Whitepaper v1.4

June 2018

RADJAV

http://www.radjav.com/

FogChain, Corp

radjav@fogchaininc.com

Table of Contents

? Introduction

RadJav is a rapid software development platform and decentralized datacenter. The goal of RadJav is to simplify app, dapp, and smart contract development and how these applications are deployed. In today’s software development environment there’s dozens of programming languages to choose from, in addition to thousands of libraries for developers to choose from, each having their own set of pros and cons. Once developers choose the programming languages and libraries they’re going to use, they must learn how to properly utilize these libraries, while ensuring they do not compromise the security of their apps or servers they’re hosting them on. Lastly, they must determine which servers will host their server-side apps, what kind of hardware their servers will need, how they can scale their servers while maintaining performance and security.

With all of these disparate moving parts - it’s no wonder that there’s so many successful cyber-attacks and data breaches.

During the entire software development cycle, if a tiny mistake is made anywhere, it can mean that hackers can gain access to all stored user data, possibly even take over the server, or in some cases spread throughout the entire network the server is attached to. We want RadJav to prevent this as much as possible, while also simplifying the development process. By giving developers tools that are easy to use, flexible, and portable, we can create a platform that developers will use for years to come.

RadJav is designed for compatibility across as many operating systems and platforms as possible, reaching the widest audience. Currently there’s nothing on the market that allows JavaScript developers to code native desktop applications and HTML5 applications using the same code. On top of this there’s nothing on the market that allows the rapid creation of 2D/3D games with multiplayer capabilities on both native desktop machines and HTML5 using the same JavaScript code. With the continuing growth of the mobile app marketplace, giving developers the chance to quickly create secure applications and games that work on desktop, web, and mobile devices using the same code is a new frontier that fills a large void.

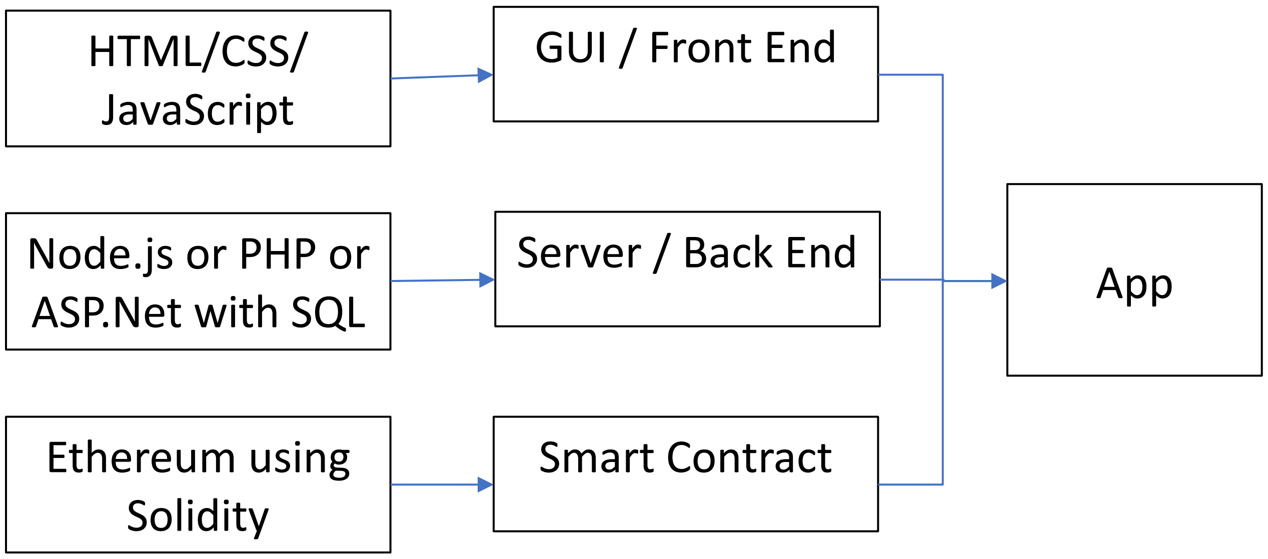
Server-side backend support is also incredibly important as well since nearly all applications today must connect to a server to relay or store data. RadJav includes server-side support and most importantly makes it easy to securely connect to the server from the client application. We're developing RadJav to be a blockchain and a decentralized datacenter, allowing developers to create their server-side applications then execute them on the decentralized datacenter. Rather than developers having to find a separate library for their desktop applications and their HTML5 applications to connect to a blockchain, RadJav includes support for this natively in an integrated platform. Developers will immediately be able to monetize their applications in ways never thought possible by having their apps mining in the background, or allowing their customers to purchase and exchange tokens with the developer. This enablement provides endless possibilities.

