Challenge Design Summary

Protected Automotive Remote Entry Device

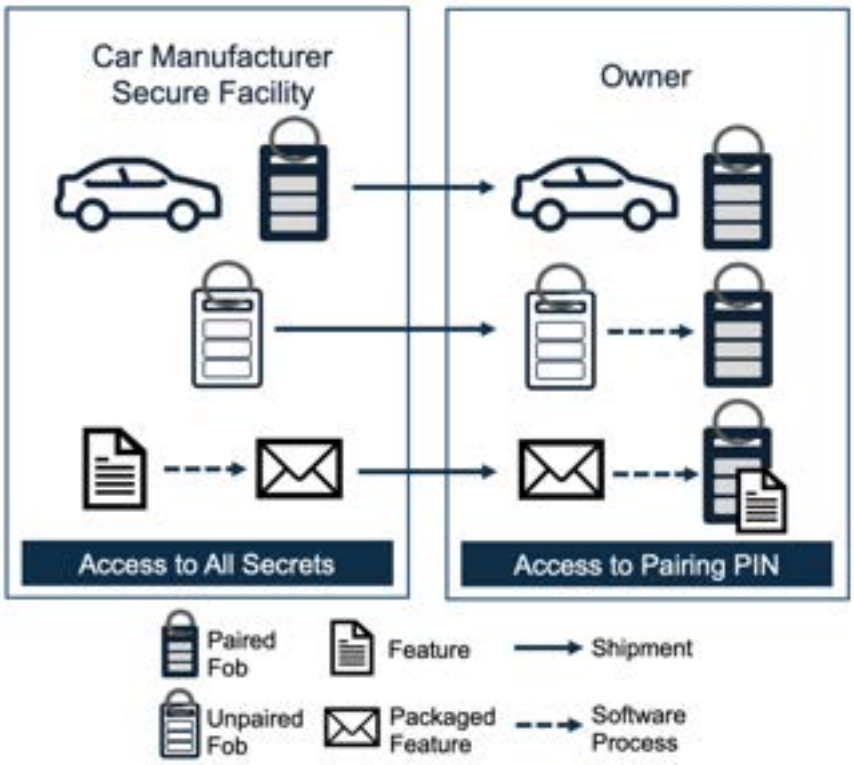
MITRE | SOLVING PROBLEMS FOR A SAFER WORLD™

TABLE OF CONTENTS

1 Challenge Overview

1.1 Motivational Scenario

You are part of an elite design and development team at a car company developing firmware for its next generation of key fobs and cars, the Protected Automotive Remote Entry Device (PARED). Although the main function of the key fob is to allow the owner to unlock and start their vehicle, the product marketing team has dreamed up new features that the key fob must support. These features are summarized below in Figure 1.



The owner of a car will receive pre-paired fobs that unlock their car.

The manufacturer will also produce unpaired fobs that any owner can pair with their car.

If a car owner pays for an upgraded feature, then the manufacturer will send them a file to install onto an existing paired fob, which will then enable the upgraded feature on their car.

Figure 1. The typical lifecycles of a car and key fobs

How will you develop these features and ensure that the system is secure? You and your team have been tasked with figuring it out!

1.2 Competition Structure (Phases and Objective)

This is a design-build-attack competition with phases for both defense and attack as summarized in Figure 2, below. There are opportunities to score points in both phases - primarily by obtaining "flags"[1] and submitting them to the live eCTF scoreboard. The objective of the competition is to earn the most points for your team through both secure design and skilled attacks.

During the Design Phase, you are acting as a team of engineers at a car manufacturer. Your job is to build out the PARED embedded software that will get provisioned on your next line of cars and key fobs to be sold to customers. Once you've completed your design, it will be put to the test as other teams attempt to identify and exploit security flaws in your system.

During the Attack Phase, you will have an opportunity to analyze all other completed designs to identify security flaws that allow you to unlock/start a car or enable additional features.

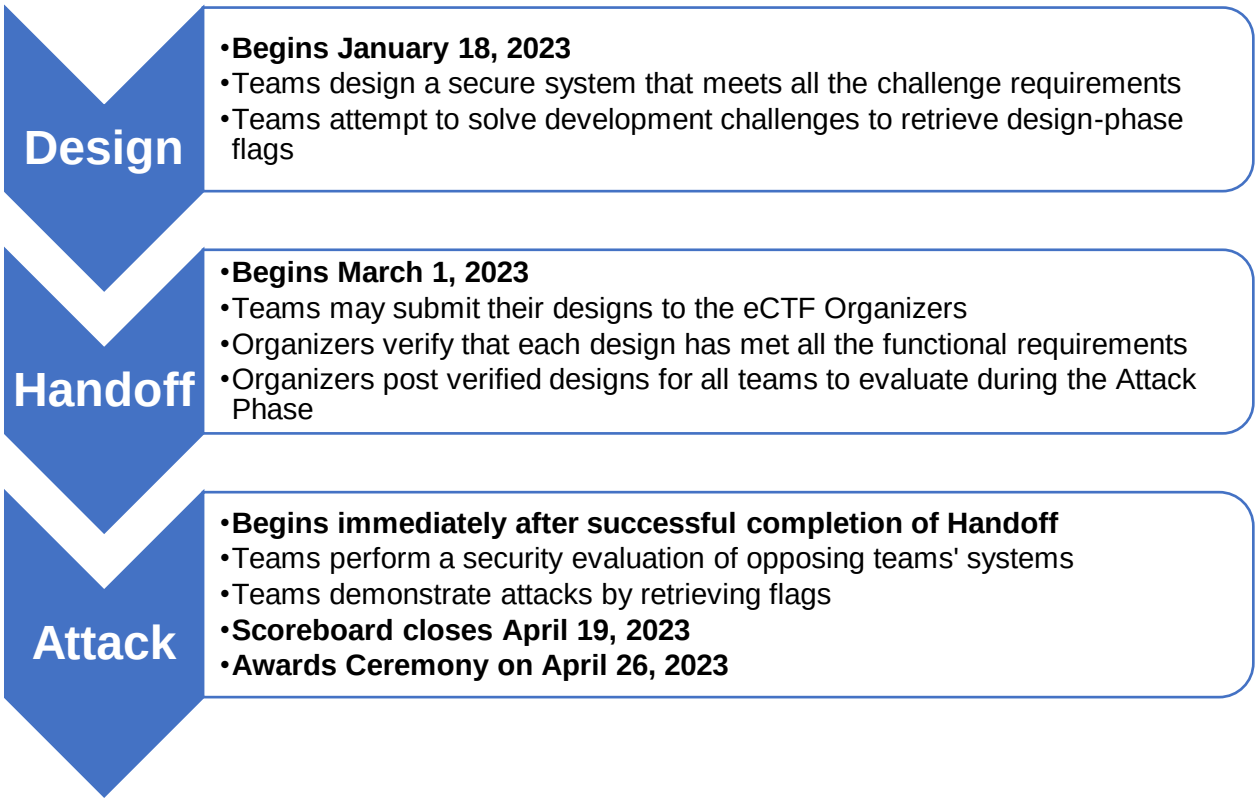


Figure 2. Competition Timeline

1.2.1 Design Phase

In the Design Phase, each team must design and implement a system that meets a set of functional requirements and security requirements. **Teams will be provided with a reference design that meets the functional requirements but intentionally does not attempt to meet any security requirement**. Teams may use the reference design as a starting point or build

[1] Flags are simply strings of characters and digits (e.g., `ectf{tempfobaccess_0123456789abcdef}`) which can be found by solving challenges or demonstrating a successful attack against another design.

their design from scratch. In either case, the directory structure of the submitted design must match the structure defined in Section 1.2 of the Technical Specifications[2].

