Challenge Description and Rules:

Secure Entertainment System

MITRE

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

Table of Contents

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

1 Challenge Overview

You are part of a design team tasked with building the first prototype of a next generation gaming console. Your employer (a large entertainment company) has outlined a set of requirements for the system and has chosen the development board that they want you to use, but the rest of the design is up to you.

Your challenge is to design and implement the software, programmable hardware (optional), and protocols for the next generation secure gaming console: The MITRE Entertainment System (MES).

Your employer wants to emphasize security in this new gaming console. The game studios which partner with your employer expect that their game binaries will be safe from Intellectual Property (IP) theft and that only users who have purchased a game are authorized to play it.

News & Analysis

Sony Sues Hacker for Selling Jailbroken PS4s on eBay

Sony bought two of the jailbroken consoles sold via eBay, allegedly by a California man. Bundled with one of the consoles was instructions on how to run the 'exploit' software code to break Sony's anti-piracy protections, the company's lawsuit claims.

By Michael Kan October 9, 2018 2:18PM EST

Figure 1. Game console piracy makes headlines!
https://www.pcmag.com/news/364298/sony-sues-hacker-for-selling-jailbroken-ps4s-on-ebay

There are several threats that the entertainment company has identified and wants to protect against:

- Users accessing accounts that they do not own
- Users injecting malicious code onto the device in order to modify and/or reveal the game binaries
- Users rolling-back installed game updates (e.g., to exploit a previously existing vulnerability)

The system that your team creates must meet the requirements specified in this document and defend against as many attacks as you (and your opponents) can think of. You must design and implement a functional gaming system which includes support tools to protect games and build, provision, and deploy your system. Once your system is completed, it will be subjected to attacks from opposing teams, while you get a chance to attack the designs from the other teams.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

1.1 Competition Phases

This is an attack-and-defend capture-the-flag, meaning there are phases of the competition for both attack AND defense, as summarized in the figure below.

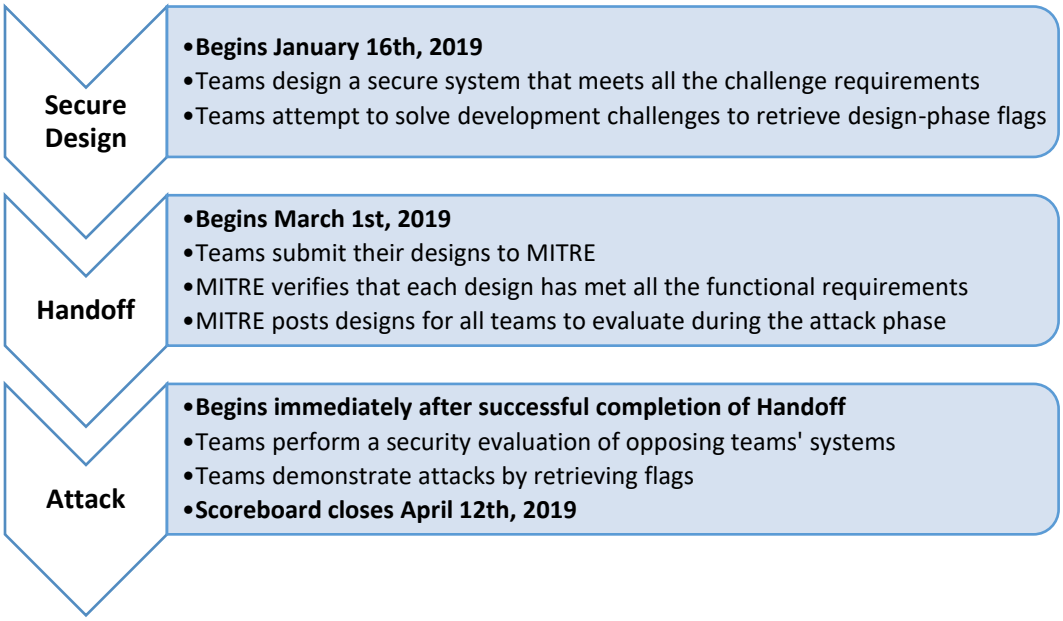

Figure 1. eCTF Phases

The Handoff phase is completed when your team submits a system that passes MITRE's requirement tests. Therefore, the date and time of transition between phases will vary between teams. Each team is allowed into the Attack phase by the eCTF organizers as soon as that team completes Handoff. 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 Team-A will move to the Attack phase, while Team-B remains in the Handoff phase until they resubmit their system with the necessary fixes to pass the tests.

There are significant scoring advantages to entering the Attack phase as soon as possible. We highly recommend setting a schedule for your design and implementation and sticking to it as closely as possible. If your schedule starts to slip, your team will need to make hard decisions about what security features can be excluded to submit a working design on time (or at least as soon as possible). Final testing always takes longer than expected and sometimes reveals tricky problems – so plan to start your final testing earlier than you think is necessary.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

2 Provided Materials (The Kit)

Building a secure gaming system from scratch is a daunting task; therefore, MITRE is providing a full example system (both hardware and software) that meets the functional requirements for the competition. You can use this system for testing and as a reference to help understand the requirements. You may also use it as a base to build on for your own system. Beware, the reference implementation, without significant security modifications, will not survive the attack phase. See Insecure Example Implementation.