Due to the security risks that come with having a decentralized datacenter, it will be rolled out in multiple versions, starting with Blockchain V1, then later V2, and V3. As we develop the datacenter we will work closely with the community to make sure its secure and will execute as quickly as possible. With Blockchain V3, RadJav applications can be distributed over the decentralized datacenter for the developer's customers to download, saving the developers the time and money with setting up their own servers. In cases where huge demand for their application arises, the developer will not have to worry about hiring additional server admins, hardware, and software for scaling their servers which can become extremely costly. On top of this, RadJav Blockchain V2 will give developers the ability to create their own coins or tokens.

Problems RadJav Solves

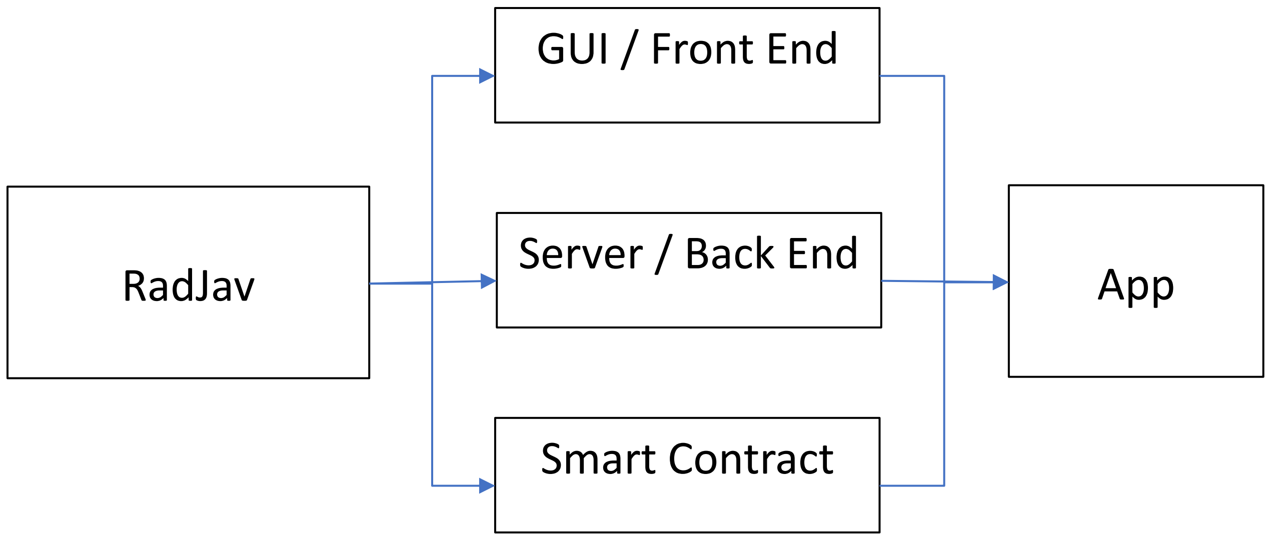
The separate developer skillset problem:

Currently for a software team to create an app that works with a smart contract, they need at least 3 different developers with 3 different skillsets. Here's a graphic to depict this scenario:



The code written by either the front-end developer or back-end developer typically does not port well, so chunks of code will have to be rewritten.

Using RadJav, the same team's skillset is simplified to this:



Having the same team know and work from a common platform is incredibly beneficial, as they can help each other and work more fluidly with one another. It removes the need for each developer to have to work around how the other's system was designed. Developers are also not bound to using solely one operating system, like the .NET framework. Additionally, they are not required to download multiple libraries, then learn how to use them and make them work with the other libraries they wish to use, like with .NET and Java.

The datacenter problem:

Nearly every app today is internet connected and connects to a server. Developers must choose the correct datacenter that is able to scale, is secure, has proper IT protocol and logs, all while remaining as cost effective as possible. Once developers find the right datacenter to use, they must install their server app on it and ensure that they are following the correct protocols to secure it. Unfortunately, quite often the connection between the app and the server can become a weapon for hackers. Hackers can utilize this connection and launch attacks against the server with the aim to breach it. This means all user input between the app and the server must be sanitized, and if a tiny mistake is made; this could mean that a hacker could use the app to steal all info from the server, or possibly even take over the entire server.

RadJav will introduce a Model-View-Controller (MVC) framework that will not only simplify the process of creating a GUI app, but make it more secure. All user input being sent to the server from the app will automatically be sanitized so the developer doesn't have to manually do it, and possibly make a mistake.

