NCO 2026 Prelims
Detailed Syllabus and
Tools Guidelines

This syllabus for the National Cybersecurity Olympiad (NCO) Prelims 2026 serves the following purposes:

- It specifies a set of required prerequisite knowledge.
- It serves as a set of guidelines to determine the suitability of a challenge for the NCO Prelims 2026. Challenges that contestants may encounter during the NCO Prelims 2026 will be within the scope of the topics and concepts stated in this syllabus.
- Schools should be able to prepare their students for the NCO Prelims 2026 by following the content of the syllabus.

In line with the traditional categories encountered in cybersecurity competitions, such as Capture The Flag (CTF) competitions, we classify a selection of topics into five main categories:

- Binary Exploitation (Pwn)
- Cryptography
- Forensics
- Web Exploitation
- Reverse Engineering

The set of topics presented aims to be as comprehensive as possible but may not be exhaustive.

Furthermore, details regarding tools and software permitted for use in NCO 2026 Prelims will be outlined, as well as a set of recommended tools that Contestants may utilise.

NCO 2026 Prelims is an **open book, no internet** contest, so Contestants are expected to prepare materials in advance in accordance with the topics described in this syllabus.

Detailed Syllabus

The following sections are fine-grained descriptions of the topics which may be covered in each of the previously outlined categories. Topics which are not explicitly mentioned but are prerequisites to stated topics are by default within scope, whereas topics which may be extensions of stated topics may not be within scope.

Binary Exploitation / Pwn

Contestants should know and understand how Linux binaries operate and their common vulnerabilities. They are expected to be familiar with common stack and heap primitives, as well as how to exploit them. Students are to know how to spot vulnerabilities in binaries and convert given primitives to arbitrary code execution.

1. Fundamentals

- GDB Debugging
- x86 Assembly and Shellcoding
- ROP Gadgets (including special gadgets like *ret2csu*)
- Stack Canary
- GOT (Global Offset Table), no/partial/full RELRO
- PIE (Position Independent Executable)
- ASLR (Address Space Layout Randomisation)
- Special-purpose Registers on *x86_64* and *x86*
- Calling Conventions on *x86_64* and *x86*
- *seccomp*

2. Basic Exploitation

- Integer Overflow
- Type Confusion
- Stack-based Buffer Overflow
- ROP Chains
- One Gadget
- Ret2Win
- Ret2Libc
- Format String Bugs and Specifiers
- Syscall tables for *x86_64* and *x86*, including SROP (*sigreturn oriented programming*)
- *ret2dlresolve*

3. GNU C Library (glibc)

- Arbitrary Write to Shell
    - *__free_hook* & *__malloc_hook*
    - *exit_funcs*
    - *setcontext32*
    - FSOP (including House of Apple and its variants)
    - Libc GOT Overwrite
- File stream attacks
    - Arbitrary read via *_IO_2_1_stdout_*
    - Arbitrary write via *_IO_2_1_stdin_*

4. Dynamic Allocator Misuse (glibc's *ptmalloc*)

- 2.22 - 2.25, Pre-*tcache*
- 2.26 - 2.31, Pre-pointer mangling
- 2.32 - 2.41, Pre-*tcache* rework
- 2.42, Post-*tcache* rework
- Heap-Based Techniques
    - Heap Overflow
    - Double free and UAF (Use-After-Free)
    - Heap Leak via *tcache*/*fastbin*
    - Libc Leak via *unsorted bin*
    - *tcache* Poisoning and *fastbin* Attacks
    - *largebin* Attack
    - Heap Grooming
    - Unsafe Unlink
    - Pre-discovered Houses

Cryptography

Contestants should be familiar with common cryptosystems, as well as known attacks and algorithms associated with them, including cases where the same attacks are applied to other systems. Students are to acquaint themselves with post-quantum cryptography as well.

1. Fundamentals

- Data Representation
- Python
- SageMath

2. Number Theory

- Modular Arithmetic
- Greatest Common Divisor, Lowest Common Multiple
- Bezout's Identity
- Fermat's Little Theorem
- Euler Totient Function
- Chinese Remainder Theorem
- Quadratic Residues
- Legendre Symbol
- Tonneli Shanks
- Miller-Rabin Primality Test
    - Carmichael Numbers

3. Asymmetric Cryptography

- RSA Encryption, Decryption
- RSA Digital Signatures
- RSA Attacks
    - Small e Big n
    - Same n Different e
    - Hastad's Broadcast Attack
    - Franklin-Reiter
    - Coppersmith's Attack and Small Roots
    - Coppersmith's Short Pad Attack
    - RSA LSB Oracle
    - RSA Padding Oracle
    - Bleichenbacher's Attack
    - Approximate GCD
    - Factorisation Techniques
        - Fermat Factorisation
        - Lenstra Elliptic-Curve Factorization (ECM)
        - Pollard's p-1 algorithm

