


# INTRO to ACTP

 5-Day course

## COURSE DESCRIPTION

The **Introduction to ACTP** course is an introductory class establishing the base for both 10-week CNO courses (Windows & Linux). This class will familiarize students with some of the computing concepts to help build a clear picture of how to interact with the machine.

This course was specially crafted by ACTP instructors to build and strengthen the core skills that will be the foundation of our other ACTP offerings. The concentrated material will provide essential understanding that will be helpful for any student looking to start with systems engineering and reverse engineering.

*Prerequisites: Students should have a solid understanding of programming in a systems programming language.*



## Relevant Today

Develops core skills that are widely applicable to modern systems and forms a firm foundation for CNO development



## Inclusive Approach

Builds a cohesive base for better understanding for both the Windows and Linux 10-week CNO programmer courses



## Topics Include:

Introduces C programming, x86 assembly, debugging techniques and reverse engineering skills

## AGENDA COURSE CONTENT

### DAY 1

- **Computer Architecture:** A look at how computing takes place from an architectural level. This portion gives students a better understanding of the context around their code and helps them to root-cause bugs and mistakes.
- **Efficient Coding Environments:** This section introduces students to compilers, linkers, and other pieces of their build toolchain. Students will better understand the tools they will work with, and be better prepared to handle warnings, errors, and problems in the build process.

### DAY 2

- **Using Documentation:** Students will be introduced to system documentation and will feel comfortable navigating through unfamiliar documentation. They will be better prepared to handle unfamiliar APIs and tools while becoming more self-reliant in their development.
- **C Programming:** Will cover common pitfalls and frustrations with the C programming language. Students will also be introduced to common design patterns used in C programming.

### DAY 3

- **Introduction to x86 Assembly:** A basic introduction to assembly will be given, allowing students to grow comfortable reading and writing assembly, as well as understanding the connection between C code patterns and their associated assembly.

### DAY 4

- **Debugging:** Students will gain experience using a debugger, finding bugs, and eliminating those bugs. They will also use a variety of debugging tools that provide specialized views into a process.

### DAY 5

- **Reverse Engineering:** Combined with their debugging exposure, students will be introduced to modern reverse engineering tools and will practice their skills against increasingly difficult targets.

**Exam and certification/completion**

*\*The full course syllabus is available upon request*

# LINUX CNO Programmer

 10-Week course [45 days]

## COURSE DESCRIPTION

The **Linux CNO Programmer Course** is an intensive, hands-on course focusing on providing students with the skills and knowledge needed to become an advanced CNO programmer, with emphasis on the Linux environment. A CNO programmer develops technologies to defend, attack and exploit computer networks. This requires a deep understanding of operating systems and software internals, combined with advanced skills in C, assembly, networking, and reverse engineering. It also requires specialized knowledge and experience that cannot be gained through conventional education or programming work. Class format combines lecture and demonstrations with practical lab assignments.

Workstations and all required technical documentation will be provided.

*Prerequisite: Bachelor's degree in Computer Science or Computer Engineering, or equivalent experience; Previous programming experience in C; Experience in Linux Programming and x86\_64 assembly.*

*Success requires an intense desire and capacity to learn; as the coursework becomes progressively more difficult, so personal motivation is critical.*





## Certified Training Program

Following completion of the three modules, students will be capable of assisting in the CNO tool development life cycle.



## Intensive, Hands-On Training

Emphasizes lab work over a lecture format, this course combines demonstrations with practical lab assignments, including two labs that function as culminating exercises.



## Qualified Assessments

Successfully completing the full course with an 80% average or better, students receive certification and are recognized as ManTech Certified Advanced Cyber Programmers (CACAP).

## 1 CNO CORE MODULE

### **PYTHON** 3 DAYS

Introduces the Python programming language emphasizing tools and techniques that are useful for CNO tasks such as test development and vulnerability research. Topics include the Python interpreter, basic types and operators, statements, functions, modules, classes, exceptions, and more.

### **NETWORKS** 5 DAYS

Explores IPv4 and IPv6 networks and sockets programming. Use Wireshark to inspect and analyze network traffic; utilize Python to write client/server applications, and to develop tools for creating and modifying packets at the Ethernet and IP layers. Concepts studied include routing, network address translation, proxies, and packet filters. Protocols include Ethernet, IP, UDP, TCP, and HTTP.