During the Design Phase, teams may score points by capturing Design Phase flags which show that teams are making progress towards a complete design. Flags must be submitted on the scoreboard by their deadlines for points to be awarded. Additionally, teams may score points by finding functional errors in the tools and example design provided by the organizers as part of a bug bounty program.

1.2.2 Handoff

Starting March 1st, each team may submit their completed design to the organizers. The organizers will then verify that the submission meets all functional requirements. If a submitted design passes functional testing, that team will move into the Attack Phase two business days after the initial submission to allow time for organizers to verify and process the submission. This two-day turnaround period may be shortened to the discretion of the organizers. Therefore, the date and time of transition from Design Phase to Attack Phase may vary between teams. For example: If Team A and Team B both submit systems on the handoff date, but only Team A's system passes the tests, then only Team A will move into the Attack Phase while Team B remains in the Design Phase until they submit a system that meets all functional requirements.

Each submission must include all source code and documentation for the submitted design. This includes all code necessary for building and running the system in accordance with the system functional requirements. The system source code and all your documentation must reside in a Git repository. Please see Section 2 in the sperate Technical Specifications document for details on how to submit your design.

Upon receiving a submission, the eCTF organizers will clone and provision the team's system via their Git repository. Then, the organizers will run a sequence of test cases that validate whether the system meets the functional requirements. ***Note**: **The test cases will not check for the correctness of any security requirements**.* The eCTF organizers will contact the submitting team within **two business days** after the submission indicating whether the system is accepted or not, determined by passing all functional tests successfully and conforming to all other rules.

Accepted Designs

If a system is accepted, the organizers will inform the team and create a handoff package that includes all source code, all documentation, and all distributed Attack Phase artifacts (See the Technical Specifications Document for more details). The team must approve of the handoff package before advancing into the Attack Phase. ***Note:** Teams are not allowed to modify their designs after reviewing and approving the handoff package. The handoff package serves as the final opportunity for teams to verify that they have not left any sensitive system materials in their repositories that they do not wish to be publicly known. If the team decides not to approve the handoff package to make a change to their design, they will have to go through the full Handoff process (i.e., functional testing and the two business-day delay) again before moving to the Attack Phase. Minor modifications may be exempted from the resubmission process at the discretion of the organizers.*

[2] The Technical Specifications are in a separate document provided by the organizers.

Rejected Designs

If a system is not accepted, the eCTF organizers will inform the team and provide an explanation for why the design did not pass testing. The submitting team must then revise their design and submit a new version to the organizers.

1.2.3 Attack Phase

During the Attack Phase, each design that has been validated during Handoff will be made available to other teams for attack. Teams will be able to load other teams' designs on a physical microcontroller with a secure bootloader provided by the organizers. When attacking the other designs, teams will earn points by capturing Attack Phase flags, which demonstrates that they were able to defeat a security requirement of another design. Additionally, teams can earn defensive points over time if their flags go uncaptured by other teams.

1.2.4 Award Ceremony

An Award Ceremony will be held on April 26 to celebrate the competition, reveal the final scores, and announce the winners, so make sure to block off your calendar! The ceremony will be hybrid with both in-person and remote components and top teams will be able to present about their hard work. There will be opportunities to receive sponsorship to cover travel costs. Keep an eye out on the competition Slack workspace for more details.

2 System Architecture

Your design will consist of four main components: host computer, car, paired fob, and unpaired fob. Figure 3 shows a high-level overview of the system architecture.

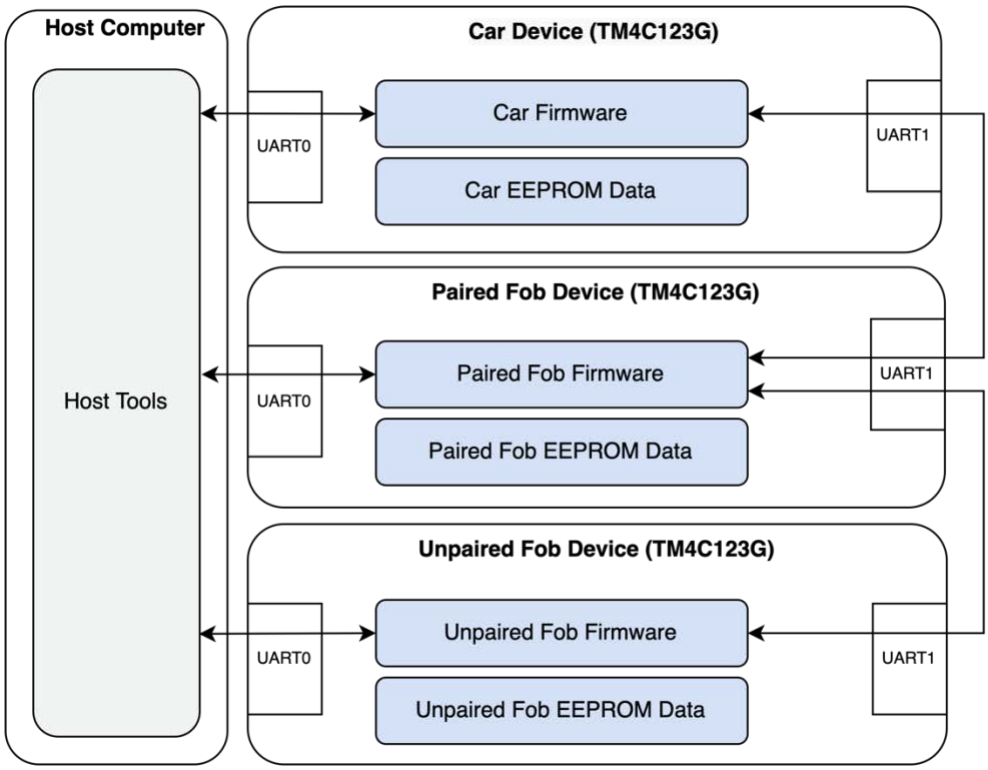


Figure 3. System Architecture

2.1 Host Computer

The host computer is a general-purpose computer used to communicate with the car and fob devices over a serial interface. These tools will be used to pair a new fob with a specific car and to instruct a paired fob to unlock a car. Since the host computer is a general-purpose computer, the host tools running on it will have access to common Linux programs, utilities, and packages. The reference design provided by the eCTF organizers implements the host tools in Python. If your team wishes to use a different language to write the host tools, you are required to ask the organizers for approval[3].