        - William's p+1 algorithm
- Discrete Logarithm Problem (Algorithms)
    - Baby Step Giant Step
    - Pohlig Hellman
    - Pollard's Rho
    - Index Calculus
- Diffie-Hellman Key Exchange
    - Man In The Middle
- Elliptic Curve Cryptography
    - Definition
    - Group Operation - Point Addition and Scalar Multiplication
    - Elliptic Curve Discrete Logarithm Problem (ECDLP)
    - Elliptic Curve Digital Signature Algorithm (ECDSA)
    - ECDSA Nonce Reuse
    - ECDSA Biased Nonce
    - Quadratic Twist Attack
    - Invalid Curve Point Attack
    - Smart's Attack
    - Frey-Ruck Attack
    - MOV Attack
- Other Cryptosystems
    - Rabin
    - Paillier
    - El Gamal

4. Symmetric Cryptography

- Bitwise Operations (Xor, And, Or, Not, etc.)
- Bit Shifting
- AES Block-Based Ciphers
    - AES ECB, CBC, CTR, GCM Block Operations
    - CBC Malleability
    - CTR Malleability
    - GCM Forbidden Attack
    - AES Internals
    - Padding Oracle Attacks
    - Linear/Affine SubBox
    - Linear Analysis
    - Differential Analysis
- General Stream Ciphers
    - RC4, ChaCha20
    - Keystream Reuse Attacks
- Pseudorandomness
    - Linear Feedback Shift Registers (LFSR)
    - Linear Congruential Generators (LCG)

        - Truncated LCGs
    - Python Random
        - Mersenne Twister
- Other Cryptosystems
    - Feistel-based Ciphers
    - Blowfish

5. Post-Quantum Cryptography

- Lattice Cryptography
    - Merkle-Hellman Knapsack Cryptosystem
    - Learning with Errors (LWE)
        - Ring-LWE
        - Module-LWE
        - Error Related Attacks (no / small Error)
    - NTRUEncrypt
- Isogeny-Based Cryptography
    - Supersingular Isogeny Diffie-Hellman (SIDH)
    - Commutative Supersingular Isogeny Diffie-Hellman (CSIDH)
    - SQISign Signature Scheme
    - Isogeny Graphs
    - Meet In The Middle Attack
    - Galbraith-Petit-Shani-Ti Attack
    - Castryck-Decru Attack

6. Other Cryptosystems

- Hashing
    - Length Extension
    - Rainbow Tables
- Shamir's Secret Sharing
    - Lagrange Interpolation
    - Share Forgery

7. Mathematical Theories

- Counting
    - Permutation and Combination
    - Pigeonhole Principle
    - Binomial Theorem
- Functions (Surjectivity, Injectivity, Inverses, Composition)
- Relations (Reflexivity, Symmetry, Transitivity, Equivalence Relations)
- Group Theory
    - Cyclic Groups, Subgroups, Group Order

    - Lagrange’s Theorem for Orders
    - Group Homomorphisms
    - Permutation Groups
    - Group Actions
- Linear Algebra
    - Vector Space and Norm
    - Matrix Multiplication and Operations
    - Determinant and Invertibility
    - Eigenvalues and Eigenvectors
    - Diagonalisation
- Lattice Reduction (LLL)
    - Shortest Vector Problem
    - Closest Vector Problem
    - Lattice Enumeration
    - Orthogonal Lattices
- Field Theory
    - Finite Fields

Forensics

Contestants should know and understand how to extract information from data files as well as interpreting these data. In general, knowledge spanning both Windows and Linux systems is expected.

1. Files

- File structures e.g. file headers, magic bytes
- Different file formats e.g. PDF, JPG, PNG, WAV
- File metadata analysis e.g. modified times, EXIF data, hashsums
- File carving
- Corrupted file repair
- Decrypting encrypted files
- Steganography

2. Disk Images

- Disk image analysis with different tools e.g. Autopsy, FTK Imager
- Different formats for disk images
- Different filesystems e.g. NTFS, FAT32, ext4
- Filesystems and common system files e.g. /etc, /bin, /home for Linux, and system hives, registry entries, user home directories for Windows
- Recovering deleted files from filesystem images
- Recovering deleted partitions from disk images

3. Networking

- OSI 7 Layer model of network communication
- Common network protocols e.g. FTP, SSH, HTTP(S), SMB, TCP/UDP
- Usage of Wireshark to interact with and analyse packet capture (.pcap) files
    - Decrypting encrypted packets
    - Extracting files transmitted over protocols (webpages, images)
    - Following TCP streams
    - Applying filters to packet captures

4. Memory Dumps

- Usage of Volatility to interact with and analyse memory dumps
    - Dumping memory from memory pages
    - Reading currently running processes
    - Identifying suspicious or unusual processes

Reverse Engineering

Contestants should know and understand how executables (e.g. EXE, ELF) work. They are expected to be able to use decompilers to read and understand such obfuscated executables.