The smart contract and dapp problem:

We believe there should be a clear difference between a smart contract and a dapp (or decentralized app). A smart contract should be an agreement between parties without the need for a third party. While on the other hand, a Dapp should be a decentralized app that runs on a blockchain without any limitations. Typically, when you look at agreements, they are very clear and straight forward. With today's smart contracts, that's typically not the case at all. The must be code reviewed and thoroughly tested, and even then, there's still a chance there could be a security vulnerability. Even with states such as Arizona that have ruled smart contracts legally binding, we still must ensure that these smart contracts do not have security vulnerabilities. Smart contracts are too important to even risk the chance of a security vulnerability, especially if it means that the security vulnerability could be legally binding.

With this in mind, we're developing RadJav's smart contract scripting language to be very simple, clear and concise. The end result is the ability to create smart contracts with very little programming knowledge, yet feel comfortable with its security and durability when deployed.

Typically, in software development the more complex the system, the more mistakes that can be made, and the more room there is for a security vulnerability or bug. By making the smart contracts as simplified and straight forward as possible, this will essentially eliminate any room for error.

Dapps on the other hand will have no limitations and developers would have the freedom to create any kind of dapp they wish. However, in all instances their dapp code should be reviewed to ensure they did not accidentally create a vulnerability. While RadJav will provide as many tools to avert the possibility of creating a vulnerability, we ultimately do not have control over dapp developers. Where possible, we will integrate our platform with several code testing services.

What is a blockchain?

A blockchain is typically a decentralized public ledger used for sending coins from one person to another quickly and securely. RadJav uses coins called XRJ coins, which are primarily meant for developers to use on our platform. Each blockchain is secured by computers called “nodes”. Each node on the network verifies transactions, and performs math problems to find “blocks” that contain all the transactions that have occurred within a timeframe.

What’s the difference between apps and dapps?

Apps

The word app is short for application, which really became popular around 2007 with the first iPhone. Apps typically are thought of as software applications that run on desktop computers or mobile devices. For example, Microsoft Word is an app that runs on Windows machines, whereas SnapChat is an app that runs on phones such as Android or iPhone.

Dapps

Dapp is short for decentralized application. This term did not become popular until the release of Ethereum. Most ICOs today are conducted on the Ethereum blockchain using their smart contracts, which on their network can be considered a dapp.

With RadJav's platform - both apps and dapps can be created in a highly secure fashion.

What are we trying to accomplish?

We firmly believe the future is in blockchain technology, and our business model revolves around delivering solutions around three big problems: increase the ease and timeliness of developing Apps and Dapps; increase security, and; dramatically improve transaction speeds and scaling efficiencies. For the next two years we will be mostly focusing on developing Blockchain V2 (XRJV2) and Blockchain V3 (XRJV3). In addition, we want to closely work with the wider developer community using RadJav and discover areas for improvement and refinement.

We will be creating tutorials, case studies, and videos on how to develop applications using RadJav. We will also be providing elementary schools, high schools, learning centers, and colleges with online workshops and tools on how to use RadJav. The more developers that use RadJav, the more applications and Dapps will be developed from it, increasing the use of the RadJav blockchain.

Since coding is difficult, one of RadJav's core missions is to make it much easier on the developers to quickly develop Dapps and apps by making our libraries as developer-friendly as possible while providing them with a platform that takes care of security, extensibility, portability, and high throughput performance. RadJav is built to be modular and flexible, giving developers the freedom to create their own themes, use their own 3D engines, create additional libraries for RadJav, and much more. RadJav's utilizes the MIT license, giving developers as much freedom to do what they want with it, with little to no restrictions.

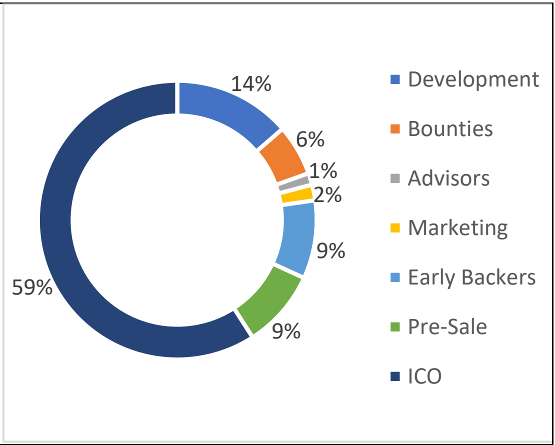
Who has say in development?

Unlike nearly every other programming language and software development platform available today, the community has a say in the development of the platform. Since we are going to be receiving a fractional amount of coins from every block mined, we do not have to rely on large corporations to give us money or charge exorbitant service or licensing fees in order to continue working on the project. We intend to sell these mined coins to businesses, so they can leverage a turn-key high performance blockchain platform and switch their websites and servers over to the XRJV3 network once it's in place.

