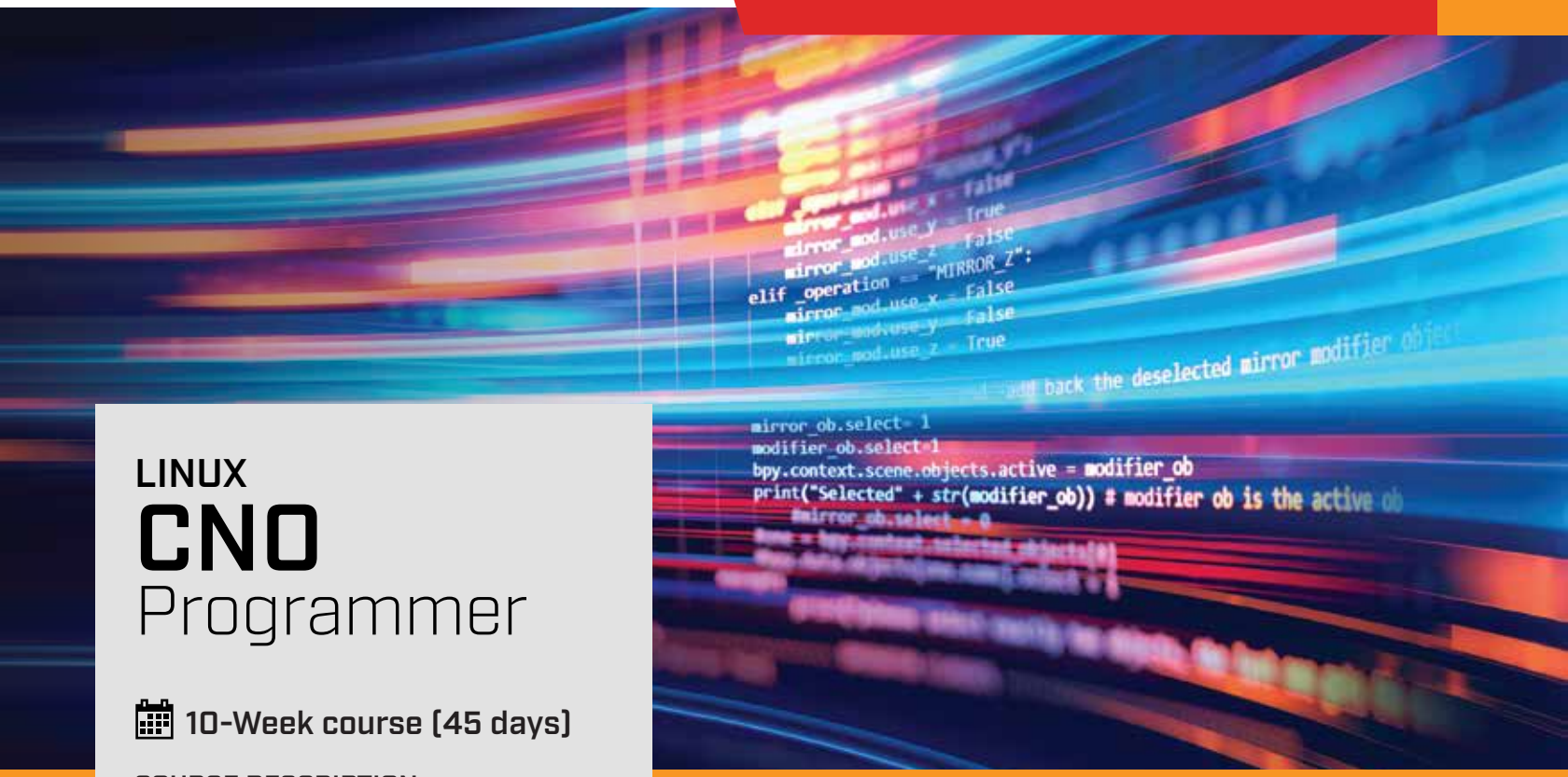
ACTP
ADVANCED CYBER TRAINING PROGRAM



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.

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.

GSA

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 (CACP).

MANTECH
ALWAYS ADVANCING

actp.mantech.com actp@mantech.com (443) 820-2195

ADVANCED CYBER TRAINING PROGRAM

LINUX CNO PROGRAMMING

10-Week course (45 days) GSA

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 MODE — 8 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. 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. 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, synchronization, 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–20250912