2.1 Hardware

MITRE will provide each team with a set of hardware and peripherals required for the competition. Teams may choose to obtain additional units of the hardware to increase testing and attack development capabilities for their team. However, your system must work with the hardware that was provided. Switching to a different platform or modifying the physical hardware during the design phase is not allowed.

Figure 2. Hardware components that were provided for all teams

Number	Component
1	2x Arty Z7-10
2	1x Tripp-Lite SD card reader
3	2x Micro-USB Cable
4	2x 8 GB MicroSD Card
5	2x MicroSD Adapter

Table 1. List of hardware components

2.1.1 The Chip — Zynq-7000 / ARM Cortex A9

The chip we will be using for this challenge is the Xilinx Zynq-7000 System on a Chip (SoC) which combines a dual-core ARM Cortex-A9 based processing system (PS) with Xilinx programmable logic (PL). While the PL can provide teams with unique design opportunities, modifying the hardware design (i.e. the PL) is not required by the competition. Check out the Datasheet for the chip.

2.1.2 Restriction: Hardware AES/HMAC Engine

You will inevitably read through Section 32: Device Secure Boot in the Zynq-7000 Technical Reference Manual (TRM). This section details the hardened AES/HMAC Engine which resides in the PL. This engine is NOT available for use by your design and will be disabled by the MITRE-provided First Stage Boot Loader (FSBL) which loads your code.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

2.1.3 eFUSE Settings

The two boards that you receive in your kit will be used for both development during the design phase AND exploitation during the attack phase. For this reason, appropriate eFUSE settings have been selected and burned at MITRE and will not be modifiable.

2.2 Code Repositories

Git repositories are provided that contain source code for an insecure example system that meets all the challenge requirements. Additionally, a repository is provided that contains instructions for setting up the required development environment and provisioning the example system. A summary of these is provided below:

- MES/
  - Arty-Z7-10/
    - *PetaLinux Project which builds U-Boot, Kernel, and FS*
  - Arty-Z7-10-hardware/
    - *Vivado Project which synthesizes the hardware configuration*
  - tools/
    - *Provision tools as described in Tool Descriptions and APIs*
- vagrant/
  - *VagrantFile and supporting files for provisioning the development VM*

2.2.1 Development Environment

Along with the example system we are also providing a Vagrantfile[1] with all the tools necessary to build and test the example. When you submit your design, you will need to include an updated customizations.sh and config.rb with all of your dependencies so that we can build and test your implementation. You may not modify any of the other vagrant files or change the folder structure. The submitted Vagrantfile should launch a VM and build any tools necessary to run or test the design. Please refer to the documentation included in the git repository for more information.

2.2.2 Insecure Example Implementation

The example implementation is split into two main components: a PetaLinux project, which contains the source used to build the system's software components and a Vivado project, which contains the hardware description used to synthesize the FPGA bitstream. The Vivado project is included in the main repository as a submodule because it is not necessary for teams to modify it, and the repository itself is rather large. Should teams be interested in modifying or adding FPGA hardware, please refer to the hardware design git repository.

NOTICE: If teams choose to modify the programmable logic, please make sure to check that you still meet the requirements specified in FPGA Hardware.

WARNING: The example system is insecure. The purpose of providing it is to give participants a reference to help in understanding the functional requirements. There was little or no effort placed on securing this system – that's your job! You may use the example system as a base and add your security features on top, or you may choose to start from scratch so that security is designed in from the start.

[1] https://www.vagrantup.com/about.html

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

2.3 First Stage Boot Loader

A boot image (BOOT.BIN) will be provided that has been signed and encrypted for use with the MITRE-provided boards in the Provided Materials (The Kit). This boot image contains a First Stage Boot Loader (FSBL) which will load and boot the gaming system image (MES.BIN) on the MITRE-provided boards.

If your team purchases additional hardware, you will need to rename MES.BIN to BOOT.BIN for the device to boot. A summary of the two boot scenarios is outlined below:

Booting a MITRE-provided Board

```
SD Card Partitions:
    boot/
        BOOT.BIN – provided by MITRE
        MES.BIN – built by you
    games/
        game-v1.0
```

Booting a purchased Board

```
SD Card Partitions:
    boot/
        ~~BOOT.BIN – provided by MITRE~~
        BOOT.BIN ← MES.BIN – rename MES.BIN to BOOT.BIN
    games/
        game-v1.0
```

2.4 Important Details

2.4.1 MES.BIN

MES.BIN is the gaming system boot image that your build system must generate. Your build system must use Xilinx's BootGen tool to generate MES.BIN and it must contain the following partitions, specified by your `SystemImage.bif`, in the following order: `zynq_fsbl.elf`, `Arty_Z7_10_wrapper.bit`, and then your images. The MITRE-provided BOOT.BIN will only boot MES.BIN boot images that have been created with this procedure.

2.4.2 MES.BIN Encryption and Authentication

During the Secure Design Phase, the MES.BIN provisioned by your tools will be unprotected as the AES and RSA hardware will be inaccessible to you. When images are distributed in the Attack Phase, the boot image will be signed and encrypted by organizers with device-specific keys. These keys are not accessible to participants. This secure boot image, MES.BIN, may include any binary data except games provisioned with the game provisioning tool (e.g. the reference design includes a u-boot image, a kernel image, and a read-only filesystem in MES.BIN).