2.2 Car and Key Fobs

The main purpose of the paired fob is to unlock a car. Additionally, an unpaired fob can be paired to work with a specific car. More detail is provided in Section 1.1 above, in the functional requirements in Section 3 below, and in the separate Technical Specifications document. These devices will use Texas Instruments Tiva-C Microcontrollers (TM4C123GH6PM)[4]. Teams will develop their car and key fob designs to run on this microcontroller. The design must work with the provided, unmodified development boards using the MITRE-provided bootloader.

[3] C, C++, Python, and Rust are all pre-approved for use in the eCTF
[4] https://www.ti.com/tool/EK-TM4C123GXL#tech-docs

2.3 Development Resources

Teams will be provided the following resources:

- **2 un-keyed Tiva-C TM4C123G Launchpad devices:** These boards will be used for development. Additionally, instructions will be provided to run the host tools on a local computer to test the entire system using the physical hardware. These devices will not be able to run Attack Phase designs provisioned by the eCTF organizers. However, the development microcontrollers can be used to build and practice attacks against designs in the Attack Phase that are compiled locally by the team from source.
- **2 keyed Tiva-C TM4C123G Launchpad devices:** These boards will be used for the Attack Phase to load attack targets (i.e., other teams' designs) that are provided by the eCTF organizers. These devices are specially configured for use in the Attack Phase and therefore will be unusable during the Design Phase. These boards will be distributed to teams as Handoff nears.
- **Set of 2.54mm Jumper Wires:** These wires will be used to connect the UART header pins between two of the Tiva-C development boards.

3 Functional Requirements

This section defines the functional requirements of your design. The requirements are presented as a series of steps. For each step, a command line tool provided by the organizers will be invoked to interact with the tools your team creates. Therefore, it is important that your design follows these requirements. The functionality described in each step will be verified by the organizers during the Handoff Phase, and you must adhere to these requirements to advance to the Attack Phase.

The functional requirements described here are intentionally high level. See Section 3 of the separate Technical Specifications document for detailed requirements including tool inputs, outputs, and environments.

3.1 Build PARED System

The complete system will be built at the car manufacturer's secure facility. This build process, which is completed before attackers get access to your system, produces the following (as shown in Figure 4):

- Host computer Docker Image
- Host Tools
- Host Secrets File
- Car Binary and EEPROM File
- Paired Fob Binary and EEPROM File
- Unpaired Fob Binary and EEPROM File

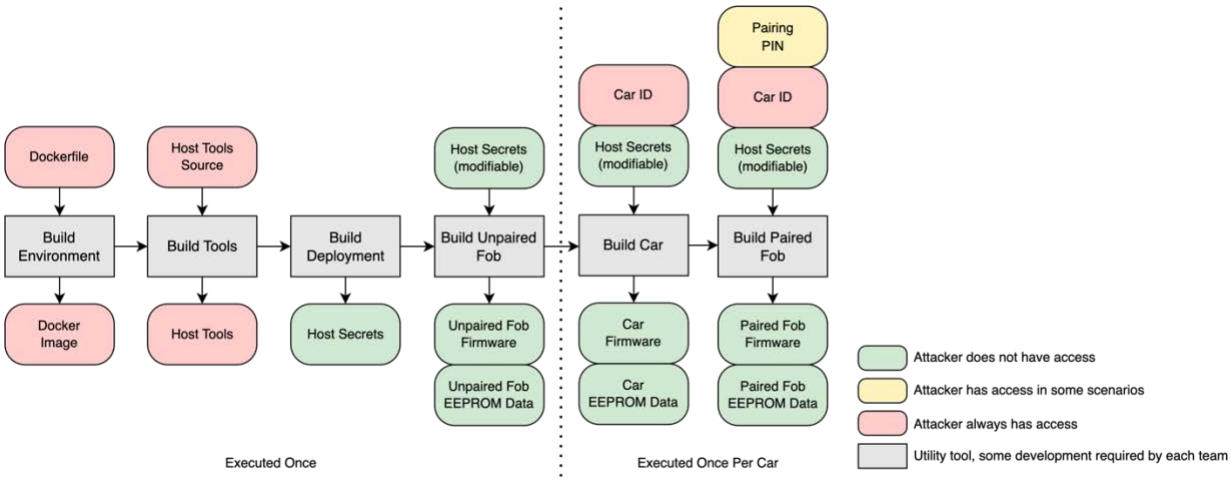


Figure 4. An overview of the build process for your PARED system

3.1.1 Build Environment

To allow other teams to use the tools you develop without needing to install a package of required software, the first step in your build process is to build a Docker[5] image. A Docker image can be thought of as a pre-built snapshot of your development environment. Docker images are built by adding a list of instructions to a Dockerfile. The instructions specify what software must be installed to execute the remaining steps (e.g., Ubuntu, Python, make, clang). After the Docker image is built, all remaining steps are run in a Docker container using this image. See the separate Technical Specifications document for details.

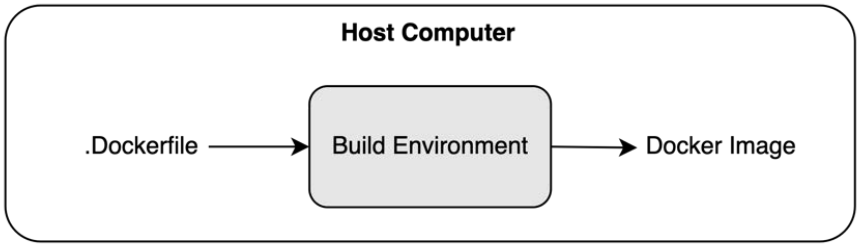


Figure 5. Build Environment Step

3.1.2 Build Tools

The second step is to build the host tools. Host tools are executable files used to interact with the car and fobs. In your PARED system, you will have host tools for packaging features, enabling features, pairing new fobs, and unlocking the car. These tools are written and saved on your local machine. During this step, a Makefile is invoked that will move your host tools into a Docker volume, building them if necessary[6]. **The resulting host tools will be given to the other teams in the Attack Phase**.

