ASSEMBLY PROGRAMMING (x86) WITH NASM

2016 – WEAKNET LABORATORIES – WeakNetLabs@Gmail.com
Course Sheet Index

Course Introduction

Pg 3 - A full introduction to Assembly Programming (X86) with NASM

Course Requirements

Pg 3 - The requirements for this course

Course Materials

Pg 4 - Obtaining all course materials

Course Outline

Pg 4 - Section 1 – Getting Started

Information on getting started with the course materials

Pg 4 - Section 2 – General Computing Requirements

Requirements for this course

Pg 5 - Section 3 – Assembly Language

Syntax, Instructions, and full lessons

Pg 6 - Section 4 – Hacking

Buffer overflow vulnerability analysis and exploiting and reverse engineering

Course View

Pg 6 - Mobile Browsers
Course Introduction

Have you ever found yourself browsing to Wikipedia to learn something about a subject and several hours later you are still at Wikipedia, but learning about something much more fundamental that makes up the foundation of the original subject that you were initially researching? Are you someone who is easily distracted, loves learning, and just needs to know how things "tick"? Are you fascinated by how a CPU processor or computer memory works? Do the fields of reverse engineering, exploit writing and vulnerability analysis, malware analysis, or even programming sound interesting to you?

This course introduces the student to the world of Assembly programming with the Netwide Assembler, NASM. To develop a strong background in Assembly can benefit anyone who is interested in reverse engineering, writing and exploring exploits, malware analysis, or interacting directly with the Linux kernel while programming.

Assembly programming is not for the faint-of-heart and is not recommended as a first computer language to learn. It takes a lot of pre-requisite learning of how processors and memory management works before even writing your first piece of code. This course, bring a culmination of different sources including hands-on experience together and explains, in depth, how your computer works and how to begin programming in assembly for the Netwide Assembler in Linux. The course material consists of different media types, such as text, images and videos.

Welcome to WeakNet Academy!

Disclaimer information for all courses can be found at: https://weaknetlabs.org/disclaimer/
Course Login Link: https://weaknetlabs.org/academy/assembly/
Registration Fee: $25.00USD

Course Requirements

The requirements for this course are outlined in the list below. These will be used throughout the course, but can be substituted in some cases with similar software that you are already proficient with.

- **Modern Web browser** - Google Chrome and Firefox recommended for viewing course materials.
- **A printer** - to print the flash cards and this document (optional but recommended)
- **Dedicated installation of Linux**, or **virtual machine software**, such as Oracle's VMWare Player (recommended). A lesson is provided on setting up and installing Weakerthan Linux 6 with VMWare Player.
- **Linux - Weakerthan Linux 6** - this Debian-based live DVD ISO has all of the prerequisite software installed and can be installed to a disk easily. Please download the latest ISO here: http://tinyurl.com/wna-nasm-wt6 and follow this video tutorial on how to set up the environment here: https://vimeo.com/144377176 This virtualized Linux environment is a free courtesy download from WeakNet Labs and should be used throughout the course. It comes with all course requirements and software pre-installed.
- **Some understanding of simple programming concepts**, such as logical constructs, loops.
Experience with higher level language syntax, such as Python, C, or Java is recommended but not required.

**Course Materials**

All course material links are available after logging into the course. These are,

- **Flash Card Sets** – for rote memorization and familiarization with terms and common practices
- **Course Videos** – Hours of clearly articulated demonstration videos can be viewed in full-screen
- **The Course Material** – In the form of an exploration friendly wiki, using the MediaWiki software
  - Screenshots and images – Clear images and screenshots to better visualize the course material
  - Code Samples – Clean syntax highlighting that can be copied to the clipboard without the lines numbers mangling the code.

**Course Outline**

Below is a course syllabus for Assembly Programming (x86) with NASM and is subject to change at any time.

*Section 1 - Getting Started*

This section is to help those new to Linux, the Linux shell (Bourne Again, Bash), The Vim text editor, and Evan's debugger (EDB). This section provides enough information to configure the software to be used throughout the lessons and course work. If you do not have these applications installed, it is highly recommended (though not necessary), to use the coursework ISO in VMWare, or Oracle's VirtualBox. If one is not provided, you can simply use the Weakerthan Linux 6 ISO.

- **Setting up an environment for this course**
  - Flash Cards
  - Installing Software
  - Weakerthan Linux 6 (Preinstalled requirements)

*Section 2 - General Computing Prerequisites*

In this section we cover topics such as, the development cycle and good program writing practices, using versioning tools- such as git, software used throughout the course, number systems, assembling and linking and loading, CPU registers, and how our CPU works with memory. This
prerequisite knowledge goes a long way for someone new to programming, especially with assembly!

- **The Development Process**
  - Development Cycle
  - Git/GitHUB Version Control
  - Self Documenting Code and Comments
  - Code Base/Code Reuse
- **Vim**
  - Introduction to Vim
  - Configuration
  - Vim as a hex editor
- **Number Systems**
  - Introduction to Binary
  - Introduction to Hexadecimal
- **Endianness**
- **Assembler and Linker**
  - ELF
  - NASM
  - LD
  - GCC
- **CPU Registers**
- **Memory Management**
  - Memory Segments
  - Memory Segment Addressing
  - Memory Stack and Stack Pointers

**Section 3 - Assembly Language**

We can now move on to writing and analyzing assembly code. It's crucial to remember that without truly grasping how the CPU handles memory addressing and uses its own internal registers, none of the assembly instruction that we are about to cover will mean anything and it would be nearly impossible to write efficient code. If you have not explored the lessons and videos above, I implore you to so before continuing, even if it is to simply brush up on your skills.