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

3 Secure Design Phase

The secure design phase encompasses the design of a secure gaming console system and the creation of supporting tools. The following sections provides the requirements for your system and support tools.

3.1 Game Console System Description

3.1.1 MITRE eCTF Shell (mesh)

The core functionality of the gaming console system is provided through a custom shell that you will need to design. The insecure example reference implements this shell within U-Boot. You may modify this to improve its security or build your own from scratch, but every submitted shell must meet the specifications outlined in the following sections.

3.1.1.1 Shell Format [update v1.1]

The shell **must** prompt for commands using the reference prompt `mesh> `. This may only be displayed when the MESH shell is ready for user input.

3.1.1.2 Login Format [update v1.1]

When there is no user currently logged in, the shell will continuously prompt for a username and then a password until valid credentials are entered.

The following procedure must login a valid username/PIN combination:

1.) Boot board
2.) Enter username
3.) Press enter
4.) Enter PIN
5.) Press enter

3.1.1.3 Commands [update v1.1]

Each shell **must** have the following commands (plus any additional ones you would like to add):

Game	Usage	Function
help	help	List all valid commands
shutdown	shutdown	Exit the shell and reset the board
logout	logout	Log out of the mesh login and allow someone else to login
list	list	List all games available for the current user to play.
play	play <game>	Launch the specified game. ~~The shell should resume once the game has been terminated (device reboot **is** allowed to return to the shell).~~ Note that this means that control should be handed off to Linux for the game to be played, and the board should be in a halted state once the game is done being played.
query	query	List all games available to download (note query does not need to check if the game is valid for the current user).
install	install <name>	Install the game, such that a play <name> will successfully launch the new game for the logged in user. Your system must be able to install any game correctly created with the provisionGame.py tool and corresponding factory secrets file for the given user. Your system must be able to install a minimum of 128 different games. See Game Installation and Storage for details.
uninstall	uninstall <name>	Uninstall an installed game for the logged in user

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

3.1.1.4 Command Format

Each command **must** have the following output format.

Command	Input Format	Output Format
help	N/A	Each command must be printed as it is listed in the `Command` column of this table. The output may include more information such as usage and be formatted in any way but the output must at a minimum include the commands as specified.
shutdown	N/A	N/A
logout	N/A	N/A
list	N/A	The name of each game is printed as it is specified in games.txt when the game is provisioned. The output may include more information such as version and be formatted in any way but the output must at a minimum include the games as specified.
play	<game> is the filename of the provisioned game as defined by the output requirements for provisionGames.py	N/A
query	N/A	The name of each game is printed as it is specified in games.txt when the game is provisioned. The output may include more information such as version and be formatted in any way but the output must at a minimum include the games as specified.
install	<name> is the filename of the provisioned game as defined by the output requirements for provisionGames.py	N/A
uninstall	<name> is the filename of the provisioned game as defined by the output requirements for provisionGames.py	N/A

3.1.1.5 UART Interface

All commands in 3.2.1.1 must work over the UART port. You *may* enable an external keyboard for interacting with your system, but this will only be an additional feature and cannot replace the UART interface. In addition, all designs must allow provisioned games to write to the UART at `/dev/ttyPS0` configured with a baud rate of `115200`. That is, when a game is provisioned and run using the `play` command in 3.2.1.1, any writes to `/dev/ttyPS0` by the game can be read over UART at a baud of `115200`.

3.1.1.6 Default Account

All systems must have a `demo` account with the PIN `00000000`. This account will have default games installed on startup. This default account will **not** be specified in the user input file given to provisionSystem.py (Python 3.x).

3.1.1.7 Flash

Your system **must not** modify the last 64K (0x00FF0000 to 0x01000000) of external flash memory.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

3.1.1.8 Username, PIN, and Game Names

Your system must support the following length and character types for the usernames, PINs, and game names.

Parameter	Length	Allowable ASCII Characters
Username	1-15	a-z, A-Z, 0-9
PIN	8	0-9
Game Name	1-31	a-z, A-Z, 0-9

3.1.1.9 User Accounts

Your system must support at least 32 users.

3.1.1.10 Game Binary Size

Your system must support game binaries that are up to 3 MBs in size.

3.1.1.11 Number of Games

Your system must support at least 128 games.

3.1.2 FPGA Programmable Logic

All teams will be provided with a reference implementation that contains several important blocks. It is recommended that any changes you make are simply modifications to the reference; however, there are three key functions (specified in the following subsections) that must remain if you decide to modify the existing design further.

3.1.2.1 HDMI

As this is a gaming console system, it will need a display. All submitted designs must have functioning HDMI-OUT ports.

3.1.2.2 USB Keyboard

Gameplay requires a keyboard, and therefore submitted designs must have a functioning USB port, with appropriate drivers.

3.1.2.3 AXI Memory

A region of FPGA based memory will be used as a DRM check. All submitted designs must have read and write access to this memory region in order to be considered valid. The memory resides from 0x40000000 to 0x40001000.