Coin Distribution

RadJav's coin distribution is allocated as follows:

Developer Count	15,000,000
Bounty Campaign	2,000,000
Bug Bounty Campaign	4,500,000
Marketing and Consultants	2,000,000
Advisers	1,500,000
Early Backers	10,000,000
Pre-Crowdsale	10,000,000
Crowdsale	65,000,000
TOTAL	**110,000,000**



What can be made with RadJav?

With the current RadJavVM 0.11 platform, native Windows apps and HTML5 apps can be created using the same code. Windows apps can connect to the XRJV1 blockchain, in addition to XRJV1 allowing the transfer of XRJ coins from one individual or another.

In a new release coming soon, RadJav will be able to create (using the same code):

- Desktop Apps
- Mobile Apps
- Web Apps
- Games
- Mobile Games
- Web Games
- Server Apps
- Smart Contracts
- Dapps

Nearly any kind of app can be made using one or a combination of the 4 aspects of RadJav listed below.

The Primary Aspects of RadJav

Graphical User Interface (GUI)

A GUI application is your typical desktop application. RadJav applications can be coded using all the common GUI objects such as textboxes, buttons, lists, combo boxes, etc. It’s also been designed in a way where themes can be added to the developer’s GUI application. The HTML5 version of RadJav allows GUI applications to be embedded into already existing HTML, allowing for a seamless transition. On desktop machines, RadJav uses wxWidgets, while on HTML5 it uses Dojo. Eventually custom GUI objects will be available allowing developers to create responsive desktop, mobile, and HTML5 applications all using the same code. An additional library may be created that would be similar to jQuery, giving web developers the ability to reuse a lot of their existing code on RadJav projects.

Responsive

A responsive OpenGL UI framework would be created that would be a UI designer's dream. Using this framework, it would allow developers to create interfaces that would stretch or respond to a user's screen on nearly any kind of device. By creating custom UI objects using OpenGL, UI designers would be able to create interfaces never thought possible before.

IOT

RadJav currently supports serial I/O communications on desktop machines, allowing RadJav to work with many IoT devices. Additional IoT support will come in the future, however it will not be our primary focus for now.

MVC Framework

RadJav is developing a new framework that we call RadJav Interact. RadJav Interact would be a Model-View-Controller (MVC)framework that would sync with data stored on the server automatically as data changes. This framework would allow for developers to create secure apps, very quickly.

Using the GUI aspect, the following apps can be created:

- Desktop Apps
- Mobile Apps
- Web Apps

Game Development

RadJav would give game developers the chance to create either 2D or 3D games. RadJav will utilize a built-in 3D engine, using Ogre 3D on desktop, and Three.js on HTML5. Additionally, RadJav has been designed so different 3D engines can be used, instead of the ones included. RadJav's game development tools also gives developers access to the user's GPU giving them access to massive amounts of computational power. The 3D engine will also feature lots of additional easy-to-use classes, methods, and other tools to help aide game developers quickly create their games.

Using the game development aspect, the following apps can be created:

- Desktop Games
- Mobile Games
- Web Games

Server Applications

A server application communicates with other computers over a network or the internet, serving them data. RadJav will feature its own built-in web server, web socket server, light database, raw TCP/UDP socket server over IPv4 and IPv6. RadJav Interact will include a web socket server and database that connects to GUI applications, and sync with GUI objects as the database is updated.

Using the server aspect, the following can be created:

- Web Servers
- Game Servers
- App Servers

The Blockchain

The decentralized datacenter will allow RadJav applications to be executed for a fee given to the machine executing the application. Due to the complexity of the decentralized datacenter, it will release in three versions, V1, V2, and V3.

As of July 2017, Blockchain V1 (XRJV1) is currently running and accessible by coding with RadJav, using the XRJV1 GUI wallet, or the XRJV1 Electrum Wallet. Any type of RadJav application can access the blockchain and the decentralized datacenter. For more information about the different versions, please see the sections Blockchain V1 (XRJV1), Blockchain V2 (XRJV2) and Blockchain V3 (XRJV3) in the following pages.

Using the blockchain aspect the following can be created:

- Smart Contracts

- Dapps

Blockchain V1 (XRJV1)

This is the first version of the RadJav blockchain. It’s simply a modified version of the Bitcoin 0.14.2 source code. The primary use of XRJV1 is to send and receive coins over the RadJav network, in addition to having the ability to easily integrate with the Bitcoin network. It is a secure and immutable ledger. This also helps developers who have never dealt with blockchain technologies before, understand how they work and realize their potential. Its current specifications are:

- Number of coins distributed per block: 16
- Average Block time: 3 minutes
- Ticker Symbol: XRJ
- Mining Algorithm: SHA-256
- Maximum Coins: 3,000,000,000
    - This maximum coin count will remain the same for XRJV2 and XRJV3 as well.
- Segwit and CSV are activated.
- The following BIPs are activated as well: BIP34, BIP65, BIP66, BIP68, BIP112, BIP113, BIP141, BIP143, BIP147
- This fork will be abandoned upon the successful main net launch of XRJV2, and should be the only time the RadJav blockchain ever forks

Blockchain V2 (XRJV2)

XRJV2 is currently in the planning phase, we are open to community participation and feedback.

