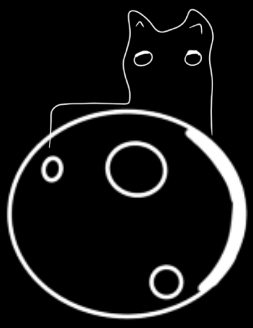

SkyPenguinLabs

Samples

Time Managing The Reverse Engineering Process

SPL-REC4

Powered by SkyPenguinLabs

Introduction

> Learning How To Time Manage The Reverse Engineering Process

As a reverse engineer, I am sure you have spent some time on an engagement or doing some of your research, which was against some timeline, only resulting in much more anguish when you can not pop that deliverable as nicely as you wanted to.

Because reverse engineering tasks take so long to complete, and may often come with a large number of challenges, which can often drain the amount of time on an engagement, and additionally, may have specific segments during the reversing process that make you miss some things, we need to learn how to properly account for the missing time.

In this small and cheap lesson, I wanted to make sure that you walk away with some form of tricks and unique tips that can help you manage the reverse engineering process.

As some people may also predict here, we will be covering much more than learning how to target properly- but also understanding what takes a lot of your time up: such as sifting through documentation, writing too much, over explaining, and simple things that seem like they take 0 time, but take more time than you expect.

The purpose of this lesson is to bring upfront, something many people fail to criticize, and bring up when introducing beginners to the world of reverse engineering. Which for many people suck! Of course, this may be due to how CTFs are often designed, with limited time constraints. Of course, this is for the **surface level**, which most beginners fall into when they start surfing Google for some answers.

Without further to mention, or say, let's get into this! WOOOOOOOOOOOO

Powered by SkyPenguinLabs

Table of Contents (ToC)

> Leaning How to Time Manage the Reverse Engineering Process

The following contents describe the lessons in a summary-like depth.

Powered by SkyPenguinLabs

Section 0x00

> Prerequisites

This course is designed for beginners who have a basic understanding of the reverse engineering world. This lesson, since it's based on theoreticals, does not really require anything heavy. Additionally, no technical environments will be shown outside of the occasional demonstration of the concepts in theory.

Without further to disclose, let's switch over to the crux of the content :D

Powered by SkyPenguinLabs

Section 0x01

> Understanding the 'What' and Context of a Situation

As reverse engineers, we are going to be coming across a large range of various scenarios that end up messing with the time on the clock. In these scenarios, it is important to be able to thoroughly grasp every single part of the situation (context) and the 'what' of the situation.

These can be broken down into a simple bullet point.

- Situational Context - Reverse engineering often applies methodologies *(like other fields in the security space)* to context-dependent scenarios. Of course, 'context dependent' is a way of saying that you can not be certain of your approach until you understand the situational context that goes behind the task at hand. For example, saying 'I will use strings to reverse engineer a file format' when you do not even know the better half of 'what' the file format is, will result in a large amount of confusion down the line.

A massive part of managing time during a reverse engineering operation lies in the way your operations are set up and organized. Part of organizing your operations relies on organizing around the context.

For example, if you, as a reverse engineer, spend 30 minutes of your time before an engagement setting up plugins, making sure you have a proper environment, and creating file names, etc, there is going to eventually be a point where either

- A) You get completely sidetracked with an issue in your organization *(if you are not good at organizing things or use known methods of organization to make things easier)*

- B) You have to wire through a mess of stuff you sped through trying to make it fit into the timeline when it should have been done before.

This is operational. And it is where a large number of people waste their time.

- **Note**: An unfortunate real-world note: Sometimes, at the most unfortunate times possible, you may get handed an operation, or an engagement, which has a tight schedule, WHILST ALSO not receiving enough information from the client. This wastes more time because you either end up arguing with the client to give you more info, or end up wasting more time on something that probably would not even be used when the app gets deployed (*such as a function that the developers did not tell you is for development only*) or simply just missing the scope. Sometimes, arguing wastes more time than it is worth, and you would be better off without the information.

Had you been given the correct context, the correct scope, and information on the situation, you may not have wasted so much time scrambling around trying to collect pieces of missing information.

That being said, there are also INTENTIONAL scenarios where you are given 0 context. This is known as a black box scenario. And is perfectly normal to see in any testing space, reverse engineering or not.

As you can imagine, in both the note and the paragraph above, these are scenarios that are both in the dark with information.

- *So how do you pick up *context* on the situation?*

Have you ever read those big, fat reverse engineering books that walk you through individual components of a specific application?

Sadly, what most authors _*do not*_ tell you is that when reversing an application, you have to take into account the engineering aspect of how the app was designed. And oftentimes, this is how you are going to get your context right for deploying specific techniques in the field.

Let's take this into account.