3.2 Security Goals

3.2.1 Confidentiality

Games should be protected to prevent reverse-engineering or stealing of intellectual property. Anyone with access to the raw firmware could easily extract sensitive information such as proprietary algorithms or security mechanisms. To ensure confidentiality, the games should be protected throughout their lifetime (e.g., while at rest [stored on the SD card], while being transferred to the entertainment system). Only a user, whom the game has been provisioned for, on a valid entertainment system should be able to extract the game binary in order to install/play it.

3.2.2 Integrity and Authentication

Your system should contain some mechanism(s) for validating that a given game is from a legitimate source and was not altered in any way. Games or other software that fail integrity or authentication checks should not be installed or loaded; otherwise, nothing would prevent modified/malicious/invalid software from being executed on the device. Only protected software that is created by the deploy script should be accepted by the console system.

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

3.3 Functional Requirements

3.3.1 Game Installation and Storage

Submitted systems will need to be able to store provisioned games on a separate partition of the SD than that of the boot image. Games will be "installed" from this other partition, which will act as the equivalent of an installation disk. In addition, all "installed" games must be saved in non-volatile memory, i.e., rebooting the device should not remove all games that have been installed. The actual game binary does not need to be stored in non-volatile memory but rather some record of its installation.

3.3.2 Game Versioning [update v1.1]

All games should include a version number in their metadata; a user should not be able to install a lower version of a game that has been previously installed by that user (even if the newer version was then uninstalled). Note that this means versioning is user-specific. For example, if `2048-v1.0` is only provisioned for `user1` and `2048-v2.0` is only provisioned for `user2` then `user2` could install `2048-v2.0` and `user1` could STILL install `2048-v1.0`.

3.3.3 PL Memory Region Read/Write Access [update v1.1]

The system must have an AXI interface that can be written to and read from at memory addresses 0x40000000 to 0x40001000 by games provisioned for the system. This will be used as hardware check which some games will use to validate the system is legitimate. Gaining access to the value stored in this register will also be a flag. A memory device driver, mounted at /dev/mesh_drm, must be available to provisioned games which enables read/write access to 0x40000000 to 0x40001000.

3.3.4 Command Timing [update v1.1]

The system must adhere to the timing/performance requirements outlined below. Remember that if this were a real gaming system, it would need to be useable and enjoyable to users/customers, and nobody likes waiting for their console to boot up. If your system does not meet these Max Time requirements, it will not be accepted during Handoff. If your system performs *faster* than these times, you will be rewarded with bonus points (see section 6.4).

Operation	Max Time For Completion
Boot Up	20 Seconds
'Install' command	10 Seconds
'Play' command	20 Seconds
Login	5 Seconds
Total: Boot Up, Login, and 'Play'	40 Seconds

Table 1 - Max Times Allowed

3.3.5 Non-destructive operation

The system must NOT attempt to prevent attacks by making the system unusable, as this is not a valid technique for systems that need to be deployed to customers. Any method used to try and destroy the hardware (e.g., thousands of writes in rapid succession to flash) is not permitted. Designs that contain such countermeasures will not be accepted during Handoff.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

3.4 Support Tool Descriptions and APIs

In addition to the game console system, you must also design and implement four (4) support tools to:

1.) Build and provision system components **(provisionSystem.py)**
2.) Securely package, install, and run games **(provisionGames.py)**
3.) Package the system components into a boot image **(packageSystem.py)**
4.) Deploy your system and games on an SD card **(deploySystem.py)**

Examples of these support tools are provided in the insecure example. The example tools may be modified by your design team, however, the tools must conform to the requirements described in the following subsections.

3.4.1 The Provisioning Process

The provisioning process, shown below in Figure 3, begins with **provisionSystem.py** which takes two files (`Users.txt` and `Default.txt`) as input and creates several output files:

1.) `u-boot.elf` – the second-stage bootloader image
2.) `Arty_Z7_10_wrapper.bit` – the hardware configuration bitstream
3.) `image.ub` – the linux kernel and filesystem
4.) `SystemImage.bif` – the boot image description
5.) `FactorySecrets.txt` – the file used for communicating information between the system provisioning and game provisioning processes.

