eCTF Getting Started Guide

Preparing Your Team for Kickoff

MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

1 Getting Started

Welcome to the 2023 Embedded Capture the Flag (eCTF) Competition! This document will introduce you to the basics of the eCTF and help you get your machine's environment set up. The purpose of this competition is to help develop your skills in software, hardware, embedded systems, and security. Don't worry if some of these are new to you; the point is to learn!

2 A Brief Overview of the eCTF

The eCTF competition is a design-build-attack competition with phases for both defense and attack. During the Design Phase, competitors will be tasked with designing and implementing secure embedded software based on a set of functional and security requirements. Later, participants will get the opportunity to attack other teams' designs to compromise the security requirements.

To help prepare you for success, it is useful to understand the basic architecture of an eCTF design and what tools should be installed before the competition starts.

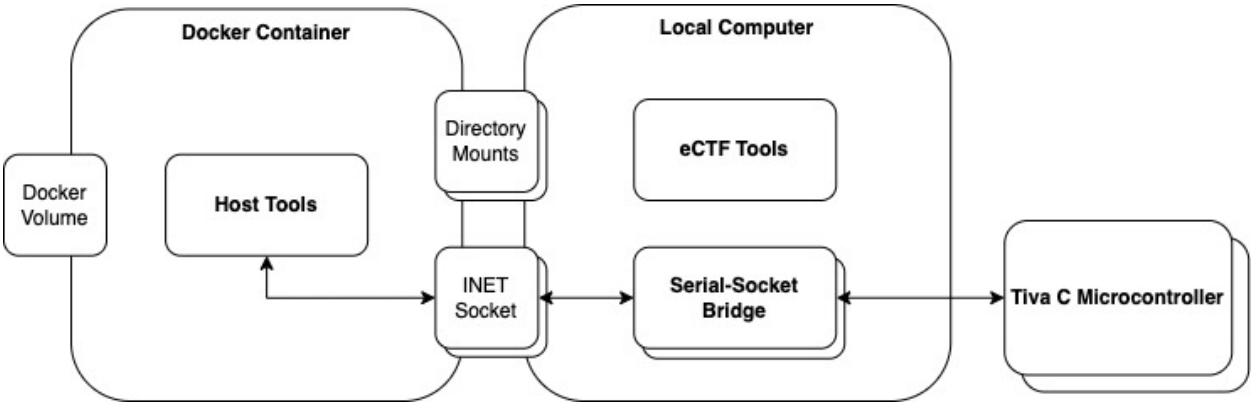

Figure 1. System Architecture for an eCTF Design

First, your team is responsible for developing embedded software that will run on a Texas Instruments (TI) microcontroller on the Tiva-C development board (https://www.ti.com/tool/EK-TM4C123GXL). Examples provided by TI and any code provided by the organizers will be written in C. Additionally, your team will be responsible for developing a set of host tools written in Python to interact with your embedded design. However, you can choose a different language if it is approved by the organizers[1]. To ensure that each team can run the tools you develop without installing additional software, they will execute inside a Docker container provided by your team.

The following sections will provide information about setting up your development environment and links to learning resources that may be helpful for the competition.

[1] C, C++, Rust, and Python 3 are all pre-approved languages and may be used without consulting the organizers

3 Machine Setup

To set up your machine for the eCTF, we require that you install the following list of software:

- Git
- Docker
- Python
- Stellaris® ICDI Drivers
- UNIFLASH

Additionally, we recommend the following optional software:

- Visual Studio Code with Python, C/C++, and Cortex-Debug Extensions
- OpenOCD
- ARM GNU Toolchain

3.1 Required Software

Each of the following sections details the software required to participate in the eCTF.

3.1.1 Git

Git is an open-source version control system. It will allow your team to collaborate on a single code base while managing a history of all the changes you make.

Git is required to submit your design for testing and progression to the Attack Phase. This makes it easy for the organizers to download your code for testing and allows your team to tag a specific version of your code you want to submit.

