Hacker Qualification Standard

Information Warfare Group
United States Naval Academy

Hacker Qualification Standard (HQS), written by the USNA IWG for training purposes. This version was compiled 10 Feb 2016.

Authors

The following individuals have made significant contribution to this publication:

- Blair ”ENOENT” Mason ’16
- Lucas ”foppedisk” Foppe ’17
- Dan ”flackjacket” Flack ’17
- Dennis ”deveynull” Devey ’17
- Jordan ”kaiser” Wilhelm-Wenzel ’17
- Lamont ”supreme” Brown ’18

Contents

Chapter 1

Binary Exploitation

Binary exploitation is the art of using a program's bugs in order to force it to behave in a way that was not originally intended. This behavior is often spawning a shell, but may also be altering other program variables or disclosing a program's internal state. Specifically, Binary Exploitation deals with the process of exploiting binary executables, rather than those written in interpreted languages.

1.1 Basics

This section teaches you the basics of binary exploitation. Upon completion of this section you will have the capability to perform exploits using classical techniques on vulnerable programs.

Prerequisites: x86 Programming - Core

1.1.1 References

a) ENOENT: *Simple Exploit Walkthrough* `http://rbmj.github.io/iwg/exploit/walkthrough`

b) Aleph One: *Smashing the Stack for Fun and Profit* `http://phrack.org/issues/49/14.html`

c) aviv: *Making Shell Code Exploit Ready* `http://www.usna.edu/Users/cs/aviv/classes/si485h/f15/lec/10/lec.html`

d) Wikipedia: *Shellcode (Wikipedia)* `http://en.wikipedia.org/wiki/Shellcode`

e) blexim: *Basic Integer Overflows* `http://phrack.org/issues/60/10.html`

f) Adrián: *How Effective is ASLR on Linux Systems* `http://securityetalii.es/2013/02/03/how-effective-is-aslr-on-linux-systems/`

g) Tom Leek and tylerl: *NX bit: does it protect the stack?* `http://security.stackexchange.com/questions/47807`

h) Bulba and Kil3r: *Bypassing StackGuard and StackShield (section: StackGuard Overview)* `http://phrack.org/issues/56/5.html`

1.1.2 Tasks

1. Perform a simple stack overflow to redirect control flow to a location within a program
   - References: (b)
   - Task: executeme
2. Write and use a standard x86 shellcode on std{in,out}, and know the differences between writing shellcode and typical assembly code
   - References: (a), (c)
   - Task: stacksmash
3. Write and use an x86 socket reuse shellcode
   - References: (d)
   - Task: reuse
4. Write and use an x86 port bind shellcode

- References: (d)
- Task: portbind

5. Exploit a program based on integer overflow
   - References: (e)
   - Task: intoverflow
6. Know how ASLR mitigates exploits and the relative strength of ASLR on 32 and 64 bit platforms
   - References: (f)
   - Task: Board Question
7. Know how NX (aka W^X or DEP) mitigates exploits
   - References: (g)
   - Task: Board Question
8. Know how stack canaries mitigate exploits
   - References: (h)
   - Task: Board Question

Chapter 2

Network Forensics

Network forensics is the branch of digital forensics relating to analysis of computer network traffic in order to figure out what happened. In a CTF context, network forensics means packet captures, giant logs of all the packets that went over a wire, and you will have to find the flag, oftentimes with nothing in the way of a hint other than the title of the challenge. After completing this training, you too will be able to throw yourselves into the ether and become one with the packet capture, while simultaneously attempting to stave off madness and early onset glaucoma. Knowing how to use the tools to automate discovery is helpful, but most of the time it comes down to knowing the protocols, knowing what normal traffic looks like, and then identifying what looks "weird" in the million odd packets you've been given. It's a dirty job, but somebody has to do it.

2.1 Basics

This section teaches you the network forensics

Prerequisites: *None*.