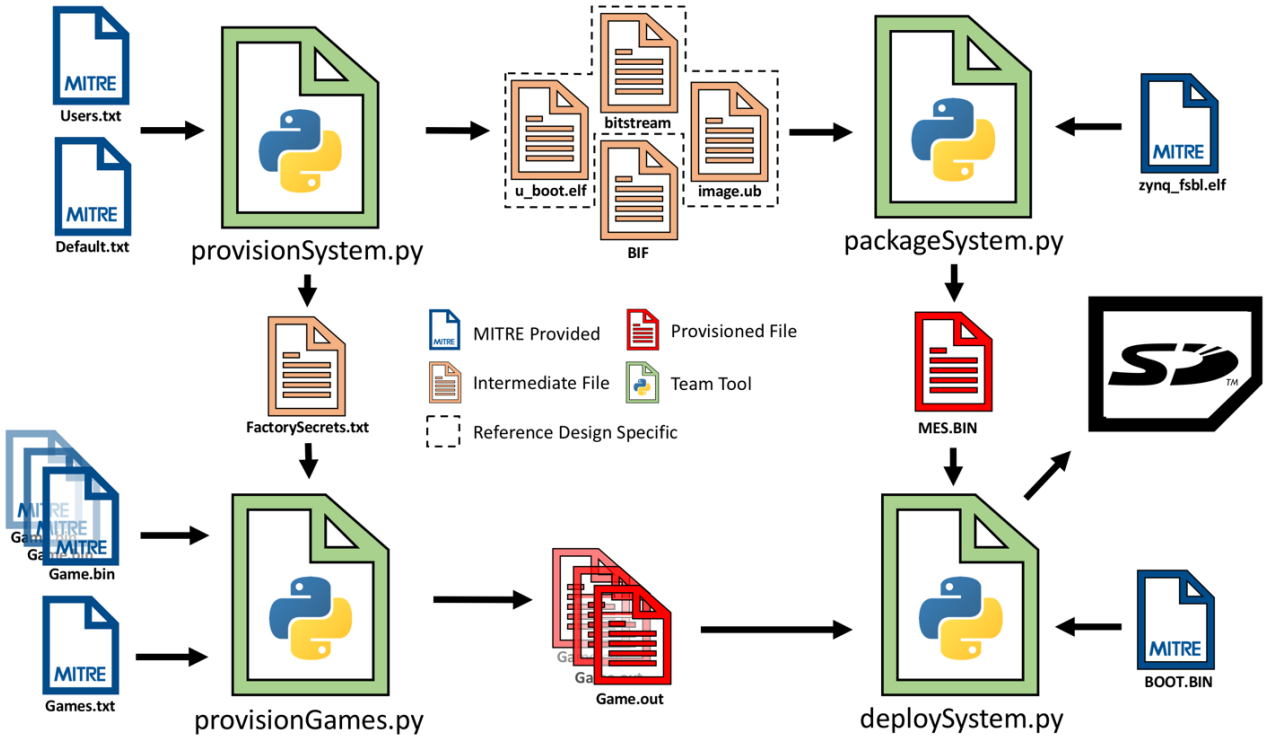

Figure 3: High-level functionality of support tools

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

The **provisionGames.py** script is then used to provision a list of games that it reads from `Games.txt`. Each input game binary has a corresponding output game binary.

The **packageSystem.py** script packages the system boot image, MES.BIN, specified in SystemImage.bif.

The **deploySystem.py** script does the following final provisioning steps:

- partitions and formats the SD (if needed)
- copies MES.BIN, the MITRE-provided BOOT.BIN, and the provisioned games onto an attached SD card.

3.4.2 provisionSystem.py (Python 3.x)

Syntax: "`python provisionSystem.py <userfile> <default games>`"

Description: This python3 script is used to provision an instance of your system with the list of user accounts in `Users.txt` and a list of preinstalled games for the `demo` account in `Default.txt`.

Arguments:

- `<userfile>` – Filename for ASCII text file that contains user information in a MITRE-defined Format. For information on length and allowable character sets refer to Username, PIN, and Game Names

```
# <user> <PIN>
User1 98765567
User2 87653998
```

- `<default games>` – Filename for ASCII text file that contains games which **must** be preinstalled for the demo account in a MITRE-defined format

```
# <gameName> <version>
DemoGame 1.0
2048 2.0
```

Outputs:

- `SystemImage.bif` – contains information for packaging your system in a MITRE-defined format. Note: the bitstream, `Arty_Z7_10_wrapper.bit`, must be the first image after the MITRE-specified bootloader line.

```
MITRE_Entertainment_System
{
        [bootloader] $ECTF_PETALINUX/tools/files/zynq_fsbl.elf
        // Participant's  Bitstream
        $ECTF_PETALINUX/Arty-Z7-10/images/linux/Arty_Z7_10_wrapper.bit
        // Paritcipant's Images
        $ECTF_PETALINUX/ Arty-Z7-10/images/linux/u-boot.elf
        $ECTF_PETALINUX/ Arty-Z7-10/images/linux/image.ub
}
```

- `FactorySecrets.txt` – contains information for provisionGames.py in a participant-defined format

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

Optional Outputs:

- This script may have additional outputs such as a U-Boot image, Kernel, and/or Filesystem. For example, in the reference design, provisonSystem.py creates u-boot.elf, image.ub and Arty_Z7_10_wrapper.bit and copies them to the paths specified in the outputted `SystemImage.bif`.

3.4.3 provisionGames.py (Python 3.x)

Syntax: "`python provisionGames.py <factorysecretsfile> <games>`"

Description: This python3 script is used to attach metadata and/or modify the games specified in `Games.txt` to be installable from the MITRE Entertainment System interface, using the install command, as described in 3.1.1.3. The purpose of the <game name> attribute is to allow multiple files with different file names to be considered as the same game (e.g., game-v1.00 and game-v2.00 can be stored in `/media/sd` simultaneously, but would be installed to the same game). The output of this script should be a file with metadata attached to it, stored in the directory "games" with the naming convention "<name>-v<version>", where "name" is the name argument and "version" is the version argument expressed as "major.minor", where "major" is the major version number and "minor" is the minor version number. Note that version 2.20 is a greater version than version 2.2. For reference, game names can be assumed to be a string of alpha-numeric characters (not containing spaces). `FactorySecrets.txt`, which is output by `provisionSystem.py`, can be used to communicate information from the system provisioning process to the game provisioning process.