### **ASSEMBLY** 3 DAYS

Covers the x86 (IA-32) and x86-64 (AMD64) assembly languages. Learn to read, write, and debug assembly code with topics including registers, flags, types, operators, memory addressing, the stack, Linux calling conventions, and string processing instructions. Introduces GDB and Make to develop and debug labs.

### **SOFTWARE REVERSE ENGINEERING** 5 DAYS

Introduces tools and techniques for analysis and exploitation of real-world vulnerabilities. Specifically analyzing x86 and x86-64 executable files. Utilize Ghidra, GDB, and other tools to perform both static and dynamic reverse engineering. Learn how to: identify data types, structures, function prototypes, imports, exports, and other constructs and document findings; Analyze disassembled functions and manually produce equivalent C code; use debugger to analyze running programs, using techniques such as break on access, conditional breakpoints, and tracing.

### **CNO CORE CRUCIBLE** 1 DAY

Applies earlier learning concepts and teaming to analyze and exploit a botnet to observe network traffic, reverse engineer protocols, and develop tools for communicating with botnet nodes. Successful communication with the botnet yields additional CNO challenges to complete and score points in a Capture-The-Flag (CTF) style event.

## 2 USER MODE DEVELOPMENT MODULE

### **LINUX SYSTEMS PROGRAMMING** 4 DAYS

Introduces the Unix programming environment that are guided and aided by POSIX APIs and Glibc extensions to create a variety of system tools. Emphasis on using development and debugging tools such as manpages, source headers, GNU Make, GDB, valgrind, Ghidra, and objdump. Comfortably develop Linux system tools entirely from the terminal, without need for external references.

### **LINUX INTERNALS** 4 DAYS

Explores the internals of the user space portion of the Linux environment. Learn the intricacies of the Linux process model, develop their own shell, interact with device files, and learn advanced filesystem concepts. Deep dive into the ELF format; create tools to view/ manipulate such files, how the system performs dynamic loading/linking. Topics include system calls, interaction of standard libraries with the kernel, extraction of system information using the proc file system, Linux security paradigm and access control methods.

### **LINUX CNO USER MODE DEVELOPMENT** 5 DAYS

Provides instruction on fundamental techniques and best practices for CNO tool development. Lab assignments focus on code injection, hooking, and hardening. Create tools to alter program execution, access sensitive memory, create new threads running custom payloads in existing programs, and more. Make programs that break out of sandboxes, avoid detection by PSPs, and are difficult to reverse engineer by standard tools.

### **VULNERABILITY RESEARCH & EXPLOITATION** 5 DAYS

Applies industry standard tools to discover hidden vulnerabilities; analyze; and exploit these vulnerabilities in several types of software. Use reverse engineering and advanced debugging techniques learned previously to analyze exceptional conditions to determine if and how the target may be exploited. Numerous vulnerability types are treated and explored, including use after free, buffer overflows, type confusion, heap corruption, race conditions, uninitialized data, and more. Topics also include developing fuzzers, exploiting targets in the presence of stack canaries, crafting custom payloads, heap buffer overflows, NX, ASLR, and other protections.

### **LINUX USER MODE CRUCIBLE** 2 DAYS

Facilitates collaborative teams to analyze and exploit multiple bots over the network. Use earlier coursework to reverse engineer a variety of targets, develop exploits, and CNO tooling to turn uncovered vulnerabilities into reliable capabilities.

## 3 KERNEL MODE DEVELOPMENT MODULE

### **LINUX KERNEL INTERNALS** 5 DAYS

Introduces the Linux kernel architecture and fundamentals of driver development by configuring, compiling, debugging, and installing a modern kernel. Examines the details of kernel components such as the Memory Manager, I/O Manager, Scheduler, and Object Manager. An emphasis given to kernel functionality and data structures frequently exploited by CNO tools. During lab assignments, learn to create loadable kernel modules and modify the kernel directly to interact with major subsystems and gain familiarity with the inner workings. Topics include analyzing crash dumps, writing a simple driver, remote kernel debugging, IO processing, function hooking, and synchronization.

### **LINUX CNO KERNEL MODE DEVELOPMENT** 3 DAYS

Examines how to write CNO drivers for the Linux kernel. Internal workings of subsystems are unveiled, highlighting APIs and code useful for CNO development. Create code to perform keylogging, access, and modification of network traffic, hijacking of interrupts, access, and modification of process memory, and more. Topics include reverse engineering and exploiting a vulnerable driver, logging keystrokes, utilizing kernel callback routines, creation of covert channels, kernel object manipulation, and injecting code into user processes.