2.1.1 References

a) NIST: *Guide To Integrating Forensic Technoques into Incident Response: Chapter 6* http://csrc.nist.gov/publications/nistpubs/800-86/SP800-86.pdf

b) OWASP: *Logging Cheat Sheet* https://www.owasp.org/index.php/Logging_Cheat_Sheet

c) Terry Slattery: *Network Redundancy ot Resilience?* http://www.nojitter.com/post/240151667/network-redundancy-or-resilience

d) [various]: *Wireshark Tutorial*

e) [various]: *Wireshark Network Analysis: The Official Wireshark Certified Network Analyst Study Guide*

f) [various]: *tcpdump* http://www.tcpdump.org/tcpdump_man.html

g) deveynull: *Protocols and what they're used for* https://docs.google.com/document/d/1jH29-Y5tKCq8Mu8QkyXUEebsDRQHgRzid3dM08R7LPY/edit

h) Wireshark: *Capturing Live Network Data* https://www.wireshark.org/docs/wsug_html_chunked/ChapterCapture.html

i) Wireshark: *Security* https://wiki.wireshark.org/Security

j) Wireshark: *Capture Setup* https://wiki.wireshark.org/CaptureSetup/CapturePrivileges

k) [various]: *tshark* https://www.wireshark.org/docs/man-pages/tshark.html

l) Mihalis Tsoukalos: *Using tshark to Watch and Inspect Network Traffoc* http://www.linuxjournal.com/content/using-tshark-watch-and-insp

m) hackertarget: *tshark tutorial and filter examples* https://hackertarget.com/tshark-tutorial-and-filter-examples/

n) [various]: *Converting PcapNG to PCAP* http://pcapng.com/

o) [various]: *COnverting pcap to pcapng* https://stackoverflow.com/questions/17995186/convert-pcap-pcap-ng-pcap-ng-tools-libraries

p) [various]: *Export pcap data to csv* https://stackoverflow.com/questions/8092380/export-pcap-data-to-csv-timestamp-bytes-uplink-down

q) Netresec: *NetworkMiner* http://www.netresec.com/?page=NetworkMiner

r) Jack Wallen: *Monitor your Network the Open Source way with Etherape* http://www.ghacks.net/2009/01/23/monitor-your-network-the-open-source

s) InfoSec: *Xplico* http://resources.infosecinstitute.com/xplico/

t) Paterva: *Maltego* https://www.paterva.com/web6/products/maltego.php

u) Russ McRee: *Malware behavior analysis* http://holisticinfosec.blogspot.com/2010/04/malware-behavior-analysis-studying.html

v) deveynull: *Net Forensics OSINT* https://docs.google.com/document/d/1hMd2aydd8xWItZZHYsgmpslVn-6lXNXb-4FYLpfXCL4/edit

w) a: *a* http://ferryas.lecturer.pens.ac.id/NetSa_Papers/UsingSnort.pdf

2.1.2 Tasks

1. Perform a simple stack overflow to redirect control flow to a location within a program
   - References: (a)
   - Task: log1
2. Discuss various log sources.
   - References: (a), (b)
   - Task: log2

3. Explain the importance of network visibility and the difference between resilience and redundancy
    - References: (a), (c)
    - Task: log3
4. Can complete the Wireshark tutorial
    - References: (d)
    - Task: Board Question
5. Discuss Wireshark and its uses.
    - References: (d), (e)
    - Task: Board Question
6. Define tcpdump.
    - References: (f)
    - Task: Board Question
7. Discuss capture filters.
    - References: (d), (e)
    - Task: Board Question
8. Discuss traffic colorization.
    - References: (d), (e)
    - Task: Board Question
9. Discuss display filters and how to filter out 'noise'.
    - References: (d), (e)
    - Task: Board Question
10. Discuss following streams and reassembling data.
    - References: (d), (e)

- Task: Board Question

11. Discuss saving, exporting, and printing packets.
    - References: (d), (e)
    - Task: Board Question
12. Define the following application protocols:
    - References: (g)
    - Task: Board Question
13. Discuss how to capture live traffic.
    - References: (h)
    - Task: Board Question
14. Discuss the security risk of running Wireshark as root.
    - References: (i), (j)
    - Task: Board Question
15. Can complete the tshark tutorial
    - References: (k)
    - Task: Board Question
16. Discuss tShark and what situations you would use it.
    - References: (k), (l), (m)
    - Task: Board Question
17. Understand the difference between pcap, pcapn, and csv, and how to convert between them.
    - References: (n), (o), (p)
    - Task: Board Question
18. Discuss netWorkMiner and what situations you would use it.

