SkyPenguinLabs

Samples

Static Analysis for Ethereum Detection using Go w/ELF

SPL-REC5

Powered by SkyPenguinLabs

Introduction

> Writing Software Using Go to Detect the Presence of Ethereum in ELF Files

The world of finances has been changing for quite some time. And with the addition of blockchain, developers have spent a large amount of time trying to implement these systems into regular everyday apps.

This may be for the atypical licensing system or for the most malicious cases of wallet drainers. Regardless, there is a large amount of implementation and cross between web3.0 systems and web2.0 applications happening today, which we see on our desktops which look like desktop apps but are web2.0 apps which also integrate these systems.

All of these apps need to rely on remote services which communicate with on-chain applications, or rely on an address or some sort of identifier that maps to a deployed smart contract on-chain.

As reverse engineers, good or bad, knowing if a program is attempting to communicate with a blockchain is important. This is because it allows us to view the true intention of the program, or a part of it anyway. Though, that already, seems like a task. Because "a blockchain" encompasses 1 out of a possible 1,000+ blockchains in the WORLD. Each blockchain, no matter whether derived from popular chains such as Ethereum or Solana, and no matter the layer, have their own characteristics. As you can imagine, it would be pretty helpful to learn how to automate the detection of these characteristics, so we can build a sort of self-library for recognition. Instead of wasting our time on the same repeated processes.

In this lesson, we will be covering one out of thousands of these chains, known as Ethereum, and attempting to build a basic ethereum-spy which can check if a given ELF contains the presence of Ethereum related calls, programs, or related information using the Go programming language.

The purpose of this is so that when inside of focused engagements, or scenarios where investigation becomes apparent on the blockchain, relevant to specific chains, the reverse engineer takes away skill that allows them to identify unique, detectable characteristics for that chain, thus being able to build tool sets which automate this same case scenario further.

Powered by SkyPenguinLabs

Table of Contents (ToC)

> Auto Detecting The Precense of Ethereum in (ELF) Files using Go

The lesson contents below, walk through the process of understanding how the web3 world integrates into the web2.0 space, which then bleeds into understanding apps which rely on the blockchain and how they integrate them technically, finally ending on how we can dig through those apps and detect their presence if hidden.

Section 0x00

> Prerequisites

In order to work through this course, it is preferred that the reader have a core understanding of what blockchains are, what web3 as a field is, what ELF is *(its structure more-so)*, and a baseline understanding of computer science fundamentals related to operating systems, architectures, the x64 ISA.

We will be using the Go programming language in this lesson, so as you can imagine, some Go programming experience is required. Though most of the code that is referenced or explained is very easy to break down and logically comprehend piece by piece and was engineered with simplicity in mind, for the sole purpose of allowing the reader to interpret ways to advance the engineering based on what they learned in this lesson. While a requirement for understanding the structure of ELF is required, this is for ELF as a format, not for any different compiler, or system which modifies, extends, or instruments the structure in any shape or form. So requiring how to instrument it, modify it, or break it is not required.

This is pretty much the foundation of a fresh reverse engineer who decided to take a dive into reverse engineering.

Outside of that, there are no other requirements for this lesson. The below is recommended but not required to comprehend the contents of this lesson.

- If you would like to follow along with reversing, feel free to download the application exe from this link and use IDA-Pro or IDA-Free.

- If you would like to follow along with the development, you will need some sort of text editor, not even an IDE, but those are nice. And the Go compiler, specifically version 1.23.5 Linux.

With that being said, you should have all of the knowledge necessary if you satisfy the contents above to at least grasp the theory and understanding. This is also not that complex of a lesson so we expect it to be relatively easy and suited for beginners :D

The next page begins the start of the lessons actual contents!

Powered by SkyPenguinLabs

Section 0x01

> Web2.0, Desktop App Madness, and the Blockchain

Ever since the world of blockchain has hit the internet by storm and it has skyrocketed into mass popularity within recent years, this push has also continued to drive development efforts in every single portion of the financial world, both technically, and non-technically speaking.

This drive has been kind of wonky, and a very wild and emotional ride. Primarily because the crypto world is flooded with cybercriminals. And when I mean flooded, I do mean FLOODED with cybercriminals. If it is not a cybercriminal, it is some random 12 year old trying to rugpull all your tokens using pump.fun

The issue with this is that it stops both security researchers and developers from wanting to enter the space willingly, as it creates a divide, and a form of uncomfortability amongst the people being involved, whether it is due to the sake of their job or the general interest in financial security.