All Platforms

- https://git-scm.com/downloads

3.1.2 Docker

Docker is a lightweight containerization system. It allows you to package all your tools with the software required to run them.

Docker is used in the eCTF to create an environment for your host tools to execute in. This allows teams to use different programming languages, tools, or libraries during the Design Phase without requiring other teams to download additional software to use their design. Instead, each team will deliver a Dockerfile that builds an image capable of running all their tools.

Windows and Mac (Docker Desktop)

- Windows Installation Instructions (WSL2 backend recommended)
- Mac Installation Instructions
    - Once you download and run the installer, you should start Docker Desktop if it does not start automatically and agree to the terms. Docker must be

run in the background when running the eCTF tools. Make sure that you have a command line environment that can run Docker commands.

Linux (Docker Engine)

- Linux Installation Instructions
  - Select your Linux distribution on this page for specifics
  - Be sure to follow the post-installation instructions to set up your user with privileges to run Docker containers
  - Also look at the instructions on how to start Docker engine automatically at startup (you can start it per-session with sudo service docker start)

3.1.3 Python

Python is a highly readable language with substantial support, which makes it easy to get started with powerful development capability. Setting up a Python virtual environment (such as pyenv) makes it easy to handle dependencies. Python is used in the eCTF tools repository we provide, as well as our reference design example. **The reference design requires Python 3.7 or above.**

All Platforms

There are many of methods to install python on your system:

- python.org: https://www.python.org/downloads/
- pyenv: https://github.com/pyenv/pyenv
- other options: https://realpython.com/installing-python/

3.1.4 Stellaris ICDI Drivers

The development board we will use for this competition is the Texas Instruments TM4C123G LaunchPad Evaluation Kit. You can find a lot of helpful resources on the product page at https://www.ti.com/tool/EK-TM4C123GXL.

The board has an integrated In-Circuit Debug Interface (ICDI). This is what is used to support programming and debugging of the hardware. To use it you need to install the correct drivers.

All Platforms

- https://www.ti.com/tool/STELLARIS_ICDI_DRIVERS.

3.1.5 UNIFLASH

To program the flash memory on the development board, you need to download a programmer. UNIFLASH is compatible with Windows, MacOS, and Linux. There is also an LMFLASHPROGRAMMER, but it has export restrictions and is only compatible with Windows.

All Platforms

- https://www.ti.com/tool/UNIFLASH.

3.2 Recommended Software

Each of the following sections details the software recommended by the organizers for competing in the eCTF.

3.2.1 Visual Studio Code[2]

While there are many good IDEs, we recommend using Visual Studio Code (VSCode) because of its support for in-IDE debugging through the Cortex-Debug extension. Additionally, it is easier to setup than other IDEs we have used.

All Platforms

- https://code.visualstudio.com/download
- Extension installation: https://code.visualstudio.com/docs/editor/extension-marketplace
- We recommend installing the following VSCode extensions for a better coding experience and for debug support:
    - Python
    - C/C++
    - Cortex-Debug

3.2.2 OpenOCD

Open On-Chip Debugger (OpenOCD) is an open-source tool that supports debugging through a JTAG interface on many development boards. OpenOCD can be used with the Cortex-Debug extension in Visual Studio Code to provide an interactive debugging environment for the Texas Instrument development board we will use.

To configure Visual Studio Code with the Cortex-Debug extension and OpenOCD, you will need to create a launch configuration in Visual Studio Code that tells the extension information about your debug session such as the source code location, the path to your OpenOCD executable, and the board configuration files for OpenOCD to use.

[2] TI's default IDE is Code Composer Studio (CCS). Since the eCTF competition uses a custom build environment, the competition is not compatible with CCS. CCS can be utilized as a normal text editor or debugger (setup is complex and we will not be providing support for this), but use of VSCode is recommended.