- References: (q)
- Task: Board Question

19. Discuss Etherape and what situations you would use it.
    - References: (r)
    - Task: Board Question
20. Discuss xplico and what situations you would use it.
    - References: (s)
    - Task: Board Question
21. Discuss Maltego and what situations you would use it.
    - References: (t), (u)
    - Task: Board Question
22. Discuss various OSINT tools and what situations you could use them.
    - References: (v)
    - Task: Board Question
23. Discuss how you could use an IDS for network forensics.
    - References: (w)
    - Task: Board Question

Chapter 3

Networks

Networks are foundational to information security. Networks enable the global flow of data and communications. This chapter will provide a variety of objectives with supporting links to guide an initial exploration into the subject of computer networks. This is a good chapter to start on. Some of the challenges for this section are still under construction.

There are references for this entire chapter at `https://drive.google.com/a/usna.edu/file/d/0B6XeW7tp_mQzRUZyTHBxMklzQ2c/view?usp=sharing`

3.1 Introduction to the TCP/IP Stack

This section is a general introduction to broad networks concepts, which will be necessary in the following sections. The TCP/IP stack is quite powerful, and it has been the backbone of all major computer networks since ARPANET, with few philosophical changes to the protocol suite in nearly 50 years of running the world's networks.

Prerequisites: *None*.

3.1.1 References

a) Wikipedia: *Circuit switching* https://en.wikipedia.org/wiki/Circuit_switching

b) Tim Garcia: *[IC322] Introduction & Performance* https://drive.google.com/open?id=0B6XeW7tp_mQzY2lrOUZCSHNUd0k

c) [StackOverflow] JP Doherty: *Flow Control vs. Congestion Control* http://stackoverflow.com/questions/16473038/

d) Charles Kozierok: *TCP/IP Architecture and the TCP/IP Model* http://www.tcpipguide.com/free/t_TCPIPArchitectureandtheTCPIPModel.htm

e) Charles Kozierok: *TCP/IP Protocols* http://www.tcpipguide.com/free/t_TCPIPProtocols.htm

f) IETF/R. Braden: *RFC1122 - Requirements for Internet Hosts* https://tools.ietf.org/html/rfc1122

g) Wikipedia: *Encapsulation* https://en.wikipedia.org/wiki/Encapsulation_%28networking%29

h) Charles Kozierok: *Data Encapsulation, PDUs, and SDUs* http://www.tcpipguide.com/free/t_DataEncapsulationProtocolDataUnitsPDU
htm

i) Wikipedia: *OSI Model* https://en.wikipedia.org/wiki/OSI_model

3.1.2 Tasks

1. Explain the difference between circuit switched and packet switched networks

    - Task: Board Question

2. Explain the four sources of delay in packet switched networks

- Task: Board Question

3. Know and understand the 5 layer TCP/IP stack
   - References: (d), (f) (section 1.1.3)
   - Task: Board Question
4. Understand the concept of a protocol, and name the protocols at each level of the TCP/IP stack
   - References: (e)
   - Task: Board Question
5. Understand the concept of encapsulation
   - References: (g), (h)
   - Task: Board Question
6. Know how the 7-layer OSI model corresponds to the standard TCP/IP stack
   - References: (i)
   - Task: Board Question

Chapter 4

Reverse Engineering

Reverse engineering is the art of converting a program's machine code to a higher-level language. Doing so allows one to find vulnerabilites in the program or ways to get around blocks in the program. It requires understanding how processors execute machine code, reading assembly code, and reconizing patterns of compilers.

4.1 Basics of CPU architecture

This sections teaches you the basics of how a CPU is built. This knowledge will make reversing assembly code easier because you will know what each instruction is doing to the hardware. Upon completion you will be able to describe the basic components of a CPU.

Prerequisites: *None.*

4.1.1 References

4.1.2 Tasks

Chapter 5

Unix Proficiency

Unix (and Unix-like, e.g. Linux) is the operating system of choice for most production servers. Its simple, coherent design is both powerful and flexible. Most advanced computer users prefer to use Unix, and in general it provides the best environment for most tasks in the information security field. This chapter is designed to teach the essential knowledge to be comfortable in a Unix environment and the prerequisite skills for other chapters.

5.1 Basics

This section contains all of the knowledge necessary to be proficient in using Unix and the shell. One of the most basic components of this is knowledge of the various commands that are common to most production Unix machines. The following commands are considered important enough to be required knowledge. Those marked with an asterisk (*) need to be memorized (usage and important switches); for those without, simple knowledge of the command name and function are sufficient.