What is it?

XRJV2 would be able to execute public and private smart contracts submitted by developers. A smart contract would be able to execute code on a certain date/time, send and receive transactions, hold coins, and much more depending on how the smart contract is coded. Due to the non-deterministic behavior of Javascript, it will not be used for creating smart contracts. A separate scripting language will be created that ensures guaranteed deterministic behavior, and will be announced at a later date.

Smart Contracts Should be Agreements

We believe that smart contracts should be agreements without the need of a third party. Typically, when you look at an agreement, it is simply a bunch of if/then statements with definitions at the beginning of the agreement. Our smart contracts would be very basic, ideally so basic that even those who don't know much about programming can look at one of RadJav's smart contracts and feel comfortable with it. The goal of the scripting language would be to make it as simple as possible, and very restrictive from a programmer's perspective. There would be no recursion, very limited OOP, no nested loops, very restricted (if any) calls to other smart contracts, and the call stack would remain small. Even without all these features, it would still be Turing Complete. Code reviewing a RadJav smart contract should be relatively easy and quick.

Oh, and don't worry - Dapps or server apps created for the XRJV3 network wouldn't have any restrictions and developers would be free to create whatever they desire.

How is it better than other chains?

Current blockchains have scaling and data processing issues, usually in the form of slow transactions at around 7-15 transactions per second. RadJav intends to address this from the start. Our goal is to reach around 1,000 transactions per second, then go up from there.

Additionally, other blockchains are not actively trying to improve the performance of their network in terms of bandwidth, latency, storage capacity, and script processing performance. Our blockchain is incentivized to improve all these areas itself over time. This is achieved with the nodes that have the best network connections, and script processing performance earning the most coins.

Proof-of-Competition

This first-of-a-kind performance methodology is built with scale and efficiency in mind. So how do we try and achieve this goal? By creating a hyper competitive network based on a node's network performance and computational speed. At the same time, any nodes that are not performing well will be repositioned to process lower priority transactions, or placed on the sidelines, where they do not earn coins. All nodes are placed into teams where each team has their own blockchain, and are managed by referees who enforce the security of the blockchain. Teams are only created as existing teams become overloaded. For more information on RadJav's Proof-Of-Competition, please see our technical whitepaper.

Network Updates

All of the blockchain's code would be stored on every node. Whenever a new proposal to upgrade the network is suggested, all eligible nodes can vote on whether or not to implement the new proposal. If 80% of the eligible nodes has voted yes to the proposal, then the code can be written by Higher Edge Software, uploaded to the testnet where it goes through testing. After testing, another vote will be made on the proposal's code, if again 80% of the eligible nodes vote in favor of the new code, the entire network will store the newest code and execute it. This will ensure that the network will never have to fork. For more information about upgrading the network, please see the technical whitepaper.

Self-regulating

The network would have a set of rules in place that would be regulated by referees selected by the network. Referees would be human controlled nodes that would be selected by background screening companies chosen by the network. On average, every team would be managed by 1 referee for every 25 nodes. In the event a smart contract or dapp breaks a rule, a referee can place a lock on the smart contract, which will flag that contract for other referees to review. Should 60% of the referees reviewing the contract find the contract guilty of breaking the rule, the remaining coins to pay for the fees would be sent to a random charity selected by the network. Any funds stored in the smart contract would be returned to where they originated. For more information about referees and rules, please see the technical whitepaper.

Network Defense

For every mined block, some coins are automatically placed into "pools" reserved for defending the network. A legal defense pool would be available for when the network needs to defend itself from legal regulations anywhere in the world, or it could be used to hire lobbyists to fight for it. Or legal packages can be created for ICOs to use for the network. The possibilities are endless. Law firms can apply to be selected by the network, and hired as needed. The bug fix and security vulnerabilities pool would be reserved for others to find bugs on the platform. Anyone would be able to submit a report to the network, and the network would determine the correct amount of coins to pay according to their report and their solution provided. Lastly, there would be a DDOS protection pool. This pool would be reserved for network selected DDOS protection services for when the network comes under attack from DDOS attacks.