All Platforms

- OpenOCD: https://openocd.org/pages/getting-openocd.html
- SVD File: https://github.com/posborne/cmsis-svd/blob/master/data/TexasInstruments/TM4C123GH6PM.svd

Example Visual Studio Code launch.json Configuration

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Cortex Debug",
            "cwd": "${workspaceFolder}",
            "executable": "${workspaceRoot}/path/to/application.elf",
            "request": "launch",
            "type": "cortex-debug",
            "runToEntryPoint": "main",
            "servertype": "openocd",
            "device": "TM4C123GH6PM",
            "configFiles": [
                "interface/ti-icdi.cfg",
                "board/ti_ek-tm4c123gxl.cfg"
            ],
            "svdFile": "${workspaceRoot}/TM4C123GH6PM.svd"
        }
    ]
}
```

3.2.3 ARM GNU Toolchain

Our custom build process for the embedded software that will be loaded onto the TI board will run in a Docker container. Therefore, it is not necessary to install a compiler on your local machine. However, it may be helpful to have if you want to do any local builds of your code or debug any build issues. A popular cross-compilation toolchain for ARM is the GNU Toolchain.

All Platforms

- https://developer.arm.com/downloads/-/arm-gnu-toolchain-downloads

4 Learning Resources

If you do not already have a strong background in Python and C, we'd highly recommend spending some time learning them and becoming familiar with them prior to the competition. If you have some experience using these tools, it can still be helpful to look through the list of modules and brush up on any skills you're less comfortable with. We have also included some links for other topics that are important for this competition. This list is not exhaustive and is made to give you a good starting point. You are encouraged to do your own research as you see fit.

4.1 Python Resources

- https://docs.python.org/3/tutorial/index.html
- https://www.learnpython.org/

4.2 C Resources

- "The C Programming Language" by Kernighan and Ritchie
- https://www.learn-c.org/

4.3 Embedded Resources

- https://embeddedartistry.com/beginners/
- https://embedded.fm/blog/ese101

4.4 Cybersecurity Resources

- http://myemates.com/styled/cybersecurity.html
    - Recommended content:
        - Cybersecurity design principles
        - Cryptography
        - Encryption
        - Hashing
        - Buffer overflows

5 Texas Instruments Resources

The development board used for this competition will be the TI TM4C123G LaunchPad Evaluation Kit. It is never too early to start becoming familiar with the hardware! Look through the resources TI offers and test out example code for yourself.

- Main landing page for all resources related to the evaluation kit we will use
  - https://www.ti.com/tool/EK-TM4C123GXL
- Software Development Kit (SDK) with helpful software libraries and example projects for the TM4C ARM® Cortex®-M4F device family
  - https://www.ti.com/tool/SW-TM4C
  - *Note: The distribution method for this is specific to Windows, but the resources can also be extracted and used on both MacOS and Linux.*
- TM4C123G Workshop with tutorials
  - https://software-dl.ti.com/trainingTTO/trainingTTO_public_sw/GSW-TM4C123G-LaunchPad/TM4C123G_LaunchPad_Workshop_Workbook.pdf
  - *Note: Many of these instructions are specific to Windows and use CCS. However, the material can still be relevant for learning on other platforms.*
- Another Workshop link with tutorial videos
  - https://training.ti.com/tiva-c-series-workshop?keyMatch=TIVA%20C%20SERIES
  - *Note: Like other TI resources, this series involves working on Windows and with CCS, but the information is also helpful for other platforms.*

6 Previous Competitions

Now that you've reviewed all the basics and have your board set up, you can familiarize yourself with what you can expect during the competition. We have posted the rules documents, technical specifications documents, example designs, and team poster submissions from the previous competitions. Although these were different design challenges, the format of the competition and expectations of the participants are similar. Additionally, reading through the prior team poster submissions can help you learn about how to create your design for this year.

You can find the links to resources from past competitions at https://ectf.mitre.org/past-competitions/.

Note: When looking at previous resources, there may be information about environment setup that will not be as useful to you for the upcoming competition. Specifically, it isn't required that you download CCS, and is recommended that you use an alternative IDE and programmer such as VSCode instead.