\*The full course syllabus is available upon request

ACTP-CS-LINCNO-2023-0221

# WINDOWS CNO Programmer

 10-Week course [45 days]

## COURSE DESCRIPTION

The **Windows CNO Programmer Course** is an intensive, hands-on course focusing on providing students with the skills and knowledge needed to become an advanced CNO programmer, with emphasis on the Windows platform. A CNO programmer develops technologies to defend, attack and exploit computer networks. This requires a deep understanding of operating systems and software internals, combined with advanced skills in C, assembly, networking, and reverse engineering. It also requires specialized knowledge and experience that cannot be gained through conventional education or programming work. Class format combines lecture and demonstrations with practical lab assignments.

Workstations and all required technical documentation will be provided.

*Prerequisite: Bachelor's degree in Computer Science or Computer Engineering, or equivalent experience; Previous programming experience in C; Experience in Windows Programming and x86 assembly.*

*Success requires an intense desire and capacity to learn; as the coursework becomes progressively more difficult, so personal motivation is critical.*



```
elif_operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
elif_operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add back the deselected mirror modifier object
mirror_ob.select= 1
modifier_ob.select=1
bpy.context.scene.objects.active = modifier_ob
print("Selected" + str(modifier_ob)) # modifier ob is the active ob
#mirror_ob.select = 0
#bpy.context.scene.objects.active = mirror_ob
#bpy.data.objects[mirror_ob.name].select = 1
```



## Certified Training Program

Following completion of the three modules, students will be capable of assisting in the CNO tool development life cycle.



## Intensive, Hands-On Training

Emphasizes lab work over a lecture format, this course combines demonstrations with practical lab assignments, including two labs that function as culminating exercises.



## Qualified Assessments

Successfully completing the full course with an 80% average or better, students receive certification and are recognized as ManTech Certified Advanced Cyber Programmers (CACAP).

## 1 CNO CORE MODULE

### **PYTHON** 3 DAYS

Introduces the Python programming language emphasizing tools and techniques that are useful for CNO tasks such as test development and vulnerability research. Topics include the Python interpreter, basic types and operators, statements, functions, modules, classes, exceptions, and more.

### **NETWORKS** 5 DAYS

Explores IPv4 and IPv6 networks and sockets programming. Use Wireshark to inspect and analyze network traffic; utilize Python to write client/server applications, and to develop tools for creating and modifying packets at the Ethernet and IP layers. Concepts studied include routing, network address translation, proxies, and packet filters. Protocols include Ethernet, IP, UDP, TCP, and HTTP.

### **ASSEMBLY** 3 DAYS

Covers the x86 (IA-32) and x86-64 (AMD64) assembly languages. Learn to read, write, and debug assembly code with topics including registers, flags, types, operators, memory addressing, the stack, Windows calling conventions, string instructions, and the WinDbg debugger.

### **SOFTWARE REVERSE ENGINEERING** 5 DAYS

Introduces tools and techniques for analysis and exploitation of real-world vulnerabilities. Specifically analyzing x86 and x86-64 executable files. Utilize Ghidra, WinDbg, and other tools to perform both static and dynamic reverse engineering. Learn how to: identify data types, structures, function prototypes, imports, exports, and other constructs and document findings; Analyze disassembled functions and manually produce equivalent C code; use debugger to analyze running programs, using techniques such as break on access, conditional breakpoints, and tracing.

### **CNO CORE CRUCIBLE** 1 DAY

Applies earlier learning concepts and teaming to analyze and exploit a botnet to observe network traffic, reverse engineer protocols, and develop tools for communicating with botnet nodes. Successful communication with the botnet yields additional CNO challenges to complete and score points in a Capture-The-Flag (CTF) style event.

## 2 USER MODE DEVELOPMENT MODULE

### **WINDOWS SYSTEMS PROGRAMMING** 4 DAYS

Introduces Windows development tools and the Win32 API. Create 64-bit C programs and dynamic libraries that use APIs for file I/O, registry access, memory management, sockets, thread operations, asynchronous I/O, IPC mechanisms, and more.

### **WINDOWS INTERNALS** 4 DAYS

Explores the internals of the user space portion of the Windows platform. Broadens Win32 API skills introduced in previously to describe the advanced Windows operating system concepts used in implementation. Introduces the underlying executive system services and describes, in-detail, other major elements of Windows. Topics include handles, processes and threads, security constructs, and memory manipulation.

