gain unauthorized access to classified data by exploiting a pre-programmed weakness due to careless design or implementation, or planted as a 'trap door' in the application or in the programming and operating systems supporting the application.

The security threat posed by this mode of use depends on whether the application is designed in such a way as to assure that each user is fully controlled in all actions he may take on the system.”

Exploitation

"By supplying addresses outside of the space allocated to the users program, it is often possible to get the monitor to obtain unauthorized data for that user, or at the very least, generate a set of conditions in the monitor that causes a system crash."

Anderson, J. P. (1972). Computer Security Technology Planning Study (Volume II).

Outline

1. Introduction
2. Reversing
3. Exploitation
4. **Review**
5. Path Forward

Review

Reversing

- Linux
- `hexdump`
- Ghidra
- IDA Free
- `gdb`

Exploitation

- `python`
- pwntools (python library: https://github.com/Gallopsled/pwntools)
- IVPCP: Inputs → Vulnerabilities → Primitives → Compose → Payload

Review High-level

Reversing (Understanding)

1) A program is a series of numbers
2) You can interpret those numbers to derive meaning
3) You can change the numbers directly if you control the environment

---

Exploitation (Control)

4) When unable to change the numbers directly, you can still change input
5) Input is a form of control which can be extended and amplified
6) Unconstrained input often leads to unconstrained control

Outline

1. Introduction
2. Reversing
3. Exploitation
4. Review
5. **Path Forward**

Resources: Reverse-Engineering

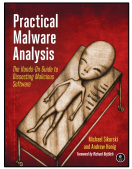
Subreddits (www.reddit.com)
- /r/ReverseEngineering
- /r/REMath
- /r/RELounge
- /r/UIC
- /r/Malware

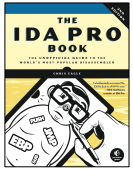
CrackMe's
- https://crackmes.one)
- crackmes.de (no longer up)
- https://github.com/syr0x/crackmes.de
- https://crackmes.one/user/crackmes.de

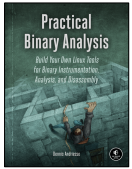
Wargames
- http://smashthestack.org/wargames.html
- http://overthewire.org/wargames/
- http://reversing.kr/
- https://www.wechall.net/

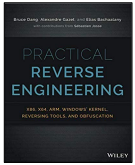
Capture-the-Flags (CTFs)
- https://ctftime.org/

Reading
- Sikorski, M., & Honig, A. (2012). Practical malware analysis: the hands-on guide to dissecting malicious software. No Starch Press.
- Eagle, C. (2011). The IDA pro book: the unofficial guide to the world's most popular disassembler. No Starch Press.
- Andriesse, D. (2018). Practical Binary Analysis. No Starch Press.
- Dang, B., Gazet, A., Bachaalany, E., & Josse, S. (2014). Practical Reverse Engineering: X86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation (1st ed.). Wiley Publishing.









Resources: Exploitation

Shellcoding

- https://syscalls.kernelgrok.com/
- http://shell-storm.org/shellcode/

Pwnables

- http://www.pwnable.xyz
- http://www.pwnable.kr
- http://www.pwnable.tw

Exploit Challenges

- https://trailofbits.github.io/ctf/
- https://github.com/RPISEC/MBE
- http://exploit.education
- http://exploit.courses

Wargames

- http://smashthestack.org/wargames.html
- http://overthewire.org/wargames/
- https://io.netgarage.org/

CTFs

- https://ctftime.org/

Reading

- Erickson, J. (2008). Hacking: The art of Exloitation, 2nd Edition. Assembly (2nd ed.). No Starch Press.
- Anley, C., Heasman, J., Linder, F., & Richarte, G. (2007). The Shellcoder's Handbook (2nd ed.). Wiley.
- Dowd, M., McDonald, J., & Schuh, J. (2006). The Art of Software Security Assessment: Identifying and Preventing Software Vulnerabilities. Addison-Wesley Professional.
- McGraw, G. (2004). Exploiting Software: How to Break Code. Addison-Wesley Professional.
- Klein, T. (2011). A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security. No Starch Press

Next Steps

- For references, slides are here:
  - https://github.com/JeremyBlackthorne/Public-Presentations
- Download the BCI image
- Email/Twitter/Slack message me with questions

Reverse-Engineering

1. Complete reversing exercises on BCI image
2. Complete challenges here: http://reversing.kr/
3. Complete ”pwning” challenges in a CTF: https://ctftime.org/

Exploitation

5. Complete exploitation exercises on BCI image
6. Complete challenges here: http://pwnable.kr/
7. Complete RE challenges in a CTF: https://ctftime.org/

Contact Info

- Boston-Cyber.Eventbrite.com
- www.BostonCyber.org
- @BosCybernetics
- jblackthorne@bostoncybernetics.org
- @0xJeremy
- BostonCyber.slack.com

BCI Classroom – Harvard Square

Path Forward

- 2-hour workshop immediately following this seminar
- *(Follow-up workshop Mann Student Center, Dunham Lab 2-4pm)*

References (1/2)

[1] "MIT Lincoln Laboratory: Cyber Security and Information Sciences: Cyber Systems Assessments." [Online]. Available: https://www.ll.mit.edu/mission/cybersec/CSA/CSA.html. [Accessed: 20-Apr-2018].

[2] N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet Dossier," *Symantec-Security Response*, vol. Version 1., no. February 2011, pp. 1–69, 2011.

[3] "NotPetya Technical Analysis – A Triple Threat: File Encryption, MFT Encryption, Credential Theft." [Online]. Available: https://www.crowdstrike.com/blog/petrwrap-ransomware-technical-analysis-triple-threat-file-encryption-mft-encryption-credential-theft/. [Accessed: 05-Dec-2019].

[4] Karspersky Lab, "Gauss : Abnormal distribution," *Kaspersky Lab Glob. Res. Anal. Team*, no. August, p. 48, 2012.

[5] "*privacy not included - Ring Video Doorbell." [Online]. Available: https://foundation.mozilla.org/en/privacynotincluded/products/ring-video-doorbell/. [Accessed: 05-Dec-2019].

References (2/2)

[6] "Coders' Rights Project Reverse Engineering FAQ | Electronic Frontier Foundation." [Online]. Available: https://www.eff.org/issues/coders/reverse-engineering-faq. [Accessed: 05-Dec-2019].

[7] "Why American Farmers Are Hacking Their Tractors With Ukrainian Firmware - VICE." [Online]. Available: https://www.vice.com/en_us/article/xykkkd/why-american-farmers-are-hacking-their-tractors-with-ukrainian-firmware. [Accessed: 05-Dec-2019].

[8] "SimCity Offline Mode Achieved By Hacker." [Online]. Available: https://www.cinemablend.com/games/SimCity-Offline-Mode-Achieved-By-Hacker-53642.html. [Accessed: 05-Dec-2019].

[9] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM Errors in the Wild: A Large-scale Field Study," in *Proceedings of the Eleventh International Joint Conference on Measurement and Modeling of Computer Systems*, 2009, pp. 193–204.