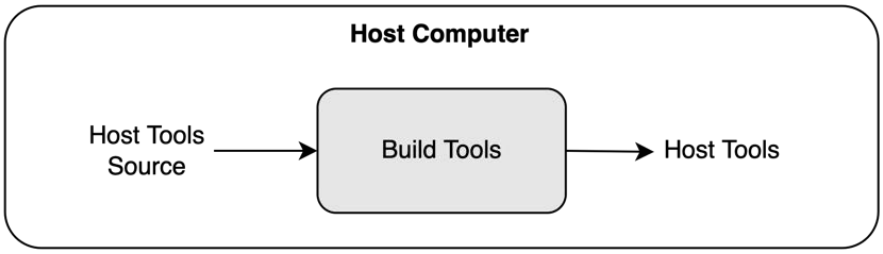


Figure 6. Build Tools Step

[5] For this section, you can think of your Docker image as a virtual machine snapshot that can share folders with your local file system. If you are unfamiliar with Docker, see Section 3.1.2 of the Getting Started Guide for an overview and install instructions. Additionally, Section 1 of the Technical Specifications document goes into detail on the Docker architecture used for this competition.
[6] If you use Python or another interpreted language for your host tools, you will only need to copy the scripts over, but the build tools step provides an opportunity to build host tools written in compiled languages.

3.1.3 Build Deployment

Third, you will create a deployment that represents a single instance or use of your PARED system. During this step, a Makefile is invoked that may generate secret data required for provisioning the system and store it in one or more files in a volume called the Host Secrets. This may include cryptographic key material, seeds, entropy, or any other data needed to build other devices. All cars and fobs within this deployment will share the same Host Secrets. **Attackers will never be given access to the Host Secrets generated in this step**.

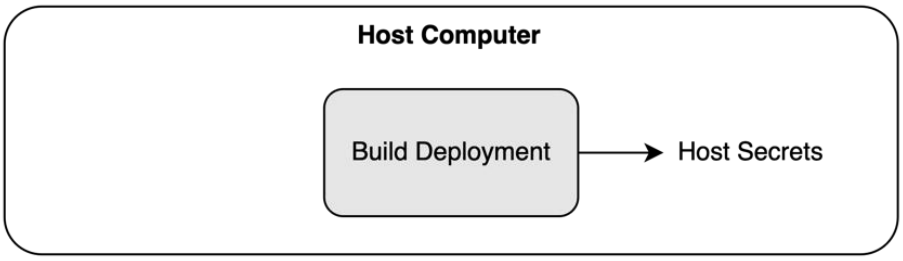


Figure 7. Build Deployment Step

3.1.4 Build Unpaired Fob, Paired Fob, and Car

In the final three build steps, you will build binary files containing the complete programs for the unpaired fobs, paired fobs, and cars that will run on the microcontrollers. These build steps may read and modify the Host Secrets to provision cryptographic or other secret material into the binaries. In addition, these steps must create EEPROM files associated with each target (unpaired fob, paired fob, car) that contain additional data or secrets that the firmware may need to operate correctly. Any data stored in these files are loaded into the appropriate EEPROM when your system is loaded onto the microcontrollers. **The plaintext firmware and EEPROM files produced in these steps are not given to attackers *except for the car and paired fob of Car 0 that will not contain any flags***.

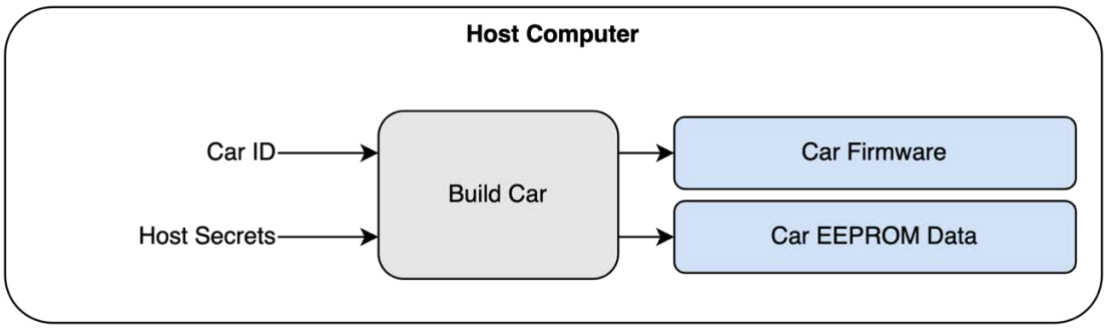


Figure 8. Build Car Step

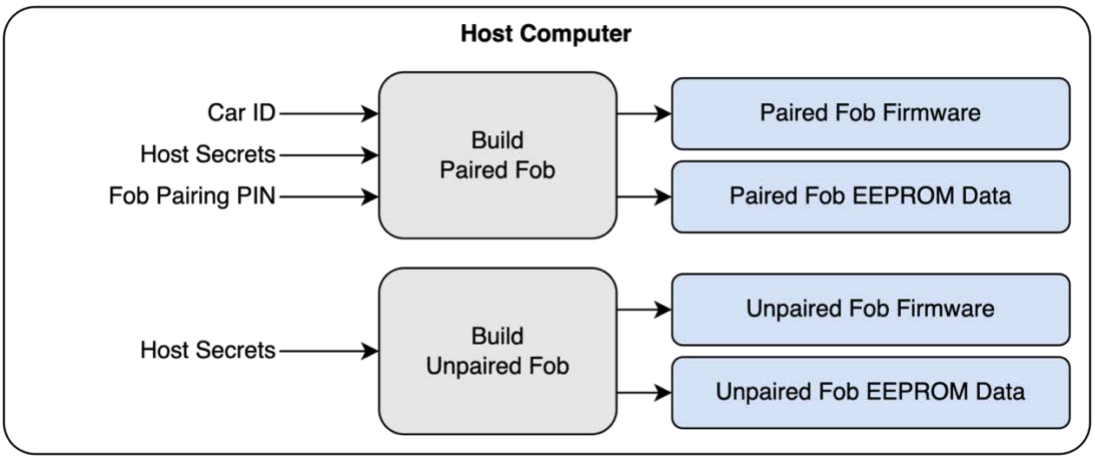


Figure 9. Build Unpaired Fob and Build Paired Fob Steps

3.2 Load Devices

After building the system, the firmware and EEPROM contents are loaded onto the microcontrollers. Teams will be provided with the tools to load their designs. **Teams will not be able to modify any part of this step**.