### **WINDOWS CNO USER MODE DEVELOPMENT** 5 DAYS

Provides instruction on fundamental techniques and best practices for CNO tool development. Lab assignments focus on writing position-independent code packages that can import symbols from system DLLs, injecting payload code into another process, safely hooking functions, and hiding files from the user. Additional topics discussed include CNO network communications, stealth techniques, common hardening techniques, and Windows security features.

### **VULNERABILITY RESEARCH & EXPLOITATION** 5 DAYS

Applies industry standard tools to discover hidden vulnerabilities; analyze; and exploit these vulnerabilities in several types of software. Use reverse engineering and advanced debugging techniques learned previously to analyze exceptional conditions to determine if and how the target may be exploited. Numerous vulnerability types are treated and explored, including use after free, buffer overflows, type confusion, heap corruption, race conditions, uninitialized data, and more. Topics also include developing fuzzers, exploiting targets in the presence of stack canaries, crafting custom payloads, heap buffer overflows, NX, ASLR, and other protections.

### **WINDOWS USER MODE CRUCIBLE** 2 DAYS

Facilitates collaborative teams to analyze and exploit multiple bots over the network. Use earlier coursework to reverse engineer a variety of targets, develop exploits, and CNO tooling to turn uncovered vulnerabilities into reliable capabilities.

## 3 KERNEL MODE DEVELOPMENT MODULE

### **WINDOWS KERNEL INTERNALS** 5 DAYS

Introduces the Windows kernel architecture and fundamentals of driver development by configuring, compiling, debugging, and installing a modern kernel. Examines the details of kernel components such as the Memory Manager, I/O Manager, Scheduler, and Object Manager. An emphasis given to kernel functionality and data structures frequently exploited by CNO tools. During lab assignments, learn to create loadable kernel modules and modify the kernel directly to interact with major subsystems and gain familiarity with the inner workings. Topics include analyzing crash dumps, writing a simple driver, remote kernel debugging with WinDbg, IO processing, function hooking, and synchronization.


### **WINDOWS CNO KERNEL MODE DEVELOPMENT** 3 DAYS

Examines how to write CNO drivers for the Windows kernel. Internal workings of subsystems are unveiled, highlighting APIs and code useful for CNO development. Create code to perform keylogging, access, and modification of network traffic, hijacking of interrupts, access, and modification of process memory, and more. Topics include reverse engineering and exploiting a vulnerable driver, logging keystrokes, utilizing kernel callback routines, creation of covert channels, kernel object manipulation, and injecting code into user processes.

\*The full course syllabus is available upon request

ACTP-CS-WINCNO-2023-0221

# ADVANCED Vulnerability Research

 5-Day course

## COURSE DESCRIPTION

The **Advanced Vulnerability Research** course is ACTP's most technical and challenging offering. Designed for those who are looking to take their VR skills to the next level, this five-day course will teach students how to find vulnerabilities using techniques such as fuzzing, symbolic execution, and code property graphs.

Written by a team of vulnerability research experts, this course will teach students how to find bugs such as use-after-free, type confusion, memory corruption, and logic bugs in software applications like web browsers and operating system kernels.

*Prerequisites: Students are expected to have a firm grasp of x86-64 assembly, strong programming skills in C and C++, be comfortable with reverse engineering software using IDA and Ghidra, and have completed the Vulnerability Research and Exploitation module within either the Windows or Linux CNO Programmer course.*



## Relevant Today

ManTech's expert ACTP instructors teach you the skills required to find unique vulnerabilities in modern software applications.



## Inclusive Approach

Builds off foundational knowledge gained from the CNO programmer course to take students to the next level.



## Topics Include:

Advanced bug class discovery via fuzzing, symbolic execution, and code property graphs. Also, advanced reverse engineering, deep dives and root cause analysis of modern real world vulnerabilities.

## AGENDA COURSE CONTENT

### DAY 1

- **Code Auditing:** A thorough review of auditing techniques across source code and binaries. This section gives students the deep technical insights to approach looking for vulnerabilities in open source and closed source applications using advanced code auditing techniques and writing custom analysis scripts for Ghidra.
- **Integer Errors and Race Conditions:** Students will become familiar with identifying undefined integer operations and dangerous synchronization patterns that lead to exploitable bugs.