This divide, distracts a large amount of people from forgetting portions of this world that still exist despite that, the portions nobody talks about.

Firstly, it's important to get this out of the gate.

There is a huge difference between cryptocurrency and the blockchain, the blockchain is what cryptocurrencies build off of. The blockchain itself is a digital, decentralized ledger that securely records and verifies transactions across a network of nodes.

Hint at the word transactions. Transactions typically may involve something financial. However, the blockchain is not just for transactions. For example, to sign a message on the chain, you are making and committing an action to the chain via a signed message. This is a message. Not a transaction. While messages get composed into transactions, they themselves are not solely financial transactions in the sense that many people view them.

With this distinction out of the way, we can back into how people steer away from blockchain.

Speaking of, despite the primary reason people steer away from the blockchain, being cybercriminals whose only interest is to exploit the financial side of the blockchain for some form of gain, the development continues.

So does the integration.

Which changes the way we as reverse engineers need to tackle applications. But **what integration** Do I mean?

For starters - I recently did a talk at HackSpaceCon2025 about the ElectronJS world, and how I saw that recently playing HUGELY into reverse engineering operations recently (*specifically around 2022 and up*). This was due to ElectronJS gaining a wide amount of popularity around that time surrounding their previous updates which changed the entire structure of the framework.

This of course, caused an influx in developers creating Desktop applications utilizing the ElectronJS framework, to develop official looking desktop applications that were both centralized and decentralized at the same time. Ultimately, this is a pretty weird thing to wrap your head around but stick with me. This is the integration.

These applications often use the decentralized blockchain networks for a variety of reasons. This can be something as simple as authorization to validate that you are not a bot + a specific user who read through and followed the apps instructions to install a wallet, connect it, and etc to use the app in the first place, or something as overly complex as utilizing custom on-chain smart contracts to handle transactions between multiple parties in an escrow account.

Regardless, they are going to be reaching out to SOME remote source.

For those of you that are new: Exploring remote portions of an application are a widely discussed and important concept in the reverse engineering space. Regardless of the application, but dependent on the scenario (*such as professional, scoped scenarios that limit targeting anything outside of the application client*) looking for what an application is trying to do outside of the applications execution environment (*external*) and outside of the system (*remotely*) will lead you to services, APIs, and etc which can give you further internal information on systems or components involved. For example, if an app has a standard stripe endpoint, or if we see the app utilizes a KeyAuth endpoint, we know the login system is local and easy to defeat as it relies on a third party auth system (*KeyAuth*).

I say blockchain networks, because in order to interact with any blockchain out there, you have to first interact with a server or deployed system on the internet that hosts the network of nodes for the blockchain to operate off of. Typically, in popular chains, such Litecoin, you will find LTC RPC Nodes or APIs which is what the program will reach out to, in order to start interacting with programs deployed to that network.

The way the network gets indexed, programs get executed, instructions get tracked and put on the ledger differs every single chain. While they all explore the same concepts, each chain is programmatically built out differently or similar to existing chains with extended features and technologies.

For example, one really weird and technical blockchain is the Aurora blockchain, built on top of Ethereum, which leverages a feature called Aurora XCC Callbacks or Aurora Cross Contract Callback. This is a really funky system that allows EVM *(Ethereum Virtual Machine)* based smart contracts to call NEAR (*WASM*) contracts, and handle the results in Solidity. This is something natively, the Ethereum blockchain can not do.

As you can imagine, if this is the way developers are going to work with features on-chain, what do the developers do on the floor trying to integrate this? What mess must it create?

Truly, if you look at MOST crypto apps on the market, a lot of them, especially run with custom tokens (*which ideally is supposed to be for something entirely different than what they are being used for*) are AI generated, slapped up there.

Those apps we do not care about, because those apps we do not see in our laps when working on engagements, and we never see those apps in the field, since they aren't malware more than they are pop-up scam projects.

Because there is a LOT to unpack here, let's start from the most basic angle and try to see why we care about this integration that has happened within the traditional web and app dev landscape.

Well, as reverse engineers, if the development landscape changes, so will the methodologies, fields and types of reversing, as well as the type of tools we employ most commonly will change and rotate, which means the theory will as well.