XRJV2 Release

When XRJV2 releases, the RadJav blockchain will fork, and Proof-of-Work will switch to Proof-of-Competition. The network would remain a decentralized peer-to-peer network, and would never need to fork again.

Network Fees

Fees would be collected by miners hosting the smart contracts and tokens every hour. The developers that submit their contracts or wish to create their tokens would have to submit their fees at the same time. To keep tokens usable on the blockchain, developers must pay fees per hour. The more the token is used, the higher the fee. For more information about the fees, please review our technical whitepaper.

Blockchain V3 (XRJV3)

What is it?

XRJV3 would be able to execute dapps and server-side applications submitted by developers. It would be an upgrade of XRJV2. Server applications created with RadJav could be encrypted and submitted to the decentralized datacenter where nodes on the XRJV3 network would execute them. One of the primary focus areas of this development will be to make the server-side code, data in memory, and stored data as private as possible, from the node itself and throughout the rest of the network. Developers submitting their applications to the network would have an address they would send their coins to where they would be deducted every hour the application is using the CPU, GPU, hard drive storage, and network bandwidth. The submitted application and any data that it stores would reside on the node's hard drive, separate from the blockchain

Where's the details?

The technical details and fees associated with Blockchain V3 will come in a later whitepaper, since we would like community feedback and guidance. All application data stored on the node's hard disk would be encrypted. As the RadJav application is executed, the data stored in RAM would be encrypted as well. Ideally, once XRJV3 is in place, if possible, we would like to remove all coin transfer fees.

Currently we are investigating the possible ways of executing a RadJav XRJV2/XRJV3 dapp privately. Some of the methods we are looking at include:

1. Homomorphic Encryption
    a. A developer encrypts their application, along with any other files, then sends it to be executed on the decentralized datacenter. Either the V8 Javascript engine will have to be modified to allow for the execution of homomorphic encrypted Javascript, or a virtual machine will have to be created that will execute simpler code using homomorphic encryption. This would ensure that all data being written to and read from storage, all data in memory, and all processing data would

remain completely private and uncrackable. The two biggest hurdles we would be facing would be the speed at which dapps could execute and allow the homomorphic encrypted Javascript to properly execute.

2. Trusted Master Key Nodes
    a. A developer encrypts their application using an application specific private key, along with any other files, then sends it to be executed on a random node on the decentralized datacenter. After the dapp has been sent, the developer's private key associated with the dapp is sent to a trusted master key node, most likely a referee. When it is time for the dapp to be executed on the random node, the trusted master key node sends the developers private key to the random node, so the node can then decrypt the dapp and execute it.
3. Secured Web Server/Light Database/File Storage
    a. We would develop a secure web server, light database, and file storage system that the nodes could host without being able to see what data is stored on the system. Only the developer and the developer's clients accessing the system would be able to see the data they are permitted to see. The web server would simply serve the HTML files directly to the client. In the event we go this route, the fee structure would change so that fees would be collected by the node for all data stored on the node, and the length of time that the data remains on the system. Fees would also be collected for how long the web server/light database/file storage has been operating for the developer. Additionally, in the event we go this route, we will continue to work on implementing a secure and private system that would execute RadJav server-side applications and XRJV2 smart contracts.

In an attempt to keep the size of the blockchain down to a minimum, only a few random nodes on a team would be executing a single RadJav application at a time. This small group of nodes would contain the application and would be keeping an eye on each other to make sure no hostile activities are going on, and that they are executing what they are supposed to be executing. This also helps for when one node goes down or times out, another can pick up and continue the application.

Network Fees

Fees would be collected by nodes hosting the RadJav applications. They would collect fees for the amount of time the application has been executing, their GPU usage, amount of storage used, and how much bandwidth has been used by the application. The smart contracts executing along with their RadJav application would also be collecting fees as well. The developer would have to continue to submit XRJ coins to an address to ensure their apps remain running on the XRJV3 network. The fee costs and how often developers would have to submit additional coins would be determined during the planning phase of XRJV3.

? Development Plan

The most important part to get right with RadJav is ensuring the platform is a secure development platform, especially with server development. We must ensure that all known vulnerabilities have been patched, and that we will not be inadvertently creating new ones.

Thankfully a lot of these problems have already been fixed for us. Since we're using existing open source libraries that have been around for nearly two decades, have been thoroughly tested and debugged, all we're doing is connecting the dots between the different libraries and databases and integrating them with RadJav. We're creating an easy-to-use platform using existing well-known tools that developers can use, so they don't have to hunt them down themselves and use multiple disparate systems to complete their projects.