Arguments:

- `<factorysecretsfile>` – Filename for ASCII text file that contains information from the system provisioning process
- `<games>` – Filename for ASCII text file that contains game information in a MITRE-defined Format

```
# <game path> <game name> <version> <user> […<user>]
path/to/game gameName 1.0 User1 User2
path/to/demo demoGame 1.0 Demo
```

Outputs:

- `<game name>-v<version>` – for each game it outputs a file that is named according to this format

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

3.4.4 packageSystem.py (Python 3.x)

Syntax: **"**`python packageSystem.py path/to/SystemImage.bif"`

Description: This python3 script is used to package MES.bin.

Arguments:

- `path/to/SystemImage.bif` – bif to use when generating MES.BIN

Outputs:

- `path/to/MES.BIN` – output path for packaged boot image

Warning: As described in section 2.4.2, only images specified in the *SystemImage.bif* will be encrypted and signed by MITRE when your system is provisioned in the attack phase.

3.4.5 deploySystem.py (Python 3.x)

Syntax: **"**`python deploySystem.py /dev/<sd-device> path/to/BOOT.BIN path/to/MES.BIN path/to/games [--noformat]"`

Description: This python3 script is used to partition and format the SD card, copy MES.BIN to the FAT partition, and copy the games to the ext4 partition. By default, this script will format the SD card; however, to save time when deploying, you can specify --noformat to skip that step. **It should not be necessary to edit this script unless you would like to load images not specified in SystemImage.bif onto the SD card.**

Arguments:

- `/dev/<sd-device>` – SD device to deploy onto (optional)
- `path/to/BOOT.BIN` – MITRE-provided boot image
- `path/to/MES.BIN` – gaming system boot image
- `path/to/games` – directory where provisioned games are

Note: deploySystem.py is a script for the initial deployment of your gaming system (a script run at the factory). Additional games, provisioned via the **provisionGames.py** script, may be distributed after the system has been deployed. These games, which will be manually copied to the games partition on the SD card, must be playable.

3.5 Development Environment Requirements

In addition to the functional requirements specified above, your submission must abide by the following rules.

When you submit your design, the vagrant repository you provide:

1.) Must only contain modifications to team/customizations.sh and config.rb
2.) Cannot change any of the configurations made by the VagrantFile.
3.) Clone the correct release of your petalinux repository to the provisioned VM at /home/vagrant/MES.
4.) Leave the default account username and password unchanged.
5.) Allow connections over SSH with password authentication for the user `vagrant`.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

When you submit your design, the petalinux repository you provide:

1.) Must abide by the following folder structure

```
/home/vagrant/MES
    |-- tools
        |-- provisionSystem.py
        |-- provisionGames.py
        |-- packageSystem.py
        |-- deploySystem.py
        |-- files/
            |-- generated/
                |-- SystemImage.bif
                |-- FactorySecrets.txt
                |-- MES.bin
                |-- games/
```

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

4 Handoff Phase

After the completion of the Secure Design Phase, each team must submit their design to MITRE. Submissions should include all source code, documentation, and supporting files necessary to build. All of this should be made available through the modification of the provided customizations.sh and config.rb, which will allow the administrators to easily recreate your team's development environment. Remember, you may ONLY modify the provided customizations.sh and config.rb files.

After receiving a team's design, MITRE will provision a system and validate that the system meets the functional requirements described in this document. Any design that will not build or does not meet these requirements will not be able to progress to the attack phase of the competition. MITRE will contact each team with handoff status within two (2) business days after a team's submission, whether it is accepted as functional or not. If a design is not accepted, teams will be allowed to address any problems identified and submit a revised version. Once a team's design has been accepted, they may not submit further designs (e.g. to patch security flaws that are identified after submission).

All source code and documentation, as well as the build environment/Vagrantfile, will be provided to other teams during the attack phase to discourage security-by-obscurity, as well as to accelerate attack development and encourage more sophisticated techniques for both sides. Exemplary documentation will be worth extra points at the discretion of the MITRE competition committee – see Documentation Points for details.

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

5 Attack Phase

During the Attack Phase, each design that has been validated during the Handoff Phase will be available for attack. For each design, the files listed below will be made available to all attacking teams. These files will be created by running the necessary build tools. The final gaming system boot image, MES.BIN, will be encrypted and signed by keys only accessible to the organizers. An attack phase BOOT.BIN will be distributed which is capable of booting these organizer-protected gaming system boot images.

- Source code for the entertainment system and provisioning tools
- Documentation
- Pre-built system files
    - BOOT.BIN
    - MES.BIN

Warning: As described in section 2.4.2, only images specified in the SystemImage.bif will be encrypted and signed by MITRE when your system is provisioned in the attack phase.

We strongly encourage responsible disclosure[2] if any vulnerabilities are discovered on open-source or commercial components used as part of the system. If desired, MITRE can help to coordinate the responsible disclosure of weaknesses to appropriate parties.

[2] https://en.wikipedia.org/wiki/Responsible_disclosure

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

5.1 Flag Descriptions

The format for any discovered flag will be of the form `ectf{<flag_name>_<16 hex characters>}`. For example, '`ectf{rollback_0123456789abcdef}`'. The specific flag names can be found below.