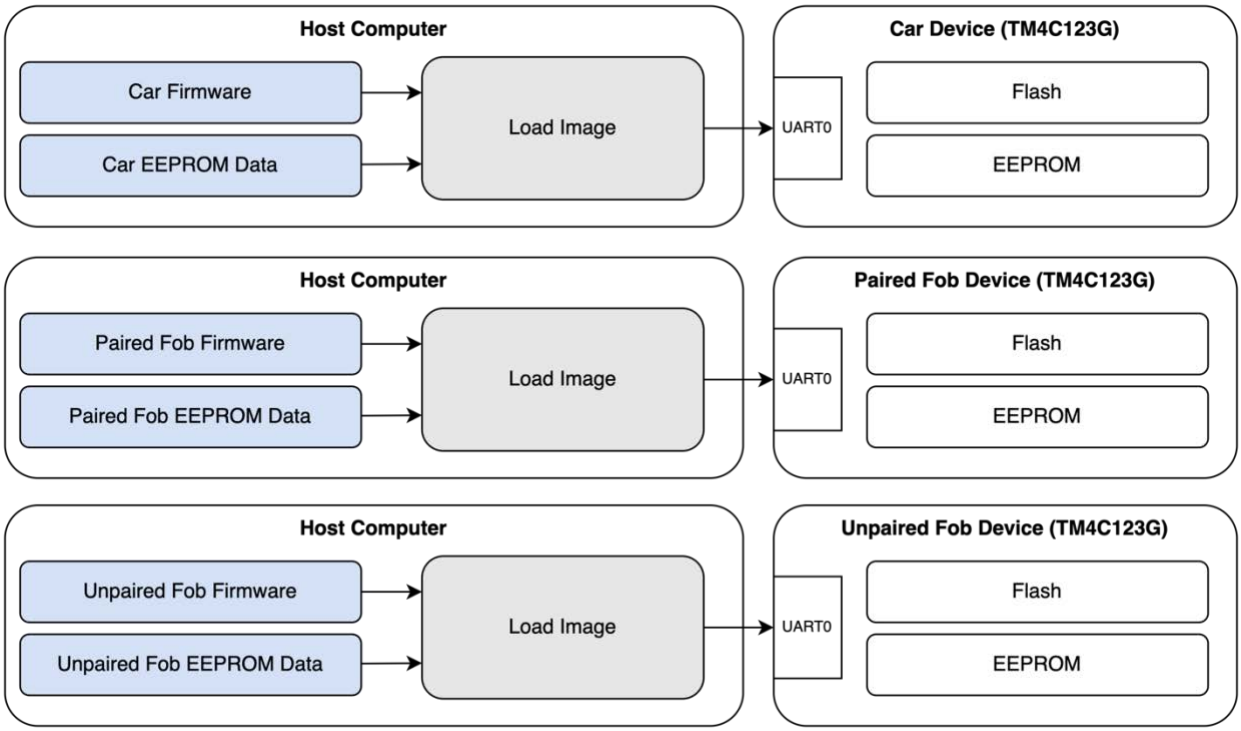


Figure 10. Device Load Step

3.3 Host Tools

The host tools are executable files that are run by the car owner in an insecure environment. Your PARED system will produce four host tools, summarized in Figure 11 below: Package Feature, Enable Feature, Pair Fob, Unlock and Start Car. The following sections in this document elaborate on the functionality of these host tools, and Section 3 of the Technical Specifications document discusses the inputs and outputs to these tools in more detail.

Figure 11. An overview of available tools for your PARED system

3.3.1 Package Feature

The package feature tool creates a binary file that will later be used to configure a fob. This tool will be given the feature number to package and the car number the feature is intended for. It will be able to read and modify the Host Secrets. **Attackers will be given access to the packaged feature produced in this step in many scenarios.**

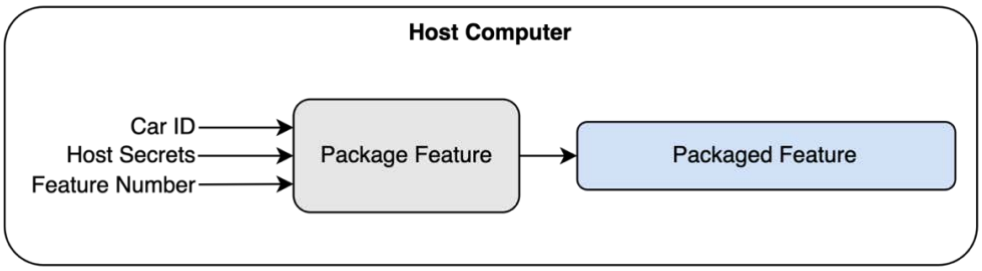


Figure 5. Package Feature Step

3.3.2 Pair Fob

The pair fob tool pairs an unpaired fob with a specific car when provided the correct pairing PIN and a fob that is already paired with that car. After pairing, both keys should be able to unlock the car. This tool will not have access to Host Secrets.

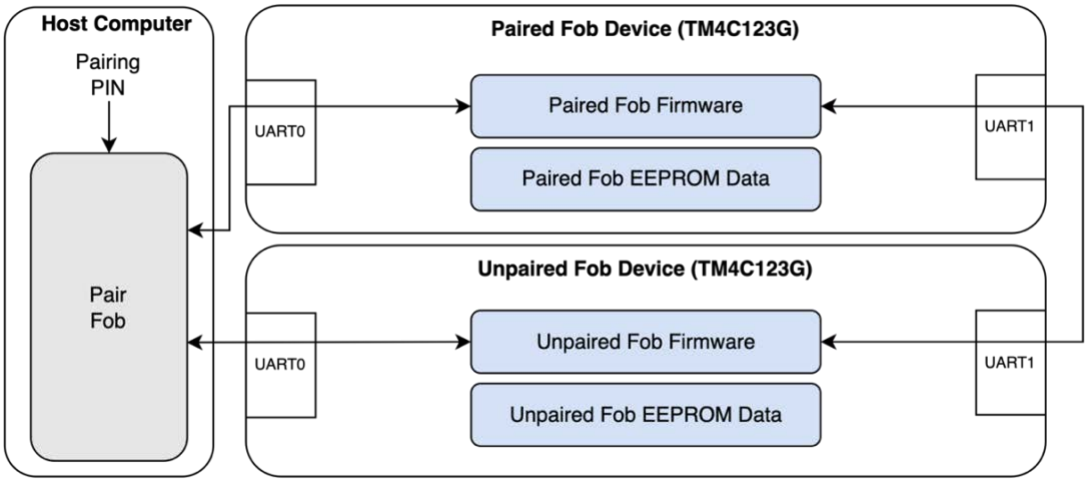


Figure 6. Pair Fob Step

3.3.3 Enable Feature

This tool sends a packaged feature to the paired fob device. Once the paired fob has the feature enabled, the car should recognize it when that fob is later used to unlock and start the car. This tool will not have access to the Host Secrets.

