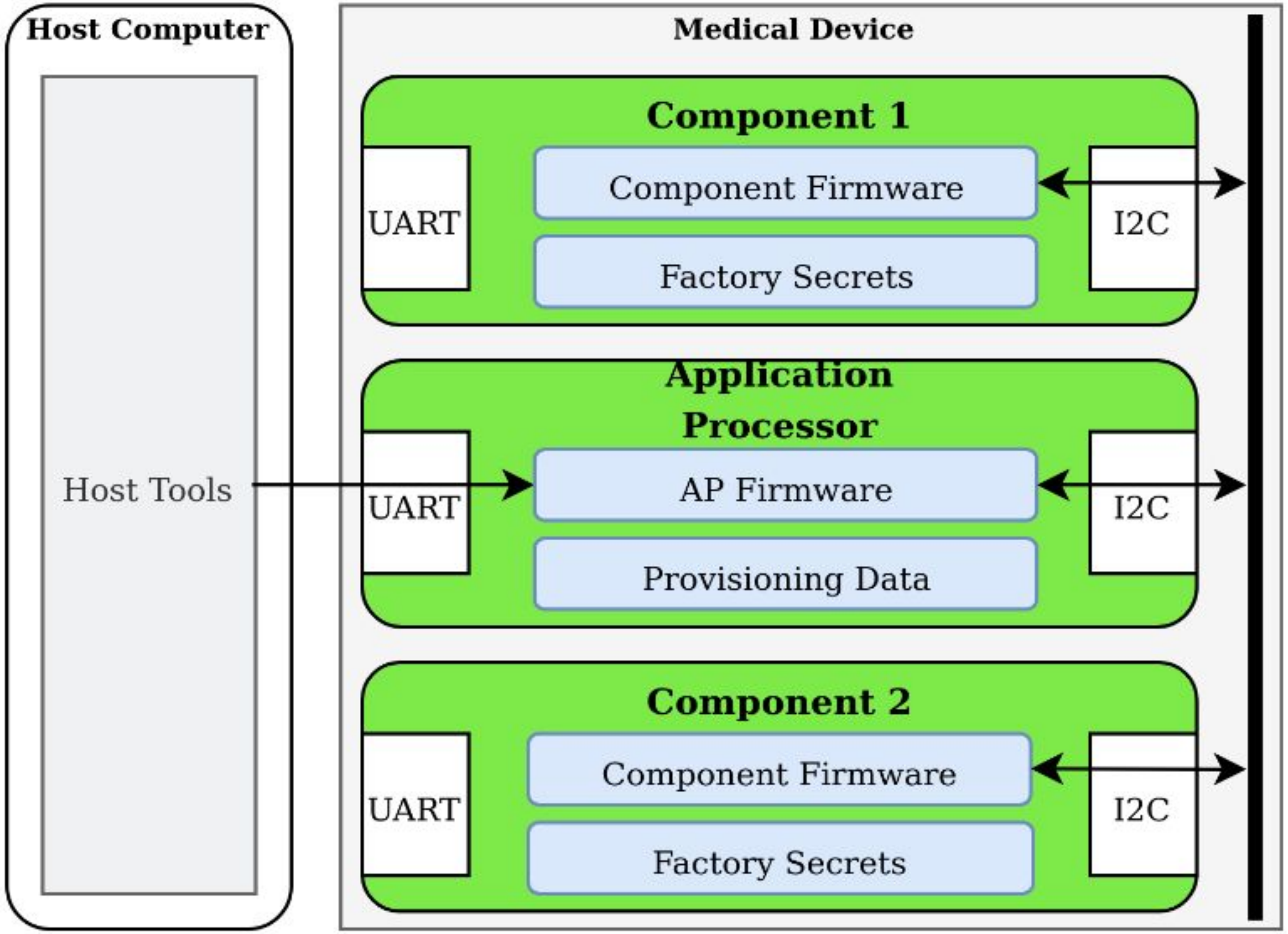
Mitre eCTF Design Phase

Medical Device Edition
Now - April 17 11 AM

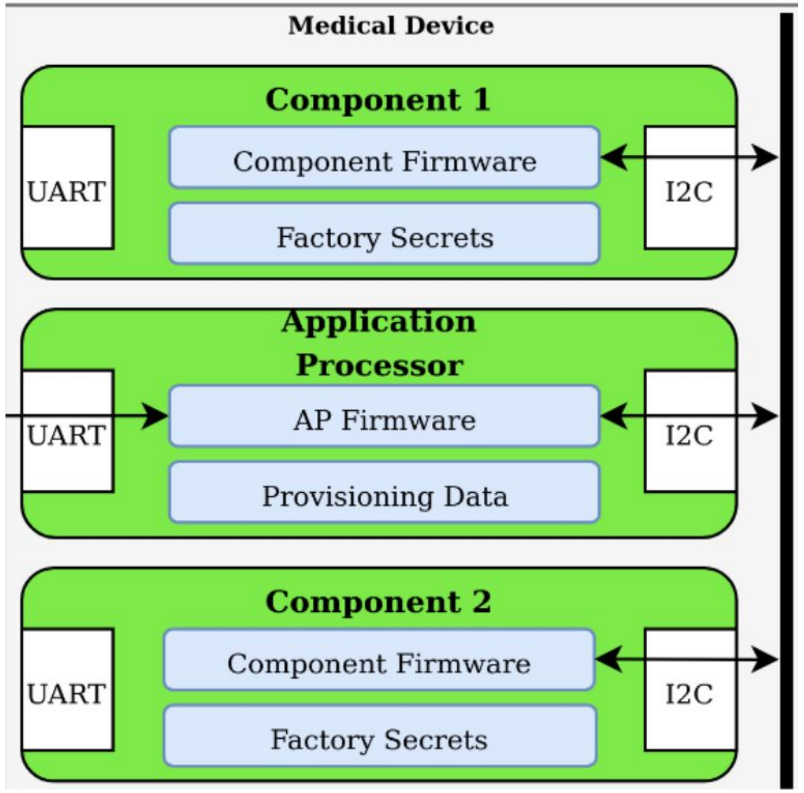
MITRE has an eCTF!

This Year's Theme is Medical Devices

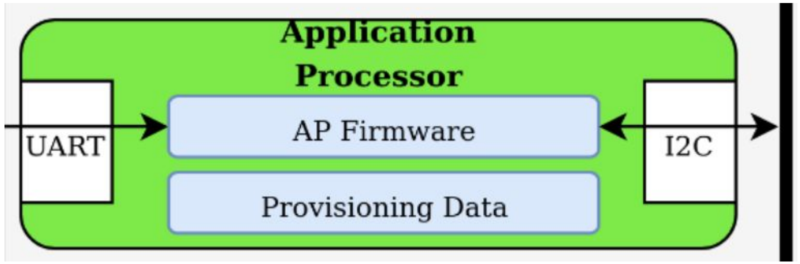
System Architecture


Host Computer
Host Tools
Medical Device
Component 1
UART
Component Firmware
Factory Secrets
I2C
Application Processor
UART
AP Firmware
Provisioning Data
I2C
Component 2
UART
Component Firmware
Factory Secrets
I2C

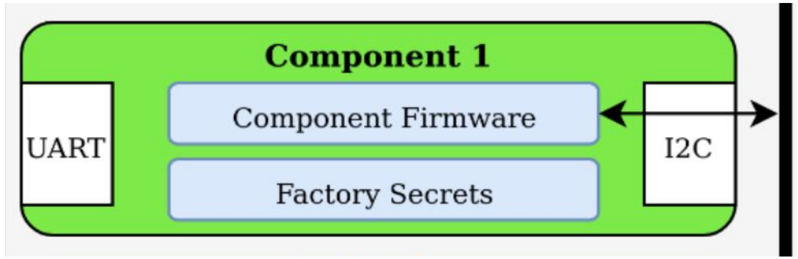
The MISC - Medical Infrastructure Supply Chain