- **Machine Instructions**
- **Opcodes and the Linux Kernel**
  - Opcodes and System Calls
- **Assembly Program Segments**
- **CPU Instructions**
- **Programming Lessons**
  - Hello World!
  - Makefile Syntax
  - Variables
  - Defining Custom Functions
  - Jumping
Simple Arithmetic
Program Arguments
User Input
Bitwise Operators
Odd or Even Input
Including Custom Libraries
C Library Functions
Stack Frames
Advanced Input and Arithmetic
- Debugging Applications
  - Debugging Symbols
  - GDB
  - EDB

Section 4 - Hacking

Using our knowledge of Assembly in these next few lessons, we will be hacking. We will examine compiled code for a serial key to bypass the anti-piracy security, and even write a buffer overflow which will pass shells commands to the shell.

- Buffer Overflow Vulnerability Exploiting
  - NOP Sled
  - Shell Code Payload
  - Bypassing Kernel Protection
  - Setting Up GDB
  - Buffer Overflow
- Reverse Engineering
  - Bypassing Serial/Product Keys

Course View

WeakNet Academy works well with most modern mobile devices. When using a mobile device, the course material will automatically format to your screens viewport. The only difference should be the code snippets that use the `<syntaxhighlight>` tags. To avoid this you can switch to desktop mode, which looks well on most tablets, using the “Desktop” link at the bottom of the page as shown in the screenshot below.

1 Modern – HTML5 compatible browsers.
To switch back to the mobile view, simply click the “Mobile View” link in the same place at the bottom of the page.

Below is an image of the “Desktop” view on an iPad Air 2.

Below is a screenshot taken from the Blackberry Classic of the mobile view.
**Source Code**

Below is the full Assembly language syntax for a "Hello World" application,

```
section .data
    msg db "Hello World. From Weaknet Lab", 0xA

section .text
    global _start

    .start
    mov AL, 4 ; sys_write() (single bytes only require the low value byte of a register)
    mov DL, 1 ; arg1: /dev/stdout file descriptor (our screen)
    mov ECX, msg ; arg2: string as defined in data section (to an address 4 bytes, requires mov DL, msg_len)
    mov DL, msg_len ; arg3: length of string (single byte also)
    int 0x80 ; pass control to kernel

12. mov AL, 1 ; sys_exit() needs 1 argument. (single byte values for both)
13. mov DL, 0 ; return 0 to DS
14. int 0x80 ; pass control to kernel

section .data ; segment for uninitialized data

Note: AL is the lowest byte of the EAX accumulator register.
The General Purpose registers can be broken down into sections, as so,

<table>
<thead>
<tr>
<th>EAX</th>
<th>AX</th>
<th>AH</th>
<th>AL</th>
</tr>
</thead>
</table>

The same goes for BL and DL in the Assembly code above.
```

A Desktop/Laptop browser is recommended for the course lessons and using a virtualized Linux environment, but any modern browser will do.

**Disclaimer and Information**

WeakNet Labs (Douglas Berdeaux) provides each course for a registration fee, per course. There is no base registration fee that includes all courses. Each course has its own monetary value, which is the value of the login for the corresponding private wiki. You are not paying for the course materials. You are paying for the login credentials that are valid for as long as the server and server software remain intact online and support from Douglas Berdeaux via email, weaknetlabs@gmail.com (response times may vary). Please do not share your login information with anyone, or share direct links to any of the media. WeakNet Labs (Douglas Berdeaux) reserves the right to cancel your subscription at any time for any reason.

**Availability**

In the event of a server compromise, defacement, or denial of service attack, WeakNet Labs (Douglas Berdeaux) will take measures to restore the service to a new, private host temporarily until the bug or compromise has been resolved. In the event that a fatal or debilitating affliction or accident to WeakNet Labs (Douglas Berdeaux), the service will no longer be available any longer than 30 days (rent at remote host that serves these courses) and no refund will be issued to any of the students. WeakNet Labs (Douglas Berdeaux) is not responsible for loss of service due to connectivity problems between the student and the course materials but will respond to submitted issues via email (WeakNetLabs@Gmail.com) to possibly assist in troubleshooting. This course is hosted in the United
States of America and connection testing should be done by anyone outside of the United States of America before paying any registration fees. To test access to a course, please click here.

**Security and Privacy**

WeakNet Labs (Douglas Berdeaux) will not, under any circumstances, store your personal information except for your email address, which only becomes a contact in my Google account, and your login name to the course material. You will not be given the requirement to choose a password for the course work, but you have the option to change the password given to you (not recommended). This is to protect your account, and us, from fraudulent use and password re-use attacks on your other personal accounts in the event that these credentials are compromised by an attacker.

WeakNet Labs provides SSL (Secure Socket) connections to, not only the course materials, but all files and services for WeakNetLabs.org, WeakNetLabs.com, and 80211.Ninja. Please refrain from using plain-text (non HTTPS, non SSL) links while logging in, at least on public networks, and please check your browser URL bar before entering your login credentials - every time. Make sure that you are in fact at WeakNetLabs.org, WeakNetLabs.com, or 80211.Ninja.

**Support**

WeakNet Labs (Douglas Berdeaux) will provide as much technical support as possible, but it is up to you, the student, to come prepared to the course material with all prerequisite technical/computer information, software and hardware. If you are ill prepared, the registration can be cancelled and a full refund will be returned to your PayPal account (within the first 24 hours of registration).

**Thank You**

Thank you for your interest in WeakNet Labs' WeakNet Academy service!