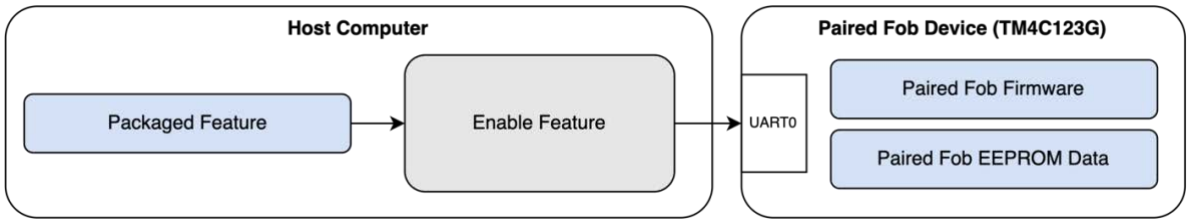


Figure 7. Enable Feature Step

3.4 Unlock and Start Car

Pressing the SW1 user button on a paired fob board will trigger communication with the car device to unlock and start the car. When unlocked, the car should print out an unlock message as well as the list of each enabled feature. These messages will be provisioned on the board by the eCTF organizers at the time your system is loaded. This tool will not have access to Host Secrets.

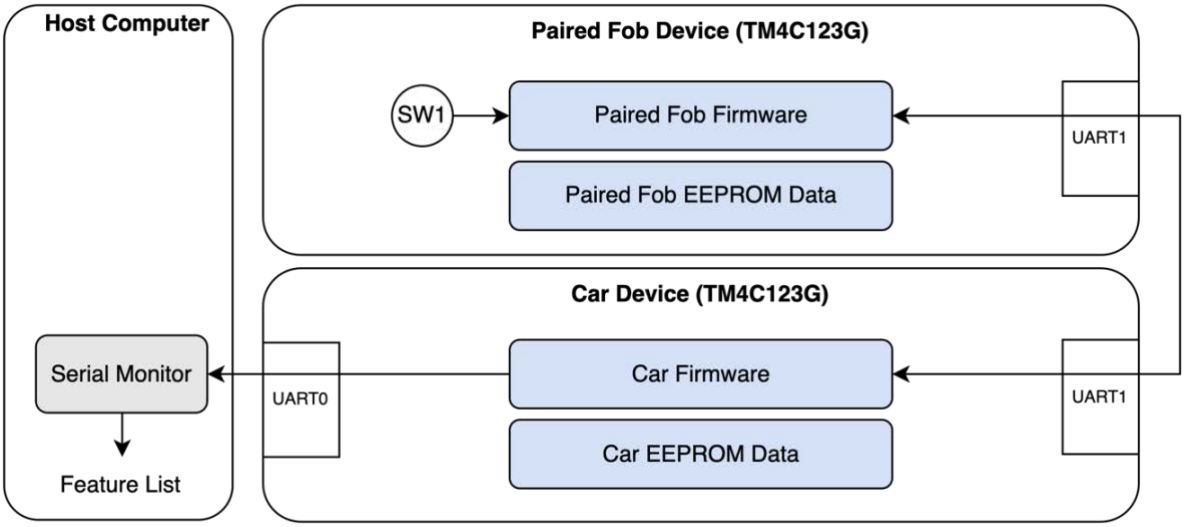


Figure 8. Unlock Car Step

4 Security Requirements

This section defines the security requirements of your design. **These properties will not be tested or evaluated during Handoff**. Instead, other teams will earn points for identifying and exploiting failures to properly meet these requirements by capturing Attack Phase Flags during the Attack Phase. Use these requirements to inform your design process, identifying and protecting critical data and code paths.

SR1: A car should only unlock and start when the user has an authentic fob that is paired with the car

Although it may seem obvious, you don't want anyone to have access to the car when they don't have the right fob.

SR2: Revoking an attacker's physical access to a fob should also revoke their ability to unlock the associated car

If you lend your car to someone (e.g., a friend or a valet), they will have physical access to your paired fob. Attackers shouldn't be able access to your car even after they have returned the original fob.

SR3: Observing the communications between a fob and a car while unlocking should not allow an attacker to unlock the car in the future

The communications between a fob and the car would be visible to attackers, so snooping attackers shouldn't be able to unlock your car later.

SR4: Having an unpaired fob should not allow an attacker to unlock a car without a corresponding paired fob and pairing PIN

Without both the paired fob and associated PIN, an attacker should not be able to pair a new fob with the car. Otherwise, an attacker with just the fob or just the PIN could make their own fob. Additionally, an attacker with access to the fob should not be able to determine the PIN, as they would then have both components to properly pair a second fob.

SR5: A car owner should not be able to add new features to a fob that did not get packaged by the manufacturer

These features must not be able to be tampered to prevent an owner from adding a feature they have not paid for and to prevent an attacker from escalating privileges on a feature-limited fob.

SR6: Access to a feature packaged for one car should not allow an attacker to enable the same feature on another car

For example, if your neighbor paid for an upgraded feature and the manufacturer sends them the upgrade, you should not be able to add that feature to your car without requesting it from the manufacturer.

5 Attack Phase Operation

The Attack Phase functionality will be implemented by the eCTF organizers. The organizers will provision a different car and set of paired fobs for each flag that is capturable by attackers. When creating the deployment, the following steps will be followed:

1. Build the environment producing a Docker image for your system
2. Build the host tools producing a package teams will use to interact with your design
3. Build the deployment producing system-wide secrets that can be used in the build process
4. Build one unpaired fob
    ⇒ This fob is the same for all cars. It can be paired for any of the cars if the attacker has the pairing PIN and a paired fob.

Then, for each of the different cars, the organizers will:

1. Package Feature 1 for the car
2. Package Feature 2 for the car
3. Build the car
4. Build the paired fob
    ⇒ Pairing PIN randomly generated by organizers
5. Protect the images to send to each of the attacking teams
    ⇒ Each team will have boards with secure bootloaders on them for use during the attack phase. The organizers will protect the raw binaries produced by the build steps before sending them to the attacking team, **except for the car 0 binaries**. These protected images will be encrypted and are not readable by attackers. With the secure bootloader installed on the microcontroller, attacking teams will be able to load the protected binaries using a tool provided by the organizers. ***Note: Attacking the secure bootloader is off-limits for this competition and teams found doing so will be disqualified from the competition.***

✓ = Full Access ✓ = Temporary Access	Car	Paired Fob	Unpaired Fob	Logic Analyzer Capture of Unlock	Pairing PIN	Packaged Feature 1	Packaged Feature 2
Car 0 Your Car (No Flags)	✓	✓	✓		✓	✓	✓
Car 1 New Car	✓		✓			✓	✓
Car 2 Temporary Fob Access	✓	✓	✓			✓	✓
Car 3 Passive Unlock	✓		✓	✓		✓	✓
Car 4 Leaked Pairing PIN	✓		✓		✓	✓	✓
Car 5 PIN Extraction, Enable Feature	✓	✓	✓			✓	

Figure 9: Attack Phase Cars

5.1 Attack Phase Cars

For each car, the attackers will have access to different resources:

Car 0: Your Car

The attacking team is provided with:

- Plaintext car binary
- Plaintext paired fob binary
- Protected car binary
- Protected paired fob binary
- Protected unpaired fob image
- Pairing PIN

The attacker has full access to this car, including the unprotected images for the car and the paired fob. This car will not have any flags for the attacker to capture, as it represents a car that was purchased from the manufacturer.