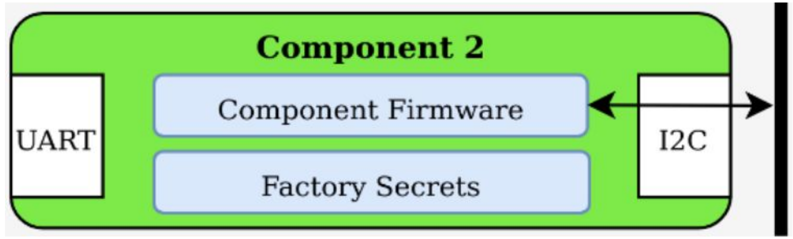


The AP - Application Processor



The Components





This year MITRE has Released
A Reference Design that implements all of these functional requirements for newer teams

… But implements none of the security requirements.

… But implements none of the security requirements.

OUR JOB

Security Requirement 1

Functional Requirement: Boot

The MISC must boot the Application Processor (AP)

SECURITY REQUIREMENT 1

The Application Processor (AP) should only boot if all expected Components are present and valid.

Security Requirement 2

Functional Requirement:
List Components

The MISC must be able to list all the Component ID currently installed.

SECURITY REQUIREMENT 2.

The components should only boot if commanded by a valid AP

Security Requirement 3

Functional Requirement:
Attest

The MISC must allow a user to retrieve the Attestation Data stored on the Components during the build process

Functional Requirement: Replace

The MISC must allow an user to replace a failing component with a new one

Validation

You need a valid Attestation PIN to get Attestation Data

You need a valid replacement token to replace a component

SECURITY REQUIREMENT 3.

The Attestation Pin and Replacement Tokens must be kept confidential

Security Requirement 4

SECURITY REQUIREMENT 4.

The Attestation Data must be kept confidential

Security Requirement 5

SECURITY REQUIREMENT 5.

The MISC must provide a secure communications channel to send and receive messages.

All Requirements can be Found Here

Security :
https://ectfmitre.gitlab.io/ectf-website/2024/specs/security_reqs.html

Functionality :
https://ectfmitre.gitlab.io/ectf-website/2024/specs/functional_reqs.html

Scoring

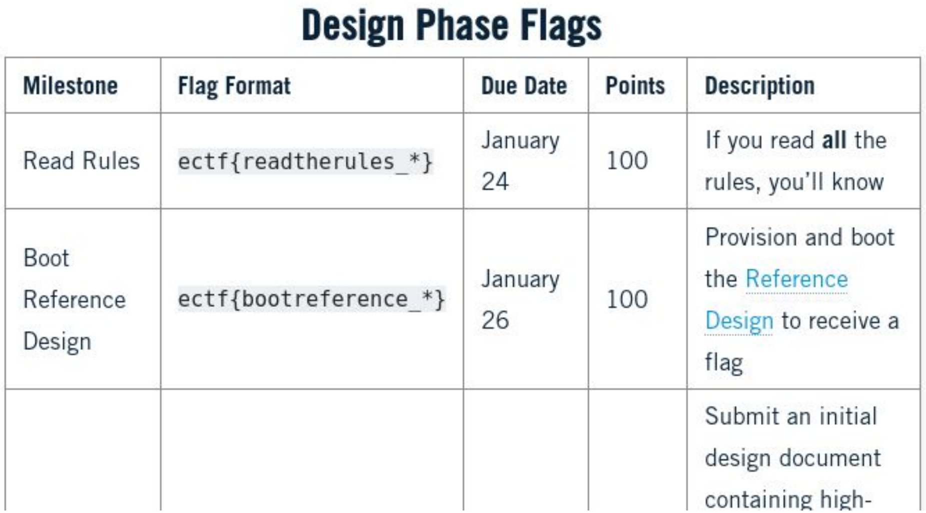
Scoring System

Teams may earn points through three different types of points:

- Design Phase Points
- Attack Phase Points
- Miscellaneous Points.