- apt-get/aptitude: (Debian) package manager
- cat: concatenate and print files
- cd: change directory
- chmod*/chown/chgrp: file ownership/permissions

- cp*: copy
- curl/wget: download file
- cut: select text columns
- diff: difference between files
- dmesg*: (Linux) dump kernel ring buffer
- echo*: print
- file*: guess file type
- find: show all files
- grep*: text search
- head*: show beginning
- ifconfig: show/configure network information
- jobs: show job information
- kill*: kill process
- less*: interactively scroll long files
- ln: link
- ls*: list files
- man*: show manual
- mkdir*: make directory
- mount/umount: mount/unmount filesystem
- mv*: move
- netstat: show network connections
- nslookup: DNS lookup

- passwd*: change password
- ping*: ping host
- pwd*: print working directory
- rm*: remove/delete file
- scp: copy over ssh
- sort: sort lines
- ssh*: secure shell (remote terminal)
- sudo*: superuser do (run as root/admin)
- tail*: show end
- top: show CPU/memory usage information
- traceroute*: Show network routing trace
- vi*: standard Unix text editor (basic editing only)
- whoami*: print current user

Prerequisites: *None.*

5.1.1 References

a) William Shotts: *The Linux Command Line (book, free pdf available)* `http://linuxcommand.org/tlcl.php`

b) [Various]: *Unix Manpages (man)*

c) *Bash Command Index* `http://ss64.com/bash/`

d) *Linux Commands* `https://linuxconfig.org/linux-commands`

e) Avishek Kumar: *Linux Directory Structure* `http://www.tecmint.com/linux-directory-structure-and-important-files-paths-explained/`

f) *Important Files and Directories* `http://www.dba-oracle.com/linux/important_files_directories.htm`

g) Fedora Project Wiki: *systemd* `https://fedoraproject.org/wiki/Systemd`

h) Joe Brockmeier: *Managing System Startup and Boot Processes* `http://www.linux.com/learn/tutorials/404619-manage-system-startup-and-`

i) Wikipedia: *man page* `https://en.wikipedia.org/wiki/Man_page`

j) *man-pages project* `https://www.kernel.org/doc/man-pages/`

k) Norman Robinson: *Reading Man Pages* `http://linuxcommand.org/reading_man_pages.php`

5.1.2 Tasks

1. Know exactly what the command line is and how it is used
   - References: (a) (pg 26-30)
   - Task: Board Question
2. Undertstand the Unix filesystem
   - References: (a) (pg 31-36)
   - Task: Board Question
3. Know the function of all of the commands listed in the intro section, and use them
   - References: (b), (c), (d)
   - Task: cmds
4. Know the syntax and semantics (including some important switches) to all commands marked with an asterisk in the intro section by memory

- References: (b), (c), (d)
- Task: cmds_mem

5. Know how to read and use a manpage, and explain the various parts that make up a manpage
    - References: (a) (pg 69-71), (k), (i), (j)
    - Task: manpg
6. Be able to manage user accounts and permissions in Unix
    - References: (a) (pg 112-131)
    - Task: accounts
7. Know the locations and purposes of important configuration files
    - References: (e), (f)
    - Task: files
8. Have a basic understanding of the init system, and be familiar with the concept of services and runlevels (Note: Debian/Ubuntu have both switched to systemd)
    - References: (h), (g)
    - Task: init

Chapter 6

Web Exploitation

Web exploitation leverages access given to servers via their public facing interfaces - webpages. Imporperly managed websites can lead to security risks. Further, webpages are exceptionally common and are an integral part of society today. The ability to exploit these pages offers a strong chance of gaining access to an otherwise protected network.

6.1 Introduction

This section introduces the basics of the web and its operation - server-side and client-side languages and different request types.