The following table lists the flags, as well as a description of each:

Flag Name	Capturing this flag proves that you can…	Requirement
Rollback	…install a "vulnerable" piece of software with a version older than the prepackaged one. To capture this flag, a current version of the software (preinstalled) must be executed prior to running the old version.	Versioning
Jailbreak	…use the jailbreak proof program or create a functional equivalent and run it to extract a flag from the PL Memory Region Read/Write Access)	Arbitrary code execution
PIN extraction	…determine the PIN of an account you don't have access to	Confidentiality
PIN bypass	…run a game from an account you don't have access to	Authentication
Intellectual Property	…read the value of the flag stored in plaintext in the binary	Confidentiality
Hacker mods	…modify an unwinnable game to gain victory	Integrity

NOTE: The jailbreak proof program will be provided when a provisioned system is requested in the attack phase.

6 Scoring

Points are scored in one of the six ways listed in the following sections.

6.1 Design Phase Milestone Flag Points

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. The milestone flags are:

Milestone	Proof	Deadline Date
Read the rules	If you read the rules, you'll know.	2/6/2019
UART MeSh	Provision and boot the reference design on the provided hardware.	2/13/2019
Boot the Demo Game	Provision, install, and play the demo game.	2/20/2019
Design Document	Submit a design document containing descriptions of how each command works on your system. Details to be released two (2) weeks after kick-off.	3/1/2019

If you focus on getting your development environment up and running, the milestone flags should be very little additional effort. Each Milestone flag has a deadline date at which point it can no longer be submitted for points. The scoreboard for submitting milestone flags will be active two (2) weeks after kick-off and link will be provided in Slack.

© 2019 The MITRE Corporation. All rights reserved
Approved for public release. Distribution unlimited 18-03747-4

6.1.1 Design Document Details [update v1.1]

The design document will serve as a high-level description of the security features of your gaming system. Specifically, the document must describe how your system and game provisioning processes along with your console command protocols protect each of the flags. You will not be held to the system description that you submit. If you choose to modify your design after submitting the document, that is OK.

We do not expect incredibly detailed design documents; however, spending time on this will benefit you in two ways: (1) the document will serve as a way to communicate your design to your advisor and team and (2) we will be able to ensure that your design does not break any competition rules. If we feel that your team made little effort to document your intended design we reserve the right to reject your submission.

You can submit this document up until the March 1st deadline; however, we recommended that you submit much earlier as it will encourage you to come to agreement on a design and allow extra time to integrate any feedback we have.

6.2 Offensive Flag Points

Each system is required to hold and protect "flags" that should only be revealed if the system is compromised. By submitting flags, a team is demonstrating that they have compromised the target system. A brief description is required for each attack that results in a flag submission.

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 difficulty of capture as more information becomes available. 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 had already captured. This is because the flag points that you are initially awarded need to be re-distributed as additional teams capture the same flag.

  

  Although counter-intuitive, it may be a good strategy to seek out and spend time attacking the most difficult targets. The most challenging flags will, in theory, be worth the most points in the end.

- 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 prevent teams from "holding" onto a flag without submitting it, the team that captures each flag first will get more points for that flag than teams that capture it later. This adjustment is due to the unequal distribution mentioned above.

6.3 Defensive Flag Points

Points will be awarded for every flag that has not been captured by other teams at a regular time interval (e.g., every day). As a result, more secure designs are likely to accrue more points than other designs. Additionally, only designs that have completed the Handoff phase are able to accrue Defensive points, so teams that take longer to submit a working design may score fewer points.

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

6.4 Performance Points

Fast system performance will be rewarded to encourage thoughtful choices on the tradeoffs between usability and security of your system. For each of the operations listed in section 3.3.4, we will reward points based on your system's performance in comparison to the maximum times allowed in section 3.3.4. Points will be awarded for **each** of the operations in the rows of Table 1 according to the following formula:

$$Points\ awarded = N * [(T - t) / T]$$

T: Max time (seconds) allowed for the operation

t: Your system's measured time for the operation

N: Max points awarded for any other single flag in the competition (TBD)

For example, if your system performs "First Boot" in 10 seconds instead of the maximum allowed (20 seconds) then you will be awarded 50% of the max points awarded for any other flag.

6.5 Documentation Points

Good documentation will be rewarded to discourage security-by-obscurity. "Good documentation" is meant to describe clear and well-commented code, useful descriptions of modules/functions/classes, 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 can be sufficient for Max points. Quality will be valued over quantity.*

The maximum number of points that can be scored for documentation is equal to the value of an Offensive 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 that is 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%** Very confusing or deceptive comments and documentation

Points for documentation will not be awarded until near the end of the attack phase to allow for proper analysis. Honest feedback on documentation from other teams will be solicited and will be factored into the final point determination.

6.6 Write-ups

There will be an opportunity for the top teams to provide write-ups for additional points. These teams will have an opportunity to submit a defense write-up as well as a single attack write-up. The Defensive write-up may discuss security measures that worked well, those that could have been improved upon, or any that were planned but could be developed in the time provided. The attack write-up is to award teams that develop interesting or novel attacks which do not directly capture an existing flag. Further details on the number of teams that may submit write-ups and the content/format of the write-ups will be provided during the attack phase.