### DAY 2

- **Fuzzing:** Students will learn how to use modern fuzzing tools such as AFL++ to find vulnerabilities in target programs. This includes hands on labs to attack network-based applications.
- **Type Confusion:** A deep dive into understanding and finding type confusion bugs.

### DAY 3

- **Advanced Fuzzing:** Building on the previous fuzzing section, students will write custom fuzzing harnesses for hard targets and learn about fuzzing exotic interfaces such as RPC and operating system calls. This section also covers emulation and instrumentation engines such as Unicorn Engine and Frida.
- **Memory Corruption:** This section gives students a deep understanding of the memory corruption bug class and how to detect these flaws during fuzzing.

### DAY 4

- **Symbolic Execution:** A deep dive into the cutting-edge VR concepts of symbolic execution from SAT and SMT solvers, intermediate representations and languages, and execution engines. Students will write their own x86-64 symbolic execution engine and learn how to use angr.


### DAY 5

- **Code Property Graphs:** An introduction to graph theory will lead students into understanding how to combine abstract syntax trees, control flow graphs, and data flow graphs into code property graphs. From there, modern tools such as Joern will show how these techniques can find subtle vulnerabilities.
- **Use-After-Free and Information Disclosures:** This section details the intricate ways that information disclosures and use-after-free vulnerabilities manifest.

**Exam and certification/completion**



# INTRO to **ARM** Assembly

 3-Day course

## COURSE DESCRIPTION

The **Introduction to ARM Assembly** course, students will learn to read, write, and debug assembly code for the ARM EABI used by smart phones and mobile devices. Topics include registers, the ARM and Thumb instruction sets and their encodings, literal pools, the stack, the ARM calling conventions, cross compilation, remote debugging with GDB, GNU inline assembly, reverse engineering pitfalls, and more. Students should have C programming experience. Prior experience with another assembly language is helpful, but not required.

Workstations, Android devices, and all required technical documentation will be provided.

*Prerequisites: Students should have C programming experience. Prior experience with another assembly language is helpful, but not required.*



## Relevant Today

60% of the world's population uses an ARM device daily. ARM processors are found in 99% of the world's smartphones and tablets today.



## Inclusive Approach

Exposure to all major platforms components, including major revisions, the Application Binary Interface (ABI), and ARM-specific hardware modules.



## Topics Include:

Registers, ARM and Thumb instruction sets and their encodings, literal pools, the stack, the ARM calling conventions, cross compilation, remote debugging with GDB, GNU inline assembly, reverse engineering pitfalls, and more.

## AGENDA COURSE CONTENT

### DAY 1

- Setup a toolchain for building ARM binaries
- Learn basic adb commands
- Setup gdb to be able to debug applications on an Android device
- Learn about basic instructions such as mov, add, multiply, divide, and shift
- Take first quiz

### DAY 2

- Learn about memory addressing and layout
- Learn about the push and pop instructions
- Learn about the stack and it's conventions
- Learn about calling conventions on ARM
- Take second quiz

### DAY 3


- Learn about syscalls
- Learn about changing modes (ARM vs Thumb)
- Learn about inline assembly
- Learn about the Neon coprocessor

#### **Exam and certification/completion**

*\*The full course syllabus is available upon request*

# ANDROID

## Programming

 5-Day course

### COURSE DESCRIPTION

The **Android Programming** course is a unique course offering students the ability to explore the Android operating system. This lab-driven class exposes students to the entire OS API, covering everything from development of Android applications using the SDK and Android Studio to how these APIs map to native libraries, the Linux kernel, and Android-specific kernel components. There is also a heavy emphasis on security at each level covered, including features and developments from the latest releases of Android.

Workstations, Android devices, and all required technical documentation will be provided.

*Prerequisites: Students should have a bachelor's degree in Computer Science, Computer Engineering, or equivalent experience. Some C and Java programming experience is preferred*

GSA



### Security

Become certified in Android programming as well as security topics within the Android OS.



### Application Development

Provides hands-on experience with both application and native app development.



### Cyber Security Research

A premier offering of ACTP aimed at cyber security researchers and programmers.