We will be coding XRJV2 and XRJV3 in RadJav, which will give the community the most flexibility in deciding the direction the blockchain should be developed. Since all the code will be hosted across all the nodes themselves, it's the nodes that get to decide what upgrades get to be made. Higher Edge Software is still the developer of the blockchain, however this gives the power to the community as to what changes should or should not be made.

With the money we raise from the crowdsale, we plan to keep costs around $3,000,000 a year. Ideally we'd like to hire a team of 20. Additionally, we plan on spending $350,000 a year on marketing RadJav to developers, such as creating online tutorials, video tutorials, attending events, and creating course curriculums on how to develop using RadJav.

? Use Cases

Create Desktop, Mobile, & Web Apps

A developer could easily create a mobile social media app in RadJav that connects to a RadJav server, and uses XRJ for any in-store purchases that need to be made in the app. The UI could be updated from the server as other people using the app post their replies. With little additional code this would be possible. If the app turns out to be popular, the developer could setup a website, then use the same code to run as a web app online for even more exposure. The developer could also make Windows, Linux, and Mac app downloads as well. It would only take minutes to make the downloads once coded with RadJav. When the servers they are hosting the application on becomes overloaded, they can easily switch over to the XRJV3 network, and have the network scale as more users begin to use their app.

Esports

The esports market is growing at a highly rapid pace. A game developer could create a game that utilizes their own custom token on the XRJV2 network. The game isn't required to be coded in RadJav in order to utilize the token. The game developer could create a smart contract that can host online tournaments for the game, and send a portion of the token used in the tournament to the winners and developers.

Making App Updates

Today if a developer wanted to create a desktop app, mobile app, and a web app, they would have to use multiple programming languages and many different libraries. With RadJav, developers can create apps and deploy them on desktop machines, the web, and on mobile devices all on the same platform. Then whenever an update needs to be made, the developer can make the changes to their code, then simply redeploy their apps. Unlike today, where if a change has to be made the developer has to go through making lots of changes to each individual app they've made utilizing all the different programming languages and libraries.

Massive Multiplayer Games

Using the XRJV3 network, a developer could create a server app that could be distributed across many nodes, with each node handling certain areas of the large multiplayer world. As a player moves from one area of the world to another, the game could make a connection to the next part of the world creating a seamless transition. Additionally, when too many players are in the same area, the load could be shared with other nodes. With this design, a multiplayer game could be created that could sustain tens of thousands of players all in the same world.

Quick and effortless blockchain implementation

Currently, if a financial institution wanted to add a blockchain to their existing web application (for example, Bitcoin), they would have to take apart their server-side code and determine the best ways to include the new code into their application. Then the developers would have to choose the best Bitcoin library to use to integrate into their existing code, and hope it doesn't interfere with anything else. Then they must test their system multiple times in many ways, and hope they fixed all the issues they found.

With RadJav, on the front-end of their web application's code, they could simply include the RadJav HTML5 library, and execute the code to send the transaction or smart contract and be done. If they need to add another field into their database to help log the transaction or save the result of the smart contract, this can be done quickly on their end, saving lots of development time, money and potential down-time. Since RadJav is built modularly, a similar method could be used on desktop applications as well, including RadJav as a library.

Custom Family Joint Accounts

With XRJV2 a smart contract could be written where a family's joint account is used. Both parents have a multi-signature lock on it so only they can withdraw all the coins they need, whenever they need, whereas the children can only withdraw 1% of the coins on the account per week. Should the parents need to go out of town, and determine a child could use more coins between Sept 5th and the 10th, they can increase that percentage to 5% only for that time range.

Decentralized Online Services

Using XRJV3, an instant messaging (IM) service could be developed where the GUI is created and communicates to the server on the blockchain. As the IM platform grows, the developer could feed more XRJ coins into the service's XRJ address, then increase the maximum node count allowing it to scale as more users start to use the service.

Datacenter Offloading

A company that has a large overhead managing their datacenter would be able to write a RadJav application to help offload their server load onto the XRJV3 network. This would be ideal for companies that offer cloud storage solutions or content delivery networks (CDNs), as this would be the easiest and most secure way for them to lower their operational costs.

Always Online Digital Rights Management (DRM) Service

Using XRJV3, a DRM service could be created to help decrypt RadJav applications stored on a user's machine allowing the software to be executed and ensure that software is only running on the machines it's supposed to be running on.

RadJav Software

RadJavVM

The RadJav Virtual Machine (RadJavVM) is the software that executes ".xrj" applications which can be installed on a user's machine. These ".xrj" applications are RadJav applications that can be GUI, 3D, server, or blockchain applications. It also includes the blockchain nodes which can be executed from either the operating system's start menu or a terminal. This software simply parses the terminals command line arguments and executes xrj applications using RadJav's library "libRadJav."

libRadJav

The driving force behind RadJav; it executes GUI, 3D, server, and blockchain applications. This C++ library can be embedded into nearly any C++ project giving that project the ability to execute RadJav applications. Since RadJav is licensed under the MIT license, developers have the freedom to include RadJav into any project for any purpose they desire.