Car 1: New Car

The attacking team is provided with:

- Protected car binary
- Protected unpaired fob binary

This car can be unlocked by compromising SR1 and doing so will release the new car unlock flag (see Attack Phase Flags).

Car 2: Temporary Fob Access

The attacking team is provided with:

- Protected car binary
- Protected paired fob binary (temporary, see below)
- Protected unpaired car binary

This car will only release a dummy flag during unlocking unless the paired fob device has been disabled using the secure MITRE bootloader. At that point, the car can be unlocked by compromising SR2 and doing so will release the temporary fob access flag (see Attack Phase Flags).

Car 3: Passive Unlock

To provision this car, the organizers will perform the following steps after building and protecting the images.

1. Load the car and paired fob images on two keyed boards
2. Record the UART communications during three unlock transactions between the paired fob and the car using a logic analyzer[7]

The attacking team is provided with:

- Protected car image

[7] If you can prove that you need a reasonable number of additional traces for an attack, the organizers can provide you with more.

- Protected unpaired fob image
- Recording of the UART communications during multiple unlock transaction

This car can be unlocked by compromising SR3 and doing so will release the passive unlock flag (see Attack Phase Flags).

Car 4: Leaked Pairing PIN

The attacking team is provided with:

- Protected car image
- Protected unpaired fob image
- Leaked pairing PIN

This car can be unlocked by a compromise of SR4 and doing so will release the leaked pairing pin flag (see Attack Phase Flags).

Car 5: PIN Extraction

The attacking team is provided with:

- Protected car image
- Protected paired fob image
- Protected unpaired fob image
- Feature 1 packaged for Car 5
- Feature 2 packaged for Cars 1-4

This car will not release any flags from an unlock. However, three flags will be available to capture from this system. The first flag available is PIN Extraction, which proves the compromise of SR4. If you recover the pairing PIN, then you can submit it to the scoreboard. The other two flags will be awarded for enabling features on the car. Feature 1 is already packaged for this car, so enabling Feature 1 will not reveal a flag. However, enabling Feature 2 will release a flag during an unlock, proving the compromise of SR5 and/or SR6.

5.2 Additional Materials and Information Available to Attackers

For each car discussed above, the additional files below will be made available to attacking teams.

- All source code (with the .git directory removed)
- The most recent documentation provided to the eCTF organizers
- The built host tools for interacting with your design
- Car ID Numbers
- Feature Numbers

6 Scoring

Any flag submitted to the scoreboard will be of the form:

```
ectf{<flag name>_<16 hex characters>}
```

For example, the “Temporary Fob Access” flag could look like:

```
ectf{tempfobaccess_0123456789abcdef}
```

6.1 Design Phase Flags

To encourage teams to stay on schedule during the Design Phase, and to give the organizers insight into each team’s progress, points will be awarded for reaching certain milestones. Each design-phase flag has a deadline date at which point it can no longer be submitted for points. See the table below for details.

MILESTONE	FLAG FORMAT	DESCRIPTION	DUE DATE
Read Rules	ectf{readrules_*}	If you read **all** the rules, you’ll know	01/25/23
Boot Reference Design	ectf{bootreference_*}	Provision and boot the example design to receive a flag (see the README)	02/01/23
Design Document	ectf{designdoc_*}	Submit an initial design document containing high-level descriptions of how each host tool and system function will work. It should be clear from your descriptions how your system meets the functional and security requirements. Your design **may** change after submitting this draft, but we recommend updating this as your design changes and including it in your final submission for documentation points.	02/03/23
Submit Reference Design for Testing	ectf{testreference_*}	Submit the reference design for testing. Learning how to use the testing infrastructure can help you validate your design continues to meet functional requirements as you develop security features.	02/08/23
Final Design Submission	ectf{attackphase_*}	The Attack Phase opens on March 1. Teams should submit their completed design for testing before or on July 13 to enter the attack phase on time.	03/01/23 – 04/19/23
Bug Bounty		See Section 6.1.2 below	

6.1.1 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 distributed on a first come, first serve

basis. Sometimes whether an issue is truly a bug (or a feature!) is a matter of opinion - the eCTF organizers reserve the right to reject bug reports for trivial issues and combine multiple similar reported bugs into one. Submitted bugs will be accepted if there is a violation of the functional requirements in the reference design that prevents it from working correctly. Submissions for typos or clarifications on documentation provided by the organizers will not be considered for additional points. However, we appreciate being notified about these mistakes so we can make the appropriate edits and provide further explanation where it is necessary.

6.2 Attack Phase Flags

Each provisioned system in the Attack Phase holds several "flags" that are only revealed if one or more security requirements are compromised. By submitting flags, an attacking team demonstrates that they have compromised security requirements of the target system. For each attack that results in a flag submission, teams must provide a brief description of how they accomplished the attack; failure to do so may result in the revocation of points. The flags are defined in the table below.

FLAG	FORMAT	DESCRIPTION	ACCESS
New Car Unlock	ectf{newcar_*}	Unlock a new car you don't have the fob for	Car 1
Temporary Fob Access	ectf{tempfobaccess_*}	Unlock a car you previously had the fob for	Car 2, Temporary access to Fob 2
Passive Unlock	ectf{passiveunlock_*}	Unlock a car you intercepted an unlock transaction for	Car 3, Logic Analyzer Capture for Car 3 unlock
Leaked Pairing PIN	ectf{leakedpin_*}	Unlock a car you have the pairing PIN for	Car 4, PIN 4
PIN Extract	ectf{pinextract_*}	Extract the pairing pin	Car 5, Fob 5
Enable Feature	ectf{feature2_*}	Enable Feature 2 on Car 5 when you have Feature 2 packaged for Car 0 - 4	Car 5, Fob 5, Feature 2 for Car 1

6.2.1 Flag Point Values

Since the vulnerabilities to be discovered in the Attack Phase come from other teams' unintentional flaws in the Design Phase, the scoring system is designed to adjust points based on estimated difficulty of capture. More specifically, the point value of any given flag will be adjusted dynamically and automatically based on multiple factors:

- If multiple teams capture the same flag, then the value of that flag will be divided among all the teams that capture it (distribution is not equal – it is weighted based on time of capture to provide more points to earlier captures). Naturally, more difficult attacks will be executed by fewer teams and therefore rewarded with more points.
  Note: *Your total score will drop each time another team captures a flag that you have already captured. This is because the flag points that you were initially awarded need to be redistributed as additional teams capture the same flag.*
- The number of points a flag is worth increases over time as it remains un-captured. This will make the difficult flags more and more appealing as the competition goes on.
- To discourage teams from "hoarding flags" without submitting them, the first capture of each flag will earn the attacker 50 bonus points in addition to retaining the largest share of the flag if also captured by other teams.

