Reverse Engineering Skills Inventory

In order to build teams with a diverse set of skills, I encourage you to consider the following skills inventory. It lists some skills that will be useful in the course, and gives a few example projects if you wish to assess yourself.

System Administration / Troubleshooting

There's a lot to this topic, but most of you are probably already well-prepared. Occassionally, you may need to diagnose and repair network connectivity problems among computers in your group. You will also need to configure virtual machines and install various operating systems. Knowing how to use Wireshark, Nmap, and other auditing and diagnosis tools will also be useful.

Programming

This is perhaps the most important skill coming into this course. You'll be able to pick some things up during the course, but having a solid foundation in one or two languages here is key.

Examine the table of tasks and programming languages, then for each box, consider whether you are able to complete the task in the given language. Don't expect to put an X in every box. Some languages were not meant for certain tasks.

	x86 assembly	C	C++	Java	Python	_____	_____
Basics							
Print "Hello, World!"							
Print the sum of numbers read from a file							
Construct a multi-module application							
Algorithms							
Compute the minimal spanning tree of a graph							
Solve the traveling salesperson problem							
Data Structures							
Traverse a linked list							
Traverse a binary tree							
Networking							
Construct a chat server and client							
Write a network multiplayer game of "pong"							

Assembly: This is the "language of reverse engineers." Some previous exposure to assembly programming (for any architecture) will be a valuable asset to your team.

1. What does the following function do?

```
    MOV eax, 0
    MOV ecx, 1
L1: ADD eax, ecx
    INC ecx
    CMP ecx, 10
    JBE L1
    RET
```

C/C++: You or someone on your team should be able to write and read moderately complex programs in C or C++. This will be a topic we cover in class, but you will advance faster if there's an expert on your team.

2. Rewrite the following as a function using a loop:

```
unsigned int fib(unsigned int n) {
    if (n == 0 || n == 1) return 1;
    return fib(n - 1) + fib(n - 2);
}
```

Java or .NET: Bytecodes and intermediate languages are a more advanced topic, but we will have a lab or two for them. Additionally, Java and C# can also serve as scripting languages. While not necessary, having someone on your team fluent on one of these platforms will prove valuable.

3. What is the output of this Java program?

```
public class Example {
    public static void main(String[] args) {
        int a = 10;
        int b = 20;
        System.out.println("Sum: " + a + b);
    }
}
```

4. What fix would you suggest to the author?

Python or other Scripting: Any scripting language is acceptable, but scripts for course materials will be provided in Python. In some cases, upon request, I may make scripts available in alternative languages. Someone with good command of an interactive interpreter will greatly benefit your team.

5. What is the output of the following Python program?

```
def example(n=5):
    a = 1; b = 1
    if n >= 1: yield a
    if n >= 2: yield b
    while (n >= 3):
        c = a + b; yield c
        a = b; b = c; n-= 1

print sum(n**2 for n in example())
```

Algorithms

This is a foundational topic of computer science. You or someone on your team should be well-versed as these will appear in programs that you are examining. Additionally, a number of algorithms (esp., graph algorithms) have application in reverse engineering.

6. Name the following algorithm:

Data: A directed graph of nodes N
Result: X
$X(n_0) = \{n_0\}$
foreach $n \in N - \{n_0\}$ **do**
  $X(n) = N$
end
while *changes in any* $X(n)$ **do**
  foreach $n \in N - \{n_0\}$ **do**
    $X(n) = \{n\} \cup \left(\bigcap_{p \in pred(n)} X(p) \right)$
  end
end

Data Structures

Most platforms provide abstractions for these, so you will not often need to implement them yourself. Nevertheless, custom implementations may appear in applications you're reverse engineering, or abstractions may be static-linked in.

7. What data structure is this probably implementing?

```
public class Node<T> {
    T val;
    Node<T> flink;
    Node<T> blink;
}
```

8. Give at least one rule regarding the relationship of nodes and their fields in this structure.

Networking

Many of the course samples will be using network sockets, esp., as these are popular applications to examine for network-based vulnerabilities. Thus, to interact with the applications provided in this course, you or someone on your team should be familiar with the sockets API in one or more programming languages.

9. Given the following two communicating functions, what might go wrong? (Assume the sockets are already properly connected)

```
ssize_t send_blob(int sockfd, unsigned int length, void* data) {
    if (send(sockfd, &length, sizeof(length), 0) < 0) {
        return -1;
    }
    return send(sockfd, data, length, 0);
}

void* recv_blob(int sockfd, ssize_t* length) {
    ssize_t l = -1;
    if (recv(sockfd, &l, sizeof(l), 0) < 0) {
        *length = -1;
        return 0;
    }
    void* buf = malloc(l);
    if (recv(sockfd, buf, l, 0) < 0) {
        free(buf);
        *length = -1;
        return 0;
    }
    *length = l;
    return buf;
}
```