RadJav HTML5 Library

The HTML5 version executes GUI, 3D, and blockchain applications. This JavaScript library can be integrated into nearly any HTML5 application, giving developers the ability to have their applications connect to the blockchain or connect to a server application.

XRJV1 Node

A separate program that can be installed on a computer to help process and store transactions over the XRJV1 network, making the computer a node. XRJV1 nodes would not be able to execute any RadJav application code. These nodes help secure the RadJav XRJV1 network, and are included in libRadJav and RadJavVM.

Nodes will have to download the entire blockchain in order to send coins or help secure the network.

Road Map

The road map is subject to change at any time for any reason.

- 4th Quarter 2017
    - Release basic 3D engine for Windows and HTML5.
        - This will include the ability to place 3D models, cameras, spotlights, point lights, directional lighting, and place materials on 3D objects.
    - Release Linux and MacOSX versions of RadJavVM.
- 1st Quarter 2018
    - GUI
        - (HTML5) Move from Dojo to our own custom GUI DOM objects.
        - (Desktop and HTML5) Include many more GUI events to existing and new GUI objects. Create an event object to pass to nearly each event. Try to close the differences (and fix bugs) between desktop and HTML5 applications.
        - (Desktop and HTML5) Add tabs, sliders, progress bars, calendar picker, tree view, system tray support, and many more generic GUI objects.
        - (Desktop) Add custom theme support for desktop applications.
    - 3D (Desktop and HTML5)
        - Add particle systems, shadow maps, and some basic collision detection.

- Audio support
  - Server
    - (Desktop) Add WebSocket server support.
    - (Desktop and HTML5) Add WebSocket client support.
    - (Desktop) Add light database.
  - Blockchain
    - Blockchain V1 (XRJV1)
      - Release the HTML5 Blockchain V1 (XRJV1) wallet
    - Blockchain V2 (XRJV2)
      - Begin planning what the initial version of Blockchain V2 (XRJV2) will feature, and possibly begin coding it towards the end of September.
      - Coding will at minimum begin October 1st.
  - General
    - Create tutorials, examples, and videos on how to code with RadJav.
    - Start creating course and RadJav application teaching little kids and high school students how to code with RadJav.
- 2nd Quarter 2018
  - GUI (Desktop and HTML5)
    - Continue adding additional GUI objects and events, making bug fixes.
  - MGUI (iOS, Android)
    - Start porting RadJav to run on mobile devices, iOS and Android would be first.

        - All native textboxes, checkboxes, and other generic GUI objects will start being put in.
    - RGUI (Desktop, Mobile, and HTML5)
        - The responsive GUI that will work between all the different operating systems.
        - Begin planning how custom GUI objects will look, feel, and operate between all the different operating systems and HTML5.
    - 3D (Desktop and HTML5)
        - Add terrain support, and a basic physics engine.
    - Server (Desktop)
        - HTTP/HTTPS server support.
        - Raw TCP/UDP IPv4, IPv6 server support.
        - (Desktop and HTML5) Begin working on a reactive WebSocket server/client that will store and retrieve data between the database and update GUI objects on screen automatically.
    - Blockchain
        - XRJV2
            - Continue coding and testing.
        - Blockchain V3 (XRJV3)
            - Begin planning how XRJV3 will operate, how nodes/applications will be banned, and for what reasons.
- 3rd Quarter 2018
    - MGUI (iOS and Android)
        - Continue adding generic GUI objects for both operating systems.
    - RGUI (Windows, Android, and HTML5)

        - Begin coding the custom GUI objects for Windows, Android, and HTML5 first. iOS, Linux, and Mac support would come later. The first objects would be frames, text boxes, check boxes, radio buttons, etc.
    - Server (Desktop and HTML5)
        - Reactive WebSocket server/client should be completed.
    - Blockchain
        - XRJV2
            - Hopefully around the 3rd Quarter of 2018 we will begin prepping XRJV2 for testing on the testnet. We will test for several weeks to month or more to ensure all bugs have been fixed and all security vulnerabilities have been patched.
            - Private smart contracts most likely will not be available at this time. They will probably launch with XRJV3.
            - Depending on community feedback and how XRJV2 performs on the testnet, we will launch in the 3rd quarter of 2018. This may be pushed back, just a fair warning.

Contact Us

[Website](#) [Slack](#) [Facebook](#) [BitcoinTalk](#) [Twitter](#) [Email](#) [White Paper](#)