Prerequisites: *None.*

6.1.1 References

a) w3 Consortium: *Hyper Text Transfer Protocol* `https://www.w3.org/Protocols/rfc2616/rfc2616.html`

b) codecademy: *Client Side vs. Server Side* `http://www.codeconquest.com/website/client-side-vs-server-side/`

c) Pavan Podila: *HTTP: The Protocol Every Web Developer Must Know Part 1* `http://code.tutsplus.com/tutorials/http-the-protocol-every-web-d`

d) cURL: *cURL Documentation* http://curl.haxx.se/docs/httpscripting.html

e) Paul Hudson: *Cookies and Sessions* http://www.hackingwithphp.com/10/0/0/cookies-and-sessions

f) Paul Hudson: *Session Fixation* http://www.hackingwithphp.com/17/1/11/be-wary-of-session-fixation

g) Wikipedia: *Session Hijacking* https://en.wikipedia.org/wiki/Session_hijacking

h) OWASP: *Session Hijacking Attack* https://www.owasp.org/index.php/Session_hijacking_attack

i) Nitin Bagoriya: *Hijacking Tutorial* http://computercracker.blogspot.com/2013/07/session-hijacking-tutorial-cookie.html

j) Paul Hudson: *PHP Security Concerns* http://www.hackingwithphp.com/17/0/0/security-concerns

k) OWASP: *PHP Security Cheatsheet* https://www.owasp.org/index.php/PHP_Security_Cheat_Sheet

l) flakjacket: *Structured Query Language Injections* https://drive.google.com/open?id=0B4HU5SojWBVvMG1oTmtYWGNLOFE

m) flakjacket: *SQL Filter Evasion* https://drive.google.com/open?id=0B4HU5SojWBVvQWI0dHZSNEdUYnM

n) OWASP: *SQL Side Channel Attacks* https://www.owasp.org/images/c/cd/Side_Channel_Vulnerabilities.pdf

o) Patrik Karlsson: *SQL Out-Of-Band Attacks* https://www.defcon.org/images/defcon-15/dc15-presentations/dc-15-karlsson.pdf

p) Johannes Dahse: *Exploiting hard filtered SQL Injections* https://websec.wordpress.com/2010/03/19/exploiting-hard-filtered-sql-injectio

q) OWASP: *Testing for Local File Inclusion* `https://www.owasp.org/index.php/Testing_for_Local_File_Inclusion`

r) OWASP: *Code Injection* `https://www.owasp.org/index.php/Code_Injection`

s) Wikipedia: *Arbitrary Code Execution* `https://en.wikipedia.org/wiki/Arbitrary_code_execution`

t) Portswigger: *Burp Suite* `https://portswigger.net/burp/`

u) FoxyProxy: *Foxy Proxy* `https://getfoxyproxy.org/`

v) Mozilla: *Web Console* `https://developer.mozilla.org/en-US/docs/Tools/Web_Console`

w) Google: *Using the Chrome Console* `https://developer.chrome.com/devtools/docs/console`

6.1.2 Tasks

1. Understands the difference between client and server side languages (JS, PHP, SQL, etc.) and protocol operation(GET, POST)
   - References: (a), (b), (c)
   - Task: Board Question
2. Can produce a manually crafted web-page request using CURL for both GET and POST.
   - References: (d)
   - Task: Board Question
3. Knows how cookies and sessions are used in applications and what role they provide in vulnerabilities
   - References: (e), (f)
   - Task: Board Question

4. Can perform a basic cookie stealing/session hijacking attack
    - References: (g), (h), (i)
    - Task: steal_cookie
5. Knows about generic issues with PHP applications (weak typed, functions, logic)
    - References: (j), (k)
    - Task: php
6. Can identify an SQL injection vulnerability and produce a generic injection to bypass the WHERE clause
    - References: (l)
    - Task: sql1
7. Can write SQL injections to perform specific tasks (UNION, Side-channel, Out-of-band)
    - References: (l), (n), (o)
    - Task: sql2
8. Can take an SQL injection and transform it in different ways to produce an injection to bypass a filter
    - References: (m), (p)
    - Task: filtet_sql
9. Can identify and leverage the ability to view arbitrary files on the remote server (LFI)
    - References: (q)
    - Task: lfi
10. Understands how to identify and exploit application vulnerabilities to lead to remote code execution (shell injections, PHP uploads, etc.)
    - References: (r), (s)

- Task: code_execution

11. Understands what tools are available to assist in development and exploitation of web apps (Burp Suite, proxies, etc.)
    - References: (t), (u), (v), (w)
    - Task: Board Question

Chapter 7

x86 Programming

Introduction: x86 Programming is writing programs in the assembly language for the Intel 80386 processor and its derivatives. This assembly language directly corresponds to the instructions executed by the processor, and is the lowest common denominator of all compiled programs. It is also the highest level representation that an arbitrary binary executable can be transformed into, so it is necessary to understand x86 Programming in order to reverse engineer and exploit binary programs.

