AlphaNET

Premise:

Hacking games are a very niche and underexplored genre. You've got the bigger games like Hacker Experience, Hacknet, Hackmud, Dark Signs/Hacking Simulator. But, there's an issue with all these games: it's nothing like actual hacking. They are essentially puzzle games, with a hacking/computer theme. Some have programming (they're the better ones!) which does make them a bit more accurate and realistic, but they're not too in-depth. Hackmud for example, makes heavy use of JavaScript scripting. But, to accomplish what in the end? Solving puzzles to get more of the in game currency. Hacking Simulator had programming (in an awful, awful original language), but it was quite limited. You could program your given server project, and do things locally, but that was about it. All these limitations frustrate me! What I want in a hacking game is a nice, open sandbox, with a complete scripting engine and API, so I can write whatever the hell I want; I want hacking that's more than just a simple puzzle; I want a complete, virtual operating system environment to play in to my heart's content!

And that's the goal of AlphaNET. AlphaNET aims to be a *complete sandbox*, with a full featured scripting API, realistic and open operating system internals, complete customization, and can't forget: Actual hacking and cracking! AlphaNET aims to be something you can immerse yourself in, something you can learn from, something you can be apart of.

Gameplay.

Obviously, as a sandbox hacking & programming game, most of the gameplay is up to the user! Write whatever cool thing you want, form communities, try to hack! But, there is one issue: real hacking is hard. Damn hard. That's why I believe most prior hacking games have gone for the limited, puzzle game-esque route. That's easily controllable, and fits more in line with a typical game. If it's completely open, people will just write unbreakable scripts, noobs won't have a single chance, and they'll get frustrated and quit! I've opted for a balanced solution to this problem that maintains the sandbox nature, and can be noob friendly. Most of the included scripts are insecure. The OS scripts, insecure, the included services, insecure, some of the included tool scripts, insecure. With open attack vectors included in the game by default, a noob can read a tutorial on the included vectors, and attempt to try to exploit them! I imagine as a consequence of this, the community will modify these included scripts to be secure, but that's opt in, not everyone will use these secure community forks. So there will always be some portion of the user base with insecure system components. These users provide a constant thing to hack, and at the higher levels, it's all up to how security-conscious people are while programming ;) But, do keep in mind, these insecure script threats pose no threat to a user's real computer! The game is heavily sandboxed, the scripting API allows no I/O access, and I've tried to stay as careful as possible while writing it. But if someone's real computer gets pwn'd, sue me!

Execution:

Tools and libraries:

- .NET Core (Cross platform C#)
- WatsonTcp (For client and server network communications)
- Jint (For the JavaScript engine)
- EntityFramework.Core 6 (The comfiest ORM! Used on the server.)
- TypeScript (For making that JavaScript a whole lot more pleasant!)
- Visual Studio 2019 (The greatest IDE)
- Visual Studio Code (For writing the TypeScript)

Projects:

- AlphaNET.Framework (C#, .NET Standard class library) - This is the core of AlphaNET, containing most of the code that makes up the client side of the game
- AlphaNET.Client.Console (C#, .NET Core Console application) - This is the console client for the game, which consumes the AlphaNET.Framework API. Contains most of the platform specific changes that need to be made, and gets the game up and running.
- AlphaNET.Server (C#, .NET Core Console application) - The tcp server implementation for AlphaNET, facilitates all the networking components of the game (Virtual sockets, and thereby communication between two seperate AlphaNET clients) It will also offer planned paid services for AlphaNET, such as AlphaHosting (Service which allows a user to control a server-side AlphaNET client (sorta like a VPS), great for those who want to run a persistent in game application (web servers, anyone?) but don't have a real server themselves to run a client on 24/7, or don't want to waste resources. This will be one of the game's only monetization sources, I promise!)
- AlphaNET.Client.Visual (C#, .NET Core MonoGame application) - The terminal can get drab sometimes! Visual offers an additional graphics API, based in MonoGame, so users can create their own full featured visual shells. Out of the box, comes with a very simple window manager, display server, and tty. Like Client.Console, it also consumes AlphaNET.Framework's API.
- AlphaNET.Editor (C#, .NET Standard Eto-based class library) - A GUI editor to interact with a AlphaNET Filesystem. You can import a filesystem binary and completely modify it, import arbitrary assets (Images, executables, scripts, etc), and even hook into a running AlphaNET client and modify a live filesystem! Additionally, it could compile TypeScript projects and debug them. (I imagine that will be the biggest use for this, who the hell wants to write huge scripts in game! Unless someone ports over vim, then I'll be more than happy to ;))
- AlphaNET.Editor.Windows (C#, .NET Framework) - Consumes AlphaNET.Editor, for Windows builds (Uses WPF)
- AlphaNET.Editor.Linux (C#, .NET Core) - Consumes AlphaNET.Editor, for Linux builds (Uses GTK 3)

AlphaNET, and all it's derivative projects will be completely cross platform, with the main platform targets being Linux, Windows, and macOS. Thanks to .NET Core and .NET Standard, this is easily achievable! Possible future targets could also include mobile platforms (Android, iOS), but I'm currently not thinking about those too hard, playing a game like this on a phone sounds gross. Additionality, web app clients are a possibility, through utilizing Blazor, and adding WebSockets support to AlphaNET.Server. Like mobile though, that sounds awful.