Design Phase Points (ignore due dates)

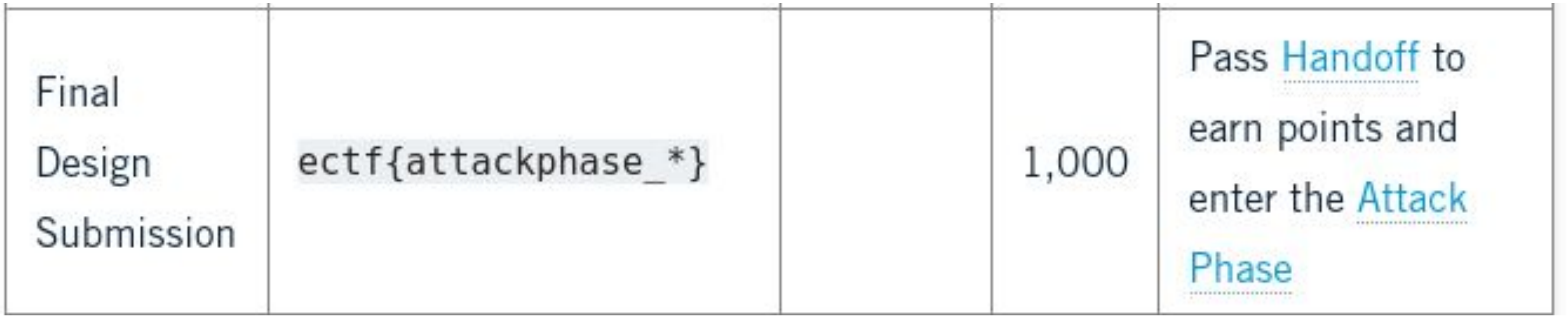
https://ectfmitre.gitlab.io/ectf-website/2024/flags/design_flags.html



Design Phase Flags

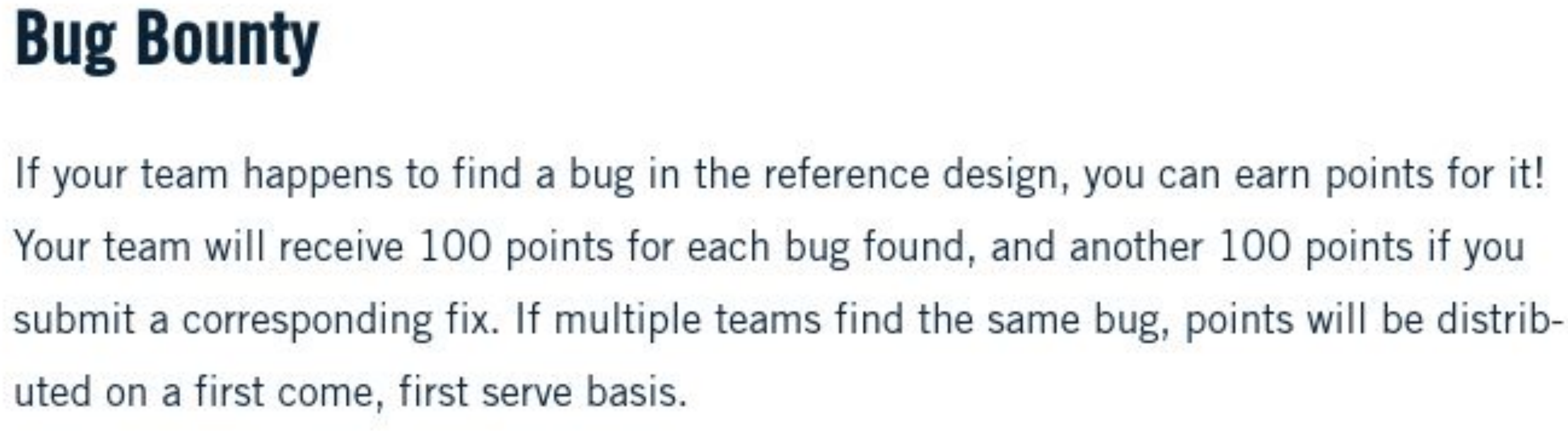
Milestone	Flag Format	Due Date	Points	Description
Read Rules	`ectf{readtherules_*}`	January 24	100	If you read **all** the rules, you'll know
Boot Reference Design	`ectf{bootreference_*}`	January 26	100	Provision and boot the Reference Design to receive a flag
				Submit an initial design document containing high-