1. Fundamentals

- The C Programming Logic (e.g. pointers)
- Assembly Instruction Sets (limited to x86-64 ISA, ARM64)
- Computer Organisation (e.g. Registers, Memory)

2. Obfuscation

- Variable/Name Obfuscation Techniques
- String Obfuscation Techniques (e.g. XOR, Base64)
- Control Flow Obfuscation (e.g. Opaque Predicates, CFG Flattening)
- Packing and Encryption (e.g. Polymorphism, Packers)
- Virtualisation/Interpretation (e.g. Byte Virtual Machines)
- Junk Code/Dead Code Insertion
- Mixed Boolean Arithmetic
- Instruction Reordering, Instruction Substitution

3. Static Analysis

- File formats and metadata (e.g. ELF File Format, PE32/PE32+ File Format)
- Disassembly
- Shellcode analysis
- Basic block identification
- Entry point identification
- Control flow graphs
- Memory mapping
- Struct demangling
- Recognising high-level constructs
- Decompilation (e.g. use of decompilers such as Ghidra SRE, IDA)
- Automated analysis with decompilers
- Intermediate Representations/Intermediate Languages (IRs/ILs) (e.g. Java Bytecode, .NET CIL)
- Lifting to IRs/ILs
- Binary Patching and Rewriting
- Anti-disassembly Techniques
- Pattern matching (e.g. YARA)

4. Dynamic Analysis

- Debugging and Breakpoints (e.g. use of debuggers such as GNU Debugger, x64dbg)
- Dynamic binary instrumentation (e.g. Frida)

- Dynamic taint analysis
- Core dumps and execution traces
- Anti-debugging Techniques
- Memory Dumping
- Function/library hooking

5. Symbolic Execution, Satisfiability, Black Box Analysis

- Bruteforcing
- Side-Channel Attacks
- Fuzzing Techniques
- Symbolic Execution (e.g. angr)
- Concolic Execution
- SMT/SAT Solvers (e.g. z3)
- Expressions synthesis

6. Emulation and Virtualization

- Binary emulation (e.g. Qiling, Unicorn)
- Virtualization (e.g. QEMU)

Web Exploitation

Contestants should know and understand the common vulnerabilities present in web servers and websites. They are expected to be familiar with exploiting both client and server-side vulnerabilities.

1. Fundamentals

- Networking
    - IP Addressing
    - Subnets, LANs, VLANs
    - TCP/IP ports
    - Domains, Subdomains and DNS
    - Proxies
- HTTP Servers
    - Request Types
        - POST, GET, HEAD etc.
        - Websockets
    - HTTP Request Structure
        - Routes
- Client Data Storage
    - Local Storage
    - Session Storage
    - Cookies
- Languages
    - HTML, JS, CSS
    - Python
    - Structured Query Language (SQL)
    - GoLang
- The Browser
    - Cross-Origin Resource Sharing (CORS)
    - Content Security Policy (CSP)
    - Document Object Model (DOM)
    - Dev Console
- Authentication
    - JSON Web Tokens (JWT)
    - Middleware

2. Cross Site Scripting (XSS) Attacks

- Basic XSS attacks
- Common filter bypasses
- CSP bypasses
- CORS bypasses

- Sanitiser bypass
    - Client-side sanitisation bypass
    - Server-side sanitisation bypass
    - Parsing Differentials
- Data exfiltration techniques
- DOM based XSS
- Stored XSS
- Reflected XSS

3. Database Injection Attacks

- SQL Injection
    - Basic SQLi
    - Time based SQLi
    - Blind SQLi
    - Error based SQLi
    - SQLi to RCE techniques
    - Common filter bypasses
- NoSQL Injection
    - MongoDB
    - GraphQL

4. Server Side Vulnerabilities

- Command Injection
- Path Traversal
- File Inclusion
- Template Injection
- Insecure Deserialisation
- Race Conditions
- Code Execution
- Timing Attacks
- Server-Side Request Forgery (SSRF)
- Prototype pollution
- Parsing Differentials
    - Proxies
    - Backend differences
- XML External Entity Injection (XXE)
- Logic Errors
- Misconfigured Settings

5. Authentication Bypass

- Broken Authentication
    - Bad logic

        - Misconfigurations
    - Open Authorisation (OAuth)
    - Single Sign On (SSO)
    - Multi Factor Authentication Bypass
    - Middleware Bypasses

6. Miscellaneous Attacks

- Esoteric language behaviour/quirks
- Library misconfigurations

Permitted Materials and Software

During the NCO 2026 Prelims, Contestants will **not** be able to access the internet beyond the contest website and whitelisted documentation. As such, Contestants are expected to download and configure any software they deem necessary for the competition **prior** to the competition itself.