6.3 Defensive Points

Defensive points will be automatically awarded over time for each uncaptured flag, beginning once a team successfully completes Handoff and enters the Attack Phase. Once another team captures one of your team's flags, that flag will no longer gain additional defensive points. The earlier your team enters the Attack Phase; the more potential defensive points are on the table.

6.4 Documentation Points

Good documentation will be rewarded to discourage security-by-obscurity. "Good documentation" includes clear and well-commented code, useful descriptions of modules/functions/classes, clear and accurate design documents, and other documents that clearly describe how to read or approach the entire code base.

We are not looking for lengthy documents that describe your implementation in excruciating detail. A concise and clear README.md, including a brief rationale for your security features, combined with well-structured and well-commented code is sufficient for Max points. Quality is valued over quantity.

The maximum number of points that can be scored for documentation is equal to the value of an uncaptured attack flag scored on the last day of the competition, with the actual amount being a percentage of that maximum:

- **Max** - Exemplary documentation, comments, and code structure; clear and easy to understand
- **75%** - Good comments and high-level documentation
- **50% -** Good comments, but lack of clear high-level documentation
- **25%** - Confusing code and little or no actual documentation
- **0%** - Confusing or deceptive comments and documentation

Points for documentation will not be awarded until the end of the Attack Phase. Honest feedback on documentation from other teams will be solicited and factored into the final point determination.

6.5 Posters

There will be an opportunity after the Attack Phase for teams to provide a "write-up" in the form of a poster for additional points. Further details on the content and format of the posters will be provided during the Attack Phase.

7 Rules

Most rules are described and explained throughout this document in the earlier sections. This section is intended as a concise summary of the most important rules.

(1) In addition to the rules provided by MITRE, participants must also adhere to all local laws and the policies and procedures stipulated by their local organization/university.

(2) MITRE reserves the right to update, modify, or clarify the rules and requirements of the competition at any time, if deemed necessary by the eCTF organizers.

(3) When submitting your secure design, all source code and documentation must be shared.
- This is to discourage security-by-obscurity, as well as to accelerate attack development and encourage more sophisticated techniques for both sides.
- Creating any part of your submission in an obfuscated manner or using an esoteric programming language is considered security-by-obscurity and is not allowed. Please contact the organizers if you have any questions about this.

(4) You may only attack the student-designed systems explicitly designated as targets, and such attacks may only occur when you are in the “Attack Phase” of the competition.
- Any MITRE provided tools or infrastructure is NOT in scope for attack.
- A secure MITRE bootloader will be running on the physical device that securely loads attack-phase designs – **this bootloader is NOT in scope for attack.**
- If you have any question about whether a component is “in scope”, please reach out to the organizers for clarification.

(5) All flags must be validated by submitting a brief description of the attack.
- Attack descriptions should be sufficiently detailed to allow the defender to correct their vulnerability. eCTF admins may remove points awarded for capturing flags that are not validated before the completion of the eCTF.

(6) Flag sharing across teams is not permitted and will result in immediate disqualification.

(7) No permanent lockouts are allowed. See the Technical Specifications Document for timing and performance requirements.

(8) If you’ve read all the rules, then you can prove it by providing the MD5 hash of this document (Version 1.0 PDF file) as a flag on the scoreboard (ignore the standard flag format and submit the full hash value).

(9) Your system must work with the platform that was provided. Switching to a different platform during the Design Phase is not allowed.

(10) All documents that are submitted (e.g., design document and write-ups) must be provided in PDF format.

8 Frequently Asked Questions

Is it OK to obfuscate our source code to make it more challenging to understand and attack?

No. Obfuscations performed at compile-time (e.g., to make binary reversing more challenging) are OK, but your source code needs to be written in a clear and maintainable fashion. It should be well-commented and/or otherwise documented clearly. Using programming languages purely for their difficulty to understand is considered obfuscation and therefore using a language other than the below list of pre-approved languages must be approved by the organizers:

- Python
- C/C++
- Rust

Can we add intentional delays during boot to make it more difficult for an attacker to collect large numbers of observations?

There should not be any intentional delays that could push performance past the timing requirements in the Technical Specifications Document.

If your system detects that it is under attack, additional delays are OK, but must be limited to no more than 5 seconds. Permanent lockouts or self-destruction is not allowed.

Can we attack another teams' development environment?

No! Everything other than the provisioned devices, the host tools, and the firmware and configuration images are considered out-of-bounds. In other words, there is **nothing** that you can attack until your team enters the Attack Phase.

Is social engineering in-scope for this competition? Can we send phishing communications to other teams to trick them into revealing their secrets?

No, please don't do this. Keep your attacks technical. We love creative ideas, but this one can easily violate state and federal laws.

Can we submit the reference design or a design with security that can be trivially defeated so we can move into the Attack Phase?

No. As this is a design-build-attack-style competition, teams must submit a design that exhibits significant effort on the design and build components. It is up to the discretion of the eCTF organizers as to what level of modification counts as "significant effort", so please contact the organizers before submitting an extremely pared down version of your design. ***Note:*** *you may submit designs up to the last day of the competition – and there have been very successful teams that did not submit on time – so don't panic if your design isn't ready on the first day of Handoff.*

Can we attack MITRE infrastructure or files that have been protected by MITRE for secure distribution of provisioned systems?

No. Any infrastructure that has been created by MITRE is off-limits for this competition. The eCTF organizers put these capabilities in place to make the competition smoother for everyone and should be considered transparent when attacking provisioned designs. ***Note:*** *when submitting Attack Phase flags to the scoreboard, the attacking team must submit a brief*

summary of how the flag was captured to the eCTF organizers. If capturing a flag involved any tampering with MITRE infrastructure the flag points will not be awarded.

What is a "provisioned system"?

A provisioned system is a set of files and devices provided by the eCTF organizers for use during the Attack Phase. In this competition, the provisioned system includes a set of binaries (protected by the organizers) that can be loaded with the secure MITRE bootloader on a physical microcontroller to run a team's design.