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

7 Award Ceremony

All teams (students and faculty advisors) who submit a working design are invited to an award ceremony at MITRE on April 18th, 2019. This event will be collocated in Bedford, Massachusetts and McLean, Virginia - your team may attend at whichever location is more convenient. During the award ceremony, the top 5 teams will be invited to give a presentation of their work during the design and attack phases of the competition. At this time, the final scoring will be revealed, including points for the write-ups, and awards will be presented.

8 Important Dates

Kickoff --- January 16th, 2019

- Competition officially kicks off.

System Hand-off --- March 1st, 2019

- System design and implementation is due.
- After MITRE has verified a submitted design, the designing team will be given access to all other verified designs for attack.
- Scoreboard opens.

Scoreboard Closes --- April 12th, 2019

- Flag submission is closed.
- Teams will be contacted for write-ups, which will be due April 18th

Award Ceremony – April 18th, 2019

- All teams who submitted a working design will be invited to MITRE for an award ceremony, where MITRE will announce the results of write-up judging and present awards.

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

9 Rules

Most rules are described and explained throughout the challenge description in the earlier sections – please read this entire document! This section is intended as a concise summary of the most important rules.

(1) In addition to the rules provided by MITRE, participants should also adhere to all 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 admins.

(3) When submitting your secure design, all source code and documentation must be shared.
   (a) This is to discourage security-by-obscurity, as well as to accelerate attack development and encourage more sophisticated techniques for both sides.

(4) During the attack phase, you may only attack the student-designed systems explicitly designated as targets.
   (a) For example, the First Stage Bootloader (FSBL) provided by MITRE and security features provided by the Arty Z7 platform are off-limits and should not be attacked.

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

(6) No permanent lock-outs are allowed. See Section 3.3.4 for maximum timing delays allowed.

(7) Team sizes are unlimited. ectf{readtherules_4c2ebaf3c2b4d9a9}
   (a) Most teams will consist of members of varying degrees of experience and skill level. Our hope is that this creates an opportunity for mentoring, where the most knowledgeable team members will help teach and guide the other members of the team. Team advisors should help manage meeting times and organization of large teams.
   (b) We want to encourage as many students to participate as possible, even if they are not willing to commit a significant amount of time to the competition.

(8) Teams may consist of students at any level: undergraduate, graduate, PhD, or a mix.

(9) Teams are limited to two write-up submissions.

(10) Your system must work with the hardware that was provided. Switching to a different platform or modifying the physical hardware during the design phase is not allowed.

If you have any questions, ask!

   (a) Join our slack channel: ectfmitre2019.slack.com
   (b) Email: ectf@mitre.org

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

10 Frequently Asked Questions

10.1 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.

10.2 Can we add intentional delays during boot or installing/playing games to make it more difficult for an attacker to collect large numbers of observations?

There should not be any intentional delays that may be noticeable to the user because a slow boot or update time will negatively impact the user experience and hurt sales for your product. For our purposes, we'll consider any delays more than 100 milliseconds to be noticeable to the user. Also, see section 6.4 regarding bonus points that you may be missing out on!

If your system detects that it is under attack, additional delays are OK, but must be limited to no more than 5 seconds per boot or update. Permanent lock-out or self-destruction is not allowed (see next question).

10.3 Is it OK to brick the board when an attack is detected?

No! This is a game system! We can't have it be so easy for an attacker to disable the entire system! Can you imagine the cost of the recalls?!

10.4 Can we physically modify the board with countermeasures during the design phase?

No. The boards are provided to teams at the beginning of the competition, so you won't have an opportunity to modify the hardware during provisioning. Everything that needs to be done to the chip for provisioning needs to be done in an automated fashion by your Build and Configure tools. You may, however, modify the boards for attacks.

10.5 How many boards can we get during the attack phase? (e.g., if we keep bricking them, can we keep getting new ones?)

Each team will be given two boards at beginning of the competition. Teams are welcome to purchase additional hardware for parallelizing efforts; however, only the provided boards will run systems provisioned in the attack phase.

10.6 Can we attack the other teams' development environment?

No! Everything other than the provisioned chips, the host tools, and the firmware images are considered out-of-bounds. In other words, there is **nothing** that you are allowed to attack until we get to the attack phase.

10.7 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 university, state, and federal regulations.

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4

11 Change Log

2019.01.16 -- 1.0 -- Initial Release

2019.02.17 -- 1.1 -- Update: Command IO formatting, Rules and Requirements Clarification

- 3.1.1.1 Shell Format – specified shell prompt format to ensure consistency when testing
- 3.1.1.2 Login Format – specified login flow to ensure consistency when testing
- 3.1.1.3 Commands – specified that install/uninstall is user specific; remove requirement for play to restart the board
- 3.1.1.4 Command Format – specified input and output format for each command
- 3.1.1.7 Flash – provisioned games do not need read/write access to the reserved flash region
- 3.3.2 Game Versioning – specified that versioning is user specific
- 3.3.3 PL Memory Region Read/Write Access – specified device driver /dev/mesh_drm for PL access
- 3.3.4 Timing Requirements – Add additional timing requirements and loosened some times
- 3.5 Development Environment Requirements – specified folder structure and vagrant up behavior
- 6.1.1 Design Phase Milestone Flag Points – added details about design document milestone

© 2019 The MITRE Corporation. All rights reserved

Approved for public release. Distribution unlimited 18-03747-4