Furthermore, Contestants **will be allowed to access offline materials**, both in softcopy and hardcopy forms e.g. saved PDFs, template scripts, printed notes.

Contestants are reminded that the following Operating Systems are officially supported:

- Windows 11 24H2+
- Ubuntu 24.04 LTS

Contestants may use other Operating Systems, however the NCO 2026 Organising Team cannot provide any guarantees that all software and challenges will run smoothly on other systems. It is heavily recommended that you use a 64-bit x86 system. Modern Mac systems (aarch64 Apple Silicon devices) may not be supported. Additionally, Contestants are discouraged from running Dual Booted systems. **Contestants will not be allowed to remotely connect to/mirror an external server/device as your playing environment.**

Beyond **OBS** screen recording and **OpenVPN** software, The NCO 2026 Organising Team will not impose any required software for the competition. However, it is heavily recommended that all contestants have the following installed to be able to interact with some of the challenges:

- Python 3.12+
- Docker
- Netcat
- An IDE or text editor e.g. Visual Studio Code

At the end of this document, a list of additional recommended tools and software will be included as well.

As there is a complete and total ban on Artificial Intelligence (AI) tools for NCO 2026 Prelims, the following software are **strictly prohibited**:

- Local Large Language Models (LLMs) e.g. Qwen3, CodeLlama
- AI Chatbots e.g. ChatGPT, Google Gemini
- Automated Code Agents e.g. Claude Code, OpenAI Codex, Gemini CLI
- Copilots/AI Assisted Code Completion e.g. Github Copilot, VSCode Copilot
- Model Context Protocol (MCP) Tools e.g. GhidraMCP, IDA MCP
- Any other LLM or Generative AI integration into external tools

In general, tools that leverage Generative AI i.e. LLMs, Diffusion Models, will be banned. The NCO 2026 Organising Team acknowledges there exists software that leverages AI as part of its

processing workflow and does not fully automate solving of challenges e.g. Pylingual, which uses NLP to decompile compiled Python. Such software ought to be permitted, but it presents an ambiguous grey area where there may or may not be an **extreme** unfair competitive advantage.

A non-exhaustive list of examples for software or tools which may be permitted includes:

- Premium/paid versions of software which provides features which may aid in certain tasks e.g. paid decompilers providing different Intermediate Representation levels
- Boilerplate or template scripts created before the contest
- Fuzzers or other bruteforcing tools that do not leverage Generative AI

It is recommended that a Contestant utilises alternative tools that do not require AI to avoid ambiguity in AI use. As the NCO 2026 Organising Team does not wish to completely restrict the tools a Contestant may use**, if a Contestant is unsure as to whether a specific tool is permitted, Proctors are to compile such queries and contact the NCO 2026 Organising Team (dcsbox42@nus.edu.sg).** Otherwise, in the event of a screen recording review or later investigation, the Contestant may be disqualified for using AI powered tools that have not been approved.

As internet access is restricted during the contest, Contestants are additionally reminded to ensure their tools and software are able to be used offline. Exceptions may be made for certain software (primarily those outlined in the recommended software section) but the NCO 2026 Organising Team provides no guarantees in this regard.

Recommended Tools and Libraries

The following section details tools and libraries recommended by the NCO 2026 Organising Team to assist with solving challenges during NCO 2026 Prelims. This is a non-comprehensive list and Contestants are recommended to install their own tools and libraries as they see fit, subject to the conditions outlined in the previous section. **Contestants will not be able to download these tools during the competition itself** due to the restrictions imposed on internet use.

General Software

- Python
- NodeJS
- Golang
- Rust Toolchain
- Netcat
- Docker
- OpenVPN

Binary Exploitation

- pwntools
- GNU Debugger (gdb)
- pwndbg or GDB Enhanced Features (gef)
- onegadget
- patchelf
- pwninit

Cryptography

- PyCryptodome
- Sagemath
- gmpy2
- sympy
- numpy
- RsaCtfTool

Forensics

- Volatility 2
- Volatility 3
- FTK Imager
- Sleuthkit Autopsy
- SQLite Browser

- xxd / hexedit
- Wireshark
- exiftool

Reverse Engineering

- objdump
- Ghidra SRE
- Interactive Disassembler (IDA)
- Radare2
- x64dbg
- WinDbg
- angr
- capstone
- z3-solver

Web Exploitation

- Burp Suite
- Firefox Browser
- Chromium