## AGENDA COURSE CONTENT

### DAY 1

- Android architecture
- AOSP
- Android Debugging Bridge
- Android device modes
- Android Studio
- Android applications
- Rooting and flashing a phone

### DAY 2

- Android Broadcast Receivers
- Android Permissions
- Android Activities
- Android Services: development, types, AIDL interface language
- Intents and Messages: types, security

### DAY 3

- Android Content Providers
- Binder driver – functionality, implementation
- Android sensors – use, types
- Android Frameworks
- Android JNI
- Native Code and Native Development Kit (NDK) introduction

### DAY 4

- Android Native Sensors
- Cross-compilation process
- Android Native Service development
- Android Native to Java Communication Models
- Android HAL

### DAY 5


- In-depth Practical

**Exam and certification/completion**

\*The full course syllabus is available upon request

ACTP-CS-ANDPROG-20191107

# ANDROID INTERNALS

 5-Day course

## COURSE DESCRIPTION

The **Android Internals** course builds on the foundation established in the Intro to Android Programming course. This course dives deeper into the Android operating system to explore concepts such as the internals of APKs, the Package Manager, the Activity Manager, Zygote, Android Services, RILD, HAL, and WiFi. Students who take this course will learn advanced architectural concepts of the Android operating system as well as develop native applications that interface with many different components of an Android device. Students will also learn how to analyze and reverse engineer APKs.

Workstations, Android devices, and all required technical documentation will be provided.

*Prerequisites: Students should have a bachelor's degree in Computer Science, Computer Engineering, or equivalent experience. Students should also have taken our Intro to Android Programming course or have sufficient experience in Android development*



## OS Internals and Security

Become certified in Android Internals as well as security topics within the Android OS.



## Application Development

Provides hands-on experience with both application and native app development.



## Cyber Security Research

A premier offering of ACTP aimed at cyber security researchers and programmers.

## AGENDA COURSE CONTENT

### DAY 1

- Decompiling and reverse engineering APKs
- APKs: structure, format, build process, manifest files, .dex and .odex files, signing process
- Package Manager internals
- Activity Manager internals
- Cross-app permission use

### DAY 2

- Zygote internals
- Android Service architecture
- Manually spawning APKs
- Tracing system server requests
- Bypassing permission checks

### DAY 3

- Boot internals
- Booting unpacking/repacking
- Init internals
- RILD internals
- RILD filtering

### DAY 4

- HAL internals
- WiFi internals
- Custom HAL module
- Direct WiFi access

### DAY 5


- Android security model in depth
  - Linux permissions
  - Linux capabilities
  - Linux secbpf
  - SELinux (SEAndroid)

**Exam and certification/completion**

\*The full course syllabus is available upon request

ACTP-CS-ANDINT-20191107

# EMBEDDED TESTING

 5-Day course

## COURSE DESCRIPTION

The **Embedded Testing** course teaches the fundamental concepts, tools and techniques required to perform proper testing of CNO tools for verification and validation according to defined project requirements. One aspect of this course is the creation of testing environments. A broad and complex topic in and of itself, this course streamlines the testing pipeline using multiple tools.

To facilitate a smooth experience, the course provides template code and scripts which assist students in performing some generalized tasks for testing. Students will build off this code to give their testing environments a solid structure and allow for automation. Alternative methods, plugins, and tools will be discussed, so that processes may be replicated after completion of this course.

*Prerequisites: Previous hardware testing experience is not required. Students will be tasked with intermediate programming objectives. Python scripting and C coding knowledge is required along with experience using Linux Operating Systems and basic knowledge of networking.*



## Relevant Today

Hands-on experience writing tests and testing virtual hardware devices from the theoretical beginning of a project to the end.



## Inclusive Approach

Develop skills using commonly available test tools with hardware components in the loop with an emphasis on automated testing.



## Topics Include:

A practice-based approach to testing scenarios requiring setup, configuration, and execution of multiple levels of tests including emulation mimicking target deployment environments.

## AGENDA COURSE CONTENT

### DAY 1

#### TESTING OVERVIEW

- Testing Overview
- Unit Testing
- Requirements and Planning
- Mocking

### DAY 2

#### TESTING METRICS AND VISUALIZATION

- Component and Integration Testing
- Testing Infrastructure
- Test Results, Artifacts and Analysis
- Code Coverage Tools

### DAY 3

#### EMULATION

- QEMU
- QEMU Internals
- Systems Testing

### DAY 4

#### TESTING ENVIRONMENTS AND HARDWARE-IN-THE-LOOP

- Testing Environments and Management

### DAY 5

Exam and certification/completion

*\*The full course syllabus is available upon request*