The big one, submission of this flag will let us go to the attack phase.



Final Design Submission	`ectf{attackphase_*}`		1,000	Pass Handoff to earn points and enter the Attack Phase

Unintended behavior in the Reference Design? LET THEM KNOW



Bug Bounty

If your team happens to find a bug in the reference design, you can earn points for it! Your team will receive 100 points for each bug found, and another 100 points if you submit a corresponding fix. If multiple teams find the same bug, points will be distrib-uted on a first come, first serve basis.

Good Documentation will be Rewarded!

So please do your best to make good designs, comment your code, and do not depend on security by obscurity

Scoreboard

https://ectfmitre.gitlab.io/ectf-website/about/scoreboard.html

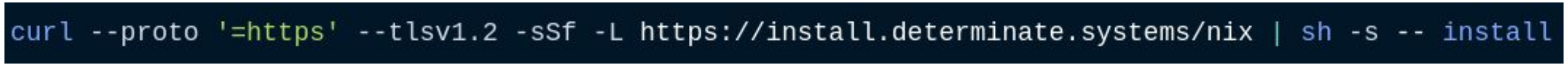
Environment Setup

Things you Need

- Linux
- Nix Package Manager

Use the determinate installer

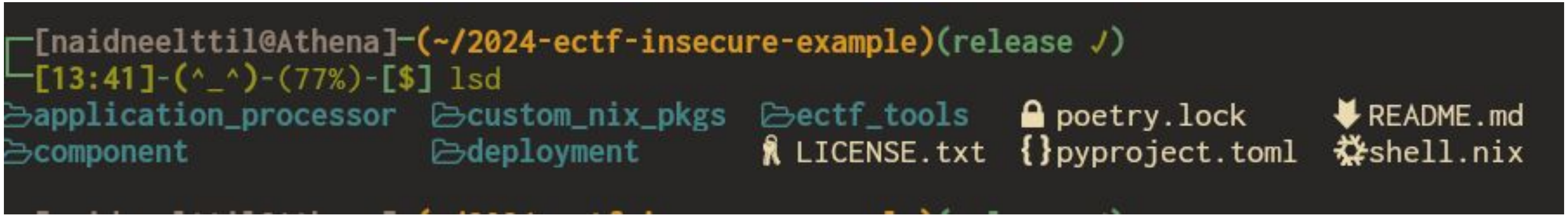
https://determinate.systems/posts/determinate-nix-installer/



```
curl --proto '=https' --tlsv1.2 -sSf -L https://install.determinate.systems/nix | sh -s -- install
```

Clone the Reference Design

https://github.com/mitre-cyber-academy/2024-ectf-insecure-example



```
┌[naidneelttil@Athena]-(~/2024-ectf-insecure-example)(release ✓)
└[13:41]-(^_^)-(77%)-[$] lsd
application_processor   custom_nix_pkgs   ectf_tools     poetry.lock       README.md
component               deployment        LICENSE.txt    {}pyproject.toml  shell.nix
```

```
nix-shell
```

```
poetry install
```

Then you can run the ectf tools with:

```
poetry run <enter command here>
```

Pre-Approved Languages

- C (Reference Design Language)
- C++
- Rust