7.1 Core

This section teaches you the basics of x86 Programming. After completing this section you will have the ability to write simple programs in x86 assembly language and have a greater understanding of how programs are laid out in memory.

There are no formal prerequisites for this section; however, you should be fairly comfortable with C programming before beginning this section. The lecture videos listed in the references subsection are highly recommended for those who are less comfortable with C programming.

By the end of this section, you should know the following instructions:

- push/pop
- mov/movzx/movsx/lea

- call/ret
- add/sub
- inc/dec
- neg/not
- and/or/xor
- shl/shr/sar
- imul/idiv
- cmp/test
- jmp/ja/jae/je/jbe/jb/jg/jge/jle/jl/jne/jz/jnz

Note that many of these opcodes have the same logical effect, but operate on signed vs. unsigned values.

Prerequisites: *None.*

7.1.1 References

a) ENOENT: *Quick x86 Architecture Introduction* `http://www.d3catur.net/resources/x86arch`

b) Paul Carter: *PCASM Tutorial* `http://www.drpaulcarter.com/pcasm`

c) Jerry Cain: *Stanford CS107 Lectures (especially lectures 2-4)* `https://see.stanford.edu/Course/CS107`

d) Thomas Finley: *Two's Complement* `https://www.cs.cornell.edu/~tomf/notes/cps104/twoscomp.html`

e) William T. Verts: *An Essay on Endian Order* `https://people.cs.umass.edu/~verts/cs32/endian.html`

f) OSDev Wiki: *ELF (through Relocation, inclusive)* `http://wiki.osdev.org/ELF`

g) aviv: *Disassembling a Program* `https://www.usna.edu/Users/cs/aviv/classes/si485h/f15/lec/03/lec.html`

h) Himanshu Arora: *Objdump Examples* `http://www.thegeekstuff.com/2012/09/objdump-examples/`

i) Kim Skak Larsen: *Intel and AT&T Syntax* `http://www.imada.sdu.dk/Courses/DM18/Litteratur/IntelnATT.htm`

j) Eli Bendersky: *Where the top of the stack is on x86* `http://eli.thegreenplace.net/2011/02/04/where-the-top-of-the-stack-is-on-x86/`

k) Intel Corporation: *Intel Reference Manual, Volume 2*

l) blexim: *Basic Integer Overflows* `http://phrack.org/issues/60/10.html`

m) Adrián: *How Effective is ASLR on Linux Systems* `http://securityetalii.es/2013/02/03/how-effective-is-aslr-on-linux-systems/`

n) Tom Leek and tylerl: *NX bit: does it protect the stack?* `http://security.stackexchange.com/questions/47807`

o) Bulba and Kil3r: *Bypassing StackGuard and StackShield (section: StackGuard Overview)* `http://phrack.org/issues/56/5.html`

7.1.2 Tasks

1. Know the difference between little and big endian, and the two's complement representation of integers.

   - References: (c), (d), (e)
   - Task: endian

2. Know what the .text, .data, .bss, and .rodata sections are, and how to use tools to examine the contents of memory sections.

- References: (f), (g), (h)
- Task: sections

3. Know the basic x86 registers and their uses (GPRs, esp, ebp, eip)
   - References: (a), (b)
   - Task: x86reg
4. Know the various names of memory sizes (e.g. dword) and the size and signedness of various C types on 32-bit Linux. Understand how data is represented in memory.
   - References: (c), (a)
   - Task: asmtypes
5. Know the difference between Intel and AT&T syntax
   - References: (i), (b)
   - Task: asmsyntax
6. Be able to write and understand x86 stack diagrams
   - References: (j)
   - Task: stackdiag
7. Know the x86 standard C caling convention
   - References: (b), (a)
   - Task: calling_convention
8. Understand the use of the C opcodes listed in the introduction to this section
   - References: (b), (a)
   - Task: opcodes
9. Demonstrate competency by writing a basic program in x86 assembly, integrating with libc

- Task: talktome

10. Demonstrate competency by writing a basic program in x86 assembly using only system calls
    - Task: talktome2
11. Demonstrate competency by writing a basic recursive function in x86 assembly
    - Task: fibonacci
12. Demonstrate competency by writing a basic iterative function in x86 assembly
    - Task: fibonacci2